Shielded Reinforcement Learning for Hybrid Systems

Asger Horn Brorholt, Peter Gjøl Jensen, Kim Guldstrand Larsen, Florian Lorber, and Christian Schilling

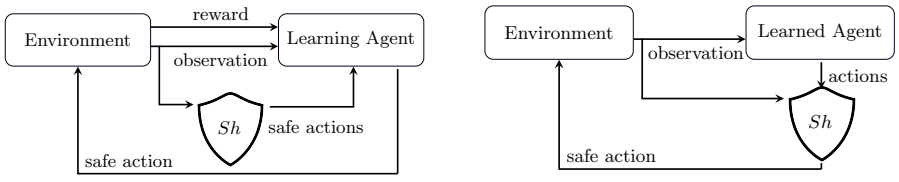
Department of Computer Science, Aalborg University, Aalborg, Denmark
{asgerhb,pgj,kgl,florber,christianms}@cs.aau.dk

Abstract. Safe and optimal controller synthesis for switched-controlled hybrid systems, which combine differential equations and discrete changes of the system's state, is known to be intricately hard. Reinforcement learning has been leveraged to construct near-optimal controllers, but their behavior is not guaranteed to be safe, even when it is encouraged by reward engineering. One way of imposing safety to a learned controller is to use a *shield*, which is correct by design. However, obtaining a shield for non-linear and hybrid environments is itself intractable. In this paper, we propose the construction of a shield using the so-called *barbaric method*, where an approximate finite representation of an underlying partition-based two-player safety game is extracted via systematically picked samples of the true transition function. While hard safety guarantees are out of reach, we experimentally demonstrate strong statistical safety guarantees with a prototype implementation and Uppaal Stratego. Furthermore, we study the impact of the synthesized shield when applied as either a pre-shield (applied before learning a controller) or a post-shield (only applied after learning a controller). We experimentally demonstrate superiority of the pre-shielding approach. We apply our technique on a range of case studies, including two industrial examples, and further study post-optimization of the post-shielding approach.

1 Introduction

Digital controllers are key components of cyber-physical systems. Unfortunately, the algorithmic construction of controllers is intricate for any but the simplest systems [37,21]. This motivates the usage of reinforcement learning (RL), which is a powerful machine-learning method applicable to systems with complex and stochastic dynamics [12].

However, while controllers obtained from RL provide near-optimal average-case performance, they do not provide guarantees about worst-case performance, which limits their application in many relevant but safety-critical domains, ranging from power converters to traffic control [45,41]. A typical way to tackle this challenge is to integrate safety into the optimization objective via *reward shaping* during the learning phase, which punishes unsafe behavior [23]. This will make the controller more robust to a certain degree, but safety violations will still be



(a) Pre-shield. (b) Post-shield.

Fig. 1: Pre- and post-shielding in a reinforcement-learning setting.

possible, and the integration of safety into the optimization objective can reduce the performance, thus yielding a controller that is neither safe nor optimal.

A principled approach to obtain worst-case guarantees is to use a *shield* that restricts the available actions [9]. This makes it possible to construct correct-by-design and yet near-optimal controllers. Fig. 1 depicts two ways of shielding RL agents: *pre-* and *post-shielding*. Pre-shielding is already applied during the learning phase, and the learning agent receives only safe actions to choose from. Post-shielding is only applied during deployment, where the trained agent is monitored and, if necessary, corrected. Such interventions to ensure safety interfere with the learned policy of the agent, potentially causing a loss in optimality.

In a nutshell, the algorithm to obtain a shield works as follows. First we compute a finite partitioning of the state space and approximate the transitions between the partitions. This results in a two-player safety game, and upon solving it, we obtain a strategy that represents the most permissive shield.

Cyber-physical systems exhibit behavior that is both continuous (the environment) and discrete (the control, and possibly the environment too). We are particularly interested in a class of systems we refer to as *hybrid Markov decision processes* (HMDPs). In short, these are control systems where the controller can choose an action in a periodic manner, to which the environment chooses a stochastic continuous trajectory modeled by a stochastic hybrid automaton [17]. While HMDPs represent many real-world systems, they are a rich extension of hybrid automata, and thus their algorithmic analysis is intractable even under serious restrictions [26]. These complexity barriers unfortunately also carry over to the above problem of constructing a shield.

In this paper, we propose a new practical technique to automatically and robustly synthesize a shield for HMDPs. The intractability in the shield-synthesis algorithm is due to the rigorous computation of the transition relation in the abstract transition system, since that computation reduces to the (undecidable) reachability problem. Our key to get around this limitation is to approximate the transition relation through systematic sampling, in a way that is akin to the *barbaric method* (a term credited to Oded Maler [30,19]).

We combine our technique with the tool Uppaal Stratego to learn a shielded near-optimal controller, which we evaluate in a series of experiments on several models, including two real-world cases. In our experiments we also find that pre-shielding outperforms post-shielding. While the shield obtained

through our technique is not guaranteed to be safe in general due to the approximation, we demonstrate that the controllers we obtain are statistically safe, and that a moderate number of samples is sufficient in practice.

Related work. Enforcing safety during RL by limiting the choices available to the agent is a known concept, which is for instance applied in the tool UPPAAL STRATEGO [18]. The term "shielding" was coined by Bloem et al. [9], who introduced special conditions on the enforcer like *shields with minimal interference* and *k-stabilizing shields* and later demonstrated shielding for RL agents [3], where they correct potentially unsafe actions chosen by the RL agent. Jansen et al. [29] introduced shielding in the context of RL for probabilistic systems. A concept similar to shielding has also been proposed for safe model predictive control [6,47]. Carr et al. [13] show how to shield partially observable environments. In a related spirit, Maderbacher et al. start from a safe policy and switch to a learned policy if safe at run time [39].

(Pre-)Shielding requires a model of the environment in order to provide safety guarantees during learning. Orthogonal to shielding, several model-free approaches explore an RL environment in a *safer* way, but without any guarantees. Several works are based on barrier certificates and adversarial examples [14,38] or Lyapunov functions [25]. Similarly, Berkenkamp et al. describe a method to provide a safe policy with high probability [7]. Chow et al. consider a relaxed version of safety based on expected cumulative cost [15]. In contrast to these model-free approaches, we assume a model of the environment, which allows us to safely synthesize a shield just from simulations before the learning phase. We believe that the assumption of a model, typically derived from first principles, is realistic, given that our formalism allows for probabilistic modeling of uncertainties. To the best of our knowledge, none of the above works can be used in practice for safe RL in the complex class of HMDPs.

Larsen et al. [35] used a set-based Euler method to overapproximate reachability for continuous systems. This method was used to obtain a safety strategy and a safe near-optimal controller. Contrary to that work, we apply both pre- and post-shielding, and our method is applicable to more general hybrid systems. We employ state-space partitioning, which is common for control synthesis [40] and reachability analysis [32] and is also used in recent work on learning a safe controller for discrete stochastic systems in a teacher-learner framework [46]. Contemporary work by Badings et al. [5] also uses a finite state-space abstraction along with sample-based reachability estimation, to compute a reach-avoid controller. The method assumes linear dynamical systems with stochastic disturbances, to obtain upper and lower bounds on transition probabilities. In contrast, our method supports a very general hybrid simulation model, and provides a safety shield, which allows for further optimization of secondary objectives.

A special case of the HMDPs we consider is the class of stochastic hybrid systems (SHSs). Existing reachability approaches are based on state-space partitioning [2,42], which we also employ in this work, or have a statistical angle [11]. We are not aware of any works that extended SHSs to HMDPs.

Outline. The remainder of the paper is structured as follows. In Section 2 we present the formalism we use. In Section 3 we present our synthesis method to obtain a safety strategy and explain how this strategy can be integrated into a shield. We demonstrate the performance of our pre- and post-shields in several cases in Section 4. Finally we conclude the paper in Section 5.

2 Euclidian and Hybrid Markov Decision Processes

In this section we introduce the system class we study in this paper: hybrid Markov decision processes (HMDPs). They combine Euclidean Markov decision processes and stochastic hybrid automata, which we introduce next. HMDPs model complex systems with continuous, discrete and stochastic dynamics.

Euclidean Markov Decision Processes A Euclidean Markov decision process (EMDP) [28,27] is a continuous-space extension of a Markov decision process (MDP). We recall its definition below.

Definition 1 (Euclidean Markov decision process). *A* Euclidean Markov decision process *of dimension k is a tuple $\mathcal{M} = (\mathcal{S}, s_0, Act, T, C, \mathcal{G})$ where*

- *$\mathcal{S} \subseteq \mathbb{R}^k$ is a bounded and closed part of k-dimensional Euclidean space,*
- *$s_0 \in \mathcal{S}$ is the initial state,*
- *Act is the finite set of actions,*
- *$T : \mathcal{S} \times Act \to (\mathcal{S} \to \mathbb{R}_{\geq 0})$ maps each state-action pair (s, a) to a probability density function over $\mathcal{S}$, i.e., we have $\int_{s' \in \mathcal{S}} T(s, a)(s')ds' = 1$,*
- *$C : \mathcal{S} \times Act \times \mathcal{S} \to \mathbb{R}$ is the cost function, and*
- *$\mathcal{G} \subseteq \mathcal{S}$ is the set of goal states.*

Example 1 (Random walk). Fig. 2 illustrates an EMDP of a (semi-)random walk on the state space $\mathcal{S} = [0, x_{max}] \times [0, t_{max}]$ (one-dimensional space plus time). The goal is to cross the $x = 1$ finishing line before $t = 1$. Two movement actions are available: fast and expensive (blue), or slow and cheap (brown). Both actions have uncertainty about the distance traveled and time taken. Given a state (x, t) and an action $a \in \{slow, fast\}$, the next-state density function $T((x, t), a)$ is a uniform distribution over the successor-state set $(x + d_x(a) \pm \epsilon) \times (t + d_t(a) \pm \epsilon)$, where $d_x(a)$ and $d_t(a)$ respectively represent the direction of movement in space and time given action a, while ϵ models the uncertainty. ◁

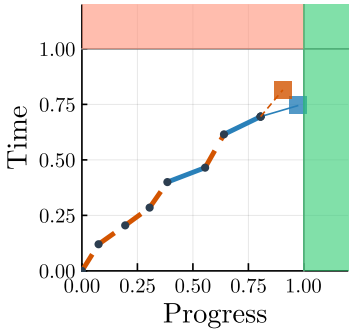
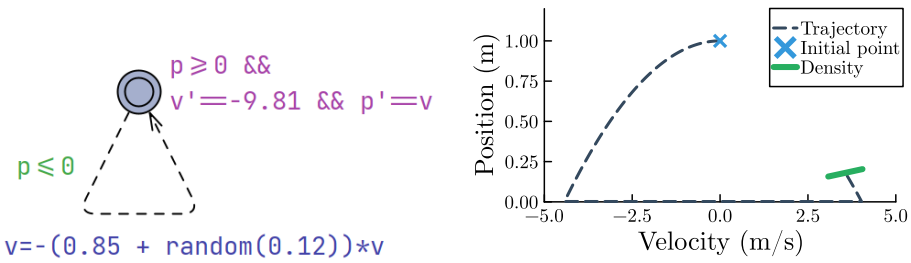


Fig. 2: A random walk with action sequence *slow, slow, slow, slow, fast, slow, fast.*

A run π of an EMDP is an alternating sequence $s_0 a_0 s_1 a_1 \ldots$ of states and actions such that $T(s_i, a_i)(s_{i+1}) > 0$ for all $i \geq 0$. A (memoryless) strategy for



(a) SHA for the bouncing ball.

(b) State density after one bounce.

Fig. 3: An SHA for the bouncing ball and a visualization after one bounce.

an EMDP is a function $\sigma : \mathcal{S} \to (Act \to [0,1])$, mapping a state to a probability distribution over Act. Given a strategy σ, the expected cost of reaching a goal state is defined as the solution to a Volterra integral equation as follows:

Definition 2 (Expected cost of a strategy). *Let $\mathcal{M} = (\mathcal{S}, s_0, Act, T, C, \mathcal{G})$ be an EMDP and σ be a strategy. If a state s can reach the goal set $\mathcal{G}$, the* expected cost *is the solution to the following recursive equation:*

$$\mathbb{E}_\sigma^{\mathcal{M}}(s) = \begin{cases} 0 & \text{if } s \in \mathcal{G} \\ \sum_{a \in Act} \sigma(s)(a) \cdot \int_{s' \in \mathcal{S}} T(s,a)(s') \cdot \left(C(s,a,s') + \mathbb{E}_\sigma^{\mathcal{M}}(s')\right) ds' & \text{if } s \notin \mathcal{G} \end{cases}$$

A strategy σ^* is optimal if it minimizes $\mathbb{E}_\sigma^{\mathcal{M}}(s_0)$. We note that there exists an optimal strategy which is deterministic.

Stochastic Hybrid Systems In an EMDP, the environment responds instantaneously to an action proposed by the agent according to the next-state density function T. In a more refined view, the agent proposes actions with some period P, and the response of the environment is a stochastic, time-bounded trajectory (bounded by the period P) over the state space. For this response, we use a stochastic hybrid system (SHS) [17,34], which allows the environment to interleave continuous evolution and discrete jumps.

Definition 3 (Stochastic hybrid system). *A* stochastic hybrid system *of dimension k is a tuple $\mathcal{H} = (\mathcal{S}, F, \mu, \eta)$ where*

- *$\mathcal{S} \subseteq \mathbb{R}^k$ is a bounded and closed part of k-dimensional Euclidean space,*
- *$F : \mathbb{R}_{\geq 0} \times \mathcal{S} \to \mathcal{S}$ is a flow function describing the evolution of the continuous state with respect to time, typically represented by differential equations,*
- *$\mu : \mathcal{S} \to (\mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0})$ maps each state s to a delay density function $\mu(s)$ determining the time point for the next discrete jump, and*
- *$\eta : \mathcal{S} \to (\mathcal{S} \to \mathbb{R}_{\geq 0})$ maps each state s to a density function $\eta(s)$ determining the next state.*

Example 2 (Bouncing ball). To represent an SHS, we use a stochastic hybrid automaton (SHA) [17], which we only introduce informally here. Fig. 3(a) shows an SHA of a bouncing ball, which we use as a running example. Here the state of the ball is given by a pair (p, v) of continuous variables, where $p \in \mathbb{R}_{\geq 0}$ represents the current height (position) and $v \in \mathbb{R}$ represents the current velocity of the ball. Initially (not visible in the figure) the value of v is zero while p is picked randomly in $[7.0, 10.0]$. The behavior of the ball is defined by two differential equations: $v' = -9.81m/s^2$ describing the velocity of a falling object and $p' = v$ stating that the rate of change of the height is the current velocity. The invariant $p \geq 0$ expresses that the height is always nonnegative. The single transition of the automaton triggers when $p \leq 0$, i.e., when the ball hits the ground. In this case the velocity reverts direction and is subject to a random dampening effect (here "`random(0.12)`" draws a random number from $[0, 0.12]$ uniformly). The state density after one bounce is illustrated in Fig. 3(b). The SHA induces the following SHS, where δ denotes the Dirac delta distribution:

- $\mathcal{S} = [0, 10] \times [-14, 14]$,
- $F((p, v), t) = ((-9.81/2)t^2 + vt + p, -9.81t + v)$
- $\mu((p, v)) = \delta\big((v + \sqrt{v^2 + 2 \cdot 9.81 \cdot p})/9.81\big)$
- $\eta((p, v)) = (p, v \cdot \mathcal{U}_{[-0.97,-0.85]})$, with uniform distribution $\mathcal{U}_{[l,u]}$ over $[l, u]$. ◁

A timed run ρ of an SHS $\mathcal{H}$ with n jumps from an initial state density ι is a sequence $\rho = s_0 s'_0 t_0 s_1 s'_1 t_1 s_2 s'_2 \ldots t_{n-1} s_n s'_n$ respecting the constraints of $\mathcal{H}$, where each $t_i \in \mathbb{R}_{\geq 0}$. The total duration of ρ is $\sum_{i=0}^{n-1} t_i$, and the density of ρ is $\iota(s_0) \cdot \prod_{i=0}^{n-1} \mu(s'_i)(t_i) \cdot \eta(s_{i+1})(s'_{i+1})$.

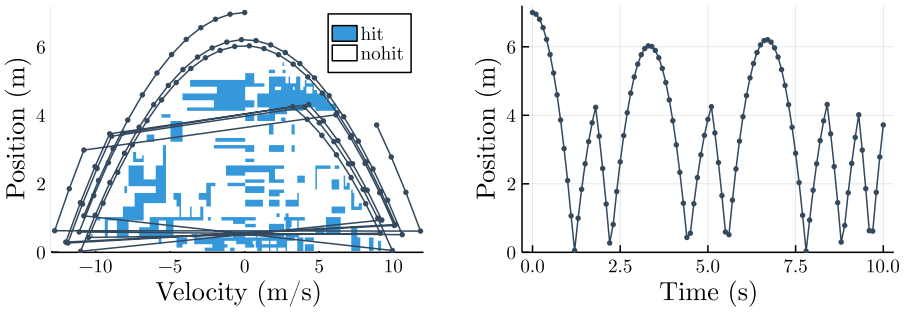
Given an initial state density ι and a time bound T, we denote by $\Delta^T_{\mathcal{H},\iota}$ the density function on $\mathcal{S}$ determining the state after a total delay of T, when starting in a state given by ι. The following recursive equation defines $\Delta^T_{\mathcal{H},\iota}$:[1]

$$\Delta^T_{\mathcal{H},\iota}(s') = \begin{cases} \iota(s') & \text{if } T = 0 \\ \displaystyle\int_s \iota(s) \cdot \int_{t \leq T} \mu(s)(t) \cdot \Delta^{T-t}_{\mathcal{H},\eta(F(t,s))}(s')\, dt\, ds & \text{if } T > 0 \end{cases}$$

For $T = 0$, the density of reaching s' is given by the initial state density function ι. For $T > 0$, reaching s' at T first requires to start from an initial state s (chosen according to ι), followed by some delay t (chosen according to $\mu(s)$), leaving the system in the state $F(t, s)$. From this state it remains to reach s' within time $(T - t)$ using $\eta(F(t, s))$ as initial state density.

Hybrid Markov Decision Processes A hybrid Markov decision process (HMDP) is essentially an EMDP where the actions of the agent are selected according to some time period $P \in \mathbb{R}_{\geq 0}$, and where the next-state probability density function T is obtained from an SHS.

[1] For SHS with an upper bound on the number of discrete jumps up to a given time bound T, the equation is well-defined.



(a) Strategy. (b) Example run for 10 seconds.

Fig. 4: Near-optimal strategy learned by UPPAAL STRATEGO.

Definition 4 (Hybrid Markov decision process). *A* hybrid Markov decision process *is a tuple* $\mathcal{HM} = (\mathcal{S}, s_0, Act, P, N, \mathcal{H}, C, \mathcal{G})$ *where* $\mathcal{S}, s_0, Act, C, \mathcal{G}$ *are defined the same way as for an EMDP, and*

- $P \in \mathbb{R}_{\geq 0}$ *is the period of the agent,*
- $N : \mathcal{S} \times Act \to (\mathcal{S} \to \mathbb{R}_{\geq 0})$ *maps each state* s *and action* a *to a probability density function determining the immediate next state under* a*, and*
- $\mathcal{H} = (\mathcal{S}, F, \mu, \eta)$ *is a stochastic hybrid system describing the responses of the environment.*

An HMDP $\mathcal{HM} = (\mathcal{S}, s_0, Act, P, N, \mathcal{H}, C, \mathcal{G})$ induces the EMDP $\mathcal{M}_{\mathcal{HM}} = (\mathcal{S}, s_0, Act, T, C, \mathcal{G})$, where T is given by $T(s, a) = \Delta^P_{\mathcal{H}, N(s,a)}$. That is, the next-state probability density function of $\mathcal{M}_{\mathcal{HM}}$ is given by the state density after a delay of P (the period) according to $\mathcal{H}$ with initial state density N.

Example 3 (Hitting the bouncing ball). Fig. 5 shows an HMDP extending the SHS of the bouncing ball from Fig. 3(a). Now a player has to keep the ball bouncing indefinitely by periodically choosing between the actions *hit* and *nohit*, (three solid transitions). The period $P = 0.1$ is modeled by a clock x with suitable invariant, guards and updates. The top transition triggered by the *nohit* action has no effect on the state (but will have no cost). The *hit* action affects the state only if the height of the ball is at least 4m ($p \geq 4$). The left transition applies if the ball is falling with a speed not greater than -4m/s ($v \geq -4$) and accelerates to a velocity of -4m/s. The right transition applies if the ball is

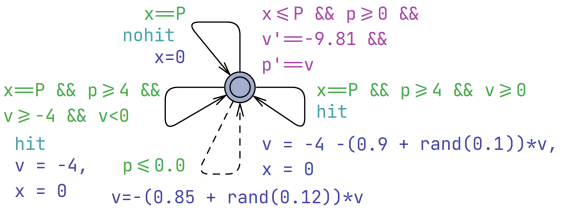


Fig. 5: An HMDP for hitting a bouncing ball.

rising, and sets the velocity to a random value in $[-v-4, -0.9v-4]$. The bottom dashed transition represents the bounce of the ball as in Fig. 3(a), which is part of the environment and outside the control of the agent.

A time-extended state (p, v, t) is in the goal set $\mathcal{G}$ if either $t \geq 120$ or $(p \leq 0.01 \wedge |v| \leq 1)$ (the ball is deemed dead). The cost (C) is 1 for the *hit* action and 0 for the *nohit* action, with an additional penalty of 1,000 for transitions leading to a dead state. Fig. 4 illustrates the near-optimal strategy σ^* obtained by the RL method implemented in UPPAAL STRATEGO and the prefix of a random run. The expected number of *hit* actions of σ^* within 120s is approximately 48. $\triangleleft$

3 Safety, Partitioning, Synthesis and Shielding

Safety In this section we are concerned with a strategy obtained for a given EMDP being *safe*. For example, a safety strategy for hitting the bouncing ball must ensure that the ball never reaches a dead state ($p \leq 0.01 \wedge |v| \leq 1$). In fact, although safety was encouraged by cost-tweaking, the strategy σ^* in Fig. 4 is *not* safe. In the following we use symbolic techniques to synthesize safety strategies.

Let $\mathcal{M} = (\mathcal{S}, s_0, Act, T, C, \mathcal{G})$ be an EMDP. A safety property φ is a set of states $\varphi \subseteq \mathcal{S}$. A run $\pi = s_0 a_0 s_1 a_1 s_2 \ldots$ is safe with respect to φ if $s_i \in \varphi$ for all $i \geq 0$. Given a nondeterministic strategy $\sigma : \mathcal{S} \to 2^{Act}$, a run $\pi = s_0 a_0 s_1 a_1 s_2 \ldots$ of $\mathcal{M}$ is an outcome of σ if $a_i \in \sigma(s_i)$ for all i. We say that σ is a safety strategy with respect to φ if all runs that are outcomes of σ are safe.

Partitioning and Strategies Given the infinite-state nature of the EMDP $\mathcal{M}$, we will resort to finite partitioning (similar to [46]) of the state space in order to algorithmically synthesize nondeterministic safety strategies. Given a predefined granularity γ, we partition the state space into disjoint regions of equal size (we do this for simplicity; our method is independent of the particular choice of the partitioning). The partitioning along each dimension of $\mathcal{S}$ is a half-open interval belonging to the set $\mathcal{I}_\gamma = \{[k\gamma, k\gamma + \gamma[\mid k \in \mathbb{Z}\}$. For a bounded k-dimensional state space $\mathcal{S}$, $\mathcal{A} = \{\mu \in \mathcal{I}_\gamma^k \mid \mu \cap \mathcal{S} \neq \emptyset\}$ provides a finite partitioning of $\mathcal{S}$ with granularity γ. For each $s \in \mathcal{S}$ we denote by $[s]_\mathcal{A}$ the unique region containing s.

Given an EMDP $\mathcal{M}$, a partitioning $\mathcal{A}$ induces a finite labeled transition system $\mathcal{T}_\mathcal{M}^\mathcal{A} = (\mathcal{A}, Act, \to)$, where

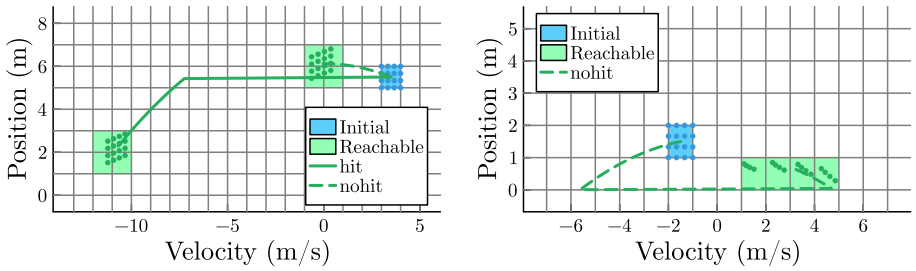
$$\mu \xrightarrow{a} \mu' \iff \exists s \in \mu.\ \exists s' \in \mu'.\ T(s, a)(s') > 0.$$

Fig. 6 shows a partitioning for the running example and displays some witnesses for transitions in the induced transition system.

Next, we view $\mathcal{T}_\mathcal{M}^\mathcal{A}$ as a 2-player game. For a region $\mu \in \mathcal{A}$, Player 1 challenges with an action $a \in Act$. Player 2 responds with a region $\mu' \in \mathcal{A}$ such that $\mu \xrightarrow{a} \mu'$.

Definition 5 (Safe regions). *Let $\varphi \subseteq \mathcal{S}$ be a safety property and $\mathcal{A}$ a partitioning. We denote by $\varphi^\mathcal{A}$ the set $\{\mu \in \mathcal{A} \mid \mu \subseteq \varphi\}$. The set of safe regions with respect to φ is the maximal set of regions $\mathbb{S}_\varphi$ such that*

$$\mathbb{S}_\varphi = \varphi^\mathcal{A} \cap \{\mu \mid \exists a.\ \forall \mu'.\ \mu \xrightarrow{a} \mu' \implies \mu' \in \mathbb{S}_\varphi\}. \tag{1}$$



(a) Scenario where the ball is rising and high enough to be hit.

(b) Scenario where the ball is too low to be hit, but bounces off the ground.

Fig. 6: State-space partitioning for Example 3. Starting in the blue region and depending on the action, the system can end up in the green regions within one time period, witnessed by simulations from 16 initial states.

Given the finiteness of $\mathcal{A}$ and monotonicity of (1), $\mathbb{S}_\varphi$ may be obtained in a finite number of interations using Tarski's fixed-point theorem [44].

A (nondeterministic) strategy for $\mathcal{T}^{\mathcal{A}}_{\mathcal{M}}$ is a function $\nu : \mathcal{A} \to 2^{Act}$. The most permissive safety strategy ν_φ obtained from $\mathbb{S}_\varphi$ [8] is given by

$$\nu_\varphi(\mu) = \{a \mid \forall \mu'.\ \mu \xrightarrow{a} \mu' \implies \mu' \in \mathbb{S}_\varphi\}.$$

The following theorem states that we can obtain a safety strategy for the original EMDP $\mathcal{M}$ from a safety strategy ν for $\mathcal{T}^{\mathcal{A}}_{\mathcal{M}}$.

Theorem 1. *Given an EMDP $\mathcal{M}$, safety property $\varphi \subseteq \mathcal{S}$ and partitioning $\mathcal{A}$, if ν is a safety strategy for $\mathcal{T}^{\mathcal{A}}_{\mathcal{M}}$, then $\sigma(s) = \nu([s]_{\mathcal{A}})$ is a safety strategy for $\mathcal{M}$.*

Approximating the 2-player Game Let $\mathcal{M}$ be an EMDP and φ be a safety property. To algorithmically compute the set of safe regions $\mathbb{S}_\varphi$ for a given partitioning $\mathcal{A}$, and subsequently the most permissive safety strategy ν_φ, the transition relation $\xrightarrow{a}$ needs to be a decidable predicate. If $\mathcal{M}$ is derived from an HMDP $\mathcal{HM} = (\mathcal{S}, s_0, Act, P, N, \mathcal{H}, C, \mathcal{G})$, this requires decidability of the predicate $\Delta^P_{\mathcal{H},N(s,a)}(s') > 0$. Consider the bouncing ball from Example 3. The regions are of the form $\mu = \{(p, v) \mid l_p \le p < u_p \wedge l_v \le v < u_v\}$. For given regions μ, μ', the predicate $\mu \xrightarrow{nohit} \mu'$ is equivalent to the following first-order predicate over the reals (note that $F((p, v), t)$ is a pair of polynomials in p, v and t):[2]

$$\exists (p, v) \in \mu.\ F((p, v), P) \in \mu' \vee \exists \beta \in [0.85, 0.97].\ \exists t' \le P.\ \exists v'.\\ F((p, v), t') = (0, v') \wedge F((0, -\beta \cdot v'), P - t') \in \mu'$$

For this simple example, the validity of the formula can be decided [43], which may however require doubly exponential time [16], and worse, when considering

[2] We assume that at most one bounce can take place within the period P.

nonlinear dynamics with, e.g., trigonometric functions, the problem becomes undecidable [33]. One can obtain a conservative answer via over-approximate reachability analysis [20]; in Section 4 we compare to such an approach and demonstrate that, while effective, that approach also does not scale. This motivates to use an efficient and robust alternative. We propose to approximate the transition relation using equally spaced samples, which are simulated according to the SHS $\mathcal{H}$ underlying the given HMDP $\mathcal{HM}$.

Algorithm 1 describes how to compute an approximation $\mu \xrightarrow{a}_{app} \mu'$ of $\mu \xrightarrow{a} \mu'$. The algorithm draws from a finite set of n evenly distributed supporting points per dimension $app[\mu] = \{s_1, \ldots, s_{n^k}\} \subseteq \mu$ and simulates $\mathcal{H}$ for P time units. A region μ' is declared reachable from μ under action a if it is reached in at least one simulation. When stochasticity is involved in a simulation, additional care must be taken. The random variables can be considered an additional dimension to be sampled from; alternatively, a worst-case value can be used if available, such as the bouncing ball with the highest velocity damping. Fig. 6 illustrates 16 ($n = 4$) possible starting points for the bouncing ball together with most pessimistic outcomes, depending on the action taken.

Algorithm 1 Approximation of $\xrightarrow{a}$

Input: $\mu \in \mathcal{A}, a \in Act$
Output: $\mu \xrightarrow{a}_{app} \mu'$ iff $\mu' \in R$
1: $R = \emptyset$
2: **for all** $s_i \in app[\mu]$ **do**
3: select $s_i' \sim N(s_i, a)$
4: simulate $\mathcal{H}$ from s_i' for P time units
5: let s_i'' be the resulting state
6: add $[s_i'']_{\mathcal{A}}$ to R
7: **end for**

The result $\xrightarrow{a}_{app}$ is an underapproximation of the transition relation $\xrightarrow{a}$, with a corresponding transition system $\widehat{\mathcal{T}}^{\mathcal{A}}_{\mathcal{M}} = (\mathcal{A}, Act, \rightarrow_{app})$. Thus if we compute a safety strategy ν from $\xrightarrow{a}_{app}$, then the strategy $\sigma(s) = \nu([s]_{\mathcal{A}})$ from Theorem 1 is not necessarily safe. However, in Section 4 we will see that this strategy is statistically safe in practice. We attribute this to two reasons. 1) The underapproximation of $\xrightarrow{a}_{app}$ can be made accurate. 2) Since $\xrightarrow{a}$ is defined over an abstraction, it is often robust against small approximation errors.

Shielding As argued above, we can obtain the most permissive safety strategy ν_φ from $\xrightarrow{a}_{app}$ over $\mathcal{A}$ and then use $\sigma_\varphi(s) = \nu_\varphi([s]_{\mathcal{A}})$ as an approximation of the most permissive safety strategy over the original HMDP. We can employ σ_φ to build a shield. As discussed in the introduction, we focus on two ways of shielding: *pre-shielding* and *post-shielding* (recall Fig. 1). In pre-shielding, the shield is already active during the learning phase of the agent, which hence only trains on sets of safe actions. In post-shielding, the shield is only applied after the learning phase, and unsafe actions chosen by the agent are corrected (which is possibly detrimental to the performance of the agent).

Fig. 7 shows examples of such strategies for the random walk (Example 1) and the bouncing ball. As can be seen, most regions of the state space are either unsafe (black) or both actions are safe (white). Only in a small area (purple) will the strategy enforce walking fast or hitting the ball, respectively. In the white area, the agent can learn the action that leads to the highest performance.

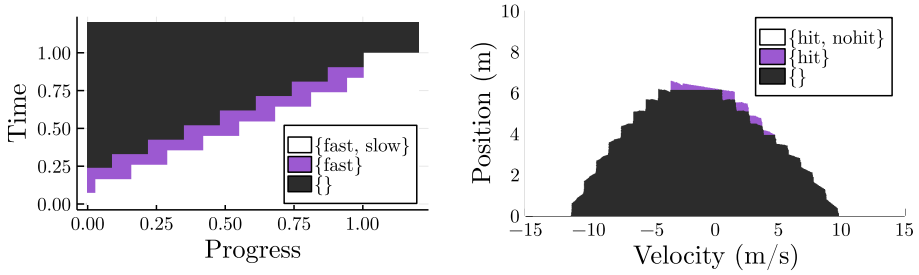


Fig. 7: Synthesized strategies for random walk (left) and bouncing ball (right).

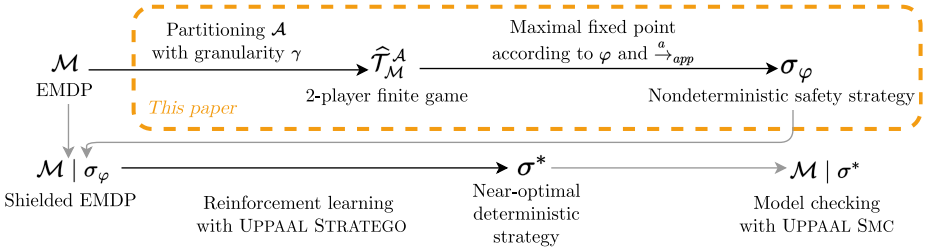


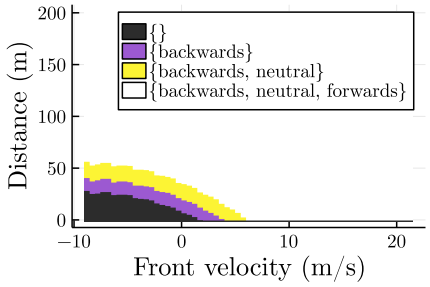
Fig. 8: Complete method for pre-shielding and statistical model checking (SMC).

We use Uppaal Stratego [18] to train a shielded agent based on σ_φ. The complete workflow of pre-shielding and learning is depicted in Fig. 8. Starting from the EMDP, we partition the state space, obtain the transition system using Algorithm 1 and solve the game according to a safety property φ, as described above. The produced strategy is then conjoined with the original EMDP to form the shielded EMDP, and reinforcement learning is used to produce a near-optimal deterministic strategy σ^*. This strategy can then be used in the real world, or get evaluated via statistical model checking. The only difference in the workflow in post-shielding is that the strategy σ_φ is not applied to the EMDP, but on top of the deterministic strategy σ^*.

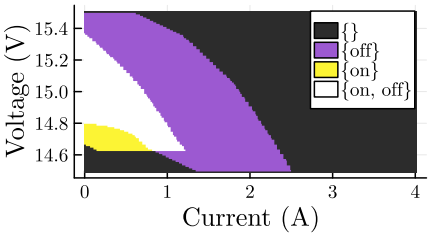
4 Experiments

In this section we study our proposed approach with regard to different aspects of our shields. In addition to the random walk (Example 1) and bouncing ball (Example 3), we consider three benchmark cases:

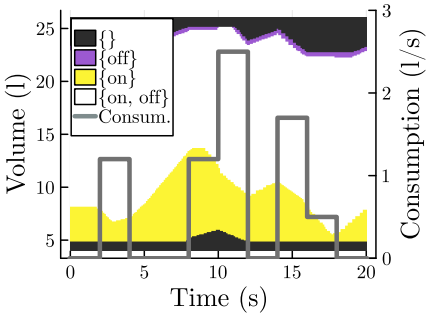
– *Cruise control* [36,35,4]: A car is controlled to follow another car as closely as possible without crashing. Either car can accelerate, keep its speed, or decelerate freely, which makes finding a strategy challenging. This model was subject to several previous studies where a safety strategy was carefully designed, while our method can be directly applied without human effort.



(a) Cruise control ($n = 4$, $\gamma = 0.5$) when the car's velocity is $0m/s$



(b) DC-DC boost converter ($n = 4$, $\gamma = 0.01$) when the output resistance is 30Ω.



(c) Oil pump ($n = 4, \gamma = 0.1$) when the pump is *on*. The periodic piecewise consumption pattern has been overlaid. Turning off the pump requires it to stay off for two seconds, which could cause an underflow in the yellow area. Conversely, the purple area shows the states where the pump *must* be turned off to avoid overflow. Since the pump is on in this projection, this can wait until the last moment.

Fig. 9: Projected views of synthesized most permissive safety strategies.

- *DC-DC converter* [31]: This industrial DC-DC boost converter transforms input voltage of 10V to output voltage of 15V. The controller switches between storing energy in an inductor and releasing it. The output must stay in ±0.5V around 15V, and the amount of switching should be minimized.
- *Oil pump* [49]: In this industrial case, flow of oil into an accumulator is controlled to satisfy minimum and maximum volume constraints, given a consumption pattern that is piecewise-constant and repeats every 20 seconds. Since the exact consumption is unknown, a random perturbation is added to the reference value. To reduce wear, the volume should be kept low.

Fig. 9 shows the synthesized most permissive safety strategies. For instance, in Fig. 9(a) we see the strategy for the cruise-control example when the controlled car is standing still. If the car in front is either close or reverses at high speed, the controlled car must also reverse (purple area). The yellow area shows states where it is safe to stand still but accelerating may lead to a collision.

We conduct four series of experiments to study different aspects of our approach. (1) The quality of our approximation of the transition relation $\xrightarrow{a}_{app}$, (2) the computational performance of our approximation in comparison with a fully symbolic approach, (3) the performance in terms of reward and safety of the pre- and post-shields synthesized with our method, and (4) the potential of post-optimization for post-shielding.

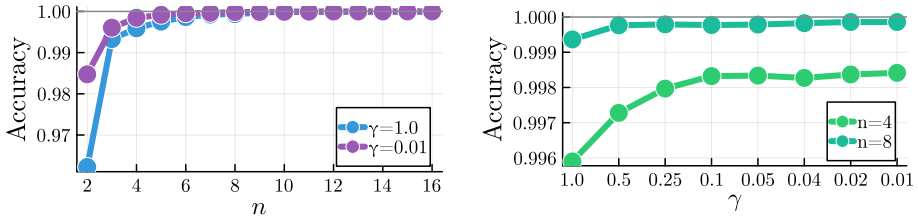


Fig. 10: Accuracy of the approximation $\xrightarrow{a}_{app}$ under different granularity γ and number of supporting points n per dimension.

All experiments are conducted on an AMD Ryzen 7 5700x with 33 GiB RAM. Our implementation is written in Julia, and we use UPPAAL STRATEGO [18] for learning and statistical model checking. The experiments are available online [1].

Quality of the Approximated Transition System In the first experiment we statistically assess the approximation quality of $\xrightarrow{a}_{app}$ wrt. the underlying infinite transition system. For varying granularity γ of $\mathcal{A}$ and numbers of supporting points n per dimension (see Section 3) we first compute $\xrightarrow{a}_{app}$ with Algorithm 1. Then we uniformly sample 10^8 states s and compute their successor states s' under a random action a. Finally we count how often $[s]_{\mathcal{A}} \xrightarrow{a}_{app} [s']_{\mathcal{A}}$ holds.

Here we consider the bouncing-ball model, where we limit the domain to $p \in [0, 15]$, $v \in [-15, 15]$. The results are shown in Fig. 10. An increase in the number of supporting points n correlates with increased accuracy. For $\gamma \leq 1$, using $n = 3$ supporting points already yields accuracy above 99%. Finer partition granularity γ increases accuracy, but less so compared to increasing n.

Comparison with Fully Symbolic Approach As described in Section 3, as an alternative to Algorithm 1 one can use a reachability algorithm to obtain an overapproximation of the transition relation $\xrightarrow{a}$. Here we analyze the performance of such an approach based on the reachability tool JULIAREACH [10]. Given a set of initial states of a hybrid automaton where we have substituted probabilities by nondeterminism, JULIAREACH can compute an overapproximation of the successor states. In JULIAREACH, we select the reachability algorithm from [24]. This algorithm uses time discretization, which requires a small time step to give precise answers. This makes the approach expensive. For instance, for the bouncing-ball system, the time period is $P = 0.1$ time units, and a time step of 0.001 time units is required, which corresponds to 100 iterations.

The shield obtained with JULIAREACH is safe by construction. To assess the safety of the shield obtained with Algorithm 1, we choose an agent that selects an action at random and let it act under the post-shield for 10^6 episodes. (We use a random agent because a learned agent may have learned to act safely most of the time and thus not challenge the shield as much.) If no safety violation was detected, we compute 99% confidence intervals for the statistical safety.

Table 1: Synthesis results for the bouncing ball under varying granularity (γ) and supporting points (n) using Algorithm 1 (top) and two choices of the time-step parameter using JuliaReach (bottom). The safety probability is computed for a 99% confidence interval. $\gamma = 0.02$ corresponds to $9.0 \cdot 10^5$ partitions, and $\gamma = 0.01$ quadruples the number of partitions to $3.6 \cdot 10^6$.

γ	$\xrightarrow{a}_{app}$ **method**	**Parameters**	**Time**	**Probability safe**
0.02	Algorithm 1	$n = 2$	1m 50s	unsafe run found
0.02		$n = 4$	2m 14s	[99.9999%, 100%]
0.02		$n = 8$	4m 02s	[99.9999%, 100%]
0.02		$n = 16$	11m 03s	[99.9999%, 100%]
0.01	Algorithm 1	$n = 2$	16m 49s	[99.9999%, 100%]
0.01		$n = 4$	19m 00s	[99.9999%, 100%]
0.01		$n = 8$	27m 21s	[99.9999%, 100%]
0.01		$n = 16$	56m 32s	[99.9999%, 100%]
0.01	JuliaReach	time step 0.002	24h 30m	considers s_0 unsafe
0.01		time step 0.001	41h 05m	safe by construction

We consider again the bouncing-ball model. JuliaReach requires a low partition granularity $\gamma = 0.01$; for $\gamma = 0.02$ it cannot prove that a safety strategy exists, which may be due to conservatism, while our method is able to synthesize a shield that, for $n \geq 4$, is statistically safe. Table 1 shows the results obtained from the two approaches. In addition, the reachability algorithm uses time discretization, and a small time step is required to find a safety strategy.

We remark that the bouncing-ball model has linear dynamics, for which reachability analysis is relatively efficient compared to nonlinear dynamics, and thus this model works in favor of the symbolic method. However, the hybrid nature of the model and the large number of queries (one for each partition-action pair) still make the symbolic approach expensive. Considering the case $\gamma = 0.01$ and $n = 4$, our method can synthesize a strategy in 19 minutes, while the approach based on JuliaReach takes 41 hours.

Fig. 11 visualizes the two strategies and shows how the two approaches largely agree on the synthesized shield – but also the slightly more pessimistic nature of the transition relation computed with JuliaReach.

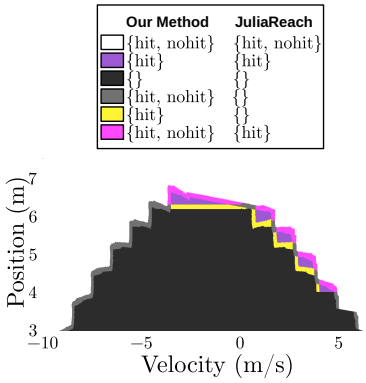
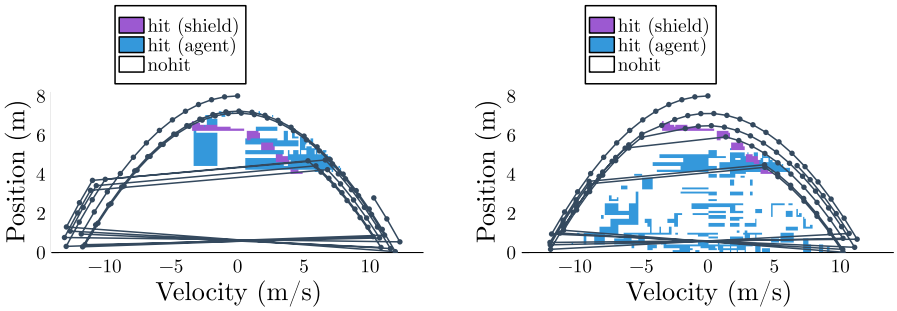


Fig. 11: Superimposed strategies of our method and JuliaReach.



(a) Pre-shield. (b) Post-shield.

Fig. 12: Learned shielded strategies for the bouncing ball.

Evaluation of Pre- and Post-shields In the next series of experiments, we evaluate the full method of obtaining a shielded agent. The first step is to approximate $\xrightarrow{a}_{app}$ using Algorithm 1 and extract the most permissive safety strategy σ_φ to be used as a shield. For the second step we have two options: pre- or post-shielding. Recall from Fig. 1 that a pre-shield is applied to the agent during training while a post-shield is applied after training.

In the case of the bouncing ball, the post-shielded agent's strategy is shown in Fig. 12(b). It consists of the unshielded strategy from Fig. 4 plus the purple regions of the safety strategy in Fig. 7(b). Correspondingly, Fig. 12(a) shows the pre-shielded strategy, which is significantly simpler because it does not explore unsafe regions of the state space. This also leads to faster convergence.

Table 2 reports the same data as in Table 1 for the other models. Overall, we see a similar trend in all tables. For a low number of supporting points (say, $n = 3$) we can obtain a safety strategy that we find to be statistically safe. In all cases, no unsafe run was detected in the statistical evaluation. The synthesis time varies depending on the model and is generally feasible. The longest computation times are seen for the oil-pump example, which has the most dimensions. Still, times are well below JuliaReach for the comparatively simple bouncing ball.

Next, we compare our method to other options to make an agent safe(r). As the baseline, we use the classic RL approach, where safety is encouraged using reward shaping. We experiment with a deterrence $d \in \{0, 10, 100, 1000\}$ (negative reward) as a penalty for safety violations for the learning agent. Note that this penalty is only applied during training, and not included in the total cost when we evaluate the agent below. As the second option, we use a post-shielded agent, to which the deterrence also applies. The third option is a pre-shielded agent. In all cases, training and evaluation is repeated 10 times, and the mean value is reported. The evaluation is based on 1000 traces for each repetition.

Table 2: Shield synthesis for different models and granularities γ computed using Algorithm 1. The safety probability is computed for a 99% confidence interval.

(a) Cruise control. $\gamma = 1$ corresponds to $1.9 \cdot 10^5$ partitions, and $\gamma = 0.5$ to $1.5 \cdot 10^6$.

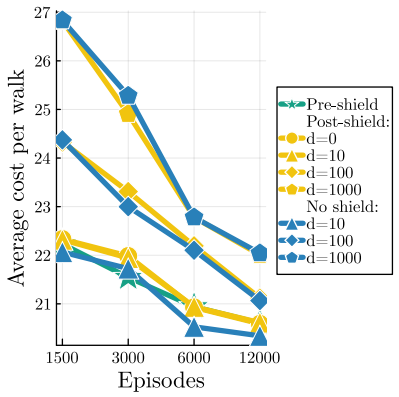
γ	n	**Time**	**Probability safe**
1	2	1m 50s	Considers s_0 unsafe
0.5	2	13m 16s	[99.9995%, 100%]
0.5	3	23m 03s	[99.9995%, 100%]
0.5	4	35m 55s	[99.9995%, 100%]

(b) DC-DC boost converter. $\gamma = 0.05$ corresponds to $3.1 \cdot 10^5$ partitions, $\gamma = 0.02$ to $1.7 \cdot 10^6$ and $\gamma = 0.01$ to $7.0 \cdot 10^6$.

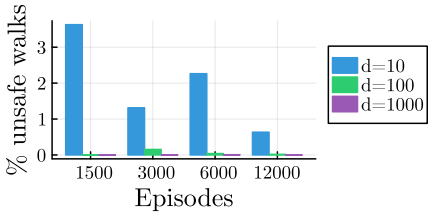
γ	n	**Time**	**Probability safe**
0.05	2	41s	[99.9995%, 100%]
0.05	3	1m 50s	considers s_0 unsafe
0.05	4	3m 30s	considers s_0 unsafe
0.02	2	3m 43s	[99.9995%, 100%]
0.02	3	8m 59s	[99.9995%, 100%]
0.02	4	18m 11s	[99.9995%, 100%]
0.01	2	15m 48s	[99.9995%, 100%]
0.01	3	38m 26s	[99.9995%, 100%]
0.01	4	1h 19m	[99.9995%, 100%]

(c) Oil pump. $\gamma = 0.2$ corresponds to $2.8 \cdot 10^5$ partitions, and $\gamma = 0.1$ to $1.1 \cdot 10^6$.

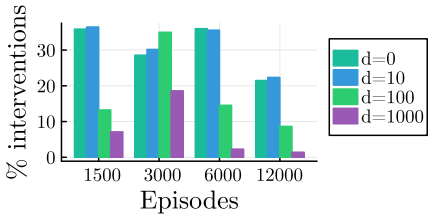
γ	n	**Time**	**Probability safe**
0.2	2	3m 07s	considers s_0 unsafe
0.1	2	32m 15s	[99.9995%, 100%]
0.1	3	1h 37m	[99.9995%, 100%]
0.1	4	5h 23m	[99.9995%, 100%]



(a) Average cost per run.



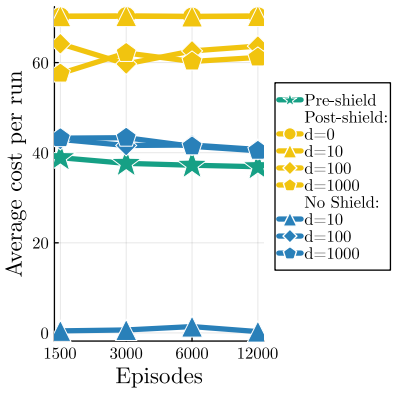
(b) Safety violations for unshielded agents



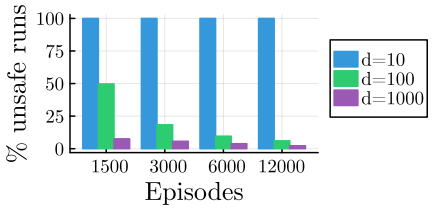
(c) Interventions for post-shielded agents.

Fig. 13: Results of shielding the random walk using $\gamma = 0.005$.

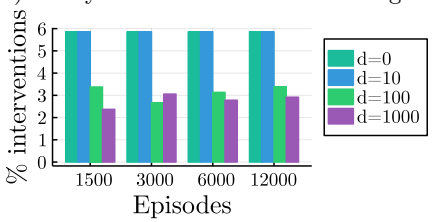
Figures 13, to 17 report the results for the different models. Each subfigure shows the following content: (a) shows the average cost of the final agent, (b) shows the amount of safety violations of the unshielded agents and (c) shows the number of times the post-shielded agents were intervened by the shield.



(a) Average *hit* actions per 120s.

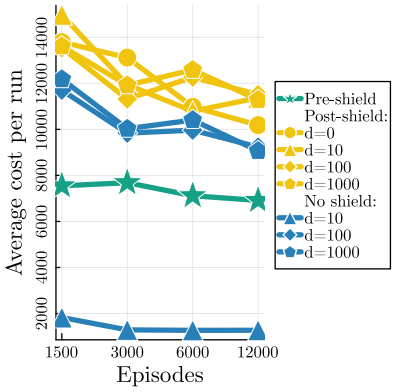


(b) Safety violations for unshielded agents.

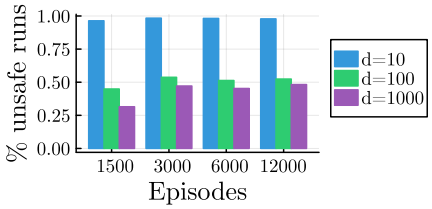


(c) Interventions for post-shielded agents.

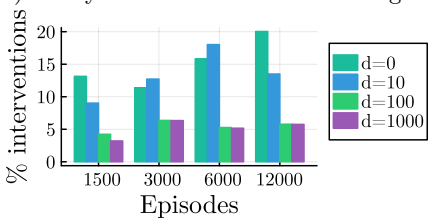
Fig. 14: Results of shielding the bouncing ball using $n = 16$, $\gamma = 0.01$.



(a) Accumulated distance per 120s.



(b) Safety violations for unshielded agents.

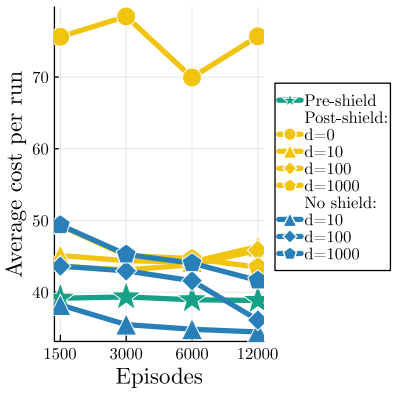


(c) Interventions for post-shielded agents.

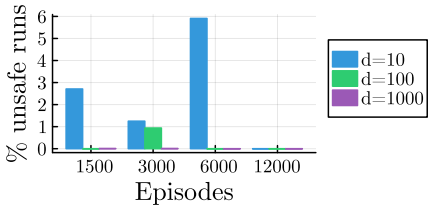
Fig. 15: Results of shielding the cruise control using $n = 4$, $\gamma = 0.5$.

Overall, we observe similar tendencies. The unshielded agent has lowest average cost at deployment time under low deterrence, but it also violates safety. Higher deterrence values improve safety, but do not guarantee it.

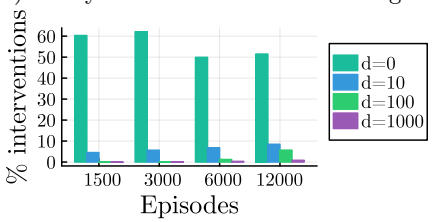
The pre-shielded agents outperform the post-shielded agents. This is because they learn a near-optimal strategy subject to the shield, while the post-shielded agents may be based on a learned unsafe strategy that contradicts the shield, and thus the shield interference can be more detrimental.



(a) Accumulated error plus number of switches per 120µs.

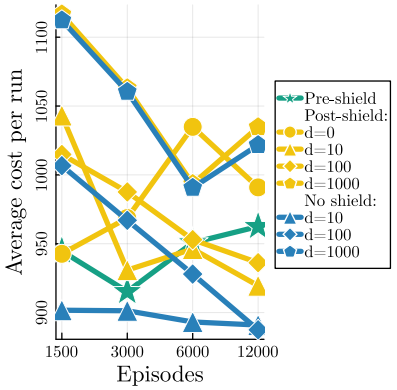


(b) Safety violations for unshielded agents.

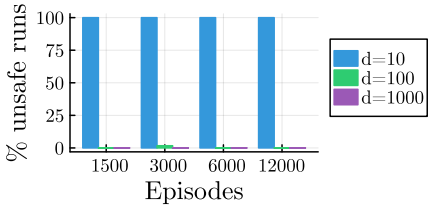


(c) Interventions for post-shielded agents.

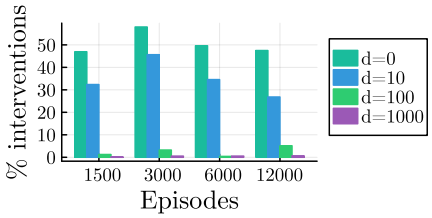
Fig. 16: Results of shielding the DC-DC boost converter using $n = 4$, $\gamma = 0.01$.



(a) Accumulated oil volume per 120s.



(b) Safety violations for unshielded agents.



(c) Interventions for post-shielded agents.

Fig. 17: Results of shielding the oil pump using $n = 4$, $\gamma = 0.1$.

Post-Shielding Optimization When a post-shield intervenes, more than one action may be valid. This leaves room for further optimization, for which we can use Uppaal Stratego. Compared to a uniform baseline, we assess three ways to resolve nondeterminism: 1) minimizing interventions, 2) minimizing cost and 3) at the preference of the shielded agent (the so-called Q-value [48]).

Table 3 shows the effect of post-optimization on the cost and the number of interventions for the cruise-control example. Notably, cost is only marginally affected, but the number of shield interventions can get significantly higher. The pre-shielded agent has lower cost than all post-optimized alternatives.

Table 3: Change of post-optimization relative to the uniform-choice strategy. The strategy was trained for 12,000 episodes with $d = 10$ and post-optimized for 4,000 episodes. Performance of the pre-shielded agent is included for comparison, but interventions are not applicable (because the shield was in place during training).

Configuration	**Cost**	**Interventions**
Baseline with uniform random choice	11371	13.50
Minimizing interventions	11791 (+3.7%)	11.43 (−15.3%)
Minimizing cost	10768 (−5.3%)	17.43 (+29.1%)
Agent preference	11493 (−1.1%)	14.55 (+7.8%)
Pre-shielded agent	6912 (−39.2%)	– –

5 Conclusion

We presented a practical approach to synthesize a near-optimal safety strategy via finite (2-player) abstractions of hybrid Markov decision processes, which are systems of complex probabilistic and hybrid nature. In particular, we deploy a simulation-based technique for inferring the 2-player abstraction, from which a safety shield can then be constructed. We show with high statistical confidence that the shields avoid unsafe outcomes in the case studies, and are significantly faster to construct than when deploying symbolic techniques for computing a correct 2-player abstraction. In particular, our method demonstrates statistical safety on several case studies, two of which are industrial. Furthermore, we study the difference between pre- and post-shielding, reward engineering and a post-shielding optimization. In general, we observe that reward engineering is insufficient to enforce safety, and secondarily observe that pre-shielding provides better controller performance compared to post-shielding.

Future work includes applying the method to more complex systems, and using formal methods to verify the resulting safety strategies, maybe based on [22].

Acknowledgments

This research was partly supported by DIREC - Digital Research Centre Denmark and the Villum Investigator Grant S4OS - Scalable analysis and Synthesis of Safe, Secure and Optimal Strategies for Cyber-Physical Systems.

References

1. Reproducibility package - shielded reinforcement learning for hybrid systems. https://github.com/AsgerHB/Shielded-Learning-for-Hybrid-Systems
2. Abate, A., Amin, S., Prandini, M., Lygeros, J., Sastry, S.: Computational approaches to reachability analysis of stochastic hybrid systems. In: HSCC. LNCS, vol. 4416, pp. 4–17. Springer (2007). https://doi.org/10.1007/978-3-540-71493-4_4

3. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: AAAI. pp. 2669–2678. AAAI Press (2018). https://doi.org/10.1609/aaai.v32i1.11797
4. Ashok, P., Kretínský, J., Larsen, K.G., Le Coënt, A., Taankvist, J.H., Weininger, M.: SOS: safe, optimal and small strategies for hybrid Markov decision processes. In: QEST. LNCS, vol. 11785, pp. 147–164. Springer (2019). https://doi.org/10.1007/978-3-030-30281-8_9
5. Badings, T.S., Romao, L., Abate, A., Parker, D., Poonawala, H.A., Stoelinga, M., Jansen, N.: Robust control for dynamical systems with non-Gaussian noise via formal abstractions. J. Artif. Intell. Res. **76**, 341–391 (2023). https://doi.org/10.1613/jair.1.14253
6. Bastani, O., Li, S.: Safe reinforcement learning via statistical model predictive shielding. In: Robotics (2021). https://doi.org/10.15607/RSS.2021.XVII.026
7. Berkenkamp, F., Turchetta, M., Schoellig, A.P., Krause, A.: Safe model-based reinforcement learning with stability guarantees. In: NeurIPS. pp. 908–918 (2017), https://proceedings.neurips.cc/paper/2017/hash/766ebcd59621e305170616ba3d3dac32-Abstract.html
8. Bernet, J., Janin, D., Walukiewicz, I.: Permissive strategies: from parity games to safety games. RAIRO Theor. Informatics Appl. **36**(3), 261–275 (2002). https://doi.org/10.1051/ita:2002013
9. Bloem, R., Könighofer, B., Könighofer, R., Wang, C.: Shield synthesis: Runtime enforcement for reactive systems. In: TACAS. LNCS, vol. 9035, pp. 533–548. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_51
10. Bogomolov, S., Forets, M., Frehse, G., Potomkin, K., Schilling, C.: JuliaReach: a toolbox for set-based reachability. In: HSCC. pp. 39–44. ACM (2019). https://doi.org/10.1145/3302504.3311804
11. Bujorianu, L.M.: Stochastic reachability analysis of hybrid systems. Springer Science & Business Media (2012)
12. Busoniu, L., de Bruin, T., Tolic, D., Kober, J., Palunko, I.: Reinforcement learning for control: Performance, stability, and deep approximators. Annu. Rev. Control. **46**, 8–28 (2018). https://doi.org/10.1016/j.arcontrol.2018.09.005
13. Carr, S., Jansen, N., Junges, S., Topcu, U.: Safe reinforcement learning via shielding under partial observability. In: AAAI. pp. 14748–14756. AAAI Press (2023). https://doi.org/10.1609/aaai.v37i12.26723
14. Cheng, R., Orosz, G., Murray, R.M., Burdick, J.W.: End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In: AAAI. pp. 3387–3395. AAAI Press (2019). https://doi.org/10.1609/aaai.v33i01.33013387
15. Chow, Y., Nachum, O., Duéñez-Guzmán, E.A., Ghavamzadeh, M.: A Lyapunov-based approach to safe reinforcement learning. In: NeurIPS. pp. 8103–8112 (2018), https://proceedings.neurips.cc/paper/2018/hash/4fe5149039b52765bde64beb9f674940-Abstract.html
16. Davenport, J.H., Heintz, J.: Real quantifier elimination is doubly exponential. Journal of Symbolic Computation **5**(1), 29–35 (1988). https://doi.org/10.1016/S0747-7171(88)80004-X
17. David, A., Du, D., Larsen, K.G., Legay, A., Mikucionis, M., Poulsen, D.B., Sedwards, S.: Statistical model checking for stochastic hybrid systems. In: HSB. EPTCS, vol. 92, pp. 122–136 (2012). https://doi.org/10.4204/EPTCS.92.9
18. David, A., Jensen, P.G., Larsen, K.G., Mikucionis, M., Taankvist, J.H.: Uppaal Stratego. In: TACAS. LNCS, vol. 9035, pp. 206–211. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_16

19. Donzé, A.: Breach, A toolbox for verification and parameter synthesis of hybrid systems. In: CAV. LNCS, vol. 6174, pp. 167–170. Springer (2010). https://doi.org/10.1007/978-3-642-14295-6_17
20. Doyen, L., Frehse, G., Pappas, G.J., Platzer, A.: Verification of hybrid systems. In: Handbook of Model Checking, pp. 1047–1110. Springer (2018). https://doi.org/10.1007/978-3-319-10575-8_30
21. Doyle, J.C., Francis, B.A., Tannenbaum, A.R.: Feedback control theory. Courier Corporation (2013)
22. Forets, M., Freire, D., Schilling, C.: Efficient reachability analysis of parametric linear hybrid systems with time-triggered transitions. In: MEMOCODE. pp. 1–6. IEEE (2020). https://doi.org/10.1109/MEMOCODE51338.2020.9314994
23. García, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. J. Mach. Learn. Res. **16**, 1437–1480 (2015). https://doi.org/10.5555/2789272.2886795
24. Guernic, C.L., Girard, A.: Reachability analysis of hybrid systems using support functions. In: CAV. LNCS, vol. 5643, pp. 540–554. Springer (2009). https://doi.org/10.1007/978-3-642-02658-4_40
25. Hasanbeig, M., Abate, A., Kroening, D.: Cautious reinforcement learning with logical constraints. In: AAMAS. pp. 483–491 (2020). https://doi.org/10.5555/3398761.3398821
26. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? J. Comput. Syst. Sci. **57**(1), 94–124 (1998). https://doi.org/10.1006/jcss.1998.1581
27. Jaeger, M., Bacci, G., Bacci, G., Larsen, K.G., Jensen, P.G.: Approximating Euclidean by imprecise Markov decision processes. In: ISoLA. LNCS, vol. 12476, pp. 275–289. Springer (2020). https://doi.org/10.1007/978-3-030-61362-4_15
28. Jaeger, M., Jensen, P.G., Larsen, K.G., Legay, A., Sedwards, S., Taankvist, J.H.: Teaching Stratego to play ball: Optimal synthesis for continuous space MDPs. In: ATVA. LNCS, vol. 11781, pp. 81–97. Springer (2019). https://doi.org/10.1007/978-3-030-31784-3_5
29. Jansen, N., Könighofer, B., Junges, S., Serban, A., Bloem, R.: Safe reinforcement learning using probabilistic shields. In: CONCUR. LIPIcs, vol. 171, pp. 3:1–3:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.CONCUR.2020.3
30. Kapinski, J., Krogh, B.H., Maler, O., Stursberg, O.: On systematic simulation of open continuous systems. In: HSCC. LNCS, vol. 2623, pp. 283–297. Springer (2003). https://doi.org/10.1007/3-540-36580-X_22
31. Karamanakos, P., Geyer, T., Manias, S.: Direct voltage control of DC-DC boost converters using enumeration-based model predictive control. IEEE Transactions on Power Electronics **29**(2), 968–978 (2013)
32. Klischat, M., Althoff, M.: A multi-step approach to accelerate the computation of reachable sets for road vehicles. In: ITSC. pp. 1–7. IEEE (2020). https://doi.org/10.1109/ITSC45102.2020.9294328
33. Laczkovich, M.: The removal of π from some undecidable problems involving elementary functions. Proceedings of the American Mathematical Society **131**(7), 2235–2240 (2003). https://doi.org/10.1090/S0002-9939-02-06753-9
34. Larsen, K.G.: Statistical model checking, refinement checking, optimization, ... for stochastic hybrid systems. In: FORMATS. LNCS, vol. 7595, pp. 7–10. Springer (2012). https://doi.org/10.1007/978-3-642-33365-1_2

35. Larsen, K.G., Le Coënt, A., Mikucionis, M., Taankvist, J.H.: Guaranteed control synthesis for continuous systems in Uppaal Tiga. In: CyPhy/WESE. LNCS, vol. 11615, pp. 113–133. Springer (2018). https://doi.org/10.1007/978-3-030-23703-5_6
36. Larsen, K.G., Mikucionis, M., Taankvist, J.H.: Safe and optimal adaptive cruise control. In: Correct System Design. LNCS, vol. 9360, pp. 260–277. Springer (2015). https://doi.org/10.1007/978-3-319-23506-6_17
37. Lewis, F.L., Vrabie, D., Syrmos, V.L.: Optimal control. John Wiley & Sons (2012)
38. Luo, Y., Ma, T.: Learning barrier certificates: Towards safe reinforcement learning with zero training-time violations. In: NeurIPS. pp. 25621–25632 (2021), https://proceedings.neurips.cc/paper/2021/hash/d71fa38b648d86602d14ac610f2e6194-Abstract.html
39. Maderbacher, B., Schupp, S., Bartocci, E., Bloem, R., Nickovic, D., Könighofer, B.: Provable correct and adaptive simplex architecture for bounded-liveness properties. In: SPIN. LNCS, vol. 13872, pp. 141–160. Springer (2023). https://doi.org/10.1007/978-3-031-32157-3_8
40. Majumdar, R., Ozay, N., Schmuck, A.: On abstraction-based controller design with output feedback. In: HSCC. pp. 15:1–15:11. ACM (2020). https://doi.org/10.1145/3365365.3382219
41. Noaeen, M., Naik, A., Goodman, L., Crebo, J., Abrar, T., Abad, Z.S.H., Bazzan, A.L.C., Far, B.H.: Reinforcement learning in urban network traffic signal control: A systematic literature review. Expert Syst. Appl. **199**, 116830 (2022). https://doi.org/10.1016/j.eswa.2022.116830
42. Shmarov, F., Zuliani, P.: Probreach: A tool for guaranteed reachability analysis of stochastic hybrid systems. In: SNR. EPiC Series in Computing, vol. 37, pp. 40–48. EasyChair (2015). https://doi.org/10.29007/mh2c
43. Tarski, A.: A decision method for elementary algebra and geometry. The RAND Corporation (1948), https://www.rand.org/pubs/reports/R109.html
44. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific J. Math. **5**(2), 285–309 (1955), https://www.projecteuclid.org/journalArticle/Download?urlId=pjm%2F1103044538
45. Vlachogiannis, J.G., Hatziargyriou, N.D.: Reinforcement learning for reactive power control. IEEE Transactions on Power Systems **19**(3), 1317–1325 (2004). https://doi.org/10.1109/TPWRS.2004.831259
46. Žikelić, D., Lechner, M., Henzinger, T.A., Chatterjee, K.: Learning control policies for stochastic systems with reach-avoid guarantees. In: AAAI. pp. 11926–11935. AAAI Press (2023). https://doi.org/10.1609/aaai.v37i10.26407
47. Wabersich, K.P., Zeilinger, M.N.: A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. Autom. **129**, 109597 (2021). https://doi.org/10.1016/j.automatica.2021.109597
48. Watkins, C.J.C.H.: Learning from Delayed Rewards. Ph.D. thesis, University of Cambridge (1989)
49. Zhao, H., Zhan, N., Kapur, D., Larsen, K.G.: A "hybrid" approach for synthesizing optimal controllers of hybrid systems: A case study of the oil pump industrial example. In: FM. LNCS, vol. 7436, pp. 471–485. Springer (2012). https://doi.org/10.1007/978-3-642-32759-9_38