# Enforcing Almost-Sure Reachability in POMDPs

**Sebastian Junges**
University of California
Berkeley, CA, USA
sjunges@berkeley.edu

**Nils Jansen**
Radboud University
Nijmegen, The Netherlands
n.jansen@science.ru.nl

**Sanjit A. Seshia**
University of California
Berkeley, CA, USA
sseshia@berkeley.edu

## Abstract

Partially-Observable Markov Decision Processes (POMDPs) are a well-known formal model for planning scenarios where agents operate under limited information about their environment. In safety-critical domains, the agent must adhere to a policy satisfying certain behavioral constraints. We study the problem of synthesizing policies that almost-surely reach some goal state while a set of bad states is never visited. In particular, we present an iterative symbolic approach that computes a winning region, that is, a set of system configurations such that all policies that stay within this set are guaranteed to satisfy the constraints. The approach generalizes and improves previous work in terms of scalability and efficacy, as demonstrated in the empirical evaluation. Additionally, we show the applicability to safe exploration by restricting agent behavior to these winning regions.

## 1 Introduction

Partially observable Markov decision processes (POMDPs) constitute the standard model for agents acting under partial information in uncertain environments [5, 51]. A common, but undecidable, synthesis problem is to find a policy for the agent that maximizes a reward objective [6]. In safety-critical domains, however, one seeks a policy that exhibits strict behavioral guarantees in form of specifications. We consider almost-sure reach-avoid specifications, where the probability to reach a set of *avoid* states is zero, and the probability to *reach* a set of goal states is one. The ability to solve the underlying problem is essential for satisfying temporal logic constraints [1]. Notably, one only needs to take a finite (yet exponential in the number of states) amount of memory into account, rendering the problem EXPTIME-complete [17]. Moreover, the solution does not depend on precise probabilities, but only on the underlying graph of the POMDP.

**State-of-the-art.** Chatterjee et al. [31] solve almost-sure specifications using *Boolean satisfiability solving* (SAT) [49]. Intuitively, the aim is to find a so-called simple policy that satisfies a specification but does not encompass memory-based decisions. Such a policy may not exist, yet, the method can be applied to a POMDP with an extended state space that accounts for finite memory [7, 42]. While effective, there are three shortcomings that we address in this paper. First, one needs to pre-define the memory a policy has at its disposal, as well as a fixed lookahead on the exploration of the POMDP. Second, the approach is only feasible if these bounds are small. Third, the approach finds a single simple policy starting from a pre-defined initial state. A single policy is overly restrictive, and for some applications it is desirable to build a set of policies from all states. The problem of determining any winning policy is related to strong cyclic planning, for instance using decision diagrams [11], while finding simple policies for reward minimization has been considered in, e.g., [9, 16].

**Contribution.** We overcome the aforementioned problems and provide a more scalable and versatile method to satisfy almost-sure properties: (1) We handle comparably larger environments that require memory. The method is amenable to POMDPs whose belief-support states count billions, and (2) We not only compute a single policy that satisfies the specification, but we determine a so-called *permissive policy*, that is, a set of policies. (3) Using our method, we directly construct a shield for almost-sure specifications on POMDPs which enforces at runtime that *no unsafe states are visited* and that, under mild assumptions, *the agent almost-surely reaches the set of desirable states*.

**Approach.** In a nutshell, our approach iteratively computes so-called *winning regions* (also: controllable or attracter regions) in a backward fashion. A winning region is a part of the belief space of the POMDP from which there exists a (permissive) policy to satisfy the specification. For almost-sure specifications, such regions are sufficiently captured within the belief-support MDP. The states of this finite MDP are formed by the set of supports for all possible belief states of the POMDP. Starting from the belief support states that shall be reached almost-surely, further states are added to the winning region if there exists a known policy that reaches these states without visiting those that are to avoid. We employ a symbolic SAT-based encoding that avoids an expensive explicit unfolding of POMDP executions and does not necessitate a pre-defined initial state. The key idea is to successively add short-cuts that correspond to the already known policies satisfying the specification. These changes to the structure of the POMDP are performed implicitly on the SAT encoding.

**Shielding.** If we require an agent to stay within the previously computed winning region, we guarantee the safe exploration of its environment [18, 22, 21]. More precisely, while an agent explores an environment, it is guaranteed to strictly adhere to the almost-sure specification if it acts according to the permissive policy. Such a safe exploration is essential for *safe reinforcement learning* [27, 38]. Indeed, a number of results *shield* unsafe actions of an agent via a pre-computation of permissive policies [24, 37, 41]. However, those methods do neither take partial observability into account, nor can they guarantee to reach desirable states. Nam and Alur [18] cover partial observability and reachability via active learning, but does not account for uncertainty.

**Experiments.** To showcase the feasibility of our method, we adopted a number of typical POMDP environments. We provide running times and demonstrate that our method scales better than the approach of Chatterjee et al. [31], and find that the winning regions we can compute are significantly larger. Regarding the construction of a safety shield, we let an agent explore the POMDP environment according to the permissive policy, thereby enforcing the satisfaction of the almost-sure specification. We visualize the resulting behavior of the agent in those environments with a set of videos.

**More related work.** Chatterjee et al. [32] compute winning regions for minimizing a reward objective via an explicit state representation. In [25, 29], almost-sure reachability is considered using an (explicitly) extended state space. Quantitative variants of reach-avoid specifications have gained attention in, e.g., [35, 40, 42]. Wang et al. [44] use an iterative SMT [14] approach for quantitative finite-horizon specifications. Contrary to our problem, their problem requires computing beliefs. Another line of work (e.g.,[46]) uses an idea similar to winning regions with uncertain specifications, but in a fully observable setting. Finally, complementary to shielding, there are approaches that guide reinforcement learning (without partial observability) using temporal logic constraints [39, 48, 45].

Various general POMDP approaches exist, e.g., [8, 20, 36, 19, 2, 13, 28]. The underlying approaches depend on discounted reward maximization and are able to satisfy almost-sure specifications with high reliability. However, enforcing probabilities that are close to 0 or 1 requires a discount factor close to 1, drastically reducing the scalability of the aforementioned approaches [40]. Moreover, probabilities in the underlying POMDP need to be precisely given, which is not always realistic [12].

## 2 Winning Beliefs, Winning Regions, and Shields

The support of a discrete probability distribution $\mu$ over $X$ is denoted *supp*$(\mu) = \{x \in X \mid \mu(x) > 0\}$, with $\mathit{Distr}(X)$ the set of all distributions. A *Markov decision process* (MDP) is a tuple $\mathcal{M} = \langle S, \mathrm{Act}, \mu_{\mathrm{init}}, \mathbf{P}\rangle$ with a set $S$ of states, an initial distribution $\mu_{\mathrm{init}} \in \mathit{Distr}(S)$, a finite set Act of actions, and a transition function $\mathbf{P} \colon S \times \mathrm{Act} \to \mathit{Distr}(S)$ for all $s \in S$ and $\alpha \in \mathrm{Act}$. Let $\mathrm{post}_s(\alpha) = \mathit{supp}(\mathbf{P}(s, \alpha))$ denote the states that may be the successors of the state $s \in S$ for action $\alpha \in \mathrm{Act}$ under the distribution $\mathbf{P}(s, \alpha)$. If $\mathrm{post}_s(\alpha) = \{s\}$ for all actions $\alpha$, $s$ is called *absorbing*.

A *partially observable MDP* (POMDP) is a tuple $\mathcal{P} = \langle \mathcal{M}, \Omega, \mathrm{obs}\rangle$ where $\mathcal{M} = \langle S, \mathrm{Act}, \mu_{\mathrm{init}}, \mathbf{P}\rangle$ is the underlying MDP with finite $S$, $\Omega$ is a finite set of observations, and $\mathrm{obs} \colon S \to \Omega$ is an observation function. More general observation functions $\mathrm{obs} \colon S \to \mathit{Distr}(\Omega)$ are possible via a (polynomial) reduction [32]. A path through an MDP is a sequence $\pi$, of states and actions. The observation function obs applied to a path yields a *trace*: a sequence $\text{obs}(\pi)$ of observations and actions.

A policy $\sigma\colon (S \times \text{Act})^* \times S \to \mathit{Distr}(\text{Act})$ maps a path $\pi$ to a distribution over actions. A policy is *observation-based*, if for each two paths $\pi$, $\pi'$ it holds that $\text{obs}(\pi) = \text{obs}(\pi') \Rightarrow \sigma(\pi) = \sigma(\pi')$. For POMDPs, the notion of a *belief* describes the probability of being in certain state based on an observation sequence. Formally, a belief $b$ is a distribution $b \in \mathit{Distr}(S)$ over the states. A state $s$ with positive belief $b(s)$ is in the *belief support*, $s \in \mathit{supp}(b)$.

### 2.1 Problem statement

The policy synthesis problem usually consists in finding a policy that satisfies a certain specification for a POMDP. We consider *reach-avoid* specifications, a subclass of indefinite horizon properties [50]. For a POMDP $\mathcal{P}$ with states $S$, such a specification is $\varphi = \langle \mathit{REACH}, \mathit{AVOID} \rangle \subseteq S \times S$. We assume that states in *AVOID* and in *REACH* are (made) absorbing. Let $Pr_b^\sigma(S')$ denote the probability to reach a set $S' \subseteq S$ of states from belief $b$ under the policy $\sigma$. More precisely, $Pr_b^\sigma(S')$ denotes the probability of all paths that reach $S'$ from $b$ when all nondeterminism is resolved by the policy $\sigma$.

**Definition 1** (Winning)**.** *A policy $\sigma$ is* winning *for $\varphi$ from belief $b$, iff $Pr_b^\sigma(\mathit{AVOID}) = 0$ and $Pr_b^\sigma(\mathit{REACH}) = 1$. A belief $b$ is* winning *for $\varphi$, if there exists a winning policy from $b$.*

A policy is winning if it reaches *AVOID* with probability zero and *REACH* with probability one (almost-surely) from all states within the current belief support. We formulate the decision problem.

**Problem 1:** Given a POMDP, a belief $b$, and a specification $\varphi$, decide whether $b$ is winning.

The problem is EXPTIME-complete [17]. We emphasize that this problem setting may be applied to general specifications in linear temporal logic formulas, by extending the POMDP with an automaton associated with the formula. Such a standard construction is for instance described in [47].

We now introduce *sets of winning beliefs* and state the more general problem of finding such sets.

**Definition 2** (Winning region)**.** *Let $\sigma$ be a policy. A set $W_\varphi^\sigma \subseteq \mathit{Distr}(S)$ of beliefs is a winning region for $\varphi$ and $\sigma$, if $\sigma$ is winning from each $b \in W_\varphi^\sigma$.*

We state three key observations. First, for qualitative reach-avoid specifications the belief probabilities are negligible, that is, *only the belief support is important*. Second, additionally, if a policy is winning for a belief with support $B$, *this policy is also winning for a belief whose support is contained in $B$*. Third, winning policies for individual beliefs may be composed to another winning policy that is winning for all of these beliefs, using the individual choices for each belief.

**Lemma 1.** *(1): If belief $b$ with support $\mathit{supp}(b) = B$ is winning, then belief $b'$ with support $\mathit{supp}(b') = B'$ and $B' \subseteq B$ is winning. (2): If the policies $\sigma$ and $\sigma'$ are winning for the beliefs $b$ and $b'$, respectively, then there exists a policy $\sigma''$ that is winning for both $b$ and $b'$.*

These observations allow us to formulate the following problem without depending on a concrete policy, and ultimately to provide an efficient solution.

**Problem 2:** Given a POMDP $\mathcal{P}$ and a specification $\varphi$, find a (large) winning region $W_\varphi$.

### 2.2 From winning regions to shields

We aim to define a *shield* that imposes restrictions on policies to satisfy the specification. Technically, we adapt so-called *permissive* policies [26, 33] for a belief support MDP. For a POMDP $\mathcal{P} = \langle \mathcal{M}, \Omega, \text{obs} \rangle$ with $\mathcal{M} = \langle S, \text{Act}, \mu_{\text{init}}, \mathbf{P} \rangle$, the finite state space of the belief-support MDP is given by $S^b = \big\{ B \subseteq S \mid \forall s, s' \in B \colon \text{obs}(s) = \text{obs}(s') \big\}$, that is, each state is the support of a belief state. Action $\alpha$ in $B$ leads (with an irrelevant positive probability) to a state $B'$, if there is an $s \in B$ and $s' \in B'$ such that $s' \in \text{post}_s(\alpha)$, that is, transitions between states within $B$ and $B'$ are mimicked.

A winning region can be interpreted as a subset of a belief-support MDP. To force an agent to stay within a winning region $W_\varphi$ for specification $\varphi$, we define a *$\varphi$-shield* $\nu\colon S^b \to 2^{\text{Act}}$ such that for any $B$ that is winning for $\varphi$ we have $\nu(B) \subseteq \{\alpha \in \text{Act} \mid \text{post}_B(\alpha) \subseteq W_\varphi\}$, that is, an action is part of the shield $\nu(B)$ if it exclusively leads to belief support states that are inside the winning region.

A shield restricts the set of actions an arbitrary policy may take. We call such restricted policies admissible. Specifically, let $B_\tau$ be the belief support after observing a sequence of observations $\tau$. Then policy $\sigma$ is $\nu$-admissible if $\mathit{supp}(\sigma(\tau)) \subseteq \nu(B_\tau)$ for every observation-sequence $\tau$. Consequently, a policy is not admissible if for some observation sequence $\tau$, the policy takes an action $\alpha \in \mathrm{Act}$ (formally: $\sigma(\tau)(\alpha) > 0$) which is not allowed by the shield (formally: $\alpha \notin \nu(B_\tau)$).

Some admissible policies may choose to stay in the winning region without progressing towards the *REACH* states. Such a policy adheres to the avoid-part of the specification, but violates the reachability part. To enforce *progress*, we adapt a notion from formal methods called *fairness*. A policy is fair if it takes every action infinitely often at any belief support state that appears infinitely often along a trace. For example, a policy that randomizes (arbitrarily) over all actions is fair. If $\varphi$ is a specification where $\mathit{REACH} = \emptyset$, we can drop the fairness assumption.

**Theorem 1.** *For a $\varphi$-shield, any* fair *$\varphi$-admissible policy satisfies $\varphi$.*

## 3 Iterative SAT-Based Computation of Winning Regions

In the remainder of the paper, we consider the computation of winning regions. In particular, we devise an approach for iteratively computing an increasing sequence of winning regions. The approach delivers a compact symbolic encoding of winning regions: For a belief (or belief-support) state from a given winning region, we can efficiently decide whether the outcome of an action emanating from the state stays within the winning region. This operation is essential for the functioning of a shield.

For increased modeling flexibility, we allow certain actions to be unavailable in a state (e.g., opening doors is only available when at a door), and it turned out to be crucial to handle this explicitly in the following algorithms. Technically, the transition function are a partial function, and the enabled actions are a set $\mathrm{EnAct}(s) = \{\alpha \in \mathrm{Act} \mid \mathrm{post}_s(\alpha) \neq \emptyset\}$. To ease the presentation, we assume that states $s, s'$ with the same observation share a set of enabled actions $\mathrm{EnAct}(s) = \mathrm{EnAct}(s')$.

### 3.1 One-shot approach to find small policies from a single belief (support) state

Let us first consider Problem 1, i.e., how to find a winning policy for a fixed belief support $B$. The number of policies is exponential in the actions and the exponentially many belief support states. Searching among doubly exponentially many possibilities is intractable in general. However, Chatterjee et al. [31] observe that often much simpler winning policies exist and provides a *one-shot approach* to find them. Concretely, a memoryless observation-based policy $\sigma\colon \Omega \to \mathit{Distr}(\mathrm{Act})$ is computed that is winning for (initial) belief support $B$ and an almost-sure reachability specification $\varphi$. This problem is NP-complete, and it is thus natural to encode the problem as a satisfiability query in propositional logic. We mildly extend the original encoding of winning policies [31]. It is both necessary and sufficient that the policy ensures *progress* with positive probability, which is encoded by means of a *ranking* of states, where reaching a lower ranked state means progress.

We introduce three sets of Boolean variables: $A_{z,\alpha}$, $C_s$ and $P_{s,j}$. If a policy takes action $\alpha \in \mathrm{Act}$ with positive probability upon observation $z \in \Omega$, then and only then, $A_{z,\alpha}$ is true. If under this policy a state $s \in S$ is reached from initial belief support $B$ with positive probability, then and only then, $C_s$ is true. We define a maximal rank $k$ to assure progress. For each state $s$ and rank $0 \leq j \leq k$, variable $P_{s,j}$ indicates rank $j$ for $s$, that is, a path from $s$ leads to $s' \in \mathit{REACH}$ within $j$ steps.

A winning policy is then obtained by finding a satisfiable solution (via a SAT solver) to the conjunction $\Psi^{\varphi}_{\mathcal{P}}(B,k)$ of the following constraints, where $S_? = S \setminus \mathit{AVOID} \setminus \mathit{REACH}$.

$$\bigwedge_{z\in\Omega}\ \bigvee_{\alpha\in\mathrm{EnAct}(z)} A_{z,\alpha} \quad\wedge\quad \bigwedge_{s\in B} C_s \quad\wedge\quad \bigwedge_{\substack{s\in S\\ \alpha\in\mathrm{EnAct}(s)}} C_s \wedge A_{\mathrm{obs}(s),\alpha} \rightarrow \bigwedge_{s'\in\mathrm{post}_s(\alpha)} C_{s'} \tag{1}$$

$$\bigwedge_{s\notin \mathit{REACH}} \neg P_{s,0} \quad\wedge\quad \bigwedge_{\substack{s\in S_?\\ 1\leq j\leq k}} P_{s,j} \leftrightarrow \Big( \bigvee_{\alpha\in\mathrm{EnAct}(s)} \big(A_{\mathrm{obs}(s),\alpha} \wedge \big( \bigvee_{s'\in\mathrm{post}_s(\alpha)} P_{s',j-1}\big)\big)\Big) \tag{2}$$

$$\bigwedge_{s\in \mathit{AVOID}} \neg C_s \quad\wedge\quad \bigwedge_{s\in S_?} C_s \rightarrow P_{s,k} \tag{3}$$

The conjunction in (1) ensures that in every observation, at least one action is taken, that the states in $B$ are in the set of reached states, and that this set is transitively closed under reachability (for the

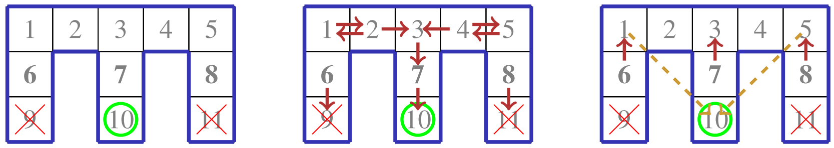

Figure 1: Example: States are cells, actions are moving in the cardinal directions (if possible), and observations are the directions with adjacent cells, e.g., the boldface states $6, 7, 8$ share an observation.

policy described by $A_{z,\alpha}$). (2) describes a ranking function. Only states in *REACH* have rank zero, and a state with positive probability to reach a state with rank $j-1$ has rank at most $j$. Finally, (3) ensures that no states in *AVOID* are reached, and that any state that is reached almost-surely reaches a state in *REACH*. By [31, Thm. 2], it holds that $\Psi_{\mathcal{P}}^{\varphi}(b, k)$ is satisfiable, if there is a memoryless observation-based policy such that $\varphi$ is satisfied. If $k = |S|$, then the reverse direction also holds. If $k < |S|$, we may miss states with a higher rank. Large $k$ values are practically intractable [31], as the encoding grows significantly with $k$. Pandey and Rintanen [43] propose to alleviate this issue by extending the SAT-solver with a dedicated handling of ranking constraints.

In order to apply this to small-memory policies, one can unfold $\log(m)$ bits of memory of such a policy into an $m$ times larger POMDP [31, 42], and then search for a memoryless policy in this larger POMDP. Chatterjee et al. [31] include a slight variation to this unfolding, allowing smaller-than-memoryless policies by enforcing the same action over various observations.

### 3.2 Incremental encoding of the winning region

We avoid the following restrictions of the one-shot approach. (1) In order to increase the likelihood of finding winning policies, we do not restrict ourselves to (very) small-memory policies, and (2) we do not have to fix a maximal rank $k$. These modifications allow us to find more winning policies, without guessing hyper-parameters. As we do not need to fix the belief-state, those parts of the winning region that are easy to find for the solver are encountered first. This optimism of finding small and easy strategies first is beneficial for the performance, but does not restrict the generality.

**Idea.** The key idea is that we iteratively add short-cuts that represent known winning policies. We find some winning policy $\sigma$ for some belief states in the first iteration, and then add a fresh action to all (original) POMDP states: This action leads to a *REACH* state, if the state is part of a wining belief-support under policy $\sigma$, and an *AVOID* state otherwise. Adding this action does not change winning regions in the POMDP (see Lemma 1), but extends the belief support states that may win by means of a memoryless policy. Instead of actually adjusting the POMDP, we realize this idea directly on the encoding. We find winning states based on a solution of the encoding, and instead of adding actions, we allow the solver to decide to follow an individual policy from each observation.

*Example.* Consider the small Cheese-POMDP [3] in Fig. 1(left) with *REACH* $= \{10\}$ and *AVOID* $= \{9, 11\}$. From belief support $B = \{6, 8\}$ there is no memoryless winning policy—In states $\{6, 8\}$ we have to go north, which prevents us from going south in state 7. However, we can find a memoryless winning policy for $\{1, 5\}$, see Fig. 1 (center). If we add shortcuts, we can now find a memoryless winning policy for $B = \{6, 8\}$, depicted in Fig. 1 (right).

**Encoding.** First, we use unbounded rather than Boolean variables for the ranking [23]. This relaxation avoids the growth of the encoding and admits arbitrarily large ranks with a fixed-size encoding. Empirically, using difference logic works well. This logic is an extension to propositional logic that can be checked using a satisfiability-modulo-theories (SMT) solver [15]. We use a data structure Win such that $\mathsf{Win}(z)$ encodes all winning belief supports with observation $z$. Internally, the data structure stores maximal winning belief supports as bit-vectors. By construction, for every $B \in \mathsf{Win}(z)$ a winning region exists, i.e., conceptually, there is a shortcut-action leading to *REACH*.

Our encoding represents an observation-based policy that can decide to take a shortcut, which means that it follows a previously computed winning policy from there (using Lemma 1). In addition to $A_{z,\alpha}$ and $C_s$ from the previous encoding, we use the following variables: The ranking of state $s$ is now encoded using a real-valued $R_s$. Furthermore, we take shortcuts in states $s$ where $D_s$ is true. For each observation, we must take the same shortcut, referred to by a positive integer-valued index $I_z$. The

**Algorithm 1** Naive Iterative Computation of the Winning Region

**Input**: POMDP $\mathcal{P}$, reach-avoid specification $\varphi$, **Output**: Winning region encoded in Win
$\mathsf{Win}(z) \leftarrow \{s \in \mathit{REACH} \mid \text{obs}(s) = z\}$ for all $z \in \Omega$
$\Phi \leftarrow \mathit{Encode}(\mathcal{P}, \varphi, \mathsf{Win})$ ▷ Create encoding as outlined (4)–(8).
**while** $\exists \nu$ s.t. $\nu \models \Phi$ **do** ▷ Call an SMT solver
  $\mathsf{Win}(z) \leftarrow \mathsf{Win}(z) \cup \{B \mid s \in B \text{ iff } \nu(C_s) = \mathtt{true}\}$ for all $z \in \Omega$
  $\Phi \leftarrow \mathit{Encode}(\mathcal{P}, \varphi, \mathsf{Win})$

policy may decide to follow a shortcut *after* we take an action starting in a state with observation $z$. If $F_z$ is true, then after taking some action from $z$-states, we want to take a shortcut. Finally, a variable $U_z \in \mathbb{B}$ encodes if the policy is winning in a belief support that is not yet in $\mathsf{Win}(z)$.

The conjunction of the following constraints yields the encoding $\Phi^{\varphi}_{\mathcal{P}}(\mathsf{Win})$:

$$\bigwedge_{z\in\Omega} \bigvee_{\alpha\in\text{EnAct}(z)} A_{z,\alpha} \quad \wedge \quad \bigwedge_{s\in\mathit{AVOID}} \neg C_s \wedge \neg D_s \tag{4}$$

$$\bigwedge_{\substack{s\in S\\ \alpha\in\text{EnAct}(s)\\ z=\text{obs}(s)}} \Big(C_s \wedge A_{\text{obs}(s),\alpha} \wedge \neg F_{z(s)} \rightarrow \bigwedge_{s'\in\text{post}_s(\alpha)} C_{s'}\Big) \wedge \Big(C_s \wedge A_{z,\alpha} \wedge F_z \rightarrow \bigwedge_{s'\in\text{post}_s(\alpha)} D_{s'}\Big) \tag{5}$$

$$\bigwedge_{s\in S_?} C_s \rightarrow \Big(\bigvee_{\alpha\in\text{EnAct}(s)} \big(A_{\text{obs}(s),\alpha} \wedge (\bigvee_{s'\in\text{post}_s(\alpha)} R_s > R_{s'})\big) \vee F_{z(s)}\Big) \tag{6}$$

$$\bigwedge_{s\in S} D_s \rightarrow I_{\text{obs}(s)} > 0 \quad \wedge \quad \bigwedge_{z\in\Omega} I_z \leq |\mathsf{Win}(z)| \quad \wedge \quad \bigwedge_{\substack{z\in\Omega\\ 0\leq i\leq|\mathsf{Win}(z)|}} \bigwedge_{\substack{s\in S\setminus\mathsf{Win}(z)_i\\ \text{obs}(s)=z}} I_z \neq i \vee \neg D_s \tag{7}$$

$$\bigvee_{z\in\Omega} U_z \quad \wedge \quad \bigwedge_{\substack{z\in\Omega\\ \mathsf{Win}(z)=\emptyset}} \Big(U_z \leftrightarrow \bigvee_{\substack{s\in S\\ \text{obs}(s)=z}} C_s\Big) \quad \wedge \quad \bigwedge_{\substack{z\in\Omega\\ \mathsf{Win}(z)\neq\emptyset}} \Big(U_z \leftrightarrow \bigvee_{X\in\mathsf{Win}(z)} \bigwedge_{\substack{s\in S\setminus X\\ \text{obs}(s)=z}} \neg C_s\Big) \tag{8}$$

Similar to before, we select at least one action and avoid states should not be reached (4). States that are reached are closed under the transitive closure, but now only if we do not switch in these states. Furthermore, we need to have a policy from the states reached after switching (5). The ranking function is updated: If we switch to an existing policy, that ensures that we reach the target, and we search for a path to the goal via a strictly descending chain of ranks (6). If we reach a state $s$ under the assumption that we follow a previously computed policy from $s$, then we must pick a policy that is winning for $s$ (7). Finally, (8) ensures extending the winning region with at least one belief support. For an observation which has no winning belief support yet, finding a policy from any state within this belief support updates the winning region. For other observations, it means finding a winning policy for a belief support that is not subsumed by a previous one.

**Naive algorithm.** Algorithm 1 starts with a winning region consisting of reach states. We encode the problem as above and use an SMT solver to find a new winning policy extending the winning region, and iterate until we find no further policy. We update $\mathsf{Win}(z)$ if the solver indicates that the winning region is extended. In each iteration, Win contains a winning region. When we find no more policies on the (conceptually) extended POMDP, we terminate.

**Theorem 2.** *If $\nu \models \Phi^{\varphi}_{\mathcal{P}}(\mathsf{Win})$, then $B_z = \{s \mid \nu(C_s) = \mathtt{true}, \text{obs}(s) = z\}$ is a winning belief support. Consequently, in any iteration, Algorithm 1 computes a winning region.*

The algorithm always terminates because the set of winning regions is finite while it does not necessarily find the maximal winning region. Formally, the winning region is the greatest fixpoint and we iterate from below. However, iterating from above requires to reason that none of the doubly-exponentially many policies is winning for a particular belief support state; whereas our approach profits from finding simple strategies early on. Unfolding of memory as discussed earlier also makes this algorithm complete, yet suffering from the same blow-up. A main advantage is that the algorithm often avoids the need for unfolding when searching for a winning policy or large winning regions.

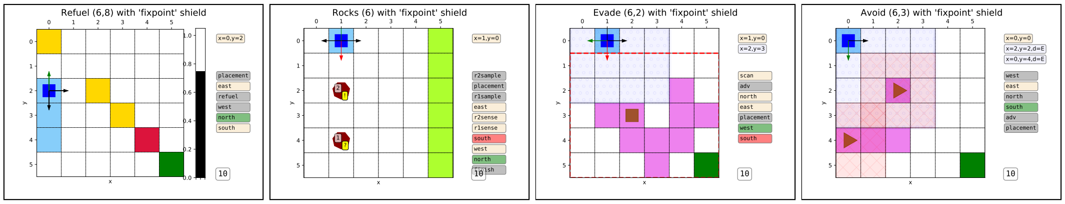


Figure 2: Video stills from simulating a shielded agent on four different benchmarks. For the actual videos, we refer to `https://github.com/sjunges/shielding-POMDPs`.

**Optimized algorithm.** We improve the naive algorithm along the following four lines: First, before updating the winning region with a policy, we aim to extend the policy as much as possible, i.e., we want to find more states with the same observation that are winning under the same policy. Therefore, we fix the variables for action choices that yield a new winning policy, and search whether we can extend this policy by finding more states and actions that are compatible with the policy. Second, we observe that large parts of the encoding stay intact, and use an incremental approach in which we first push all the constraints from the POMDP onto the stack, then all the constraints from the winning region, and finally a constraint that asks for progress. After we found a new policy, we pop the last constraint from the stack, add new constraints regarding the winning region (notice that the old constraints remain intact), and push new constraints that ask for extending the winning region to the stack. We refresh the encoding periodically to avoid an unnecessary cluttering. Third, we employ a graph-based preprocessing on the POMDP to reduce the number of SMT invocations. The first step is to find all states that violate the specification, and make them absorbing. Then, observations are determined that (according to the current winning region) are only associated with states that have winning policies. This procedure starts, as before, with a winning region that consists only of reach states. We iteratively update this region by adding states sharing a common observation $z$, where from every state with observation $z$ there is an action that leads to the current winning region within one step. Fourth, we add constraints that allow a switch to a known policy immediately, that is, not after executing an action first. These constraints yield a faster convergence of the algorithm.

## 4 Empirical Evaluation

We evaluate our iterative approach to find winning policies or large winning regions, as introduced in Section 3.2, against our adaption and implementation of the one-shot approach [31] (Section 3.1).

**Set-up.** We implemented both the one-shot algorithm from and our iterative algorithm, on top of the model-checker STORM [34] and using the SMT solver Z3 [14]. We consider two variants of the iterative algorithm. (1, for finding *fixpoints*): The optimized algorithm as described above. (2, for finding a policy from the *initial* state): The algorithm as before, but any outer iteration starts with an SMT-check to see whether we find a policy covering the initial states. We realize the latter by fixing (temporarily) the $C_s$-variables. In the first iteration, this configuration and its resulting policy closely resemble the one-shot approach. For the one-shot algorithm, we apply the novel graph-based preprocessing to identify more winning observations. We (manually, a-priori) search for the *optimal parameters*: each instance for the smallest amount of memory possible, and for the smallest maximal rank (subject to being a multiplicative of five) that yields a result. Guessing parameters as an "oracle" is hard and time-consuming. Therefore, we also investigate the performance of the one-shot algorithm by *fixing the parameters* to two memory-states and a maximal rank of 30. We picked these parameters as they provide results for most benchmarks. We use a MacBook Pro MV962LL/A with a single core, no randomization, and never hit the memory limit of 4GB. The time-out (TO) is 15 minutes.

**Benchmarks.** Our benchmarks involve agents operating in $N \times N$ grids, inspired by, e.g., [29, 31, 10, 4, 30]. While our approaches work for general POMDPs, we focus on grid worlds to enable a suitable visualization, see Fig. 2 for video stills of simulating the following benchmarks. *Refuel* concerns a rover that shall travel from one corner to the other, while avoiding an obstacle on the diagonal. Every movement costs energy and the rover may recharge at recharging stations to its full battery capacity $E$. It receives noisy information about its position and battery level. *Rocks* is a variant of *rock sample*. The grid contains two rocks which are either valuable or dangerous to collect.

| | | | *Evade* $(N,R)$ | | *Avoid* $(N,R)$ | | *Intercept* $(N,R)$ | | *Obstacle* $(N)$ | | *Rocks* $(N)$ | | *Refuel* $(N,E)$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Inst. | 6,2 | 7,2 | 6,3 | 7,4 | 7,1 | 7,2 | 6 | 8 | 4 | 6 | 6,8 | 7,7 |
| | | $\lvert S\rvert$ | 4232 | 8108 | 5976 | 13021 | 4705 | 4705 | 37 | 65 | 331 | 816 | 270 | 302 |
| | | #Tr | 28866 | 57570 | 14373 | 33949 | 18049 | 18049 | 224 | 421 | 3484 | 7292 | 1301 | 1545 |
| | | $\lvert\Omega\rvert$ | 2202 | 4172 | 3300 | 8584 | 2002 | 2598 | 4 | 4 | 65 | 74 | 36 | 35 |
| | | $\lvert S^b\rvert$ | 1.1E8 | 4.4E11 | 1.1E15 | 2.9E17 | 6.4E10 | 2.7E9 | 1.1E9 | 2.9E17 | 3.5E5 | 7.7E25 | 5.6E14 | 7.4E19 |
| **iterative** | *fixpoint* | Time | 142 | 613 | 167 | 745 | 116 | 86 | 2 | 30 | 19 | 753 | 6 | 3 |
| | | #Iter. | 4 | 6 | 3 | 4 | 8 | 8 | 68 | 150 | 36 | 284 | 40 | 30 |
| | | #solve | 681 | 1129 | 629 | 1027 | 1171 | 976 | 839 | 4291 | 1702 | 13650 | 1023 | 528 |
| | | $\lvert W\rvert$ | 1.0E8 | 4.2E11 | 1.1E15 | 2.9E17 | 9.2E4 | 2.9E4 | 4.1E7 | 3.8E14 | 3.5E5 | 7.7E25 | 1.2E11 | 2.1E8 |
| | *initial* | Time | 49 | 576 | 10 | 40 | 11 | 2 | <1 | <1 | 17 | 226 | 2 | 2 |
| | | #Iter. | 1 | 1 | 1 | 1 | 2 | 1 | 10 | 12 | 29 | 65 | 2 | 4 |
| | | #solve | 1 | 1 | 1 | 1 | 81 | 1 | 114 | 229 | 1215 | 2652 | 62 | 80 |
| | | $\lvert W\rvert$ | 5.0E7 | 1.0E11 | 3.7E5 | 6.9E10 | 6.2E3 | 2.1E3 | 4.1E5 | 4.5E9 | 4.4E4 | 1.8E13 | 8.4E6 | 3.7E4 |
| **1-shot** | *opt* | Time | 12 | 270 | 22 | 53 | 8 | 1 | 2 | 195 | 120 | TO | 2 | <1 |
| | | Mem,k | 1,20 | 1,30 | 1,30 | 1,25 | 2,10 | 1,10 | 4,15 | 5,50 | 2,10 | ? | 2,15 | 2,15 |
| | *fix* | Time | TO | TO | TO | TO | 28 | 18 | N/A | N/A | TO | TO | 11 | 37 |

Table 1: For each benchmark instance (columns), we report the name and relevant characteristics: the number of states ($|S|$), the number of transitions (#Tr, the edges in the graph described by the POMDP), the number of observations ($|\Omega|$), and the size of the belief state ($|S^b|$). For the computation of winning regions, we provide the run time (Time, in seconds), the number of iterations (#Iter.), and the number of invocations of the SMT solver (#solve), and the approximate size of the winning region ($|W|$). We then report these numbers when searching for a policy that wins from the initial state. For the 1-shot method, we provide the time for the optimal parameters (on the next line), and the time for the preset parameters, or N/A if no policy could be found with these parameters.

To find out with certainty, the rock has to be sampled from an adjacent field. The goal is to collect a valuable rock, bring it to the drop-off zone, and not collect dangerous rocks. *Evade* is a scenario where a robot needs to reach a destination and evade a faster agent. The robot has a limited range of vision ($R$), but may scan the whole grid instead of moving. A certain safe area is only accessible by the robot. *Avoid* is a related scenario where a robot shall keep distance to patrolling agents. These agents move with uncertain speed, yielding partial information about their position, but the robot may exploit their predefined routes. Details on *Intercept* and *Obstacle* are in the supplementary materials.

**Results.** Tab. 1 details the numerical benchmark results. The iterative algorithm finds winning policies *without guessing parameters* and is often *faster* versus the one-shot method with an oracle providing optimal parameters, and significantly faster versus the one-shot approach with reasonably fixed parameters. The iterative algorithm finds winning regions with billions of (belief support) states. The number of iterations reflects the number of intermediate policies used to add (multiple) shortcuts.

Moreover, we let a *shielded agent* move randomly through the grid-worlds. As the shield is correct by construction, all simulation runs indeed never reach an avoid state, and eventually reach the target (albeit after many steps). Videos (Fig. 2) of these runs allow us to visualize how the shield allows more freedom in the choice of action for larger winning regions. All approaches, including the one-shot method, allow for a certain permissiveness due to their symbolic nature. Winning regions obtained from running the iterative procedure to a fixpoint, however, are significantly larger (cf. the table), and allow for a much higher degree of permissiveness (cf. the videos).

The videos are available at `https://github.com/sjunges/shielding-POMDPs`.

## 5 Conclusion

We provided an iterative approach to find POMDP policies satisfying almost-sure reachability specifications. The significantly inmproved scalability is demonstrated on a string of benchmarks.Our approach exclusively allows to shield agents and guarantee that any exploration of an environment satisfies the specification, without needlessly restricting the freedom of the agent. While we demonstrate the effectiveness of our approach in terms of random exploration of an environment, we plan to investigate a tight interaction with state-of-the-art reinforcement learning.

## References


[1] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977.

[2] Tommi S. Jaakkola, Satinder P. Singh, and Michael I. Jordan. Reinforcement learning algorithm for partially observable markov decision problems. In *NIPS*, pages 345–352. MIT Press, 1994.

[3] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *ICML*, pages 362–370. Morgan Kaufmann, 1995.

[4] Thomas G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *ICML*, pages 118–126. Morgan Kaufmann, 1998.

[5] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998.

[6] 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.

[7] Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R Cassandra. Solving POMDPs by searching the space of finite policies. In *UAI*, pages 417–426. Morgan Kaufmann Publishers Inc., 1999.

[8] Milos Hauskrecht. Value-function approximations for partially observable Markov decision processes. *J. Artif. Intell. Res.*, 13:33–94, 2000.

[9] Pascal Poupart and Craig Boutilier. Bounded finite state controllers. In *NIPS*, pages 823–830. MIT Press, 2003.

[10] Trey Smith and Reid Simmons. Heuristic search value iteration for POMDPs, 2004.

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

[12] Brendan Burns and Oliver Brock. Sampling-Based Motion Planning with Sensing Uncertainty. In *ICRA*, pages 3313–3318. IEEE, 2007.

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

[14] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[15] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. IOS Press, 2009.

[16] Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Optimizing fixed-size stochastic controllers for POMDPs and decentralized pomdps. *Auton. Agents Multi Agent Syst.*, 21(3):293–320, 2010.

[17] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Qualitative analysis of partially-observable markov decision processes. In *MFCS*, volume 6281 of *Lecture Notes in Computer Science*, pages 258–269. Springer, 2010.

[18] Wonhong Nam and Rajeev Alur. Active learning of plans for safety and reachability goals with partial observability. *IEEE Trans. Syst. Man Cybern. Part B*, 40(2):412–420, 2010.

[19] David Silver and Joel Veness. Monte-carlo planning in large POMDPs. In *NIPS*, pages 2164–2172, 2010.

[20] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[21] Anayo K. Akametalu, Shahab Kaynama, Jaime F. Fisac, Melanie Nicole Zeilinger, Jeremy H. Gillula, and Claire J. Tomlin. Reachability-based safe learning with Gaussian processes. In *CDC*, pages 1424–1431. IEEE, 2014.

[22] Martin Pecka and Tomás Svoboda. Safe exploration techniques for reinforcement learning - an overview. In *MESAS*, volume 8906 of *Lecture Notes in Computer Science*, pages 357–375. Springer, 2014.

[23] Ralf Wimmer, Nils Jansen, Erika Ábrahám, Joost-Pieter Katoen, and Bernd Becker. Minimal counterexamples for linear-time probabilistic verification. *Theor. Comput. Sci.*, 549:61–100, 2014.

[24] Roderick Bloem, Bettina Könighofer, Robert Könighofer, and Chao Wang. Shield synthesis: - runtime enforcement for reactive systems. In *TACAS*, volume 9035 of *LNCS*, pages 533–548. Springer, 2015.

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

[26] 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.

[27] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16:1437–1480, 2015.

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

[29] María Svorenová, Martin Chmelik, Kevin Leahy, Hasan Ferit Eniser, Krishnendu Chatterjee, Ivana Cerná, and Calin Belta. Temporal logic motion planning using POMDPs with parity objectives: case study paper. In *HSCC*, pages 233–238. ACM, 2015.

[30] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *CoRR*, abs/1606.01540, 2016.

[31] 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.

[32] Krishnendu Chatterjee, Martin Chmelik, Raghav Gupta, and Ayush Kanodia. Optimal cost almost-sure reachability in POMDPs. *Artif. Intell.*, 234:26–48, 2016.

[33] Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen. Safety-constrained reinforcement learning for MDPs. In *TACAS*, volume 9636 of *LNCS*, pages 130–146. Springer, 2016.

[34] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *CAV (2)*, volume 10427 of *LNCS*, pages 592–600. Springer, 2017.

[35] Gethin Norman, David Parker, and Xueyi Zou. Verification and control of partially observable probabilistic systems. *Real-Time Systems*, 53(3):354–402, 2017.

[36] Erwin Walraven and Matthijs T. J. Spaan. Accelerated vector pruning for optimal POMDP solvers. In *AAAI*, pages 3672–3678. AAAI Press, 2017.

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

[38] Nathan Fulton and André Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *AAAI*, pages 6485–6492. AAAI Press, 2018.

[39] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Logically-correct reinforcement learning. *CoRR*, abs/1801.08099, 2018.

[40] Karel Horák, Branislav Bosanský, and Krishnendu Chatterjee. Goal-HSVI: Heuristic search value iteration for goal POMDPs. In *IJCAI*, pages 4764–4770. ijcai.org, 2018.

[41] Nils Jansen, Bettina Könighofer, Sebastian Junges, and Roderick Bloem. Shielded decision-making in MDPs. *CoRR*, abs/1807.06096, 2018.

[42] Sebastian Junges, Nils Jansen, Ralf Wimmer, Tim Quatmann, Leonore Winterer, Joost-Pieter Katoen, and Bernd Becker. Finite-state controllers of POMDPs using parameter synthesis. In *UAI*, pages 519–529. AUAI Press, 2018.

[43] Binda Pandey and Jussi Rintanen. Planning for partial observability by SAT and graph constraints. In *ICAPS*, pages 190–198. AAAI Press, 2018.

[44] Yue Wang, Swarat Chaudhuri, and Lydia E. Kavraki. Bounded policy synthesis for POMDPs with safe-reachability objectives. In *AAMAS*, pages 238–246. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[45] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. In *TACAS (1)*, volume 11427 of *Lecture Notes in Computer Science*, pages 395–412. Springer, 2019.

[46] Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration for interactive machine learning. In *NeurIPS*, pages 2887–2897, 2019.

[47] Maxime Bouton, Jana Tumova, and Mykel J. Kochenderfer. Point-based methods for model checking in partially observable markov decision processes. *CoRR*, abs/2001.03809, 2020.

[48] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Cautious reinforcement learning with logical constraints. *CoRR*, abs/2002.12156, 2020.

[49] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

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

[51] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.