Safe Reinforcement Learning via Shielding for POMDPs

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

Abstract

Reinforcement learning (RL) in safety-critical environments requires an agent to avoid decisions with catastrophic consequences. Various approaches addressing the safety of RL exist to mitigate this problem. In particular, so-called shields provide formal safety guarantees on the behavior of RL agents based on (partial) models of the agents' environment. Yet, the state-of-the-art generally assumes perfect sensing capabilities of the agents, which is unrealistic in real-life applications. The standard models to capture scenarios with limited sensing are partially observable Markov decision processes (POMDPs). Safe RL for these models remains an open problem so far. We propose and thoroughly evaluate a tight integration of formally-verified shields for POMDPs with state-of-the-art deep RL algorithms and create an efficacious method that safely learns policies under partial observability. We empirically demonstrate that an RL agent using a shield, beyond being safe, converges to higher values of expected reward. Moreover, shielded agents need an order of magnitude fewer training episodes than unshielded agents, especially in challenging sparse-reward settings.

1 INTRODUCTION

Reinforcement learning (RL) [Sutton and Barto, 1998] is a machine learning technique for decision-making in uncertain and dynamic 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 et al., 2013] or autonomous driving [Sallab et al., 2017].

One of the major limitations for 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].

Safe RL in partially observable environments suffers from uncertainty both in the agent's actions and perception. Such environments, typically modeled as partially observable Markov decision processes (POMDPs) [Kaelbling et al., 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.

We capture safety by reach-avoid specifications, a special case of temporal logic constraints Pnueli [1977]. To provide safety guarantees, we capture assumptions on the system dynamics in a *partial model of the environment*. In particular, while we need to know all potential transitions in the POMDP, probabilities and rewards may remain unspecified [Raskin et al., 2007]. Under this (necessary) assumption, we compute a *shield* that ensures verifiably safe behavior of an RL agent. While obtaining good partial models may be intricate, model-based engineering is widespread in safety-critical situations. Probabilities in these models may be rough estimates at best, but if a transition exists (with positive probability) is often much better understood.

The availability of a (partial) model allows to tap into existing work on model-based reasoning to extract the aforementioned shield. However, despite tremendous progress [Pineau et al., 2003, Walraven and Spaan, 2017, Silver and Veness, 2010], model-based reasoning, especially verification, has limitations: Even if a POMDP is completely known, scalability remains a challenge. Already, whether for a POMDP there exists a policy that satisfies a temporal logic specification is undecidable [Madani et al., 1999]. However, computing policies for qualitative reach-avoid specifications is EXPTIME-complete [Chatterjee et al., 2015]. While this still limits the application in full generality, efficient methods based on satisfiability solvers show good empirical scalability [Chatterjee et al., 2016, Junges et al., 2021].

Our contribution is the first method to shielding for POMDPs. We employ an effective integration of shields computed via satisfiability solving [Junges et al., 2021] with various state-of-the-art RL algorithms from Tensorflow [Guadarrama et al., 2018], and we provide an extensive experimental evaluation. We show the following natural effects that arise from such a combination.

- *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 inherent state estimation.
- *Safety after learning:* A *trained agent* that has an incentive to adhere to safety still behaves unsafe sometimes. Moreover, typical unwanted tradeoffs in settings with safety and (additional) performance objectives are avoided when (1) safety is (strictly) enforced via shields and (2) the agent focuses on performance.
- *RL convergence:* A shield not only ensures safety, but also significantly reduces the search space and the required amount of data for RL.

Fig. 1 shows the outline of our approach. We demonstrate effects and insights on shielded RL for POMDPs using several typical examples and provide detailed information on RL performance as well as videos showing the exploration and training process. To investigate to what extent more lightweight alternatives to a shield help RL, we experiment with a state estimator. This estimator uses the partial model to track in which states the model may be, based on the observed history. 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. Finally, 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 may help to bootstrap an RL agent.

Further related work. Several approaches to safe RL in combination with formal verification exist [Hasanbeig et al., 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 do not consider reinforcement learning [Cubuktepe et al., 2021].

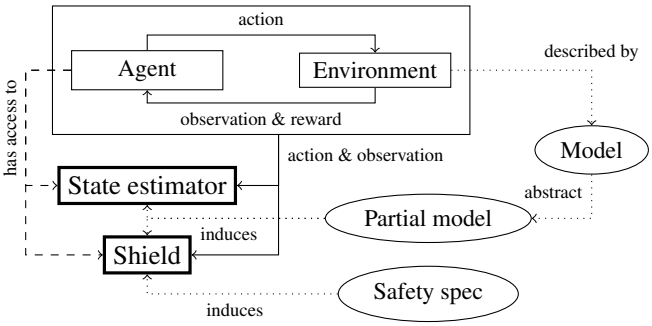


Figure 1: RL for partial-observation and partial-model information accessible via a state estimator and a shield.

2 PROBLEM STATEMENT

In this section, we introduce POMDPs as the standard model for environments for sequential decision-making under partial observability. We distinguish the learning goal of an agent that operates in such an environment, and the agent's safety constraints. We capture the former by expected rewards, the latter 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 over the states that gives the probability $I(s)$ that the agent starts in state $s \in S$, and Act is a finite space of actions for the agent to take. Z is a finite observation space and $O(z|s)$ is the probability of observing z when the environment is in state s. Finally, $\mathcal{P}(s'|s, a)$ is a transition model representing the conditional probability of moving to a state $s' \in S$ after executing action $a \in A$ in state $s \in S$. Not every action is available in every state, i.e., $\mathcal{P}$ is a partial function. The set of available actions in state s is $Act(s)$. When executing action $a \in Act$ in state $s \in S$, the agent receives a scalar reward $\mathcal{R}(s, a)$. We remark that our POMDPs have dead-ends from which an agent cannot obtain positive rewards [Kolobov et al., 2012]. We describe agent behavior via a (observation-based) policy $\pi\colon (Z \times Act)^* \times Z \to Distr(Act)$ that maps an observation sequence τ to a distribution over actions. In contrast to the (observable) MDPs, the agent may depend on the history of actions – this means that the agent has to store (aspects of) the history of observations.

Problem 1. *Given a POMDP $\mathcal{M}'$, the problem is to find a policy π that maximizes the expected discounted reward $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t\right]$ for POMDP $\mathcal{M}$, where γ^t with $0 \leq \gamma^t \leq 1$ is the discount factor and $\mathcal{R}_t$ is the reward the agent receives at time t.*

In this standard problem for POMDPs, maximizing the expected reward is the *learning goal* of the agent.

2.2 SAFETY CONSTRAINTS

In addition to the learning goal, an agent in safety-critical settings must 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 $\mathit{AVOID} \subseteq S$ and reach states from $\mathit{REACH} \subseteq S$ *almost-surely*, i.e., with probability one (for arbitrary long horizons). We denote these constraints by $\varphi = \langle \mathit{REACH}, \mathit{AVOID} \rangle$. The relation $\mathcal{M}(\pi) \models \varphi$ denotes that the agent adheres to the specification φ under the policy π. We formalize such *winning* policies in the next section.

Problem 2. *Given a POMDP $\mathcal{M}$, the problem is to find a policy π that maximizes $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t\right]$ for POMDP $\mathcal{M}$ while π is* winning*, that is, $\mathcal{M}(\pi) \models \varphi$.*

Note that an optimal solution to Problem 2 may induce a lower reward than for Problem 1, 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 main ingredients for the proposed methodology, as outlined in Figure 1. We discuss beliefs over environment states and belief supports. Then, we introduce the notion of a safety shield. Finally, we discuss the guarantees we are able to provide using shields, and the particular assumptions we have to make.

3.1 BELIEFS AND BELIEF SUPPORTS

Belief states. As the *current* state of a POMDP is not observable, agents may infer an estimation of the system state from a sequence of observations. This estimation is typically a *belief* of the form $\mathfrak{b}\colon (Z \times \mathit{Act})^* \times Z \to \mathit{Distr}(S)$, that is, a distribution that describes the probability that we are currently in a certain state based on the history so far. Consequently, a policy $\pi\colon \mathfrak{b} \to \mathit{Distr}(\mathit{Act})$ can also directly be defined on the beliefs. An agent may incrementally update the belief upon receiving new observations using a Bayesian update. This belief update depends on the transition (and observation) probabilities in the POMDP. The belief dynamics can be captured by a (fully observable) *belief MDP* in which the (infinitely many) beliefs of the POMDP are the states. Due to this infinite number of beliefs, computing a policy that maximizes the reward is generally undecidable [Madani et al., 1999]. This is in contrast to handling qualitative reach-avoid specifications, as we detail below.

Winning beliefs. For a set of states $S' \subseteq S$ of the POMDP, $Pr^{\pi}_{\mathfrak{b}}(S)$ denotes the probability to reach 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 $Pr^{\pi}_{\mathfrak{b}}(\mathit{AVOID}) = 0$ and $Pr^{\pi}_{\mathfrak{b}}(\mathit{REACH}) = 1$, i.e., if it reaches AVOID with probability zero and REACH with probability one (almost-surely) when $\mathfrak{b}$ is the initial state. Belief $\mathfrak{b}$ is* winning *for φ in $\mathcal{M}$ if there exists a winning policy from $\mathfrak{b}$.*

For multiple beliefs, we define *winning regions* (aka safe or controllable regions). A winning region (for POMDPs) is a set of winning beliefs, that is, from each belief within a winning region, there exists a winning policy.

Belief support. A state s with positive belief $\mathfrak{b}(s) > 0$ is in the *belief support*, that is, $s \in \mathrm{supp}(\mathfrak{b})$. The belief-support can be updated using only the graph of the POMDP (without probability knowledge) by a simplified belief update. The following result constitutes the basis of the correctness of our approach.

Theorem 1 (Junges et al. [2021])**.** *For a winning belief $\mathfrak{b}$, any belief $\mathfrak{b}'$ with $\mathrm{supp}(\mathfrak{b}') = \mathrm{supp}(\mathfrak{b})$ is winning.*

That means, we only need to take the finite set of belief supports into account to compute winning policies, beliefs, and regions for qualitative reach-avoid properties [Raskin et al., 2007]. Technically, one has to construct a (finite, albeit exponential) belief-support (stochastic) game that provides a suitable abstraction of the belief MDP [Junges et al., 2020]. We directly define polices on the belief support of the form $\pi^{\mathfrak{b}}\colon B \to \mathit{Act}$, where B denotes the set of all belief supports. Basically, this *pure* or *deterministic* policy chooses one unique action for each belief support $\mathrm{supp}(\mathfrak{b})$.

3.2 SHIELDS

The purpose of a shield is to prevent the agent from taking actions which would violate a (reach-avoid) specification. For avoid specifications, the shield prevents the agent from entering avoid states, or from entering states from which it is impossible to prevent reaching an avoid state in the future. Consequently, a shield ensures that an agent stays in a winning region. To stay inside this region, the agent must pick an action such that all successor states with respect to this action (from the current belief) are also inside the winning region. For reach-avoid specifications, a shield additionally prevents the agent from visiting dead-ends. A shield itself

cannot force an agent to visit reach states. However, under mild assumptions, we can additionally ensure that the agent eventually visits the reach state: It suffices to assume that the agent is fair[1]. w.r.t. the actions that stay within the winning region. We remark that most RL agents are fair.

Technically, we define a shield as a set of (winning) policies. In the literature, such a set of policies is referred to as a *permissive* policy [Dräger et al., 2015, Junges et al., 2016].

Definition 2 (Permissive policy and shield)**.** *Given a POMDP $\mathcal{M}$, a permissive policy is given by $\nu\colon \mathfrak{b} \to 2^{Act}$. A policy π is* admissible *for ν if for all beliefs $\mathfrak{b}$ it holds that $\pi(\mathfrak{b}) \in \nu(\mathfrak{b})$. A permissive policy is a φ-*shield for $\mathcal{M}$ *if all its admissible policies are winning.*

Such a set of policies allows multiple actions at each state, as long as these actions belong to policies that satisfy the specification. Note that as a consequence of Theorem 1, the computation of a shield is based on the belief support. We will detail the necessary prerequisites in the following.

3.3 SAFETY GUARANTEES

A provably-correct shielding approach necessarily requires prior knowledge on the model. We discuss the exact type of knowledge that is needed to provide safety guarantees.

Partial models. We assume the agent only has access to a partial model $\mathcal{M}' = (S, I, Act, O, Z, \mathcal{P}')$ where the transition model $\mathcal{P}'$ yields unknown, but positive probabilities. Essentially, $\mathcal{P}'$ defines a set of (possible) transitions. We say that a POMDP $\mathcal{M} = (S, I, Act, O, Z, \mathcal{P})$ and a partial model $\mathcal{M}' = (S, I, Act, O, Z, \mathcal{P}')$ have *coinciding transitions* iff it holds for all states $s, s' \in S$ and actions $a \in Act$ that $\mathcal{P}(s'|s,a) > 0$ iff $\mathcal{P}'(s'|s,a) > 0$. Intuitively, the partial model defines exactly the graph of the original POMDP. Similarly, *$\mathcal{M}'$ overapproximates the transition model of $\mathcal{M}$*, if it holds for all states $s, s' \in S$ and actions $a \in Act$ that $\mathcal{P}(s'|s,a) > 0$ if $\mathcal{P}'(s'|s,a) > 0$. The original POMDP has no transitions that are not present in the partial model.

We state the following results about the guarantees a shield is able to provide, depending on the partial model.

Theorem 2 (Reach-Avoid Shield)**.** *Let $\mathcal{M}$ and $\mathcal{M}$ be two POMDPs with coinciding transitions, and $\varphi = \langle$REACH, AVOID$\rangle$ a reach-avoid specification, then a φ-shield for the partial model $\mathcal{M}'$ is a φ-shield for $\mathcal{M}$.*

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

[1]Fairness is a notion from formal verification which ensures that an agent that visits a state infinitely often must take every (safe) action available in that state infinitely often. An agent that takes any (safe) action with positive probability is fair.

For avoid specifications, we can further relax the assumptions while still giving the same hard guarantees. Intuitively, it suffices to require that each transition in the partial model exists (with positive probability) in the (true) POMDP.

Theorem 3 (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 that provides the same hard guarantees. Nevertheless, shields may empirically significantly improve performance or safety of RL agents, as we will demonstrate in our experiments.

4 SHIELDS AND RL IN POMDPS

We instantiate Figure 1 as follows: While the environment is described as a (discrete) POMDP, we assume that the agent can only depend on partial models, as discussed in the previous section. In this section, we discuss two interfaces to this knowledge that can be used independently or in conjunction. We underpin qualitatively how these interfaces help state-of-the-art RL agents. In the experimental evaluation that follows in the next section, we see that the quantitative advantage to the RL agent is an intricate function of both the domain and the RL agent.

Using the partial model via a shield. We assume the availability of a shield that ensures reach-avoid specifications as outlined above. Following Theorem 1, such a shield can be computed symbolically using modern satisfiability solvers [Junges et al., 2021]. We exploit our definition of state estimators, belief supports, and the assumptions and results in Theorems 2 and 3. Essentially, we use a state estimator $\mathfrak{b}\colon (Z \times Act)^* \times Z \to Distr(S)$ to create a shield $\nu\colon \mathrm{supp}(\mathfrak{b}) \to 2^{Act}$ that operates directly on the belief support, see Theorem 1. For the specification φ, this shield, yields for every belief the set of *safe actions*. We restrict the available actions for the agent to these safe actions.

Using the partial model via a state estimator. As an additional, light-weight, interface, we investigate the availability of a *belief-support state estimator* as is also used by the shield internally. This estimator $(Z \times Act)^* \times Z \to 2^S$ yields, based on the sequence of observations and previous actions, the set of POMDP states that could have been reached so far. The agent can use the state estimation as an additional observation as basis for the decision-making.

4.1 SAFETY DURING LEARNING

(Only) shielded RL agents can enforce safety during learning. Notice that without the notion of a shield, the agent

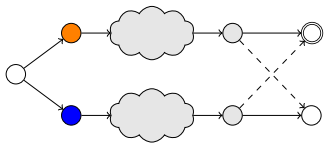


Figure 2: Illustration for *estimators accelerate learning*.

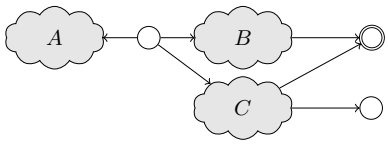


Figure 3: Illustration for *shields accelerate learning*.

must take an action first to understand that it may lead to a bad state. While an adequately constructed shield ensures that we visit the reach-states eventually (with probability one), we cannot give an upper bound on the number of steps required to visit these states. However, we can construct shields for finite-horizon reach-avoid specifications using the same methods for a modified POMDP. We remark that if the partial model is not faithful to the true POMDP, that is, it has a different graph structure, the shielded agent may violate the specification.

State estimators themselves do not directly contribute to safe exploration. However, the additional observations do help to recognize critical states. In particular, consider an action (such as switching on a light) which is useful and safe in most situations (except maybe a gas leakage). A state estimator may provide the additional observation signals that allow the RL agent to efficiently distinguish these states, thereby indirectly improving safety, even during learning.

4.2 SAFETY AFTER LEARNING

Even after successful learning, agents which keep exploring may keep violating safety. Furthermore, reward objectives and safety constraints may not coincide, e.g., the reward structure may be richer. While safety may only avoid car-crashes, learning objectives may include performance measures such as fuel consumption. The combination of objectives is non-trivial, and weighted combinations lead to a trade-off between safety and performance. Then, even in the limit (after convergence), an RL agent may keep violating the safety constraints. On the other hand, in presence of a shield, the RL agent may fully focus on the performance measures as safety is already guaranteed. The effect of state estimators before and after learning is roughly analogous.

4.3 RL CONVERGENCE SPEED

Even 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. In particular, the availability of a state estimator allows enriching the observation with a signal that compresses the history. Consider the POMDP sketch in Fig. 2, illustrating a typical example 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 signal that includes whether we are in the bottom or top part of the model, and thus significantly simplifies the learning.

Slightly orthogonal, a shield may provide incentives to (not) explore parts of the state space. Consider an environment as sketched out in Fig. 3. We have partitioned the state space into three disjoint parts. In region A, there are no avoid states (with a high negative reward) but neither are there any positive rewards, thus, region A is a dead-end. In region B, all states will eventually reach a positive reward, and in region C, there is a (small) probability that we eventually reach an avoid state with a high negative reward. 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.

4.4 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 hot-start 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, as there is no further need to integrate the shield or state estimator in these analyses. We 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 of the shield, we give immediate negative rewards whenever an action not allowed by the

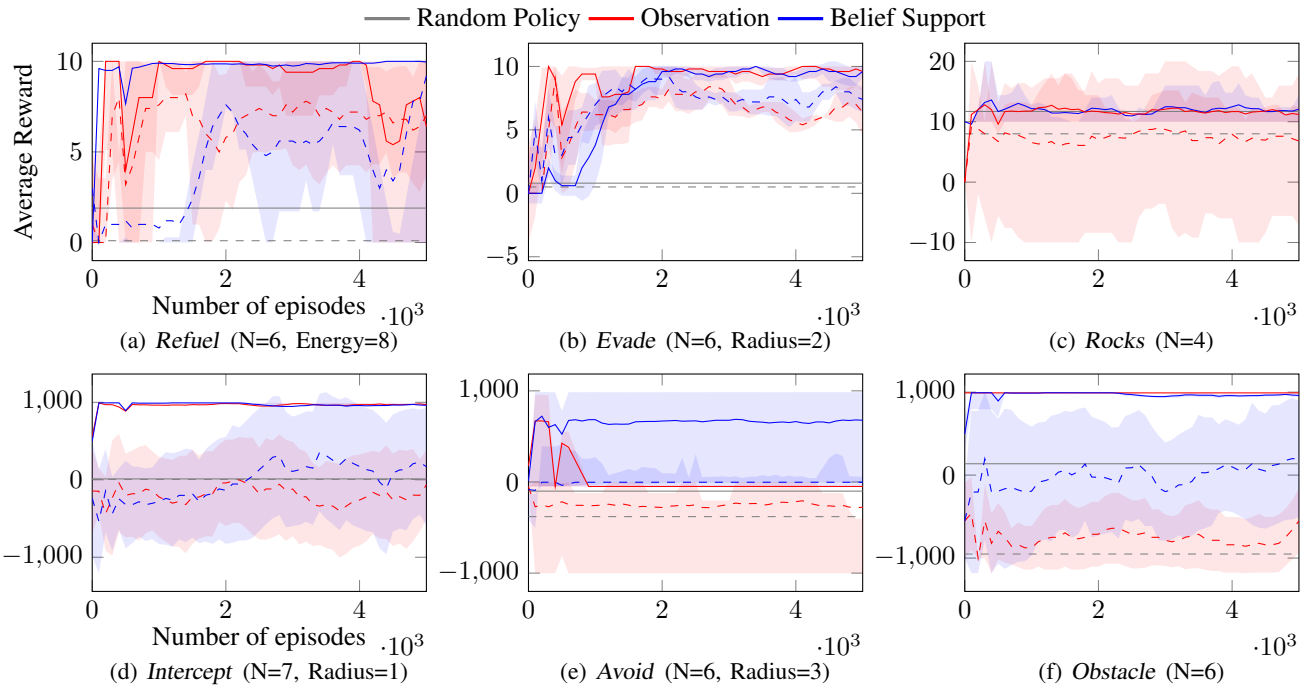


Figure 4: REINFORCE performed with (solid) and without (dashed) a shield restricting unsafe actions. The red lines show when the RL agent is trained using only the observations and the blue lines indicate when the RL agent is trained using some state estimation in the form of belief support. The gray lines are the average reward obtained by applying a random policy.

shield is taken. We decay this negative reward over time to gently fade out the effect of a shield.

When switching off state estimators, the learned agent is now no longer executable as it lacks necessary information. Naive solutions for this problem can be trivially supported, e.g. by defaulting to a fixed observation. We leave a proper study of switching off state estimators for future work.

5 EXPERIMENTS

We applied shielded RL in six tasks involving agents operating in partially observable $N \times N$ grids. We compared the shield's performance in five different deep RL methods: DQN [Mnih et al., 2015], DDQN [van Hasselt et al., 2016], PPO [Schulman et al., 2017], discrete SAC [Christodoulou, 2019] and REINFORCE [Williams, 1992].

Setup. We use the POMDP environments from [Junges et al., 2021]], in particular *Refuel*, *Evade*, *Rocks*, *Intercept*, *Avoid* and *Obstacle*. Each has a nominal *REACH* objective and a set of *AVOID* locations that trap the agent in place, for a full description of the domains and their reward structures see the domain descriptions in the Appendix. The environments come with a simulator and a belief-support tracker based on Storm [Dehnert et al., 2017]. Shields are computed using the satisfiability checker Z3 Jovanovic and de Moura [2012]. We developed bindings to Tensorflow's *TF-Agents* package [Guadarrama et al., 2018] and connect the provided state-of-the-art implementations of the aforementioned algorithms, in particular, we use the masking in TensorFlow to enforce the precomputed shield. We provide full details of the implementation, the hyperparameters and the selection method in the supplementary material. All experiments were performed using an 8-core 3.2GHz Intel Xeon Platinum 8000 series processor with 32GB of RAM. For each experiment, 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, Hausknecht and Stone, 2015].

5.1 RESULTS

In Figure 4, we demonstrate the performance of an RL agent on the aforementioned domains. In this and subsequent plots, the dashed lines indicate RL agents learning without

the benefit of the shield, while solid lines indicate that the agent uses shields. In addition, we include the For brevity, the majority of the comparisons in this section show the REINFORCE algorithm. We include the source code, the full set of results and plots for all learning methods and domains in the data appendix. 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 impacts learning.

Domains are sparse and thus challenging. This observation may not be surprising, but the domains considered are sparse. Without side-information (from the model), the deep RL algorithms struggle to handle the partially observable domains. In particular, actually reaching target states with a random policy is very unlikely, for example in *Evade* (Fig. 4(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*, one locally optimal but globally sub-optimal policy, which obtains an average reward of -100 (global optimum of $+991$). With this policy, which keeps the agent in the initial corner in the grid, the agent stays outside of the adversary's reachable space but will not attempt to move to the goal at all. Similarly, the unshielded random policy often reaches a highly negative reward: e.g., 95% of the time in *Obstacle* (Fig. 4(f)). This is a challenge for many RL agents: In Fig. 8, we illustrate the problematic performance on the *Intercept* domain for a variety of unshielded RL agents.

Shields enforce safety specifications. The shield ensures that the agent stays within the winning region by preventing it from taking actions that may result in reaching avoid states or dead-ends. Indeed, we did not observe shielded agents ever violating the safety specification.

Shields accelerate convergence. Shielded agents avoid encountering avoid states on all episodes, and other episodes are thus more frequent. Consequently, a shielded RL agent has a higher probability of achieving the sparse reward. 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. 4(f).

Shields do not enforce reaching targets quickly. As a drawback, shielding does not directly steer the agent towards a positive reward. In environments like *Evade*, even with the shield, the reward is particularly sparse, where a random policy with unsafe actions shielded has only an 8% chance of reaching the goal, as shown in Fig. 4(b). Thus 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. In Fig. 5, we show that in *Refuel*, only when exploring sufficiently long episodes, the agent converges towards an optimal policy. In this domain, the agent must rely on the uncertain dynamics to reach the goal without running out of fuel. Just before the possibility of diverting to far from a recharge station, the shield enforces backing up and recharging. It may require several attempts before the agent reaches the goal.

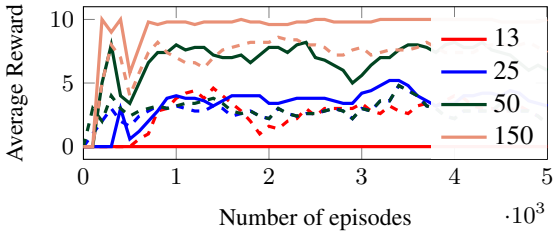
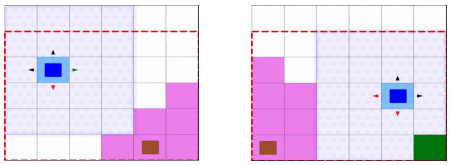


Figure 5: Variable episode maximum length for *Refuel*.

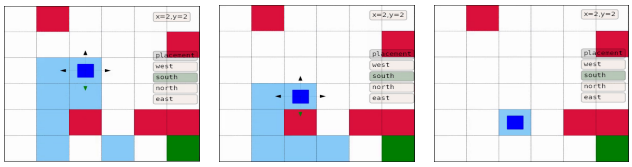


(a) *Evade* at $t = 9$ (b) *Evade* at $t = 25$

Figure 6: 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 $\{\text{south}, \text{east}\}$ and the agent takes $\{\text{scan}\}$.

Shields may have little effect on performance. For the domain *Evade* in Fig. 4(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. Such an example is illustrated in Fig. 6, where the agent can easily either take an action that is just as beneficial as the one that was restricted as in Fig. 6(a) or reduce the uncertainty by taking a scan as in Fig. 6(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 often does not suffer any negative outcome, leading to similar values of average reward.

Shields can degrade performance. Back to *Refuel*, we observe that for (very) short episodes, an unshielded agent may perform better. The agent in Fig. 5 (red dashed) takes the necessary "risk" of potentially running out of fuel and using the uncertain dynamics to reach the goal under 13 steps in many (but not all) cases. This violates the safety constraint, but the performance is better than when the (shielded) agent never reaches the goal. This effect fades out with increasing episode length, because the probability



(a) *Obstacle* at $t = 2$ (b) *Obstacle* at $t = 3$ (c) *Obstacle* at $t = 4$

Figure 7: Incremental states of *Obstacle* environment where the agent (dark blue) handles uncertainty by maintaining a belief set of states (shaded in blue). The goal (green) and obstacles (red) are static. At $t = 2$ the agent takes south and again at $t = 3$, which results in a collision at $t = 4$

that the dynamics turn out favorably increases over time.

Unsafe actions can have high average rewards. One of the challenges of RL in partially observable environments is handling a potentially ambiguous and conflicting set of states. The agent must learn to distinguish states with similar observations. This challenge is most evident in the *Obstacle* domain. Consider the agent in Fig. 7, which could occupy any one of the blue shaded states. At the agent's position at $t = 2$ in Fig. 7(a), estimated Q-values (from DQN) are roughly $(733, 784, 606, 687)$ for $(\text{west}, \text{south}, \text{north}, \text{east})$ respectively. The unshielded RL agent in this situation is willing to risk possible collision if the agent is in state $x = 2$ for the significant advantage gained by taking south for any state in $x = 1$. Then, the agent collides with the obstacle at $(x = 3, y = 4)$, yielding a -1000 penalty. When the belief support contains just the $x = 2$ states, the Q-values are $(499, -456, -417, 404)$, which indicates that the DQN algorithm is struggling to account for high uncertainty. Shields disable such actions and thus improve further convergence.

A belief-support state estimator can accelerate RL, but a shield helps more. The challenge of RL agents struggling with high uncertainty, as sketched in the previous paragraph, can also occur when shielded. Again, in the *Obstacle* domain, REINFORCE without the state estimation (red) 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), the agent quickly 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 if they have access to a state estimator.

Shielding is more effective on some RL agents than on others. In Fig. 8, we compare how shielding benefits different learning methods for the *Intercept* domain. In this example, all learning methods benefit from the shield. However, the DQN and DDQN struggle to converge to the optimal policy. Such behavior could be the result of insufficient data to properly process the state estimates from the shield.

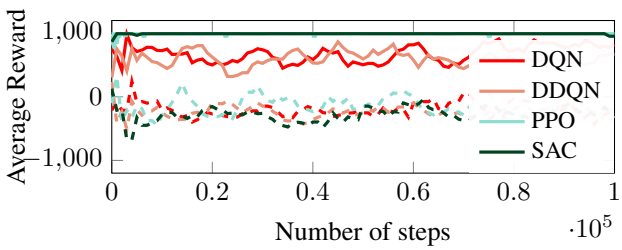


Figure 8: *Intercept* with an RL agent performing different learning methods. Each agent used the shield's belief support as the input representation.

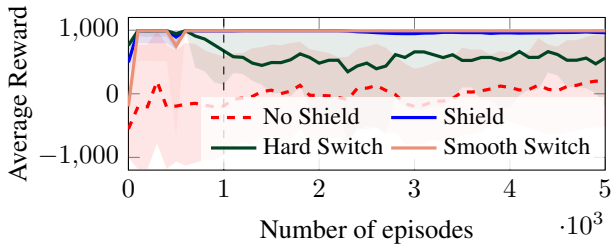


Figure 9: *Obstacle* with an RL agent that learns for the first 1000 episodes with the shield active. After 1000 episodes the shield is either switched off completely (green) or is slowly turned off with increasing probability (purple).

Shielding can bootstrap RL agents. In Fig. 9, 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 (purple). For the latter, we apply the shield with probability p, where p starts at 1 and is reduced by the learning rate α until $p = 0$. The RL agent that initially learns to use the shield, generates higher quality episodes and subsequently, when the shield is removed, the agent still maintains higher quality rollouts since it has previously experienced the sparse positive reward. The effect is even more pronounced as the shield is gradually removed, where the performance mirrors the shielded condition.

6 CONCLUSION AND FUTURE WORK

We presented 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 et al., 2006] as an additional interface to the agent.

References

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *AAAI*. AAAI Press, 2018.

Piergiorgio Bertoli, Alessandro Cimatti, and Marco Pistore. Towards strong cyclic planning under partial observability. In *ICAPS*, pages 354–357. AAAI, 2006.

Maxime Bouton, Jesper Karlsson, Alireza Nakhaei, Kikuo Fujimura, Mykel J. Kochenderfer, and Jana Tumova. Reinforcement learning with probabilistic guarantees for autonomous driving. *CoRR*, abs/1904.07189, 2019.

Krishnendu Chatterjee, Martin Chmelík, Raghav Gupta, and Ayush Kanodia. Qualitative analysis of pomdps with temporal logic specifications for robotics applications. In *ICRA*, pages 325–330. IEEE, 2015.

Krishnendu Chatterjee, Martin Chmelik, and Jessica Davies. A symbolic sat-based algorithm for almost-sure reachability with small strategies in pomdps. In *AAAI*, pages 3225–3232. AAAI Press, 2016.

Petros Christodoulou. Soft actor-critic for discrete action settings. *CoRR*, abs/1910.07207, 2019.

Murat Cubuktepe, Nils Jansen, Sebastian Junges, Ahmadreza Marandi, Marnix Suilen, and Ufuk Topcu. Robust finite-state controllers for uncertain pomdps. In *AAAI*, pages 11792–11800. AAAI Press, 2021.

Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *Computer Aided Verification (CAV)*, volume 10427 of *LNCS*, pages 592–600. Springer, 2017.

Klaus Dräger, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. Permissive controller synthesis for probabilistic systems. *Logical Methods in Computer Science*, 11(2), 2015.

Nathan Fulton and André Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *AAAI*. AAAI Press, 2018.

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

Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Neal Wu, Efi Kokiopoulou, Luciano Sbaiz, Jamie Smith, Gábor Bartók, Jesse Berent, Chris Harris, Vincent Vanhoucke, and Eugene Brevdo. TF-Agents: A library for reinforcement learning in tensorflow. https://github.com/tensorflow/agents, 2018.

Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Cautious reinforcement learning with logical constraints. In *AAMAS*, pages 483–491. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

Matthew J. Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI*, pages 29–37. AAAI Press, 2015.

Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. Safe Reinforcement Learning Using Probabilistic Shields (Invited Paper). In *CONCUR*, volume 171 of *LIPIcs*, pages 3:1–3:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

Dejan Jovanovic and Leonardo Mendonça de Moura. Solving non-linear arithmetic. In *IJCAR*, volume 7364 of *LNCS*, pages 339–354. Springer, 2012.

Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen. Safety-Constrained Reinforcement Learning for MDPs. In *TACAS*, 2016.

Sebastian Junges, Nils Jansen, and Sanjit A. Seshia. Enforcing almost-sure reachability in pomdps. *CoRR*, abs/2007.00085, 2020.

Sebastian Junges, Nils Jansen, and Sanjit A. Seshia. Enforcing almost-sure reachability in pomdps. In *CAV (2)*, volume 12760 of *LNCS*, pages 602–625. Springer, 2021.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998.

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

Andrey Kolobov, Mausam, and Daniel S. Weld. A theory of goal-oriented mdps with dead ends. In *UAI*, pages 438–447. AUAI Press, 2012.

Bettina Könighofer, Mohammed Alshiekh, Roderick Bloem, Laura Humphrey, Robert Könighofer, Ufuk Topcu, and Chao Wang. Shield synthesis. *Formal Methods in System Design*, 51(2):332–361, 2017.

Adam Laud and Gerald DeJong. The influence of reward on the speed of reinforcement learning: An analysis of shaping. Technical report, 2003.

Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *AAAI*, pages 541–548. AAAI Press, 1999.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Teodor Mihai Moldovan and Pieter Abbeel. Safe exploration in Markov decision processes. In *ICML*. icml.cc / Omnipress, 2012.

Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *IROS*, pages 2219–2225. IEEE, 2006.

Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, pages 1025–1032, 2003.

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

Louise Pryor and Gregg Collins. Planning for contingencies: A decision-based approach. *J. Artif. Intell. Res.*, 4:287–339, 1996.

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

Jean-François Raskin, Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Algorithms for omega-regular games with imperfect information. *Log. Methods Comput. Sci.*, 3(3), 2007.

Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Kumar Yogamani. Deep reinforcement learning framework for autonomous driving. *CoRR*, abs/1704.02532, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *NIPS*, pages 2164–2172. MIT Press, 2010.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587): 484–489, 2016.

Trey Smith and Reid Simmons. Heuristic search value iteration for POMDPs. In *UAI*, pages 520–527. AUAI Press, 2004.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Fei Tao, He Zhang, Ang Liu, and Andrew Y. C. Nee. Digital twin in industry: State-of-the-art. *IEEE Trans. Ind. Informatics*, 15(4):2405–2415, 2019.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, pages 2094–2100. AAAI Press, 2016.

Erwin Walraven and Matthijs Spaan. Accelerated vector pruning for optimal pomdp solvers. In *AAAI*, pages 3672–3678. AAAI Press, 2017.

Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. Solving deep memory pomdps with recurrent policy gradients. In *ICANN*, pages 697–706. Springer, 2007.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

DATA APPENDIX

DOMAIN DESCRIPTIONS

Rocks *Rocks* is a variant of *RockSample* [Smith and Simmons, 2004]. The grid contains two rocks which are either valuable or dangerous to collect. To find out with certainty, the rock has to be sampled from an adjacent field. The goal is to collect a valuable rock (+10 reward), bring it to the drop-off zone (+10), and not collect dangerous rocks (-10).

Refuel *Refuel* concerns a rover that shall travel from one corner to the other (+10 reward), while avoiding an obstacle on the diagonal. Every movement costs energy, and the rover may recharge at dedicated stations to its full battery capacity, but neither action yields a reward or cost. Collisions and empty battery levels terminate the episode. The rover receives noisy information about its position and battery level.

Evade *Evade* is a scenario where an agent needs to reach an escape door (+10 reward) and evade a faster robot. The agent has a limited range of vision (Radius), but may choose to scan the whole grid instead of moving.

Avoid *Avoid* is a related scenario where an agent attempts to reach a goal (+1000) in the opposite corner and keep a distance from patrolling robots on fixed routes that move with uncertain speed, yielding partial information about their position. If being caught, the robot receives a reward of (-1000). Furthermore, every step yields -1 reward.

Rocks	Episode Length	100
	Grid-size	4
Refuel	Episode Length	100
	Grid-size	6
	Maximum energy level	8
Evade	Episode Length	350
	Grid-size	6
	Vision radius	2
Avoid	Episode Length	100
	Grid-size	100
	Vision radius	3
Intercept	Episode Length	100
	Grid-size	7
	Vision radius	1
Obstacle	Episode Length	1
	Grid-size	100

Table 1: Constants and parameters for each environment in experimental setups.

Intercept Contrary to *Avoid*, in *Intercept* an agent aims to meet (+1000) a robot before that robot leaves the grid via one of two available exits (-1000). The agent has a view radius and observes a corridor in the center of the grid. Movements are penalized with a reward of -1.

Obstacle *Obstacle* describes an agent navigating through a maze (movement: -1) of static traps where the agent's initial state and movement distance is uncertain, and it only observes whether the current position is a trap (-1000) or exit (+1000).

HYPERPARAMETER SELECTION

Network parameters In this work we were mostly interested in comparing the effect of a shield on different RL methods and domains. Consequently, we ensured that the chosen hyperparameters were consistent between each experiment. An extensive tuning for each method and domain were outside the scope of this work. Consequently, we employed the default settings from the examples provided in the *tf-agents* [Guadarrama et al., 2018] documentation with one exception. For discrete SAC [Christodoulou, 2019], we modify the *tf-agents* Guadarrama et al. [2018] implement to handle discrete actions but also we added an LSTM layer in the actor network, see Table 2. The hyperparameter values for each learning setting are given in Tables 2 to 6

Actor Network Parameters	Hidden layers	3
	Nodes per layer	(400,300)
	LSTM size	40
	Activation function	ReLu
Critic Network Parameters	Hidden layers	2
	Nodes per layer	300
	LSTM size	40
	Activation function	tanh
Training Parameters	Optimizer	ADAM
	Learning rate	$3e-2$
	Minibatch size	64
	Discount γ	1
	Importance ratio clipping	1
	Target Update τ	0.05
	Target Update Period	5
Other Parameters	Evaluation Interval	1000
	Evaluation Episodes	10

Table 2: Hyperparameters used in discrete soft actor-critic (SAC) numerical experiments.

Q-Network Parameters	Hidden layers	1
	Nodes per layer	100
	Activation function	None
Training Parameters	Optimizer	ADAM
	Learning rate	$3e-2$
	Minibatch size	64
	Discount γ	1
Other Parameters	Evaluation Interval	1000
	Evaluation Episodes	10

Table 3: Hyperparameters used in deep Q-network (DQN) and double Q learning (DDQN) numerical experiments.

Actor Network Parameters	Hidden layers	2
	Nodes per layer	(200,100)
	Activation function	tanh
Value Network Parameters	Hidden layers	2
	Nodes per layer	(200,100)
	Activation function	ReLu
Training Parameters	Optimizer	ADAM
	Learning rate	$3e-2$
	Minibatch size	64
	Discount γ	1
Other Parameters	Evaluation Interval	1000
	Evaluation Episodes	10

Table 5: Hyperparameters used in proximal policy optimization (PPO) numerical experiments.

Q-Network Parameters	Hidden dense layers	2
	Nodes per layer	(50,20)
	Activation function	ReLu
	LSTM layer size	15
Training Parameters	Optimizer	ADAM
	Learning rate	$3e-2$
	Minibatch size	64
	Discount γ	1
Other Parameters	Evaluation Interval	1000
	Evaluation Episodes	10

Table 4: Hyperparameters used in deep recurrent Q-network (DRQN) in memory comparison experiment.

Actor Network Parameters	Hidden layers	1
	Nodes per layer	100
	Activation function	ReLu
Value Network Parameters	Hidden layers	1
	Nodes per layer	100
	Activation function	ReLu
	Value Est. Loss Coeff.	0.2
Training Parameters	Optimizer	ADAM
	Learning rate	$3e-2$
	Minibatch size	64
	Discount γ	1
Other Parameters	Evaluation Interval	100
	Evaluation Episodes	10

Table 6: Hyperparameters used in deep REINFORCE numerical experiments.

INPUT REPRESENTATION INSIGHTS

Input format The shield is more than just a state estimate. In fact, even when we include as much information as possible, in the form of a vector that stacks the observation, the belief-support state estimate and the action mask that a shield would recommend, the shielded RL agent still outperforms its unshielded counterpart. In Figure 10, a shielded RL agent with a simple observation representation (red) vastly outperforms the unshielded, high-information agent (dashed green).

Experience replay for POMDPs For the experience replay, we utilize the uniform sampled replay buffer with a mini-batch size of 64. For DQN, DDQN, PPO and discrete SAC we collect and train in step intervals and for REINFORCE, we collect data as full episode runs. We also conducted experiments where we gave the RL sequences of observations as an input for training. This experience replay technique is explored in Hausknecht and Stone [2015], where a RL agent with a DRQN can interpret partial information from multiple observations in sequnce. With that movitation we compared our discrete SAC agent (with its LSTM memory cell) for different input lengths, see Figure 11.

Learning Methods Data In Figures 12 to 15, we show the full set of experiments similar to Figure 4 in the paper for REINFORCE.

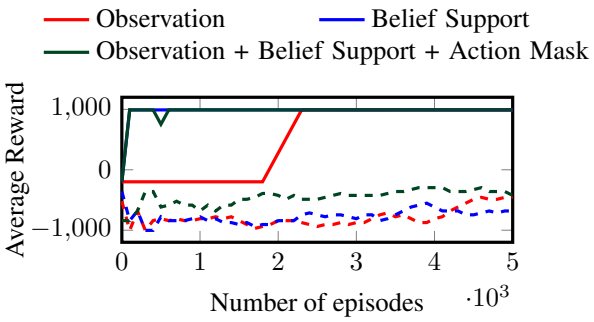


Figure 10: A comparison of three input representations for an RL agent learning on *Obstacle*. The combined representation (green) is an integer vector that contains the information of both the observation vector (red), the belief-support vector (blue) and the action mask at that instant.

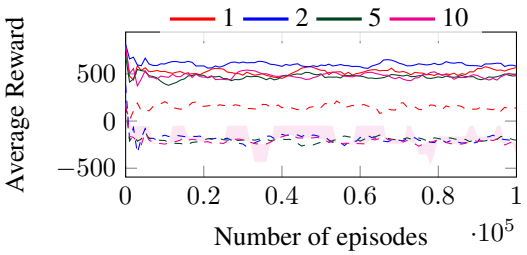
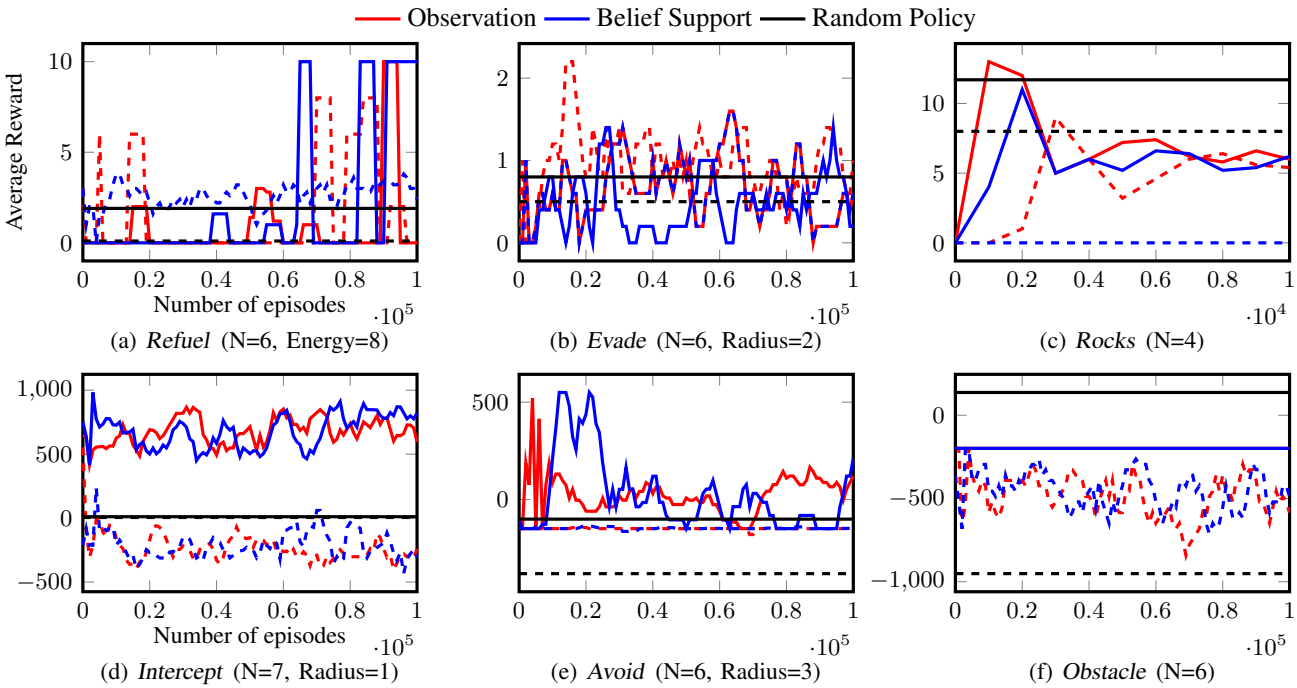


Figure 11: *Intercept* with an LSTM-based SAC agent that interprets sequences of observations through the use of a memory buffer. Each line represents a different instance of how many sequential observations was fed to each agent when learning. See Hausknecht and Stone [2015] for a detailed analysis for the interplay between partially observability and experience replay in RL agents.



(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 12: DQN performed with (solid) and without (dashed) a shield restricting unsafe actions. The red lines show when the RL agent is trained using only the observations and the blue lines indicate when the RL agent is trained using some state estimation in the form of belief support. The black lines are the average reward obtained by applying a random policy.

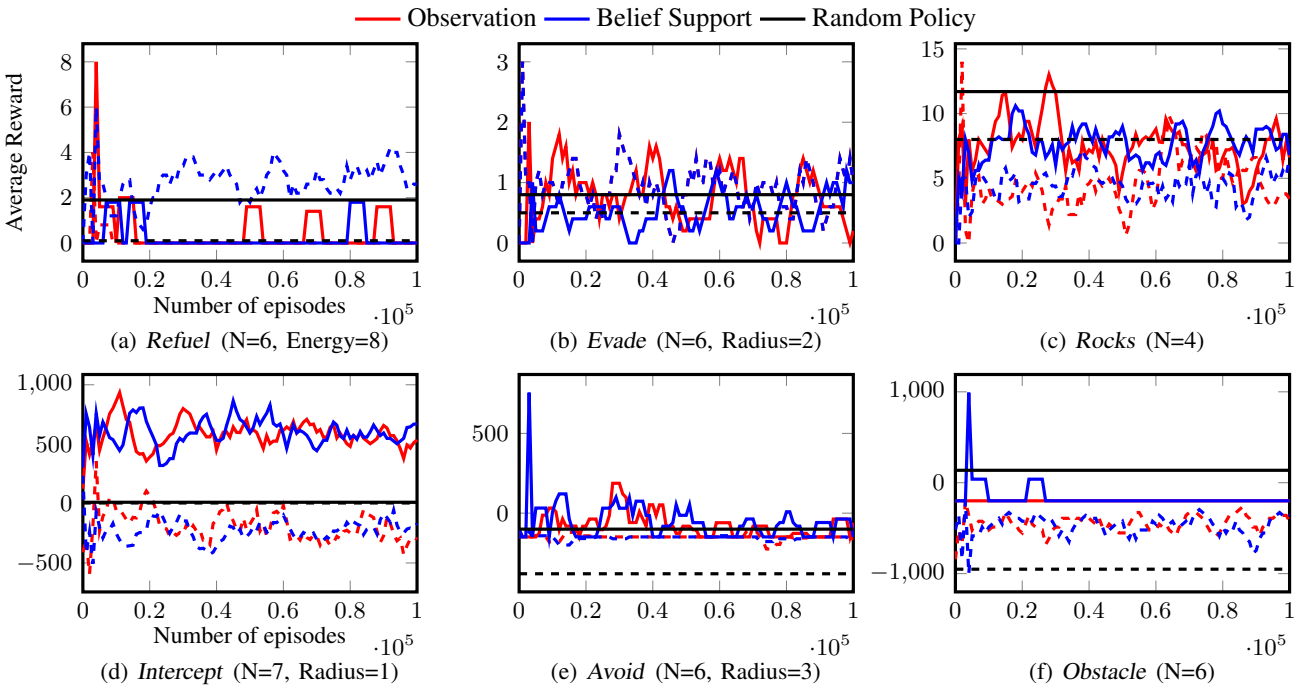


Figure 13: DDQN performed with (solid) and without (dashed) a shield restricting unsafe actions. The red lines show when the RL agent is trained using only the observations and the blue lines indicate when the RL agent is trained using some state estimation in the form of belief support. The black lines are the average reward obtained by applying a random policy.

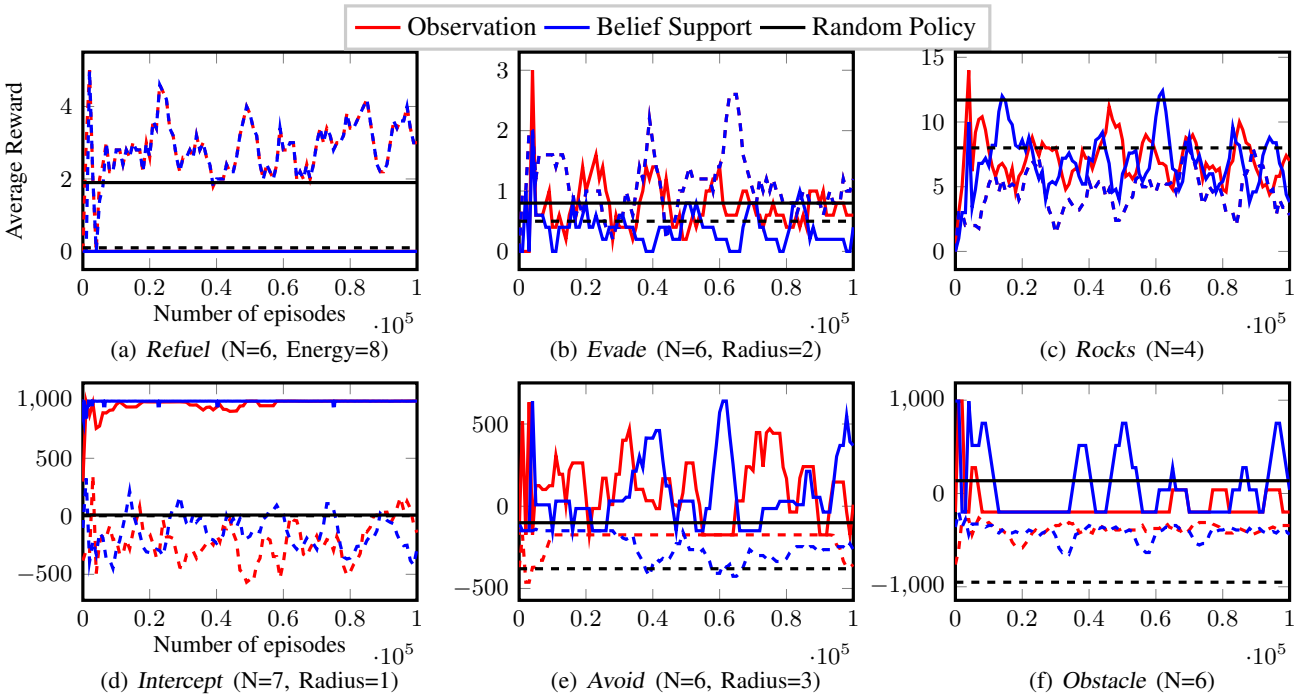
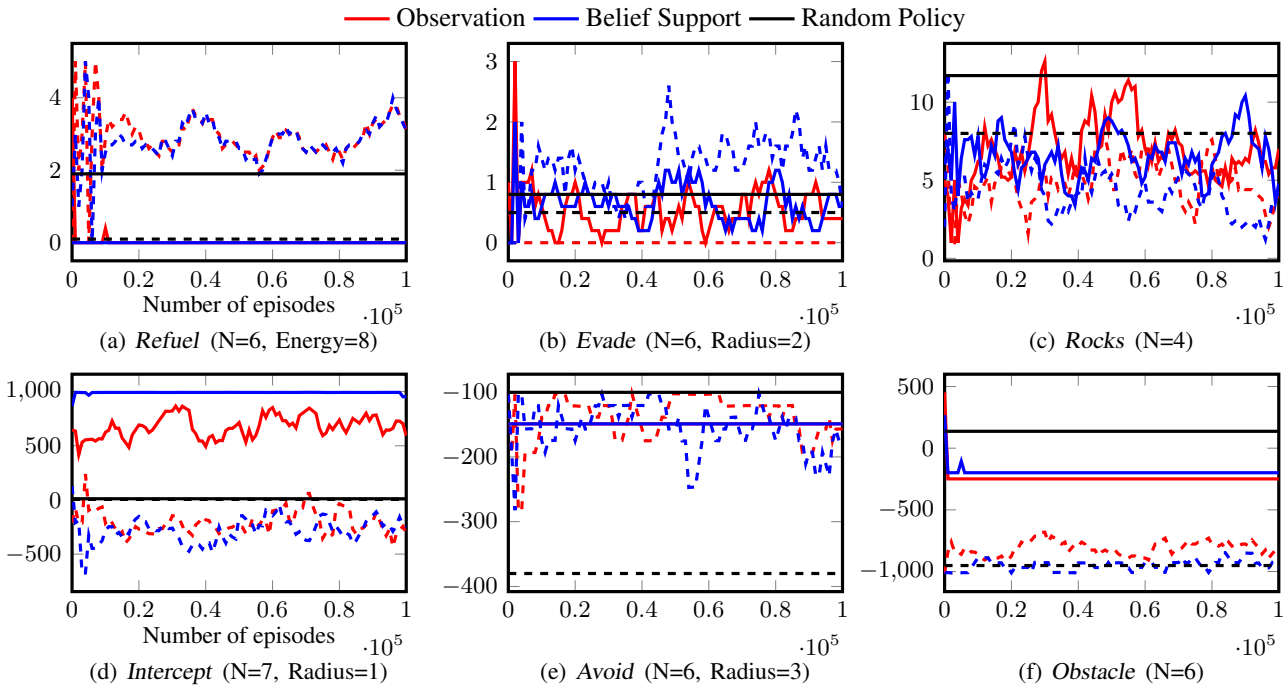


Figure 14: PPO performed with (solid) and without (dashed) a shield restricting unsafe actions. The red lines show when the RL agent is trained using only the observations and the blue lines indicate when the RL agent is trained using some state estimation in the form of belief support. The black lines are the average reward obtained by applying a random policy.



(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 15: Discrete soft-actor critic (SAC) with an LSTM architecture performed with (solid) and without (dashed) a shield restricting unsafe actions. The red lines show when the RL agent is trained using only the observations and the blue lines indicate when the RL agent is trained using some state estimation in the form of belief support.