Final Report, Formal Methods for Learned Systems 1–10, 2023

Safe RL in Unknown Stochastic Environments with Online Shielding Synthesis

Pengzhi Yang P.YANG-4@STUDENT.TUDELFT.NL
Master Student from Faculty of Electrical Engineering, Mathematics and Computer Science, Tu Delft

Abstract

Deep reinforcement learning (DRL) has made significant advancements in recent times. Nonetheless, ensuring safety remains a significant challenge that hinders the application of DRL in safety-critical domains such as autonomous driving and robotics. To tackle this issue, I examine shielding techniques that guarantee the safety and stability of DRL systems in the context of the Grid-world example. Previous studies have demonstrated that the construction of a shielding mechanism that guarantees safety requires a full understanding of a deterministic or known environment, which is impractical in real-world scenarios. In this report, I use the Grid-world example to investigate the limitations of existing shielding approaches and propose a preliminary concept for constructing DRL shields in an online manner.

Keywords: Safe Reinforcement Learning, Shielding, Online Learning

1. Introduction

In recent years, researchers have been trying to apply reinforcement learning algorithms to various real-world tasks, especially sequential decision-making problems (Sutton and Barto, 2018). To solve the explosion of continuous states problem, Mnih et al. (2015) combined deep neural networks as the feature extractor. The trained model achieved human-level control ability in multiple video games. Silver et al. (2017) introduced Monte-Carlo Tree Search (MCTS) to iterate and optimize the agent policy with the data collected by self-play. However, many scenarios also request agents' ability to interact and cooperate with each other to fulfill complex tasks. To tackle this problem, Vinyals et al. (2019) proposed to train the network with a league of agents. By leveraging the league method, the system is able to find a mixed Nash equilibrium in cooperative tasks.

Besides the developments of RL in various game scenarios, more safe-critical fields like autonomous robotics and driving research also involve DRL for better planning and control. Researchers have worked on applying reinforcement learning on manipulations - to search and execute good policies for grabbing different kinds of objects with robotic arms and hands (Kober et al., 2013; Kober and Peters, 2008). End-to-end deep reinforcement learning methods have been applied to quadrotors Song et al. (2021a,b). The training of drones to navigate a sequence of square gates quickly and safely will be accomplished using visual data captured by a single camera and state estimations from an Inertial Measurement Unit (IMU). To address the sample inefficiency issue in DRL, policies are initially trained in a simulated environment (Loquercio et al., 2019) to achieve a robust racing policy. However, the problem of effectively transferring these policies to the real world remains unresolved. This problem has been explored in quadruped robots, where DRL-based controllers were developed to drive them with agile motor skills (Hwangbo et al., 2019). Then, Peng et al. (2020) creatively employed a search process during deployment to search over the dynamic

© 2023 P. Yang.

settings to find the best adaptation. Although these methods have helped with generalizability, they still suffer from unsafe or unstable policies in unseen or unknown environments (Garcıa and Fernández, 2015).

Deep neural networks (DNN) are commonly used as function approximators in DRL systems, but they are often seen as black boxes that can lead to the mentioned unsafe problems. To address this issue, Katz et al. (2017) proposed using Reluplex, an efficient satisfiability modulo theories (SMT) solver, to analyze DNNs as a whole system and prove their properties. Comparatively, Polydoros and Nalpantidis (2017) delved into the use of model-based reinforcement learning (RL) in robotics and highlighted its potential in aiding model-free RL in unfamiliar environments. However, acquiring a precise model in real-world situations is often infeasible, and even a suboptimal model can negatively impact the overall performance. Besides, Neary et al. (2022) devised a compositional RL system that established performance benchmarks for each subtask that the trained RL subsystems must meet. These subsystems then computed a meta-policy to ensure that the overall task specifications were satisfied by the compositional system, which proved to be a relatively simpler method. Some researchers have employed reachability sets as a means of verification to establish limits that aid agents in meeting time-varying state constraints over a period (Yu et al., 2022; Bansal et al., 2017). Shao et al. (2021) executed a trajectory safeguarding mechanism that centered around reachability sets and utilized continuous controllers. Bajcsy et al. (2019) also recommended updating safe sets during the agent's exploration in an unknown environment.

Comparatively, as one of the most effective methods to enforce safety properties during run time, Shielding provides safe guarantees in a more direct way by isolating the agents from forbidden actions based on specifications (Könighofer et al., 2017; Alshiekh et al., 2018). Afterward, research has been conducted to apply Shields on autonomous vehicles (Zoon, 2021) and robotics (Bastani et al., 2021). However, these shielding approaches assume the agents observing the environment with full and precise knowledge. Thus, Carr et al. (2022) provided formal safety guarantees with partial observations. Besides ensuring system safety, their proposed method also improves the training efficiency by reducing the search space as well as the required data to train a robust model. Methods proposed by Könighofer et al. (2022), which is similar to my preliminary idea, would compute shields in an online manner. However, they only maintain the shielding with states that could be reached in near future steps. Besides, in their scenario, the agent only needs to make a decision at a few *decision states* which only performs as a simplified abstraction of real-world scenarios. Li and Bastani (2020) created a Shielding mechanism in a stochastic environment. But it requests a known dynamical model. Furthermore, researchers are also trying to apply Shield on multi-agent scenarios which could extend its practical use in more RL systems (ElSayed-Aly et al., 2021).

Main work: This report is summarized below[1]:

- In this study, I analyze the approach proposed in the paper *Safe Reinforcement Learning via Shielding* by using the Grid-world example as a case study (Alshiekh et al., 2018). I conclude that their shielding method unfairly exhibits more knowledge of the environment than their learning model. Besides, existing shielding methods cannot perform well in stochastic unknown environments.

- To address the second aforementioned issue, I present a preliminary concept for online shielding construction. The proposed method considers not only the provided specifications of the

1. Code: https://github.com/pengzhi1998/safe-rl-shielding (adjusted based on Alshiekh et al. (2018)'s source code)

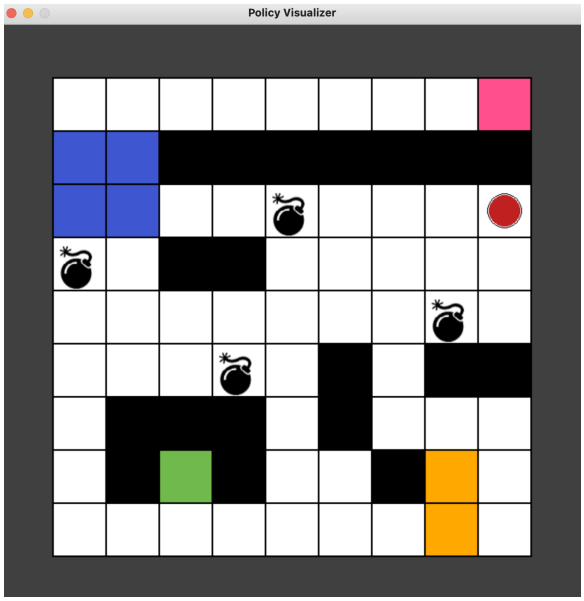

Figure 1: Simulated Grid-world environment illustrated with Pygame Visualizer.

Grid-world example but also incorporates updated information during training. The report includes an analysis of this approach as well as the expected outcomes.

2. Problem Formulation with Grid-world Example

2.1. Preliminaries

2.1.1. MDPs

We consider a discrete Markov Decision Process (MDP) problem:

$$\mathcal{M} := \langle \mathcal{S}, \mathcal{A}, \rho, P, R \rangle, \tag{1}$$

where $\mathcal{S}$ and $\mathcal{A}$ denote state and action sets respectively, $\rho(s_0)$ denotes the initial state distribution. $P(s'|S, A)$ and $R(r|S', S, A)$ denote the transition model and reward model. The control action at timestep t is given by the policy and current state s:

$$a_t = \pi(s_t) \tag{2}$$

The objective of the RL task is to find the optimal policy π^* that maximizes the expected reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{r \sim R(r|S',A,S)} \Big[\sum \gamma^t r_t \Big], \tag{3}$$

where $\gamma \in [0, 1]$ is the discount factor.

2.1.2. LTL IN REACTIVE SYSTEM

Additionally, I utilize Linear Temporal Logic (LTL) in the reactive system, which allows for the creation of formulas using operators such as X for the *next time*, G for *always*, F for *eventually*, and U for *until*. The operations are used to define specific requirements for the Grid-world example.

2.2. Case Study with Grid-world

Within this subsection, I review the shielding paper by initially presenting the environmental settings, training algorithm, and specification utilized in detail in the study by Alshiekh et al. (2018). Subsequently, using the case study example of Grid-world, I will discuss the inherent limitations of their methods. Based on this discussion, I will propose preliminary ideas that have the potential to address the aforementioned constraints in the next section.

2.2.1. ENVIRONMENTAL SETUPS

Grid-world tasks like Pacman game have been frequently employed as small-scale illustrations in the fields of artificial intelligence and reinforcement learning (DeNero and Klein, 2010). They provide discrete and finite state and action space which serves as a helpful simplification for more complex problems. As shown in Fig. 1, the red circle depicts the moving agent. Its discrete actions are constrained in *right*, *down*, *left*, *up*, and *stay*. When moving, the agent can only travel to adjacent blocks at each timestep. It is important to note that, as described by Alshiekh et al. (2018), the agent's policy is **deterministic** which is different from my proposal.

The grid itself is a 9×9 square and serves as the spatial constraint for the agent's movements. The presence of small black blocks within the grid denotes the location of walls, which the agent must avoid reaching. In addition, there exist four stationary bombs situated within the map, which the agent must avoid stepping on in three consecutive timesteps to prevent failure of the game.

The game consists of four distinct goal areas, each represented by a different color. The agent must navigate the goal areas in a specific order: *pink*, *blue*, *green*, and *yellow*. For instance, upon reaching the pink block, it should then aim to head toward the blue goal area. After successfully reaching all four colored goal areas, the agent is awarded a positive reward of 1 while the next goal is reset to the first-ordered pink goal area.

2.2.2. LEARNING STRATEGY

To simplify the scenario, in this example, the agent selects its actions based on Tabular Q learning algorithm (Watkins and Dayan, 1992). Thus, a Q-table will be maintained with $|\mathcal{S}|$ rows and $|\mathcal{A}|$, where $|\mathcal{S}|$ and $|\mathcal{A}|$ represent the total number of states and actions respectively. During the training process, the agent interacts with the environment ϵ-greedily, during which the Q values in the table will be updated. When the agent is deployed, it uses the Q-table to choose the best actions greedily:

$$a_t^* = \arg\max_a Q(s_t, a_t). \tag{4}$$

In contrast to the typical Q learning approach, in this case, we identify three action candidates $A_{candidate} = (a_t^1, a_t^2, a_t^3)$ that have the highest Q values and consider them as potential choices. Subsequently, we utilize Shield to determine the optimal safe action among these candidates.

2.2.3. SPECIFICATION

Based on the safety restrictions from Section 2.2 and the deterministic setting, I give the following specification ψ using LTL:

$$\begin{aligned} &\mathbf{G}(pos_a \in [0,8] \times [0,8]) \wedge \mathbf{G}(pos_a \notin pos_w) \wedge \\ &\mathbf{G}(\neg(pos_a \in pos_b \rightarrow \mathbf{X}stay \wedge \mathbf{XX}stay \wedge \mathbf{XXX}stay)), \end{aligned} \tag{5}$$

where pos_a depicts agent's x and y positions in the grid, while pos_w and pos_b depict walls and bombs position sets. Leveraging this specification, Shield can be automatically synthesized in the form of a Finite State Machine (FSM). It is able to prevent the agent from encountering unsafe situations.

2.2.4. CONSTRAINTS AND PROBLEMS

The research conducted in this case study highlights some issues of the shielding mechanism:

Inconsistent Observation. Alshiekh et al. (2018) claimed that Shield and Learning Agent share observations, but they used different observation formats in their Grid-world implementation. The Q-table takes in a vector consisting of the agent's x, y positions, the next goal area, and the reward obtained for the current step - $[pos_a[0], pos_a[1], csf, payoff]$. In contrast, the Shielding component of the system is provided with a binary encoding of observations. This encoding includes bits 0-3 indicating whether the agent's neighboring positions are valid or not (positions outside the map or blocked by walls are considered invalid), bit 4 indicating whether the agent is standing on a bomb, and bits 5-13 encoding information about the three candidate actions. With this environmental setting, their observations do not align with each other.

Moreover, despite the fact that neither Shield nor the Q-learning-based agent has access to past observations, Shield implicitly maintains previous information through the Finite State Machine. Specifically in the Grid-world example, this allows Shield to prevent the agent from remaining on a single bomb for more than three timesteps, while Q learning lacks access to this information. This difference further worsens the inconsistency of observations between the two approaches, which might lead to unfair comparison.

Inherent Constraints. To construct a Shield, it is typically built offline using prior specifications and knowledge of the environment (Carr et al., 2022; Bastani et al., 2021; Zoon, 2021; Könighofer et al., 2017; Li and Bastani, 2020). Some recent research has explored constructing Shield online based on near-future information (Könighofer et al., 2022). This manner of frequently recreating Shield based on only near-future information would lead to a computational burden while it has no comprehensive overview of the entire environment. Therefore, existing Shielding mechanisms' performance is limited in unknown stochastic environments.

3. Method

Here I propose to synthesize Shield based on the given ψ^S according to Equation 5 and the updated MDP abstraction $\psi^{\mathcal{M}}$ during training to solve the discussed problem from **Inherent Constraints**.

3.1. Stochastic Grid-world Example

In this scenario, we turn the deterministic Grid-world example into stochastic:

$$0 \leq P(S' = s_{t+1} | S = s_t, A = a_t) < 1. \tag{6}$$

Specifically, in this example, there is a probability of the agent staying in its current position when it chooses to move to nearby neighbors. Therefore, even though the shielding mechanism can prevent the agent from choosing to stay on a bomb for more than three timesteps, there is still a possibility of failing the game due to the stochastic nature of the environment:

the agent has stayed on one bomb for three timesteps and the shield forces it to move away, but after executing this *fleeing* action, it still stays based on the transition model. Then, this scenario will violate the specification and leads to a failed scenario.

3.2. Online Synthesis with Transition Model

In the stochastic environment, the agent and Shield do not have access to the hidden transition probabilities for movements. To address this, the training is proposed to maintain a **Transition-table** during exploration.

Algorithm 1 Online Shield Synthesis

Initialization: Experimental environment *Initiated*; Agent *Reset* at initial position; Construct $\psi^{\mathcal{S}}$ and $\psi^{\mathcal{M}}$
Shielding Preliminary Construction: Shield *Synthesized* initially based on $\psi^{\mathcal{S}}$
$safeness = True$
rollout **while** $t \leq Max\_stepsize$ **do**
  if $safeness == True$ **then**
    $\epsilon := 0.8$
    if $\epsilon < random(0, 1)$ **then**
      $A_{candidate} := Q(s_t)$
    else
      $A_{candidate}$ picked up randomly
    end
    $a_t = Shield(encoding(s_t, bomb, A_{candidate}))$
    $s_{t+1}, r_t, safeness = env(a_t)$
    Update Q-table, Transition-table
    Update $\psi^{\mathcal{M}}$ based on Transition-table
  else
    Update Shielding based on new $\psi^{\mathcal{M}}$
    Reset the agent and environment; $safeness = True$
  end
end
Save Q-Table and updated Shield for deployment and tests.

The detailed process for synthesizing the Shield is outlined in Algorithm 1 (the detailed synthesis steps are not part of this report). Before training, we first construct a preliminary Shield solely based on $\psi^{\mathcal{S}}$ as the given specification provides comprehensive prior knowledge of the environment. Thus, an initial offline shielding synthesis will improve online construction efficiency. Then, during training, the agent employs ϵ-greedy strategy and updated Q-table to choose the action candidates $A_{candidate}$. Shield will pick up the safe optimal action based on the candidates and observations.

In addition, I update the maintained transition model based on the rollout data:

$$P(S' = s_{t+1} | S = s_t, A = a_t) = \frac{N(S = s_t, A = a_t \rightarrow S' = s_{t+1})}{N(S = s_t, A = a_t)}. \tag{7}$$

where function N counts the occurrences of certain states and actions that the agent has encountered. Through a sufficient number of rollout iterations, the transition model can obtain nearly precise

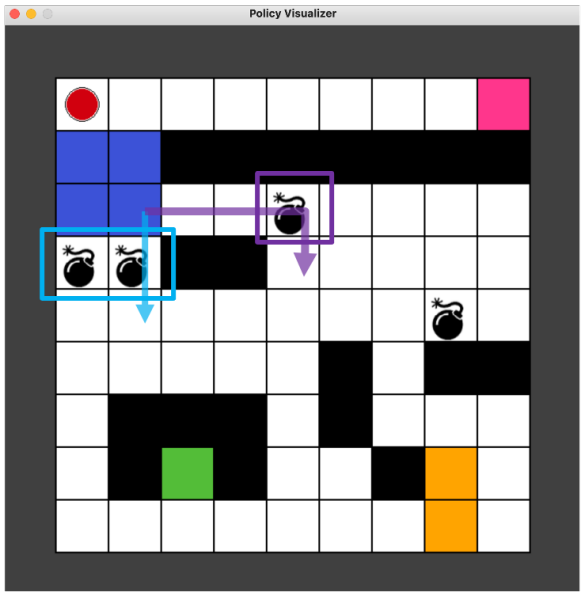

Figure 2: An example that all possible paths obstructed by bombs. If the agent decides to move to nearby blocks, there is a 40% chance that the agent will remain at the same block if it is located within the cyan area. Conversely, if it is located in the purple area, there is only a 20% probability that the agent will remain in the same block.

approximations of probabilities. With the updated MDP abstraction $\psi^{\mathcal{M}}$, we can then construct the online Shielding with minimal adjustments.

It's important to note that the online Shielding is not reconstructed after every time step, but only when the agent reaches an unsafe state. Utilizing this construction strategy, computation resources would be greatly saved. After training, only the trained Q-table and reconstructed Shield are saved for further testing and deployment.

3.3. Extension with Continuous States

For scenarios with continuous states, the Q-table can be conveniently replaced by deep learning-based RL algorithms such as Proximal Policy Optimization algorithm (Schulman et al., 2017). As for updating the transition model, I propose utilizing the Pseudo-counts technique. This technique enables us to estimate the visitations of continuous state-action-pairs by using a Density Model Network, denoted as $p(s, a)$.

4. Analysis of Algorithm

The Online Shielding Construction integrates *information* from both the provided specification and the learned transition model during training. This leads to an improvement in the effectiveness of the shield by preventing the agent from accessing unsafe states and actions in stochastic environments. And as discussed, updating Shield only when unsafe states are encountered, further enhances the system's efficiency.

4.1. Expected Results

As depicted in Fig. 2, the paths from the blue goal area to the green goal area are blocked by three bombs. Interestingly, the transition probability of staying when choosing to move away is higher in the cyan blocks compared to the purple blocks - which means the areas covered by cyan are more dangerous. Despite the cyan path appearing to be shorter, it is riskier as there is a higher likelihood for the agent to remain in the cyan bomb area when it tries to flee.

The learning model does not contain historical information, and it is not easy to gather enough failed data to train a policy sensitive to the bombs. Therefore, it is difficult for the learning-based agent to learn to avoid the cyan path purely using the Q-table. In contrast, the shielding construction leverages both the updated transition model and knowledge from the specification. As a result, the shielding automaton enforces the agent to take a safer route (i.e., the purple path).

4.2. Possible Limitations

There is a possibility that the updated abstraction $\psi^{\mathcal{M}}$ conflicts with $\psi^{\mathcal{S}}$ due to environmental features. In such a case, the shielding construction may fail, potentially leading to unsafe states and actions. Furthermore, in order to obtain an accurate estimation of the transition probabilities, a large amount of memory may be necessary, which could limit the applicability of this approach.

5. Conclusion

This report presents a new approach for constructing online shielding in stochastic environments, which aims to improve upon existing methods. To illustrate the limitations of current shielding techniques and the enhanced effectiveness of the proposed approach, I use a Grid-world toy example. Our method, online Shield synthesis, takes into account both given specifications and updated MDP during training, and is capable of being updated online. While it provides better protection for the agent against unsafe states and actions, there are some drawbacks, such as the large amount of memory needed to store the agent's previous visitations. Future research should focus on addressing these issues and conducting more comprehensive qualitative and quantitative experiments. Additionally, we plan to explore the application of our method in multi-agent scenarios with POMDP settings to further extend the potential of shielding methods.

Acknowledgments

I gratefully acknowledge the help from Sunny Wang and guidance from Prof. Anna Lukina, and their invaluable suggestions. Some of the equations are from Prof. Wendelin Böhmer's Deep RL lecture slides. I acknowledge him for those helpful materials.

References

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Andrea Bajcsy, Somil Bansal, Eli Bronstein, Varun Tolani, and Claire J Tomlin. An efficient reachability-based framework for provably safe autonomous navigation in unknown environ-

ments. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1758–1765. IEEE, 2019.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

Osbert Bastani, Shuo Li, and Anton Xu. Safe reinforcement learning via statistical model predictive shielding. In *Robotics: Science and Systems*, pages 1–13, 2021.

Steven Carr, Nils Jansen, Sebastian Junges, and Ufuk Topcu. Safe reinforcement learning via shielding for pomdps. *arXiv preprint arXiv:2204.00755*, 2022.

John DeNero and Dan Klein. Teaching introductory artificial intelligence with pac-man. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1885–1889, 2010.

Ingy ElSayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. Safe multi-agent reinforcement learning via shielding. *arXiv preprint arXiv:2101.11196*, 2021.

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

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 97–117. Springer, 2017.

Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Advances in neural information processing systems*, 21, 2008.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

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

Bettina Könighofer, Julian Rudolf, Alexander Palmisano, Martin Tappler, and Roderick Bloem. Online shielding for reinforcement learning. *Innovations in Systems and Software Engineering*, pages 1–16, 2022.

Shuo Li and Osbert Bastani. Robust model predictive shielding for safe reinforcement learning with stochastic dynamics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7166–7172. IEEE, 2020.

Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics*, 36(1):1–14, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Cyrus Neary, Christos Verginis, Murat Cubuktepe, and Ufuk Topcu. Verifiable and compositional reinforcement learning systems. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pages 615–623, 2022.

Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

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

Yifei Simon Shao, Chao Chen, Shreyas Kousik, and Ram Vasudevan. Reachability-based trajectory safeguard (rts): A safe and fast reinforcement learning safety layer for continuous control. *IEEE Robotics and Automation Letters*, 6(2):3663–3670, 2021.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Yunlong Song, HaoChih Lin, Elia Kaufmann, Peter Dürr, and Davide Scaramuzza. Autonomous overtaking in gran turismo sport using curriculum reinforcement learning. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 9403–9409. IEEE, 2021a.

Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing with deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1205–1212. IEEE, 2021b.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Dongjie Yu, Haitong Ma, Shengbo Li, and Jianyu Chen. Reachability constrained reinforcement learning. In *International Conference on Machine Learning*, pages 25636–25655. PMLR, 2022.

Job Zoon. Safe reinforcement learning by shielding for autonomous vehicles. 2021.