Safe Reinforcement Learning via Shielding under Partial Observability

Steven Carr[1], **Nils Jansen**[2], **Sebastian Junges**[2], **Ufuk Topcu**[1]
[1] The University of Texas at Austin
[2] Radboud University, Nijmegen, The Netherlands
stevencarr@utexas.edu, n.jansen@science.ru.nl, sjunges@cs.ru.nl, utopcu@utexas.edu

Abstract

Safe exploration is a common problem in reinforcement learning (RL) that aims to prevent agents from making disastrous decisions while exploring their environment. A family of approaches to this problem assume domain knowledge in the form of a (partial) model of this environment to decide upon the safety of an action. A so-called shield forces the RL agent to select only safe actions. However, for adoption in various applications, one must look beyond enforcing safety and also ensure the applicability of RL with good performance. We extend the applicability of shields via tight integration with state-of-the-art deep RL, and provide an extensive, empirical study in challenging, sparse-reward environments under partial observability. We show that a carefully integrated shield ensures safety and can improve the convergence rate and final performance of RL agents. We furthermore show that a shield can be used to bootstrap state-of-the-art RL agents: they remain safe after initial learning in a shielded setting, allowing us to disable a potentially too conservative shield eventually.

1 Introduction

Reinforcement learning (RL) (Sutton and Barto 1998) is a technique for decision-making in uncertain environments. An *RL agent* explores its environment by taking *actions* and perceiving feedback signals, usually *rewards* and *observations* on the system state. With success stories such as AlphaGo (Silver et al. 2016) RL nowadays reaches into areas such as robotics (Kober, Bagnell, and Peters 2013) or autonomous driving (Sallab et al. 2017).

A significant limitation of RL in safety-critical environments is the high cost of failure. An RL agent explores the effects of actions – often selected randomly, such as in state-of-the-art policy-gradient methods (Peters and Schaal 2006) – and will thus inevitably select actions that potentially cause harm to the agent or its environment. Thus, typical applications for RL are games (Mnih et al. 2013) or assume the ability to learn on high-fidelity simulations of realistic scenarios (Tao et al. 2019). The problem of *unsafe exploration* has triggered research on the *safety* of RL (Garcıa and Fernández 2015). Safe RL may refer to (1) changing ("*engineering*") the reward function (Laud and DeJong 2003) to encourage the agent to choose safe actions, (2) adding a second cost function ("*constraining*") (Moldovan and Abbeel 2012), or (3) blocking ("*shielding*") unsafe actions at runtime (Alshiekh et al. 2018). This paper falls into the last category.

Safe RL in partially observable environments suffers from uncertainty both in the agent's actions and perception. Such problems, typically modeled as partially observable Markov decision processes (POMDPs) (Kaelbling, Littman, and Cassandra 1998), require histories of observations to extract a sufficient understanding of the environment. Recent deep RL approaches for POMDPs, including those that employ recurrent neural networks (Hausknecht and Stone 2015; Wierstra et al. 2007), learn from these histories and can generate high-quality policies with sufficient data. However, these approaches do not guarantee safety during or after learning. While shielding for Markov decision processes (MDPs) is rather well-studied (Könighofer et al. 2017; Alshiekh et al. 2018; Fulton and Platzer 2018; Bouton et al. 2019), there is—to the best of our knowledge—no approach that integrates shielding with deep RL.

We contribute a thorough study, implementation, and experimental evaluation on integrating state-of-the-art RL for POMDPs with shields. We demonstrate effects and insights using several typical examples and provide detailed information on RL performance as well as videos showing the exploration and training process. In particular, we integrate various RL algorithms from Tensorflow (Guadarrama et al. 2018) with a shield that guarantees safety regarding so-called reach-avoid specifications, a special case of temporal logic constraints (Pnueli 1977). The computation of the shields is based on satisfiability solving (Junges, Jansen, and Seshia 2021) and requires only a *partial model of the environment*. Specifically, we need to know all potential transitions in the POMDP, while probabilities and rewards may remain unspecified (Raskin et al. 2007).

Approach. Fig. 1 shows the outline of the safe RL setting. The gray box shows a typical RL procedure. The environment in the partially observable setting is described by a POMDP model that may not be fully known. A *shield* in this setting (implicitly) requires access to a form of *state estimation* to account for a *safety specification*. This estimator uses the *partial model* of the environment to track in which states the model may be, based on the observed history. We see the shield and the state estimator as two knowledge interfaces

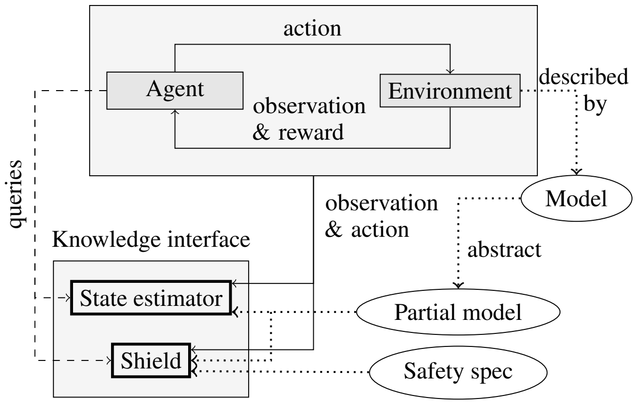


Figure 1: Safe RL with two knowledge interfaces for the agent: A state estimator and a shield, based on a partial model of the environment.

for the agent that may be used in conjunction or separately. A shield may be too conservative and overly protective, and therefore restrict the performance of RL in general. Alternatively, we investigate if (only) access to a state estimator may serve as a lightweight alternative to a shield. We show that, while the RL agent may indeed benefit from this additional information, the shield provides more safety and faster convergence than relying on just the state estimator. After learning, we may gently phase out a shield and still preserve the better performance of the shielded RL agent. Then, even an overly protective shield helps to bootstrap RL.

Summarized, our study demonstrates the following effects of shielding in a partially observable setting.

- *Safety during learning:* Exploration is only safe when the RL agent is provided with a shield. Without the shield, the agent makes unsafe choices even if it has access to the state estimation. Even an unshielded *trained agent* still behaves unsafe sometimes.
- *RL convergence rate:* A shield not only ensures safety, but may also significantly improve the convergence rate of modern RL agents by avoiding spending time to learn unsafe actions. Other knowledge interfaces like state estimators do help to a lesser extent.
- *Bootstrapping:* Due to the improved convergence rate, shields are a way to bootstrap RL algorithms, even if they are overly restrictive. RL agents can learn to mimic the shield by slowly disabling the shield.

Further related work. Several approaches to safe RL in combination with formal verification exist (Hasanbeig, Abate, and Kroening 2020; Könighofer et al. 2017; Alshiekh et al. 2018; Jansen et al. 2020; Fulton and Platzer 2018; Bouton et al. 2019). These approaches either rely on shielding, or guide the RL agent to satisfy temporal logic constraints. However, none of these approaches take our key problem of partial observability into account. Recent approaches to find safe policies for POMDPs with partial model knowledge either do not consider reinforcement learning (Cubuktepe et al. 2021) or require the agent to take catastrophic actions before learning from them (Shperberg, Liu, and Stone 2022).

2 Problem Statement

We introduce POMDPs as the standard model for sequential decision-making under partial observability. We distinguish the learning goal of an agent by expected rewards, and the safety constraints via reach-avoid safety specifications.

2.1 POMDPs

A (discrete) *partially observable Markov decision process (POMDP)* is a tuple $\mathcal{M} = (S, I, Act, O, Z, \mathcal{P}, \mathcal{R})$ where S is a finite state space. I is the initial distribution that gives the probability $I(s)$ that the agent starts in state $s \in S$, and Act is a finite space of actions. Z is a finite observation space and $O(z|s)$ is the probability of observing z when the environment is in state s. The transition model $\mathcal{P}(s'|s,a)$ represents the conditional probability of moving to a state $s' \in S$ after executing $a \in A$ in $s \in S$. When executing $a \in Act$ in $s \in S$, the agent receives a scalar reward $\mathcal{R}(s,a)$. As not every action may be available in every state, we define the set of available actions in s as $Act(s)$. We remark that POMDPs may have dead-ends from which an agent cannot obtain positive rewards (Kolobov, Mausam, and Weld 2012).

As the current state of a POMDP is not observable, agents may infer the probability of being in a certain state based on the history so far. This probability distribution is the *belief* $\mathfrak{b}\colon (Z \times Act)^* \times Z \to Distr(S)$. A *policy* $\pi\colon \mathfrak{b} \to Distr(Act)$ for the agent decides upon a distribution over actions based on the current belief. A (fully observable) *belief MDP* with the (infinitely many) beliefs of the POMDP as states captures the belief dynamics.

2.2 Learning Goal and Safety Constraints

The standard *learning goal* for POMDPs is to find a policy π that maximizes the expected discounted reward, that is, $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t\right]$, where γ^t with $0 \leq \gamma^t \leq 1$ is the discount factor and $\mathcal{R}_t$ is the reward at time t. Due to the infinite number of beliefs, computing such an optimal policy is in general undecidable, even if the entire model is known (Madani, Hanks, and Condon 1999).

An agent in safety-critical settings must (additionally) adhere to *safety constraints*. We capture these constraints using *(qualitative) reach-avoid* specifications, a subclass of indefinite horizon properties (Puterman 1994). Such specifications necessitate to *always* avoid certain bad states from *AVOID* $\subseteq S$ and reach states from *REACH* $\subseteq S$ *almost-surely*, i.e., with probability one (for arbitrary long horizons). We denote these constraints by $\varphi = \langle \textit{REACH}, \textit{AVOID} \rangle$. The *avoid* specification $\varphi = \langle \textit{AVOID} \rangle$ necessitates only to avoid bad states. Formally, $Pr^{\pi}_{\mathfrak{b}}(S)$ denotes the probability to reach a set of states $S' \subseteq S$ from the belief $\mathfrak{b}$ using the policy π.

Definition 1 (Winning). *A policy π is* winning *for specification φ from belief $\mathfrak{b}$ in POMDP $\mathcal{M}$ iff $Pr^{\pi}_{\mathfrak{b}}(\textit{AVOID}) = 0$ and $Pr^{\pi}_{\mathfrak{b}}(\textit{REACH}) = 1$. Belief $\mathfrak{b}$ is* winning *for φ in $\mathcal{M}$ if there exists a winning policy from $\mathfrak{b}$.*

The relation $\mathcal{M}(\pi) \models \varphi$ denotes that the policy π is winning according to the initial belief I. Computing such a *winning* policy is, in general, decidable and EXPTIME-complete but practically feasible methods exist (Chatterjee, Chmelik, and Davies 2016; Junges, Jansen, and Seshia 2021). Finally, for multiple beliefs, a *winning regions* (aka safe or controllable regions) is a set of winning beliefs, that is, from each belief within a winning region, there exists a winning policy.

We formulate the joint learning and safety problem we consider in our safe reinforcement learning setting.

Problem 1 (Safe Learning). *Given a POMDP $\mathcal{M}$, a safety constraint φ, and let $\pi_1, \ldots, \pi_n$ be the (training) sequence of policies employed by an RL agent. The problem is to ensure that for all policies π_i it holds that $\mathcal{M}(\pi_i) \models \varphi$ with $1 \leq i \leq n$ and the final policy π_n maximizes $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t\right]$.*

Note that the condition to satisfy φ may induce a sub-optimal reward as the agent has to strictly adhere to the safety constraint while collecting rewards.

3 State Estimators and Shields

In this section, we present the two knowledge interfaces for RL agents in the shielding approach for POMDPs, refer to Fig. 1. In particular, we discuss belief supports as concrete state estimators for the agent, and introduce the notion of a shield (for POMDPs). We discuss which guarantees we can provide for shields that are computed on partial models.

3.1 Belief Supports as State Estimators

If the transition (and observation) probabilities in the POMDP were known, the agent could incrementally compute a Bayesian update of the belief and use that to estimate its current state. However, this is a strong assumption. Instead, we rely on the so-called belief support. A state s is in the *belief support* B for a belief $\mathfrak{b}$, if $s \in \mathrm{supp}(\mathfrak{b})$. Thus, the belief-support $B \in \mathcal{B}$, with $\mathcal{B}$ the set of all belief supports, is a set of states. The belief-support can be updated using only the graph of the POMDP (without probability knowledge) by a simplified belief update. In particular, we can compute the unique belief support $(B' \mid B, a, z)$ that can be reached from B with action a and observation z. We exploit the following result from (Junges, Jansen, and Seshia 2021).

Theorem 1. *If a belief $\mathfrak{b}$ is winning for a reach avoid specification φ, any belief $\mathfrak{b}'$ with $\mathrm{supp}(\mathfrak{b}') = \mathrm{supp}(\mathfrak{b})$ is winning.*

Intuitively, all beliefs that share a belief-support are winning, therefore, we can directly call a belief-support winning. We now define the first knowledge interface for the RL agent.

Knowledge interface 1: the state estimator. A *belief-support state estimator* $\sigma\colon (Z \times Act)^* \times Z \to \mathcal{B}$ takes as input the observation-action history and returns the current belief support to the RL agent. This estimator can be implemented by repeatedly updating the belief-support independently of the probabilities in a POMDP. We provide the agent with state estimation as additional observation signal.

3.2 Shields

The purpose of a shield is to prevent the agent from taking actions that would violate a (reach-avoid) specification. A shield allows only actions that enable the agent to stay in a winning region. That means, when taking an action from some winning belief support, any next belief support reached by taking this action must belong to a winning region.

We first define policies on the belief support of the form $\pi\colon \mathcal{B} \to Act$. Recall that $\mathcal{B}$ denotes the set of all belief supports. This (*deterministic*) policy chooses one unique action for each belief support $B \in \mathcal{B}$. For shields, we use a more liberal notion of *permissive* policies that select sets of actions (Dräger et al. 2015; Junges et al. 2016). Given a POMDP $\mathcal{M}$, a permissive policy is given as $\nu\colon \mathcal{B} \to 2^{Act}$. Any action in $\nu(B)$ is called *allowed*. Intuitively, one can think of a permissive policy as defining a set of policies: In particular, a policy π is *admissible* for ν if for all belief supports $B \in \mathcal{B}$ it holds that $\pi(B) \in \nu(B)$[1].

Shields can be defined as permissive policies for staying in a winning region with respect to reach-avoid specifications.

Definition 2 (Shield). *For a specification φ, a permissive policy ν is a φ-shield, if for any B winning for φ, $a \in \nu(B)$, and $z \in Z$, it holds that $(B' \mid B, a, z)$ implies B' is winning.*

Shields provide guarantees such that any policy that agrees with the shield is winning (Junges, Jansen, and Seshia 2021). We first state the formal correctness for avoid specifications.

Theorem 2. *Let φ be an avoid-specification. For any φ-shield for $\mathcal{M}$ all admissible policies are winning.*

Shields for avoid-specifications may block all actions and create deadlocks. Instead, we employ shields for reach-avoid specifications that prevent the agent from visiting any dead-ends. A shield itself cannot force an agent to visit reach states. However, under mild assumptions, we can ensure that the agent eventually visits the reach states: A policy is *fair* if in any state which is visited infinitely often, it also takes every allowed action infinitely often (Baier and Katoen 2008). For example, a policy that takes every allowed action with positive probability is fair.

Theorem 3. *Let φ be a reach-avoid-specification. For a φ-shield for $\mathcal{M}$, all fair and admissible policies are winning.*

Knowledge interface 2: the shield. We use shields as computed via (Junges, Jansen, and Seshia 2021) that ensure reach-avoid specifications as outlined above. Essentially, an agent may use a state estimator $\sigma\colon (Z \times Act)^* \times Z \to \mathcal{B}$ in conjunction with a shield $\nu\colon \mathcal{B} \to 2^{Act}$ to compute allowed actions. We restrict the available actions for the RL agents to these allowed actions.

3.3 Shields on Partial Models

A shield for a POMDP relies only on the belief-support. Therefore, it is also a shield for all POMDPs with the same underlying graph-structure. We formalize this statement.

Partial models. We assume the agent has only access to a *graph-preserving approximation* $\mathcal{M}'$ of a POMDP $\mathcal{M}$ that differs only in the transition models $\mathcal{P}'$ and $\mathcal{P}$, and potentially in the reward functions. It holds that $\mathcal{P}(s'|s,a) > 0 \iff \mathcal{P}'(s'|s,a) > 0$ for all states $s, s' \in S$ and actions $a \in Act$. Similarly, $\mathcal{M}'$ *overapproximates the transition model of* $\mathcal{M}$,

[1]Admissibility can be defined for more general classes of policies, see (Junges, Jansen, and Seshia 2021).

if it holds for all states $s, s' \in S$ and actions $a \in Act$ that $\mathcal{P}(s'|s,a) > 0 \implies \mathcal{P}'(s'|s,a) > 0$. The original POMDP has no transitions that are not present in the partial model.

Guarantees. We state the following guarantees provided by a shield, in relation to the two approximation types.

Theorem 4 (Reach-Avoid Shield)**.** *Let $\mathcal{M}'$ be a graph-preserving approximation of $\mathcal{M}$ and φ a reach-avoid specification, then a φ-shield for $\mathcal{M}'$ is a φ-shield for $\mathcal{M}$.*

This theorem follows directly from Theorem 1. Knowing the exact set of transitions with (arbitrary) positive probability for a POMDP is sufficient to compute a φ-shield.

For avoid specifications, we can further relax the assumptions. It suffices to require that each transition in the partial model exists (with positive probability) in the (true) POMDP.

Theorem 5 (Avoid Shield)**.** *Let $\mathcal{M}'$ overapproximate the transition model of $\mathcal{M}$, and let $\varphi' = \langle AVOID \rangle$ be an avoid specification, then a φ'-shield for the partial model $\mathcal{M}'$ is a φ'-shield for the POMDP $\mathcal{M}$.*

If the partial model is further relaxed, it is generally impossible to construct a shield with the same hard guarantees.

4 RL with Partial Model Knowledge

Recall the general setup in Fig. 1: While the environment is described as a POMDP, the agent has only access to a partial model via the knowledge interfaces as explained in the previous section. This section discusses the potential benefits of using these interfaces.

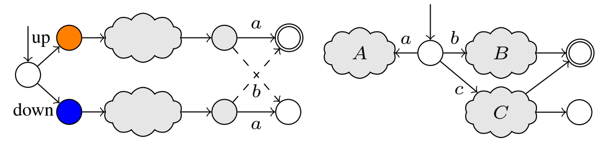
4.1 Safety

Safety during learning. Shielded RL agents are guaranteed to be safe during learning provided that the partial model is adequate as formalized by Theorem 4. Furthermore, assuming the RL agent is fair as defined above, it is guaranteed that they will eventually reach the *REACH* states. This guarantee does not come with an upper bound on the number of steps[2]. In contrast, an unshielded agent takes actions first to learn that it may lead to an *AVOID* state. State estimators are thus not sufficient to ensure safety, as they only reason about history and not about the future.

Safety after learning. In general, safety objectives encoded as reward and performance objectives (also encoded as reward) may allow for non-trivial trade-offs, which harm the ability to learn to adhere to safety objectives. Shielded RL agents do not face this tradeoff as they must adhere to the explicit safety constraints.

State estimators and safety. While state estimators cannot guarantee safety, they may improve safety. In particular, consider an action (such as switching on a light) which is useful and safe in most but not all situations (e. g., a gas leak). A state estimator provides the additional observation signals that allow the RL agent to efficiently distinguish these states, thereby indirectly improving safety, even during learning.

[2]Shields can also be computed for finite-horizon or cost-bounded reach-avoid properties, which come with a guarantee on finite steps.



(a) Example for estimators (b) Example for shields

Figure 2: Figures for illustrating benefits of knowledge interfaces in terms of convergence rate, see Section 4.2.

4.2 RL Convergence Rate

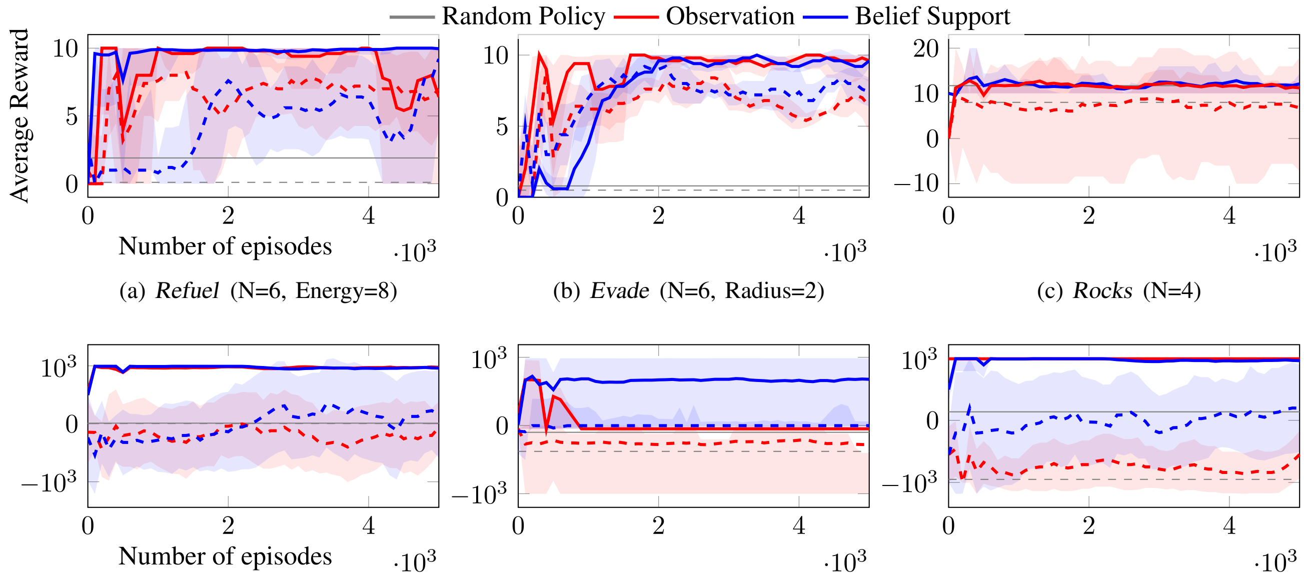
Beyond providing safety guarantees, learning in partially observable settings remains a challenge, especially when rewards are sparse. The availability of a partial model provides potential to accelerate the learning process.

Using state estimates. A state estimator enriches the observation with a signal that compresses the history. Consider the POMDP sketch in Fig. 2(a), abstracting a setting where the agent early on makes an observation (*orange*, top) or (*blue*, bottom), must learn to remember this observation until the end, where it has to take either action a (solid) when it saw *orange* before, or action b (dashed) when it saw *blue* before. State estimation provides a belief support that clarifies whether we are in the bottom or top part of the model, and thus trivializes the learning.

Using shields. A shield may provide incentives to (not) explore parts of the state space. Consider an environment as sketched out in Fig. 2(b). We have partitioned the state space into three disjoint parts. In region A, there are no bad states (with a high negative reward) but neither are there any reach states, thus, region A is a dead-end. In region B, all states will eventually reach, and in region C, there is a (small) probability that we eventually reach an avoid state. An agent has to learn that it should always enter region B from the initial state. However, if it (uniformly) randomly chooses actions (as an RL agent may do initially) it will only explore region B in one third of the episodes. If the high negative reward is not encountered early, it will take quite some time to skew the distribution towards entering region B. Even worse, in cases where the back-propagation of the sparse reward is slow, region A will remain to appear attractive and region C may appear more attractive whenever back-propagation is faster. The latter happens if the paths towards positive reward in region C are shorter than in region B.

4.3 Bootstrapping: Learning From the Shield

Finally, it is interesting to consider the possibility of disabling the additional interfaces after an initial training phase. For example, this allows us to bootstrap an agent with the shield and then relax the restrictions it imposes. Such a setting is relevant whenever the shield is overly conservative – e.g., entering some avoid-states is unfortunate but not safety-critical. It may also simplify the (formal) analysis of the RL agent, e.g., via neural network verification (Katz et al. 2017; Carr, Jansen, and Topcu 2021), as there is no further need to integrate the shield or state estimator in these analyses. We



(a) *Refuel* (N=6, Energy=8) (b) *Evade* (N=6, Radius=2) (c) *Rocks* (N=4)

(d) *Intercept* (N=7, Radius=1) (e) *Avoid* (N=6, Radius=3) (f) *Obstacle* (N=6)

Figure 3: REINFORCE with (solid) and without (dashed) a shield. The red lines show when the RL agent is trained using only observations from the environment, and the blue lines indicate when the RL agent is trained using the state estimator. The gray lines are the rewards, averaged over multiple evaluations, obtained via a policy that randomly selects from available actions.

investigate two ways to disable these interfaces and to evaluate agent performance after this intervention: either a *smooth transition* or *sudden deactivation*.

When switching off shields *suddenly*, the agent will be overly reliant on the effect of the shield. While it remembers some good decisions, it must learn to avoid some unsafe actions. We want to encourage the agent to learn to not rely on the shield. To support this idea, we propose a *smooth* transition: When switching off the shield, we do so gradually, applying the shield with some probability p, which will allow negative rewards whenever an action not allowed by the shield is taken. We decay the probability that the shield is applied over time to gently fade out its the effect. [3]

5 Experiments

We applied shielded RL in six challenging domains with partial observability. We compare multiple state-of-the-art deep RL agents. The experiments focus on the *safety*, *convergence rate*, and the *ability to learn* from additional interfaces as outlined in the three subsections of Section 4. We include the full set of results, and plots for all learning methods and domains in the technical appendix[4].

Setup. We use *Storm* (Hensel et al. 2022) as framework to interface the model, the shield, and the state estimator. All shields are computed within few minutes, details are given in the Appendix A.1. We developed bindings to Tensorflow's *TF-Agents* package (Guadarrama et al. 2018) and use its masking function to implement the precomputed shield. All experiments were performed on 8x3.2GHz Intel Xeon Platinum 8000 series processor with 32GB of RAM.

We employ a set of grid-based scenarios from (Junges, Jansen, and Seshia 2021). *Refuel* and *Obstacle* involve guiding a noisy agent to a goal location while avoiding hazardous situations. *Avoid* and *Evade* aim to guide an agent to a goal while avoiding collisions with one or more moving robots. *Intercept* attempts to prevent an adversary escaping by catching it in time. Finally, *Rocks* is a variant of RockSample (Smith and Simmons 2004) where the agent collects valuable rocks and delivers them. Detailed descriptions of the environments are given in Appendix A.1.

We use five deep RL methods: DQN (Mnih et al. 2015), DDQN (van Hasselt, Guez, and Silver 2016), PPO (Schulman et al. 2017), discrete SAC (Christodoulou 2019) and REINFORCE (Williams 1992). Unless otherwise specified, we limited episodes to a maximum of 100 steps and calculated the average reward across 10 evaluation episodes. Due to the sparse reward nature of the domains and for the sake of readability, we performed smoothing for all figures across a five-interval window. In episodal RL algorithms, such as REINFORCE, we trained on 5000 episodes with an evaluation interval every 100 episodes, and in the step-based RL algorithms, such as DQN, DDQN, PPO and discrete SAC, we trained on 10^5 steps with an evaluation interval every 1000 steps. Additionally, in the discrete SAC, we use long short-term memory (LSTM) as comparison to recent LSTM-based deep RL methods on POMDPs (Wierstra et al. 2007;

[3] When switching off state estimators, the learned agent is no longer executable as it lacks necessary information. Solutions, e.g., defaulting to a fixed observation, are not part of this work.

[4] Source code may be located at https://github.com/stevencarrau/safe_RL_POMDPs

Learning Setting	No. violations	
	During	*After*
No shield	3153	1023
Shield	0	0
Sudden switch-off	1867	502
Smooth switch-off	27	5

Table 1: The number of violations per episode for REINFORCE learning agent *during* and *after* learning across 5000 episodes, averaged across the six domains. An RL agent can have, at most, one violation per episode.

Hausknecht and Stone 2015). Details on the hyperparameters and the selection method are given in Appendix A.4.

5.1 Main Results

We make some key observations for REINFORCE. Results in Sec. 5.2 and Appendix B clarify that, in general, these observations hold for other RL algorithms.

The *no shield* and *shield* rows in Tab. 1 demonstrate that in line with the formal guarantees, **(only) the shielded agents never violate the safety specification**.

In Fig. 3, we demonstrate the performance of REINFORCE. In these and subsequent plots, the dashed lines indicate RL agents learning without the shield, while solid lines indicate that the agent is shielded. **Shielding accelerates convergence.** In Fig. 3(e) & 3(f) we observe that the addition of **the state estimator** (blue) **improves the convergence rate** over simply having the agent attempt to learn from observations (red). As the presented domains are partially observable with sparse reward, they are challenging settings for RL. Consequently, we see that REINFORCE does not always converge. We discuss details in Sec. 5.2.

In Fig. 4, we show how an RL agent performs when it initially learns using a shield and then that shield is either completely deactivated after 1000 episodes (green) or is switched-off with a smooth transition (orange). For the latter, we apply the shield with probability p and reduce p from 1 to 0 by learning rate α. The shielded RL agent generates higher-quality episodes and subsequently, **when the shield is removed, the agent still maintains higher quality episodes** since it has previously experienced the sparse positive reward. The effect is even more pronounced as the shield is smoothly removed, where the performance mirrors the shielded condition. We also refer to Tab. 1 for aggregates on safety violations when switching off shields. The **smooth deactivation cannot prevent safety violations completely, but shows a considerable improvement over the unshielded version**.

5.2 Detailed Results

In the sequel, we highlight important elements of the challenges presented in sparse domains, the shield's improved performance, and how the belief support and its representation impact learning.

Domains are sparse and thus challenging. As discussed, Fig. 3 & Fig. 6 indicate that in the sparse-reward domains and

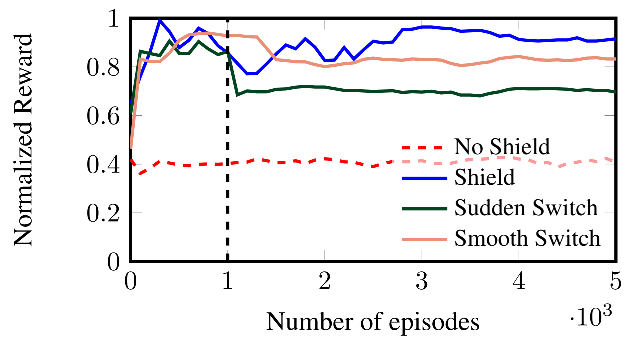


Figure 4: Normalized reward across all domains for RL agent (with the state estimator)[5] that learns for the first 1000 episodes with the shield active. After 1000 episodes the shield is either switched off completely (green) or is slowly turned off with increasing probability (yellow).

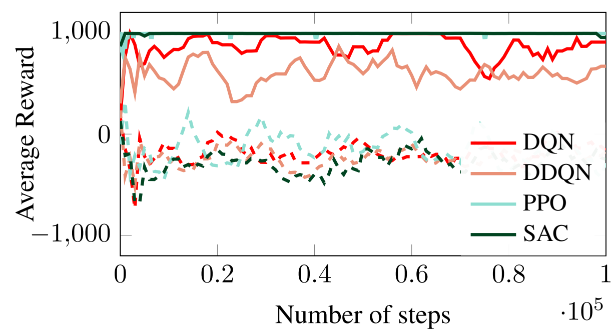


Figure 5: RL agents[5] employing different learning methods for *Intercept*.

under partial observation, without using additional knowledge from the given partial model, the deep RL algorithms struggle. In particular, reaching target states with a random policy is very unlikely, e.g., in *Evade* (Fig. 3(b)), a random policy without a shield reaches the target approximately 1% of the time. Likewise, when the agent attempts to learn a policy for *Avoid*, it converges to a locally optimal but globally sub-optimal policy, with an average reward of -100 (global optimum of $+991$). This policy that remains in the left corner stays outside of the adversary's reachable space, but will not move towards the goal at all. Similarly, the unshielded random policy often reaches a highly negative reward: e.g., 95% of the time in *Obstacle* (Fig. 3(f)). In POMDP settings converging to a local optimum is a challenge for many RL agents: In Fig. 5, we illustrate the problematic performance on the *Intercept* domain for a variety of unshielded RL agents.

Ablation study: full observability and reward shaping. In Fig. 6, we investigate the RL agent's[5] performance in more detail. In particular, when artificially making the domain fully observable, REINFORCE learns the optimal policy quickly for all domains (even in the unshielded condition), which

[5]Each RL agent used the belief support (via the state estimator) for the policy input representation.

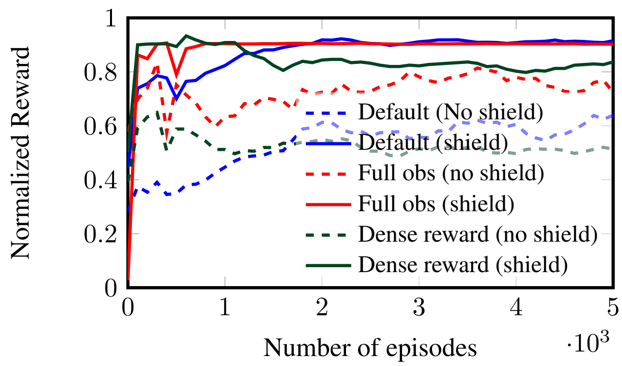


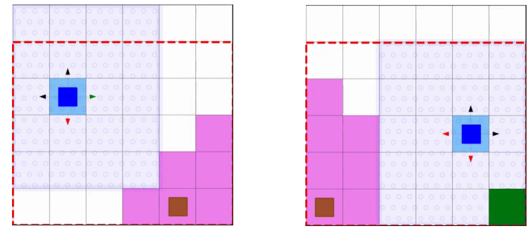
Figure 6: Empirical study comparing the different inputs for performing REINFORCE on a reward normalized across all domains. Dashed lines imply the RL agent[5] is not shielded.

demonstrates how difficult it is to learn in POMDPs. To overcome the sparse reward we can use reward shaping to guide the learner (Kim et al. 2015; Hlynsson and Wiskott 2021). While reward shaping[6] may help, it leads to nontrivial side-effects, especially for POMDPs. We observe that a dense reward function helps the RL agent to converge faster than in the default domain (see first 1000 episodes in Fig. 6). However, the dense reward may harm the final performance, as exemplified by the performance for the dense reward compared to the default. In particular, it seems to encourage risky exploration in unshielded settings. The shield provides the best improvement of performance when viewed in isolation (each solid line gives consistently higher returns than its dashed equivalent).

Shields improve convergence rate. Shielded agents prevent encountering avoid states in all episodes, and other episodes are thus more frequent. Consequently, a shielded RL agent has a higher probability of obtaining a positive reward even if the reward is sparse. For instance, in *Obstacle*, an unshielded random policy averages approximately 12 steps before crashing. In contrast, the shielded policy, which cannot crash, averages approximately 47 steps before reaching the goal. For RL agents that rely on episodic training, such as REINFORCE, the shield greatly improves the agent's convergence rate, see Fig. 3(f). These efficiencies hold even for the step-based RL agents, such as those presented in Fig. 5. However, the DQN and DDQN struggle to converge to the optimal policy. Such behavior could result from insufficient data to properly process the state estimates from the shield.

State estimators improve convergence (less). The challenge of RL agents struggling with high uncertainty, as sketched in the previous paragraphs, can also occur with a shield. Again, in the *Obstacle* domain, REINFORCE without the state estimation (red in Fig. 3) needs to learn both how to map the observation to the possible states, and then also how this would map into a value function, which it does only after spending roughly 2000 episodes. In comparison, with access to the belief support (blue in Fig. 3), the agent quickly

[6]Details for specific shaping in Fig. 6 are in Appendix A.3.



(a) *Evade* at $t = 9$ (b) *Evade* at $t = 25$

Figure 7: Incremental states of *Evade* where the agent (dark blue square) has a belief set of states (shaded in pink). The goal (green) is static. At $t = 9$, the shield prevents $\{\text{south}\}$ and the agent takes $\{\text{east}\}$ and at $t = 25$, the shield prevents $\{\text{south, east}\}$ and the agent takes $\{\text{scan}\}$.

learns to estimate the value function. Thus, even shielded, access to a state estimator can help. Vice versa, a shield does significantly improve agents, even with a state estimator.

Limitation: Shields alone do not enforce reaching targets quickly. As a drawback, shielding does not directly steer the agent towards a positive reward. In environments like *Evade*, where the reward is particularly sparse, a random policy with a shield has only an 8% chance of reaching the goal, see Fig. 3(b). In particular, it takes many episodes before even collecting any positive reward. Shielded agents do thus not alleviate the fact that episodes may need to be long.

Limitation: Shields may have little effect on performance. For the domain *Evade* in Fig. 3(b), the RL agent is only marginally improved by the addition of the shield. In this domain, the shield is much less restrictive, often not restricting the agent's choice at all. Consider Fig. 7, where the agent can easily either take an action that is just as beneficial as the one that was restricted as in Fig. 7(a), or reduce the size of the belief support by taking a scan as in Fig. 7(b). Further, in *Evade*, the shield is restricting the agent from taking actions that result in collisions with a very low probability. When the unshielded agent takes these potentially unsafe actions, it rarely suffers any negative outcome, leading to similar values of average reward. In principle, this may even degrade the performance of the shield. A similar problem occurs if the episodes are too short to ensure reaching the target, as detailed in Appendix B.1.

6 Conclusions

We presented a thorough study and an efficient open-source integration of model-based shielding and data-driven RL towards safe learning in partially observable settings. The shield ensures that the RL agent never visits dangerous avoid-states or dead-ends. Additionally, the use of shields helps to accelerate state-of-the-art RL. For future work, we will investigate the use of model-based distance measures to target states (Jansen et al. 2020) or contingency plans (Pryor and Collins 1996; Bertoli, Cimatti, and Pistore 2006) as an additional interface to the agent.

Acknowledgments

This research has been partially supported by ONR N00014-22-1-2254, ARO W911NF-20-1-0140, ONR N00014-20-1-2115. Additionally this work was partially supported by the ERC Starting Grant 101077178 (DEUCE).

References

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe Reinforcement Learning via Shielding. In *AAAI*. AAAI Press.

Baier, C.; and Katoen, J.-P. 2008. *Principles of Model Checking*. MIT Press.

Bertoli, P.; Cimatti, A.; and Pistore, M. 2006. Towards Strong Cyclic Planning under Partial Observability. In *ICAPS*, 354–357. AAAI.

Bouton, M.; Karlsson, J.; Nakhaei, A.; Fujimura, K.; Kochenderfer, M. J.; and Tumova, J. 2019. Reinforcement Learning with Probabilistic Guarantees for Autonomous Driving. *CoRR*, abs/1904.07189.

Carr, S.; Jansen, N.; and Topcu, U. 2021. Task-aware verifiable RNN-based policies for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 72: 819–847.

Chatterjee, K.; Chmelik, M.; and Davies, J. 2016. A Symbolic SAT-Based Algorithm for Almost-Sure Reachability with Small Strategies in POMDPs. In *AAAI*, 3225–3232. AAAI Press.

Christodoulou, P. 2019. Soft Actor-Critic for Discrete Action Settings. *CoRR*, abs/1910.07207.

Cubuktepe, M.; Jansen, N.; Junges, S.; Marandi, A.; Suilen, M.; and Topcu, U. 2021. Robust Finite-State Controllers for Uncertain POMDPs. In *AAAI*, 11792–11800. AAAI Press.

Dräger, K.; Forejt, V.; Kwiatkowska, M. Z.; Parker, D.; and Ujma, M. 2015. Permissive Controller Synthesis for Probabilistic Systems. *LMCS*, 11(2).

Fulton, N.; and Platzer, A. 2018. Safe Reinforcement Learning via Formal Methods: Toward Safe Control Through Proof and Learning. In *AAAI*. AAAI Press.

Garcıa, J.; and Fernández, F. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1): 1437–1480.

Guadarrama, S.; Korattikara, A.; Ramirez, O.; Castro, P.; Holly, E.; Fishman, S.; Wang, K.; Gonina, E.; Wu, N.; Kokiopoulou, E.; Sbaiz, L.; Smith, J.; Bartók, G.; Berent, J.; Harris, C.; Vanhoucke, V.; and Brevdo, E. 2018. TF-Agents: A library for Reinforcement Learning in TensorFlow. https://github.com/tensorflow/agents. Accessed: 2022-10-22.

Hasanbeig, M.; Abate, A.; and Kroening, D. 2020. Cautious Reinforcement Learning with Logical Constraints. In *AAMAS*, 483–491. International Foundation for Autonomous Agents and Multiagent Systems.

Hausknecht, M. J.; and Stone, P. 2015. Deep Recurrent Q-Learning for Partially Observable MDPs. In *AAAI*, 29–37. AAAI Press.

Hensel, C.; Junges, S.; Katoen, J.; Quatmann, T.; and Volk, M. 2022. The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.*, 24(4): 589–610.

Hlynsson, H. D.; and Wiskott, L. 2021. Reward Prediction for Representation Learning and Reward Shaping. In *IJCCI*, 267–276. SCITEPRESS.

Jansen, N.; Könighofer, B.; Junges, S.; Serban, A.; and Bloem, R. 2020. Safe Reinforcement Learning Using Probabilistic Shields (Invited Paper). In *CONCUR*, volume 171 of *LIPIcs*, 3:1–3:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Junges, S.; Jansen, N.; Dehnert, C.; Topcu, U.; and Katoen, J. 2016. Safety-Constrained Reinforcement Learning for MDPs. In *TACAS*.

Junges, S.; Jansen, N.; and Seshia, S. A. 2021. Enforcing Almost-Sure Reachability in POMDPs. In *CAV*, volume 12760 of *LNCS*, 602–625. Springer.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1): 99–134.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *CAV*, 97–117. Springer.

Kim, H.; Lim, W.; Lee, K.; Noh, Y.; and Kim, K. 2015. Reward Shaping for Model-Based Bayesian Reinforcement Learning. In *AAAI*, 3548–3555. AAAI Press.

Kober, J.; Bagnell, J. A.; and Peters, J. 2013. Reinforcement learning in robotics: A survey. *Int. J. Robotics Res.*, 32(11): 1238–1274.

Kolobov, A.; Mausam; and Weld, D. S. 2012. A Theory of Goal-Oriented MDPs with Dead Ends. In *UAI*, 438–447. AUAI Press.

Könighofer, B.; Alshiekh, M.; Bloem, R.; Humphrey, L.; Könighofer, R.; Topcu, U.; and Wang, C. 2017. Shield synthesis. *Formal Methods in System Design*, 51(2): 332–361.

Laud, A.; and DeJong, G. 2003. The Influence of Reward on the Speed of Reinforcement Learning: An Analysis of Shaping. In *ICML*, 440–447. AAAI Press.

Madani, O.; Hanks, S.; and Condon, A. 1999. On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems. In *AAAI*, 541–548. AAAI Press.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5602.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Moldovan, T. M.; and Abbeel, P. 2012. Safe Exploration in Markov Decision Processes. In *ICML*. icml.cc / Omnipress.

Peters, J.; and Schaal, S. 2006. Policy Gradient Methods for Robotics. In *IROS*, 2219–2225. IEEE.

Pnueli, A. 1977. The temporal logic of programs. In *Foundations of Computer Science*, 46–57. IEEE.

Pryor, L.; and Collins, G. 1996. Planning for Contingencies: A Decision-based Approach. *J. Artif. Intell. Res.*, 4: 287–339.

Puterman, M. L. 1994. *Markov Decision Processes*. John Wiley and Sons.

Raskin, J.; Chatterjee, K.; Doyen, L.; and Henzinger, T. A. 2007. Algorithms for Omega-Regular Games with Imperfect Information. *Log. Methods Comput. Sci.*, 3(3).

Sallab, A. E.; Abdou, M.; Perot, E.; and Yogamani, S. K. 2017. Deep Reinforcement Learning framework for Autonomous Driving. *CoRR*, abs/1704.02532.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347.

Shperberg, S. S.; Liu, B.; and Stone, P. 2022. Learning a Shield from Catastrophic Action Effects: Never Repeat the Same Mistake. *CoRR*, abs/2202.09516.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T. P.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489.

Smith, T.; and Simmons, R. 2004. Heuristic search value iteration for POMDPs. In *UAI*, 520–527. AUAI Press.

Sutton, R. S.; and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Tao, F.; Zhang, H.; Liu, A.; and Nee, A. Y. C. 2019. Digital Twin in Industry: State-of-the-Art. *IEEE Trans. Ind. Informatics*, 15(4): 2405–2415.

van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI*, 2094–2100. AAAI Press.

Wierstra, D.; Förster, A.; Peters, J.; and Schmidhuber, J. 2007. Solving deep memory POMDPs with recurrent policy gradients. In *ICANN*, 697–706. Springer.

Williams, R. J. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8: 229–256.