Safe Reinforcement Learning Using Probabilistic Shields

Nils Jansen
Radboud University, Nijmegen, The Netherlands

Bettina Könighofer
Graz University of Technology, Austria

Sebastian Junges
University of California, Berkeley, CA, USA

Alex Serban
Radboud University, Nijmegen, The Netherlands

Roderick Bloem
Graz University of Technology, Austria

Abstract

This paper concerns the efficient construction of a safety shield for reinforcement learning. We specifically target scenarios that incorporate uncertainty and use Markov decision processes (MDPs) as the underlying model to capture such problems. Reinforcement learning (RL) is a machine learning technique that can determine near-optimal policies in MDPs that may be unknown before exploring the model. However, during exploration, RL is prone to induce behavior that is undesirable or not allowed in safety- or mission-critical contexts. We introduce the concept of a probabilistic shield that enables RL decision-making to adhere to safety constraints with high probability. We employ formal verification to efficiently compute the probabilities of critical decisions within a safety-relevant fragment of the MDP. These results help to realize a shield that, when applied to an RL algorithm, restricts the agent from taking unsafe actions, while optimizing the performance objective. We discuss tradeoffs between sufficient progress in the exploration of the environment and ensuring safety. In our experiments, we demonstrate on the arcade game PAC-MAN and on a case study involving service robots that the learning efficiency increases as the learning needs orders of magnitude fewer episodes.

2012 ACM Subject Classification Computing methodologies → Reinforcement learning; Theory of computation → Verification by model checking; Theory of computation → Reinforcement learning; Computing methodologies → Markov decision processes

Keywords and phrases Safe Reinforcement Learning, Formal Verification, Safe Exploration, Model Checking, Markov Decision Process

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2020.3

Category Invited Paper

Supplementary Material `http://shieldrl.nilsjansen.org`

1 Introduction

Recent years showed increased use of reinforcement learning (RL) in solving tasks such as complex games [60] or robotic manipulation [67]. In RL, an agent perceives the surrounding environment and acts towards maximizing a long-term reward. A major open challenge for systems employing RL is the *safety* of decision-making [61, 30]. In particular during the exploration phase – when an agent chooses random actions in order to examine its surroundings – it is important to avoid actions that may cause unsafe outcomes. The area of *safe exploration* investigates how RL agents may be forced to adhere to safety requirements during this phase [54, 4].

31st International Conference on Concurrency Theory (CONCUR 2020).
Editors: Igor Konnov and Laura Kovács; Article No. 3; pp. 3:1–3:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A suite of methods that deliver theoretical guarantees are so-called *safety-shields* [15]. Shields prevent an agent from taking unsafe actions at runtime. To this end, the performance objective is extended with a constraint specifying that unsafe states should *never* be visited. This new safety objective ensures there are no violations during the exploration phase.

We propose to incorporate constraints that enforce safety violations to occur *only with small probability.* If an action increases the probability of a safety violation by more than a threshold δ with respect to the optimal safety probability, the shield blocks the action. Consequently, an agent augmented with a shield is *guided* to satisfy the safety objective during exploration (or as long as the shield is used). The shield is *adaptive* with respect to δ, as a high value for δ yields a stricter shield, a smaller value a more permissive shield. The value for δ can be changed on-the-fly, and may depend on the individual minimal safety probabilities at each state. Moreover, in case there is not suitable safe action with respect to δ, the shield can always pick the optimal action as a fallback.

We base our formal notion of a probabilistic shield on MDPs, which constitute a popular modeling formalism for decision-making under uncertainty [69] and is widely used in model-based RL. We assess safety by means of probabilistic *temporal logic constraints* [7] that limit, for example, the probability to reach a set of critical states in the MDP.

In order to assess the risk of one action, we (1) construct a behavior model for the environment using model-based RL [26]. By plugging this model into any concrete scenario, we obtain an MDP. To construct the shield, we (2) use a model-based verification technique known as *model checking* [23, 7] that assesses whether a system model satisfies a specification. In particular, we obtain precise *safety probabilities of any possible decision* within the MDP. These probabilities can be looked up efficiently and compared to the threshold δ. The shield then readily (3) augments either model-free or model-based RL.

We identify three key challenges. Firstly, model checking – as any model-based technique – is susceptible to scalability issues. A key advantage of using a separate safety objective is that we may analyze safety on just a fraction of the system, the *safety-critical MDP*. In our experiments, these MDP fragments are at least ten orders of magnitude smaller than a full model of the system, rendering model checking applicable to realistic scenarios. We introduce further optimizations based on problem-specific abstraction techniques.

Secondly, without randomness, all states are either absolutely safe or unsafe. However, in the presence of randomness, safety may be seen as a quantitative measure: in some states certain actions induce a large risk, while others may be considered *relatively* safe. Therefore, we propose an *adaptive* notion of shielding, in which the pre-selection of actions is not based on absolute thresholds.

Lastly, shielding may *restrict* exploration and lead to suboptimal policies. Therefore, it should not be considered in isolation. The trade-off between optimizing the performance objective and the achieved safety is intricate. Intuitively, accepting small short term risks may allow for efficient exploration and limit the risk long-term. To this end, we provide and discuss mechanisms that allow to adjust the shield based on such observations.

We apply shielding to two distinct use cases: the arcade game PAC-MAN and a new case study involving service robots in a warehouse. Shielded RL leads to improved policies for both case studies with fewer safety violations and performance superior to unshielded RL.

Related Work

Safety in RL

Safe RL [33, 54] is concerned with providing safety guarantees for learned agents. Our work focusses on the safe exploration [52], we refer to [33] for other types of safe RL. Using their taxonomy, shielding is an instance of "teacher provides advice" [24], where a teacher with additional information about the system guides the RL agent to pick the right actions. Apprenticeship learning [1] is a closely related variant where the teacher gives (positive) examples and has been used in the context of verification [72]. Some of the work does not assume a model for the environment, making the problem intrinsically harder – and often limiting the safety during exploration. We refer to [52, 31, 21, 32, 37, 36, 38] for some interesting approaches.

Shielding and non-probabilistic specifications

In the remainder of the work, we focus on related work that assumes that the (relevant) dynamics are known. Let us focus on discrete systems first. Exploiting the progress in reactive synthesis [13] provides a shield [44], a maximally permissive policy that contains all actions that will not violate the safety specification. This approach has been shown to be successful in combination with RL [3], and has some noticeable variants. In [68], the temporal specification is extended with an unknown reward function. To enforce complex timing behavior, shields from timed safety properties given as timed automata were considered in [14]. Finally, shielding multi-agent systems is considered in [11], and a shield for almost-sure specifications in partially observable MDPs is introduced in [41].

Shielding and probabilistic specifications

The difference and novel contribution of this paper with the aforementioned papers is rooted in the consideration of stochastic behavior, which is natural to RL. Intuitively, without stochasticities, a learning agent does not take any risk, which is unrealistic in most scenarios. Moreover, often one cannot assume that a 100% (or almost-sure) safety is realizable. A permissive policy for these cases is provided in [29]. Such policies can also be computed from abstract environments [51]. However, as there is no notion of maximally-permissive policies for probabilistic guarentees, multiple permissive policies need to be deployed [42]. Furthermore, the computation of these permissive policies is even more expansive than probabilistic model checking itself, harming scalability. A similar approach to ours was developed independently in [16], but targets a different application area and does not consider scalability issues of formal verification.

Continuous domains

In [71], shielding with qualitative guarantees for continuous domains is discussed. For probabilistic properties and continuous MDPs, additional assumptions are necessary to provide guarantees. Often, these assumption help make some ranking or barrier function [10, 53, 22, 34, 2], and have been extended to work with uncertain specifications [64]. `UPPAAL STRATEGO` provides a number of algorithms combining safety synthesis with optimizing RL for continuous space MDPs [25, 40].

Reinforcement learning in verification

The recent area of programmatic reinforcement learning aims to find (simple) programs rather than policies represented by deep neural networks, such that these programs can be verified [8, 66, 65]. These approaches can be seen as an extension to guided policy synthesis [50]. Such programs have also been used as shields [73]. More generally, Ashok et al. [6, 5] consider *post-processing* controllers into decision trees. Recently, also the verification of recurrent neural network controllers has been investigated [18]. These approaches do not aim to generate permissive shields, but do post-processing to provide guarantees about the final result. Simlarly, methods from reinforcement learning have been successfully employed to improve the scalability of verification methods for MDPs [17, 57, 45] or other areas of formal methods [20, 49].

2 Problem Statement

2.1 Foundations

A *probability distribution* over a countable set X is a function $\mu\colon X \to [0,1]$ with $\sum_{x\in X}\mu(x) = 1$. $\mathit{Distr}(X)$ denotes all distributions on X. The support of $\mu \in \mathit{Distr}(X)$ is $\mathit{supp}(\mu) = \{x \in X \mid \mu(x)>0\}$.

▶ **Definition 1** (MDP). *A* Markov decision process *(MDP)* $\mathcal{M} = (S, \mathit{Act}, \mathcal{P}, r)$ *has a set* S *of* states*, a finite set* Act *of* actions*, a (partial)* probabilistic transition function $\mathcal{P}\colon S \times \mathit{Act} \to \mathit{Distr}(S)$*, and an* immediate reward function $r\colon S \times \mathit{Act} \to \mathbb{R}_{\geq 0}$*. For all* $s \in S$ *the available actions are* $\mathit{Act}(s) = \{\alpha \in \mathit{Act} \mid \mathcal{P}(s,\alpha) \neq \bot\}$ *and we assume* $|\mathit{Act}(s)| \geq 1$*.*

MDPs operate by means of nondeterministic *choices* of actions at each state. A *policy* for an MDP is a function $\sigma\colon S^* \to \mathit{Distr}(\mathit{Act})$ with $\mathit{supp}(\sigma(s_1 \dots s_n)) \subseteq \mathit{Act}(s_n)$ and S^* a finite sequence of states.

In formal methods, safety properties are often specified as *linear temporal logic* (LTL) properties [55]. For an MDP $\mathcal{M}$, probabilistic model checking [43, 47] employs value iteration or linear programming to compute the probabilities of *all states and actions of the MDP* to satisfy an LTL property φ. Specifically, we compute $\eta^{\max}_{\varphi,\mathcal{M}}\colon S \to [0,1]$ or $\eta^{\min}_{\varphi,\mathcal{M}}\colon S \to [0,1]$, which yields for all states the minimal (or maximal) probability over all possible policies to satisfy φ. For instance, for φ encoding to reach a set of states T, $\eta^{\max}_{\varphi,\mathcal{M}}(s)$ describes the maximal probability to "eventually" reach a state in T.

2.2 Setting

We define a setting where one controllable agent (the *avatar*) and a number of uncontrollable agents (the *adversaries*) operate within an *arena*. The arena is a compact, high-level description of the underlying model. From this arena, the potential states and actions of all agents may be inferred. For safety considerations, the reward structure can be neglected, effectively reducing the state space for our model-based safety computations. Formally, an *arena* is a directed graph $G = (V, E)$ with a finite sets V of nodes and $E \subseteq V \times V$ of edges. The agent's *position* is defined via the current node $v \in V$. The agent *decides* on a new edge $(v, v') \in E$ and determines its next position v'. Some (combinations of) agent positions are safety-critical, as they e.g., correspond to collisions or falling off a cliff. A safety property may describe reaching such positions, or use any other property expressible in (the safety fragment of) temporal logic.

While the underlying model for the arena suffices to specify the safe behavior, it is not sufficiently succinct to model the performance via rewards. Consider an edge that is safety-relevant, but the agent is only rewarded the first time taking this edge. In a flat model with rewards, two different edges are necessary to model this behavior. However, the reward (and thus the difference between these edges) is not needed to assess the safety, and the safety-relevant model may be pruned to an exponentially smaller model. We use a *token* function that implicitly extends the underlying model by a reward structure, enabling a separation of concerns between safety and performance.

Technically, we associate edges with a token function $\circ \colon E \to \{0, 1\}$, indicating the status of an edge. Tokens can be (de-) activated and have an associated *reward* earned upon taking edges with an active token.

▶ **Example 2** (Autonomous driving). An autonomous taxi (the avatar) operates within a road network encoded by an arena. The taxi has to visit several points to pick up or drop off passengers [28, 35]. Upon visiting such a point, a corresponding token activates and a reward is earned, afterwards the token is deactivated permanently. Meanwhile, the taxi has to account for other traffic participants or further environmental factors (the adversaries). A sensible safety specification may restrict the probability for collision with other cars to 0.5%. Note that the token structure is not relevant for such a specification.

▶ **Example 3** (Robot logistics in a smart factory). Take a factory floor plan with several corridors with machines. The adversaries are (possibly autonomous) transporters moving parts within the factory. The avatar models a specific service unit moving around and inspecting machines where an issue has been raised (as indicated by a token), while accounting for the behavior of the adversaries. Corridors might be to narrow for multiple (facing) robots, which poses a safety critical situation. The tokens allow to have a *state-dependent* cost, either as long as they are present (indicating the costs of a broken machine) or for removing the tokens (indicating costs for inspecting the machine). A similar scenario has been investigated in [12].

2.3 Problem

Consider an environment described by an arena as above and a safety specification. We assume stochastic behaviors for the adversaries, e.g, obtained using RL [58, 59] in a training environment. In fact, this stochastic behavior determines all actions of the adversaries via probabilities. The underlying model is then an MDP: the avatar executes an action, and upon this execution the next exact positions (the state of the system) are determined stochastically.

We compute a δ-shield that prevents avatar decisions that violate this specification by more than a threshold δ with respect to the optimal safety probability. We evaluate the shield using a model-based or model-free RL avatar that aims to optimize the performance. The shield therefore has to handle an intricate tradeoff between strictly focussing on (short and midterm) safety and performance.

3 Constructing Shields for MDPs

We outline the workflow of our approach in Fig. 1, and describe it below. We employ a separation of concerns between the model-based shield construction and potentially model-free reinforcement learning (RL). First, we construct a *behavior model* for each adversary. Based on this model and a concrete arena, we construct a compact MDP model: the *safety-relevant MDP quotient*. In this MDP, we compute the *shield* which enables safe RL for the full MDP. We now detail the individual technical steps to realize our proposed method.

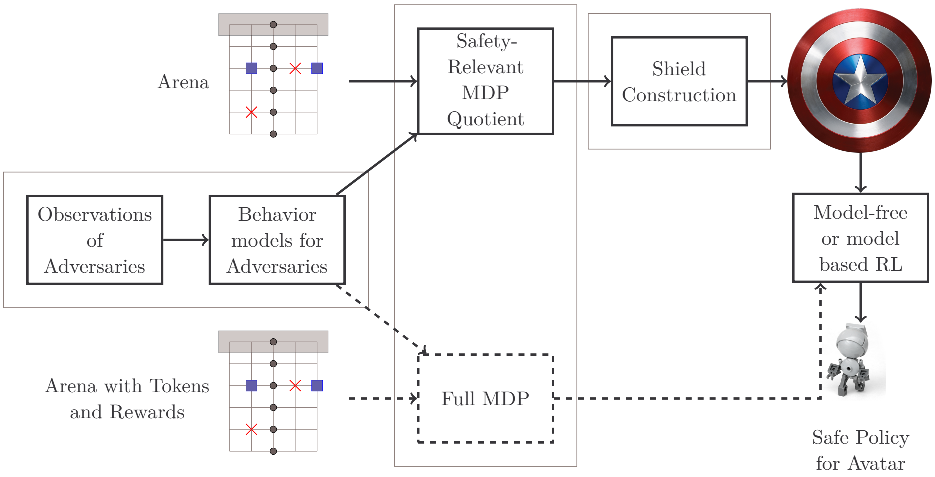
Arena
Safety-Relevant MDP Quotient
Shield Construction
Observations of Adversaries
Behavior models for Adversaries
Model-free or model based RL
Arena with Tokens and Rewards
Full MDP
Safe Policy for Avatar

Figure 1 Workflow of the Shield Construction.

3.1 Behavior Models for Adversaries

We learn an adversary model by observing behavior in a set of similar, but smaller arenas, until we gain sufficient confidence that more training data would not change the behavior significantly [58]. To reduce the size of the training set, we devise a data augmentation technique using domain knowledge of the arenas [46, 70]. In particular, we abstract away from the precise configuration of the arena by partitioning the graph into zones that are relative to the view-point of the adversary (e. g., near or far, north or south, east or west). The intuitive assumption is that the specific position of an adversary is not important, but some key information is (e.g., the relation to the position of the avatar). This approach (1) speeds up the learning process and (2) renders the resulting behavior model applicable for varying the concrete instance of the same setting.

Zones are uniquely identified by a coloring with a finite set C of colors. Formally, for an arena $G = (V, E)$, *zones relative to a node* $v \in V$ are given by a function $z_v \colon V \to C$. For nodes $x, y \in V$, with $z_v(x) = z_v(y)$, the assumption is that the adversary in v behaves similarly regardless whether the avatar is in x or y. From our observations, we extract a *histogram* $h \colon E \times C \to \mathbb{N}$, where $h(e, c)$ describes how often the adversary takes an edge $e = (v, v') \in E$ while the avatar is in a node u with $z_v(u) = c$. We translate these likelihoods into distributions over possible edges in the arena.

▶ **Definition 4** (Adversary Behavior). *For an arena $G = (V, E)$, zones $z_u \colon V \to C$ for every $u \in V$, and a histogram $h \colon E \times C \to \mathbb{N}$, the* adversary behavior *is a function $B \colon V \times C \to Distr(E)$ with*

$$B(v, c) = \frac{h\big((v, v'), c\big)}{\sum_{(v,v')\in E} h\big((v, v'), c\big)} \;.$$

While we employ a simple normalization of likelihoods, alternatively one may also utilize, e. g., a softmax function which is adjustable to favor more or less likely decisions [62].

3.2 Safety-Relevant Quotient MDP

The construction of the MDP $\mathcal{M} = (S, Act, \mathcal{P})$ augments an arena by behavior models B_i. First, the *states* $S = V^{m+1} \times \{0, \dots, m\}$ encode the positions for all agents and whose turn it is. The *decision states* of the safety-relevant MDP $\mathcal{M}$ are $S_d = \{s_d \in S \mid s_d = (\dots, 0)\}$, i.e., it's the turn of the avatar. The *actions* $Act = \{\alpha_0\} \cup Act_E$ with $Act_E = \{\alpha_e \mid e \in E\}$ determine the movements of the avatar and the adversaries. For $(v, \ldots, 0) = s_d \in S_d$ (the avatar moves next), the available actions are $\alpha_e \in Act(s_d) \subseteq Act_e$, where α_e corresponds to an outgoing edge of v. For $(v, \ldots, 0) = s_d \in S_d$, α_e with $e = (v, v')$ leads with probability one to a state $s_e = (v', \ldots, 1)$. For $(v, \ldots, v_i, \ldots, i > 0)$ (an adversary moves next), there is a unique action α_0 where v_i is changed to v_i', randomly determined according to the behavior B_i, which also updates i to $i + 1$ modulo m. These transitions induce the only probabilistic choices in the MDP. A policy only has to choose an action at decision states. At all other states, only the unique action α_0 emanates. Consequently, a policy for $\mathcal{M}$ is a policy for the avatar.

Under the assumption that the reward function is known and not discovered during the exploration of the MDP, one can build the full MDP for the arena (V, E) and the token function $\circ \colon E \to \{0, 1\}$. Then, one can compute the reward-optimal and safe policy without need for further learning techniques. As there are 2^E token configurations, the state space blows up exponentially, which prevents the successful application of model checking or planning techniques for anything but very small examples.

3.3 Shield Construction

For the safety-relevant MDP $\mathcal{M}$, a set of unsafe states $T \subseteq S$ should preferably not be reached from any state. The property $\varphi = \lozenge T$ encodes the **violation** of this safety constraint, that is, eventually reaching T within $\mathcal{M}$. The shield needs to limit the probability to *satisfy* φ. We evaluate all decision states $s_d \in S_d$ with respect to this probability: We compute $\eta^{\min}_{\varphi,\mathcal{M}}(s_e)$, i.e., the minimal probability to satisfy φ from s_e, which is the state reached after taking action $\alpha_e \in Act_e$ in s_d.

▶ **Definition 5** (Action-valuation). *An* action-valuation *for each available action $\alpha_e \in Act_e$ at each decision state $s_d \in S_d$ is given by*

$$\mathrm{val}^{\mathcal{M}}_{s_d} \colon Act(s_d) \to [0, 1], \text{ with } \mathrm{val}^{\mathcal{M}}_{s_d}(\alpha_e) = \eta^{\min}_{\varphi,\mathcal{M}}(s_e) \ .$$

The optimal action-value *for s_d is $optval^{\mathcal{M}}_{s_d} = \min_{\alpha' \in Act} \mathrm{val}^{\mathcal{M}}_{s_d}(\alpha')$, the set of all action-valuations at s_d is $ActVals_{s_d}$.*

We now define a shield for the safety-relevant MDP $\mathcal{M}$ using the action values. Specifically, a δ*-shield* for $\delta \in [0, 1]$ determines a set of actions at each decision state s_d that are δ-optimal for the specification φ. All other actions are "shielded" or "blocked".

▶ **Definition 6** (Shield). *For action-valuation $\mathrm{val}^{\mathcal{M}}_{s_d}$ and $\delta \in [0, 1]$, a* δ-shield for state $s_d \in S_d$ *is given by*

$$shield^{s_d}_{\delta} \colon ActVals_{s_d} \to 2^{Act(s_d)}$$

with $shield^{s_d}_{\delta} \mapsto \{\alpha \in Act(s_d) \mid \delta \cdot \mathrm{val}^{\mathcal{M}}_{s_d}(\alpha) \le optval^{\mathcal{M}}_{s_d}\}$.

Intuitively, δ enforces a constraint on actions that are acceptable with respect to the optimal probability. The shield is *adaptive* with respect to δ, as a high value for δ yields a stricter shield, a smaller value a more permissive shield. The shield is stored using a lookup-table, and the value for δ can then be changed on-the-fly. In particularly critical situations, the shield can enforce the decision-maker to resort to (only) the optimal actions w.r.t. the safety objective. A δ-shield for the MDP $\mathcal{M}$ is built by constructing and applying δ-shields to all decision states.

▶ **Definition 7** (Shielded MDP). *The* shielded MDP $\mathcal{M}_{\sqcup} = (S, Act\,\mathcal{P}_{\sqcup})$ *for a safety-relevant quotient MDP* $\mathcal{M} = (S, Act, \mathcal{P})$ *and a δ-shield for all $s_d \in S_d$ is given by the transition probability $\mathcal{P}_{\sqcup}$ with $\mathcal{P}_{\sqcup}(s,\alpha) = \mathcal{P}(s,\alpha)$ if $\alpha \in shield^{s}_{\delta}(\mathit{val}^{\mathcal{M}}_{s})$ and $\mathcal{P}_{\sqcup}(s,\alpha) = \perp$ otherwise.*

▶ **Lemma 8.** *If MDP $\mathcal{M}$ is deadlock-free if and only if the shielded MDP $\mathcal{M}_{\sqcup}$ is deadlock-free.*

We compute the shield relative to optimal values $\mathit{optval}^{\mathcal{M}}_{s_d}$. Consequently, for $\delta = 1$, only optimal actions are preserved, and for $\delta = 0$ no actions are blocked.

▶ **Theorem 9.** *For an MDP $\mathcal{M}$ and a δ-shield, it holds for any state s that $\mathit{val}^{\mathcal{M}}_{s} = \mathit{val}^{\mathcal{M}_{\sqcup}}_{s}$.*

As optimal actions for the safety objective are not removed, optimality w.r.t. safety is preserved in the shielded MDP. Thus, during construction of the shield, we compute the action-valuations in fact *for the shielded MDP*. Observe that computing a shield for a state is done *independently* from the application of the shield to other states.

3.4 Guaranteed Safety

A δ-shield ensures that only actions that are δ-optimal with respect to an LTL property φ are allowed. In particular, for each action $\alpha \in Act_e$ at state s_e, we use the *minimal* probability $\eta^{\min}_{\varphi,\mathcal{M}}(s_e)$ to satisfy φ, see Def. 5. Under *optimal* (subsequent) choices, the value $\eta^{\min}_{\varphi,\mathcal{M}}(s_e)$ will be achieved. In contrast, a sequence of bad choices may violate φ with high probability. A more conservative notion would be to use the minimal action value while assuming that in all subsequent states the worst-case decisions corresponding to the maximal probabilities are taken. These values are computable by model checking. Regardless of subsequent choices, at least $\mathit{val}^{\mathcal{M}}_{s_d}(\alpha_e)$ is then guaranteed. A sensible notion to construct a shield would then be to impose a threshold $\lambda \in [0,1]$ such that only actions with $\mathit{val}^{\mathcal{M}}_{s_d}(\alpha_e) \leq \lambda$ are allowed. A shield with such a guaranteed safety probability may induce a shielded MDP (Def. 7) that is *not deadlock free*. Moreover, the shield may become too restrictive for the agent.

3.5 Scalable Shield Construction

Although we apply model checking only in the safety-relevant MDP, scalability issues for large applications remain. We employ several optimizations towards computational tractability.

Finite Horizon

For infinite horizon properties, the probability to violate safety (in the long run) is often one in our examples. Furthermore, our learned MDP model is inherently an approximation of the real world. Errors originating from this approximation accumulate for growing horizons. Thus, we focus on a finite horizon such that the action values (and consequently, a policy for the avatar) carry only guarantees for the next steps. This assumption also allows us to prune the safety-relevant MDP (see below), increasing the scalability.

Piecewise Construction

Computing a shield for all states in an MDP concurrently yields a large memory footprint. To alleviate this footprint, we compute the shield states independently, in accordance with Theorem 9. The independent computation prunes the relevant part of the MDP, as the number of states reachable within the horizon is drastically reduced. Additionally, the independent computation allows for parallelizing the computation.

Independent Agents

The explosion of state spaces stems mostly from the number of agents. Here, an important observation is that we can consider agents independently. For instance, the probability for the avatar to crash with an adversary is stochastically independent from crashing with the others. Instead of determining the shield for all adversaries at once, we perform computations for each agent individually, and combine them via the inclusion-exclusion principle. Afterwards, the shield is composed from the shields dedicated to individual adversaries.

Abstractions

We observe that for finite horizon properties and piecewise construction, adversaries may be far away – beyond the horizon – without a chance to reach the avatar. We do not need to consider such (positions for) adversaries, as in these states, the shield will not block any actions.

Fewer Decision States. Depending on the setting, there might be only a few critical situations in which the agent requires shielding to ensure safety. The shield can be computed for this critical states only. Consequently, the agent makes shielded decisions in the adapted decision states, and unshielded decisions in all other ones.

Shielding versus Performance. A shield which is *minimally invasive* gives the RL agent the most freedom to optimize the performance objective. We propose two methods to alleviate invasiveness, all of them assume *domain knowledge* of the rationale behind the decision procedure.

Iterative Weakening. During runtime, we may observe that the progress of the avatar regarding the performance objective is not increasing anymore. Then, we weaken the shield by $\delta - \varepsilon$, allowing additional actions. As soon as progress is made, we reset δ to its former value. The adaption of shield^s_δ to $\mathit{shield}^s_{\delta-\varepsilon}$ can be done on the fly, without new computations.

Adapted Specifications. If the goal of the decision maker is known *and* can be captured in temporal logic, we may adapt the original specification accordingly. There are often natural trade-offs between safety and performance. These trade-offs might be resolved via weights, but this process is often undesirable [56] and similar to reward engineering. Instead, optimizing the conditional performance while assuming to stay sufficiently safe [63], avoids side-effects of attaching some weights to the safety specification.

4 Implementation and Numerical Experiments

4.1 Set-up

We run experiments using an Intel Core i7-4790K CPU with 16 GB of RAM using 4 cores. We give the timing results for a single CPU. Since the shield may be computed in a multi-threaded architecture, this time can be divided by the number of cores available.

The supplementary materials, namely the source code and videos are available online[1].

We demonstrate the applicability of our approach by means of two case studies. For both case studies, we learn the adversary behavior in small arenas, individually for each adversary. These behavior models are applicable to any benchmark instance, as they are independent of concrete positions.

[1] http://shieldrl.nilsjansen.org

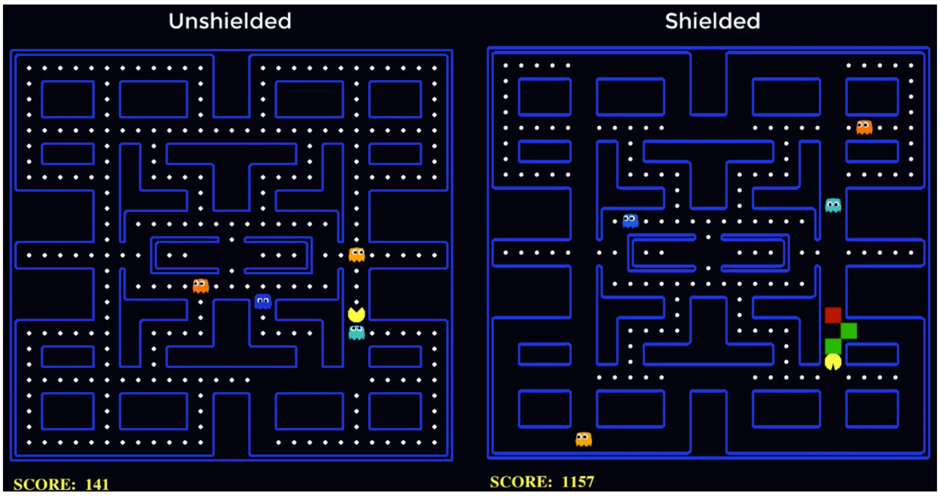

Figure 2 Video still for classic PAC-MAN.

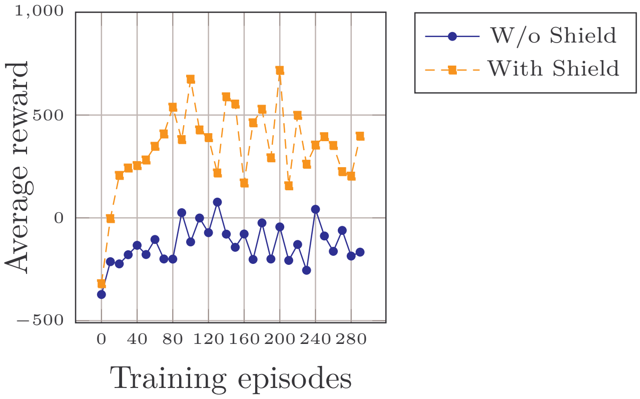

Figure 3 Training scores classic PAC-MAN.

For the arcade game PAC-MAN, PM (the avatar) aims to collect *PAC-dots* in a *maze* and not get caught by *ghosts* (the adversaries). We model various instance of the game (with different sizes) as an arena, where tokens represent the dots at each position in the maze, such that a dot is either present or collected. The score (reward, performance) is positively affected (+10) by collecting a dot and negatively by time (each step: -1). If PM either collects all dots (+500) or is caught (-500), the game is restarted. RL approaches exist [9], but they suffer from the fact that during the exploration phase PM is often caught by the ghosts, achieving very poor scores. The safety specification places a lower bound on the probability of reaching states in the underlying MDP that correspond to being caught.

We also consider a warehouse floor plan with several corridors. A similar scenario has been investigated in [12]. In the arena, nodes describe crossings, the edges the corridors with shelves, and the distances the corridor length. The agents are fork-lift units picking up packages from the shelves and delivering them to the exit; tokens represent the presence of a package at its position. The avatar is a specific (yellow) fork-lift unit that has to account for other units, the adversaries. The performance (reward) is positively affected by loading and delivering packages (+20, respectively) and negatively by time (each step: -1). Delivering all

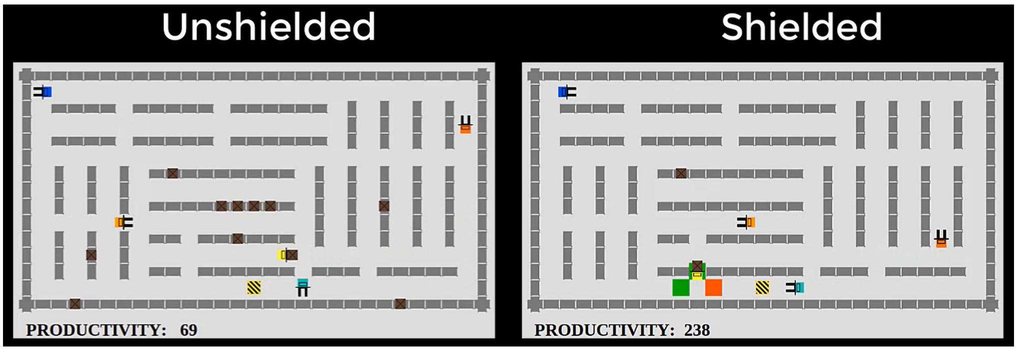

Figure 4 Video still for warehouse.

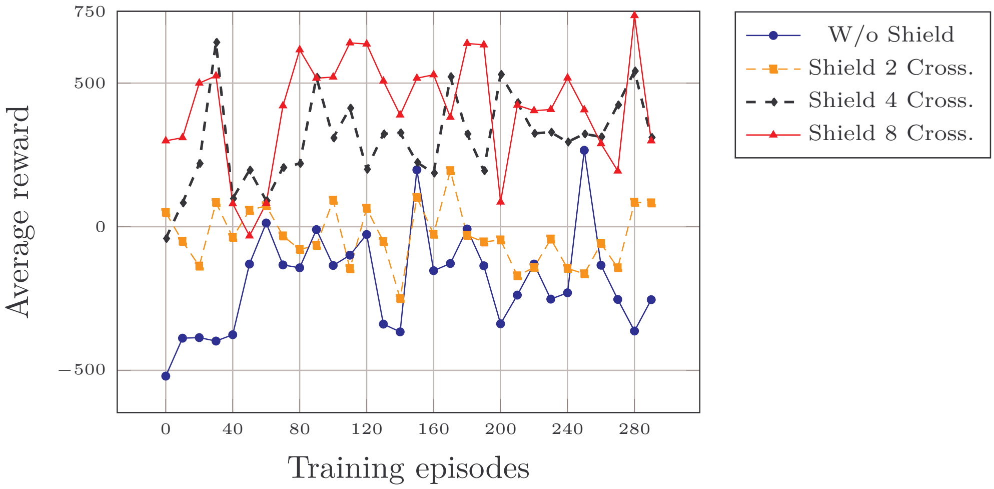

Figure 5 Training scores warehouse.

packages yields a large bonus (+500) and a collision leads to a large punishment (-500), both cases end the scenario. Corridors might be too narrow for multiple (facing) units, which poses a safety-critical situation. Most crucial is the crowded area near the exit, since all units have to deliver the packages to the exit.

Transferring the stochastic adversary behavior to any arena (without tokens) yields a concrete safety-relevant MDP. In particular, we specify an arena with the positions of the avatar and the adversaries as well as their behavior in the high-level `PRISM`-language [48]. We employ a script that automatically generates arenas to enable a broad set of benchmarks. Taking, e.g., the PAC-MAN arena from Fig. 2, the considered MDP has roughly 10^{12} states (compared to 10^{50} for the full MDP). For a safety-relevant MDP, we compute a δ-shield (with iterative weakening) via the model checker `Storm` [27], using a horizon of 10 steps. The immense size even of safety-relevant MDPs requires optimizations such as a piecewise and independent shield construction. Moreover, a multi-threaded architecture lets us construct shields for very large examples. In particular, we perform model checking for (many) MDPs of roughly 10^6 states. The computation time for the largest PM instance takes about 6 hours (single-threaded), while memory is not an issue due to the piecewise shield construction.

Table 1 Average scores and win rates for PM.

Size, #Ghosts	#Model Checking	time (s)	Score w/o Shield	Score w. Shield	Win Rate w/o Shield	Win Rate w. Shield
9x7,1	5912	584	-359,6	535,3	0,04	0,84
17x6,2	5841	1072	-195,6	253,9	0,04	0,4
17x10,3	51732	3681	-220,79	-40,52	0,01	0,07
27x25,4	269426	19941	-129,25	339,89	0,00	0,00

We compare RL to shielded RL on different instances. The key comparison criterion is the performance (detailed above) during learning. Our implementation is based on an existing PAC-MAN environment[2] using an approximate Q-learning agent [62] with the following feature vectors:

- for PAC-MAN: (1) distance to the closest dot, (2) whether a ghost collision is imminent, and (3) whether a ghost is one step away.
- for Warehouse: (1) has the unit loaded or unloaded, (2) the distance to the next package and (3) to the exit, (4) whether another unit is three steps away and (5) one step away.

The results are basic reflex controllers. The Q-learning uses the learning rate $\alpha = 0.2$ and the discount factor $\gamma = 0.8$ for the Q-update and an ϵ-greedy exploration policy with $\epsilon = 0.05$. One episode lasts until either the game is restarted. We describe results for the training phase of RL (300 episodes).

4.2 Results

Figures 2 and 4 show screenshots of a series of recommended videos (available in the supplementary material). Each video compares how RL performs either shielded or unshielded on a instance of the case study. In the shielded version, at each decision state in the underlying MDP, we indicate the risk of decisions from low to high by the colors green, orange, red.

Consider PAC-MAN in detail: Figure 3 depicts the scores obtained during RL. The curves (blue, solid: unshielded, orange, dashed: shielded) show the average scores for every ten training episodes. Table 1 shows results for instances in increasing size. We list the number of model checking calls and the time to construct the shield. We list the scores with and without shield, and the *winning rate* capturing the ratio of successfully ended episodes. For all instances, we see a large difference in scores due to the fact that PM is often rescued by the shield. The winning rates differ for most benchmarks, favoring shielded RL. For three or four ghosts, a shield with a ten-step horizon cannot guide PM to avoid being encircled by the ghosts long enough to successfully end the game. Nevertheless, the shield often safes PM, leading to superior scores. Moreover, the shield helps learning an optimal policy much faster as fewer restarts are needed.

For the warehouse case study, we choose to vary the decision states, i.e., the positions of the avatar for which we compute a shield. We present results for shielding the 2–8 crossings closest to the exit. Figure 5 shows the average score for the different variants, Table 2 summarizes average score and win rate. Unsurprisingly, the score gets better the more states are shielded. Furthermore, we have seen that shielding even more states has only a very limited effect.

[2] http://ai.berkeley.edu/project_overview.html

Table 2 Average scores and win rates for warehouse.

Crossings shielded	0	2	4	8
Score	-186	-27.6	303	420
Win Rate	0.16	0.31	0.59	0.71

5 Conclusion and Future Work

We developed the concept of shields for MDPs. Utilizing probabilistic model checking, we maintained probabilistic safety measures during reinforcement learning. We addressed inherent scalability issues and provided means to deal with typical trade-off between safety and performance. Our experiments showed that we improved the state-of-the-art in safe reinforcement learning.

For future work, we will extend the applications to more arcade games and employ deep recurrent neural networks as means of decision-making [39, 19, 18]. Another interesting direction is to explore (possibly model-free) learning of shields, instead of employing model-based model checking.

References

1 Pieter Abbeel and Andrew Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 1–8. ACM, 2005.

2 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.

3 Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *AAAI*, pages 2669–2678. AAAI Press, 2018.

4 Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016. arXiv:1606.06565.

5 Pranav Ashok, Mathias Jackermeier, Pushpak Jagtap, Jan Kretínský, Maximilian Weininger, and Majid Zamani. dtcontrol: decision tree learning algorithms for controller representation. In *HSCC*, pages 30:1–30:2. ACM, 2020.

6 Pranav Ashok, Jan Kretínský, Kim Guldstrand Larsen, Adrien Le Coënt, Jakob Haahr Taankvist, and Maximilian Weininger. SOS: safe, optimal and small strategies for hybrid markov decision processes. In *QEST*, volume 11785 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2019.

7 Christel Baier and J.P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

8 Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *NeurIPS*, pages 2499–2509, 2018.

9 UC Berkeley. Intro to AI – Reinforcement Learning , 2018. URL: http://ai.berkeley.edu/reinforcement.html.

10 Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *NIPS*, pages 908–919, 2017.

11 Suda Bharadwaj, Roderick Bloem, Rayna Dimitrova, Bettina Könighofer, and Ufuk Topcu. Synthesis of minimum-cost shields for multi-agent systems. In *ACC*, pages 1048–1055. IEEE, 2019.

12 Arthur Bit-Monnot, Francesco Leofante, Luca Pulina, Erika Ábrahám, and Armando Tacchella. Smartplan: a task planner for smart factories. *CoRR*, abs/1806.07135, 2018. arXiv:1806.07135.

13 Roderick Bloem, Krishnendu Chatterjee, and Barbara Jobstmann. Graph games and reactive synthesis. In *Handbook of Model Checking*, pages 921–962. Springer, 2018.

14 Roderick Bloem, Peter Gjøl Jensen, Bettina Könighofer, Kim Guldstrand Larsen, Florian Lorber, and Alexander Palmisano. It's time to play safe: Shield synthesis for timed systems. *CoRR*, abs/2006.16688, 2020. arXiv:2006.16688.

15 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.

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

17 Tomáš Brázdil, Krishnendu Chatterjee, Martin Chmelík, Vojtěch Forejt, Jan Křetínský, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. Verification of Markov decision processes using learning algorithms. In *ATVA*, 2014.

18 Steven Carr, Nils Jansen, and Ufuk Topcu. Verifiable rnn-based policies for pomdps under temporal logic constraints. *CoRR*, abs/2002.05615, 2020. arXiv:2002.05615.

19 Steven Carr, Nils Jansen, Ralf Wimmer, Alexandru Constantin Serban, Bernd Becker, and Ufuk Topcu. Counterexample-guided strategy improvement for pomdps using recurrent neural networks. In *IJCAI*, pages 5532–5539. ijcai.org, 2019.

20 Jia Chen, Jiayi Wei, Yu Feng, Osbert Bastani, and Isil Dillig. Relational verification using reinforcement learning. *Proc. ACM Program. Lang.*, 3(OOPSLA):141:1–141:30, 2019.

21 Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. *AAAI*, 2019.

22 Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A Lyapunov-based approach to safe reinforcement learning. In *NIPS*, pages 8103–8112, 2018.

23 Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model checking.* MIT Press, 2001.

24 Jeffery A. Clouse and Paul E. Utgoff. A teaching method for reinforcement learning. In *ML*, pages 92–110. Morgan Kaufmann, 1992.

25 Alexandre David, Peter Gjøl Jensen, Kim Guldstrand Larsen, Marius Mikucionis, and Jakob Haahr Taankvist. Uppaal stratego. In *TACAS*, volume 9035 of *LNCS*, pages 206–211. Springer, 2015.

26 Peter Dayan and Yael Niv. Reinforcement learning: the good, the bad and the ugly. *Current opinion in neurobiology*, 18(2):185–196, 2008.

27 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.

28 Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

29 Klaus Dräger, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. Permissive controller synthesis for probabilistic systems. *Log. Methods Comput. Sci.*, 11(2), 2015.

30 Richard G Freedman and Shlomo Zilberstein. Safety in AI-HRI: Challenges complementing user experience quality. In *AAAI Fall Symposium Series*, 2016.

31 Jie Fu and Ufuk Topcu. Probably approximately correct mdp learning and control with temporal logic constraints. In *RSS*, 2014.

32 Nathan Fulton and André Platzer. *Verifiably Safe Off-Model Reinforcement Learning*, volume 11427 of *Lecture Notes in Computer Science*, pages 413–430. Springer, 2019.

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

34 Javier García and Fernando Fernández. Probabilistic policy reuse for safe reinforcement learning. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 13(3):14, 2019.

35 OpenAI Gym. Taxi-v2, 2018. URL: `https://gym.openai.com/envs/Taxi-v2/`.

36 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 *LNCS*, pages 395–412. Springer, 2019.

37 Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Logically-correct reinforcement learning. *CoRR*, abs/1801.08099, 2018. `arXiv:1801.08099`.

38 Mohammadhosein Hasanbeig, Yiannis Kantaros, Alessandro Abate, Daniel Kroening, George J. Pappas, and Insup Lee. Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. In *CDC*, pages 5338–5343. IEEE, 2019.

39 Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527*, 2015. `arXiv:1507.06527`.

40 Manfred Jaeger, Peter Gjøl Jensen, Kim Guldstrand Larsen, Axel Legay, Sean Sedwards, and Jakob Haahr Taankvist. Teaching stratego to play ball: Optimal synthesis for continuous space mdps. In *ATVA*, volume 11781 of *Lecture Notes in Computer Science*, pages 81–97. Springer, 2019.

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

42 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.

43 Joost-Pieter Katoen. The probabilistic model checking landscape. In *LICS*, pages 31–45. ACM, 2016.

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

45 Jan Kretínský, Guillermo A. Pérez, and Jean-François Raskin. Learning-based mean-payoff optimization in an unknown MDP under omega-regular constraints. In *CONCUR*, volume 118 of *LIPIcs*, pages 8:1–8:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

46 Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

47 Marta Z. Kwiatkowska. Model checking for probability and time: from theory to practice. In *LICS*, page 351. IEEE CS, 2003.

48 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

49 Gil Lederman, Markus N. Rabe, Sanjit Seshia, and Edward A. Lee. Learning heuristics for quantified boolean formulas through reinforcement learning. In *ICLR*. OpenReview.net, 2020.

50 Sergey Levine and Vladlen Koltun. Guided policy search. In *ICML (3)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1–9. JMLR.org, 2013.

51 George Mason, Radu Calinescu, Daniel Kudenko, and Alec Banks. Assured reinforcement learning with formally verified abstract policies. In *ICAART (2)*, pages 105–117. SciTePress, 2017.

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

53 M. Ohnishi, L. Wang, G. Notomista, and M. Egerstedt. Barrier-certified adaptive reinforcement learning with applications to brushbot navigation. *IEEE Transactions on Robotics*, pages 1–20, 2019.

54 Martin Pecka and Tomas Svoboda. Safe exploration techniques for reinforcement learning–an overview. In *MESAS*, pages 357–375. Springer, 2014.

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

56 Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *J. Artif. Intell. Res.*, 48:67–113, 2013.

57 Dorsa Sadigh, Eric S Kim, Samuel Coogan, S Shankar Sastry, and Sanjit A Seshia. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *CDC*, pages 1091–1096. IEEE, 2014.

58 Dorsa Sadigh, Nick Landolfi, Shankar S Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state. *Autonomous Robots*, 42(7):1405–1426, 2018.

59 Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, 2016.

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

61 Ion Stoica, Dawn Song, Raluca Ada Popa, David Patterson, Michael W Mahoney, Randy Katz, Anthony D Joseph, Michael Jordan, Joseph M Hellerstein, Joseph E Gonzalez, et al. A berkeley view of systems challenges for AI. *CoRR*, abs/1712.05855, 2017. `arXiv:1712.05855`.

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

63 Florent Teichteil-Königsbuch. Stochastic safest and shortest path problems. In *AAAI*. AAAI Press, 2012.

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

65 Abhinav Verma, Hoang Minh Le, Yisong Yue, and Swarat Chaudhuri. Imitation-projected programmatic reinforcement learning. In *NeurIPS*, pages 15726–15737, 2019.

66 Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 5052–5061. PMLR, 2018.

67 Angelina Wang, Thanard Kurutach, Kara Liu, Pieter Abbeel, and Aviv Tamar. Learning robotic manipulation through visual planning and acting. arXiv preprint, 2019. `arXiv:1905.04411`.

68 Min Wen, Rüdiger Ehlers, and Ufuk Topcu. Correct-by-synthesis reinforcement learning with temporal logic constraints. In *IROS*, 2015.

69 Douglas J White. Real applications of Markov decision processes. *Interfaces*, 15(6):73–83, 1985.

70 Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2016.

71 Meng Wu, Jingbo Wang, Jyotirmoy Deshmukh, and Chao Wang. Shield synthesis for real: Enforcing safety in cyber-physical systems. In *FMCAD*, pages 129–137. IEEE, 2019.

72 Weichao Zhou and Wenchao Li. Safety-aware apprenticeship learning. In *CAV (1)*, volume 10981 of *Lecture Notes in Computer Science*, pages 662–680. Springer, 2018.

73 He Zhu, Zikang Xiong, Stephen Magill, and Suresh Jagannathan. An inductive synthesis framework for verifiable reinforcement learning. In *PLDI*, pages 686–701. ACM, 2019.