Technical University of Munich
TUM School of Computation, Information and Technology

Guaranteeing Complex Safety Specifications for Autonomous Vehicles via Reinforcement Learning with Formal Methods

Hanna Krasowski

Complete reprint of the dissertation approved by the TUM School of Computation, Information and Technology of the Technical University of Munich for the award of the

Doktorin der Ingenieurwissenschaften (Dr.-Ing.)

Chair:

Prof. Debarghya Ghoshdastidar, Ph.D.

Examiners:

1. Prof. Dr.-Ing. Matthias Althoff
2. Prof. Majid Zamani, Ph.D.
3. Prof. Roderick Bloem, Ph.D.

The dissertation was submitted to the Technical University of Munich on 19.02.2024 and accepted by the TUM School of Computation, Information and Technology on 18.06.2024.

Acknowledgments

First and foremost, I thank my advisor, Matthias Althoff, for the opportunity to conduct exciting research and continued guidance and support. I am grateful for the past four years of productive and intellectually stimulating work together. Thanks to Aaron Ames for inviting me to his lab and for insightful academic and career advice. I am grateful to Majid Zamani, Roderick Bloem, and Debarghya Ghoshdastidar for inspiring my research and serving on my dissertation committee.

It was a pleasure to be part of the exceptional research teams at the CPS group in Munich and the AMBER lab in Pasadena. Working alongside you has been instrumental in my professional and personal growth. In particular, I sincerely thank my co-authors, Xiao Wang, Niklas Kochdumper, Yinqiang Zhang, Marlon Müller, Jakob Thumm, Lukas Schäfer, Prithvi Akella, and Stanley Bak, and my colleagues, Mark, Gerald, Florian, Josefine, Hannah, Victor, Philipp, Jonathan, Michael, Roland, Eivind, Marius, Maegan, and Rachel. Thanks also to my mentor, Alexander, who always had an open ear. Students I advised at TUM created software that contributed to the success of my research, and I owe special thanks to Andreas, Bruno, Benedikt, Fabian, Matthias, and Stefan.

I gratefully acknowledge public funding of my research through the DFG research training group ConVeY, the BMBF project TRAITS, and the DAAD program IFI. This funding enabled me to attend workshops and conferences and supported my research stay at Caltech. The researchers I met through these programs raised inspiring questions and helped me to advance my projects, and I am especially thankful to fellow ConVeY researchers. Further, I thank the anonymous reviewers, program chairs, and handling editors for constructive feedback on my manuscripts. I am grateful to Ute, Katja, Elisabeth, Mikaela, Janine, and Amy for managing administrative matters seamlessly.

On a personal note, I am incredibly thankful to Anja, Anjuli, Charlotte, Elisabeth, and Moritz for their unwavering and genuine support. Lastly, this dissertation benefited immensely from contributions by countless people, and I thank everyone who helped create an intellectually inspiring and pleasant community that made my doctorate exciting and enjoyable. Thank you.

Munich, February 2024 Hanna Krasowski

Abstract

Reinforcement learning is capable of solving intricate motion planning tasks required for operation of autonomous vehicles. These real-world motion planning tasks are often safety-critical since failure would be disastrous for humans and the environment. Existing reinforcement learning methods typically consider safety aspects softly through the reward function or constraint optimization. Yet, these methods achieve safety in expectation only and do not guarantee safe behavior of autonomous vehicles in out-of-distribution states encountered during operation. Thus, ensuring safety of reinforcement learning-based programs for autonomous vehicles is an open research question.

In this thesis, reinforcement learning methods are combined with verification algorithms to guarantee compliance with safety specifications. To this end, we formalize safety specifications and design safety verification algorithms based on formal methods, mainly set-based reachability analysis. To represent the diverse notions of safety, we derive different safety specifications, i.e., avoiding unsafe states, staying in safe states, and complying with complex temporal logic specifications. Our formal safety specifications enable verification via formal methods to identify safe actions for autonomous vehicles. The integration of our verification algorithms into reinforcement learning methods guarantees compliance with safety specifications always or probabilistically. Our safe reinforcement learning approaches are designed for motion planning environments with static and dynamic obstacles in continuous state space. Conceptually, the proposed approaches are application-independent. Yet, through exploitation of application-specific properties, we increase the performance of our motion planners, e.g., for autonomous driving or autonomous vessel navigation.

Numerical evaluations confirm that our integration of formal verification algorithms into reinforcement learning guarantees safety during both training and deployment. We demonstrate the feasibility of safe reinforcement learning for complex motion planning tasks for the applications of autonomous driving and autonomous vessel navigation. Additionally, we evaluate our safe reinforcement learning approaches on hardware, thereby validating real-time capability. Our safe reinforcement learning agents typically converge similarly fast or faster than reinforcement learning agents that are informed about safety specifications through the reward function. In conclusion, our research enables safe motion planning of reinforcement learning-based autonomous vehicles and ensures compliance with complex safety specifications.

Zusammenfassung

Reinforcement Learning ist in der Lage, komplizierte Bewegungsplanungsaufgaben autonomer Fahrzeuge zu lösen. In realen Umgebungen sind diese Bewegungsplanungsaufgaben meist sicherheitskritisch, da ein Versagen zu katastrophalen Folgen für Mensch und Umwelt führen kann. Bestehende Reinforcement-Learning-Methoden berücksichtigen Sicherheitsaspekte typischerweise in weicher Form entweder in der Belohnungsfunktion oder Nebenbedingungen in die Reinforcement-Learning-Agentenoptimierung. Diese Methoden erreichen Sicherheit jedoch lediglich im Erwartungswert der Belohnungsfunktion und bieten keine Garantien für sicheres Verhalten autonomer Fahrzeuge in Zuständen, die während des Trainierens des Agenten nicht auftreten. Die Gewährleistung von Sicherheitsgarantien für autonome Fahrzeuge, deren Bewegungsplanung auf Reinforcement Learning basiert, ist eine offene Forschungsfrage.

In dieser Arbeit entwickeln wir Verifikationsalgorithmen und integrieren diese in Reinforcement-Learning-Algorithmen, um die Einhaltung von Sicherheitsspezifikationen zu garantieren. Zu diesem Zweck formalisieren wir Sicherheitsspezifikationen und entwerfen Verifikationsalgorithmen mittels formaler Methoden, insbesondere mengenbasierter Erreichbarkeitsanalyse. Aus verschiedenen Sicherheitsanforderungen leiten wir formale Spezifikationen ab, wie das Vermeiden unsicherer Zustände, das Verbleiben in sicheren Zuständen und die Einhaltung komplexer Regeln in temporaler Logik. Unsere Sicherheitsspezifikationen ermöglichen die Verifikation sicherer Aktionen autonomer Fahrzeuge mit Hilfe formaler Methoden. Durch Integration der Verifikation in Reinforcement-Learning-Algorithmen garantieren wir die Einhaltung der Sicherheitsspezifikation durch autonome Fahrzeuge entweder immer oder probabilistisch. Unsere sicheren Algorithmen sind für Bewegungsplanungsumgebungen mit statischen und dynamischen Hindernissen im kontinuierlichen Zustandsraum entworfen. Die entwickelten sicheren Algorithmen sind konzeptionell anwendungsunabhängig. Durch die Ausnutzung anwendungsspezifischer Eigenschaften erhöhen wir die Performanz der Bewegungsplanung für autonome Fahren und autonome Schiffsnavigation.

Experimente bestätigen, dass die Integration unserer Verifikationsalgorithmen in Reinforcement-Learning-Algorithmen Sicherheitsspezifikationen sowohl während des Trainierens als auch im Einsatz garantiert. Wir demonstrieren die Machbarkeit von beweisbar sicherem Reinforcement Learning für komplexe Bewegungsplanungsaufgaben für autonomes Fahrens und autonome Schiffsnavigation. Darüber hinaus evaluieren wir zwei allgemeine Ansätze für sicheres Reinforcement Learning auf Hardware und validieren damit deren Echtzeitfähigkeit. Die sicheren Agenten konvergieren typischerweise ähnlich schnell oder schneller als Agenten, die lediglich durch die Belohnungsfunktion über Sicherheitsanforderungen informiert werden. Zusammenfassend ermöglicht unsere Forschung eine auf Reinforcement Learning basierende Bewegungsplanung autonomer Fahrzeuge unter Einhaltung komplexer Sicherheitsspezifikationen.

Contents

1 Introduction

Reinforcement learning (RL) solved tasks that seemed too difficult for machines only a few years ago. For instance, the RL-based programs AlphaZero and AlphaGo outperformed human experts in chess and Go, respectively, and received widespread media attention. The capabilities of RL to solve complex tasks can also be observed from research on autonomous cyber-physical systems (CPSs) where an autonomous robot fulfills a task in the physical world. For example, RL performed significantly better in drone racing tasks than classical control. These impressive achievements of RL methods are rooted in learning by trial and error and enable flexible adaption to complex tasks. Conceptually, RL methods entail a reward function that quantifies success and error with a value: the reward. The objective of RL is to maximize the expected reward in order to find the optimal policy, i.e., behavior, for a specified task. During training, the RL agent learns the policy by combing exploitation of feedback on previous actions from the reward function and random exploration. The random exploration in RL inherently contradicts guaranteeing safety specifications. Thus, training and evaluation of RL agents for autonomous CPSs is to date mainly done in computer simulation. However, simulation can never feature the fidelity of the real world so that the potential of RL for CPS is not fully realized. Therefore, safely training and evaluating RL approaches in the real world is a promising strategy to advance their performance.

Formal verification methods can provide the necessary guarantees for safe training and deployment. Formal verification requires a system model and a safety specification to identify safe actions for RL agents. The system model includes the autonomous CPS, relevant environment components like obstacles, and the sets of admissible system states. The safety specification is a formally evaluable definition of safety. There are two overarching verification concepts: offline and online verification. For offline verification, the learned policy is verified for all admissible system states. Yet, this verification approach is infeasible for many tasks of autonomous CPSs due to the amount of admissible system states requiring verification. For example, one would need to determine all traffic situations that could potentially occur for an autonomously driving car and verify the policy for all of them. Even if there is a comprehensive safety specification and a model for verifying a policy, every policy update necessitates re-verification. Online verification approaches address this issue by computing safe actions at each time step in which the RL agent decides for an action given the current system state. Thus, not all admissible system states have to be known in advance. Integrating online verification approaches into RL safeguards policies so that guarantees with respect to safety specifications are provided.

In practice, synthesizing a system model and safety specifications for CPSs is challenging. Autonomous CPSs operate in complex environments with dynamic obstacles, which have

to be reflected in the system model. For the CPS, a model is often available or can be safety-conformably estimated based on recorded trajectories. Models of dynamic obstacles are generally more difficult to obtain due to higher uncertainty. For instance, for other traffic participants, uncertainty is typically large since we do not know their future movements with certainty. The challenge regarding safety specifications is their formalization. While humans understand safety intuitively (e.g., "never collide"), specifying safety formally is often tedious. This is rooted in two issues. On the one hand, the notion of safety is often more complex than collision avoidance and necessitates for example compliance with various traffic rules. On the other hand, even for collision avoidance safety must be tightly specified to avoid that only trivially safe actions are verifyable, e.g., a standstill at an urban intersection.

In this work, we propose different safe RL methods that utilize verification approaches to achieve probabilistic and hard safety guarantees for motion planning tasks of autonomous CPSs. We investigate multiple degrees of complexity of the safety specifications. The considered motion planning tasks include uncertainty through disturbances in the CPSs or dynamic obstacles like other traffic participants. Our approaches are developed for autonomous vehicles operating in continuous space such as cars, vessels, and drones. While the physical world is always modeled by a continuous space, the RL policy learns either high-level decisions described by a finite set of discrete actions, or control inputs that are directly executable for the autonomous vehicle and stem from a continuous action space. We explore both types of action spaces. Whenever high-level decisions are key for task fulfillment, we use discrete action spaces for efficient computation. In cases where it is important to leverage the entire action space, we use continuous action spaces that contain all admissible control inputs.

1.1 Literature Review

To conceptualize our contributions toward safe RL with formal methods for motion planning tasks of autonomous vehicles, we offer three perspectives on the related literature. In Section 1.1.1, we introduce the field of safe RL by clustering the research based on the type of safety guarantee. In Section 1.1.2, we show how formal methods are combined with RL with a focus on verification approaches and applications of temporal logic, as these are closely related to our work. Finally, we give a brief overview of research on motion planning tasks of autonomous vehicles and the most prominent methods in Section 1.1.3.

1.1.1 Safe Reinforcement Learning

Safe RL approaches were initially categorized and described in 2015 [1]. In this survey, safe RL is defined as methods for obtaining a policy for tasks in which it is important to ensure safety and performance during training and/or deployment of RL agents. RL agent paraphrases a repeatedly updated policy during training, and the final trained policy during deployment. Note that training in RL is equivalent to the process of learning a policy. Safe RL can be distinguished in approaches that modify the optimization problem of maximizing the expected reward and those that modify the exploration of RL agents to ensure safety [1]. Since 2015, the field of safe RL has grown rapidly due to the development of deep RL algorithms using neural

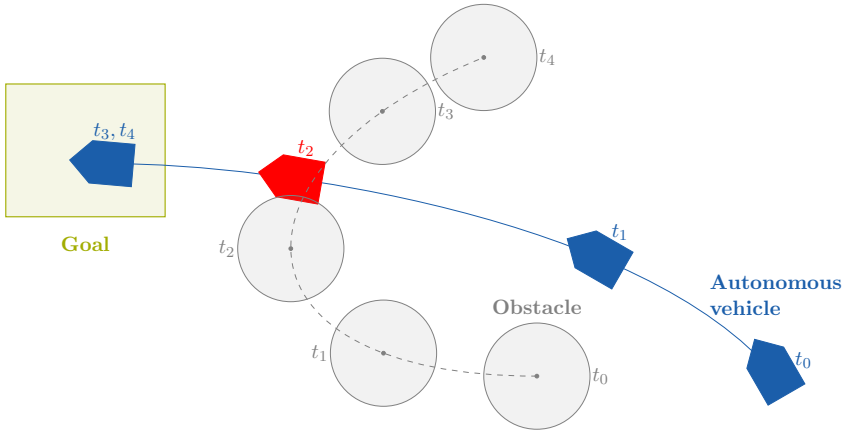
networks as policies. This development allowed for a broader applicability of RL including autonomous CPSs [2]. We categorize the growing literature by the degree of safety assurance, i.e., soft constraints, probabilistic guarantees, and hard guarantees, and exemplify the research in each categories with a few examples.

Soft constraints RL approaches with soft constraints consider safety in their optimization problem of maximizing the expected reward as a soft constraint. They do so via integration of safety objectives into the reward function or as soft constraints of the optimization problem by using Lagrangian methods to ensure feasibility of the optimization problem. These approaches usually converge to a safer policy at the end of the training, but are typically unsafe during training of the policy, especially for the first episodes after randomly initializing the policy. An illustrative example is provided in Figure 1.1a, where the autonomous vehicle slightly collides with the dynamic obstacle at time t_2.

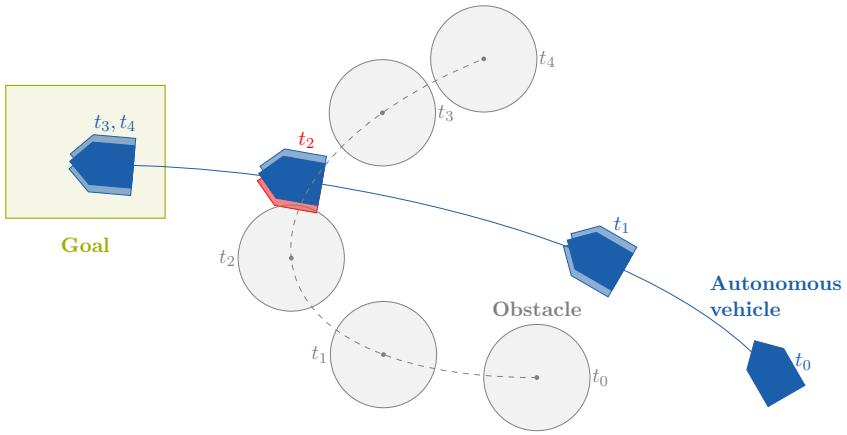
The vast majority of safe RL research integrates safety aspects through the reward function. Next to manually defining an application-specific safety component for the reward function [3–6], there are also more systematic approaches to obtain a reward function. Systematic approaches include using temporal logic specifications to synthesize reward functions [7, 8] or learning the reward function from expert demonstrations [9]. Despite its simplicity, the integration of safety aspects in the reward function can be tedious due to time-consuming manual tuning or unsuited data for learning the reward function. This problem is often aggravated if there exists a trade-off between safety and performance objectives. Another prominent method is constraint RL where safety constraints are included through a Lagrangian dual optimization problem [10, 11]. In particular, these methods improve the policy based on the optimization problem of maximizing the expected reward subject to constraints. Safety constraints are commonly included as constraint functions [12–16] or as temporal logic formulas [17–20]. Generally, soft constraint RL methods are advantageous for non-safety-critical tasks, where safety violations are not catastrophic, since they do not require a system model.

Probabilistic guarantees Probabilistic guarantees certify safe behavior in probabilistic bounds. A motion plan with probabilistic guarantees is illustrated in Figure 1.1b for the same motion planning task as in Figure 1.1a. The opaque vehicles symbolize the confidence interval bounds. At time t_2, there is a probabilistic guarantee for collision avoidance of the autonomous vehicle. RL approaches that provide probabilistic guarantees require less system knowledge than approaches with hard guarantees because a safety specification or a system model can be learned over time. In particular, one can synthesize a model with a certain confidence from sampled data [21, 22], or iteratively expand a safe state set while only allowing for violations up to a certain probabilistic budget [23–26]. Additionally, if disturbances are assumed to follow unbounded probability distributions, only probabilistic safety guarantees can be provided [27].

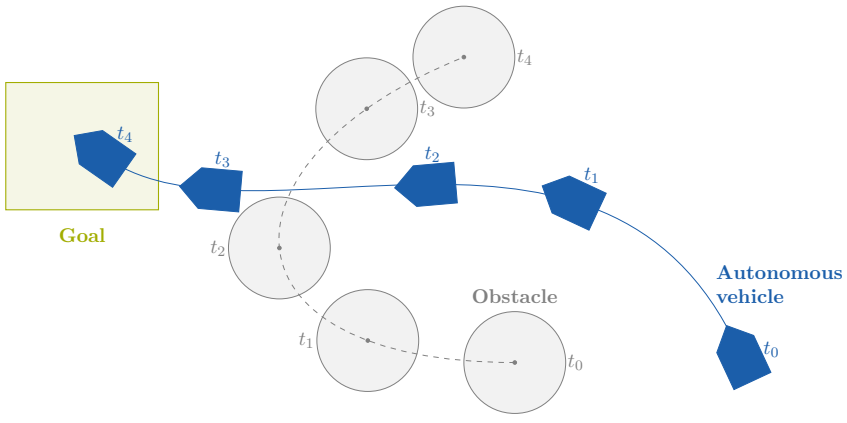
Other approaches determine the probability of safety specified via temporal logic for actions based on discrete system models [28, 29]. In [28], probabilistic model checking is utilized to synthesize a safety shield. This safety shield blocks actions online if they are riskier than a



(a) Soft safety constraints.



(b) Probabilistic safety guarantee.



(c) Hard safety guarantee.

Figure 1.1: Autonomous vehicle solving the motion planning task to reach a goal and avoid a dynamic obstacle with different degrees of safety assurance.

specified threshold. The authors evaluate their approach on multiple grid world examples where they observe that adding a safety shield results in higher rewards already at the beginning of the training. Another method is to describe the safety specification with probabilistic computation tree logic and determine a safe state set with a network of probabilistic timed automata [29]. The proposed safe set contains all states that fulfill the probabilistic specification. In numerical experiments on an autonomous driving benchmark, the probabilistic guarantee for avoiding collisions and keeping the speed of the autonomous vehicle within a specified range is validated.

Hard guarantees Hard guarantees always ensure compliance with safety specification. Figure 1.1c shows a motion plan with hard safety guarantees. The autonomous vehicle has to slow down so that the dynamic obstacle can pass in front and reaches the goal later at time t_4 compared to the motion plans in Figure 1.1a and Figure 1.1b. Thus, to ensure hard safety guarantees, the efficiency of goal reaching is reduced. Conceptually, hard guarantees for RL agents can be obtained by offline verification of a trained RL agent or by including online verification to restrict the RL agent to safe actions during training and deployment. Offline verification approaches [30–32] are usually not suited for motion planning tasks of autonomous CPSs due to the necessity of specifying all admissible system states. Provably safe RL approaches provide hard safety guarantees during training and deployment. There are three conceptual approaches of provably safe RL.

First, action replacement replaces unsafe actions with a safe action that is selected independently of the unsafe action, and is the commonly used provably safe RL approach [33–37]. For example, Hunt et al. [37] determine the safety of actions based on a symbolic state representation and a model of the system, and replace unsafe actions with a safe action sampled uniformly at random from the set of safe actions. Second, action projection corrects unsafe actions proposed by the RL agent to the closest safe action before execution. Action projection is mainly used when safety specifications can be easily formulated as constraints for an optimization problem that identifies the closest safe action. Thus, action projection approaches are often based on model predictive control (MPC) [38, 39] or control barrier functions [40–42]. Third, action masking restricts the space of selectable actions to the set of safe actions. In contrast to action replacement and action projection, action masking requires the set of safe actions explicitly. This is often only feasible for discrete action spaces as in [43–45].

Challenges To obtain safety guarantees, which are essential for safety-critical tasks, a system model and specification are necessary. There are usually probabilistic or hard assumptions for the system model that includes environment components necessary in order to achieve hard or probabilistic guarantees for safe RL. For most motion planning tasks, an additional challenge is that dynamic obstacles are part of the environment. For these dynamic obstacles, safety-conformant prediction methods have to be identified to predict their future behavior in order to verify actions of RL agents.

The complexity allowed for safety specifications is relatively low when considering safe RL with hard safety guarantees, i.e., most approaches can only handle safety specifications that are described through safe sets or unsafe sets. However, this does not reflect the complexity

of safety specifications present for real-world motion planning tasks such as extensive traffic rules for driving on roads.

1.1.2 Reinforcement Learning with Formal Methods

There is a trend in combining formal methods with RL to reduce the engineering effort necessary for stable and efficient training of agents and to guide the RL process toward the most relevant regions for learning. The most relevant directions for our work are formal verification approaches, which verify the safety of actions during training, after training, or during deployment, and the integration of temporal logic specifications in the RL process. Note that the integration of temporal logic specifications may consider safety only or, alternatively, the full task of the RL agent, i.e., include both safety and performance objectives.

Verification of agents Generally, the verification of RL agents either verifies the system offline, e.g., before deployment, or is integrated online to correct unsafe actions if necessary. Since RL agents are often represented by deep neural networks, offline verification usually checks that complex non-linear functions comply with safety specifications [30–32, 46]. For example, Bastani et al. [30] convert a neural network representing an RL agent into a decision tree to reduce the complexity for the subsequent verification. This decision tree is human-interpretable and a standard SMT solver [47] is employed to verify Boolean safety specifications.

In this dissertation, we mostly regard online verification approaches for RL agents operating in continuous state spaces. The most common methods to verify such agents online are forward invariant sets [48] where control barrier functions [40–42, 49] are prominent, set-based reachability analysis [36, 50, 51], Hamiltonian-Jacobi-Issacs (HJI) reachability analysis [33, 52], and MPC safety filters [38, 39, 53]. For forward invariant sets, there exists a control input that keeps the system within the set. Control barrier functions represent a forward invariant set for control-affine systems through a differentiable function. Commonly, control barrier functions are used to constrain the optimization problem projecting unsafe actions to safe actions. For example, Marvi and Kiumarsi [41] propose such an approach for linear time-invariant systems and decrease the conservativeness of the initially specified control barrier function as they become more certain about the system dynamics. Set-based reachability analysis computes the forward evolution of the system and can be used to determine if this evolution eventually intersects unsafe areas. For example in [36], unsafe actions are replaced by the closest randomly sampled safe action. The set-based reachability approach for verifying actions is applicable to general nonlinear systems with uncertainties and evaluated on a high-fidelity autonomous driving task and an aerial navigation task. In contrast, HJI reachability analysis computes a backward reachable set that can be used to determine a safe state set for which a goal is reachable without intersecting with unsafe states. For example, Fisac et al. [52] introduce a general safe RL framework based on action replacement with a failsafe controller, which intervenes whenever the action suggested by the RL agent would lead to leaving the safe state set. MPC safety filters formulate a model-predictive optimization problem with safety constraints. Commonly, the optimization objective is a norm on the the distance between the proposed and the safe action. Thus, MPC safety filters are usually used in action projection as in [39]. Note

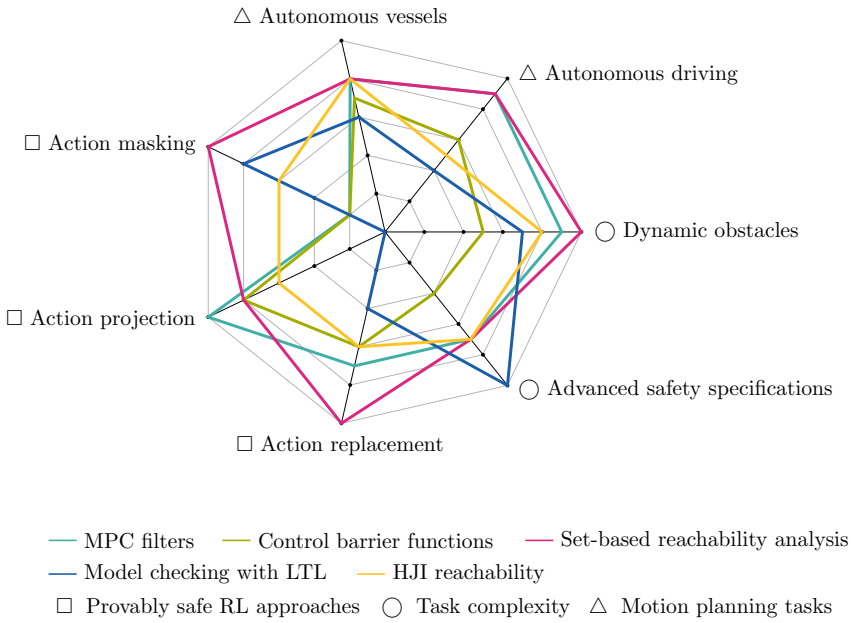


Figure 1.2: Capabilities of online verification methods integrated in RL.

that for RL agents with discrete state and action spaces, model checking with Linear Temporal Logic (LTL) specifications is a common approach [34, 35]. For example, Alshiekh et al. [34] identify the safe actions given LTL specifications with a safety shield, which is based on model checking and replaces unsafe actions proposed by the RL agent.

We illustrate our qualitative assessment of the different capabilities of prominent online verification methods with a spider diagram in Figure 1.2. One group of categories is the applicability to the three provably safe RL concepts. Here, set-based reachability analysis is well suited for all three concepts while for action projection control barrier functions and MPC safety filters are also common. The second group of categories represent the potential complexity of motion planning tasks: applicability with dynamic obstacles and possibility to integrate advanced safety specifications. While set-based reachability analysis is especially well-suited for dynamic environments with uncertainties, model checking with LTL seamlessly integrates advanced safety specifications. The third group of categories indicates the applicability to the motion planning tasks where MPC and set-based reachability analysis both seem to be promising. We discuss prominent motion planning approaches for the two applications in detail in Section 1.1.3.

Temporal logic specifications There are three popular approaches to combine temporal logic specifications with RL: synthesizing a reward function for the full task based on a temporal logic specification, using temporal logic formulas as constraints when updating the RL policy, and online and offline verification of temporal logic formulas. Studies for the last category, e.g., [34, 35], have already been illustrated in previous sections. To systematically design RL rewards, temporal logic can be used to formalize the task, potentially including safety specifications. This temporal logic formula is used to design the reward function either by using

robustness measures associated with the formula [7, 54, 55] or by transforming the temporal logic formula into an automaton that generates the reward [8, 56–59]. For some algorithms, the policy is guaranteed to converge to the optimal policy, which maximally satisfies the temporal logic specification [60, 61].

When temporal logic formulas are used to constrain the policy update, they commonly describe safety specifications [17–20]. De Giacomo et al. [17] introduce restraining specifications formalized by a LTL formula, which is interpreted over finite traces. The evaluations of the specifications can be observed by the agent and the reward is extended with specification reward components, resulting in a loose integration of the temporal logic specification in the RL process. In contrast, Hasanbeig et al. [20] introduce a product Markov Decision Process (MDP) of the original RL MDP and a limit-deterministic Büchi automaton, which is constructed based on the temporal logic specification. Based on this product MDP they introduce a pessimistic and an optimistic learner, whereby the pessimistic learner ensures that only low-risk actions are executed.

Challenges Using temporal logic specifications for generating a reward function, which captures the full task, is more systematic and automatic than manual design. However, the temporal logic specification then contains components that indicate performance and safety, e.g., the goal should be reached in the future and obstacles should always be avoided. Consequently, the resulting agent finds the best solution with respect to all the components. However, there are tasks where safety and performance are conflicting, e.g., efficiency and safety in driving on highways. In these cases, the agent would find a trade-off between safety and performance where safety might be compromised for performance, which is often undesired.

Verification approaches on the other hand achieve hard or probabilistic guarantees. For online verification approaches, continuous state and action spaces with environment uncertainty are challenging due to the infinite amount of potentially unsafe and safe actions and the influence of the uncertainty. Verification methods for continuous state and action spaces are computationally more complex than for discrete spaces, thus requiring the design of efficient algorithms. Further, uncertainty strongly influences the conservatism of the verification result. Yet, most motion planning systems operate in continuous spaces with significant environmental uncertainties. Offline verification on the other hand necessitates re-verification whenever the safety specification or model changes.

1.1.3 Motion Planning for Cyber-Physical Systems

There is a variety of solutions for motion planning tasks of autonomous CPSs. Safety is always an important part of motion planning tasks as collisions should be avoided. The motion planning methods are employed, e.g., for robotic manipulation [62], mobile robots [63], and autonomous vehicles [64–66]. In this dissertation, we regard non-cooperative motion planning tasks of autonomous vehicles and have an in-depth look into autonomous driving and autonomous vessel navigation, which we also focus our literature review on.

Autonomous driving There is an immense amount of research on motion planning for autonomous driving [64, 67, 68]. For conciseness, we will only provide an overview of conceptual approaches with a few explanatory examples. The literature can be categorized into classical motion planning and machine learning-based motion planning [64]. Classical motion planning approaches commonly utilize user-defined cost functions for determining the quality of a motion plan and compute cost-minimizing trajectories. In particular, the resulting motion plan is the cost-optimal trajectory when generated with optimization-based methods such as MPC [69, 70], or the lowest-cost trajectory when identified with search-based [71, 72] or sampling-based methods [73–76]. For search-based methods, motion primitives are commonly employed. Motion primitives are samples of the action space and represent a partial kinematic-feasible movement of the vehicle. The search algorithm connects the motion primitives to a trajectory so that the lowest-cost trajectory is efficiently found. In contrast, sampling-based approaches generate multiple potential trajectories in the state space of the autonomous vehicle and then identify the trajectory with the lowest cost.

The most important machine learning approaches used for motion planning in autonomous driving are imitation learning and RL. Imitation learning is often conducted as behavior cloning or inverse optimal control [68]. In behavioral cloning, a policy is learned based on expert demonstrations [77–79]. For example, Prakash et al. [77] use the driving simulator CARLA [80] to create realistic driving trajectories and add sampling of critical traffic situations to further improve the policy. Inverse optimal control approaches learn a reward function from expert demonstrations. During operation of the autonomous car, the reward-optimal trajectory provided by a classical sampling method is selected [81–83]. Lee et al. [82] propose an imitation learning approach that is extended with adversarial training samples to improve training stability. They evaluate their approach on a racing simulator and on a small racing car. RL agents for autonomous driving learn by interacting with the traffic situation and receive feedback on their performance through a reward [40, 84–88]. For instance, Cheng et al. [40] train an RL agent in a continuous state and action space for a adaptive cruise control task while ensuring safety via control barrier functions.

Safety in autonomous driving is usually interpreted as avoiding collisions with traffic participants and is sometimes extended to traffic rules, e.g., keeping a safe distance to the leading vehicle [89]. Safe RL approaches often consider one specific operation domain, e.g., highways [84, 90, 91], intersections [85, 92], and/or single-lane adaptive cruise control [43, 86]. For example, Bouton et al. [92] achieve safe intersection driving in traffic situations with other vehicles and pedestrians. They propose a discrete action space, mask out unsafe actions, and revert to the safest action if no action is guaranteed to avoid collisions. For highway driving, Wang [90] employs control barrier functions to achieve safe driving in continuous action spaces. The considered notion of safety is keeping a safe distance to the leading vehicle in the same lane and to the following and leading vehicle on the target lane of lane-changing maneuvers.

Autonomous vessel navigation In comparison to autonomous driving, there is much less research on motion planning for autonomous vessels. Yet, this is an important application where safe autonomy might be achieved earlier. Contributing factors to this could be the

significantly fewer traffic rules specifying safety on the water and the slower movements of vessels favoring real-time computation requirements. Maritime motion planning approaches usually consist of three building blocks: a guidance system to generate reference trajectories, a control system to produce control signals that track the reference trajectory, and an state observer, which estimates the true state of the vessel [93]. Many recently proposed motion planning approaches mainly regard the guidance and control system while the state observation is assumed to be near perfect and often not discussed in detail.

The most prominent motion planning approaches are RL [6, 94–99], MPC [100–103], and search-based methods with motion primitives [104–107] or sampled trajectories [108]. For search-based and sampling-based methods, the vessel cannot reach any position in the continuous state space while their computational efficiency is commonly high. In contrast, MPC identifies the optimal control input in a continuous space given a cost function by solving an optimization problem. For example, the study [102] shows promising results for traffic situations with multiple dynamic obstacles and even validate their control approach in real-world tests. Most of the recently proposed motion planning approaches for autonomous vessels are based on RL. For example, the studies [6, 94] consider maritime traffic situations with dynamic and static obstacles and design a reward function so that the RL agent follows a path, avoids collisions, and considers collision avoidance traffic rules. Heiberg et al. [6] even perform extensive evaluation on realistic maritime traffic scenarios with recorded movements of other vessels on a marine map with multiple islands.

Safety in motion planning tasks for autonomous vessels is usually specified as collision avoidance with static and dynamic obstacles and/or traffic rule compliance. The most researched traffic rules for vessel navigation are the collision avoidance rules described in the Convention on the International Regulations for Preventing Collisions at Sea (COLREGS) [109]. Motion planning approaches that are based on MPC or RL commonly include components in their cost or reward function that incentivise compliance with these collision avoidance rules [6, 96, 100, 102, 103].

Challenges Although collision avoidance is the most important safety aspect for motion planning tasks, legal safety [110] required for autonomous vehicles is often more complex. If autonomous vehicles do not achieve legal safety, they will likely not be certified for commercial deployment. For autonomous driving and autonomous vessel navigation, traffic rules specify legal safety. However, there is limited research on the comprehensive integration of traffic rules in autonomous vehicles, which is a necessary step to obtain certifiable safety.

Motion planning approaches are similar across applications as demonstrated for autonomous driving and autonomous vessel navigation. Yet, the individual variant is often tailored to a specific application as exploiting application-specific knowledge often leads to more efficient methods and less conservative behavior of the autonomous vehicles with respect to safety specifications. At the same time, autonomous CPSs are not only conquering transportation on the road and water, but are also gaining presence in everyday tasks. There are countless applications for robots performing motion planning tasks in our daily lives. Thus, there is a need for motion planning approaches that generalize across applications while being safe. In

particular, it would be ideal to develop generalizing methods that improve task performance through interactions with the specific task environment while ensuring safety specifications.

1.2 Contributions

In this dissertation, we address the aforementioned challenges by developing safe RL approaches with probabilistic and hard guarantees for complex safety specifications. The considered motion planning applications are autonomous vehicles, in particular, cars, mobile robots, drones, and vessels. The autonomous vehicles perform their task in environments with static and dynamic obstacles. The research reproduced in Appendix A regards safety specifications where unsafe sets have to be avoided or the system has to stay in a safe set at all times. Since many motion planning tasks exhibit more complex safety specifications, e.g., traffic rules with temporal dependencies, we showcase how natural language traffic rules can be formalized with temporal logic and present safe RL approaches that achieve guarantees with respect to temporal logic specifications in Appendix B.

Appendix A.1 There is an increasing amount of research on safe RL with hard guarantees, i.e., provably safe RL, while there is no unified terminology and no established benchmark for provably safe RL approaches, yet. In [K1], we identify and describe the three main conceptual approaches of provably safe RL and introduce a unifying terminology. We conduct a systematic literature review to survey existing provably safe RL research and categorize them with respect to our identified conceptual approaches. Additionally, we thoroughly compare different provably safe RL approaches on two stabilization tasks for a inverted pendulum and a two-dimensional quadrotor.

Appendix A.2 Many provably safe RL approaches are tailored to specific applications such as autonomous highway driving. In [K2], we propose a general action projection approach for CPSs with nonlinear dynamics, which projects unsafe actions so that potentially time-varying unsafe sets are always avoided. Our approach is based on a set-based reachability analysis algorithm that preserves the parametrization of the action set such that intersections of the reachable set of the CPS with unsafe sets can be easily formulated as constraints for an optimization problem. Additionally, we propose several extensions that improve the computational efficiency and potentially reduce the conservativeness of the action projection. We show the transferability between CPSs and the real-time capability on four benchmarks with static and time-varying unsafe sets.

Appendix A.3 and Appendix A.4 RL-based approaches are often proposed for motion planning tasks of autonomous cars. In [K3] and [K4], we develop provably safe RL approaches for highway driving and urban intersection driving. We propose a discrete high-level action space that reflects meaningful decisions for the autonomous vehicle in the specific traffic situation, e.g., lane-changing and lane-following decisions. Our verification approaches identify all unsafe actions and mask them out of the RL action space. For situations where no discrete

action can be verified, we engage a failsafe motion planner. We evaluate our approaches on real-world traffic data sets, i.e., data from highway situations for [K3] and data from urban intersections for [K4]. We conduct an ablation study for our safety verification in [K4]. Our ablation study shows that including the safety verification for lane-changing and lane-following actions improves the goal-reaching rate. However, including the verification of actions for safe intersection passing significantly reduces the goal-reaching rate.

Appendix B.1 The performance drop observed in [K4] results from a conservative specification of intersection safety that does not regard right-of-way traffic rules at intersections. Traffic rules are usually specified in legal text and are not directly machine interpretable. Thus, we formalize maritime collision avoidance traffic rules with metric temporal logic in [K5] and parameterize them to ease potential adaptions. To validate the formalized rules, we extract approximately 500 vessel encounters from recorded maritime traffic data and evaluate the rule compliance for these encounters. For these traffic situations, we observe a plausibly high rule compliance.

Appendix B.2 Most safe RL approaches that provide guarantees require an explicit system model to perform verification. In [K6], we propose a safe RL approach for continuous state and action spaces that does not require an explicit system model and still achieves probabilistic guarantees for complex safety specifications formalized via temporal logic. To this end, we combine probabilistic verification with a safe RL approach inspired by continuous action masking in a three-step process. The iterative process allows to separate safety and performance objectives in distinct steps. We implement our approach on a mobile robot evasion task and evaluate our trained agent in simulation and on hardware.

Appendix B.3 We are the first to propose a safe RL approach for autonomous navigation of vessels on the open sea that guarantees traffic rule compliance [K7]. We build our safety verification upon the formalization proposed in [K5] and derive a statechart that models the hierarchy of the formalized rules. To determine safe actions, we develop a maneuver synthesis algorithm, which searches for maneuvers that comply with the standard collision avoidance traffic rules. If other vessels do not comply with these rules, an emergency maneuver must be performed, which is formalized by a novel temporal logic rule. To incorporate the emergency maneuver rule in our verification of safe actions, we introduce an emergency controller that intervenes in emergency situations similar to the previously used fail-safe planners for autonomous driving. Our resulting safe RL agent always complies with the maritime traffic rules during both training and deployment in critical maritime traffic situations.

The respective publications are reprinted in Appendix A and Appendix B and below is the list of these publications:

[K1] **H. Krasowski***, J. Thumm*, M. Müller, L. Schäfer, X. Wang, and M. Althoff. “Provably safe reinforcement learning: Conceptual analysis, survey, and benchmarking”. In: *Transactions on Machine Learning Research* (2023).

[K2] N. Kochdumper*, **H. Krasowski***, X. Wang*, S. Bak, and M. Althoff. "Provably safe reinforcement learning via action projection using reachability analysis and polynomial zonotopes". In: *IEEE Open Journal of Control Systems* 2 (2023), pp. 79–92.

[K3] **H. Krasowski***, X. Wang*, and M. Althoff. "Safe reinforcement learning for autonomous lane changing using set-based prediction". In: *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–7.

[K4] **H. Krasowski***, Y. Zhang*, and M. Althoff. "Safe reinforcement learning for urban driving using invariably safe braking sets". In: *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*. 2022, pp. 2407–2414.

[K5] **H. Krasowski** and M. Althoff. "Temporal logic formalization of marine traffic rules". In: *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*. 2021, pp. 186–192.

[K6] **H. Krasowski**, P. Akella, A. D. Ames, and M. Althoff. "Safe reinforcement learning with probabilistic guarantees satisfying temporal logic specifications in continuous action spaces". In: *Proc. of the IEEE Conf. on Decision and Control (CDC)*. 2023, pp. 4372–4378.

[K7] **H. Krasowski** and M. Althoff. "Provable traffic rule compliance in safe reinforcement learning on the open sea". In: *arXiv:2402.08502* (2024).

** Indicates multiple lead authors that contributed equally.*

The following publications are excluded from the dissertation as they do not substantially contribute to the central research questions. The first two publications [K8, K9] introduce open-source research projects, i.e., CommonOcean[1] and CommonRoad-RL[2]. Both tools were used as implementation basis for some of the included publications, which is indicated within the respective publications. The other two publications [K10, K11] investigate control for underactuated vessels in ocean currents.

[K8] **H. Krasowski** and M. Althoff. "CommonOcean: Composable benchmarks for motion planning on oceans". In: *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*. 2022, pp. 1676–1682.

[K9] X. Wang, **H. Krasowski**, and M. Althoff. "CommonRoad-RL: A configurable reinforcement learning environment for motion planning of autonomous vehicles". In: *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*. 2021, pp. 466–472.

[K10] A. Doering*, M. Wiggert*, **H. Krasowski**, M. Doshi, P. F. Lermusiaux, and C. J. Tomlin. "Stranding risk for underactuated vessels in complex ocean currents: Analysis and controllers". In: *Proc. of the IEEE Conf. on Decision and Control (CDC)*. 2023, pp. 7055–7060.

[1] commonocean.cps.cit.tum.de
[2] commonroad.in.tum.de/tools/commonroad-rl

[K11] M. Killer*, M. Wiggert*, **H. Krasowski**, M. Doshi, P. F. Lermusiaux, and C. J. Tomlin. "Maximizing seaweed growth on autonomous farms: a dynamic programming approach for underactuated systems navigating on uncertain ocean currents". In: *arXiv:2307.01916* (2023).

The remainder of this dissertation is structured as follows: In Chapter 2, we introduce preliminaries for our research and describe our solution concepts to integrate formal methods in RL to achieve safety guarantees. We discuss our work and identify future research directions in Chapter 3. We contextualize and reproduce our studies [K1–K4] in Appendix A and [K5–K7] in Appendix B.

2 Safe Reinforcement Learning with Formal Methods

This thesis hypothesizes that we can combine RL with formal methods in order to obtain effective controllers with safety guarantees. In this chapter, we present preliminary concepts of RL and formal methods, and sketch our solution concepts to achieve safe RL controllers.

2.1 Preliminaries

We denote sets by calligraphic letters, vectors by lowercase letters, and functions in Italic font. A task describes the overall problem to be solved by the autonomous CPS, e.g., a mobile robot has to reach a goal area while avoiding collisions with obstacles. The autonomous CPS is abbreviated by ego system. We commonly describe the ego system with a dynamic system model $\dot{x} = f(x(t), u(t))$. The dynamic system model specifies how the ego system evolves over time given an initial state $x_0 \in \mathcal{X}_0$ and a control function $u(t)$, which produces a control input for the continuous time t. The state trajectory of the ego system is $x(t)$ while the state of the ego system is from a continuous domain $x \in \mathcal{X} \subset \mathbb{R}^N$. The control input to the dynamic system model is $u \in \mathcal{U} \subset \mathbb{R}^M$.

However, the complete system, which models all components necessary for task fulfillment, usually consists of the ego system and environment components, i.e., objects that can interact with the ego system. We often do not know the complete system model as we only observe environment components such as traffic participants partially. Section 2.1.1 introduces the concept of RL and provides an overview of utilized RL algorithms. To determine safety of actions or states, formal methods require a model and specification. Thus, we introduce the relevant models, safety specifications, and verification methods in Section 2.1.2.

2.1.1 Reinforcement Learning

The fundamental model for RL is the MDP. An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$. The observation space $\mathcal{S}$ is often called state space in the RL literature. However, the observation space is not necessarily the same as the state space of the ego system $\mathcal{X}$ and is fully observable in the context of this work. Thus, we use $x \in \mathcal{X}$ to describe the state of the ego system, e.g., velocity, position, and orientation of a mobile robot, whereas we use $s \in \mathcal{S}$ to describe an observation of the environment and ego system, e.g., the position and velocity of the autonomous vehicle, relative position to an obstacle, and friction coefficient between the

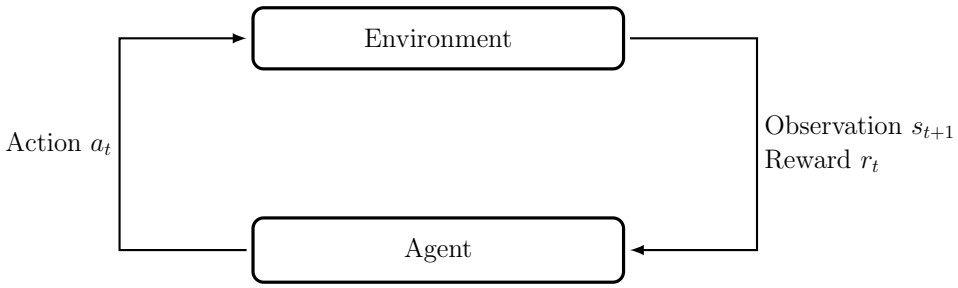


Figure 2.1: Standard RL process.

autonomous vehicle and the ground. The ego system, i.e., RL agent, selects actions $a \in \mathcal{A}$ from the action space $\mathcal{A}$. Examples for actions are high-level actions, such as lane changing or lane following, or low-level control inputs, such as acceleration and steering angle. Both action and observation space can be continuous or discrete. The transition function $T\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{P}$ returns the transition probability $\sigma \in \mathbb{P}$ for the next observation s' when taking action a given observation s for a discrete MDP. For an MDP with continuous spaces, T denotes the probability density function in the observation space. The discount factor γ weights the relevance of future rewards. The reward function $R\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ returns a reward r that indicates how beneficial a transition was with respect to fulfilling the task. For many tasks, the transition function T and reward function R are unknown. In these cases, the policy $\pi \in \Pi$ for task fulfillment cannot be determined by the MDP but must be learned from interactions with the environment. This process is called RL and is depicted in Figure 2.1. The RL agent selects an action a_t based on the observation s_t. The selected action is executed in the environment, i.e., the ego system and the environment components are evolved for one time step. Then, the RL agent receives the reward r_t as feedback and the next observation s_{t+1}, which is used to select the next action and the cycle repeats. The reward function is usually designed to reinforce the RL agent's beneficial behaviors and penalize unwanted behavior. An illustrative example is a mobile robot that has to avoid static obstacles, i.e., receives a negative reward when it hits an obstacle, and has to reach the room door, i.e., receives a positive reward when it gets closer to the door. The goal of RL is to find the optimal policy π^* that maximizes the expectation of the discounted reward for infinite time horizons:

$$\pi^* = \max_{\pi \in \Pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \, r_t \right]. \tag{2.1}$$

In practice, the problem is relaxed to a finite time horizon problem, where after N time steps, the system is reset to a new initial observation, i.e., a new episode is started. This eases learning the policy, because even if the RL agent maneuvered itself in a dead-end situation, it will be eventually reset to a meaningful initial observation for the next episode.

The policy can be explicitly determined or implicitly described by the Q-function $Q(s, a)$. The best action based on a Q-function for a given observation is the action for which the Q-function returns the highest value. At the start of the RL training process, the Q-function or the policy are usually randomly initialized. Thus, initially these functions do not contain

any knowledge about how to best solve the task. The RL agent explores the environment for some time steps (see Figure 2.1) and uses the gathered knowledge to update the Q-function or policy. One important aspect of this training process is the trade-off between exploration and exploitation of knowledge from past interactions with the environment. In the standard RL approaches, the exploration is conducted by a random selection of actions.

Neural networks together with increased computation power changed the field of RL significantly, because deep RL algorithms were proposed, which learn a neural network to approximate a policy or Q-function. For the research in this dissertation, we use different deep RL algorithms. To provide an overview of the used deep RL algorithms, we first define properties that clarify the fundamental differences between algorithms. Model-based algorithms learn or utilize a model of the environment. They use this environment model to predict the future evolution of the environment and increase sample efficiency by adding learning tuples (s_t, a_t, s_{t+1}, r_t) that are generated by this model. In contrast, model-free algorithms learn a policy or Q-function solely based on learning tuples generated by the agent interacting with the environment. These model-free algorithms are distinguished based on the origin of the learning tuples used for updating a policy or Q-function. On-policy algorithms only use the learning tuples generated since the last policy update. Off-policy algorithms can use all learning tuples generated in the training process for improving the policy or Q-function. Additionally, an important concept for many model-free RL algorithms is the actor-critic idea, where the policy is the actor and the critic is the state value function, for which the Q-function values are summed over actions. Based on the state value function and the reward function, an advantage function is computed and integrated in the loss function used for the policy update. In this work, we consider model-free on-policy and off-policy algorithms, which are the most commonly employed RL algorithms for motion planning tasks of autonomous vehicles [67].

Figure 2.2 provides an overview of the used model-free algorithms[1], and displays the development. The core ideas of the algorithms in Figure 2.2 are:

- Deep Q-network (DQN) [111] was the first approach using deep neural networks to approximate the Q-function. The main idea is to define a temporal difference error, which serves as a gradient for updating the Q-function.
- Important extensions of DQN are dueling DQN [112], double DQN [113], and experience replay [114], which mainly improve the robustness and sample efficiency of the original DQN.
- Trust Region Policy Optimization (TRPO) [115] addresses the instability of vanilla policy optimization [116] by adding a constraint, which ensures that the policy update is not too large.
- Asynchronous Advantage Actor-critic (A3C) [117] uses the actor-critic idea for policy optimization by estimating the policy and the value function with a neural network. Multiple actor-learners can be run in parallel to increase computational efficiency and asynchronously update the global actor network.

[1]Note that we did not utilize TRPO in our research and added it for completeness, as the mainly used PPO is based on it.

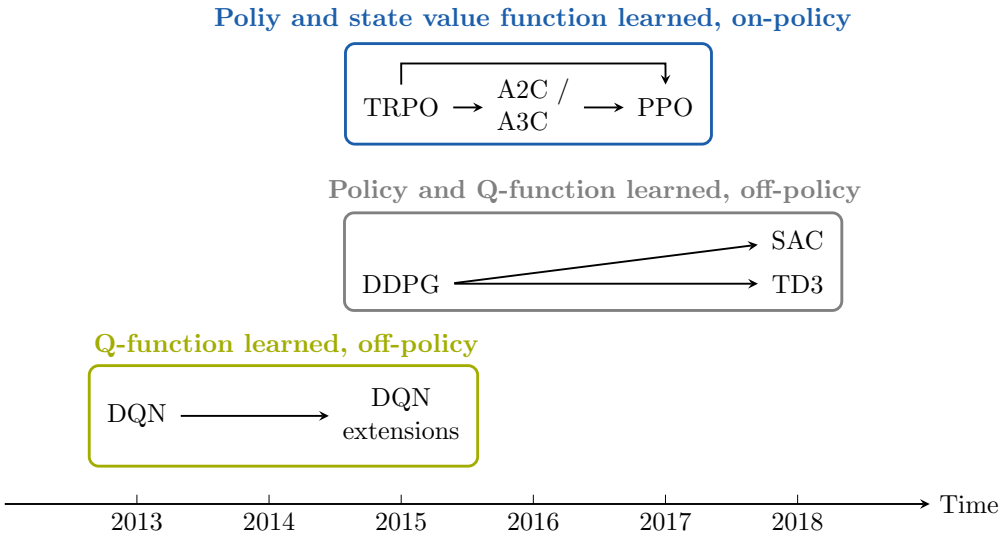


Figure 2.2: Evolution of prominent model-free deep RL algorithms. Note that the first algorithm for policy learning has been proposed in 1999 [116].

- Proximal Policy Optimization (PPO) [118] simplifies the idea of TRPO by introducing a surrogate objective, for which the maximal value is not too far from the current value, and thus, the update of the policy is not too large. PPO is normally implemented as an actor-critic formulation.
- Deep Deterministic Policy Gradient (DDPG) [119] extends the concept of DQN to continuous action spaces by learning an approximator for maximizing the Q-function. Based on this approximator, a deterministic policy can be learned by simple gradient ascent.
- The Twin Delayed Deep Deterministic (TD3) policy gradient algorithm [120] robustifies DDPG by taking the pessimistic Q-value of two learned Q-function networks when calculating the temporal difference error, updating the policy less frequently than the Q-networks, and adding a noise to the actions, which reduces unwanted exploitation of the Q-function.
- Soft Actor-critic (SAC) [121] is also based on DDPG, but learns a stochastic policy. Additionally, the algorithm is robustified by entropy regularization and two Q-function networks.

For implementation details, we refer the interested reader to the respective publications.

2.1.2 Formal Methods

Formal methods are rigorous methods for specifying and verifying systems [122]. In this section, we present three concepts for the formal verification methods we employed in Appendix A and Appendix B: temporal logic specifications, verification with set-based reachability analysis, and probabilistic verification based on scenario programs. For verification approaches, we need to provide a model and a specification. The verification algorithm then verifies or falsifies the

specification for the given model. In particular, for action verification of autonomous CPSs, we often have an ego system model and a prediction model for the evolution of the environment components.

Models

Systems are described depending on their operating space, e.g., discrete systems can be modeled by finite state machines and dynamic systems by differential equations. Motion planning tasks are usually real-world tasks and, thus, defined in a continuous space. A general nonlinear continuous-time system can be modeled by

$$\dot{x}(t) = f(x(t), u(t), w(t)), \tag{2.2}$$

where w is a time-dependent disturbance and stems from the disturbance set $\mathcal{W} \subset \mathbb{R}^Z$. Generally, the disturbance set can be unbounded. Then, often only probabilistic verification is meaningful. For CPS, which digitally compute control inputs, the control input can only be changed at discrete time steps. Additionally, digital sensors measure the system state at discrete times. Therefore, we often describe these systems by a discrete-time system

$$x_{t+1} = g(x_t, u_t, w_t), \tag{2.3}$$

where x_{t+1} denotes the next state and $w_t \in \mathcal{W}$ the disturbance at time t.

The disturbances for both system models can capture epistemic and aleatory uncertainties. Aleatory uncertainty describes system inherent uncertainty that cannot be reduced by a refined model whereas epistemic uncertainty is due to imperfection of the model. Thus, epistemic uncertainty is often dominating when we employ a model that is significantly simplified with respect to the true system model. We commonly use such simpler models because verification complexity reduces for them, leading to lower computation time. Another reason is that for dynamic obstacles in our systems, we often have to derive a model that encloses the behavior of multiple dynamic obstacles, e.g. all cars, based on only a few recorded trajectories. Thus, we need models that overestimate physical limits of dynamic obstacles as the simple point-mass model can do for cars.

Safety specifications

The most common safety specification is that unsafe states should be globally avoided. For example, an unsafe state for an autonomous vehicle is being too close to the leading vehicle, so if the leading vehicle suddenly breaks, a collision is inevitable. These sets of unsafe states are usually identified based on task knowledge, e.g., potential occupancy of traffic participants, or off-road states. Another perspective on this type of safety specifications is ensuring that the system stays within a safe set. In this case, a forward invariant set can be computed [123–126], which by definition ensures that there always exists a control input that keeps the system within the set for an infinite time horizon. For the autonomous driving example, this would be a set of states for which the vehicle can always come to a safe stop with respect to the leading

vehicle, even if the leading vehicle performs emergency braking.

Temporal logic describes a family of languages that can formalize temporal specifications. Temporal logic specifications are often evaluated on discrete-time traces of predicates where predicates detect specific system states or system properties. Next to the Boolean operators, e.g., $\wedge$, $\implies$, $\neg$, the most common temporal operators are G, F, and U. The future globally operator $\mathrm{G}(\phi)$ evaluates to true if and only if the predicate ϕ is true for all future time steps, whereas ϕ only has to be true for at least one future time step for the future operator $\mathrm{F}(\phi)$. The until operator $\phi_1 \,\mathrm{U}\, \phi_2$ returns true for a trace iff ϕ_1 holds true for all time steps until ϕ_2 holds true. The exact semantics and syntax are specified for each temporal logic language individually. However, based on this high-level notion of the global operators, we can introduce the two safety specifications used in this work.

Safety Specification 1 (Temporal logic)**.** *General safety specifications in temporal logic have the structure* $\mathrm{G}(\neg\, \Phi)$*, where* Φ *is a temporal logic formula describing situations that always should be avoided.*

Safety Specification 2 (Avoid)**.** *The avoid specification* $\mathrm{G}(\neg$`in_unsafe_state`$)$ *is a special case of Safety Specification 1 for which the temporal logic formula* Φ *is the predicate* `in_unsafe_state`*. The predicate* `in_unsafe_state` *evaluates to true if and only if the ego system is in an unsafe state. This is often computed by checking if the reachable states ego system intersect with sets of unsafe states.*

Commonly used temporal languages are LTL, Computation Tree Logic (CTL), and Signal Temporal Logic (STL) [127, 128]. Since we consider systems operating in continuous state and input spaces, we employed STL [129]. Additionally, we used Metric Temporal Logic (MTL) [130, 131] for time-discrete systems with expressive specifications, because MTL extends the LTL semantics by allowing for the evaluation of temporal operators for time intervals. The MTL formulas are based on Boolean predicates. STL evaluates the compliance of continuous signals while also allowing for temporal operators specified for time intervals. Additionally, a robustness measure ρ can be identified for a STL or MTL formula. This robustness measure quantifies the degree of compliance or violation and can be used for formulating optimization problems, which identify the highest possible system compliance.

Verification methods

We applied two verification methods in this dissertation: set-based reachability analysis [132–134] and probabilistic verification based on scenario programs [135–137]. Set-based reachability analysis algorithms compute the evolution of a closed-loop or open-loop dynamic system over time. The algorithms are based on set operations, which eases the integration of bounded uncertainties and approximation errors. In this work, we use algorithms that compute the forward reachable set:

$$\mathcal{R}(t_f) = \{\xi(t_f, x_0, u(\cdot), w(\cdot)) \in \mathcal{X} \,|x_0 \in \mathcal{X}_0, \forall t \in [t_0, t_f] : u(t) \in \mathcal{U}, w(t) \in \mathcal{W}\}, \tag{2.4}$$

where $\xi(t_f, x_0, u(\cdot), w(\cdot))$ denotes the state at time t_f when solving the models (2.2) or (2.3) for the control input trajectory $u(\cdot)$ and disturbance trajectory $w(\cdot)$ and the initial state x_0. For discrete-time systems, we can acquire ξ directly from (2.3), whereas for continuous-time systems, we need to compute a solution of the differential equations modeling the system. To obtain time-interval forward reachable sets from time-point forward reachable sets, an enclosure of the reachable sets between two time points is computed. As the physical part of the CPS evolves continuously, we require time-interval reachable sets to verify that the computed control input is safe over the entire time horizon. Once the time-interval reachable sets are computed, we verify that they do not intersect with unsafe sets or are fully contained in a safe set. If this is the case, the system will not enter an unsafe state or leave the safe set for the evaluated time horizon. The algorithms for computing reachable sets depend on the system model and the chosen set representations.

The second verification approach we employ is probabilistic verification based on the concept of scenario programs. The idea is that we obtain $K \in \mathbb{N}^+$ trajectories $\{\zeta_{p_i}\}_{i=1}^{K}$ by uniformly sampling initial conditions p_i from a parameter space $\mathcal{P}$, whereby a trajectory ζ is a sequence of system states x_j, $\forall j \in [t_0, t_t]$ with t_0 and t_t being the initial and last time step. For each sampled trajectory, we evaluate the robustness measure ρ that corresponds to a STL formula Ψ. Then, we identify the minimal robustness value ρ_K^*. Based on [138, Thm. 2], we can describe the probability and confidence that ρ_K^* underperforms the $(1-\epsilon)$-percentile:

$$\mathbb{P}_\mu^K \left[\mathbb{P}_\mu[\rho(\zeta_p) \geq \rho_K^*] \geq 1 - \epsilon \right] \geq 1 - (1-\epsilon)^K, \tag{2.5}$$

where μ indicates the distribution from which the trajectory samples are drawn. We only sample trajectories to obtain the probabilistic guarantee; thus, no explicit knowledge of a system model is necessary. Additionally, we are not limited to systems with bounded disturbances, but only require that the underlying distribution from which we generate the trajectories is time-invariant.

2.2 Problem Statement

Safety is essential for many motion planning tasks of CPS. For most of these tasks, safety is specified in natural language written in a legal document or as a system requirement. However, natural language specifications leave room for interpretation, which makes them difficult to integrate into motion planning approaches for autonomous vehicles.

Problem 1. *Safety specifications for real-world motion planning tasks are likely ambiguous and complex when the intentions and behavior of other agents in the environment are unknown.*

Motion planning tasks are often intricate, i.e., multiple dynamic obstacles with unclear intentions are present, there are uncertainties in the observations, and environmental constraints need to be considered. Model-free RL has shown that it can accomplish impressive and generalizing performance for such tasks [62, 64, 65]. However, most motion planning tasks are safety-critical, i.e., collisions should never occur and traffic rules must be respected. This criticality makes standard RL inapplicable due to its inherent randomness and the need to explore

unsafe behavior to learn from it. Even if we can explore unsafe behavior in simulation, we often cannot test or verify all admissible system states in simulation before deploying a motion planner on an autonomous vehicle in the physical world. Thus, we are either satisfied with probabilistic safety guarantees based on a formal safety specification or need a model to employ online verification algorithms. However, it is challenging to achieve efficient online verification, especially for complex motion planning tasks with complex safety specifications.

Problem 2. *Can we design safe and efficient algorithms for motion planning tasks by combining model-free RL and formal methods?*

2.3 Solution Concept

To address these problems, we propose formalizing the notion of safety and integrating formal methods into the RL training and deployment to provide safety guarantees. We detail the solution steps in this section.

Formalizing safety

To apply verification algorithms, we need a formal safety specification and a model for the safety-relevant part of the complete system. We can use Safety Specification 2 to formalize safety for many motion planning tasks. In practice, we compute unsafe sets based on obstacles, with which the ego system should not collide, or invariably safe sets based on the ego system dynamics and obstacles. Yet, some safety specifications are not trivially transformable into avoidance of unsafe states, e.g., if a specific maneuver has to be performed. In these cases, we use Safety Specification 1, which allows us to formalize arbitrary temporal safety specifications based on signals or predicates depending on the temporal logic language. To obtain hard safety guarantees, a model that conformantly describes all safety-relevant system behaviors needs to be available. While for the ego system we often know a model, for dynamic obstacles, we usually employ simple models such as point-mass or kinematic models with bounded disturbance since they allow for more efficient verification algorithms and conveniently overapproximate the physical possible behaviors of dynamic obstacles. For the employed probabilistic verification approach, we do not require an explicit system model, but only an environment that generates system trajectories. Based on the safety specification and the model, we can then use verification methods to verify the safety of actions, states, or trajectories.

Safe reinforcement learning

The RL setup consists of an environment to interact with, a reward function, an observation space, and potentially a translation of discrete actions from the action space to the input space of the ego system. A well-tuned reward function reflecting the tasks is essential to train RL agents successfully, while a task usually consists of safety and performance components. Since verification algorithms handle the safety objective, one could argue that the reward function must mainly reflect performance objectives. This assumption would also ease training since safety and performance can conflict in some situations. Still, including safety objectives in

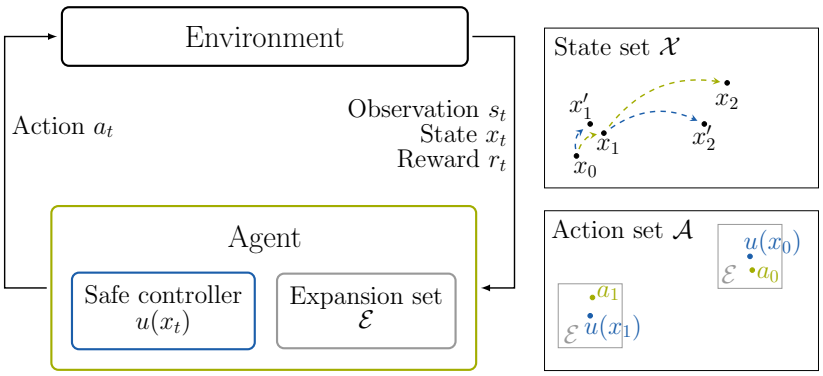


Figure 2.3: Safe RL process guided by a safe controller on the left. On the right is an example trajectory of an RL agent in the state set and the corresponding action selection in the action set, which is constrained by the safe controller and expansion set.

the reward can lead to an RL agent that relies less on the verification algorithm. Next to the reward function, the observation space is designed to inform the agent about its own state and the environment situation. The environment to interact with includes the ego system dynamics and the possibility of evolving the environment situation, mainly dynamic obstacles, over time. In particular, for the dynamic obstacles in the environment, this can be built on a (reactive) model, e.g., an intelligent driver model [139], on recorded environment trajectories, or a combination of both. Given such a RL setup, we employ two ideas to achieve safe RL.

Idea 1 (Safe reinforcement learning with probabilistic guarantees)**.** *Given a safety specification, controller, stochastic disturbance, and environment, we can determine the probabilistic safety guarantee for the system. We can then use RL to improve performance objectives while constraining the RL action space to the disturbance set around the control input to approximately maintain the probabilistic guarantee.*

As the controller only needs to be probabilistically verifiable for a safety specification, it can either be synthesized based on this specification or learned from safe demonstrations. The reward function of the RL setup does not need to include safety components, because the action space is constrained based on the previously verified safe controller. Figure 2.3 displays the guided RL process. The controller provides the control input at each time step. The RL agent can then select an action in the expansion set around it, which is the same as the disturbance set used during the previous controller verification step. Due to the distribution shift from uniform sampling of disturbances to the learned policy, the probabilistic safety guarantee is only approximately maintained and needs to be re-verified as the last step. We present this concept in detail and evaluate it for a safe evasion task of a mobile robot in Appendix B.2.

Idea 2 (Provably safe reinforcement learning)**.** *Given a safety specification and a model, an online verification algorithm is developed to identify unsafe actions and safe actions based on the current state of ego systems. This online verification is integrated into RL during training and deployment, and ensures by design that only safe actions are executed, i.e., provides hard guarantees.*

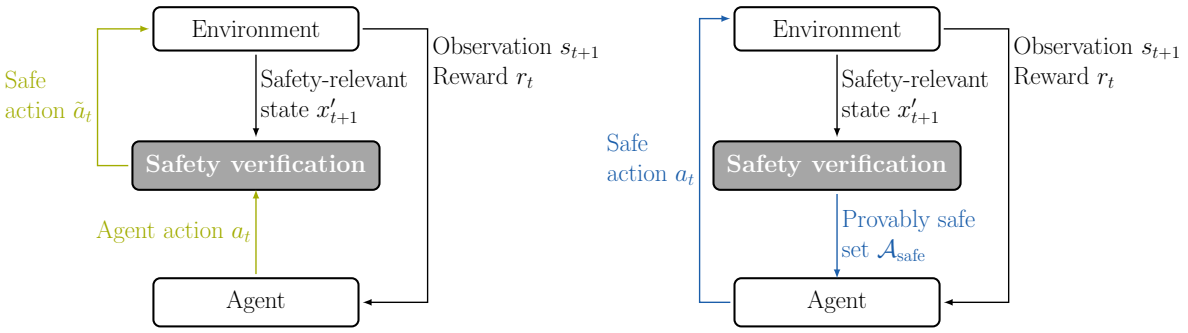


Figure 2.4: Provably safe RL processes: on the left action projection and action replacement are summarized by the green path; on the right action masking is described by the blue path.

Since the fundamental learning process of model-free RL algorithms is always as depicted in Figure 2.1, the online safety verification module can be added between the environment and agent while still being transferable between RL algorithms. The integration of the safety verfication module leads to two provably safe RL processes, which are illustrated in Figure 2.4. Conceptually, the online verification can either provide the safe action set $\mathcal{A}_{\text{safe}}(x')$ given the safety-relevant states x' of the ego system and of environment components or, alternatively, verify the action a proposed by the agent and correct it to a safe action $\tilde{a}$ if necessary. Approaches for which a safe set is computed before the agent selects an action are action masking approaches. For approaches that correct actions after the agent proposed one, we differentiate between action projection and action replacement. Action projection identifies the safe action closest to the action proposed by the agent. In contrast, action replacement corrects unsafe actions with a safe action independent of the action proposed by the agent. For motion planning tasks, the safety-relevant states x' commonly contain uncertain positions and velocities of other agents or static obstacles in the environment. We use this information to compute overapproximative unsafe sets, which are then used as an input to the verification algorithm.

Appendix A presents the publications executing provably safe RL for avoidance safety specifications. In particular, we conceptualize, survey, and benchmark provably safe RL literature in Appendix A.1. Appendix A.2 proposes an application-independent action projection approach using set-based reachability analysis such that the closest safe action is identified in a continuous actions space. In Appendix A.3 and Appendix A.4, we present two action masking approaches for autonomous driving on highways and at urban intersections. Lastly, in Appendix B.1, we formalize complex safety specifications, which reflect maritime traffic rules, and propose an action masking approach for autonomous vessels navigating on the open sea that complies with these complex safety specifications in Appendix B.3.

3 Discussion and Conclusion

3.1 Provably Safe Reinforcement Learning for Guaranteed Collision Avoidance

Summary The papers reproduced in Appendix A propose provably safe RL approaches for Safety Specification 2. We introduce the main concepts for provably safe RL and identify three distinct approaches to correct actions so that only safe actions are executed: action replacement, action projection, and action masking. The clarified terminology and concepts, as well as the systematic literature review in Appendix A.1, supports researchers to more clearly present their provably safe RL approaches and to easily gain an overview of provably safe RL research. Furthermore, our conceptualization facilitates the identification of research gaps in provably safe RL. The benchmarking of provably safe RL hints toward significant empirical differences between the approaches to correct the action and identifies action replacement as the best-suited approach for the two examined control benchmarks. However, the deciding factor for selecting a provably safe RL approach is often the ease of integration of the safety specification into the RL process. Thus, all three approaches are useful.

The article in Appendix A.2 introduces an action projection approach based on set-based reachability analysis for continuous action spaces. The action projection approach can be applied for nonlinear continuous systems and potentially time-varying unsafe sets. We use a zonotope as the action set and perform forward reachability analysis such that the relation between actions and the reachable sets is preserved. The reachable sets are intersected with the unsafe sets and result in constraints that depend on the action space parameters. These constraints are integrated into a constraint optimization problem that identifies the safe action closest to the action proposed by the RL agent. We evaluate our approach on four motion planning tasks ranging from avoiding static obstacles with the F1TENTH car to a quadrotor that tracks a trajectory while avoiding unsafe states. The results show the real-time capability, positive effects on training convergence, and the feasibility of handling time-varying unsafe sets for an ego system with nonlinear dynamics.

Finally, we develop two action masking approaches for the application of autonomous driving in Appendix A.3 and Appendix A.4. We utilize discrete action spaces that reflect meaningful high-level decisions and transform them into control inputs with a low-level planner. Safety is ensured if a trajectory corresponding to a high-level decision does not intersect unsafe sets and is contained in invariably safe sets in the state space. The unsafe and safe sets are computed with set-based reachability analysis. If no high-level decision can be verified, we engage a fail-safe planner, which is either a provably safe adaptive cruise controller or a braking trajectory.

Our empirical evaluation validates that our verification approach eliminates collisions caused by the ego vehicle. For highway driving, we observe that adding the safety verification improves training convergence while only slightly impairing goal-reaching performance. For urban intersection driving, the safety verification part, which ensures safe behavior for lane-changing and lane-following, shows higher goal-reaching rates than the unsafe baseline. However, lane safety is insufficient to guarantee that the ego vehicle does not cause collisions when driving at urban intersections. Adding intersection safety verification ensures collision avoidance but significantly reduces the goal-reaching performance. The reduction in performance mainly originates from the ego vehicle waiting until the intersection is surely free, as the intersection safety verification does not consider right-of-way traffic rules at intersections. The performance drop clearly demonstrates the boundaries of safety specifications through safe and unsafe sets only and motivates our work in Appendix B.

Future work One of the main benefits of provably safe RL is that it is based on online verification, i.e., we verify the safety of actions based on the current system state, which includes the states of environment components. This feature makes provably safe RL particularly well-suited for real world deployment. However, most provably safe RL approaches, including ours, are mainly evaluated in simulation. We showcase real-time capability on the F1TENTH car with the action projection approach, but more thorough testing on other physical systems is needed to evaluate if the runtime efficiency is sufficient for higher-dimensional systems operating at higher frequencies. Additionally, our approaches assume that computations happen instantaneously. Thus, future work should extend the online verification to anytime verification approaches similar to [140]. To showcase that RL is well-suited for complex tasks such as autonomous urban driving, the autonomous vehicle EDGAR [141] could be a state-of-the-art test platform for the proposed provably safe RL approaches.

So far, most provably safe RL approaches are tailored to a specific application. Our benchmarking of provably safe RL approaches is one of few works that compares different provably safe RL approaches on benchmarks. Thus, it is theoretically and empirically not well understood which approaches are best suited or even feasible for which task specifications. Open-sourcing code together with papers helps other researchers to reproduce and compare their approaches to existing ones, eventually easing the empirical investigation of differences and applicability. To this end, we publish the source code whenever we own all necessary intellectual property rights and further contribute by developing the open-source software tools CommonRoad-RL [K9] and CommonOcean [K8]. There are only limited works [37, 39, 45] that theoretically investigate the influence of provably safe RL approaches on the learning process. Future research in this direction might identify levers to improve sample efficiency and convergence of provably safe RL approaches.

3.2 Reinforcement Learning with Temporal Logic Safety Specifications

Summary Temporal logic is well suited to specify safety specifications for motion planning tasks of CPS since they often include temporal dependencies. Since safety requirements are typically stated in natural language, which is not directly interpretable for a CPS, safety specifications need to be formalized with temporal logic in a first step. Our research, which is reproduced in Appendix B.1, presents how maritime traffic rules for collision avoidance on the open sea can be transferred from natural language to MTL formulas. We implement a monitor for the formalized rules and evaluate them on encounter situations of vessels based on recorded maritime traffic data. Our results align with our expectation that most vessels on the open sea comply with the specified maritime traffic rules. Given the formalized safety specification of the traffic rules, it is an open research question how to best ensure that CPS operating in a continuous state space comply with them. We approach these question with a safe RL approach for continuous action spaces that generalizes between systems and provides probabilistic safety guarantees in Appendix B.2. Our second solution, presented in Appendix B.3, is a provably safe RL approach for discrete action spaces that entails an online verification approach to identify rule-compliant action for navigation of autonomous vessels on the open sea.

For the safe RL approach with probabilistic guarantees in Appendix B.2, we use probabilistic verification to obtain probabilistic bounds on the minimal robustness measure for a system. The robustness measure quantifies the satisfaction or violation of a temporal logic specification. To address the issue of potentially conflicting performance and safety objectives, our approach combines probabilistic verification and safe RL in separate steps. First, a stochastic safe controller is verified with respect to the temporal logic safety objective. This controller is then used to constrain the RL action space while the RL agent optimizes for performance. The advantage of our proposed approach is that only limited system knowledge is necessary to obtain an RL agent that is both safe and performant. Yet, a critical pre-condition is that we can obtain a safe controller that is initially verifiable.

Our provably safe RL approach for autonomous vessels navigating on the open sea in Appendix B.3 extends the formalized maritime traffic rules and develops a hierarchical safety verification approach for discrete action spaces. For that, we introduce a rule-compliant maneuver synthesis and an emergency verification. If all vessels comply with the regular collision avoidance rules for vessels on the open sea, the synthesized maneuvers lead to collision avoidance. However, if a vessel does not comply with the collision avoidance rules, the other vessel is obliged to perform an emergency maneuver that minimizes the risk of collision. This emergency maneuvering is reflected by our emergency controller and detected by the emergency verification. The result of the safety verification is the set of traffic rule compliant actions, which are subsequently used to constrain the selection of actions of the RL agent. In other words, unsafe actions are masked out for the RL agent. We evaluate our approach in critical maritime traffic situations and observe that our approach guarantees traffic rule compliance and achieves collision avoidance without significantly impairing goal-reaching performance.

Future work The satisfaction of a temporal logic formula often depends on the future states of environment components. For example, the safety of a collision avoidance maneuver for an autonomous vessel on the open sea depends on the maneuvering of the other vessel. As the future can only be coarsely predicted, this poses the challenge of making decisions based on uncertain predictions. If these predictions enclose all admissible behaviors, they tend to significantly constrain the decision space. Conversely, if only the most likely prediction is considered, we might miss out on the consequences of almost equally likely predictions. Thus, future work should investigate how mixed prediction approaches, e.g., a combination of set-based prediction and probabilistic predictions [142], can be integrated in safe RL approaches. Another direction is to design more sophisticated probabilistic predictions that are learned based on traffic data, e.g., predictions based on neural networks [143–145].

Another open research problem is the identification of efficient and general approaches for motion planning tasks in continuous state and action spaces with uncontrollable environment components, e.g., traffic participants, that are always compliant with complex temporal logic specifications of traffic rules. First approaches in this direction combine capable model checking techniques available for discrete spaces with set-based prediction methods to obtain specification-compliant continuous state sets, e.g., for autonomous driving [146]. These safe state sets could be combined together with a variation of the action projection approach proposed in Appendix A.2 to achieve provably safe action projection. Another solution to this research problem could involve automaton learning based on a continuous system [147], where the automaton predicts specification compliance of continuous actions that are abstracted to discrete actions.

In this dissertation, we propose multiple safe RL approaches that achieve probabilistic or hard safety guarantees. We consider different degrees of expressiveness for the guaranteed safety specifications and evaluate our approaches on motion planning tasks for autonomous vehicles. We show that hard safety guarantees can be achieved given a safety-conformant system model, and probabilistic guarantees can be realized for arbitrary robustness measures of temporal logic formulas without an explicit system model. Thus, our work paves the way toward RL agents for motion planning tasks of CPSs that exhibit guarantees for complex safety specifications.

Abbreviations

A3C Asynchronous Advantage Actor-critic.

COLREGS Convention on the International Regulations for Preventing Collisions at Sea.

CPS cyber-physical system.

CTL Computation Tree Logic.

DDPG Deep Deterministic Policy Gradient.

DQN Deep Q-network.

HJI Hamiltonian-Jacobi-Issacs.

LTL Linear Temporal Logic.

MDP Markov Decision Process.

MPC model predictive control.

MTL Metric Temporal Logic.

PPO Proximal Policy Optimization.

RL reinforcement learning.

SAC Soft Actor-critic.

STL Signal Temporal Logic.

TD3 Twin Delayed Deep Deterministic.

TRPO Trust Region Policy Optimization.

List of Figures

Bibliography

[1] J. García and F. Fernández. "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16.42 (2015), pp. 1437–1480.

[2] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig. "Safe learning in robotics: From learning-based control to safe reinforcement learning". In: *Annual Review of Control, Robotics, and Autonomous Systems* 5 (2022), pp. 411–444.

[3] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura. "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2018, pp. 2034–2039.

[4] B. Lütjens, M. Everett, and J. P. How. "Safe reinforcement learning with model uncertainty estimates". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2019, pp. 8662–8668.

[5] M. El-Shamouty, X. Wu, S. Yang, M. Albus, and M. F. Huber. "Towards safe human-robot collaboration using deep reinforcement learning". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2020, pp. 4899–4905.

[6] A. Heiberg, T. N. Larsen, E. Meyer, A. Rasheed, O. San, and D. Varagnolo. "Risk-based implementation of COLREGs for autonomous surface vehicles using deep reinforcement learning". In: *Neural Networks* 152 (2022), pp. 17–33.

[7] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta. "Q-learning for robust satisfaction of signal temporal logic specifications". In: *Proc. of the IEEE Conf. on Decision and Control (CDC)*. 2016, pp. 6565–6570.

[8] A. Camacho, R. T. Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. "LTL and beyond: Formal languages for reward function specification in reinforcement learning". In: *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*. 2019, pp. 6065–6073.

[9] S. Arora and P. Doshi. "A survey of inverse reinforcement learning: Challenges, methods and progress". In: *Artificial Intelligence* 297.103500 (2021).

[10] E. Altman. "Constrained Markov decision processes with total cost criteria: Lagrangian approach and dual linear program". In: *Mathematical Methods of Operations Research* 48.3 (1998), pp. 387–417.

[11] J. Achiam, D. Held, A. Tamar, and P. Abbeel. "Constrained policy optimization". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. 2017, pp. 22–31.

[12] Y. Chow, O. Nachum, E. Duéñez-Guzmán, and M. Ghavamzadeh. "A Lyapunov-based approach to safe reinforcement learning". In: *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*. 2018, pp. 8103–8112.

[13] A. Stooke, J. Achiam, and P. Abbeel. "Responsive safety in reinforcement learning by PID Lagrangian methods". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. 2020, pp. 9133–9143.

[14] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge. "Projection-based constrained policy optimization". In: *Proc. of the Int. Conf. on Learning Representations (ICLR)*. 2020, pp. 1–21.

[15] Z. Marvi and B. Kiumarsi. "Safe reinforcement learning: A control barrier function optimization approach". In: *Int. Journal of Robust and Nonlinear Control* 31.6 (2021), pp. 1923–1940.

[16] D. Kim and S. Oh. "TRC: Trust region conditional value at risk for safe reinforcement learning". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 2621–2628.

[17] G. De Giacomo, L. Iocchi, M. Favorito, and F. Patrizi. "Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications". In: *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. 2021, pp. 128–136.

[18] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee. "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees". In: *Proc. of the IEEE Conf. on Decision and Control (CDC)*. 2019, pp. 5338–5343.

[19] M. Hasanbeig, D. Kroening, and A. Abate. "Towards verifiable and safe model-free reinforcement learning". In: *Proc. of the Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis*. 2019, pp. 1–9.

[20] M. Hasanbeig, A. Abate, and D. Kroening. "Cautious reinforcement learning with logical constraints". In: *Proc. of the Int. Conf. on Autonomous Agents and Multi Agent Systems (AAMAS)*. 2020, pp. 483–491.

[21] M. Zanon and S. Gros. "Safe reinforcement learning using robust MPC". In: *IEEE Transactions on Automatic Control* 66.8 (2021), pp. 3638–3652.

[22] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa. "Safe exploration in continuous action spaces". In: *arXiv:1801.08757* (2018).

[23] M. Turchetta, F. Berkenkamp, and A. Krause. "Safe exploration in finite Markov decision processes with Gaussian processes". In: *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*. 2016, pp. 4312–4320.

[24] F. Berkenkamp, A. P. Schoellig, M. Turchetta, and A. Krause. "Safe model-based reinforcement learning with stability guarantees". In: *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*. 2017, pp. 908–918.

[25] T. Mannucci, E.-J. van Kampen, C. de Visser, and Q. Chu. "Safe exploration algorithms for reinforcement learning controllers". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.4 (2018), pp. 1069–1081.

[26] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg. "Recovery RL: Safe reinforcement learning with learned recovery zones". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4915–4922.

[27] J. H. Gillula and C. J. Tomlin. "Reducing conservativeness in safety guarantees by learning disturbances online: Iterated guaranteed safe online learning". In: *Robotics: Science and Systems* 8.1 (2013), pp. 81–88.

[28] B. Könighofer, J. Rudolf, A. Palmisano, M. Tappler, and R. Bloem. "Online shielding for stochastic systems". In: *Proc. of the NASA Formal Methods (NFM)*. 2021, pp. 231–248.

[29] C. Yang, J. Liu, H. Sun, J. Sun, X. Chen, and L. Zhang. "Safe reinforcement learning for CPSs via formal modeling and verification". In: *Proc. of the IEEE Int. Joint Conf. on Neural Networks Proc. (IJCNN)*. 2021, pp. 1–8.

[30] O. Bastani, Y. Pu, and A. Solar-Lezama. "Verifiable reinforcement learning via policy extraction". In: *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*. 2018, pp. 2499–2509.

[31] H. D. Tran, F. Cai, D. Manzanas Lopez, P. Musau, T. T. Johnson, and X. Koutsoukos. "Safety verification of cyber-physical systems with reinforcement learning control". In: *ACM Transactions on Embedded Computing Systems* 18.5s (2019), pp. 1–22.

[32] L. M. Schmidt, G. D. Kontes, A. Plinge, and C. Mutschler. "Can you trust your autonomous car? Interpretable and verifiably safe reinforcement learning". In: *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*. 2021, pp. 171–178.

[33] A. K. Akametalu, S. Kaynama, J. F. Fisac, M. N. Zeilinger, J. H. Gillula, and C. J. Tomlin. "Reachability-based safe learning with Gaussian processes". In: *Proc. of the IEEE Conf. on Decision and Control (CDC)*. 2014, pp. 1424–1431.

[34] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. "Safe reinforcement learning via shielding". In: *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*. 2018, pp. 2669–2678.

[35] B. Könighofer, F. Lorber, N. Jansen, and R. Bloem. "Shield synthesis for reinforcement learning". In: *Proc. of the Int. Symposium on Leveraging Applications of Formal Methods (ISoLA)*. 2020, pp. 290–306.

[36] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan. "Reachability-based trajectory safeguard (RTS): A safe and fast reinforcement learning safety layer for continuous control". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3663–3670.

[37] N. Hunt, N. Fulton, S. Magliacane, T. N. Hoang, S. Das, and A. Solar-Lezama. "Verifiably safe exploration for end-to-end reinforcement learning". In: *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*. 2021, pp. 1–11.

[38] K. P. Wabersich and M. N. Zeilinger. "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems". In: *Automatica* 129.1 (2021), pp. 109597–109614.

[39] S. Gros, M. Zanon, and A. Bemporad. "Safe reinforcement learning via projection on a safe set: How to achieve optimality?" In: *IFAC-PapersOnLine* 53.2 (2020), pp. 8076–8081.

[40] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks". In: *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*. 2019, pp. 3387–3395.

[41] Z. Marvi and B. Kiumarsi. "Reinforcement learning with safety and stability guarantees during exploration for linear systems". In: *IEEE Open Journal of Control Systems* 1 (2022), pp. 322–334.

[42] X. Li, Z. Serlin, G. Yang, and C. Belta. "A formal methods approach to interpretable reinforcement learning for robotic planning". In: *Science Robotics* 4.37 (2019).

[43] N. Fulton and A. Platzer. "Safe reinforcement learning via formal methods: Toward safe control through proof and learning". In: *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*. 2018, pp. 6485–6492.

[44] N. Fulton and A. Platzer. "Verifiably safe off-model reinforcement learning". In: *Proc. of the Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 2019, pp. 413–430.

[45] S. Huang and S. Ontañón. "A closer look at invalid action masking in policy gradient algorithms". In: *The Int. Florida Artificial Intelligence Research Society Conf. Proc. (FLAIRS)* 35 (2022).

[46] M. Anand and M. Zamani. "Formally verified neural network control barrier certificates for unknown systems". In: *IFAC-PapersOnLine* 56.2 (2023), pp. 2431–2436.

[47] L. de Moura and N. Bjørner. "Z3: An efficient SMT solver". In: *Proc. of the Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 2008, pp. 337–340.

[48] G. Anderson, A. Verma, I. Dillig, and S. Chaudhuri. "Neurosymbolic reinforcement learning with formally verified exploration". In: *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*. Vol. 33. 2020, pp. 6172–6183.

[49] B. Zhong, S. Liu, M. Caccamo, and M. Zamani. "Towards trustworthy ai: sandboxing ai-based unverified controllers for safe and secure cyber-physical systems". In: *Proc. of the IEEE Conf. on Decision and Control (CDC)*. 2023, pp. 1833–1840.

[50] M. Selim, A. Alanwar, S. Kousik, G. Gao, M. Pavone, and K. H. Johansson. "Safe reinforcement learning using black-box reachability analysis". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10665–10672.

[51] M. Selim, A. Alanwar, M. W. El-Kharashi, H. M. Abbas, and K. H. Johansson. "Safe reinforcement learning using data-driven predictive control". In: *Proc. of the Int. Conf. on Communications, Signal Processing, and their Applications (ICCSPA)*. 2022, pp. 1–6.

[52] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. "A general safety framework for learning-based control in uncertain robotic systems". In: *IEEE Transactions on Automatic Control* 64.7 (2019), pp. 2737–2752.

[53] O. Bastani. "Safe reinforcement learning with nonlinear dynamics via model predictive shielding". In: *Proc. of the American Control Conf. (ACC)*. 2021, pp. 3488–3494.

[54] X. Li, C.-I. Vasile, and C. Belta. "Reinforcement learning with temporal logic rewards". In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2017, pp. 3834–3839.

[55] P. Varnai and D. V. Dimarogonas. "On robustness metrics for learning STL tasks". In: *Proc. of the American Control Conf. (ACC)*. 2020, pp. 5394–5399.

[56] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. "Omega-regular objectives in model-free reinforcement learning". In: *Proc. of the Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. 2019, pp. 395–412.

[57] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan. "Modular deep reinforcement learning for continuous motion planning with temporal logic". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7973–7980.

[58] M. Hasanbeig, D. Kroening, and A. Abate. "LCRL: Certified policy synthesis via logically-constrained reinforcement learning". In: *Proc. of the Int. Conf. on Quantitative Evaluation of Systems (QEST)*. 2022, pp. 217–231.

[59] R. Alur, O. Bastani, K. Jothimurugan, M. Perez, F. Somenzi, and A. Trivedi. "Policy synthesis and reinforcement learning for discounted LTL". In: *Proc. of the Int. Conf. on Computer Aided Verification (CAV)*. 2023, pp. 415–435.

[60] R. Alur, S. Bansal, O. Bastani, and K. Jothimurugan. "A framework for transforming specifications in reinforcement learning". In: *Principles of Systems Design*. Springer Nature Switzerland, 2022, pp. 604–624.

[61] C. Yang, M. L. Littman, and M. Carbin. "On the (in)tractability of reinforcement learning for LTL objectives". In: *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)* (2022), pp. 3650–3658.

[62] H. Guo, F. Wu, Y. Qin, R. Li, K. Li, and K. Li. "Recent trends in task and motion planning for robotics: A survey". In: *ACM Computing Surveys* 55.13s (2023).

[63] H. Chai, Y. Li, R. Song, G. Zhang, Q. Zhang, S. Liu, J. Hou, Y. Xin, M. Yuan, G. Zhang, and Z. Yang. "A survey of the development of quadruped robots: Joint configuration, dynamic locomotion control method and mobile manipulation approach". In: *Biomimetic Intelligence and Robotics* 2.1 (2022), p. 100029.

[64] S. Teng, X. Hu, P. Deng, B. Li, Y. Li, Y. Ai, D. Yang, L. Li, Z. Xuanyuan, F. Zhu, and L. Chen. "Motion planning for autonomous driving: The state of the art and future perspectives". In: *IEEE Transactions on Intelligent Vehicles* 8.6 (2023), pp. 3692–3711.

[65] G. Kulathunga and A. Klimchik. "Survey on motion planning for multirotor aerial vehicles in plan-based control paradigm". In: *Remote Sensing* 15.21 (2023).

[66] A. Wibisono, M. J. Piran, H.-K. Song, and B. M. Lee. "A survey on unmanned underwater vehicles: Challenges, enabling technologies, and future research directions". In: *Sensors* 23.17 (2023).

[67] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Sallab, S. Yogamani, and P. Perez. "Deep reinforcement learning for autonomous driving: A survey". In: *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2022), pp. 4909–4926.

[68] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li. "End-to-end autonomous driving: Challenges and frontiers". In: *arXiv:2306.16927* (2023).

[69] A. Liniger, A. Domahidi, and M. Morari. "Optimization-based autonomous racing of 1:43 scale RC cars". In: *Optimal Control Applications and Methods* 36.5 (2015), pp. 628–647.

[70] P. Scheffe, T. M. Henneken, M. Kloock, and B. Alrifaee. "Sequential convex programming methods for real-time optimal trajectory planning in autonomous vehicle racing". In: *IEEE Transactions on Intelligent Vehicles* 8.1 (2023), pp. 661–672.

[71] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. "Path planning for autonomous vehicles in unknown semi-structured environments". In: *Int. Journal of Robotics Research* 29.5 (2010), pp. 485–501.

[72] W. Xu, J. Pan, J. Wei, and J. M. Dolan. "Motion planning under uncertainty for on-road autonomous driving". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2014, pp. 2507–2512.

[73] M. Werling, J. Ziegler, S. Kammel, and S. Thrun. "Optimal trajectory generation for dynamic street scenarios in a Frenet frame". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2010, pp. 987–993.

[74] G. Würsching and M. Althoff. "Sampling-based optimal trajectory generation for autonomous vehicles using reachable sets". In: *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*. 2021, pp. 828–835.

[75] H. Li, G. Yu, B. Zhou, P. Chen, Y. Liao, and D. Li. "Semantic-level maneuver sampling and trajectory planning for on-road autonomous driving in dynamic scenarios". In: *IEEE Transactions on Vehicular Technology* 70.2 (2021), pp. 1122–1134.

[76] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu. "Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications". In: *IEEE/ASME Transactions on Mechatronics* 21.2 (2016), pp. 740–753.

[77] A. Prakash, A. Behl, E. Ohn-Bar, K. Chitta, and A. Geiger. "Exploring data aggregation in policy learning for vision-based urban autonomous driving". In: *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[78] J. Zhang and K. Cho. "Query-efficient imitation learning for end-to-end simulated driving". In: *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*. Vol. 31. 1. 2017, pp. 2891–2897.

[79] F. Codevilla, E. Santana, A. M. Lopez, and A. Gaidon. "Exploring the limitations of behavior cloning for autonomous driving". In: *Proc. of the IEEE/CVF Int. Conf. on Computer Vision (ICCV)*. 2019.

[80] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. "CARLA: An open urban driving simulator". In: *Proc. of the Annual Conf. on Robot Learning*. 2017, pp. 1–16.

[81] S. Hu, L. Chen, P. Wu, H. Li, J. Yan, and D. Tao. "ST-P3: End-to-end vision-based autonomous driving via spatial-temporal feature learning". In: *Proc. of the European Conf. on Computer Vision*. 2022, pp. 533–549.

[82] G. Lee, D. Kim, W. Oh, K. Lee, and S. Oh. "MixGAIL: Autonomous driving using demonstrations with mixed qualities". In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2020, pp. 5425–5430.

[83] H. Wang, P. Cai, Y. Sun, L. Wang, and M. Liu. "Learning interpretable end-to-end vision-based motion planning for autonomous driving with optical flow distillation". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2021, pp. 13731–13737.

[84] Z. Cao, S. Xu, X. Jiao, H. Peng, and D. Yang. "Trustworthy safety improvement for autonomous driving using reinforcement learning". In: *Transportation Research Part C: Emerging Technologies* 138.103656 (2022).

[85] L. Wen, J. Duan, S. E. Li, S. Xu, and H. Peng. "Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization". In: *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–7.

[86] Z. Li, U. Kalabic, and T. Chu. "Safe reinforcement learning: Learning with supervision using a constraint-admissible set". In: *Proc. of the American Control Conf. (ACC)*. 2018, pp. 6390–6395.

[87] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. "Learning to drive in a day". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2019, pp. 8248–8254.

[88] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone. "Reward (mis)design for autonomous driving". In: *Artificial Intelligence* 316.103829 (2023).

[89] N. Mehdipour, M. Althoff, R. D. Tebbens, and C. Belta. "Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges". In: *Automatica* 152.110692 (2023).

[90] X. Wang. "Ensuring safety of learning-based motion planners using control barrier functions". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4773–4780.

[91] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker. "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning". In: *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2156–2162.

[92] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer. "Safe reinforcement learning with scene decomposition for navigating complex urban environments". In: *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1469–1476.

[93] T. I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011. ISBN: 9781119991496.

[94] E. Meyer, A. Heiberg, A. Rasheed, and O. San. "COLREG-compliant collision avoidance for unmanned surface vehicle using deep reinforcement learning". In: *IEEE Access* 8 (2020), pp. 165344–165364.

[95] D.-H. Chun, M.-I. Roh, H.-W. Lee, J. Ha, and D. Yu. "Deep reinforcement learning-based collision avoidance for an autonomous ship". In: *Ocean Engineering* 234.109216 (2021).

[96] X. Xu, P. Cai, Z. Ahmed, V. S. Yellapu, and W. Zhang. "Path planning and dynamic collision avoidance algorithm under COLREGs via deep reinforcement learning". In: *Neurocomputing* 468 (2022), pp. 181–197.

[97] P. Zhai, Y. Zhang, and W. Shaobo. "Intelligent ship collision avoidance algorithm based on DDQN with prioritized experience replay under COLREGs". In: *Journal of Marine Science and Engineering* 10.5 (2022).

[98] L. Zhao and M. I. Roh. "COLREGs-compliant multiship collision avoidance based on deep reinforcement learning". In: *Ocean Engineering* 191 (2019), pp. 106436–106450.

[99] S. Guo, X. Zhang, Y. Zheng, and Y. Du. "An autonomous path planning model for unmanned ships based on deep reinforcement learning". In: *Sensors* 20.2 (2020).

[100] T. A. Johansen, T. Perez, and A. Cristofaro. "Ship collision avoidance and COLREGs compliance using simulation-based control behavior selection with predictive hazard assessment". In: *IEEE Transactions on Intelligent Transportation Systems* 17.12 (2016), pp. 3407–3422.

[101] A. Tsolakis, D. Benders, O. De Groot, R. R. Negenborn, V. Reppa, and L. Ferranti. "COLREGs-aware trajectory optimization for autonomous surface vessels". In: *IFAC-PapersOnLine* 55.31 (2022), pp. 269–274.

[102] D. K. Kufoalor, E. Wilthil, I. B. Hagen, E. F. Brekke, and T. A. Johansen. "Autonomous COLREGs-compliant decision making using maritime radar tracking and model predictive control". In: *Proc. of the European Control Conf. (ECC)*. 2019, pp. 2536–2542.

[103] B. O. H. Eriksen, M. Breivik, E. F. Wilthil, A. L. Flåten, and E. F. Brekke. "The branching-course model predictive control algorithm for maritime collision avoidance". In: *Journal of Field Robotics* 36.7 (2019), pp. 1222–1249.

[104] J. Zhang, H. Zhang, J. Liu, D. Wu, and C. G. Soares. "A two-stage path planning algorithm based on rapid-exploring random tree for ships navigating in multi-obstacle water areas considering COLREGs". In: *Journal of Marine Science and Engineering* 10.10 (2022).

[105] P. Stankiewicz and M. Kobilarov. "A primitive-based approach to good seamanship path planning for autonomous surface vessels". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2021, pp. 7767–7773.

[106] T. T. Enevoldsen, C. Reinartz, and R. Galeazzi. "COLREGs-informed RRT* for collision avoidance of marine crafts". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2021, pp. 8083–8089.

[107] H. T. L. Chiang and L. Tapia. "COLREG-RRT: An RRT-based COLREGs-compliant motion planner for surface vehicle navigation". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2024–2031.

[108] A. Lazarowska. "A trajectory base method for ship's safe path planning". In: *Procedia Computer Science* 96 (2016), pp. 1022–1031.

[109] *COLREGs: Convention on the international regulations for preventing collisions at sea.* International Maritime Organization (IMO), 1972.

[110] B. Vanholme, D. Gruyer, B. Lusetti, S. Glaser, and S. Mammar. "Highly automated driving on highways based on legal safety". In: *IEEE Transactions on Intelligent Transportation Systems* 14.1 (2013), pp. 333–347.

[111] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. "Playing Atari with deep reinforcement learning". In: *arXiv:1312.5602* (2013).

[112] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. De Freitas. "Dueling network architectures for deep reinforcement learning". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. 2016, pp. 1995–2003.

[113] H. van Hasselt, A. Guez, and D. Silver. "Deep reinforcement learning with double q-learning". In: 2016, pp. 2094–2100.

[114] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. "Prioritized experience replay". In: *arXiv:1511.05952* (2016).

[115] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. "Trust region policy optimization". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. 2015, pp. 1889–1897.

[116] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in Neural Information Processing Systems (NIPS)* 12 (1999).

[117] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. "Asynchronous methods for deep reinforcement learning". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. 2016, pp. 1928–1937.

[118] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal policy optimization algorithms". In: *arXiv:1707.06347* (2017).

[119] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. "Continuous control with deep reinforcement learning". In: *arXiv:1509.02971* (2015).

[120] S. Fujimoto, H. Van Hoof, and D. Meger. "Addressing function approximation error in actor-critic methods". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. 2018, pp. 2587–2601.

[121] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *Proc. of the Int. Conf. on Machine Learning (ICML)*. 2018, pp. 1861–1870.

[122] E. M. Clarke and J. M. Wing. "Formal methods: state of the art and future directions". In: *ACM Computing Surveys* 28.4 (1996), pp. 626–643.

[123] L. Schäfer, F. Gruber, and M. Althoff. "Scalable computation of robust control invariant sets of nonlinear systems". In: *IEEE Transactions on Automatic Control* 69.2 (2024), pp. 755–770.

[124] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. "Control barrier functions: Theory and applications". In: *Proc. of the European Control Conf. (ECC)*. 2019, pp. 3420–3431.

[125] M. A. Ben Sassi and A. Girard. "Computation of polytopic invariants for polynomial dynamical systems using linear programming". In: *Automatica* 48.12 (2012), pp. 3114–3121.

[126] M. Korda, D. Henrion, and C. N. Jones. "Convex computation of the maximum controlled invariant set for polynomial control systems". In: *SIAM Journal on Control and Optimization* 52.5 (2014), pp. 2944–2969.

[127] S. Konur. "A survey on temporal logics for specifying and verifying real-time systems". In: *Frontiers of Computer Science* 7.3 (2013), pp. 370–403.

[128] E. Bartocci, C. Mateis, E. Nesterini, and D. Ničković. "Survey on mining signal temporal logic specifications". In: *Information and Computation* 289.104957 (2022).

[129] O. Maler and D. Ničković. "Monitoring temporal properties of continuous signals". In: *Proc. of the Workshop on Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems (FTRTFT)*. 2004, pp. 152–166.

[130] R. Alur and T. A. Henzinger. "Real-time logics: Complexity and expressiveness". In: *Information and Computation* 104.1 (1993), pp. 35–77.

[131] P. Thati and G. Rou. "Monitoring algorithms for metric temporal logic specifications". In: *Electronic Notes in Theoretical Computer Science* 113 (2005), pp. 145–162.

[132] M. Althoff. "Reachability analysis and its application to the safety assessment of autonomous cars". Dissertation. Technische Universität München, 2010.

[133] M. Wetzlinger, A. Kulmburg, A. Le Penven, and M. Althoff. "Adaptive reachability algorithms for nonlinear systems using abstraction error analysis". In: *Nonlinear Analysis: Hybrid Systems* 46.101252 (2022).

[134] M. Wetzlinger, N. Kochdumper, S. Bak, and M. Althoff. "Fully automated verification of linear systems using inner and outer approximations of reachable sets". In: *IEEE Transactions on Automatic Control* 68.12 (2023), pp. 7771–7786.

[135] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer. "A survey of algorithms for black-box safety validation of cyber-physical systems". In: *Journal of Artificial Intelligence Research* 72 (2021), pp. 377–428.

[136] B. Weng, G. A. Castillo, W. Zhang, and A. Hereid. "On safety testing, validation, and characterization with scenario-sampling: A case study of legged robots". In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2022, pp. 5179–5186.

[137] V. Murali, A. Trivedi, and M. Zamani. "A scenario approach for synthesizing k-inductive barrier certificates". In: *IEEE Control Systems Letters* 6 (2022), pp. 3247–3252.

[138] P. Akella, M. Ahmadi, and A. D. Ames. "A scenario approach to risk-aware safety-critical system verification". In: *arXiv:2203.02595* (2022).

[139] I. I. Delice and S. Ertugrul. "Intelligent modeling of human driver: A survey". In: *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*. 2007, pp. 648–651.

[140] F. Gruber and M. Althoff. "Anytime safety verification of autonomous vehicles". In: *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*. 2018, pp. 1708–1714.

[141] P. Karle, T. Betz, M. Bosk, F. Fent, N. Gehrke, M. Geisslinger, L. Gressenbuch, P. Hafemann, S. Huber, M. Hübner, S. Huch, G. Kaljavesi, T. Kerbl, D. Kulmer, T. Mascetta, S. Maierhofer, F. Pfab, F. Rezabek, E. Rivera, S. Sagmeister, L. Seidlitz, F. Sauerbeck, I. Tahiraj, R. Trauth, N. Uhlemann, G. Würsching, B. Zarrouki, M. Althoff, J. Betz, K. Bengler, G. Carle, F. Diermeyer, J. Ott, and M. Lienkamp. "EDGAR: An autonomous driving research platform – From feature development to real-world application". In: *arXiv:2309.15492* (2024).

[142] D. Greene, J. Liu, J. Reich, Y. Hirokawa, A. Shinagawa, H. Ito, and T. Mikami. "An efficient computational architecture for a collision early-warning system for vehicles, pedestrians, and bicyclists". In: *IEEE Transactions on Intelligent Transportation Systems* 12.4 (2011), pp. 942–953.

[143] E. Meyer, L. F. Peiss, and M. Althoff. "Deep occupancy-predictive representations for autonomous driving". In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. 2023, pp. 5610–5617.

[144] Z. D. Guo, B. A. Pires, B. Piot, J.-B. Grill, F. Altché, R. Munos, and M. G. Azar. "Bootstrap latent-predictive representations for multitask reinforcement learning". In: *Proc. of the. Int. Conf. on Machine Learning (ICML)*. 2020, pp. 3875–3886.

[145] S. Recanatesi, M. Farrell, G. Lajoie, S. Deneve, M. Rigotti, and E. Shea-Brown. "Predictive learning as a network mechanism for extracting low-dimensional latent space representations". In: *Nature Communications* 12.1417 (2021).

[146] E. Irani Liu and M. Althoff. "Specification-compliant driving corridors for motion planning of automated vehicles". In: *IEEE Transactions on Intelligent Vehicles* 8.9 (2023), pp. 4180–4197.

[147] B. K. Aichernig, M. Tappler, and F. Wallner. "Benchmarking combinations of learning and testing algorithms for automata learning". In: *Formal Aspects of Computing* 36.1 (2024).

A Provably Safe Reinforcement Learning for Motion Planning with Collision Avoidance

Guarantees for safety specifications are necessary to employ RL-based components in autonomous vehicles performing safety-critical tasks. Motion planning tasks are often safety-critical since collisions should never happen as they could seriously injure humans, harm the environment, or damage the autonomous vehicle. Therefore, RL-based components need to be verified so that safety guarantees are provided.

The publications reproduced in this chapter develop provably safe RL approaches, which provide hard safety guarantees, based on model-free RL algorithms. Safety verification of actions is conducted online and safe actions are identified with set-based reachability analysis. All publications regard variants of Safety Specification 2. In Appendix A.1 and Appendix A.4, the actions of the RL agents are restricted such that the RL agents never leave an invariably safe set. In Appendix A.2 and Appendix A.3, we ensure that the RL agents always avoid unsafe sets. The investigated motion planning tasks are mostly autonomous driving of cars. Additionally, we examine stabilization and path tracking tasks for autonomous aerial vehicles.

A.1 Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking

Summary RL is based on random exploration, which often contradicts safety. Yet, for autonomous vehicles performing safety-critical tasks, we have to ensure that safety specifications are met. To ensure safety for an RL agent, actions have to be verified before allowing execution. The field of provably safe RL develops methods that achieve hard guarantees for safety specifications during both training and deployment of RL agents. However, there is no consistent terminology for this field to date and most provably safe RL research lacks comparison to other provably safe RL methods.

This work structures, surveys, and benchmarks provably safe RL research. In particular, we identify three conceptual categories to ensure safety during training and deployment: action replacement, action projection, and action masking. Action replacement identifies unsafe actions proposed by the RL agent and replaces them with safe actions before they are executed. Action projection also identifies unsafe actions, but instead of replacing them with a safe action, these approaches project unsafe actions to the closest safe action. Action masking determines the set of safe actions and restricts the action selection of the RL agent to this set.

Our systematic literature review shows that most research on provably safe RL designs action replacement or action projection approaches. Additionally, different learning tuples are used to optimize the policy in the provably safe RL literature. A learning tuple describes the interaction of the RL agent and environment through the previous and current observation, the action used for transitioning between the observations, and the corresponding reward. We investigate the difference between learning tuples proposed in the literature for five RL algorithms on an inverted pendulum and a two-dimensional quadrotor benchmark. Comparing the three conceptual approaches shows that action replacement is the most robust and effective across RL algorithms and learning tuples on the two benchmarks.

Author contributions H.K. and X.W. initiated the project of comparing different provably safe RL approaches. M.M. implemented the comparison on the inverted pendulum benchmark and evaluated the experiments. H.K. and J.T. conducted the systematic literature review. H.K, J.T., and M.A. designed the categorization. H.K. and J.T. implemented, conducted, and evaluated the experiments on the two-dimensional quadrotor benchmark. L.S. provided feedback on the implementation and the robust control invariant sets for both benchmarks. H.K. and J.T. structured the presentation of the manuscript and wrote the manuscript. M.M., L.S., X.W., and M.A. provided feedback improving the manuscript.

TUM Graduate School This publication has been declared a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking

Hanna Krasowski* *hanna.krasowski@tum.de*
Jakob Thumm* *jakob.thumm@tum.de*
Marlon Müller *marlon.mueller@tum.de*
Lukas Schäfer *lukas.schaefer@tum.de*
Xiao Wang *xiao.wang@tum.de*
Matthias Althoff *althoff@tum.de*

School of Computation, Information and Technology
Technical University of Munich

Reviewed on OpenReview: *https://openreview.net/forum?id=mcNOezbnzO*

Abstract

Ensuring the safety of reinforcement learning (RL) algorithms is crucial to unlock their potential for many real-world tasks. However, vanilla RL and most safe RL approaches do not guarantee safety. In recent years, several methods have been proposed to provide hard safety guarantees for RL, which is essential for applications where unsafe actions could have disastrous consequences. Nevertheless, there is no comprehensive comparison of these provably safe RL methods. Therefore, we introduce a categorization of existing provably safe RL methods, present the conceptual foundations for both continuous and discrete action spaces, and empirically benchmark existing methods. We categorize the methods based on how they adapt the action: action replacement, action projection, and action masking. Our experiments on an inverted pendulum and a quadrotor stabilization task indicate that action replacement is the best-performing approach for these applications despite its comparatively simple realization. Furthermore, adding a reward penalty, every time the safety verification is engaged, improved training performance in our experiments. Finally, we provide practical guidance on selecting provably safe RL approaches depending on the safety specification, RL algorithm, and type of action space.

1 Introduction

Reinforcement learning (RL) contributes to many recent advancements in challenging research fields such as robotics (El-Shamouty et al., 2020; Zhao et al., 2020), autonomous systems (Kiran et al., 2022; Ye et al., 2021), and games (Mnih et al., 2013; Silver et al., 2017). A vanilla RL agent typically explores randomly and executes undesired actions multiple times to learn how to achieve the highest possible reward. However, safety is important for many applications. Therefore, safe RL emerged where the learning process is adapted such that the agent considers safety aspects next to performance during training and/or operation (García & Fernández, 2015). There are different degrees of how safety is considered for safe RL approaches. First, some approaches only incorporate safety aspects without formal guarantees. Here, the agent chooses safe actions with higher probability, e.g., by adding a reward component that indicates risk or adapting the exploration such that the agent follows a safe heuristic. Second, some approaches provide probabilistic safety guarantees, e.g., by using probabilistic models for safe actions (Könighofer et al., 2021). Still, hard safety guarantees for RL agents are necessary whenever failures are disastrous and need to be avoided at all costs during training and deployment. Such safety-critical applications include autonomous driving, human-robot collaboration,

*Equal contribution.

or energy grids. We refer to this third subcategory of safe RL methods providing hard safety guarantees for both training and operation as *provably safe RL*.

We provide for the first time a consistent conceptual framework for *provably safe RL* in both continuous and discrete action spaces, a comprehensive literature survey, and a comparison between provably safe RL approaches on two widely used control benchmarks. The characteristic difference between provably safe RL approaches is how they adapt the actions of the agent. Therefore, we propose classifying them into three categories: *action replacement*, *action projection*, and *action masking*. Our proposed categorization of provably safe RL provides a concise presentation of the research field, supports researchers implementing provably safe RL through clear terminology and a comprehensive literature review, and outlines ideas for future research within the three categories. Furthermore, we evaluate the methods experimentally. Our three main findings of our experiments were that all provably safe RL methods were indeed safe, that action replacement performed best on average over five tested RL algorithms, and that adding a penalty to the reward when using the safety function further improved performance.

Our contributions are fourfold. First, we introduce a comprehensive classification of provably safe RL methods and their formal description. This categorization allows us to compare and benchmark the effects of choosing a specific type of action modification on the ability of agents to learn. Second, we propose the first formulation of action masking for continuous action spaces. Third, we provide a structured and comprehensive survey of previous provably safe RL works and assign them to the three categories. Finally, we are the first to evaluate the performance of all three provably safe RL methods on two common control benchmarks. This comparison provides insights into the strengths and weaknesses of the different provably safe RL approaches and allows us to provide advice on selecting the best-suited provably safe RL approach for a specific problem independent of the safety verification method used.

The remainder of this paper is structured as follows. First, we briefly review the historical development of safe RL and provably safe RL in Section 1.1. We describe preliminary concepts in Section 2 and introduce our proposed categorization. Then, we show how the related provably safe RL literature fits our categorization in Section 3. Section 4 compares the different provably safe RL categories experimentally on a two-dimensional (2D) quadrotor stabilization task. Section 5 discusses the results of our experimental evaluation and the practical considerations following them. Finally, we conclude this work in Section 6.

1.1 Evolution towards provably safe RL

The notion of risk and safety in RL is discussed at least since the 1990s (Heger, 1994). The reasons for combining safety and RL were to focus the learning on relevant or safe regions and improve the convergence speed. Thus, the field of safe RL started developing, and in 2015, García & Fernández (2015) were the first to cluster safe RL. They provide two high-level categories: approaches that modify the optimization criterion and approaches that modify the exploration. Since 2015, significant advances in model-free RL and the increased applicability of deep RL changed the research focus of safe RL. Notably, the higher efficiency of model-free deep RL made its real-world application tangible and amplified the need for formal guarantees in safe RL. This is also apparent from the survey by Brunke et al. (2022), who investigated recent developments at the intersection of control and learning for safe robotics. As a goal of the broader field of safe learning for control, they identify methods with as little as possible system knowledge while ensuring formal safety guarantees (Brunke et al., 2022, Fig. 4). Among existing safe RL approaches, provably safe RL research is a growing field located at this frontier as it provides hard safety guarantees during both learning and deployment. While a few papers mentioned in Brunke et al. (2022) are part of provably safe RL, it is not a focus of their work. In the following paragraphs, we use the common classification by safety specification type, which can be *soft constraints*, *probabilistic guarantees*, and *hard guarantees*, to locate provably safe RL in the field of safe RL.

Soft constraints Approaches with soft constraints consider safety directly in their optimization objective so that the agent can explore all actions and states regardless of safety. Thus, these methods can be unsafe during training, especially initially, but usually converge to a safer policy without formal safety guarantees after sufficiently many training steps. The simplest way to inform an RL agent about safety constraints is through its reward function. Despite its elegance, the reward function approach has many potential pitfalls.

First, the reward function might be ill-defined, either from manual tuning or when learned from human input. When manually defined, the reward function might overlook certain features or fine details, leading to a hackable reward (Skalse et al., 2022) from which the agent learns an unsafe behavior. Learning the reward function from human feedback (Christiano et al., 2017) is also error-prone because communicating safety constraints alongside performance metrics is hard for sparse, nonlinear, conditional, or seldom occurring constraints. Second, even if the reward function is defined correctly, the trained policy is not guaranteed to be safe, e.g., it was shown by Packer et al. (2018) that RL agents struggle with out-of-distribution states during deployment. Third, the agent might learn to perform actions safely but ignore the task objective due to goal misgeneralization (Langosco et al., 2022). Still, the majority of safe RL research considers safety aspects as soft constraints, so we provide a short overview of soft constraint methods in the following paragraph.

To reduce the burden of manual reward specification, a recent line of work formalizes the task and its safety specifications as a temporal logic formula. Then, the temporal logic formula is transformed into the RL reward either by directly using the robustness measure associated with the formula as the reward (Aksaray et al., 2016; Li et al., 2017; Varnai & Dimarogonas, 2020) or by transforming the temporal logic formula into an automaton that generates the reward (Camacho et al., 2019; Hahn et al., 2019; Cai et al., 2021; Hasanbeig et al., 2022; Alur et al., 2023). For some algorithms, it can even be ensured that the policy converges to the optimal policy, which maximally satisfies the temporal logic specification (Alur et al., 2022; Yang et al., 2022). Another way to inform an RL agent about safety than through the reward is by formulating a constrained optimization problem. Many recent advances have been made in constrained RL (Altman, 1998; Achiam et al., 2017; Stooke et al., 2020), for which the policy aims to maximize the reward while satisfying user-defined specifications. The specifications can be formulated as constraint functions (Chow et al., 2018; Stooke et al., 2020; Yang et al., 2020; Marvi & Kiumarsi, 2021) or as temporal logic formulas (De Giacomo et al., 2021; Hasanbeig et al., 2019a;b; 2020). The main advantage of soft constraint methods over probabilistic or hard constraint methods is that no explicit model of the agent dynamics or the environment is required as the agent learns the safety aspects through experience. Thus, such safe RL methods have a high potential in non-critical settings, where unsafe actions do not cause major damage.

Probabilistic guarantees Probabilistically safe RL approaches rely on probabilistic models or synthesize a model from sampled data. Here, the action and state space can be restricted based on probabilities. Nonetheless, unsafe actions are sometimes not detected and might occur occasionally. Several works (Turchetta et al., 2016; Berkenkamp et al., 2017; Mannucci et al., 2018) try to determine the maximal set of safe states by starting from an often user-defined conservative set and extending it with the gathered learning experience. Other methods (Könighofer et al., 2021; Thananjeyan et al., 2021; Dalal et al., 2018; Zanon & Gros, 2021; Yang et al., 2021; Gillula & Tomlin, 2013) are based on formulating probabilistic models that identify the probability of safety for an action. In general, approaches that rely on probabilistic methods are especially applicable if one cannot bound measurement errors, modeling errors, and disturbances by sets.

Hard guarantees Provably safe RL features hard safety guarantees, which are fulfilled by integrating prior system knowledge into the learning process. Here, the agent only explores safe actions and only reaches states fulfilling the safety specifications. Provably safe RL already fulfills the given safety specifications during the learning process, which is essential when training or fine-tuning agents on safety-critical tasks in the physical world. Thus, we exclude approaches that only verify learned policies (Bastani et al., 2018; Schmidt et al., 2021) from our survey. We focus on model-free RL algorithms that do not explicitly learn or use a model of the system dynamics to optimize the policy. Generally, deploying learned controllers in the physical world became increasingly realistic in recent years, and thus, the need for provably safe RL grew, and more provably safe RL approaches were developed. With this work, we aim to structure and provide practical insights into this growing field.

2 Conceptual analysis

We introduce three provably safe RL classes by providing their formal description in one comprehensive conceptual framework. This framework clarifies the differences between the three classes and eases the following literature review and benchmarking.

Markov decision process The RL agent learns on a Markov decision process (MDP) that is described by the tuple $(\mathbb{S}, \mathbb{A}, T, r, \gamma)$. Hereby, we assume that the set of states $\mathbb{S}$ is fully observable with bounded precision. Partially observable MDPs can be handled using methods like particle filtering (Sunberg & Kochenderfer, 2018) and are not further discussed in this work. The action space $\mathbb{A}$ and state space $\mathbb{S}$ can be continuous or discrete. $T(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ is the transition function, which in the discrete case returns the probability that the transition from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ occurs by taking action $\boldsymbol{a}$. In the continuous case, $T(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ denotes the probability density function of the transition. We assume that the transition function is stationary over time. For each transition, the agent receives a reward $r : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ from the environment. The discount factor $0 < \gamma < 1$ weights the relevance of future rewards. The policy or value function that the action learns can be optimized for an infinite or finite episode horizon p.

Safety of a system For provably safe RL, it is required that the safety of states and actions is verifiable. Otherwise, no formal claims about the safety of a system can be made. Thus, we first introduce the set of safe states $\mathbb{S}_{\boldsymbol{s}} \subseteq \mathbb{S}$ containing all states for which all safety specifications are fulfilled[1]. For verifying the safety of actions, we use a safety function $\varphi : \mathbb{S} \times \mathbb{A} \to \{0, 1\}$

$$\varphi(\boldsymbol{s}, \boldsymbol{a}) = \begin{cases} 1, & \text{if } (\boldsymbol{s}, \boldsymbol{a}) \text{ is verified safe} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Conceptually, this is mainly done by over-approximating the set of states that are reachable by taking action $\boldsymbol{a}$ in state $\boldsymbol{s}$, and then validating if the reachable set of states is a subset of $\mathbb{S}_{\boldsymbol{s}}$ and if all these reachable states are verified safe until the episode horizon p. In other words, for each of these reachable states, there exists at least one action that keeps the system within $\mathbb{S}_{\boldsymbol{s}}$ until episode termination. To formalize this concept, we define the set of provably safe actions $\mathbb{A}_\varphi(\boldsymbol{s}) = \{\boldsymbol{a} | \varphi(\boldsymbol{s}, \boldsymbol{a}) = 1\}$ for a given state $\boldsymbol{s}$. The set of provably safe actions $\mathbb{A}_\varphi(\boldsymbol{s})$ is a subset of all safe actions[2] $\mathbb{A}_{\boldsymbol{s}}(\boldsymbol{s})$, i.e., $\mathbb{A}_\varphi(\boldsymbol{s}) \subseteq \mathbb{A}_{\boldsymbol{s}}(\boldsymbol{s}) \subseteq \mathbb{A}$. The safe action set $\mathbb{A}_{\boldsymbol{s}}(\boldsymbol{s})$ includes all safe actions, while the provably safe action set $\mathbb{A}_\varphi(\boldsymbol{s})$ only includes actions that are verified as safe by the safety function $\varphi(\boldsymbol{s}, \boldsymbol{a})$. In other words, the safety function possibly returns that an action is unsafe, which is indeed safe, while it never predicts truly unsafe actions to be safe. Moreover, we define the set of provably safe states based on the safety function: $\mathbb{S}_\varphi = \{\boldsymbol{s} | \exists \boldsymbol{a} \in \mathbb{A},\, \varphi(\boldsymbol{s}, \boldsymbol{a}) = 1\} \subseteq \mathbb{S}_{\boldsymbol{s}}$. Consequently, for a verified state-action tuple all reachable next states $\boldsymbol{s}'$ are in $\mathbb{S}_\varphi$. All provably safe RL approaches rely on the availability of provably safe actions and states to achieve a provably safe system:

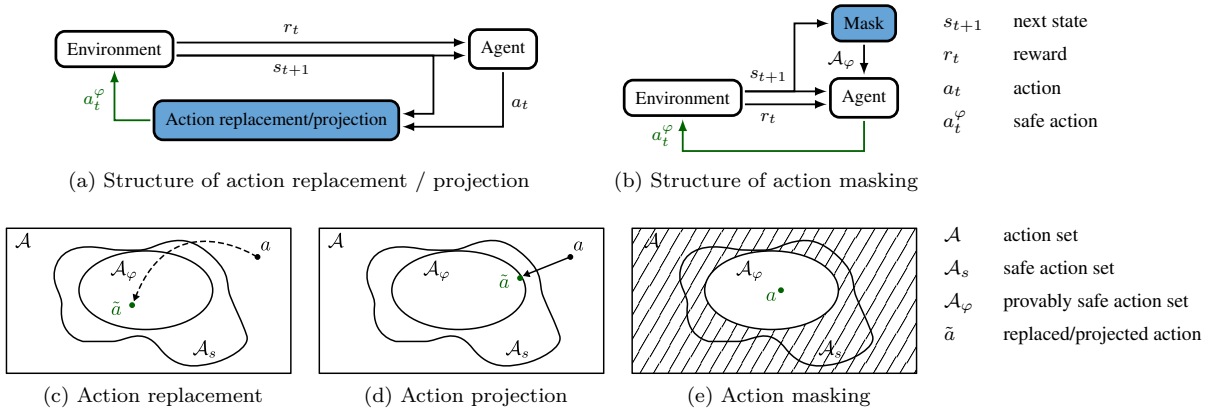
Proposition 1 *Let the system be initiated in a provably safe state $\boldsymbol{s}_0^\varphi \in \mathbb{S}_\varphi$. Then, there exists a sequence of provably safe actions that ensures $\boldsymbol{s} \in \mathbb{S}_{\boldsymbol{s}}$ at all times until the episode horizon p.*

The proof can be easily obtained from the definitions by induction as $\varphi(\boldsymbol{s}, \boldsymbol{a}) = 1$ if $\boldsymbol{s}' \in \mathbb{S}_{\boldsymbol{s}} \wedge \mathbb{A}_\varphi(\boldsymbol{s}') \neq \emptyset$. Note that if the episode horizon is finite, the last state of an episode is verified safe if it is contained in $\mathbb{S}_{\boldsymbol{s}}$. A provably safe action must not necessarily exist for this last state.

We define $\mathbb{S}_{\boldsymbol{s}}$ relatively broad since the safety specifications are usually task-specific and can take various forms such as stability, not entering a time-invariant or time-varying unsafe set, and temporal logic specifications. Depending on the safety specification and system under consideration, different verification methods are applicable and encoded in the safety function $\varphi(\boldsymbol{s}, \boldsymbol{a})$. We discuss the concrete verification methods used by previous works in Section 3. Although $\varphi(\boldsymbol{s}, \boldsymbol{a})$ only verifies safety for a given state $\boldsymbol{s}$ and action $\boldsymbol{a}$, it may take more than the next state into account. To make this more graspable, we shortly explain two concepts for

[1] The state space is often augmented from the state space for classical control purposes to a state space that includes other safety-relevant dimensions.

[2] Note that "taking no action" is commonly considered to be part of the action space, most often with the action $\boldsymbol{a} = [0, \ldots, 0]^\top$.



(a) Structure of action replacement / projection (b) Structure of action masking

(c) Action replacement (d) Action projection (e) Action masking

Figure 1: Structure of the three types of provably safe RL methods. The post-posed action replacement or projection methods (a) alter unsafe actions before sending them to the environment. In contrast, preemptive action masking approaches (b) allow the agent to only choose from the safe action space and, therefore, only output safe actions to the environment. Figures (c-e) highlight the differences between the three approaches in the action space. Here, action replacement (c) replaces unsafe actions with actions from the safe action space, action projection (d) projects unsafe actions to the closest safe action, and action masking (e) lets the agent choose solely from the safe action set.

calculating the provably safe states set $\mathbb{S}_\varphi$. For our benchmarks in Section 4, we compute a control invariant set as $\mathbb{S}_\varphi$. Thus, the online verification only consists of checking that all reachable states are contained in this control invariant set. Another concept is predicting the reachable states until a specified time horizon while starting from $\boldsymbol{s}$ and applying $\boldsymbol{a}$ for the first time step and an action sequence for the consecutive time steps. The verification checks that all reachable states are in $\mathbb{S}_{\boldsymbol{s}}$ and that either the reachable set at the prediction horizon is contained in a safe terminal set or the prediction horizon is the episode horizon. If the verification succeeds, we know that $\boldsymbol{s} \in \mathbb{S}_\varphi$.

Provably safe RL relies on model knowledge to provide safety guarantees, i.e., a conformant model that covers the safety-relevant system and environment dynamics. Hereby, the verification process can use an abstraction of the real system dynamics as long as it is conformant (Roehm et al., 2019; Liu et al., 2023) to the real system, i.e., it over-approximates both aleatoric and epistemic uncertainties and covers all relevant safety aspects. This eases efficient verification, as the complexity of the abstraction is usually significantly lower than the complexity of the underlying MDP. In systems where such a safety model is unavailable, provably safe RL is not applicable, and only non-provably safe approaches, as discussed in Section 1, can be used. In practice, the safety specifications are often weakened to legal or passive safety. Hereby, inevitable safety violations caused by other agents are not considered to be the fault of the agent and are, therefore, not considered unsafe. Examples of proving legal safety have been presented for autonomous driving (Pek et al., 2020) and robotics (Bouraine et al., 2012).

There are multiple ways to ensure provable safety for RL systems, which we summarize in three categories: action replacement, where the safety method replaces all unsafe actions from the agent with safe actions, action projection, which projects unsafe actions onto the safe action space, and action masking, where the agent can only choose actions from the safe action space. We choose this categorization, as it represents the three main approaches found in the literature to modify actions and thereby ensure safety for RL. Action replacement and action projection alter the action after the agent returns it. In contrast, action masking lets the agent exclusively choose from the safe action space. Figure 1 displays the basic concept of these methods. The following subsections describe the concept, mathematical formalization, and practical implications of each approach.

2.1 Action replacement

The first approach to ensure the safety of actions is to replace any unsafe action returned by the agent with a safe action before its execution. The first step of action replacement is to evaluate the safety of the suggested action $\boldsymbol{a} \in \mathbb{A}$ using $\varphi(\boldsymbol{s}, \boldsymbol{a})$. If the action sampled from the policy $\boldsymbol{\pi}(\mathbf{a}|\boldsymbol{s})$ is not verified as safe, it is replaced with a provably safe replacement action $\tilde{\boldsymbol{a}} = \boldsymbol{\psi}(\boldsymbol{s})$, where $\boldsymbol{\psi} : \mathbb{S} \to \mathbb{A}_\varphi$ is called replacement function. Following this procedure, it is guaranteed that only safe actions $\boldsymbol{a}^\varphi$ with

$$\boldsymbol{a}^\varphi = \begin{cases} \boldsymbol{a} \sim \boldsymbol{\pi}(\mathbf{a}|\boldsymbol{s}), & \text{if } \varphi(\boldsymbol{s}, \boldsymbol{a}) = 1 \\ \boldsymbol{\psi}(\boldsymbol{s}), & \text{otherwise} \end{cases} \tag{2}$$

are executed. We discuss how this action replacement alters the MDP in the Appendix and additionally refer the interested reader to Hunt et al. (2021).

There are two general replacement functions found in the literature, *sampling* and *failsafe*. In sampling, the replacement function $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$ uniformly samples a random action from $\mathbb{A}_\varphi(\boldsymbol{s})$. The other approach is to use a failsafe controller $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ as replacement action, which could also stem from human feedback. In time-critical and complex scenarios, where building $\mathbb{A}_\varphi(\boldsymbol{s})$ online becomes too time-consuming, $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ is the only feasible option.

2.2 Action projection

In contrast to action replacement, where the replacement action is not necessarily related to the action of the agent, action projection returns the closest provably safe action with respect to the original action and some distance function. For this, we define the optimization problem

$$\begin{aligned} \tilde{\boldsymbol{a}} = \arg\min_{\hat{\boldsymbol{a}}} \quad & \text{dist}\left(\boldsymbol{a}, \hat{\boldsymbol{a}}\right) \\ \text{subject to} \quad & \varphi(\boldsymbol{s}, \hat{\boldsymbol{a}}) = 1, \end{aligned} \tag{3}$$

where $\text{dist}(\cdot)$ describes an arbitrary distance function, e.g., a p-norm. Note, that it might not be possible to define such a distance function, especially in discrete action spaces. The constraints are often defined explicitly by n constraint functions $f_i(\tilde{\boldsymbol{a}}, \boldsymbol{s}) \leq 0, \forall\, i \in 1, \ldots, n$ that confine the next state to the set of provably safe states, i.e., $\boldsymbol{s}' \in \mathbb{S}_\varphi$. The optimization problem in (3) minimizes the alteration of the actions while satisfying the safety specification, which is usually expressed through constraints for the optimization problem. Following Proposition 1, the optimization problem in (3) must always be feasible.

The most prominent ways to formulate the safety constraints for action projection are based on control barrier functions (CBFs) or robust model predictive control (MPC). For the first method, the constraints are defined by CBFs (Wieland & Allgöwer, 2007) that translate state constraints to control input constraints. We formulate the CBFs according to Taylor et al. (2020) as it is an intuitive formulation for RL. Consider a nonlinear control-affine system

$$\dot{\boldsymbol{s}} = \boldsymbol{m}(\boldsymbol{s}) + \boldsymbol{b}(\boldsymbol{s})\, \tilde{\boldsymbol{a}}, \tag{4}$$

where $\boldsymbol{s} \in \mathbb{S} \subseteq \mathbb{R}^N$ is the continuous state with N dimensions, and $\tilde{\boldsymbol{a}} \in \mathbb{A} \subset \mathbb{R}^M$ is the continuous control input with M dimensions and $\boldsymbol{m}(\boldsymbol{s})$ and $\boldsymbol{b}(\boldsymbol{s})$ are locally Lipschitz continuous functions. Then, the function h is a CBF if there exists an extended class $\mathbb{K}$ function α such that (Wabersich et al., 2023, Eq. 10)

$$\nabla h(\boldsymbol{s})(\boldsymbol{m}(\boldsymbol{s}) + \boldsymbol{b}(\boldsymbol{s})\tilde{\boldsymbol{a}}) \geq \alpha(h(\boldsymbol{s})). \tag{5}$$

If the system dynamics are not exactly known, the nominal model can be extended with bounded disturbances $\boldsymbol{d}$ to model the unknown system dynamics:

$$\dot{\boldsymbol{s}} = \boldsymbol{m}(\boldsymbol{s}) + \boldsymbol{b}(\boldsymbol{s})\, \tilde{\boldsymbol{a}} + \boldsymbol{d}. \tag{6}$$

To reduce conservatism, disturbances can be modeled as state and input-depended $\boldsymbol{d}(\boldsymbol{s}, \tilde{\boldsymbol{a}})$, and the maximal occurred disturbance can be learned from data as presented in Taylor et al. (2021). The limitation to control-affine systems makes formulating the constrained optimization problem efficient, e.g., for a Euclidean norm

as the distance function, (3) results in a quadratic program. A downside of using CBFs is that $h(\boldsymbol{s})$ is not trivial to find, especially in environments with dynamic obstacles.

For the second common projection method, we formulate the optimization problem with MPC according to Wabersich & Zeilinger (2021). There, the constraint in (3) is satisfied if we find an action sequence that steers the system from the current state $\boldsymbol{s}$ into the safe terminal set $\mathbb{M}$ within a finite prediction horizon $L \in \mathbb{N}$ while respecting input and state constraints, which reflect the safety specification (Wabersich & Zeilinger, 2021, Eq. 5):

$$\begin{aligned} \tilde{\boldsymbol{a}} = \arg\min_{\hat{\boldsymbol{a}}} \quad & \operatorname{dist}(\boldsymbol{a}, \hat{\boldsymbol{a}}) \\ \text{subject to} \quad & \boldsymbol{s}_{l+1} = \boldsymbol{g}(\boldsymbol{s}_l, \boldsymbol{a}_l), \boldsymbol{s}_0 = \boldsymbol{s}, \\ & \forall\, l \in \{1, ..., L-1\} \,:\, \boldsymbol{s}_l \in \mathbb{S}_{\boldsymbol{s}}, \\ & \boldsymbol{s}_L \in \mathbb{M}, \\ & \forall\, l \in \{0, ..., L-1\} \,:\, \boldsymbol{a}_l \in \mathbb{A}, \\ & \hat{\boldsymbol{a}} = \boldsymbol{a}_0, \end{aligned} \tag{7}$$

where $\boldsymbol{s}_l$ and $\boldsymbol{a}_l$ are the predicted state and action l steps ahead of the current time step.[3] The function $\boldsymbol{g}(\cdot, \cdot)$ is obtained by time discretizing a smooth continuous-time nonlinear system, whose dynamics are governed by $\dot{\boldsymbol{s}} = f(\boldsymbol{s}, \boldsymbol{a})$. The safe terminal set $\mathbb{M} \subseteq \mathbb{S}$ is control invariant, i.e., after the agent has entered $\mathbb{M}$, the associated invariance-enforcing controller keeps the agent inside this set indefinitely. If the optimization problem is solvable, $\tilde{\boldsymbol{a}}$ is executed. If it is not solvable, the control sequence from the previous state is used as a backup plan until the safe terminal set is reached or the optimization problem is solvable again (Schürmann et al., 2018). For perturbed systems of the form $\dot{\boldsymbol{s}} = f(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{d})$ with a bounded disturbance $\boldsymbol{d}$, robust MPC schemes, e.g., Schürmann et al. (2018), have to be employed and output-feedback MPC schemes, e.g., Gruber & Althoff (2021), account for measurement uncertainties. Similar to the CBF approach, conservatism can be reduced by learning the disturbance bounds from data (Hewing et al., 2020). Note that, for an environment with dynamic obstacles, the safe terminal set can be time-dependent, and we are unaware of a straightforward integration where Proposition 1 still holds.

2.3 Action masking

The two previous approaches modify unsafe actions from the agent. In action masking, we do not allow the agent to output an unsafe action in the first place (preemptive method). Hereby, a mask is added to the agent so that it can only choose from actions in the provably safe action set. In addition to Proposition 1, action masking in practice requires an efficient function $\boldsymbol{\eta}(\boldsymbol{s}) : \mathbb{S} \to \mathcal{P}(\mathbb{A})$, where $\mathcal{P}$ denotes the powerset, that determines a sufficiently large set of provably safe actions $\mathbb{A}_\varphi \subseteq \mathbb{A}$ for a given state $\boldsymbol{s}$. The policy function $\boldsymbol{\pi}$ is informed by the function $\boldsymbol{\eta}(\boldsymbol{s})$ and the action selection is adapted such that only actions from $\mathbb{A}_\varphi$ can be selected:

$$\boldsymbol{a} \sim \boldsymbol{\pi}(\mathbf{a}|\boldsymbol{\eta}(\boldsymbol{s}), \boldsymbol{s}) \in \mathbb{A}_\varphi. \tag{8}$$

If $\boldsymbol{\eta}(\boldsymbol{s})$ can only verify one or a few actions efficiently, the agent cannot learn properly because the agent cannot explore different actions and find the optimal one among them. Ideally, the function $\boldsymbol{\eta}(\boldsymbol{s})$ achieves $\mathbb{A}_\varphi = \mathbb{A}_s$.

The action masking approaches for discrete and continuous action spaces are not easily transferable into each other, and will therefore be discussed separately in this subsection. For discrete actions, the safety of each action is typically verified in each state using $\varphi(\boldsymbol{s}, \boldsymbol{a})$ and all verified safe actions are added to $\mathbb{A}_\varphi(\boldsymbol{s})$, i.e., $\boldsymbol{\eta}$ iterates over all actions for the current state $\boldsymbol{s}$ with $\varphi(\boldsymbol{s}, \boldsymbol{a})$ to identify $\mathbb{A}_\varphi(\boldsymbol{s})$. Intuitively, the discrete action mask is an informed drop-out layer added at the end of the policy network. We define the resulting safe policy $\boldsymbol{\pi}_m(\boldsymbol{a}|\boldsymbol{s})$ based on Huang & Ontañón (2022, Eq. 1) as

$$\boldsymbol{\pi}_m(\boldsymbol{a}|\boldsymbol{s}) = \varphi(\boldsymbol{s}, \boldsymbol{a}) \frac{\boldsymbol{\pi}(\mathbf{a}|\boldsymbol{s})}{\sum_{\boldsymbol{a}^\varphi \in \mathbb{A}_\varphi(\boldsymbol{s})} \boldsymbol{\pi}(\boldsymbol{a}^\varphi|\boldsymbol{s})}. \tag{9}$$

[3] We omitted in (7) that $\boldsymbol{s}_1, ..., \boldsymbol{s}_L, \boldsymbol{a}_1, ..., \boldsymbol{a}_{L-1}$ are decision variables of the optimization problem to improve the readability.

The integration of masking in a specific learning algorithm is not trivial. The effects on policy optimization methods are discussed in Krasowski et al. (2020); Huang & Ontañón (2022). For RL algorithms that learn the Q-function, we exemplary discuss the effects of discrete action masking for deep Q-network (DQN) (Mnih et al., 2013), which is most commonly used for Q-learning with discrete actions. During exploration with action masking, the agent samples its actions uniformly from $\mathbb{A}_\varphi$. When the agent exploits the Q-function, it chooses only the best action among $\mathbb{A}_\varphi$, i.e., $\arg\max_{\boldsymbol{a}\in\mathbb{A}_\varphi} Q(\boldsymbol{s},\boldsymbol{a})$. The temporal difference error for updating the Q-function $Q(\boldsymbol{s},\boldsymbol{a})$ is (Mnih et al., 2013, Eq. 3)

$$r(\boldsymbol{s},\boldsymbol{a}) + \gamma \max_{\boldsymbol{a}'} Q(\boldsymbol{s}',\boldsymbol{a}') - Q(\boldsymbol{s},\boldsymbol{a}), \tag{10}$$

where the action in the next state is $\boldsymbol{a}' \in \mathbb{A}_\varphi$ in contrast to the vanilla temporal difference error where the maximum Q-value for the next state is searched among actions from $\mathbb{A}$. Using the adapted temporal difference error in (10), the learning updates are performed only with Q-values of actions relevant in the next state instead of the full action space.

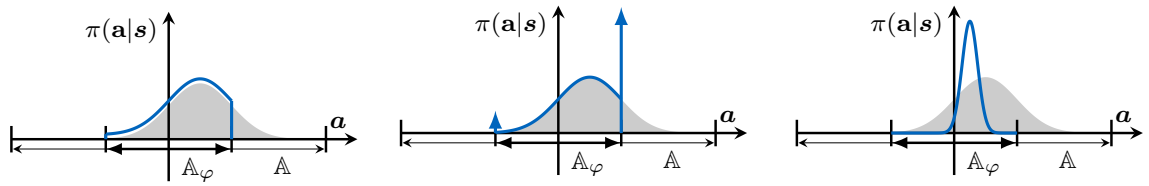
To comprehensively compare the different provably safe RL approaches on discrete and continuous action spaces in Section 4, we propose a simple formulation for continuous masking since there is no existing approach. We formulate this form of continuous action masking as a transformation of the action of agents to the provable safe action set. Our approach requires both $\mathbb{A}$ and $\mathbb{A}_\varphi$ to be axis-aligned boxes with the same center. We propose to transform the action space $\mathbb{A}$ into $\mathbb{A}_\varphi$ by applying the transformation

$$\tilde{\boldsymbol{a}} = (\boldsymbol{a} - \min(\mathbb{A})) \frac{\max(\mathbb{A}_\varphi) - \min(\mathbb{A}_\varphi)}{\max(\mathbb{A}) - \min(\mathbb{A})} + \min(\mathbb{A}_\varphi) \tag{11}$$

to the actions $\boldsymbol{a} \in \mathbb{A}$, where $\min(\cdot)$ and $\max(\cdot)$ return a vector containing the minimal and maximal value of the given set in each dimension respectively, and all operations are evaluated element-wise. For example, given a two-dimensional continuous action space $\mathbb{A} = [0,1] \times [-1,2]$, then $\min(\mathbb{A}) = [0,-1]^\top$. Note that the representation of $\mathbb{A}_\varphi$ as an axis-aligned box centered in $\mathbb{A}$ can be under-approximative and, thus, lead to conservative behavior. To overcome this limitation, more complex set representations, such as the zonotopes (Althoff et al., 2021), for the action spaces ($\mathbb{A}$ and $\mathbb{A}_\varphi$) in combination with solving an optimization problem that maximizes the size of $\mathbb{A}_\varphi$ could be investigated. A less sophisticated yet possibly effective approach is searching for a good latent interval representation of and transformation to $\mathbb{A}_\varphi$ by applying principal component analysis to a set of $\mathbb{A}_\varphi$ for different states as a pre-computing step. Since the action spaces for RL are defined a priori, there must always be a valid transformation from $\boldsymbol{a}$ to $\tilde{\boldsymbol{a}} \in \mathbb{A}_\varphi$ and such that the operation is time-invariant for all state-action pairs. In the next section, we discuss the effect of the three provably safe RL approaches on the policy distribution and exploration.

2.4 Impact on the distribution of actions

The three previously presented provably safe RL methods have different effects on the resulting distribution of actions, as illustrated for a one-dimensional continuous action space and a probabilistic policy in Figure 2. For action projection, all actions that are not verified safe $\boldsymbol{a} \notin \mathbb{A}_\varphi$ are projected to the boundary of the provably safe action set $\partial\mathbb{A}_\varphi$. Therefore, actions on $\partial\mathbb{A}_\varphi$ are disproportionately explored compared to the interior of $\mathbb{A}_\varphi$. A similar effect can occur in action replacement depending on the replacement strategy, e.g., with $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$, the failsafe action is explored more often. However, if the random sampling strategy $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$ is used, as shown in Figure 2 (a), the likelihood of all safe actions being explored increases equally. The sampling strategy, therefore, fosters exploration and discourages exploitation, as all non-provably safe actions lead to uniformly distributed exploration in the safe action space. The distribution of actions for both action replacement and projection differs from the distribution of the current policy, which might be problematic for on-policy algorithms. In action masking, we only map the exploration from $\mathbb{A}$ to $\mathbb{A}_\varphi$. Thus, the exploration strategy is not affected by action masking. In this aspect, our approach in (11) is conceptually similar to action normalization, which is commonly used in RL (Sutton & Barto, 2018, Ch. 16.8).



(a) Action replacement (b) Action projection (c) Action masking

Figure 2: Idealized impact of the provably safe RL methods on the probability density function of the RL policy for a single action in a given state. The probability density function of the original RL policy $\pi(\mathbf{a}|\boldsymbol{s})$ and the provably safe policy $\pi(\tilde{\mathbf{a}}|\boldsymbol{s})$ are depicted as the gray area and the blue line respectively. Figure (a) shows action replacement with the random sampling strategy. Figure (b) displays action projection, where the vertical arrows are scaled Dirac delta distributions that stem from the fact that the unsafe parts of the original policy distribution are projected to the boundary of $\mathbb{A}_\varphi$. Figure (c) depicts our proposed implementation of continuous action masking.

2.5 Learning tuples

When changing the RL action, the training of the agent can be conducted with four possible learning tuples:

- *naive* - learning based on the action $\boldsymbol{a}$ returned by the policy network of the agent and the reward $r(\boldsymbol{s}, \boldsymbol{a}^\varphi)$ corresponding to the executed action $\boldsymbol{a}^\varphi$, which we denote by the tuple $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}', r(\boldsymbol{s}, \boldsymbol{a}^\varphi))$. This ensures that the agent is updated according to its current policy. Learning with the original action $\boldsymbol{a}$ should benefit on-policy learning, where the policy is updated based on experience collected using the most recent policy.

- *adaption penalty* - is *naive* with a penalty $r^*(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{a}^\varphi) = r(\boldsymbol{s}, \boldsymbol{a}^\varphi) + r_{\text{penalty}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{a}^\varphi)$ if an unsafe action was selected, which we denote by the tuple $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}', r^*(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{a}^\varphi))$. In action projection, the penalty r_{penalty} can include a term that is proportional to the projection distance $\text{dist}(\boldsymbol{a}, \boldsymbol{a}^\varphi)$, as proposed by Wang (2022).

- *safe action* - learning based on the safe and possibly adapted action and the corresponding reward, denoted by the tuple $(\boldsymbol{s}, \boldsymbol{a}^\varphi, \boldsymbol{s}', r(\boldsymbol{s}, \boldsymbol{a}^\varphi))$. By using the *safe action* tuple, we are correctly rewarding the agent for the actually performed transition. However, this requires updating the agent with an action that did not stem from its current policy $\boldsymbol{\pi}(\mathbf{a}|\boldsymbol{s})$, which is an expected behavior in off-policy but not on-policy learning. So, the safe action tuple might be a better fit for off-policy than for on-policy RL.

- *both* - in case the RL agent proposes an unsafe action, both the *adaption penalty* and the *safe action* tuples are used for learning.

In all cases, the next state $\boldsymbol{s}'$ and reward r are the true state and reward received from the environment after executing the safe action $\boldsymbol{a}^\varphi$.

Action masking is always paired with the *naive* or *adaption penalty* learning tuple in the literature. The *adaption penalty* is related to the reduction of the action space due to masking or a safety component in the reward function, and not a sparse reward as for action projection and action replacement. For discrete action masking, the naive and the *safe action* tuples are equivalent since the agent is only allowed to choose provably safe actions (see Figure 1). For continuous action masking, using the *safe action* tuple with the transformed action $\tilde{\boldsymbol{a}}$ leads to inconsistencies in learning because every action is transformed if $\mathbb{A}_\varphi \neq \mathbb{A}$.

3 Literature review

In this section, we summarize previous works in provably safe RL and assign them to the proposed categories. To identify the related literature, we used the search string `TITLE-ABS("reinforcement learning") AND TITLE(learning) AND [TITLE(safe*) OR TITLE(verif*) OR TITLE(formal*) OR TITLE(shield*)] AND LIMIT-TO(LANGUAGE,"English")`[4] for the Scopus[5] and IEEEXplore[6] search engine, which led to 620 papers already removing duplicates. Then, we screened papers by title and abstracts to identify 160 seemingly relevant papers. After close inspection, we identified 47 of these 160 papers as provably safe RL works. We give a condensed overview of all application-independent provably safe RL works in Table 1, and cluster all 47 provably safe RL works in Table 2 by their application. Some approaches in Table 1 are presented for unbounded disturbance. In such a setting, hard safety guarantees are generally not achievable. Still, we include approaches that would be provably safe with the assumption that the disturbance is bounded.

Action replacement One of the earliest provably safe RL works is Alshiekh et al. (2018), which constructed a so-called safety shield from linear temporal logic formulas. For that, they construct the verification function $\varphi(\boldsymbol{s}, \boldsymbol{a})$ by converting the linear temporal logic formulas into an automaton, on which they perform model checking. The advantage of online model checking is that linear temporal logic constraints can be guaranteed for general nonlinear systems, and the online complexity is linear in the number of discrete states. However, the method is only applicable to small discrete state spaces, as constructing the automaton offline has exponential complexity in the number of discrete states, and the online checking complexity also increases exponentially with the formula length (Baier & Katoen, 2008). In Alshiekh et al. (2018), the agent outputs n ranked actions, which are all checked for safety using $\varphi(\boldsymbol{s}, \boldsymbol{a})$. The shield executes the highest-ranked safe action or replaces the action with $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$ if none of the n actions is safe. They update the agent with the *safe action* learning tuple but also propose that the *both* learning tuple can be used to obtain additional training information. Similarly to Alshiekh et al. (2018), Könighofer et al. (2020) show that both probabilistic and deterministic shields increase the sample efficiency and performance for both action replacement and masking methods.

Akametalu et al. (2014) propose action replacement based on Hamilton-Jacobi-Isaacs reachability analysis, which was later extended by Fisac et al. (2019) to a general safe RL framework. They determine $\mathbb{S}_\varphi$ using Hamilton-Jacobi-Isaacs reachability analysis given bounded system disturbances. Safety is guaranteed by replacing any learned action on the border of $\mathbb{S}_\varphi$ with $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ stemming from an Hamilton-Jacobi-Isaacs optimal controller to guide the system back inside the safe set. Hamilton-Jacobi-Isaacs reachability analysis can verify reach-avoid problems with arbitrary non-convex sets (Wabersich et al., 2023) and disturbances that stem from a compact set. However, constructing the safe set scales exponentially in complexity with the number of state dimensions, which makes Hamilton-Jacobi-Isaacs reachability analysis infeasible for systems with more than four state dimensions (Chen & Tomlin, 2018). Fisac et al. (2019) argue that replacing the unsafe action with the action that maximizes the distance to the unsafe set increases performance in uncertain real-world environments compared to action projection. However, Hamilton-Jacobi-Isaacs approaches suffer from the curse of dimensionality and are, therefore, only feasible for systems with specific characteristics (Herbert et al., 2021).

Shao et al. (2021) use a trajectory safeguard based on set-based reachability analysis for $\varphi(\boldsymbol{s}, \boldsymbol{a})$. Set-based reachability analysis is applicable to reach-avoid problems for general nonlinear systems with uncertainties in the initial state, system dynamics, and input disturbances as long as they stem from a compact set, see, e.g., Althoff et al. (2021). Set-based reachability analysis has polynomial complexity in the state dimension for most set representations, as discussed by Althoff et al. (2021). However, compared to Hamilton-Jacobi-Isaacs reachability analysis, set-based reachability analysis cannot handle arbitrary non-convex sets but depends on specific set representations. Shao et al. (2021) sample n new actions randomly in the vicinity of the action if the action is unsafe. They then execute the closest safe action to the original (unsafe) action. If none of the n new actions is safe, a failsafe action $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ is executed. Shao et al. (2021) train their agent

[4]Documentation at http://schema.elsevier.com/dtds/document/bkapi/search/SCOPUSSearchTips.htm
[5]scopus.com
[6]ieeexplore.ieee.org

Table 1: Comparison of application-independent provably safe RL approaches.

Reference	**Verification Method**	**Space**		**Learning Tuple**	**Environment**[1]
		State	Action		
Action replacement					
Akametalu et al. (2014)	HJI[2] reachability analysis	cont.	cont.	special RL alg.	1D quadrotor [stoch.], cart-pole [stoch.]
Fisac et al. (2019)	HJI reachability analysis	cont.	cont.	N/A	1D quadrotor [stoch.]
Alshiekh et al. (2018)	model checking of automaton constructed from LTL[3]	disc.	disc.	*safe action*	Grid world [stoch.]
Könighofer et al. (2020)	model checking of automaton constructed from LTL	disc.	disc.	*adaption penalty*	ACC[4] [stoch.]
Anderson et al. (2020)	robust control invariant set	cont.	cont.	N/A	pendulum [det.], reach-avoid [det.], others [det.]
Hunt et al. (2021)	theorem proving of $\mathtt{d}\mathcal{L}$[5] formulas	disc.	disc.	*naive*	Grid world [stoch.]
Bastani (2021)	MPC[6]	cont.	cont.	deployment only	bicycle [det.], cart-pole [det.]
Shao et al. (2021)	set-based reachability analysis	cont.	cont.	*naive*	3D quadrotor [det.], highway driving [det.]
Selim et al. (2022b)	set-based reachability analysis	cont.	cont.	*adaption penalty*	3D quadrotor [stoch.], mobile robot [stoch.]
Action projection					
Pham et al. (2018)	verification of affine constraints for actions	cont.	cont.	*adaption penalty*	manipulator [det.]
Cheng et al. (2019)	CBF[7]	cont.	cont.	*safe action*	ACC [stoch.], pendulum [stoch.]
Li et al. (2019a)	CBF synthesized from LTL	cont.	cont.	*adaption penalty*	manipulator [det.]
Gros et al. (2020)	MPC	cont.	cont.	*naive*	2D LTI[8] system [stoch.]
Wabersich & Zeilinger (2021)	MPC	cont.	cont.	*naive*	3D quadrotor [stoch.], pendulum [stoch.]
Marvi & Kiumarsi (2022)	CBF	cont.	cont.	*adaption penalty*	2D LTI system [det.]
Selim et al. (2022a)	set-based reachability analysis	cont.	cont.	*naive*	3D quadrotor [stoch.], mobile robot [stoch.]
Kochdumper et al. (2023)	set-based reachability analysis	cont.	cont.	*adaption penalty*	3D quadrotor [stoch.]
Action masking					
Fulton & Platzer (2018)	theorem proving of $\mathtt{d}\mathcal{L}$ formulas	cont.	disc.	*naive*	ACC [det.]
Fulton & Platzer (2019)	theorem proving of $\mathtt{d}\mathcal{L}$ formulas	cont.	disc.	*naive*	ACC [stoch.]
Huang & Ontañón (2022)	verification of affine equality constraints for actions	disc.	disc.	*naive*	Grid world [N/A]
This study	set-based reachability analysis	cont.	cont.	*naive*	pendulum [det.], 2D quadrotor [stoch.]

Abbreviations: [1]stoch.: stochastic environment model, det.: deterministic environment model, [2]Hamilton-Jacobi-Isaacs (HJI), [3]linear temporal logic (LTL), [4]adaptive cruise control (ACC), [5]differential dynamic logic ($\mathtt{d}\mathcal{L}$), [6]model predictive control (MPC), [7]control barrier function (CBF), [8]linear time-invariant (LTI).

on the *naive* learning tuple. Selim et al. (2022b) also verify the safety of actions with set-based reachability analysis. They propose an informed replacement for $\boldsymbol{\psi}(\boldsymbol{s})$ such that the reachable set of the controlled system is pushed away from the unsafe set $\mathbb{S} \setminus \mathbb{S}_{\boldsymbol{s}}$. The authors further propose a method to account for unknown system dynamics using a so-called black-box reachability analysis. They use the *adaption penalty* learning tuple and showcase that their approach achieves provable safety in three use cases.

Hunt et al. (2021) build the verification function $\varphi(\boldsymbol{s}, \boldsymbol{a})$ using theorem proving of differential dynamic logic formulas. Using $\varphi(\boldsymbol{s}, \boldsymbol{a})$, they determine $\mathbb{A}_{\varphi}(\boldsymbol{s})$ for discrete action spaces and use $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ for replacement. They further show how provably safe end-to-end learning can be accomplished using controller and model monitors. They train the RL agent using the *naive* learning tuple on a drone example. The work of Anderson et al. (2020) proposes to define $\mathbb{S}_{\varphi}$ as a robust control invariant set and construct the safety function $\varphi(\boldsymbol{s}, \boldsymbol{a})$ based on a worst-case linear model of the system dynamics. A further notable work is Bastani (2021), which proposes a model predictive shield alongside the trained policy, which uses $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ for action replacement.

Table 2: Overview of applications in provably safe RL.

	Action Replacement	**Action Projection**	**Action Masking**
Aerial Vehicles	‡‡Akametalu et al. (2014); Shyamsundar et al. (2017); ‡‡Fisac et al. (2019); Anderson et al. (2020); †Harris & Schaub (2020); †Shao et al. (2021); †Selim et al. (2022b); †Nazmy et al. (2022)	†Wabersich & Zeilinger (2021); †Selim et al. (2022a); †Kochdumper et al. (2023)	N/A
Autonomous Driving	†Chen et al. (2020); Könighofer et al. (2020); †Shao et al. (2021); †Chen et al. (2022a); †Lee & Kwon (2022); ‡‡Wang et al. (2023); †Evans et al. (2023)	Cheng et al. (2019); †Wang (2022); †Hailemichael et al. (2022b); †Hailemichael et al. (2022a); ‡‡Kochdumper et al. (2023)	Fulton & Platzer (2018); †Mirchevska et al. (2018); Fulton & Platzer (2019); †Krasowski et al. (2020); Brosowsky et al. (2021); †Krasowski et al. (2022)
Power Systems	†Ceusters et al. (2023)	†Eichelbeck et al. (2022); †Chen et al. (2022b); †Zhang et al. (2023); †Yu et al. (2023)	†Tabas & Zhang (2022)
Robotic Manipulation	†Thumm & Althoff (2022)	‡‡Pham et al. (2018); ‡‡Li et al. (2019a)	N/A
Control Benchmarks	Akametalu et al. (2014); Anderson et al. (2020); Bastani (2021); Shao et al. (2021)	Cheng et al. (2019); Gros et al. (2020); Wabersich & Zeilinger (2021); Marvi & Kiumarsi (2022)	N/A
Grid World Games	Alshiekh et al. (2018); Hunt et al. (2021)	N/A	Huang & Ontañón (2022)
Miscellaneous	*Active suspension*: Li et al. (2019b); *Computing networks*: †Wang et al. (2022); *Mobile robot*: †Selim et al. (2022b)	*Mobile robot*: †Selim et al. (2022a); *Engine emission*: †Norouzi et al. (2023)	*Computing networks*: Seetanadi et al. (2020); *Traffic signal*: †Müller & Sabatelli (2022)

Note: Studies using high-fidelity simulators are marked with †, and ‡‡ indicates physical experiments. Additionally, papers occur multiple times in case they demonstrate their approach for different applications.

The most popular method by publications is action replacement. This is also visible from the large variety of application-specific approaches, e.g., for aerial vehicles (Shyamsundar et al., 2017; Harris & Schaub, 2020; Nazmy et al., 2022), autonomous driving (Chen et al., 2020; 2022a; Lee & Kwon, 2022; Wang et al., 2023; Evans et al., 2023), power systems (Ceusters et al., 2023), robotic manipulation (Thumm & Althoff, 2022), active suspension systems (Li et al., 2019b), and traffic engineering in computing networks (Wang et al., 2022). In particular, Ceusters et al. (2023) compare fail-safe action replacement and sampling-based action replacement. They observe that both methods have higher initial performance than the unsafe RL baseline, and fail-safe action replacement leads to better performance than the sampling-based version.

Action projection Research on action projection is usually conducted on continuous action and state spaces. The main differentiating factor between studies in this category is the specification of the optimization problem for the projection. To begin with, the work of Pham et al. (2018) guarantees safety using a differentiable constrained optimization layer called OptLayer. Their approach is restricted to quadratic programming problems, so the system model and constraints have to be linear. Despite these limitations, they show the effectiveness of their approach on a collision avoidance task with a simple robotic manipulator.

Cheng et al. (2019) specify the safety constraint $\varphi(\boldsymbol{s}, \boldsymbol{a}) = 1$ of the optimization problem via CBFs. Thus, the optimization problem in (3) becomes a quadratic program. Theoretically, the CBF approach is applicable to general control-affine systems with disturbances from a compact set for reach-avoid specifications. However, finding a CBF is not trivial, and synthesizing them can be exponential in the system dimension (Ames et al., 2019). To solve (3) more efficiently online, Cheng et al. (2019) add a neural network to the approach

in Section 2.2, which approximates the correction due to the CBF. The action is then shifted by the approximated value prior to optimization. This shift improves the implementation efficiency while still guaranteeing safety, as the action is often already safe after the shift, and no optimization problem needs to be solved. Their safe learning with CBFs shows faster convergence speed than vanilla RL when learning on a pendulum and a car following task. Li et al. (2019a) propose a method to construct a continuous CBF from an automaton, which is defined by linear temporal logic formulas. They further construct a guiding reward from the given automaton to improve the learning performance. The proposed approach is capable of learning a high-dimensional cooperative manipulation task safely. The authors of Marvi & Kiumarsi (2022) define a different problem, where the system model is assumed to be deterministic but unknown. They learn an optimal controller and the system dynamics iteratively while decreasing the conservativeness of their CBF in each iteration. The approach is provably safe for linear time-invariant (LTI) systems without disturbances.

Gros et al. (2020) and Wabersich & Zeilinger (2021) implement the optimization problem as a robust MPC problem as defined in (7). MPC is applicable to reach-avoid problems with measurement and state disturbances. However, when controlling high-speed systems, robust MPC is often limited to linear systems (Zeilinger et al., 2014). Gros et al. (2020) mainly discuss how the learning update has to be adapted if action projection is used for different RL algorithms. For Q-learning, they find that no adaption of the learning algorithm is necessary when the *naive* tuple is used. For policy gradient methods, they argue that the projection must also be included in the gradient for stable learning (Gros et al., 2020, Sec. 3). One downside of the robust MPC formulation of Gros et al. (2020) and Wabersich & Zeilinger (2021) is that dynamic constraints originating from moving obstacles or persons in the environment are not trivial to integrate. They approximate the dynamics of the system with a Gaussian process (GP) so that hard safety guarantees are impossible to prove. However, they could guarantee hard safety specifications if they would assume deterministic system dynamics with bounded disturbance as the aforementioned approaches do. Gros et al. (2020) evaluate their approach on a simple 2D LTI system, and Wabersich & Zeilinger (2021) show the efficacy of their approach on a pendulum and a quadrotor task.

Contrary to their previous work in Selim et al. (2022b), Selim et al. (2022a) propose to solve an optimization problem to find the closest safe action instead of using an informed replacement. They again use set-based reachability analysis to construct $\varphi(\boldsymbol{s}, \boldsymbol{a})$. They test their approach on a quadrotor and mobile robot benchmark. Kochdumper et al. (2023) utilize set-based reachability analysis to verify actions in $\varphi(\boldsymbol{s}, \boldsymbol{a})$. They formulate the projection for a parameterization of the action space and arrive at a mixed-integer quadratic problem with polynomial constraints. Their approach achieves provable safety for nonlinear systems with bounded disturbances, and they demonstrate their approach on two quadrotor tasks, autonomous driving on highways, and a physical F1TENTH car.

Next to the conceptual approaches, action projection algorithms are also specifically proposed for many cyber-physical systems, such as autonomous driving (Wang, 2022; Hailemichael et al., 2022a;b), power systems (Eichelbeck et al., 2022; Chen et al., 2022b; Zhang et al., 2023; Yu et al., 2023), and engine emission control (Norouzi et al., 2023). Wang (2022) compares the deployment of a discrete action masking approach with her continuous action projection approach. The goal-reaching performance is lower for the discrete action masking approach. However, this could be due to the coarse discretization of the action space in three actions.

Action masking To the best of our knowledge, the existing literature considers action masking only for discrete action spaces. The work Huang & Ontañón (2022) analyzes the effect of discrete action masking on the policy gradient algorithm in RL, but they assume that $\mathbb{A}_s$ is known, which is typically only the case in game and grid world environments.

The two main works investigating action masking are Fulton & Platzer (2018) and Fulton & Platzer (2019). They construct controller and model monitors based on theorem proving of differential dynamic logic specifications, see Platzer (2008). The controller monitor is used to build the mask $\boldsymbol{\eta}(\boldsymbol{s})$, and the model monitor verifies if the underlying system model is correct based on previous transitions. In each state, the agent can choose from the set of actions that the controller monitor verified as safe. Identifying the correct system can be challenging, thus an approach to automatically generate candidates is introduced as well. Their approach

is provably safe if the initial model is correct (Fulton & Platzer, 2018) or multiple models are given, from which at least one is correct (Fulton & Platzer, 2019) for all times. They validate their provably safe action masking on adaptive cruise control tasks.

In addition to the works mentioned above, there are works investigating action masking for the specific application of autonomous driving (Mirchevska et al., 2018; Krasowski et al., 2020; Brosowsky et al., 2021; Krasowski et al., 2022), power systems (Tabas & Zhang, 2022), adaptive routing in computing networks (Seetanadi et al., 2020), and urban traffic signal control (Müller & Sabatelli, 2022). The only application-specific approach that compares action masking with other provably safe RL approaches is Brosowsky et al. (2021). They observe that their masking approach converges slightly faster than action projection.

4 Experimental comparison

In this section, we evaluate the performance of the three provably safe RL classes and the four learning tuples introduced in Section 2. For our comparison, we select an inverted pendulum and a 2D quadrotor stabilization task[7], as these benchmarks are commonly evaluated in related works presented in Table 1. The provably safe state set $\mathbb{S}_\varphi$ is the same for all three approaches and, therefore, comparable. We add system disturbances to the benchmarks to make them more realistic and show that the provably safe RL approaches can handle disturbances sampled from a compact disturbance set. Despite their low dimensionality, our results are likely transferable to real-world systems since real-world complexity is often reduced in practice by using lower-dimensional abstract models and an additional disturbance term. Conformance checking techniques (Roehm et al., 2019; Liu et al., 2023) can then guarantee that the abstract model incorporates recorded real-world behaviors of the system.

The algorithms shown in this section are action replacement with $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$, action projection using affine constraints, and action masking. We compare each configuration on ten random seeds and five common RL algorithms[8]: continuous Twin Delayed Deep Deterministic policy gradient algorithm (TD3) (Fujimoto et al., 2018), continuous soft actor-critic (SAC) (Haarnoja et al., 2018), discrete DQN (Mnih et al., 2013), and continuous and discrete proximal policy optimization (PPO) (Schulman et al., 2017).

4.1 Environments

We compare the provably safe RL approaches on an inverted pendulum and a 2D quadrotor stabilization task.

Inverted pendulum The state of the pendulum is defined as $\boldsymbol{s} = \left[\theta, \dot{\theta}\right]^\top$, and follows the dynamics

$$\dot{\boldsymbol{s}} = \begin{pmatrix} \dot{\theta} \\ \frac{g}{l}\sin(\theta) + \frac{1}{ml^2}a \end{pmatrix}, \tag{12}$$

where a is the one-dimensional action, g is gravity, m is the mass of the pendulum, l its length, and friction and damping are ignored. We discretize the dynamics using the explicit Euler method. The actions are bounded by $|a| \leq 30\text{rad}\,\text{s}^{-1}$. The desired equilibrium state is $\boldsymbol{s}^* = [0, 0]^\top$. The observation and reward are identical to the *OpenAI Gym Pendulum-V0*[9] environment.

2D quadrotor The quadrotor in our experiments can only fly in the x-z-plane and rotate around the y-axis with angle θ. The state of the system is defined as $\boldsymbol{s} = \left[x, z, \dot{x}, \dot{z}, \theta, \dot{\theta}\right]^\top$ and the action as $\boldsymbol{a} = [a_1, a_2]^\top$.

[7] Our implementation is available at CodeOcean: doi.org/10.24433/CO.9209121.v1 .
[8] All implementations are based on `stable-baselines3` (Raffin et al., 2021).
[9] Available at: gymnasium.farama.org/environments/classic_control/pendulum/

The system dynamics

$$\dot{\boldsymbol{s}} = \begin{pmatrix} \dot{x} \\ \dot{z} \\ a_1 k \sin(\theta) \\ -g + a_1 k \cos(\theta) \\ \dot{\theta} \\ -d_0\theta - d_1\dot{\theta} + n_0 a_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ w_1 \\ w_2 \\ 0 \\ 0 \end{pmatrix} \tag{13}$$

are based on Mitchell et al. (2019), where w_1, w_2 represent the system disturbance, and k, d_0, d_1, and n_0 are constant parameters (see Table 4). We linearize the dynamics using a first-order Taylor expansion at the equilibrium point $\boldsymbol{s}^* = [0, 1, 0, 0, 0, 0]^\top$ and obtain the discrete-time system for the linearized dynamics. We sample the disturbance $\boldsymbol{w} = [w_1, w_2]^\top$ uniformly, independent, and identically distributed from a compact disturbance set $\mathbb{W} \subset \mathbb{R}^2$. The actions range from $\boldsymbol{a}_{\min} = \left[-1.5 + \frac{g}{K}, -\frac{\pi}{12}\right]^\top$ to $\boldsymbol{a}_{\max} = \left[1.5 + \frac{g}{K}, \frac{\pi}{12}\right]^\top$. The reward is defined as $r(\boldsymbol{s}, \boldsymbol{a}) = \exp\left(-\left\|\boldsymbol{s} - \boldsymbol{s}^*\right\|_2 - \frac{0.01}{2}\left\|\frac{\boldsymbol{a} - \min(\mathbb{A})}{\max(\mathbb{A}) - \min(\mathbb{A})}\right\|_1\right)$.

4.2 Computation of the safe state set

To obtain a possibly large set of provably safe states $\mathbb{S}_\varphi$ and a provably safe controller for our environments, we use the scalable approach for computing robust control invariant sets of nonlinear systems presented in Schäfer et al. (2023): for every state $\boldsymbol{s}_0 \in \mathbb{S}_\varphi \subset \mathbb{S}_{\boldsymbol{s}}$, there exists a provably safe action $\tilde{\boldsymbol{a}}_0 \in \mathbb{A}$ so that $\boldsymbol{s}_1 = \boldsymbol{g}(\boldsymbol{s}_0, \tilde{\boldsymbol{a}}_0, \boldsymbol{w}_0) \in \mathbb{S}_\varphi$ with a bounded disturbance $\boldsymbol{w}_0 \in \mathbb{W} \subset \mathbb{R}^O$ where $\mathbb{W}$ has O dimensions. Hence, $\varphi(\boldsymbol{s}_0, \tilde{\boldsymbol{a}}_0) = 1$ for every $\boldsymbol{s}_0 \in \mathbb{S}_\varphi$. In this work, we assume the disturbance to be constant in between sampling times. Note that we use the obtained provable safe controller for the failsafe replacement function $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$. The algorithm in Schäfer et al. (2023) provides an explicit representation of $\mathbb{S}_\varphi$, which enables a fair comparison of our provably safe RL implementations. To retrieve $\mathbb{A}_\varphi$ from $\mathbb{S}_\varphi$ at a given state $\boldsymbol{s}$, we first convert $\mathbb{S}_\varphi$ from generator representation, which is used in Schäfer et al. (2023), into halfspace representation, i.e., $\mathbb{S}_\varphi = \{\boldsymbol{s} | \boldsymbol{C}\boldsymbol{s} \leq \boldsymbol{q}\}$, using the open-source toolbox CORA (Althoff, 2015). We evaluate the safety function given the state $\boldsymbol{s}_k \in \mathbb{S}_\varphi$ and an action $\boldsymbol{a}_k \in \mathbb{A}$ by computing the reachable set $\mathcal{R}(k+1)$ at the next time step, which encloses the states that are reachable for all $\boldsymbol{w}_k \in \mathbb{W}$ (Althoff, 2015). The reachable set can be represented as a zonotope, i.e., $\mathcal{R}(k+1) = \{\boldsymbol{s}_{k+1} | \boldsymbol{s}_{k+1} = \boldsymbol{c} + \boldsymbol{G}\boldsymbol{\beta}, |\boldsymbol{\beta}|_\infty \leq 1\}$. If $\mathcal{R}(k+1) \subseteq \mathbb{S}_\varphi$, the action $\boldsymbol{a}$ is verified as safe, i.e., $\varphi(\boldsymbol{s}_{k+1}, \boldsymbol{a}_{k+1}) = 1$, which holds if and only if (Schürmann et al., 2020, Theorem 2)

$$\boldsymbol{C}\boldsymbol{c} + |\boldsymbol{C}\boldsymbol{G}|\mathbf{1} \leq \boldsymbol{q}\,, \tag{14}$$

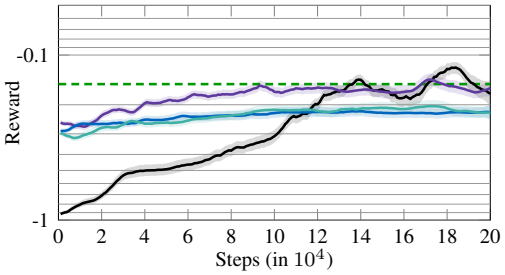
where the absolute value is applied elementwise and $\mathbf{1}$ denotes a vector full of ones of appropriate dimension. The approach of Schäfer et al. (2023) allows us to compute $\mathbb{S}_\varphi$ for high-dimensional nonlinear systems. However, the conversion to halfspace representation is computationally too expensive for higher dimensional systems. Therefore, we plan to develop generator-based versions of our provably safe RL methods in future work to mitigate this shortcoming.

4.3 Results

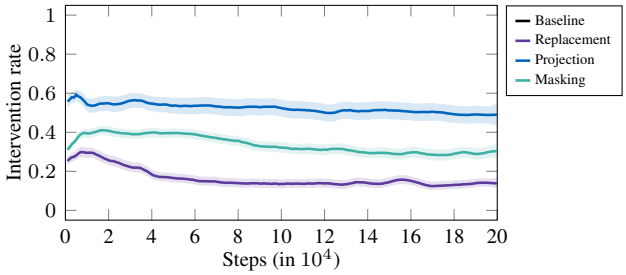
The 2D quadrotor task is the main comparison environment in this work as it is more complex and shows the differences between provably safe RL approaches clearer than the inverted pendulum task. We evaluate the differences between the provably safe RL algorithms in Figure 3 and the effect of different learning tuples in Figure 4. All training runs on all individual algorithms and environments are presented in the Appendix, including a comparison between the $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$ and $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$ replacement function.

Comparison of provably safe RL algorithms The safety violation evaluation of the baselines in Figure 3d shows that the baseline algorithms fail to guarantee safety during training in the 2D quadrotor stabilization task. All provably safe RL algorithms guarantee safety as expected. Between the baselines, TD3 converges significantly faster than all other algorithms.

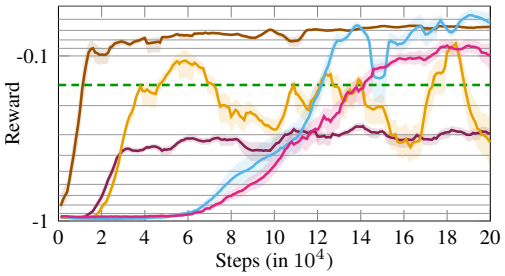
Figures 3a and 3b show the performance of the three provably safe RL categories (a) action replacement using $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$, (b) action projection, and (c) action masking together, with the baselines averaged over all



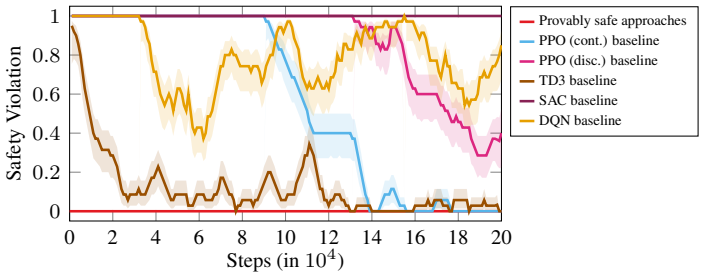
(a) Reward using provably safe RL algorithms.



(b) Intervention rate of provably safe RL algorithms.



(c) Reward using baselines.



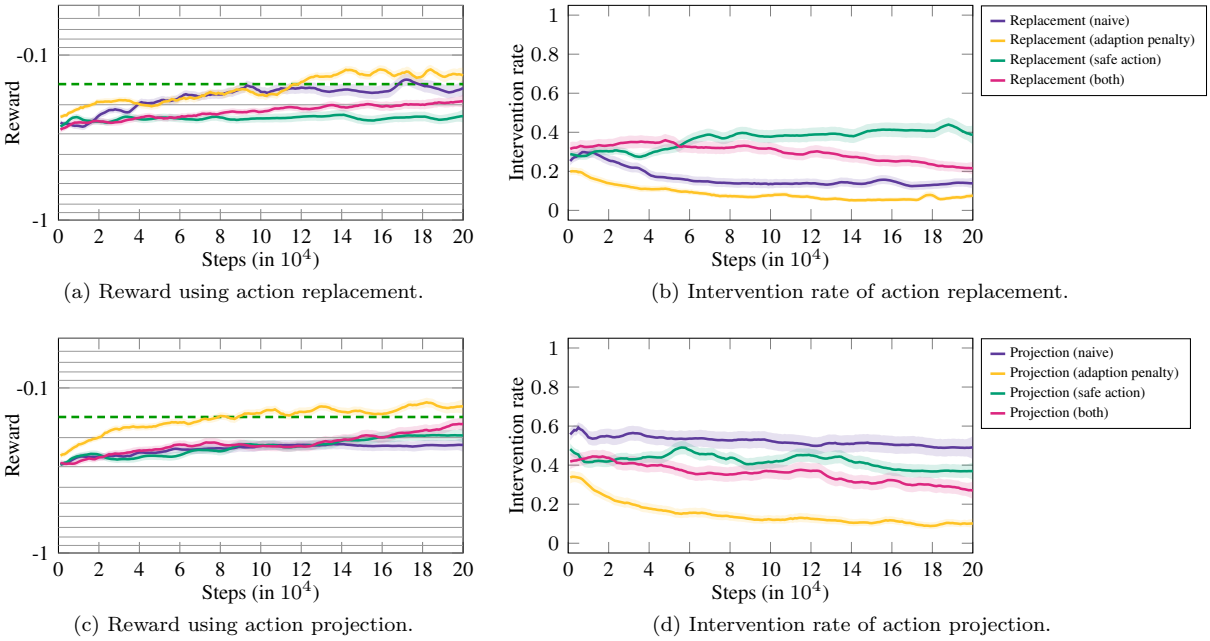
(d) Safety violation rate of baselines.

Figure 3: Training curves for the 2D quadrotor benchmark. Top: Comparison of the three provably safe RL classes and the unsafe benchmarks averaged over all algorithms trained with the *naive* tuple. Bottom: Comparison of benchmark algorithms TD3, SAC, DQN, continuous and discrete PPO. All training runs were conducted on ten random seeds per algorithm. The left column depicts the reward. The right column shows the safety violations for the baselines, and the safety intervention rate for the provably safe RL algorithms.

five RL implementations and trained using the *naive* learning tuple. For a better comparison of the reward curves in Figure 3a, we added a dashed green line that indicates the final training reward averaged over all five RL baselines and ten random seeds. The reward comparison shows that action replacement performs better than action projection and masking on average.

We also compare the intervention rate of the three safety mechanisms. For action replacement and projection, our intervention rate metric indicates the share of RL steps per episode in which the safety function altered the action. For action masking, the intervention rate compares the average volume of the provably safe action set over an episode with the volume of the provably safe action set at the equilibrium point of the system, e.g., $V_{\mathbb{A}_\varphi,\text{episode}}/V_{\mathbb{A}_\varphi,\text{equilibrium}}$. Figure 3b shows that action replacement relies significantly less on the safety mechanism than projection and masking. Generally, we report that a lower intervention rate often coincides with a higher reward.

Comparison of learning tuples We evaluate the impact of different learning tuples on the performance and intervention rates averaged over all five RL algorithms in Figure 4. When action masking is used, only safe actions can be sampled, i.e., only the *naive* tuple is meaningful; so we omit action masking from this evaluation. For both action replacement and projection, the *adaption penalty* tuple leads to the highest performance and lowest safety intervention rate, even outperforming the average over the baselines. In action projection, the *naive* tuple performs significantly worse than in action replacement. The *safe action* and *both* tuples seem to be only beneficial when using action projection and decrease performance when using action replacement.



(a) Reward using action replacement.

(b) Intervention rate of action replacement.

(c) Reward using action projection.

(d) Intervention rate of action projection.

Figure 4: Evaluation of the training tuples for the 2D quadrotor averaged over the five RL algorithms TD3, SAC, DQN, continuous and discrete PPO on ten random seeds per algorithm. The left column depicts the reward and the right column the safety intervention rate. The top row shows the different learning tuples for action replacement and the bottom row for action projection.

5 Discussion

Our experiments confirm the theoretical statement that provably safe RL methods are always safe when Proposition 1 holds. In the two tested environments, the investigated RL baselines show non-zero safety violations during training, even after convergence. Subsequently, we discuss five statements resulting from the experiments, provide intuitions for implementing provably safe RL, summarize the limitations, and identify future research directions in provably safe RL.

Selecting a provably safe RL approach First, we want to summarize our experience with the three provably safe RL classes on the two investigated benchmarks. Action replacement was the easiest method to implement for continuous action spaces. It shows very good performance and low intervention rates. Hereby, using a failsafe action for replacement with an adaption penalty is simple to implement and was among the best-performing methods in our experiments. Still, the random sampling of safe actions might outperform the failsafe action. So, if sampling from the safe action space is readily available, e.g., in discrete action spaces, a failsafe controller is unnecessary. Action projection tends to be problematic in practice due to small numerical errors, resulting in infeasible optimization problems. We, therefore, have to reuse previous optimization results, e.g., as in Schürmann et al. (2018) for robust MPC, or use a failsafe controller if the optimization problem is not solvable. Together with the higher intervention rates compared to action replacement and the complex implementation, we would not recommend action projection based on our experience. However, if one already has a CBF or MPC formulation, it might be the most suitable solution. Action masking is particularly easy to implement for discrete action spaces and performs well in that setting. However, for action masking in continuous action spaces, there is no efficient and general algorithm yet that can handle safe action spaces significantly diverging from an axis-aligned box. Generally, the different approaches can also be combined to some extent. For example, if the optimization problem for action projection becomes infeasible, a failsafe controller can be used.

Selecting a learning tuple The *adaption penalty* learning tuple performed best, especially when using action projection. In our experiments, a simple constant reward penalty already improved the performance. Other environments may require more careful reward tuning, or the *adaption penalty* tuple could fail altogether due to reward hacking (Skalse et al., 2022) or goal misgeneralization (Langosco et al., 2022). The results in Figure 4 further show that using the safe action $\tilde{\boldsymbol{a}}$ in the training tuple, i.e., configurations *both* and *safe action*, benefits the performance of action projection methods but impairs the training with action replacement. This effect can result from the fact that action replacement alters the action more than action projection, leading to a lower likelihood that the altered action stems from the RL policy. The evaluations in the Appendix show that this effect is prominent when using PPO. This on-policy algorithm assumes that the current batch of training data stems from the current policy. Hence, we would recommend using the *adaption penalty* tuple when possible and only using *safe action* in combination with off-policy methods.

Convergence The training of provably safe RL agents converges similarly fast or faster than the baselines in our experiments. In contrast, the performance (measured by the reward) at the beginning of the training is better for provably safe RL agents. One reason for the faster convergence is the exploration setting. Since $\mathbb{A}_\varphi \subseteq \mathbb{A}$, the provably safe agents learn in a usually smaller action space than the baselines, which can accelerate training. However, in some cases, the baseline agents might be better informed about the environment dynamics by exploring unsafe actions. Generally, the verification method should aim for $\mathbb{A}_\varphi = \mathbb{A}_{\boldsymbol{s}}$ such that the provably safe agents can explore the full safe action space. Another reason for convergence differences can be that action replacement and action projection potentially correct the action after the forward pass through the policy. Thus, the gradient calculation might need correction as well. So far, there is only little theoretical work on this (Hunt et al., 2021; Gros et al., 2020), and it is unclear if, in practice, a correction of the gradient is necessary or if using the *adaption penalty* tuple is sufficient. Furthermore, the change in the distribution of the actions can impact the exploration strategy, as discussed in Section 2.4.

Computational complexity The computational complexity of the three approaches highly depends on the scenario-specific implementation. For action projection, the main implementation challenge is to guarantee that the optimization problem is always feasible. If the optimization problem can be formulated as a quadratic program, the computational complexity is polynomial, as shown by Vavasis (2001). On the contrary, the computational complexity of action replacement and action masking depends highly on the algorithm that identifies the safety of actions. For discrete action masking, the computational complexity apparently scales linearly with the total number of actions $\mathcal{O}(|\mathbb{A}|)$. For action replacement, we only need a single safe action, so in the ideal case, e.g., using a failsafe controller, the computational complexity is constant with respect to the total number of actions. Suppose an action replacement approach needs to determine the entire set of safe actions, it obviously has the same computational complexity as action masking with respect to the number of actions. The computational complexity for identifying the continuous safe action space depends on the task-specific implementation. One possibility is to compute the safe action space using set-based reachability analysis, where we want to point the interested reader to Althoff et al. (2021) for different approaches.

Online vs. offline implementation *Online* and *offline* have two notions in provably safe RL: online vs. offline safety verification and online vs. offline RL. The safety function usually needs to be evaluated online since, for continuous state spaces, pre-computing the safe action set for all states is often not feasible. Thus, the computational complexity of the safety function is important for real-time applications, as discussed in the previous paragraph. If the state and action space are discrete, it can be possible to pre-compute the safe actions offline (Alshiekh et al., 2018; Huang & Ontañón, 2022). Generally, safety is only guaranteed if the safety function is integrated between the agent and environment to correct actions (see Figure 1). In this study, we compare online on-policy and off-policy RL algorithms (Levine et al., 2020) since they are used for existing provably safe RL research. Still, provably safe RL can also be used for offline RL where the safety function would be integrated during deployment and most likely also during the data gathering phase if this phase is conducted in a safety-critical environment. However, more specific advice on offline provably safe RL needs to be substantiated with experimental evaluations and, thus, is a topic for future research.

Limitations of provably safe RL There are limitations of provably safe RL that follow from the conceptual analysis in Section 2. Most importantly, safety has to be verifiable, i.e., there must be a safety function $\varphi(\boldsymbol{s}, \boldsymbol{a})$, which complies with Proposition 1. For this safety function, system knowledge is necessary, and especially for systems with a high number of continuous state variables, the safety function is potentially complex to compute. Additionally, safety guarantees are strongly tied to the safety function. If the safety function provides complex guarantees (e.g., ensures temporal logic specifications), it is usually computationally more expensive than for simpler guarantees (e.g., system stays within safe state set). Second, safety can only be decided if the state of the system is correctly observed within noise bounds. Thus, for a provably safe autonomous system, the perception module also needs to be verified such that it provides observations that are correct within the noise bounds. Third, there is often a trade-off between safety and performance since, for many tasks, these two objectives are only partially aligned. For example, if an automated vehicle drives faster, it reaches its destination earlier, but collisions are more difficult to avoid at higher speeds. Since provably safe RL ensures safe behavior, there is no such trade-off as safety is always prioritized over performance. Thus, the safety function $\varphi(\boldsymbol{s}, \boldsymbol{a})$ should not be too conservative since the agent would only perform trivial safe actions e.g., standing still at the side of the road forever. Lastly, comparing provably safe RL approaches is challenging as we need to define a safety function $\varphi(\boldsymbol{s}, \boldsymbol{a})$ that is efficiently usable by different approaches. Furthermore, the notion of safety is usually application-specific, so different application-specific approaches are hard to compare. We provide the first comparison of provably safe RL on two common continuous stabilization tasks, but further research is necessary to make more substantial claims about the most promising provably safe RL approaches.

Future research based on proposed taxonomy Most action projection approaches discussed in Section 3 project the RL action on the border of $\mathbb{A}_\varphi$. In our experiments, we encountered two negative side effects related to this action projection implementation: First, the projection to the border of $\mathbb{A}_\varphi$ often leads to a relatively small $\mathbb{A}_\varphi$ in the next RL step, quickly resulting in a very small set $\mathbb{A}_\varphi$ if the RL agent proposes a few unsafe actions after each other. Second, small numerical errors can cause unsafe actions and must be considered in the safety verification. Therefore, future action projection research should investigate objective functions for (3) that achieve a more robust behavior while still depending on the action the RL agent proposed, e.g., projecting the action not to the border but by a learnable margin inside the provably safe action set. Action masking is a promising technique but has mainly been used with discrete action spaces in grid world environments and games, e.g., the Atari benchmark (Huang & Ontañón, 2022). Our proposed continuous action masking approach only applies to specific environments and performed well for the pendulum but showed mixed results for the 2D quadrotor. Thus, future research should investigate ways to extend continuous action masking to general convex or non-convex representations of $\mathbb{A}_\varphi$ to improve its applicability to more complex benchmarks and reduce the conservativeness of $\mathbb{A}_\varphi$. Additionally, it should be investigated if the agent should be informed about the reduction of the action space in action masking. This could result in improved convergence and an agent that is more aware of the action mask, similar to the effect of the *adaption penalty* tuple for action replacement and action projection. The evaluation of the considered benchmarks shows that action replacement performs better than action projection and masking, as discussed previously. However, it is still unclear how important the replacement strategy $\boldsymbol{\psi}(\boldsymbol{s})$ is for the convergence and performance of the agent, especially when applied to more complex tasks. Thus, future action replacement research should empirically and theoretically investigate this question.

Improving the applicability of provably safe RL Despite the promising previous work discussed in Section 3, there are only few works on high-dimensional nonlinear systems and limited real-world applications. We suggest five major factors where future research would improve applicability. First, some approaches need to be computationally more efficient to be real-world applicable. The computational efficiency of verification methods is especially relevant and should be improved, as discussed previously. Second, we observe that the learning tuple used has a significant influence on the performance of the agent for some RL algorithms. Also, there needs to be more theoretical research on how provably safe RL approaches influence convergence to an optimal policy. More empirical and theoretical research on the effects of provably safe RL and its learning tuples for convergence is desirable. Third, common benchmarks are necessary to evaluate new provably safe RL approaches. Additionally, the three action correction strategies should be compared on more complex benchmarks to clarify if our observations can be extended to them. Such benchmarking

would make research on provably safe RL more comparable, ease starting research on provably safe RL, and provide more evidence to decide on the best-suited provably safe RL approach. Fourth, recent work shows a low variety of safety specifications, mainly comprising stabilization and reach-avoid specifications. On the contrary, real-world safety is more complex, e.g., traffic rules such as waiting at a red light and safely but quickly moving at a green light. Finally, provably safe RL requires expert knowledge of verification methods. Future research could mitigate this through modular and automatic approaches, where fewer engineering decisions are necessary and more parameters are tuned automatically. With these advances, provably safe RL could bring the best elements of RL and formal specifications together towards RL methods that require as little expert knowledge as necessary and provide formal guarantees for complex safety specifications to achieve reliable and trustworthy cyber-physical systems.

6 Conclusion

In conclusion, we categorize provably safe RL methods to structure the literature from a machine learning perspective. We present provably safe RL methods from a conceptual perspective and discuss necessary assumptions. Our proposed categorization into action replacement, action projection, and action masking supports researchers in comparing their works and provides valuable insights into the selection process of provably safe RL methods. The comparison of four implementations of provably safe RL on a 2D quadrotor and an inverted pendulum stabilization benchmark provides further insights into the best-suited method for different tasks. We further present practical recommendations for selecting a provably safe RL approach and a learning tuple, which will be valuable for researchers who are new to RL or formal methods. Lastly, as discussed in Section 5, our proposed taxonomy and experimental evaluation yield multiple promising future research directions.

Acknowledgments

The authors gratefully acknowledge the partial financial support of this work by the research training group ConVeY, funded by the German Research Foundation under grant GRK 2428, by the project TRAITS under grant number 01IS21087, funded by the German Federal Ministry of Education and Research, by the Horizon 2020 EU Framework Project CONCERT under grant number 101016007, by the project justITSELF funded by the European Research Council (ERC) under grant agreement number 817629, and by the German Federal Ministry for Economics Affairs and Climate Action project VaF under grant number KK5135901KG0.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 22–31, 2017.

Anayo K. Akametalu, Shahab Kaynama, Jaime F. Fisac, Melanie N. Zeilinger, Jeremy H. Gillula, and Claire J. Tomlin. Reachability-based safe learning with Gaussian processes. In *Proc. of the IEEE Conf. on Decision and Control (CDC)*, pp. 1424–1431, 2014.

Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications. In *Proc. of the IEEE Conf. on Decision and Control (CDC)*, pp. 6565–6570, 2016.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, pp. 2669–2678, 2018.

Matthias Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pp. 120–151, 2015.

Matthias Althoff, Goran Frehse, and Antoine Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):369–395, 2021.

Eitan Altman. Constrained Markov decision processes with total cost criteria: Lagrangian approach and dual linear program. *Mathematical Methods of Operations Research*, 48(3):387–417, 1998.

Rajeev Alur, Suguman Bansal, Osbert Bastani, and Kishor Jothimurugan. *A Framework for Transforming Specifications in Reinforcement Learning*, pp. 604–624. Springer Nature Switzerland, 2022.

Rajeev Alur, Osbert Bastani, Kishor Jothimurugan, Mateo Perez, Fabio Somenzi, and Ashutosh Trivedi. Policy synthesis and reinforcement learning for discounted LTL. In *Proc. of the Int. Conf. on Computer Aided Verification (CAV)*, pp. 415–435, 2023.

Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *Proc. of the European Control Conference (ECC)*, pp. 3420–3431, 2019.

Greg Anderson, Abhinav Verma, Isil Dillig, and Swarat Chaudhuri. Neurosymbolic reinforcement learning with formally verified exploration. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 6172–6183, 2020.

Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT press, 2008.

Osbert Bastani. Safe reinforcement learning with nonlinear dynamics via model predictive shielding. In *Proc. of the American Control Conf. (ACC)*, pp. 3488–3494, 2021.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, pp. 2499–2509, 2018.

Felix Berkenkamp, Angela P. Schoellig, Matteo Turchetta, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, pp. 908–918, 2017.

Sara Bouraine, Thierry Fraichard, and Hassen Salhi. Provably safe navigation for mobile robots with limited field-of-views in unknown dynamic environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 174–179, 2012.

Mathis Brosowsky, Florian Keck, Jakob Ketterer, Simon Isele, Daniel Slieter, and Marius Zöllner. Safe deep reinforcement learning for adaptive cruise control by imposing state-specific safe sets. In *Proc. of the IEEE Intelligent Vehicles Symp. (IV)*, pp. 488–495, 2021.

Lukas Brunke, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

Mingyu Cai, Mohammadhosein Hasanbeig, Shaoping Xiao, Alessandro Abate, and Zhen Kan. Modular deep reinforcement learning for continuous motion planning with temporal logic. *IEEE Robotics and Automation Letters*, 6(4):7973–7980, 2021.

Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 6065–6073, 2019.

Glenn Ceusters, Luis Ramirez Camargo, Rüdiger Franke, Ann Nowé, and Maarten Messagie. Safe reinforcement learning for multi-energy management systems with known constraint functions. *Energy and AI*, 12, 2023.

Dong Chen, Longsheng Jiang, Yue Wang, and Zhaojian Li. Autonomous driving using safe reinforcement learning by incorporating a regret-based human lane-changing decision model. In *Proc. of the American Control Conf. (ACC)*, pp. 4355–4361, 2020.

Mo Chen and Claire J. Tomlin. Hamilton-Jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):333–358, 2018.

Shengduo Chen, Yaowei Sun, Dachuan Li, Qiang Wang, Qi Hao, and Joseph Sifakis. Runtime safety assurance for learning-enabled control of autonomous driving vehicles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 8978–8984, 2022a.

Yize Chen, Yuanyuan Shi, Daniel Arnold, and Sean Peisert. SAVER: Safe learning-based controller for real-time voltage regulation. In *Proc. of the IEEE Power and Energy Society General Meeting (PESGM)*, pp. 1–5, 2022b.

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. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, pp. 3387–3395, 2019.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A Lyapunov-based approach to safe reinforcement learning. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, pp. 8103–8112, 2018.

Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv*, abs/1801.0, 2018.

Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio Patrizi. Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pp. 128–136, 2021.

Michael Eichelbeck, Hannah Markgraf, and Matthias Althoff. Contingency-constrained economic dispatch with safe reinforcement learning. In *Proc. of the IEEE Int. Conf. on Machine Learning and Applications (ICMLA)*, pp. 597–602, 2022.

Mohamed El-Shamouty, Xinyang Wu, Shanqi Yang, Marcel Albus, and Marco F. Huber. Towards safe human-robot collaboration using deep reinforcement learning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 4899–4905, 2020.

Benjamin D. Evans, Hendrik W. Jordaan, and Herman A. Engelbrecht. Safe reinforcement learning for high-speed autonomous racing. *Cognitive Robotics*, 3:107–126, 2023.

Jaime F. Fisac, Anayo K. Akametalu, Melanie N. Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2019.

Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 2587–2601, 2018.

Nathan Fulton and André Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, pp. 6485–6492, 2018.

Nathan Fulton and André Platzer. Verifiably safe off-model reinforcement learning. In *Proc. of the Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 413–430, 2019.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015.

Jeremy H. Gillula and Claire J. Tomlin. Reducing conservativeness in safety guarantees by learning disturbances online: Iterated guaranteed safe online learning. *Robotics: Science and Systems*, 8(1):81–88, 2013.

Sebastien Gros, Mario Zanon, and Alberto Bemporad. Safe reinforcement learning via projection on a safe set: How to achieve optimality? *IFAC-PapersOnLine*, 53(2):8076–8081, 2020.

Felix Gruber and Matthias Althoff. Scalable robust output feedback MPC of linear sampled-data systems. *Proc. of the IEEE Conf. on Decision and Control (CDC)*, pp. 2563–2570, 2021.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 1861–1870, 2018.

Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. In *Proc. of the Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 395–412, 2019.

Habtamu Hailemichael, Beshah Ayalew, Lindsey Kerbel, Andrej Ivanco, and Keith Loiselle. Safety filtering for reinforcement learning-based adaptive cruise control. *IFAC-PapersOnLine*, 55(24):149–154, 2022a.

Habtamu Hailemichael, Beshah Ayalew, Lindsey Kerbel, Andrej Ivanco, and Keith Loiselle. Safe reinforcement learning for an energy-efficient driver assistance system. In *IFAC-PapersOnLine*, volume 55:37, pp. 615–620, 2022b.

Andrew Harris and Hanspeter Schaub. Spacecraft command and control with safety guarantees using shielded deep reinforcement learning. In *AIAA Scitech 2020 Forum*, volume 1, 2020.

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 *Proc. of the IEEE Conf. on Decision and Control (CDC)*, pp. 5338–5343, 2019a.

Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Towards verifiable and safe model-free reinforcement learning. In *Proc. of the Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis*, pp. 1–9, 2019b.

Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Cautious reinforcement learning with logical constraints. In *Proc. of the Int. Conf. on Autonomous Agents and Multi Agent Systems (AAMAS)*, pp. 483–491, 2020.

Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. LCRL: Certified policy synthesis via logically-constrained reinforcement learning. In *Proc. of the Int. Conf. on Quantitative Evaluation of Systems (QEST)*, pp. 217–231, 2022.

Matthias Heger. Consideration of risk in reinforcement learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 105–111, 1994.

Sylvia Herbert, Jason J. Choi, Suvansh Sanjeev, Marsalis Gibson, Koushil Sreenath, and Claire J. Tomlin. Scalable learning of safety guarantees for autonomous systems using Hamilton-Jacobi reachability. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 5914–5920, 2021.

Lukas Hewing, Kim P. Wabersich, Marcel Menner, and Melanie N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):269–296, 2020.

Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. *The Int. Florida Artificial Intelligence Research Society Conf. Proc. (FLAIRS)*, 35, 2022.

Nathan Hunt, Nathan Fulton, Sara Magliacane, Trong Nghia Hoang, Subhro Das, and Armando Solar-Lezama. Verifiably safe exploration for end-to-end reinforcement learning. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pp. 1–11, 2021.

B. Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A.Al Sallab, Senthil Yogamani, and Patrick Perez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2022.

Niklas Kochdumper, Hanna Krasowski, Xiao Wang, Stanley Bak, and Matthias Althoff. Provably safe reinforcement learning via action projection using reachability analysis and polynomial zonotopes. *IEEE Open Journal of Control Systems*, 2:79–92, 2023.

Bettina Könighofer, Florian Lorber, Nils Jansen, and Roderick Bloem. Shield synthesis for reinforcement learning. In *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*, pp. 290–306, 2020.

Bettina Könighofer, Julian Rudolf, Alexander Palmisano, Martin Tappler, and Roderick Bloem. Online shielding for stochastic systems. In *Proc. of the NASA Formal Methods (NFM)*, pp. 231–248, 2021.

Hanna Krasowski, Xiao Wang, and Matthias Althoff. Safe reinforcement learning for autonomous lane changing using set-based prediction. In *Proc. of the IEEE Int. Intelligent Transportation Systems Conf. (ITSC)*, pp. 1–7, 2020.

Hanna Krasowski, Yinqiang Zhang, and Matthias Althoff. Safe reinforcement learning for urban driving using invariably safe braking sets. In *Proc. of the IEEE Int. Intelligent Transportation Systems Conf. (ITSC)*, pp. 2407–2414, 2022.

Lauro Langosco Di Langosco, Jack Koch, Lee D. Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 12004–12019, 2022.

Dongsu Lee and Minhae Kwon. ADAS-RL: Safety learning approach for stable autonomous driving. *ICT Express*, 8(3):479–483, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv:2005.01643*, 2020.

Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3834–3839, 2017.

Xiao Li, Zachary Serlin, Guang Yang, and Calin Belta. A formal methods approach to interpretable reinforcement learning for robotic planning. *Science Robotics*, 4(37), 2019a.

Zhaojian Li, Tianshu Chu, and Uroš Kalabić. Dynamics-enabled safe deep reinforcement learning: Case study on active suspension control. In *Proc. of the IEEE Conf. on Control Technology and Applications (CCTA)*, pp. 585–591, 2019b.

Zemin Eitan Liu, Quan Zhou, Yanfei Li, Shijin Shuai, and Hongming Xu. Safe deep reinforcement learning-based constrained optimal control scheme for HEV energy management. *IEEE Transactions on Transportation Electrification*, 9(3):4278–4293, 2023.

Tommaso Mannucci, Erik-Jan van Kampen, Cornelis de Visser, and Qiping Chu. Safe exploration algorithms for reinforcement learning controllers. *IEEE Transactions on Neural Networks and Learning Systems*, 29 (4):1069–1081, 2018.

Zahra Marvi and Bahare Kiumarsi. Safe reinforcement learning: A control barrier function optimization approach. *International Journal of Robust and Nonlinear Control*, 31(6):1923–1940, 2021.

Zahra Marvi and Bahare Kiumarsi. Reinforcement learning with safety and stability guarantees during exploration for linear systems. *IEEE Open Journal of Control Systems*, 1:322–334, 2022.

Branka Mirchevska, Christian Pek, Moritz Werling, Matthias Althoff, and Joschka Boedecker. High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning. In *Proc. of the IEEE Int. Intelligent Transportation Systems Conf. (ITSC)*, pp. 2156–2162, 2018.

Ian M. Mitchell, Jacob Budzis, and Andriy Bolyachevets. Invariant, viability and discriminating kernel under-approximation via zonotope scaling. In *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pp. 268–269, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv*, abs/1312.5, 2013.

Arthur Müller and Matthia Sabatelli. Safe and psychologically pleasant traffic signal control with reinforcement learning using action masking. *Proc. of the IEEE Conf. on Intelligent Transportation Systems (ITSC)*, pp. 951–958, 2022.

Islam Nazmy, Andrew Harris, Morteza Lahijanian, and Hanspeter Schaub. Shielded deep reinforcement learning for multi-sensor spacecraft imaging. In *Proc. of the American Control Conf. (ACC)*, pp. 1808–1813, 2022.

Armin Norouzi, Saeid Shahpouri, David Gordon, Mahdi Shahbakhti, and Charles Robert Koch. Safe deep reinforcement learning in diesel engine emission control. *Proc. of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, 2023.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning, 2018.

Christian Pek, Stefanie Manzinger, Markus Koschi, and Matthias Althoff. Using online verification to prevent autonomous vehicles from causing accidents. *Nature Machine Intelligence*, 2(9):518–528, 2020.

Tu-Hoa Pham, Giovanni De Magistris, and Ryuki Tachibana. OptLayer - practical constrained optimization for deep reinforcement learning in the real world. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 6236–6243, 2018.

André Platzer. Differential dynamic logic for hybrid systems. *Journal of Automated Reasoning*, 41(2): 143–189, 2008.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Hendrik Roehm, Jens Oehlerking, Matthias Woehrle, and Matthias Althoff. Model conformance for cyber-physical systems: A survey. *ACM Transactions on Cyber-Physical Systems*, 3(3):1–26, 2019.

Lukas Schäfer, Felix Gruber, and Matthias Althoff. Scalable computation of robust control invariant sets of nonlinear systems. *IEEE Transactions on Automatic Control*, (early acces):1–15, 2023.

Lukas M. Schmidt, Georgios D. Kontes, Axel Plinge, and Christopher Mutschler. Can you trust your autonomous car? interpretable and verifiably safe reinforcement learning. In *Proc. of the IEEE Intelligent Vehicles Symp. (IV)*, pp. 171–178, 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, abs/1707.0, 2017.

Bastian Schürmann, Niklas Kochdumper, and Matthias Althoff. Reachset model predictive control for disturbed nonlinear systems. In *Proc. of the IEEE Conf. on Decision and Control (CDC)*, pp. 3463–3470, 2018.

Bastian Schürmann, Riccardo Vignali, Maria Prandini, and Matthias Althoff. Set-based control for disturbed piecewise affine systems with state and actuation constraints. *Nonlinear Analysis: Hybrid Systems*, 36 (Art. no. 100826), 2020.

Gautham Nayak Seetanadi, Karl-Erik Årzén, and Martina Maggio. Adaptive routing with guaranteed delay bounds using safe reinforcement learning. In *ACM Int. Conf. Proc. Series*, pp. 149–160, 2020.

Mahmoud Selim, Amr Alanwar, M. Watheq El-Kharashi, Hazem M. Abbas, and Karl H. Johansson. Safe reinforcement learning using data-driven predictive control. In *Proc. of the Int. Conf. on Communications, Signal Processing, and their Applications (ICCSPA)*, pp. 1–6, 2022a.

Mahmoud Selim, Amr Alanwar, Shreyas Kousik, Grace Gao, Marco Pavone, and Karl H. Johansson. Safe reinforcement learning using black-box reachability analysis. *IEEE Robotics and Automation Letters*, 7 (4):10665–10672, 2022b.

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.

Suhas Shyamsundar, Tommaso Mannucci, and Erik-Jan Van Kampen. Reinforcement learning based algorithm with safety handling and risk perception. In *Proc. of the IEEE Symp. Series on Computational Intelligence (SSCI)*, pp. 1–7, 2017.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Van Den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Joar Max Viktor Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2022.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID Lagrangian methods. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 9133–9143, 2020.

Zachary Sunberg and Mykel Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 28(1): 259–263, 2018.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Leaning: An Introduction*. A Bradford Book, 2nd edition, 2018.

Daniel Tabas and Baosen Zhang. Computationally efficient safe reinforcement learning for power systems. In *Proc. of the American Control Conf. (ACC)*, pp. 3303–3310, 2022.

Andrew Taylor, Andrew Singletary, Yisong Yue, and Aaron D. Ames. Learning for safety-critical control with control barrier functions. In *Proc. of the Conf. on Learning for Dynamics and Control*, pp. 708–717, 2020.

Andrew J. Taylor, Andrew Singletary, Yisong Yue, and Aaron D. Ames. A control barrier perspective on episodic learning via projection-to-state safety. *IEEE Control Systems Letters*, 5(3):1019–1024, 2021.

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E. Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery RL: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021.

Jakob Thumm and Matthias Althoff. Provably safe deep reinforcement learning for robotic manipulation in human environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 6344–6350, 2022.

Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration in finite Markov decision processes with Gaussian processes. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, pp. 4312–4320, 2016.

Peter Varnai and Dimos V. Dimarogonas. On robustness metrics for learning STL tasks. In *Proc. of the American Control Conf. (ACC)*, pp. 5394–5399, 2020.

Stephen A. Vavasis. *Complexity Theory: Quadratic Programming*. Springer, Boston, MA., 2001.

Kim P. Wabersich and Melanie N. Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129(1):109597–109614, 2021.

Kim P. Wabersich, Andrew J. Taylor, Jason J. Choi, Koushil Sreenath, Claire J. Tomlin, Aaron D. Ames, and Melanie N. Zeilinger. Data-driven safety filters: Hamilton-Jacobi reachability, control barrier functions, and predictive methods for uncertain systems. *IEEE Control Systems Magazine*, 43(5):137–177, 2023.

Chengyu Wang, Luhan Wang, Zhaoming Lu, Xinghe Chu, Zhengrui Shi, Jiayin Deng, Tianyang Su, Guochu Shou, and Xiangming Wen. SRL-TR2: A safe reinforcement learning based trajectory tracker framework. *IEEE Transactions on Intelligent Transportation Systems*, 24(6):5765–5780, 2023.

Linghao Wang, Miao Wang, and Yujun Zhang. A safe training approach for deep reinforcement learning-based traffic engineering. In *Proc. of the IEEE Int. Conf. on Communications (ICC)*, pp. 1450–1455, 2022.

Xiao Wang. Ensuring safety of learning-based motion planners using control barrier functions. *IEEE Robotics and Automation Letters*, 7(2):4773–4780, 2022.

Peter Wieland and Frank Allgöwer. Constructive safety using control barrier functions. *IFAC Proc. Volumes*, 40(12):462–467, 2007.

Cambridge Yang, Michael L. Littman, and Michael Carbin. On the (in)tractability of reinforcement learning for LTL objectives. *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 3650–3658, 2022.

Chenchen Yang, Jing Liu, Haiying Sun, Junfeng Sun, Xiang Chen, and Lipeng Zhang. Safe reinforcement learning for CPSs via formal modeling and verification. In *IEEE Int. Joint Conf. on Neural Networks Proc. (IJCNN)*, pp. 1–8, 2021.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. Projection-based constrained policy optimization. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, pp. 1–21, 2020.

Fei Ye, Shen Zhang, Pin Wang, and Ching Yao Chan. A survey of deep reinforcement learning algorithms for motion planning and control of autonomous vehicles. In *Proc. of the IEEE Intelligent Vehicles Symp. (IV)*, pp. 1073–1080, 2021.

Peipei Yu, Hongcai Zhang, Yonghua Song, Hongxun Hui, and Ge Chen. District cooling system control for providing operating reserve based on safe deep reinforcement learning. *IEEE Transactions on Power Systems*, pp. 1–13, 2023.

Mario Zanon and Sebastien Gros. Safe reinforcement learning using robust MPC. *IEEE Transactions on Automatic Control*, 66(8):3638–3652, 2021.

Melanie N. Zeilinger, Davide M. Raimondo, Alexander Domahidi, Manfred Morari, and Colin N. Jones. On real-time robust model predictive control. *Automatica*, 50(3):683–694, 2014.

Jin Zhang, Yuxiang Guan, Liang Che, and Mohammad Shahidehpour. Ev charging command fast allocation approach based on deep reinforcement learning with safety modules. *IEEE Transactions on Smart Grid*, 2023.

Wenshuai Zhao, Jorge Pena Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In *Proc. of the IEEE Symp. Series on Computational Intelligence (SSCI)*, pp. 737–744, 2020.

A Appendix

MDP modification with action replacement

Action replacement alters the MDP on which the agent learns. Hunt et al. (2021) discuss this modification for discrete action spaces and uniformly sampling from the safe action space. We generalize this discussion to using any replacement function and continuous action spaces. To this end, we define $\boldsymbol{\psi}(\boldsymbol{s})$ so that it randomly samples the replacement action $\tilde{\boldsymbol{a}}$ according to a replacement policy $\boldsymbol{\pi}_r(\tilde{\boldsymbol{a}}|\boldsymbol{s})$ with $\sum_{\tilde{\boldsymbol{a}}\in\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r(\tilde{\boldsymbol{a}}|\boldsymbol{s})=1$ for the discrete case, and $\int_{\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r(\tilde{\boldsymbol{a}}|\boldsymbol{s})\,d\tilde{\boldsymbol{a}}=1$ for the continuous case, and $\forall\tilde{\boldsymbol{a}}\in\mathbb{A}_\varphi(\boldsymbol{s}):\boldsymbol{\pi}_r(\tilde{\boldsymbol{a}}|\boldsymbol{s})\geq 0$. In the example of uniform sampling from $\mathbb{A}_\varphi(\boldsymbol{s})$, the replacement policy is $\boldsymbol{\pi}_r(\tilde{\boldsymbol{a}}|\boldsymbol{s})=1/V_{\mathbb{A}_\varphi(\boldsymbol{s})}$ where $V_{\mathbb{A}_\varphi(\boldsymbol{s})}$ is the volume of $\mathbb{A}_\varphi(\boldsymbol{s})$. By replacing unsafe actions, the transition function of the MDP changes to

$$T_\varphi(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')=\begin{cases}T(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}'), & \text{if } \varphi(\boldsymbol{s},\boldsymbol{a})=1\\ T_r(\boldsymbol{s},\boldsymbol{s}'), & \text{otherwise,}\end{cases} \tag{15}$$

$$T_r(\boldsymbol{s},\boldsymbol{s}')=\sum_{\tilde{\boldsymbol{a}}\in\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r(\tilde{\boldsymbol{a}}|\boldsymbol{s})\,T(\boldsymbol{s},\tilde{\boldsymbol{a}},\boldsymbol{s}'). \tag{16}$$

The reward function of the MDP changes accordingly to

$$r_\varphi(\boldsymbol{s},\boldsymbol{a})=\begin{cases}r(\boldsymbol{s},\boldsymbol{a}), & \text{if } \varphi(\boldsymbol{s},\boldsymbol{a})=1\\ r_r(\boldsymbol{s}), & \text{otherwise,}\end{cases} \tag{17}$$

$$r_r(\boldsymbol{s})=\sum_{\tilde{\boldsymbol{a}}\in\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r(\tilde{\boldsymbol{a}}|\boldsymbol{s})\,r(\boldsymbol{s},\tilde{\boldsymbol{a}}). \tag{18}$$

In the continuous case, we get $T_r(\boldsymbol{s},\boldsymbol{s}')$ by marginalizing the transition probability density function over $\mathbb{A}_\varphi(\boldsymbol{s})$:

$$T_r(\boldsymbol{s},\boldsymbol{s}')=\int_{\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r(\tilde{\boldsymbol{a}}|\boldsymbol{s})\,T(\boldsymbol{s},\tilde{\boldsymbol{a}},\boldsymbol{s}')d\tilde{\boldsymbol{a}}. \tag{19}$$

Analogously, we have that

$$r_r(\boldsymbol{s})=\int_{\mathbb{A}_\varphi(\boldsymbol{s})}\boldsymbol{\pi}_r(\tilde{\boldsymbol{a}}|\boldsymbol{s})\,r(\boldsymbol{s},\tilde{\boldsymbol{a}})d\tilde{\boldsymbol{a}}. \tag{20}$$

Environment parameters

We provide an overview of all environment-specific parameters in Table 3 and Table 4.

Table 3: Environment parameters of the pendulum.

Parameter	Value
Gravity g	$9.81\,\mathrm{m\,s^{-2}}$
Mass m	$1\,\mathrm{kg}$
Length l	$1\,\mathrm{m}$

Hyperparameters for learning algorithms

We specify the hyperparameters for all learning algorithms (see Table 5 for PPO, Table 6 for TD3, Table 7 for DQN, and Table 8 for SAC) that are different from the Stable Baselines3 (Raffin et al., 2021) default values. Additionally, the code for the experiments is available at the CodeOcean capsule doi.org/10.24433/CO.9209121.v1 to reproduce our results.

Table 4: Environment parameters of the 2D quadrotor.

Parameter	Value
Gravity g	$9.81\,\mathrm{m\,s^{-2}}$
k	$1\,1/\mathrm{kg}$
d_0	70
d_1	17
n_0	55
$\mathbb{W}$	$[[-0.1, 0.1], [-0.1, 0.1]]$

Table 5: Hyperparameters for PPO.

Parameter	Pendulum	2D quadrotor
Learning rate	1×10^{-4}	5×10^{-5}
Discount factor γ	0.98	0.999
Steps per update	2048	512
Optimization epochs	20	30
Minibatch size	16	128
Max gradient clipping	0.9	0.5
Entropy coefficient	1×10^{-3}	2×10^{-6}
Value function coefficient	0.045	0.5
Clipping range	0.3	0.1
Generalized advantage estimation λ	0.8	0.92
Activation function	ReLU	ReLU
Hidden layers	2	2
Neurons per layer	32	64
Training steps	60k	200k

Table 6: Hyperparameters for TD3.

Parameter	Pendulum	2D quadrotor
Learning rate	3.5×10^{-3}	2×10^{-3}
Replay buffer size	1×10^{4}	1×10^{5}
Discount factor γ	0.98	0.98
Initial exploration steps	10×10^{3}	100
Steps between model updates	256	5
Gradient steps per model update	256	10
Minibatch size per gradient step	512	512
Soft update coefficient τ	5×10^{-3}	5×10^{-3}
Gaussian smoothing noise σ	0.2	0.12
Activation function	ReLU	ReLU
Hidden layers	2	2
Neurons per layer	32	64
Training steps	60k	200k

Table 7: Hyperparameters for DQN.

Parameter	Pendulum	2D quadrotor
Learning rate	2×10^{-3}	1×10^{-4}
Replay buffer size	5×10^{4}	1×10^{6}
Discount factor γ	0.95	0.999 99
Initial exploration steps	500	100
Steps between model updates	8	2
Gradient steps per model update	4	4
Minibatch size per gradient step	512	64
Maximum for gradient clipping	10	100
Update frequency target network	1×10^{3}	1×10^{3}
Initial exploration probability ϵ	1.0	0.137
Linear interpolation steps of ϵ	6×10^{3}	1×10^{4}
Final exploration probability ϵ	0.1	0.004
Activation function	tanh	tanh
Hidden layers	2	2
Neurons per layer	32	64
Training steps	60k	200k

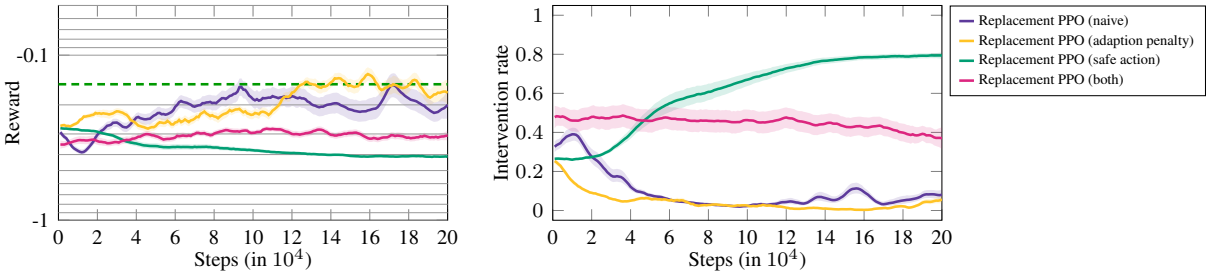
Table 8: Hyperparameters for SAC.

Parameter	Pendulum	2D quadrotor
Learning rate	3×10^{-4}	3×10^{-4}
Replay buffer size	1×10^{6}	5×10^{5}
Discount factor γ	0.99	0.98
Initial exploration steps	100	1000
Steps between model updates	1	32
Gradient steps per model update	1	32
Minibatch size per gradient step	256	512
Entropy coefficient	learned	1×10^{-1}
Soft update coefficient τ	5×10^{-3}	1×10^{-2}
Activation function	ReLU	ReLU
Hidden layers	2	2
Neurons per layer	32	64
Training steps	60k	200k

Full evaluation

In this section, we present all training results of the five RL algorithms TD3, SAC, DQN, and PPO continuous and discrete. We compare these algorithms on the inverted pendulum and 2D quadrotor environment on ten random seeds. The tested algorithms are action replacement with $\boldsymbol{\psi}_{\text{sample}}(\boldsymbol{s})$ and $\boldsymbol{\psi}_{\text{failsafe}}(\boldsymbol{s})$, action projection, and action masking. When action replacement is used with a failsafe controller, we omit *safe action* and *both* because for discrete action spaces, the failsafe controller might use an action that is not in the discrete action space. This is due to the failsafe controller proposing actions from the continuous action space.

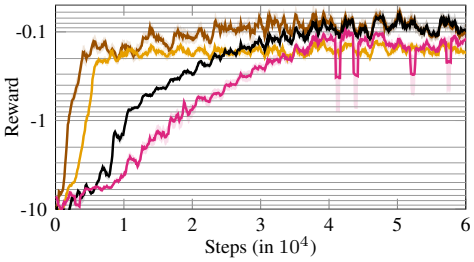
First, we present the effect of the learning tuples on the on-policy algorithm PPO in Figure 5. This comparison clearly shows the negative effect of the *safe action* tuple on the training performance of PPO. Figure 6 depicts the aggregated training results for the pendulum as previously discussed for the 2D quadrotor in Figures 3, 4 and 5. Figures 7 to 10 depict how the reward and intervention rate evolve during training for all investigated configurations. Finally, the Tables 9 and 10 show statistical results for deploying the learned models for the two benchmarks.



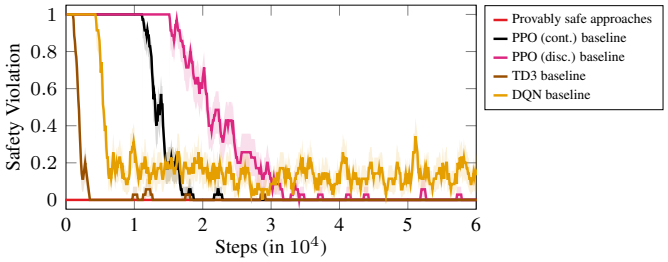
(a) Reward of action replacement – PPO.

(b) Intervention rate of action replacement – PPO.

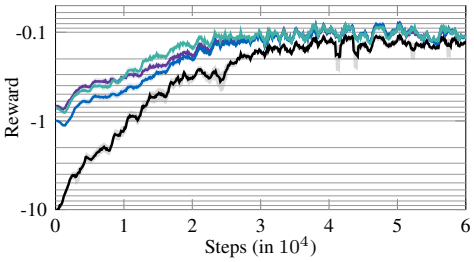
Figure 5: Evaluation of the training tuples for the 2D quadrotor averaged over the continuous and discrete PPO training runs using action replacement with ten random seeds each. The left column depicts the reward and the right column the safety intervention rate.



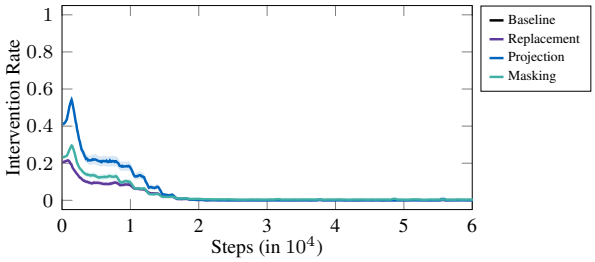
(a) Reward of baselines.



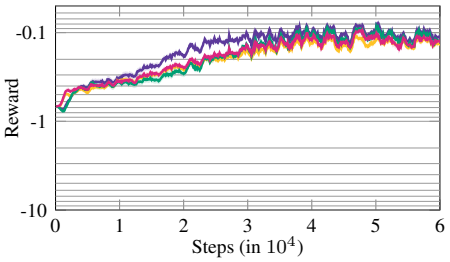
(b) Safety violation rate of baselines.



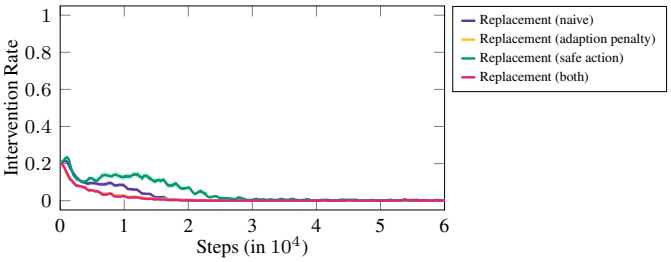
(c) Reward of provably safe RL algorithms.



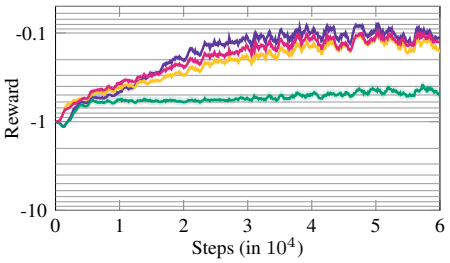
(d) Intervention rate of provably safe RL algorithms.



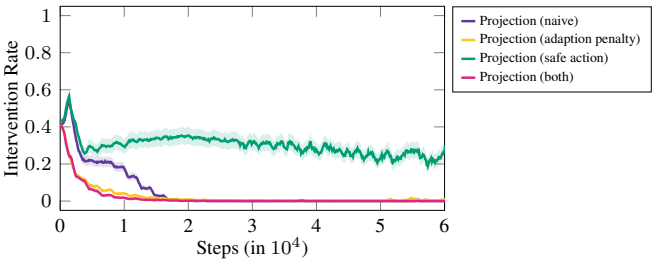
(e) Reward of action replacement.



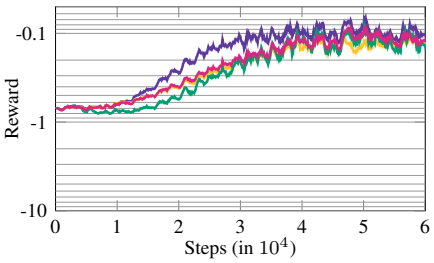
(f) Intervention rate of action replacement.



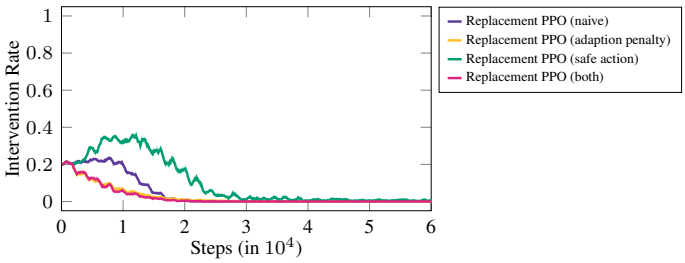
(g) Reward of action projection.



(h) Intervention rate of action projection.

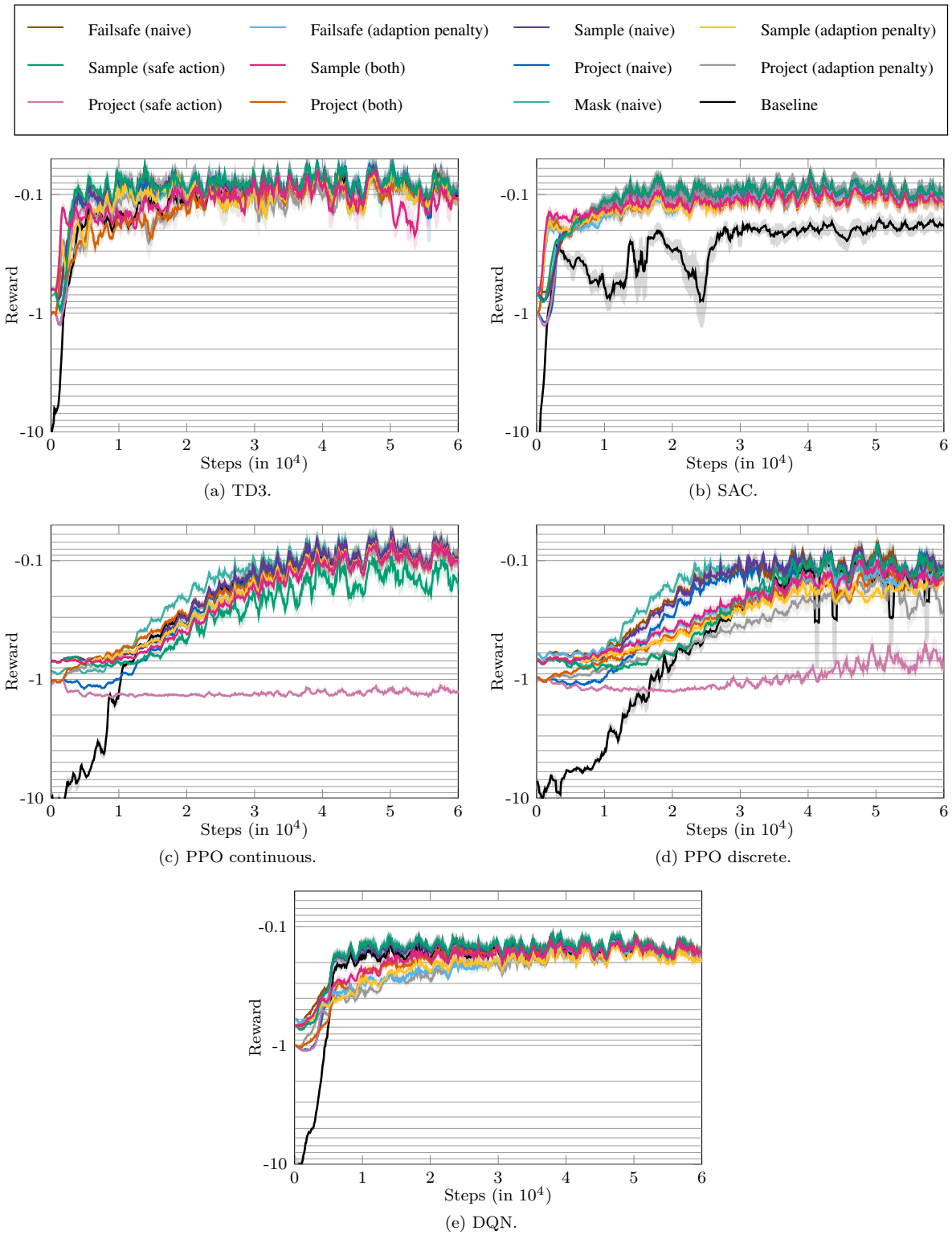


(i) Reward of action replacement – PPO.



(j) Intervention rate of action replacement – PPO.

Figure 6: Evaluation of the training tuples for the pendulum averaged over ten random seeds each. The left column depicts the reward and the right column the safety intervention rate. We would like to refer the reader to Figures 3, 4 and 5 for the corresponding 2D quadrotor results.



(a) TD3.

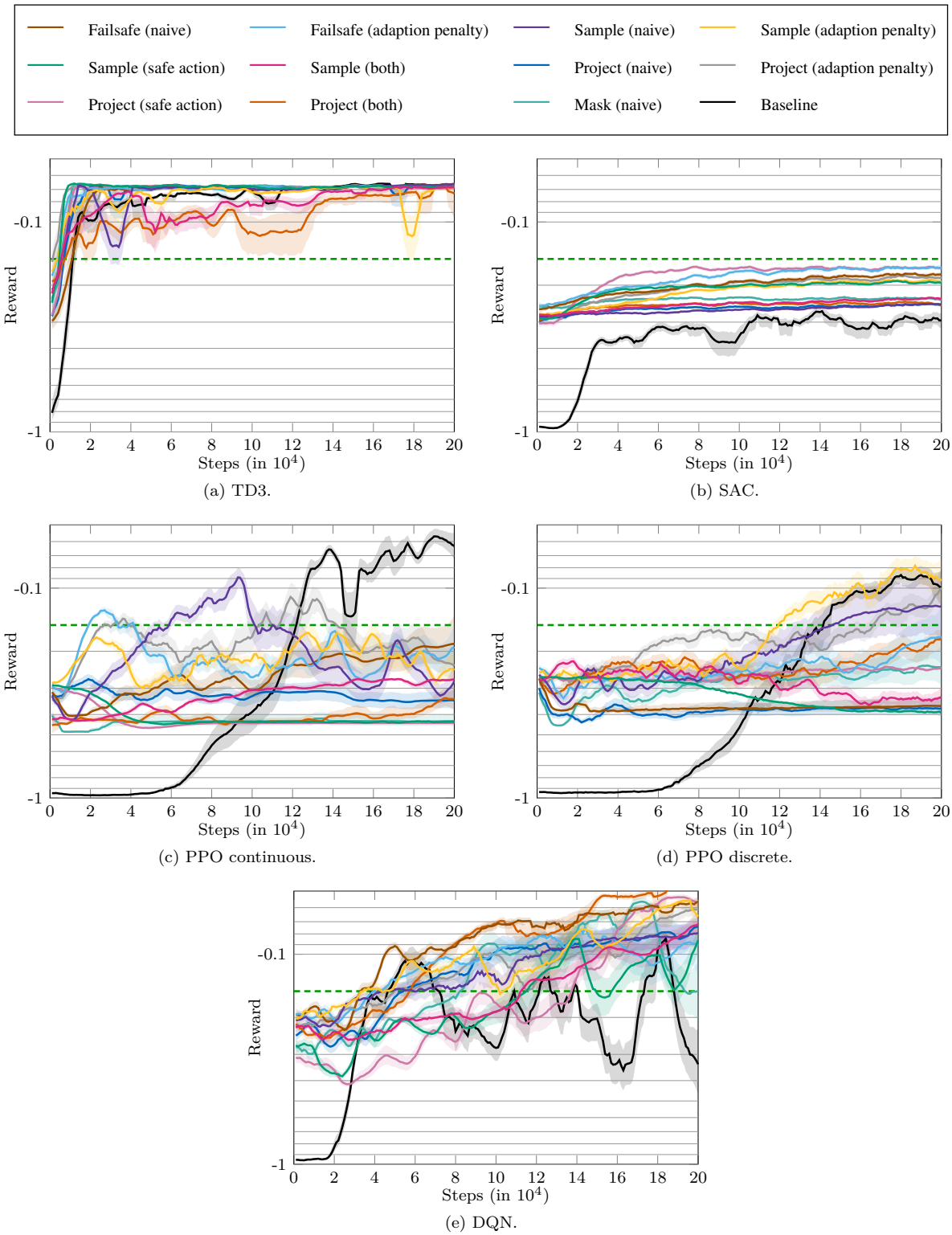
(b) SAC.

(c) PPO continuous.

(d) PPO discrete.

(e) DQN.

Figure 7: **Pendulum:** Average reward and standard deviation per training step for TD3, SAC, DQN, PPO discrete, and PPO continuous. For each configuration, ten training runs with different random seeds were conducted. Each subplot contains all implemented variants. Note that the reward for the *adaption penalty* variants is still r and the adaption penalty r^* is not included in the curves for better comparability.



(a) TD3.

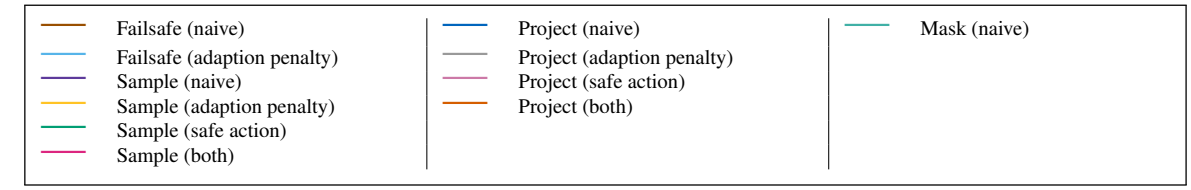
(b) SAC.

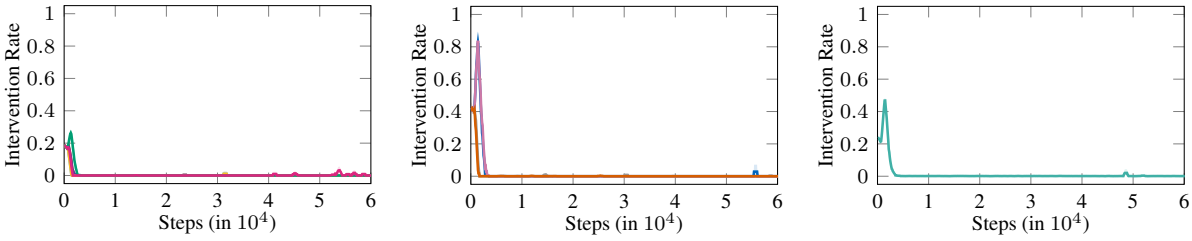
(c) PPO continuous.

(d) PPO discrete.

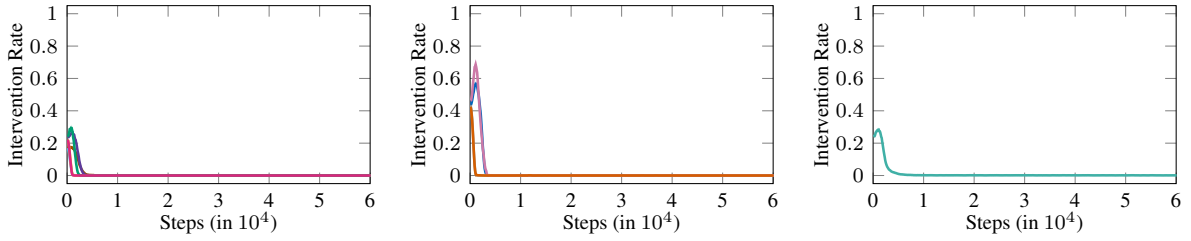
(e) DQN.

Figure 8: **2D quadrotor:** Average reward and standard deviation per training step for TD3, SAC, DQN, PPO discrete, and PPO continuous. For each configuration, ten training runs with different random seeds were conducted. Each subplot contains all implemented variants. Note that the reward for the *adaption penalty* variants is still r and the adaption penalty r^* is not included in the curves for better comparability.

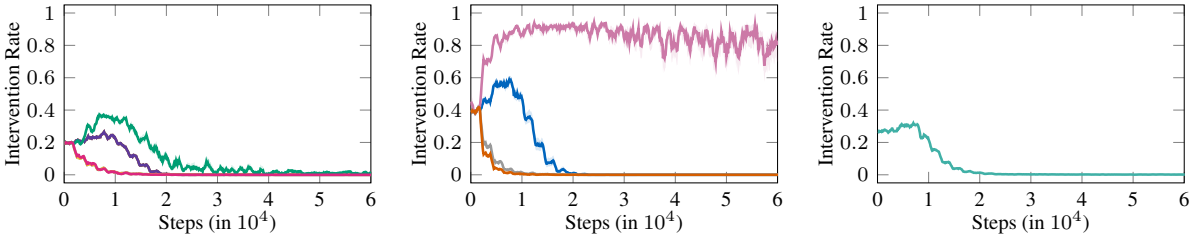




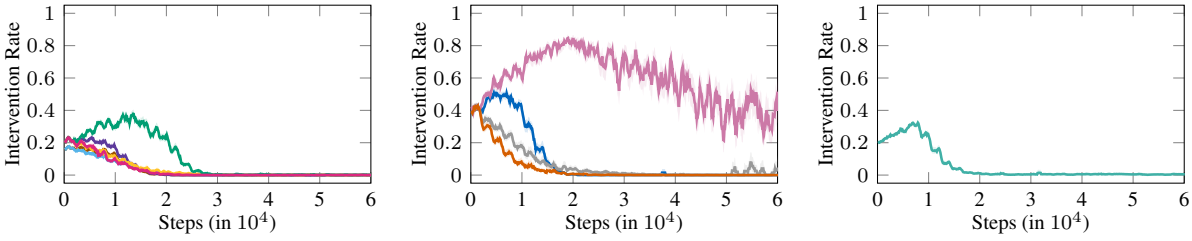
(a) Replacement TD3. (b) Projection TD3. (c) Masking TD3.



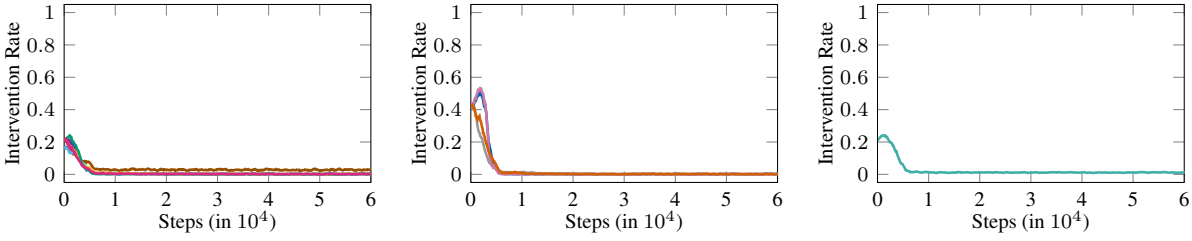
(d) Replacement SAC. (e) Projection SAC. (f) Masking SAC.



(g) Replacement PPO cont. (h) Projection PPO cont. (i) Masking PPO cont.

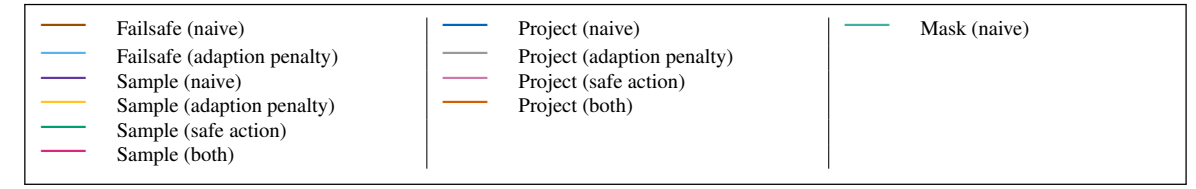


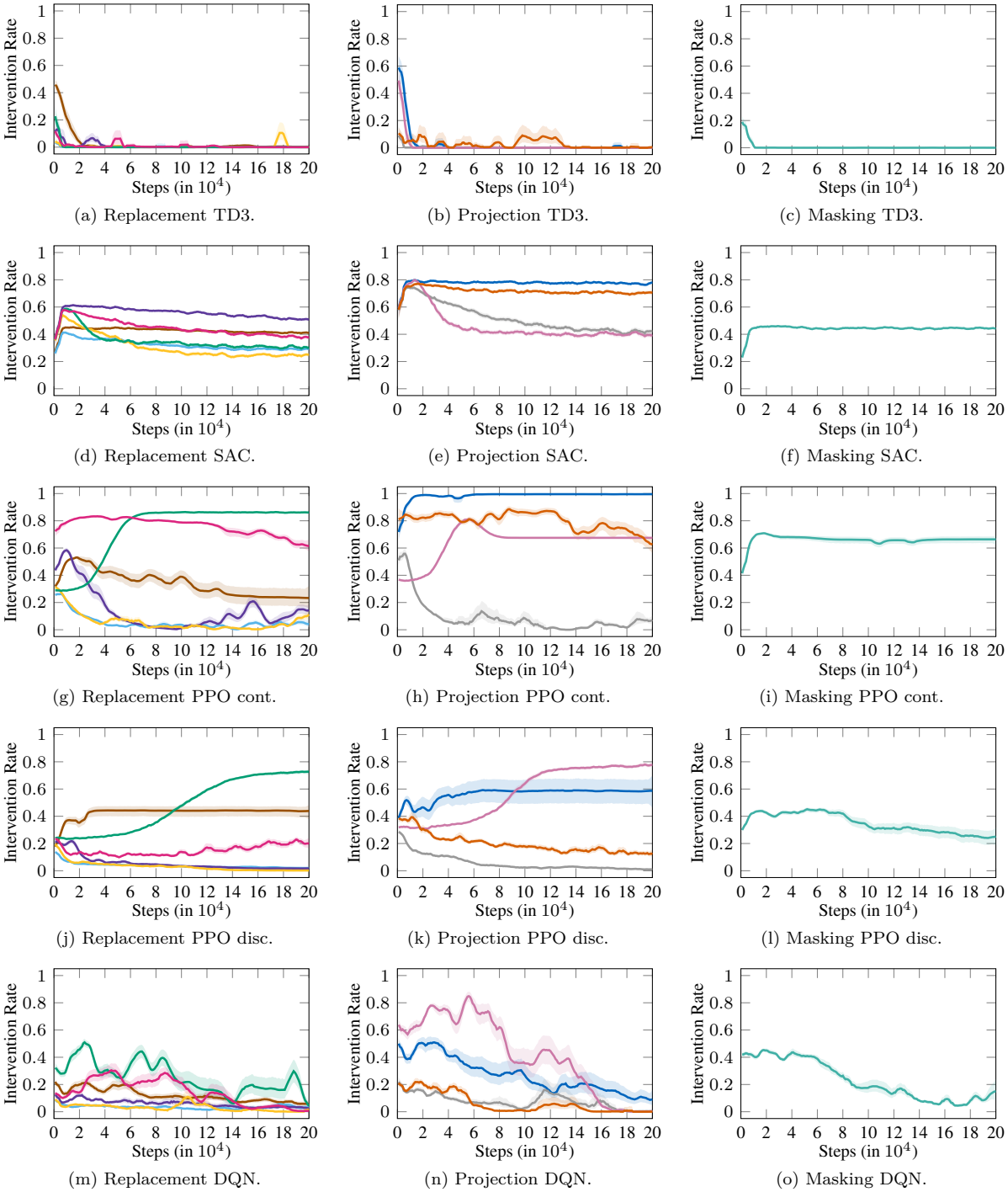
(j) Replacement PPO disc. (k) Projection PPO disc. (l) Masking PPO disc.



(m) Replacement DQN. (n) Projection DQN. (o) Masking DQN.

Figure 9: **Pendulum:** Intervention rate for TD3, SAC, PPO discrete, PPO continuous, and DQN.





(a) Replacement TD3. (b) Projection TD3. (c) Masking TD3.

(d) Replacement SAC. (e) Projection SAC. (f) Masking SAC.

(g) Replacement PPO cont. (h) Projection PPO cont. (i) Masking PPO cont.

(j) Replacement PPO disc. (k) Projection PPO disc. (l) Masking PPO disc.

(m) Replacement DQN. (n) Projection DQN. (o) Masking DQN.

Figure 10: **2D quadrotor:** Intervention rate for TD3, SAC, PPO discrete, PPO continuous. and DQN.

Table 9: Mean and standard deviation of 30 pendulum deployment episodes.

Approach	Reward		Intervention Rate		Safety Violation	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
PPO (continuous)						
Projection (SafeAction)	-1.14	0.42	0.76	0.29	0.00	0.00
Projection (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.00	0.00
Projection (Both)	-0.07	0.07	0.00	0.00	0.00	0.00
Projection (Naive)	-0.06	0.07	0.00	0.00	0.00	0.00
Sample (SafeAction)	-0.13	0.16	0.01	0.02	0.00	0.00
Sample (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.00	0.00
Sample (Both)	-0.07	0.07	0.00	0.00	0.00	0.00
Sample (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
FailSafe (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.00	0.00
FailSafe (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
Baseline (Naive)	-0.06	0.07	—	—	0.00	0.00
Masking (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
PPO (discrete)						
Projection (SafeAction)	-0.52	0.55	0.28	0.42	0.00	0.00
Projection (AdaptionPenalty)	-0.14	0.12	0.00	0.00	0.00	0.00
Projection (Both)	-0.09	0.09	0.00	0.00	0.00	0.00
Projection (Naive)	-0.15	0.27	0.03	0.10	0.00	0.00
Sample (SafeAction)	-0.09	0.08	0.00	0.00	0.00	0.00
Sample (AdaptionPenalty)	-0.13	0.16	0.00	0.00	0.00	0.00
Sample (Both)	-0.10	0.08	0.00	0.00	0.00	0.00
Sample (Naive)	-0.09	0.08	0.00	0.00	0.00	0.00
FailSafe (AdaptionPenalty)	-0.09	0.07	0.00	0.00	0.00	0.00
FailSafe (Naive)	-0.08	0.07	0.00	0.00	0.00	0.00
Baseline (Naive)	-0.08	0.07	—	—	0.00	0.00
Masking (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
TD3						
Projection (SafeAction)	-0.07	0.07	0.00	0.00	0.00	0.00
Projection (AdaptionPenalty)	-0.08	0.08	0.00	0.00	0.00	0.00
Projection (Both)	-0.07	0.07	0.00	0.00	0.00	0.00
Projection (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
Sample (SafeAction)	-0.07	0.07	0.00	0.00	0.00	0.00
Sample (AdaptionPenalty)	-0.09	0.07	0.00	0.00	0.00	0.00
Sample (Both)	-0.09	0.07	0.00	0.00	0.00	0.00
Sample (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
FailSafe (AdaptionPenalty)	-0.09	0.07	0.00	0.00	0.00	0.00
FailSafe (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
Baseline (Naive)	-0.07	0.07	—	—	0.00	0.00
Masking (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
DQN						
Projection (SafeAction)	-0.07	0.07	0.00	0.00	0.00	0.00
Projection (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.00	0.00
Projection (Both)	-0.07	0.08	0.00	0.00	0.00	0.00
Projection (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
Sample (SafeAction)	-0.07	0.07	0.00	0.00	0.00	0.00
Sample (AdaptionPenalty)	-0.09	0.07	0.00	0.00	0.00	0.00
Sample (Both)	-0.07	0.08	0.00	0.00	0.00	0.00
Sample (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
FailSafe (AdaptionPenalty)	-0.07	0.07	0.00	0.00	0.00	0.00
FailSafe (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
Baseline (Naive)	-0.07	0.07	—	—	0.00	0.00
Masking (Naive)	-0.07	0.07	0.00	0.00	0.00	0.00
SAC						
Projection (Naive)	-0.08	0.07	0.00	0.00	0.00	0.00
Projection (AdaptionPenalty)	-0.10	0.09	0.00	0.00	0.00	0.00
Projection (SafeAction)	-0.08	0.07	0.00	0.00	0.00	0.00
Projection (Both)	-0.10	0.08	0.00	0.00	0.00	0.00
Sample (Naive)	-0.08	0.07	0.00	0.00	0.00	0.00
Sample (AdaptionPenalty)	-0.10	0.08	0.00	0.00	0.00	0.00
Sample (SafeAction)	-0.08	0.07	0.00	0.00	0.00	0.00
Sample (Both)	-0.09	0.08	0.00	0.00	0.00	0.00
FailSafe (Naive)	-0.08	0.07	0.00	0.00	0.00	0.00
FailSafe (AdaptionPenalty)	-0.09	0.09	0.00	0.00	0.00	0.00
Baseline (Naive)	-0.11	0.09	—	—	0.00	0.00
Masking (Naive)	-0.08	0.07	0.00	0.00	0.00	0.00

Note: — indicates that there is no intervention rate for the baselines as they don't implement a safety verification.

Table 10: Mean and standard deviation of 30 2D Quadrotor deployment episodes.

Approach	Reward		Intervention Rate		Safety Violation	
	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
PPO (continuous)						
Projection (SafeAction)	-0.44	0.00	0.68	0.00	0.00	0.00
Projection (AdaptionPenalty)	-0.33	0.07	0.44	0.05	0.00	0.00
Projection (Both)	-0.39	0.07	0.57	0.13	0.00	0.00
Projection (Naive)	-0.31	0.10	0.47	0.25	0.00	0.00
Sample (SafeAction)	-0.43	0.00	0.86	0.00	0.00	0.00
Sample (AdaptionPenalty)	-0.28	0.12	0.10	0.12	0.00	0.00
Sample (Both)	-0.36	0.03	0.41	0.20	0.00	0.00
Sample (Naive)	-0.39	0.06	0.23	0.13	0.00	0.00
FailSafe (AdaptionPenalty)	-0.31	0.11	0.08	0.06	0.00	0.00
FailSafe (Naive)	-0.27	0.11	0.27	0.29	0.00	0.00
Baseline (Naive)	-0.86	0.01	—	—	0.94	0.01
Masking (Naive)	-0.43	0.09	0.57	0.28	0.00	0.00
PPO (discrete)						
Projection (SafeAction)	-0.44	0.01	0.74	0.21	0.00	0.00
Projection (AdaptionPenalty)	-0.24	0.13	0.02	0.02	0.00	0.00
Projection (Both)	-0.35	0.11	0.16	0.14	0.00	0.00
Projection (Naive)	-0.34	0.14	0.46	0.37	0.00	0.00
Sample (SafeAction)	-0.42	0.02	0.82	0.01	0.00	0.00
Sample (AdaptionPenalty)	-0.11	0.10	0.00	0.00	0.00	0.00
Sample (Both)	-0.38	0.09	0.32	0.18	0.00	0.00
Sample (Naive)	-0.13	0.16	0.02	0.03	0.00	0.00
FailSafe (AdaptionPenalty)	-0.19	0.15	0.01	0.03	0.00	0.00
FailSafe (Naive)	-0.34	0.13	0.41	0.21	0.00	0.00
Baseline (Naive)	-0.11	0.03	—	—	0.00	0.00
Masking (Naive)	-0.25	0.14	0.28	0.21	0.00	0.00
TD3						
Projection (SafeAction)	-0.21	0.03	0.28	0.10	0.00	0.00
Projection (AdaptionPenalty)	-0.22	0.03	0.26	0.10	0.00	0.00
Projection (Both)	-0.21	0.04	0.28	0.12	0.00	0.00
Projection (Naive)	-0.25	0.05	0.26	0.17	0.00	0.00
Sample (SafeAction)	-0.18	0.03	0.07	0.01	0.00	0.00
Sample (AdaptionPenalty)	-0.21	0.04	0.13	0.05	0.00	0.00
Sample (Both)	-0.21	0.06	0.06	0.03	0.00	0.00
Sample (Naive)	-0.19	0.04	0.12	0.09	0.00	0.00
FailSafe (AdaptionPenalty)	-0.20	0.03	0.05	0.02	0.00	0.00
FailSafe (Naive)	-0.26	0.09	0.05	0.02	0.00	0.00
Baseline (Naive)	-0.90	0.05	—	—	0.95	0.02
Masking (Naive)	-0.16	0.03	0.03	0.03	0.00	0.00
DQN						
Projection (SafeAction)	-0.06	0.02	0.00	0.01	0.00	0.00
Projection (AdaptionPenalty)	-0.05	0.00	0.00	0.00	0.00	0.00
Projection (Both)	-0.05	0.01	0.00	0.00	0.00	0.00
Projection (Naive)	-0.09	0.06	0.12	0.24	0.00	0.00
Sample (SafeAction)	-0.07	0.01	0.01	0.02	0.00	0.00
Sample (AdaptionPenalty)	-0.06	0.03	0.00	0.00	0.00	0.00
Sample (Both)	-0.07	0.02	0.00	0.00	0.00	0.00
Sample (Naive)	-0.05	0.01	0.00	0.00	0.00	0.00
FailSafe (AdaptionPenalty)	-0.06	0.02	0.02	0.04	0.00	0.00
FailSafe (Naive)	-0.07	0.03	0.10	0.10	0.00	0.00
Baseline (Naive)	-0.24	0.37	—	—	0.20	0.40
Masking (Naive)	-0.15	0.16	0.14	0.26	0.00	0.00
SAC						
Projection (Naive)	-0.20	0.01	0.52	0.02	0.00	0.00
Projection (AdaptionPenalty)	-0.19	0.00	0.49	0.01	0.00	0.00
Projection (SafeAction)	-0.19	0.00	0.49	0.01	0.00	0.00
Projection (Both)	-0.20	0.01	0.49	0.01	0.00	0.00
Sample (Naive)	-0.15	0.01	0.05	0.00	0.00	0.00
Sample (AdaptionPenalty)	-0.15	0.01	0.05	0.00	0.00	0.00
Sample (SafeAction)	-0.15	0.01	0.05	0.00	0.00	0.00
Sample (Both)	-0.16	0.02	0.05	0.00	0.00	0.00
FailSafe (Naive)	-0.21	0.01	0.17	0.02	0.00	0.00
FailSafe (AdaptionPenalty)	-0.17	0.01	0.04	0.00	0.00	0.00
Baseline (Naive)	-0.88	0.02	—	—	0.96	0.03
Masking (Naive)	-0.14	0.02	0.00	0.00	0.00	0.00

Note: — indicates that there is no intervention rate for the baselines as they don't implement a safety verification.

CC BY 4.0 DEED

Attribution 4.0 International

Canonical URL : https://creativecommons.org/licenses/by/4.0/

See the legal code

You are free to:

Share — copy and redistribute the material in any medium or format for any purpose, even commercially.

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation .

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Notice

This deed highlights only some of the key features and terms of the actual license. It is not a license and has no legal value. You should carefully review all of the terms and conditions of the actual license before using the licensed material.

Creative Commons is not a law firm and does not provide legal services. Distributing, displaying, or linking to this deed or the license that it summarizes does not create a lawyer-client or any other relationship.

Creative Commons is the nonprofit behind the open licenses and other legal tools that allow creators to share their work. Our legal tools are free to use.

- Learn more about our work
- **Learn more about CC Licensing**
- Support our work
- Use the license for your own material.
- Licenses List
- Public Domain List

Footnotes

appropriate credit — If supplied, you must provide the name of the creator and attribution parties, a copyright notice, a license notice, a disclaimer notice, and a link to the material. CC licenses prior to Version 4.0 also require you to provide the title of the material if supplied, and may have other slight differences.
- More info

indicate if changes were made — In 4.0, you must indicate if you modified the material and retain an indication of previous modifications. In 3.0 and earlier license versions, the indication of changes is only required if you create a derivative.
- Marking guide
- More info

technological measures — The license prohibits application of effective technological measures, defined with reference to Article 11 of the WIPO Copyright Treaty.
- More info

exception or limitation — The rights of users under exceptions and limitations, such as fair use and fair dealing, are not affected by the CC licenses.
- More info

publicity, privacy, or moral rights — You may need to get additional permissions before using the material as you intend.
- More info

A.2 Provably Safe Reinforcement Learning via Action Projection using Reachability Analysis and Polynomial Zonotopes

Summary Pre-computing a time-invariant set of safe states is efficient for online verification of safe actions. However, if dynamic obstacles are present in the environment, unsafe states are time-varying. Thus, the set of safe states is time-varying as well. This safe set is often indirectly computed by predicting the set of unsafe states online and ensuring that the RL agent never enters unsafe states. While time-varying unsafe states are challenging for other online verification approaches like control barrier functions, set-based reachability analysis unobstructedly integrates time-varying unsafe sets.

This article proposes an action projection approach based on set-based reachability analysis for CPSs operating in environments with static and dynamic obstacles. Our approach can be applied to CPSs with control input constraints for continuous action spaces and bounded uncertainties in the nonlinear system model. The safety specification is verified if potentially time-varying unsafe state sets, represented by polytopes, are avoided at all times. To this end, a mixed-integer quadratic program is derived, which projects unsafe actions to safe actions. Additionally, we propose several extensions to improve the computational runtime of the optimization problem and provide solutions to incorporate spatial extensions of the CPS or complex control laws.

We perform extensive numerical experiments and show that our action projection approach always fulfills the safety specification. In particular, we demonstrate the real-time capability on the F1TENTH car, the ease of integrating time-varying unsafe sets on an autonomous driving task, the benefits of adding the action projection already during training on a two-dimensional quadrotor, and the scalability to high-dimensional systems on a three-dimensional quadrotor.

Author contributions N.K., H.K., and X.W. developed the concept for provably safe reinforcement learning in continuous action spaces. N.K. implemented the mixed-integer quadratic program for projecting actions. The implementation and evaluation for the different benchmark systems was distributed as follows: N.K. and X.W. - F1TENTH car, X.W. - autonomous driving, N.K. - three-dimensional quadrotor, and H.K. - two-dimensional quadrotor. N.K, H.K., and X.W. structured the presentation of the manuscript and wrote the manuscript. S.B. and M.A. provided feedback on the concept and helped improving the manuscript.

TUM Graduate School This publication is *not* a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

IEEE CSS

Received 17 October 2022; revised 24 January 2023; accepted 23 February 2023. Date of publication 13 March 2023; date of current version 31 March 2023. Recommended by Senior Editor Lacra Pavel.

Digital Object Identifier 10.1109/OJCSYS.2023.3256305

Provably Safe Reinforcement Learning via Action Projection Using Reachability Analysis and Polynomial Zonotopes

NIKLAS KOCHDUMPER [1,2], HANNA KRASOWSKI [1], XIAO WANG [1], STANLEY BAK [2], AND MATTHIAS ALTHOFF [1]

[1]Department of Computer Engineering, Technical University of Munich, 85748 Garching, Germany
[2]Department of Computer Science, Stony Brook University, Stony Brook, NY 11794 USA

CORRESPONDING AUTHORS: NIKLAS KOCHDUMPER; HANNA KRASOWSKI; XIAO WANG (e-mail: niklas.kochdumper@stonybrook.edu; hanna.krasowski@tum.de; xiao.wang@tum.de)

This work was supported by the European Research Council (ERC) through the Project justITSELF under Grant 817629, in part by the German Research Foundation through the Research Training Group ConVeY under Grant GRK 2428, and in part by the Air Force Office of Scientific Research and the Office of Naval Research under Grants FA9550-19-1-0288, FA9550-21-1-0121, FA9550-23-1-0066, and N00014-22-1-2156. *(Niklas Kochdumper, Hanna Krasowski, and Xiao Wang contributed equally to this work.)*

This article has supplementary downloadable material available at https://doi.org/10.1109/OJCSYS.2023.3256305, provided by the authors.

ABSTRACT While reinforcement learning produces very promising results for many applications, its main disadvantage is the lack of safety guarantees, which prevents its use in safety-critical systems. In this work, we address this issue by a safety shield for nonlinear continuous systems that solve reach-avoid tasks. Our safety shield prevents applying potentially unsafe actions from a reinforcement learning agent by projecting the proposed action to the closest safe action. This approach is called action projection and is implemented via mixed-integer optimization. The safety constraints for action projection are obtained by applying parameterized reachability analysis using polynomial zonotopes, which enables to accurately capture the nonlinear effects of the actions on the system. In contrast to other state-of-the-art approaches for action projection, our safety shield can efficiently handle input constraints and dynamic obstacles, eases incorporation of the spatial robot dimensions into the safety constraints, guarantees robust safety despite process noise and measurement errors, and is well suited for high-dimensional systems, as we demonstrate on several challenging benchmark systems.

INDEX TERMS Action projection, reach-avoid problems, reachability analysis, reinforcement learning.

I. INTRODUCTION

Reinforcement learning has been successfully applied to find solutions for many challenging applications, such as robotics [1], autonomous driving [2], and power systems [3]. Many of these applications are safety-critical, so that the lack of safety guarantees for standard reinforcement learning controllers prevents their deployment in the real world. We aim to overcome this limitation with a novel safety shield for reinforcement learning agents that considers the very general case of disturbed nonlinear continuous systems with input constraints that have to avoid dynamic obstacles. Note that our safety shield can be applied to arbitrary unsafe controllers, while reinforcement learning is the main focus of this work.

A. STATE OF THE ART

We first provide a summary of the current state of the art in safety-related methods of reinforcement learning. The term *safe reinforcement learning* refers to approaches that aim to obtain safe agents, but do not provide hard safety guarantees. One example for this is constrained reinforcement learning [4], [5], where the objective of the training phase is to maximize the reward while satisfying safety constraints. While advantages of this technique are that no system model is required and that even complex temporal logic safety specifications [6], [7] can be considered, the obvious disadvantage is that hard safety guarantees can be provided during neither training nor deployment. The same is true for probabilistic

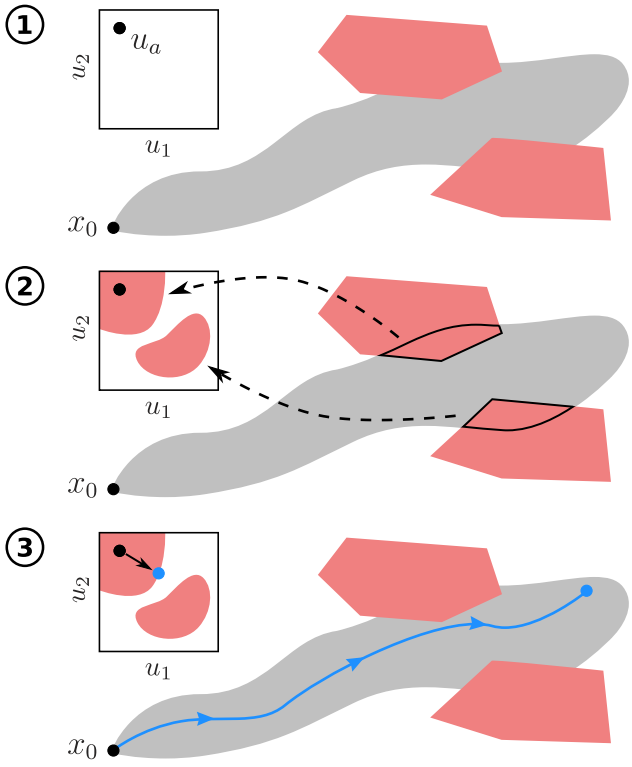


FIGURE 1. Steps for action projection using parameterized reachability analysis, where the reachable set is depicted in gray and the unsafe regions are shown in red: 1) Computation of the reachable set for all actions starting from the current state x_0. 2) Extraction of action constraints from the intersections between the reachable set and unsafe regions. 3) Projection of the action u_a outputted by the agent to the closest safe action.

approaches [8], [9] that aim to identify the safety probability of an action. Overall, safe reinforcement learning techniques can be used for non-critical applications, where unsafe actions do not cause major damage; however, these methods are not suited for safety-critical systems.

In contrast to safe reinforcement learning, *provably safe reinforcement learning* approaches provide hard safety guarantees. They can be divided into the three main categories: *action masking*, *action replacement*, and *action projection* [10]. In action masking [11], [12], a mask that only allows the agent to choose actions from the set of safe actions is applied. One disadvantage of this method is that it is often hard to explicitly compute the set of safe actions, especially for continuous action spaces, where the set of safe actions often has a very complex non-convex shape as shown in Fig. 1. In addition, it is non-trivial to correctly consider the masking during training so that the reinforcement learning algorithm is not perturbed [13]. For action replacement [14], [15], [16], [17], unsafe actions returned by the agent are replaced by safe actions. As replacement, one can either use a single safe action obtained from a failsafe planner [14] or via human feedback [15] or one can sample from the set of safe actions [16], [17]. Also the well-known simplex architecture [18], [19], [20], where a safe controller is used as a backup for an unsafe controller, can be categorized as action replacement. One disadvantage of action replacement is that the difference between the original action and the replacement action can be very large, which might prevent the agent from completing its task. Action projection tries to avoid this issue by finding the safe action that is closest to the action suggested by the agent.

Since our approach applies action projection, we discuss this category in more detail. The most prominent methods for action projection are *control barrier functions* [21], [22], *model predictive control* [23], [24], and *parameterized reachability analysis* [25]. A control barrier function is a level-set function that divides the state space into a safe and a potentially unsafe region. Here, action projection is formulated as an optimization problem, where the correction of the action is minimized, such that the system stays inside the safe region defined by the control barrier function. While an advantage is that control barrier functions can for static environments guarantee safety for infinite time, the method also has several disadvantages: 1) It is often not easy to find a suitable control barrier function, especially in the presence of dynamic obstacles. 2) Control barrier functions are often quite conservative since they usually exclude many states that are safe. 3) The approach is often limited to control affine systems because the optimization problems would otherwise become non-convex. 4) It is challenging to consider input constraints as well as process noise and measurement errors. The second method is model predictive control, which also formulates the projection as an optimization problem, but uses the safety constraint that the system should not enter any unsafe regions for a certain finite prediction horizon, which avoids the requirement for a control barrier function. However, one downside is that it is often not possible to guarantee that the solution is robustly safe despite process noise and measurement errors since for nonlinear systems these uncertainties usually cannot be encoded directly into the optimization problem. Our safety shield is based on the parameterized reachability analysis approach, which is visualized in Fig. 1: The first step is to compute the reachable set for all available actions. Since this reachable set is parameterized by the actions, one can directly extract the safety constraints for action projection from the intersection between the reachable sets and the unsafe regions. Since process noise as well as measurement errors can conveniently be integrated into reachability analysis, this approach is very well suited for guaranteeing robust safety.

Due to its advantageous properties, several approaches apply reachability analysis to guarantee safety. One method [26] uses the Hamilton-Jacobi reachability framework [27] to compute the backward reachable set starting from the unsafe sets — a state is safe for all possible actions if it is outside of the backward reachable set. This has the disadvantage that for each unsafe set a different backward reachable set has to be computed. Moreover, the Hamilton-Jacobi framework requires gridding the state space so that the computational complexity of the approach grows exponentially with the system dimension. Another method [28] applies reachability analysis for black-box systems and uses a differentiable

collision check that is based on constrained zonotopes [29] to efficiently push the reachable set for the proposed action away from unsafe sets. This, however, has the drawback that the reachable set has to be recomputed after each correction update of the action, which is computationally demanding. The method closest to our approach is a *reachability-based trajectory safeguard* [25], which computes the parameterized reachable set for a simplified trajectory-generating model and determines a safe action satisfying the constraints extracted from the reachable set via random sampling. While this approach can be computationally efficient for some systems, sampling methods often fail to find feasible solutions, especially in high-dimensional action spaces.

B. CONTRIBUTIONS AND OUTLINE

We present a novel safety shield that is based on action projection using parameterized reachability analysis. This safety shield extends our previous work on dependency preserving reachability analysis [30], [31], [32] by a method for correcting unsafe actions, and we additionally also study the effect online verification has on the learning process. Unlike the related approach in [25], our safety shield directly operates on the original nonlinear system model rather than on a simplified trajectory-generating model. Moreover, in contrast to [25], we use conservative polynomialization [30] instead of conservative linearization [33] for reachability analysis, which enables us to efficiently capture the nonlinear effects the actions have on the system. Another advantage over [25] is that we use mixed-integer optimization instead of random sampling for projection, which always finds the action with the smallest correction. Finally, the various design choices provided by our safety shield enable the user to fine-tune its performance for the considered application.

The remainder of this paper is structured as follows: After introducing some preliminaries in Section II, we provide the problem definition in Section III. Our main contribution is the reachability-based safety shield for reinforcement learning presented in Section IV, for which we discuss several extensions in Section V. Finally, we demonstrate our approach on several numerical examples in Section VI and conclude with a discussion of its properties in Section VII.

II. PRELIMINARIES

We first introduce our notation and define the set representations that we use in this paper.

A. NOTATION

Sets are denoted by calligraphic letters, matrices by uppercase letters, and vectors by lowercase letters. Given a vector $a \in \mathbb{R}^n$, $a_{(i)}$ is the i-th entry and the p-norm is denoted by $\|a\|_p$. Given a matrix $A \in \mathbb{R}^{n\times m}$, $A_{(i,\cdot)}$ represents the i-th matrix row, $A_{(\cdot,j)}$ the j-th column, and $A_{(i,j)}$ the j-th entry of matrix row i. The concatenation of two matrices C and D is denoted by $[C\,D]$, $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix, and the symbols $\mathbf{0}$ and $\mathbf{1}$ represent vectors of zeros and ones of proper dimension. We further introduce an n-dimensional interval as $\mathcal{I} := [\underline{x}, \overline{x}]$, $\forall i\, \underline{x}_{(i)} \leq \overline{x}_{(i)}$, $\underline{x}, \overline{x} \in \mathbb{R}^n$. Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$, their Minkowski sum is $\mathcal{S}_1 \oplus \mathcal{S}_2 = \{s_1 + s_2 \,|\, s_1 \in \mathcal{S}_1,\ s_2 \in \mathcal{S}_2\}$ and their Cartesian product is $\mathcal{S}_1 \times \mathcal{S}_2 = \{[s_1^T\ s_2^T]^T \,\big|\, s_1 \in \mathcal{S}_1,\ s_2 \in \mathcal{S}_2\}$.

B. SET REPRESENTATIONS

Our approach relies on several different set representations, which we introduce here. Let us begin with polytopes, for which we use the halfspace representation:

Definition 1 (Polytope): Given a constraint matrix $A \in \mathbb{R}^{s\times n}$ and a constraint offset $b \in \mathbb{R}^s$, the halfspace representation of a polytope $\mathcal{P} \subseteq \mathbb{R}^n$ is

$$\mathcal{P} := \left\{ x \in \mathbb{R}^n \,\middle|\, A\,x \leq b \right\}.$$

We use the shorthand $\mathcal{P} = \langle A, b \rangle_P$.

Zonotopes are a special type of polytopes that can be represented efficiently using generators:

Definition 2 (Zonotope): Given a center vector $c \in \mathbb{R}^n$ and a generator matrix $G \in \mathbb{R}^{n\times p}$, a zonotope $\mathcal{Z} \subset \mathbb{R}^n$ is

$$\mathcal{Z} := \left\{ c + \sum_{i=1}^{p} G_{(\cdot,i)}\,\alpha_i \,\middle|\, \alpha_i \in [-1, 1] \right\}$$

with so-called factors α_i. We use the shorthand $\mathcal{Z} = \langle c, G \rangle_Z$.

An extension to zonotopes are polynomial zonotopes [30], which can represent non-convex sets. We use the sparse representation of polynomial zonotopes [32]:[1]

Definition 3 (Polynomial Zonotope): Given a constant offset $c \in \mathbb{R}^n$, a generator matrix of dependent generators $G \in \mathbb{R}^{n\times h}$, a generator matrix of independent generators $G_I \in \mathbb{R}^{n\times q}$, and an exponent matrix $E \in \mathbb{N}_0^{p\times h}$, a polynomial zonotope $\mathcal{PZ} \subset \mathbb{R}^n$ is

$$\mathcal{PZ} := \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \,\middle|\, \alpha_k, \beta_j \in [-1, 1] \right\}.$$

The scalars α_k are called dependent factors and β_j independent factors. We use the shorthand $\mathcal{PZ} = \langle c, G, G_I, E \rangle_{PZ}$.

Polynomial zonotopes can equivalently represent intervals, zonotopes, polytopes, and Taylor models [32, Sec. II.B]. Moreover, due to their polynomial nature, they are closely related to polynomial level sets:

Definition 4 (Polynomial Level Set): Given a vector of coefficients $a \in \mathbb{R}^h$, an offset $b \in \mathbb{R}$, and an exponent matrix $E \in \mathbb{N}_0^{n\times h}$, a polynomial level set $\mathcal{LS} \subseteq \mathbb{R}^n$ is

$$\mathcal{LS} := \left\{ x \in \mathbb{R}^n \,\middle|\, \sum_{i=1}^{h} \left(\prod_{k=1}^{n} x_{(k)}^{E_{(k,i)}} \right) a_{(i)} \leq b \right\}.$$

We use the shorthand $\mathcal{LS} = \langle a, b, E \rangle_{LS}$.

[1] In contrast to [32, Def. 1] we do not integrate the constant offset c into G. Moreover, we omit the identifier vector used in [32] for simplicity

III. PROBLEM FORMULATION

We consider general nonlinear disturbed systems with input constraints defined by the ordinary differential equation

$$\dot{x}(t) = f\left(x(t), u(t), w(t)\right), \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the system state, $u(t) \in \mathbb{R}^m$ is the control input, $w(t) \in \mathbb{R}^z$ is the process noise, $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^z \to \mathbb{R}^n$ is a Lipschitz continuous function, and $t \in \mathbb{R}^+$ is time. The process noise is bounded by a compact set $w(t) \in \mathcal{W} \subset \mathbb{R}^z$ and the system has to satisfy the input constraints defined by the convex set $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$. The set $\mathcal{W}$ can for example be determined from measurements of the real physical system using conformance checking [34].

Given a nonlinear system defined as in (1), the goal is to solve a reach-avoid problem, where the system state should be steered from the current state $x_0 = x(0)$ to a goal set $\mathcal{G} \subseteq \mathbb{R}^n$ while avoiding collisions with potentially time-varying unsafe sets $\mathcal{F}_i \subset \mathbb{R}^n$, $i = 1, \ldots, o$, where o denotes the number of unsafe sets. In case the measurements of the system state are subject to a measurement error $v(t) \in \mathcal{V}$, the goal becomes to steer all states in the set $x_0 \oplus \mathcal{V}$ to the goal set. We aim to solve reach-avoid problems with reinforcement learning, where we train an agent to return the control inputs $u_a(t)$ for a given state $x(t)$ steering the system to the goal set while avoiding obstacles. However, we have no guarantee that the behavior learned by the agent is safe. Therefore, we add a safety shield that is based on reachability analysis to obtain formal guarantees:

Definition 5 (Reachable Set): Let $\xi(t, x_0, u(\cdot), w(\cdot))$ denote the solution of (1) at time t for an initial state $x_0 = x(0)$, control input trajectory $u(\cdot)$ and process noise trajectory $w(\cdot)$. The reachable set at time t is

$$\mathcal{R}(t) := \left\{ \xi\left(t, x_0, u(\cdot), w(\cdot)\right) \,\middle|\, x_0 \in \mathcal{X}_0, \; \forall \tau \in [0, t] : \; w(\tau) \in \mathcal{W} \right\},$$

where $\mathcal{X}_0 \subset \mathbb{R}^n$ is the initial set and $\mathcal{W} \subset \mathbb{R}^z$ is the set of process noise.

For our safety shield, we consider that $\mathcal{U}$, $\mathcal{W}$, and $\mathcal{V}$ are represented as zonotopes, and $\mathcal{G}$ and $\mathcal{F}_i$ are represented as polytopes in halfspace representation. Moreover, we use polynomial zonotopes to represent reachable sets. In case other agents are present in the environment, we can apply set-based methods [35] to safely predict their future behavior and obtain the corresponding time-varying unsafe sets.

IV. SAFETY SHIELD

As visualized in Fig. 1, the high-level idea behind our safety shield is to compute the reachable set for a time horizon of t_f and the set of all control inputs satisfying the input constraints $\forall t \in [0, t_f] : \; u(t) \in \mathcal{U}$ rather than a single control input trajectory $u(\cdot)$. The intersection of this reachable set with the unsafe sets then yields constraints that define safe control inputs, which we can use to formulate the projection of the control input u_a provided by the reinforcement policy to the closest safe control input as an optimization problem. We first consider input trajectories that are constant over time for simplicity and discuss more advanced control strategies later in Section V-A. For constant control inputs, we can compute the reachable set using the extended system dynamics

$$\begin{bmatrix} \dot{x}(t) \\ \dot{u}(t) \end{bmatrix} = \begin{bmatrix} f\left(x(t), u(t), w(t)\right) \\ \mathbf{0} \end{bmatrix} \tag{2}$$

together with the initial set $\mathcal{X}_0 = x_0 \times \mathcal{U}$, where we omit the set of measurement errors $\mathcal{V}$ for simplicity. For reachability analysis, we use the conservative polynomialization algorithm [30], which encloses the nonlinear dynamics in (1) by a differential inclusion $\dot{x} \in p(x(t), u(t), w(t)) \oplus \mathcal{E}$ consisting of a polynomial approximation $p(x(t), u(t), w(t))$ and the abstraction error $\mathcal{E}$. This reachability algorithm explicitly preserves dependencies between the initial states and the reachable states [31]. Since with the extended system dynamics in (2), the control inputs become part of the system state, we can therefore directly determine from the reachable set which control inputs steer the system to unsafe regions. Let us demonstrate this dependency preservation by an example:

Example 1: As a running example we consider the system

$$\dot{x}_1 = 4 + 2\, x_2\, u_1 + w_1, \; \dot{x}_2 = 1.7 + u_1\, u_2$$

with initial state $x_0 = [0\,0]^T$, set of process noise $\mathcal{W} = [-0.01, 0.01]$, and time horizon $t_f = 1$ s. Moreover,

$$\mathcal{F} = \left\langle \left[[-4 \;\; -1]^T \; [-1 \;\; -4]^T \right], [-14 \;\; -8]^T \right\rangle_P$$

is the unsafe set and

$$\mathcal{U} = \left\{ \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \alpha_2 \,\middle|\, \alpha_1, \alpha_2 = [-1, 1] \right\}$$

is the set of control inputs. With the conservative polynomialization algorithm we obtain the final reachable set

$$\mathcal{R}(t_f) = \left\{ \begin{bmatrix} 3.4 \\ 1.2 \end{bmatrix} + \begin{bmatrix} 0.34 \\ 0.5 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0.25 \\ -0.5 \end{bmatrix} \alpha_2 + \begin{bmatrix} -0.49 \\ 0.5 \end{bmatrix} \alpha_1 \alpha_2 + \begin{bmatrix} 0.25 \\ 0 \end{bmatrix} \alpha_1^2 + \begin{bmatrix} 0.25 \\ 0 \end{bmatrix} \alpha_1^2 \alpha_2 + \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} \beta_1 \,\middle|\, \alpha_1, \alpha_2, \beta_1 \in [-1, 1] \right\},$$

which is visualized in Fig. 2 . Since the reachable set $\mathcal{R}(t_f)$ and the input set $\mathcal{U}$ are parameterized by the same factors α_1 and α_2, we have a direct analytical relation between the control inputs and the corresponding reachable states.

We now exploit the analytical relation between the control inputs and the reachable states to determine the set of safe control inputs. As demonstrated in the example above, the control input $u(t) \in \mathcal{U} = \langle c_u, G_u \rangle_Z$ is unambiguously defined by the factors α via the relation $u = c_u + G_u \alpha$ through the definition of a zonotope in Def. 2. Instead of determining the set of safe control inputs directly, we therefore determine the safe set for α instead, since this simplifies the computations as it becomes apparent later. The independent generators of

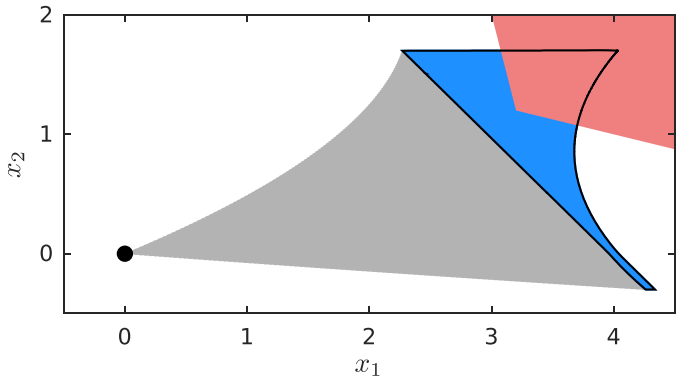


FIGURE 2. Reachable set for the system from Example 1, where the initial state x_0 is shown as a black dot, the final reachable set $\mathcal{R}(t_f)$ is depicted in blue with a black border, and the unsafe set $\mathcal{F}$ is shown in red.

the polynomial zonotope $\mathcal{R}(t_f)$ represent uncertainties that results from abstraction errors during reachability analysis as well as from the process noise. Consequently, a control input is safe only if the reachable set does not intersect the unsafe sets for any possible value of the independent factors β_j. We formulate this in the following theorem, which extends our previous results for unsafe sets given as halfspaces [31, Sec. 4.1] to the more general case of polytopes:

Theorem 1: Given is an unsafe set $\mathcal{F} = \langle A, b \rangle_P \subset \mathbb{R}^n$ consisting of s halfspace constraints and the reachable set $\mathcal{R}(t) = \langle c, G, G_I, E \rangle_{PZ} \subset \mathbb{R}^n$ of the system in (2) computed with the conservative polynomialization algorithm [30] for the initial set $\mathcal{X}_0 = x_0 \times \mathcal{U}$, $x_0 \in \mathbb{R}^n$, $\mathcal{U} = \langle c_u, G_u \rangle_Z \subset \mathbb{R}^m$ and the set of process noise $\mathcal{W} \subset \mathbb{R}^z$. The following constraints on the zonotope factors α that parameterize the control input ensure that there exists no trajectory that enters the unsafe set:

$$\forall \alpha \in [-\mathbf{1}, \mathbf{1}] \cap \bigcup_{l=1}^{s} \mathcal{LS}_l,\ \forall w(\cdot) \in \mathcal{W} : \xi(t, x_0, c_u + G_u\alpha, w(\cdot)) \notin \mathcal{F}$$

with

$$\mathcal{LS}_l = \left\langle -A_{(l,\cdot)}G, A_{(l,\cdot)}c - \sum_{j=1}^{q} \left|A_{(l,\cdot)}G_{I(\cdot,j)}\right| - b_{(l)}, E \right\rangle_{LS}$$

for $l = 1, \ldots, s$.

Proof: A single point $x \in \mathbb{R}^n$ is located outside the unsafe set $\mathcal{F}$ if it is fully located outside of at least one halfspace:

$$\bigvee_{l=1}^{s} A_{(l,\cdot)}x > b_{(l)} \Rightarrow x \notin \mathcal{F}. \tag{3}$$

Moreover, due to dependency preservation of reachability analysis, it holds according to [31, Thm. 1] that the disturbed trajectory $\xi(t, x_0, c_u + G_u\alpha, w(\cdot))$ for a specific control input $u = c_u + G_u\alpha$ is contained inside the reachable subset obtained by restricting the factors $\alpha_k \in [-1, 1]$ in the definition of polynomial zonotopes in Def. 3 to the corresponding concrete value for $\alpha = [\alpha_1 \ \ldots \ \alpha_p]^T$:

$$\forall \alpha_k \in [-1, 1] : \xi(t, x_0, c_u + G_u\alpha, w(\cdot)) \in c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \left\{ \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \,\middle|\, \beta_j \in [-1, 1] \right\}.$$

Finally, combining this with (3) under the consideration that the constraints should hold for all values of the independent factors β_j yields

$$\forall \alpha_k, \beta_j \in [-1, 1] : \bigvee_{l=1}^{s} A_{(l,\cdot)}c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) A_{(l,\cdot)}G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j A_{(l,\cdot)}G_{I(\cdot,j)} > b_{(l)} \Rightarrow \xi(t, x_0, c_u + G_u\alpha, w(\cdot)) \notin \mathcal{F},$$

which results in the statement of the theorem after bringing the constant offset and the independent generators to the other side of the inequality. ■

Remark 1: A geometric interpretation of Theorem 1 is that we first bloat the obstacle $\mathcal{F}$ by the uncertainty given by the independent generators through pushing each polytope halfspace outward. Next, we obtain the constraints via intersecting with the part of the polynomial zonotope spanned by the dependent generators, where the intersection between each halfspace of the bloated polytope $\mathcal{F}$ corresponds to a polynomial level set constraint for the factors α.

Theorem 1 defines a feasible region $\alpha \in [-\mathbf{1}, \mathbf{1}] \cap \bigcup_l \mathcal{LS}_l$ for the factors α that parameterize the control input such that the intersection between a reachable set at a specific point in time and a single unsafe set is empty. However, to guarantee safety we have to consider the reachable set for the whole time horizon $t \in [0, t_f]$, which consists of a sequence of reachable sets $\mathcal{R}(\tau_0), \mathcal{R}(\tau_1), \ldots, \mathcal{R}(\tau_f)$ for consecutive time intervals $\tau_0, \tau_1, \ldots, \tau_f$. Moreover, we might also have more than one unsafe set. So overall we obtain one feasible region $\alpha \in [-\mathbf{1}, \mathbf{1}] \cap \bigcup_l \mathcal{LS}_l$ for each pair of reachable sets and unsafe sets resulting in an intersection. The feasible region for α to guarantee safety for all time intervals and all unsafe sets is given by the intersection of the feasible regions for single pairs:

$$\alpha \in [-\mathbf{1}, \mathbf{1}] \cap \bigcap_{r=1}^{\nu} \bigcup_{l=1}^{s_r} \underbrace{\langle a_{rl}, b_{rl}, E_{rl} \rangle_{LS}}_{\mathcal{LS}_{rl}},$$

where the level sets $\mathcal{LS}_{rl}$ are obtained from Theorem 1 and ν is the number of intersecting pairs. To efficiently check if a reachable set represented by a polynomial zonotope intersects an obstacle represented by a polytope, the polynomial zonotope refinement algorithm [36] can be used. This algorithm recursively splits the polynomial zonotope along the longest generator until the intersection with the polytope can either be proven or disproven using zonotope enclosures of the split polynomial zonotopes. Overall, given a vector of factors $\alpha_a \in \mathbb{R}^p$ that corresponds to the control input $u_a = c_u + G_u\alpha_a \in \mathcal{U} = \langle c_u, G_u \rangle_Z$ provided by the reinforcement learning policy, we can formulate the projection to the closest safe control

input as an optimization problem:

$$\min_{\alpha\in[-\mathbf{1},\mathbf{1}]} \|\alpha-\alpha_a\|_2^2 \text{ s.t. } \alpha\in\bigcap_{r=1}^{\nu}\bigcup_{l=1}^{s_r}\langle a_{rl},b_{rl},E_{rl}\rangle_{LS}.$$

This is a disjunctive programming problem, which can be formulated as a mixed-integer quadratic program with polynomial constraints using the convex hull relaxation [37]:

$$\min_{\alpha\in[-\mathbf{1},\mathbf{1}]} \|\alpha-\alpha_a\|_2^2 \tag{4}$$

subject to

$$\sum_{i=1}^{h}\left(\prod_{k=1}^{p}\alpha_{rl(k)}^{E_{rl(k,i)}}\right)a_{rl(\cdot,k)}\lambda_{rl}\le -b_{rl}\,\lambda_{rl},$$

$$\lambda_{rl}\in\{0,1\},\ \alpha=\sum_{l=1}^{s_r}\lambda_{rl}\,\alpha_{rl},\ \sum_{l=1}^{s_r}\lambda_{rl}=1,$$

for $r=1,\dots,\nu$ and $l=1,\dots,s_r$. Here, the disjunction is realized using the binary variables $\lambda_{rl}\in\{0,1\}$ which modify the corresponding polynomial constraints to be either active ($\lambda_{rl}=1$) or inactive ($\lambda_{rl}=0$). Let us demonstrate the optimization for our running example:

Example 2: As shown in Fig. 2, for the nonlinear system in Example 1 only the final reachable set $\mathcal{R}(t_f)$ intersects the unsafe set $\mathcal{F}$. We consequently obtain the feasible region for α by applying Theorem 1 to the sets $\mathcal{R}(t_f)$ and $\mathcal{F}$, which yields $\alpha\in\mathcal{LS}_1\vee\mathcal{LS}_2$ with

$$\mathcal{LS}_1=\Big\{[\alpha_1\,\alpha_2]^T\in\mathbb{R}^2\,\big|\,1.86\,\alpha_1+0.5\,\alpha_2-1.46\,\alpha_1\alpha_2+\alpha_1^2+\alpha_1^2\alpha_2\le -1.2\Big\}$$

and

$$\mathcal{LS}_2=\Big\{[\alpha_1\,\alpha_2]^T\in\mathbb{R}^2\,\big|\,2.34\,\alpha_1-1.75\,\alpha_2+1.51\,\alpha_1\alpha_2+0.25\,\alpha_1^2+0.25\,\alpha_1^2\alpha_2\le -0.3\Big\}.$$

Given $\alpha_a=[0.3\,0]^T$, the optimization problem (4) becomes

$$\min_{\alpha_1,\alpha_2\in[-1,1]}(\alpha_1-0.3)^2+\alpha_2^2$$

subject to

$$1.86\,\alpha_{11(1)}\lambda_{11}+0.5\,\alpha_{11(2)}\lambda_{11}-1.46\,\alpha_{11(1)}\alpha_{11(2)}\lambda_{11}+\alpha_{11(1)}^2\lambda_{11}+\alpha_{11(1)}^2\alpha_{11(2)}\lambda_{11}\le -1.2\,\lambda_{11},$$

$$2.34\,\alpha_{12(1)}\lambda_{12}-1.75\,\alpha_{12(2)}\lambda_{12}+1.51\,\alpha_{12(1)}\alpha_{12(2)}\lambda_{12}+0.25\,\alpha_{12(1)}^2\lambda_{12}+0.25\,\alpha_{12(1)}^2\alpha_{12(2)}\lambda_{12}\le -0.3\,\lambda_{12},$$

$$\lambda_{11},\lambda_{12}\in\{0,1\},\ \lambda_{11}+\lambda_{12}=1$$

$$\begin{bmatrix}\alpha_1\\ \alpha_2\end{bmatrix}=\lambda_{11}\begin{bmatrix}\alpha_{11(1)}\\ \alpha_{11(2)}\end{bmatrix}+\lambda_{12}\begin{bmatrix}\alpha_{12(1)}\\ \alpha_{12(2)}\end{bmatrix},$$

which has the optimal solution $\alpha=[\alpha_1\,\alpha_2]^T=[0.04\,0.2]^T$. The feasible regions for α_1 and α_2 are shown in Fig. 3 .

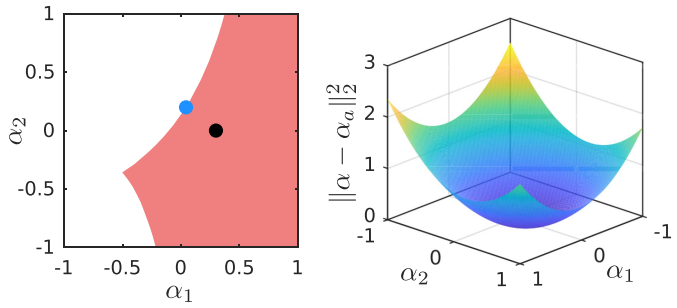


FIGURE 3. Domain (left) and objective function (right) for the optimization problem from Example 2. For the domain plot the set of infeasible values is shown in red, the desired value α_a is visualized as a black dot, and the optimal solution to the optimization problem is depicted as a blue dot.

In the presence of measurement errors $v(t)\in\mathcal{V}$ we can apply the same overall approach but have to change the initial set to $\mathcal{X}_0=(x_0\oplus\mathcal{V})\times\mathcal{U}$, where the set $\mathcal{V}$ has to be represented by independent generators to ensure that safety is guaranteed for all possible values of the measurement errors. While we focused on the conservative polynomialization algorithm [30] for simplicity, our safety shield is also compatible with other reachability approaches as long as they preserve dependencies between initial states and reachable states. This is for example the case for algorithms that compute reachable sets using the Picard-Lindelöf iteration together with Taylor models [38].

The safety shield can be used during reinforcement learning or for a learned agent. For every decision step, the action suggested by the agent is corrected to the closest safe action by (4) only if it violates safety constraints. If the safety shield is used during learning, it can be beneficial to adapt the reward to inform the agent about corrections of actions [10].

V. EXTENSIONS

We now discuss several extensions for our safety shield.

A. DIFFERENT TYPES OF CONTROL LAWS

For the basic safety shield presented in Section IV, for simplicity we considered that the control input is kept constant for the whole planning horizon. Since this is very restrictive and would in practice often prevent us from finding a feasible solution, we now discuss how to realize more advanced control strategies. Note that the reinforcement learning agent has to match the control law used for the safety shield.

a) Piecewise Constant Control Law: One simple but very effective extension to constant control inputs are piecewise constant control inputs. Instead of determining a single control input from the input set $\mathcal{U}$, we determine control inputs for all piecewise constant segments from the set $\mathcal{U}\times\cdots\times\mathcal{U}$. We can still use the extended system in (2), but have to reset the initial set for reachability analysis to $\mathcal{R}(t_i)\times\mathcal{U}$ after each of the $i=1,\dots,\mu$ piecewise constant time segments $[t_{i-1},t_i]$ with $t_i=i\cdot t_f/\mu$, where $\mathcal{R}(t_i)$ is the final reachable set from the previous segment.

b) Polynomial Control Law: Another possibility is to use control laws that are polynomial functions with respect to

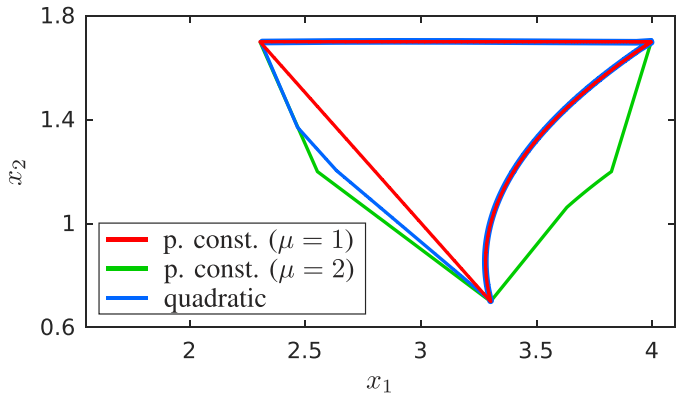


FIGURE 4. Final reachable set for the system in Example 1 for a quadratic control law and piecewise constant control laws with different numbers of segments μ.

time. We consider the quadratic case for simplicity since the extension to general polynomials is straightforward. For a quadratic control law $u(t) = c_{(1)} + c_{(2)}t + c_{(3)}t^2$ parameterized by the vector of coefficients $c \in \mathbb{R}^3$, we can use the extended system

$$\begin{bmatrix} \dot{x}(t) \\ \dot{c} \\ \dot{t} \end{bmatrix} = \begin{bmatrix} f\left(x(t), c_{(1)} + c_{(2)}t + c_{(3)}t^2, w(t)\right) \\ \mathbf{0} \\ 1 \end{bmatrix}$$

together with the initial set $x_0 \times \mathcal{C} \times 0$. In the optimization problem (4) we then determine the values for the parameter vector c, where we add the constraint $c_{(1)} + c_{(2)}t + c_{(3)}t^2 \in \mathcal{U}$ to ensure that the input constraints are satisfied. The initial set $\mathcal{C} \subset \mathbb{R}^3$ for the coefficient vector c can be determined by estimating the feasible values for c such that the constraint $c_{(1)} + c_{(2)}t + c_{(3)}t^2 \in \mathcal{U}$ is satisfied for the whole time horizon.

c) Feedback Control: We can also apply a feedback control law $u(t) = u_{ref}(t) + K(x(t) - x_{ref}(t))$ with a fixed feedback matrix $K \in \mathbb{R}^{m \times n}$, where both piecewise constant or polynomial control inputs can be used for the reference control input $u_{ref}(t)$ corresponding to the reference trajectory $x_{ref}(t)$. For the safety shield, we then compute the reachable set for the extended system

$$\begin{bmatrix} \dot{x}(t) \\ \dot{u}_{ref}(t) \\ \dot{x}_{ref}(t) \end{bmatrix} = \begin{bmatrix} f\left(x(t), u_{ref}(t) + K(x(t) - x_{ref}(t)), w(t)\right) \\ \mathbf{0} \\ f\left(x_{ref}(t), u_{ref}(t), w(t)\right) \end{bmatrix}$$

using the initial set $x_0 \times \mathcal{U} \times x_0$. In the optimization problem (4) we then determine the optimal parameter for the reference control inputs $u_{ref}(t)$, where we add the constraint $u_{ref} + K(x(t) - x_{ref}(t)) \in \mathcal{U}$ to satisfy the input constraints.

A comparison of the different control laws presented in this section is shown in Fig. 4 for the system in Example 1. The results demonstrate that even for a piecewise constant control law with only $\mu = 2$ segments we already obtain a larger reachable set than with a quadratic control law, which increases our chances to find a safe control input. While piecewise constant control laws therefore seem to be preferable, their rapidly changing values often negatively impact comfort or durability for many systems, which can be avoided with polynomial control laws.

For all control strategies we apply the following control scheme: We plan for a time horizon of t_f, but execute the resulting control law for only a shorter time period $t_c < t_f$ before planning a new trajectory. This increases the chances to avoid getting stuck in dead ends and ensures that we can react quickly to dynamic changes in the environment.

B. SPATIAL DIMENSIONS OF MOBILE ROBOTS

So far we considered the case where the safety constraints are specified directly by the system state. For collision avoidance, however, this setup is usually not sufficient since we additionally have to consider the shape and spatial dimension of mobile robots, e.g., cars, vessels, or drones, which we want to control safely. While for many other approaches this poses a huge problem, incorporating spatial dimensions of the robot into our safety shield is quite straightforward since we simply have to replace the reachable set with the occupancy set. Given the reachable set $\mathcal{R}(t)$ that typically describes all possible positions of a reference point on the robot as well as all possible robot orientations, the occupancy set is defined as

$$\mathcal{O}(t) = \left\{ o(x, d) \,\middle|\, x \in \mathcal{R}(t),\ d \in \mathcal{D} \right\}, \tag{5}$$

where the function $o : \mathbb{R}^n \times \mathbb{R}^\delta \to \mathbb{R}^\gamma$ describes how the space occupied by the robot is computed from the system state and the set $\mathcal{D}$ specifying the spatial dimension of the robot.

Example 3: Let us consider a car where the states $x_{(1)}$ and $x_{(2)}$ describe the x- and y-position of its center, and state $x_{(3)}$ describes the orientation of the car. Then the function $o(x, d)$ that defines the space occupied by the car is given as

$$o(x, d) = \begin{bmatrix} x_{(1)} + \cos(x_{(3)})\, d_{(1)} - \sin(x_{(3)})\, d_{(2)} \\ x_{(2)} + \sin(x_{(3)})\, d_{(1)} + \cos(x_{(3)})\, d_{(2)} \end{bmatrix}, \tag{6}$$

where the shape of the car is for simplicity enclosed by a rectangle, so that $d \in \mathcal{D} = [-l/2, l/2] \times [-w/2, w/2]$ with l and w denoting the length and width of the car.

To compute the occupancy set (5) from the reachable set $\mathcal{R}(t)$ and the set $\mathcal{D}$ using polynomial zonotopes, we suggest two approaches:

1) We can compute a Taylor series expansion enclosure of the function $o(x, d)$ and evaluate it in a set-based way to obtain the occupancy set $\mathcal{O}(t)$ [39, Sec. 4.4].
2) Since polynomial zonotopes can be converted to Taylor models [32, Prop. 4] we can apply Taylor model arithmetic [40] to evaluate (5) and then convert the resulting set back to a polynomial zonotope.

The resulting safety constraints that we obtain from the intersections between the occupancy set $\mathcal{O}(t)$ and obstacles have to hold for all values $d \in \mathcal{D}$. To ensure this, we could represent the set $\mathcal{D}$ with independent generators before computing $\mathcal{O}(t)$, similarly as we did for the set of measurement

errors in Section IV. However, since the set $\mathcal{D}$ is in general much larger than the set of measurement errors $\mathcal{V}$, this would often yield very conservative results. A better approach is to project out all factors that correspond to the set $\mathcal{D}$ using Fourier-Motzkin elimination [41, Chapter 4.4]. Let us demonstrate this by an example:

Example 4: We consider the constraint

$$\forall \alpha_3 \in [-1, 1]: \ \alpha_1 + \alpha_2 + \alpha_3 + \alpha_1^2 \alpha_3 \leq 1.5, \tag{7}$$

from which we want to eliminate α_3. The first step of Fourier-Motzkin elimination is to solve all constraints for α_3, which yields

$$\alpha_3 \leq \frac{1.5 - \alpha_1 - \alpha_2}{1 + \alpha_1^2}, \quad \alpha_3 \leq 1, \quad \alpha_3 \geq -1. \tag{8}$$

Next, we have to form all combinations of the constraints in (8) that result in a non-empty solution, yielding the constraints

$$\frac{1.5 - \alpha_1 - \alpha_2}{1 + \alpha_1^2} \geq -1 \quad \Rightarrow \quad -\alpha_1^2 + \alpha_1 + \alpha_2 \leq 2.5$$

$$\frac{1.5 - \alpha_1 - \alpha_2}{1 + \alpha_1^2} \leq 1 \quad \Rightarrow \quad \alpha_1^2 + \alpha_1 + \alpha_2 \geq 0.5,$$

which represent an equivalent formulation of (7).

Since Fourier-Motzkin elimination requires that the constraints are solvable for the variable that is eliminated, all terms that violate this condition have to be removed first by applying a zonotope enclosure [32, Prop. 5].

C. MIXED-INTEGER LINEAR PROGRAM FORMULATION

For some systems, solving the nonlinear mixed-integer optimization problem (4) might be computationally too expensive, especially when we have to evaluate the safety shield in real-time for online application. Therefore, we now discuss how to obtain a feasible and close to optimal solution using mixed-integer linear programming, which is significantly faster. To achieve this, we enclose the polynomial zonotopes that represent the reachable set with zonotopes using [32, Prop. 5]. Since zonotopes are linear in the factors α, the feasible region for α calculated using Theorem 1 is then given as a union of polytopes $\bigcup_l \langle A_l, b_l \rangle_P$ instead of a union of polynomial level sets. Consequently, if we additionally minimize the L^1-norm instead of the L^2-norm, we can simplify the optimization problem (4) to

$$\min_{\alpha \in [-\mathbf{1}, \mathbf{1}]} \|\alpha - \alpha_a\|_1 \text{ s.t. } \alpha \in \bigcap_{r=1}^{\nu} \bigcup_{l=1}^{s_r} \langle A_{rl}, b_{rl} \rangle_P,$$

which can be formulated as a mixed-integer linear program using Balas' Theorem [42]:

$$\min_{\alpha \in [-\mathbf{1}, \mathbf{1}]} \|\alpha - \alpha_a\|_1 \tag{9}$$

subject to

$$A_{rl}\, \widehat{\alpha}_{rl} \leq \lambda_{rl}\, b_{rl}, \ -\mathbf{1}\, \lambda_{rl} \leq \widehat{\alpha}_{rl} \leq \mathbf{1}\, \lambda_{rl},$$

$$\lambda_{rl} \in \{0, 1\}, \ \alpha = \sum_{l=1}^{s_r} \widehat{\alpha}_{rl}, \ \sum_{l=1}^{s_r} \lambda_{rl} = 1,$$

for $r = 1, \ldots, \nu$ and $l = 1, \ldots, s_r$. The structure of this optimization problem is very similar to (4), except that we introduced the new variables $\widehat{\alpha}_{rl} = \alpha_{rl} \lambda_{rl}$ to avoid the bilinear terms and obtain a linear program. Due to the over-approximation of all nonlinear terms of the polynomial zonotope by the zonotope enclosure, it holds that every feasible solution for (9) is a feasible solution to the original problem (4), but some values that are feasible for (4) will not be feasible for (9). Note that if the system dynamics (1) is linear, we directly obtain a mixed-integer linear program in the form of (9). Moreover, we can always first check if the desired value α_a satisfies the original nonlinear constraints and only perform the simplification to a mixed-integer linear program if it does not. A mixed-integer quadratic program can be obtained in a similar way as the mixed-integer linear program by enclosing all generators that belong to higher-order polynomials by a zonotope. Finally, since mixed-integer programming can be highly parallelized, the computation time for optimization can always be reduced by using a more powerful machine with more cores.

D. CONSTRAINT GROUPING

Since the time step size for reachability analysis is usually relatively small, it often happens that many reachable sets for consecutive time intervals intersect the same obstacle, resulting in a lot of very similar constraints. We can reduce the computation time by grouping similar constraints together, as we demonstrate with the following example:

Example 5: The two constraints

$$1.1\, \alpha_1 + 0.7\, \alpha_1 \alpha_2 \leq 0.3$$

$$1.3\, \alpha_1 + 0.5\, \alpha_1 \alpha_2 \leq 0.3$$

on $\alpha_1, \alpha_2 \in [-1, 1]$ can be grouped to the single constraint

$$\forall \epsilon_1 \in [1.1, 1.3], \ \forall \epsilon_2 \in [0.5, 0.7]: \ \epsilon_1\, \alpha_1 + \epsilon_2\, \alpha_1 \alpha_2 \leq 0.3.$$

To eliminate the new variables ϵ_1 and ϵ_2 we represent their domains as a summation of the center with a zero-centered uncertainty as $\epsilon_1 \in 1.2 + \widetilde{\epsilon}_1$, $\epsilon_2 \in 0.6 + \widetilde{\epsilon}_2$ with $\widetilde{\epsilon}_1, \widetilde{\epsilon}_2 \in [-0.1, 0.1]$, which finally yields

$$1.2\, \alpha_1 + 0.6\, \alpha_1 \alpha_2 \leq \min_{\substack{\alpha_1, \alpha_2 \in [-1,1] \\ \widetilde{\epsilon}_1, \widetilde{\epsilon}_2 \in [-0.1, 0.1]}} 0.3 - \epsilon_1\, \alpha_1 - \epsilon_2\, \alpha_1 \alpha_2,$$

where a lower bound for the optimal value of the minimization problem can be computed using interval arithmetic [43].

In addition to the number of constraints, constraints grouping also decreases the number of integer variables for the optimization in (4), which reduces computation time. Since integer variables are required only if the safe region for the agent is non-convex, another strategy to accelerate the optimization is to replace non-convex safe regions by the largest convex subset [44].

TABLE 1. States n, control inputs m, planning horizon t_f, number of pre-computed reachable sets, and extensions applied for each benchmark.

Benchmark	n	m	t_f	Sets	Extensions
F1tenth car	5	2	2 s	5	V-A.a, V-B, V-C, V-E
Auto. driving	4	2	0.8 s	60	V-A.c, V-B, V-C, V-D, V-E
Quadrotor 2D	6	2	0.5 s	256	V-E
Quadrotor 3D	10	3	3 s	1	V-B, V-C, V-E

E. REACHABLE SET PRE-COMPUTATION

In order to reduce the computation time for our safety shield, we can pre-compute the reachable set starting from an initial set $\mathcal{X}_0$ offline, and then apply the reachable subset approach [31] to efficiently extract the reachable set for the current state $x_0 \in \mathcal{X}_0$ during online execution. Since for nonlinear systems the accuracy of the reachable set enclosure depends on the size of the initial set, we cannot make $\mathcal{X}_0$ too large but instead have to divide the relevant state space into sets of suitable size. The number of required sets for such a division grows exponentially with the system dimension, so that this approach is not suited for high-dimensional systems. However, for many systems the differential equation $\dot{x}(t) = f(x(t), u(t), w(t))$ describing the system dynamics is invariant with respect to transformations of certain states [45, Sec. 4.1]. For example, the dynamics of a car are invariant with respect to translations of the car's position and with respect to rotations of the car's orientation. In this case only the state space for the states that are not invariant has to be divided since we can always apply a suitable state space transformation to set the invariant states to 0.

VI. EXPERIMENTAL EVALUATION

We now demonstrate the performance of our safety shield on several benchmark systems, where each benchmark highlights different properties of our approach. If not explicitly stated otherwise, all computations are carried out in Python on a 2.9 GHz quad-core i7 processor with 32 GB memory. We use the CORA toolbox [46] to pre-compute reachable sets, proximal policy optimization [47] for reinforcement learning, Gurobi to solve the mixed-integer linear and quadratic programs, and CasADi together with the BONMIN solver to solve mixed-integer nonlinear programs.[2] Benchmark parameters as well as the applied extensions from Section V are listed in Table 1. We published our implementation on CodeOcean[3] and created a video showing our results.[4]

A. F1TENTH RACECAR

To demonstrate that our safety shield is fast and robust enough to be applied to a real system, we conduct experiments on an F1tenth racecar [48], whose dynamics are described by a kinematic single-track model. Moreover, the car contains a low-level PI controller with gains $k_P = 8$ and $k_I = 1$ that takes as input the desired velocity and realizes the required acceleration. Overall, this results in the model

$$\dot{s}_x = \cos(\psi)\, v,\ \dot{s}_y = \sin(\psi)\, v,\ \dot{\psi} = u_2 + w_2,$$
$$\dot{v} = k_P(u_1 - v) + k_I\, e_I + w_1,\ \dot{e}_I = u_1 - v, \tag{10}$$

where the system state consists of the x- and y-position of the center s_x, s_y, the velocity v, the orientation ψ, and the integrated error of the PI controller e_I. The control inputs are the desired velocity u_1 and the steering angle u_2, which are bounded by the set $\mathcal{U} = [0, 0.5]\text{m s}^{-1} \times [-0.3, 0.3]\text{rad}$. To ensure that the model (10) encloses all possible behaviors of the real system, we performed conformance checking using the AROC toolbox [49] to determine the process noise as well as the measurement error from data traces we recorded from the real car, which results in the sets

$$\mathcal{W} = [-0.007, 0.0035]\text{m s}^{-2} \times [-0.0104, 0.0132]\text{rad s}^{-1}$$
$$\mathcal{V} = [-0.0584, 0.0446]\text{m s}^{-1} \times [-0.0438, 0.0466]\text{m s}^{-1}$$
$$\times [-0.0933, 0.0561]\text{m s}^{-2} \times [-0.0446, 0.0593]\text{rad s}^{-1}$$
$$\times [-0.0005, 0]\text{m s}^{-2}.$$

To incorporate the size of the car, we use the output function in (6) with length 0.51 m and width 0.31 m.

For control we use a piecewise constant control law with $\mu = 2$ segments and a planning horizon of $t_f = 2$ s, and we replan as soon as the previous computation is finished. Moreover, we simplify the optimization problem for action projection to a mixed-integer quadratic program, which on average took 0.14 s to solve during our experiments. The car uses a 1.9 GHz six-core ARMv8 processor with 7.6 GB memory and is equipped with a LiDAR sensor. To obtain the unsafe sets $\mathcal{F}_i$, we enclose all points measured by the LiDAR by a union of polytopes. Moreover, while the velocity and the integrated error can be directly obtained from the car's internal sensors, we use a particle filter [50] to determine the position and orientation of the car in the environment from LiDAR measurements. For our experiments, we then applied reinforcement learning to train an agent on four environment maps that were similar to but slightly different from the map we used for the experiments on the real F1tenth car. In addition to the system state, we used the LiDAR measurements and the position of the goal set as observations for the agent, and we did not use the safety shield during training.

As shown in Fig. 5, without the safety shield, the trained agent is unsafe since the car crashes into the obstacle. With our safety shield, however, the car avoids the obstacle and safely reaches the goal set. This not only demonstrates that our safety shield successfully works on a real system, but also that the modifications to the control inputs suggested by the reinforcement learning policy are small enough for the agent to still fulfill its objective.

[2] [Online]. Available: https://www.gurobi.com/ and https://web.casadi.org/
[3] [Online]. Available: https://codeocean.com/capsule/9949621/tree/v1
[4] [Online]. Available: https://youtu.be/6ISKxO4DDWA

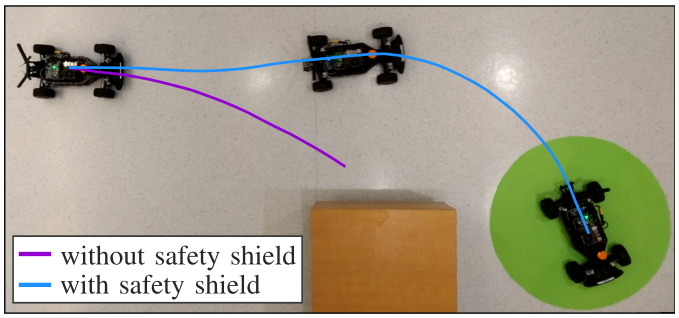


FIGURE 5. Trajectories driven by the F1tenth racecar with and without the safety shield, where the green area is the goal set and the orange area is the obstacle.

TABLE 2. Results for the evaluation of our safety shield on 2000 common road traffic scenarios.

Agent	Collisions	Goal Reached	Time
without safety shield	10 (0.5%)	1907 (95.35%)	-
with safety shield	0 (0%)	1925 (96.25%)	0.043 s
constraint grouping	0 (0%)	1924 (96.20%)	0.035 s

B. AUTONOMOUS DRIVING

In order to show that our safety shield can handle very complex reach-avoid problems that include dynamic obstacles, we consider the motion planning benchmarks for autonomous cars provided by the CommonRoad database [51]. As system dynamics we use the kinematic single track model from [20, Sec. VII] with the same input set $\mathcal{U}$ and set of process noise $\mathcal{W}$ as in [20, Sec. VII]. This model is very similar to the model in (10), with the only difference that the acceleration instead of the desired velocity is used as a control input. The car we consider is a BMW 320i, which has a length of 4.51 m and a width of 1.61 m. To guarantee safety even though the intentions of the other cars are unclear, we use the tool SPOT [52] to compute all possible occupancies of the other traffic participants that apply to traffic rules using set-based prediction.

To counteract the large process noise for this benchmark, we use a feedback controller $u(t) = u_{ref} + K(x(t) - x_{ref}(t))$ for the safety shield, where the reference input u_{ref} is piecewise constant with $\mu = 2$ segments. The feedback matrix $K \in \mathbb{R}^{m \times n}$ is determined by applying an LQR control approach with state weighting matrix $Q = I_4$ and input weighting matrix $R = I_2$ to the linearized system. Moreover, we use a planning horizon of $t_f = 0.8$ s and replan after $t_c = 0.4$ s. We apply reinforcement learning to train an agent that aims to safely control the car, where we do not use the safety shield during training. The observations for the agent are selected from [53, Table II]. In particular, we use the state of the ego vehicle, the distances of the ego vehicle to road/lane boundaries as well as to the goal set, and the states of surrounding vehicles.

The effect of the safety shield is highlighted by the results for 2000 traffic scenarios shown in Table 2: While the original agent collides with other traffic participants in 10 scenarios, our safety shield successfully prevents all collisions. Moreover, applying the safety shield does not lead to a reduced goal-reaching percentage, but instead even increases the number of scenarios for which the goal set is reached. Table 2 also demonstrates the effect of constraint grouping (see Section V-D), which reduces the average computation time for solving the optimization problem, but slightly decreases the goal reaching percentage due to the increased conservatism. In Fig. 6 the results for one specific traffic scenario are visualized. There, the agent without the safety shield changes the lane too early and collides with the adjacent truck, whereas the agent with the safety shield changes the lane just in time and finally reaches the goal set in the end.

C. QUADROTOR 2D

Next, we compare our safety shield with a safe reinforcement learning approach that modifies the optimization criterion. In particular, we incorporate the safety specification as a violation penalty in the reward function. For this, we consider a benchmark problem from the safe-control-gym [54] featuring a trajectory tracking task for a two-dimensional quadrotor. As shown in Fig. 7, the trajectory that should be tracked is partially located inside an unsafe region, so that there exists a conflict between tracking performance and safety constraint satisfaction. The dynamics of the quadrotor are according to [54, Eq. (3)] given as

$$\begin{aligned} \ddot{s}_x &= \sin(\psi)\,(u_1 + u_2)/m + w_1 \\ \ddot{s}_z &= \cos(\psi)\,(u_1 + u_2)/m - g + w_2 \\ \ddot{\psi} &= (u_2 - u_1)\,a/\left(\sqrt{2}\,I_{yy}\right) + w_3, \end{aligned}$$

where $m = 0.027$ kg is the mass, $g = 9.81\ \mathrm{m\,s^{-2}}$ is the gravitational acceleration, $a = 0.0397$ m is distance from each motor pair to the center of mass of the quadrotor, and $I_{yy} = 1.4 \cdot 10^{-5}\ \mathrm{kg/m^2}$ is the moment of inertia. The system state consists of the x- and z-positions s_x, s_z as well as the pitch angle ψ of the quadrotor together with the corresponding velocities. To decouple forward thrust and tilting torque, the input set for the control inputs u_1 and u_2 that represent the thrusts generated by the two rotors is restricted to

$$\mathcal{U} = \left\langle \begin{bmatrix} 0.1323\,\mathrm{N} \\ 0.1323\,\mathrm{N} \end{bmatrix}, \begin{bmatrix} 0.0125\,\mathrm{N} & 0.0015\,\mathrm{N} \\ 0.0125\,\mathrm{N} & -0.0015\,\mathrm{N} \end{bmatrix} \right\rangle_Z.$$

The process noise w_1, w_2, w_3 is bounded by the set $\mathcal{W} = 0.01 \cdot [-\mathbf{1}, \mathbf{1}]$.

For the safety shield we use a constant control input with a planning horizon of $t_f = 0.5$ s, where we replan after $t_c = 0.02$ s. To perform action projection, we solve the original nonlinear optimization problem, which takes 0.004 s on average during our experiments. The main reason for the fast computation time is that the safe region for the quadrotor is convex, which results in an optimization problem without any integer variables. We train three different agents: A baseline agent that should track the trajectory and gets no information

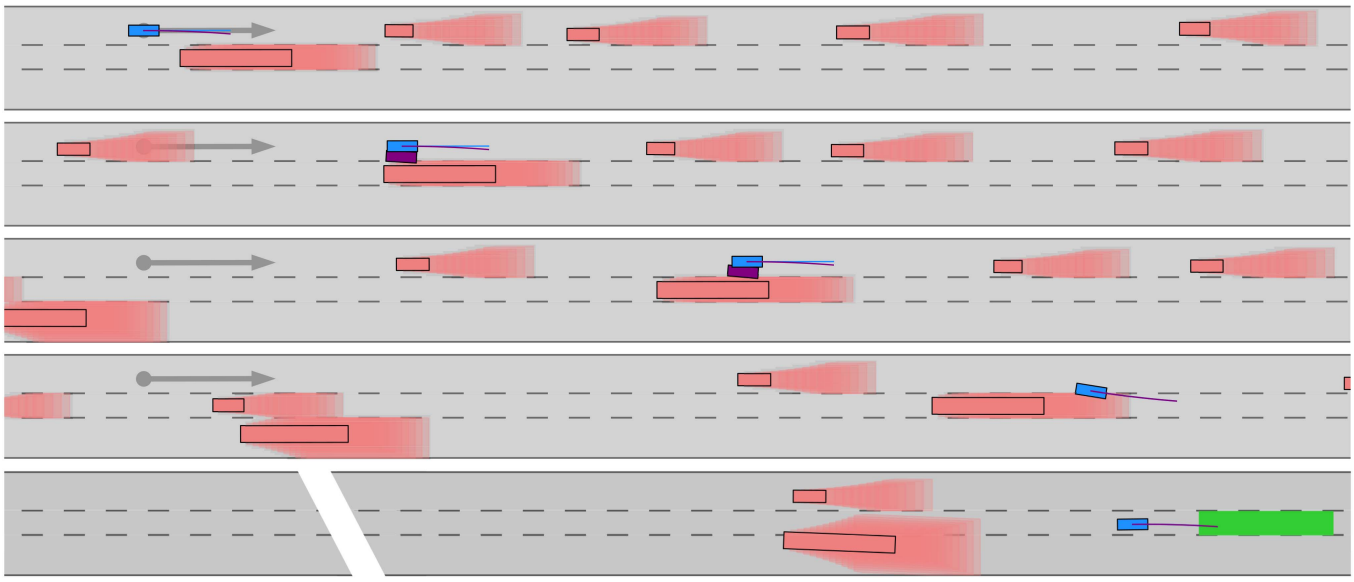


FIGURE 6. Results for the CommonRoad scenario *DEU_LocationALower-33_16_T-1* visualized at times 0 s, 1.2 s, 2.8 s, 4.4 s, and 9.2 s (from top to bottom), where the agent without safety shield is depicted in purple, the agent with safety shield is depicted in blue, the dynamic obstacles are depicted in red, and the goal set is depicted in green.

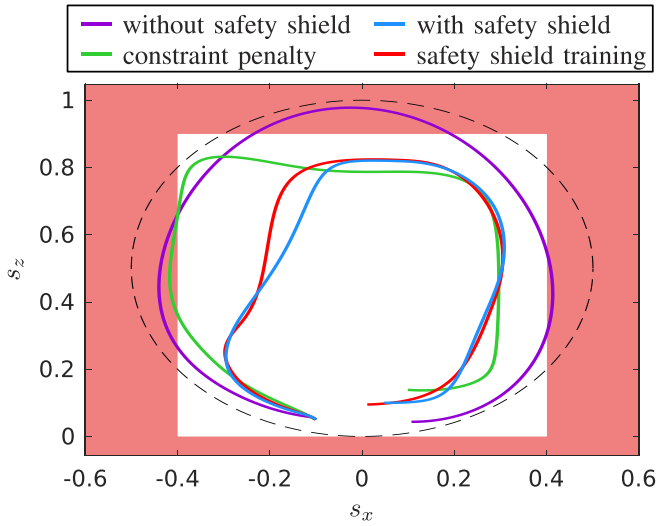


FIGURE 7. Trajectories for the 2D quadrotor benchmark featuring the baseline agent with and without safety shield, the constraint-penalty agent, and the agent that is trained with the safety shield. The trajectory that should be tracked is visualized by the dashed black line and the unsafe regions are depicted in red.

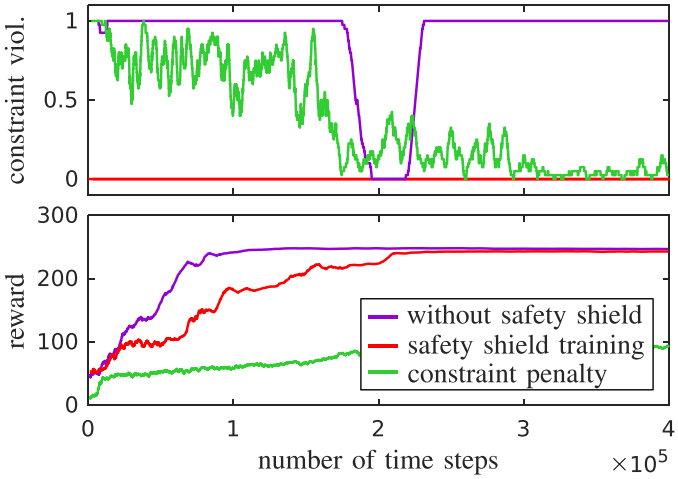


FIGURE 8. Episode rewards and constraint violations for the 2D quadrotor benchmark observed during training without safety shield, with safety shield, and with constraint penalty.

about the constraints, a constraint-penalty agent where the reward is extended with a penalty for constraint violation, and a safe agent that is trained with the safety shield. As shown in Fig. 8, the safe and baseline agents converge after 400 000 training steps while the agent with constraint penalty needs 2 million training steps to converge. Moreover, only the safe agent never violates any constraints during training, and could therefore also be used for training directly on the real physical system.

The results for deploying the different trained agents are shown in Fig. 7. As expected, the baseline agent without the safety shield violates the safety constraints since they were not considered during training. Also, the constraint-penalty agent violates the constraints, which demonstrates that it is not sufficient to incorporate the safety constraints into the training process. Only the two agents that apply our safety shield stay inside the safe region for all times, where the agent that uses the safety shield during training achieves a smoother trajectory compared to the baseline agent.

D. QUADROTOR 3D

To compare our safety shield with reachability-based trajectory safeguard [25], we consider the three-dimensional quadrotor benchmark from [25, Sec. V.B]. Reachability-based trajectory safeguard [25] applies the safety shield to a simplified trajectory-generating model and the resulting trajectory is then tracked by a low-level controller that uses

the original nonlinear system model. For the quadrotor, the trajectory-generating model for each of the three spatial directions $i \in \{1, 2, 3\}$ is

$$\dot{x}_i = v_i + a_i t - (2a_i + 3(v_i - u_i))t^2 + (a_i + 2(v_i - u_i))t^3$$
$$\dot{v}_i = 0, \qquad \dot{a}_i = 0, \qquad \dot{t} = 1,$$

where x_i is the quadrotor position, v_i and a_i are respectively the velocity and the acceleration at the beginning of the trajectory, and t is time. The input u_i to the system is the peak velocity reached at time $t = 1.5\,\text{s}$, which is bounded by the set $\mathcal{U} = \{u = [u_1\, u_2\, u_3]^T \mid \|u\|_2 \leq 5\,\text{m s}^{-1}\}$. To apply our safety shield, we tightly inner-approximate the set $\mathcal{U}$ with a zonotope using the method described in [55, Sec. IV]. A similar trajectory-generating model is used to decelerate the quadrotor from the peak velocity back to velocity 0, so that the overall planning horizon is $t_f = 3\,\text{s}$. We consider the same control task as in [25, Sec. V.B], which is to safely navigate the quadrotor through a 100 m long tunnel with randomly generated box obstacles. For our experiments, we deployed the same trained reinforcement learning agent as used in [25] on 100 tunnels with different obstacles and compared the conservatism of the two safety shields in terms of the required control input correction $\|u - u_a\|_2$ at each intervention of the safety shield. While both safety shields had to intervene for 5078 out of 5760 time steps, the average control input modification for our approach is with $1.13\,\text{m s}^{-1}$ smaller than the average modification $1.22\,\text{m s}^{-1}$ for the safety shield from [25], which increases the chances that the agent can successfully complete its task.

VII. DISCUSSION

Finally, let us discuss some properties of our safety shield.

A. SAFETY GUARANTEES FOR INFINITE TIME

Our basic safety shield approach can guarantee safety only for the finite time horizon t_f. To obtain safety guarantees for an infinite time horizon, one can either combine our safety shield with a fail-safe planner [56] that takes over when the safety shield cannot determine a safe trajectory anymore, or one can modify the safety shield in such a way that the system always stops in a safe final state at the end of the planning horizon [25].

B. COMPUTATIONAL COMPLEXITY

The two main steps required for our safety shield are computing the reachable set and solving the mixed-integer optimization problem (4) for action projection. The complexity of the conservative polynomialization algorithm for reachability analysis is $\mathcal{O}(n^5)$ with respect to the system dimension according to [57, Sec. 4.1.4]. However, for many benchmarks one can apply the pre-computation discussed in Section V-E to avoid computing reachable sets online. Solving a mixed-integer optimization problem is in general NP-hard [58]. But, as we demonstrated with the numerical experiments in Section VI, by applying the simplification to a mixed-integer linear program in Section V-C and/or constraint grouping in Section V-D we can solve this optimization efficiently.

C. SAFE COMPUTATION TIME CONSIDERATION

As demonstrated by the experiments in Section VI, even with all the speed-ups discussed in Section V, the calculations required for our safety shield still need a certain amount of computation time that, depending on the system, might be too long to simply be neglected. Therefore, in order to consider the required computation time in a formally correct manner, we can apply the following well-known procedure [59]: We allocate a certain computation time t_{comp} for the calculations and use reachability analysis to predict the reachable states for the allocated computation time. By using this set as the initial set for our safety shield, we can guarantee safety even though the required calculations are not instantaneous. If the computation does not finish in the allocated computation time, we either stick to the safe solution from the previous time step or apply a failsafe maneuver.

D. CONSERVATISM OF THE SAFETY SHIELD

Due to over-approximation errors, our safety shield might not be able to always find a feasible solution if one exists. In particular, there are four sources of conservatism:

- Since the exact reachable set cannot be computed for general nonlinear systems, we compute a tight enclosure instead (e.g., we aim to minimize the Hausdorff distance between the enclosure and the exact set).
- Due to dependency preservation, the abstraction error for reachability analysis is computed on the reachable set for the whole input set rather than the smaller reachable set for a specific control input, which results in additional conservatism.
- For bloating the obstacles by the set of uncertainties defined by the independent generators, we use an over-approximative Minkowski sum in Theorem 1 that simply pushes the obstacle halfspaces outward.
- Since we choose a certain type of control law in advance, we restrict the space of possible control inputs.

However, all of these over-approximation errors can be made arbitrarily small: The over-approximation for reachability converges to zero if the time step size is reduced and/or the reachable set is split, which also eliminates the error introduced by dependency preservation. Moreover, the approximative Minkowski sum in Theorem 1 can be replaced by the exact one and every control law can be approximated arbitrary close by a piecewise constant control law with an infinite number of piecewise constant segments.

E. PARAMETER TUNING

Since the settings for reachability analysis can be tuned automatically [60], [61], the main design parameters for our safety shield in addition to the type of control law discussed in Section V-A are the planning horizon t_f and the replanning time t_c. A longer planning horizon t_f often yields better control performance due to the larger lookahead, but also increases

the computation time. Especially in the presence of dynamic obstacles, a small replanning time t_c is desirable in order to be able to quickly react to a changing environment. However, a small t_c requires the approach to be faster in order to run in real-time. Finally, the extensions discussed in Section V-C, V-D, and V-E all reduce the computation time at the cost of introducing more conservatism.

VIII. CONCLUSION

We presented a novel safety shield for nonlinear continuous systems with input constraints that can be added to reinforcement learning agents in order to prevent them from applying unsafe actions. Since our safety shield uses set-based computations in the form of reachability analysis to determine which actions are safe and which are unsafe, it can guarantee robust safety despite process noise and measurement errors. Moreover, because our approach applies highly parallelized mixed-integer programming to project the action from the agent to the closest safe action, it is possible to reduce the computation time by using a more powerful machine with more cores. Finally, we demonstrated with several numerical examples as well as experiments on a real system that our safety shield modifies the actions proposed by the reinforcement learning agent as little as necessary for robust safety.

ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force or the United States Navy.

REFERENCES

[1] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. Symp. Ser. Comput. Intell.*, 2020, pp. 737–744.
[2] B. R. Kiran et al., "Deep reinforcement learning for autonomous driving: A survey," *Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022.
[3] Z. Zhang, D. Zhang, and R. C. Qiu, "Deep reinforcement learning for power system applications: An overview," *CSEE J. Power Energy Syst.*, vol. 6, no. 1, pp. 213–225, 2019.
[4] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 22–31.
[5] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Projection-based constrained policy optimization," in *Proc. Int. Conf. Learn. Representations*, 2019.
[6] M. Hasanbeig, A. Abate, and D. Kroening, "Cautious reinforcement learning with logical constraints," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2020, pp. 483–491.
[7] X. Wang, C. Pillmayer, and M. Althoff, "Learning to obey traffic rules using constrained policy optimization," in *Proc. Int. Conf. Intell. Transp. Syst.*, 2022, pp. 2415–2421.
[8] B. Könighofer, J. Rudolf, A. Palmisano, M. Tappler, and R. Bloem, "Online shielding for stochastic systems," in *Proc. NASA Formal Methods Symp.*, 2021, pp. 231–248.
[9] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *J. Artif. Intell. Res.*, vol. 24, pp. 81–108, 2005.
[10] H. Krasowski, J. Thumm, M. Müller, X. Wang, and M. Althoff, "Provably safe reinforcement learning: A theoretical and experimental comparison," 2022, *arXiv:2205.06750*.
[11] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *Proc. Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–7.
[12] D. Isele, A. Nakhaei, and K. Fujimura, "Safe reinforcement learning on autonomous vehicles," in *Proc. Int. Conf. Intell. Robots Syst.*, 2018, pp. 6162–6167.
[13] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," in *Proc. Int. FLAIRS Conf.*, 2022.
[14] J. Thumm and M. Althoff, "Provably safe deep reinforcement learning for robotic manipulation in human environments," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 6344–6350.
[15] W. Saunders, G. Sastry, A. Stuhlmüller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," in *Proc. Int. Conf. Auton. Agents MultiAgent Syst.*, 2018, pp. 2067–2069.
[16] N. Hunt, N. Fulton, S. Magliacane, T. N. Hoang, S. Das, and A. Solar-Lezama, "Verifiably safe exploration for end-to-end reinforcement learning," in *Proc. 24th Int. Conf. Hybrid Systems: Comput. Control*, 2021, pp. 1–11.
[17] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2669–2678.
[18] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The Simplex architecture for safe online control system upgrades," in *Proc. Amer. Control Conf.*, 1998, pp. 3504–3508.
[19] D. T. Phan, R. Grosu, N. Jansen, N. Paoletti, S. A. Smolka, and S. D. Stoller, "Neural simplex architecture," in *Proc. NASA Formal Methods Symp.*, 2020, pp. 97–114.
[20] B. Schürmann, M. Klischat, N. Kochdumper, and M. Althoff, "Formal safety net control using backward reachability analysis," *IEEE Trans. Autom. Control*, vol. 67, no. 11, pp. 5698–5713, Nov. 2022.
[21] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 3387–3395.
[22] Z. Marvi and B. Kiumarsi, "Safe reinforcement learning: A control barrier function optimization approach," *Int. J. Robust Nonlinear Control*, vol. 31, no. 6, pp. 1923–1940, 2021.
[23] O. Bastani, "Safe reinforcement learning with nonlinear dynamics via model predictive shielding," in *Proc. Amer. Control Conf.*, 2021, pp. 3488–3494.
[24] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, 2021, Art. no. 109597.
[25] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan, "Reachability-based trajectory safeguard (RTS): A safe and fast reinforcement learning safety layer for continuous control," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3663–3670, Apr. 2021.
[26] J. H. Gillula and C. J. Tomlin, "Guaranteed safe online learning via reachability: Tracking a ground target using a quadrotor," in *Proc. Int. Conf. Robot. Automat.*, 2012, pp. 2723–2730.
[27] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent HamiltonJacobi formulation of reachable sets for continuous dynamic games," *IEEE Trans. Autom. Control*, vol. 50, no. 7, pp. 947–957, Jul. 2005.
[28] M. Selim, A. Alanwar, S. Kousik, G. Gao, M. Pavone, and K. H. Johansson, "Safe reinforcement learning using black-box reachability analysis," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 10665–10672, Oct. 2022.
[29] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, "Constrained zonotopes: A new tool for set-based estimation and fault detection," *Automatica*, vol. 69, pp. 126–136, 2016.
[30] M. Althoff, "Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets," in *Proc. Int. Conf. Hybrid Syst.: Comput. Control*, 2013, pp. 173–182.
[31] N. Kochdumper, B. Schürmann, and M. Althoff, "Utilizing dependencies to obtain subsets of reachable sets," in *Proc. Int. Conf. Hybrid Syst.: Comput. Control*, 2020, article 1.
[32] N. Kochdumper and M. Althoff, "Sparse polynomial zonotopes: A novel set representation for reachability analysis," *IEEE Trans. Autom. Control*, vol. 66, no. 9, pp. 4043–4058, Sep. 2021.
[33] M. Althoff, O. Stursberg, and M. Buss, "Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization," in *Proc. Int. Conf. Decis. Control*, 2008, pp. 4042–4048.

[34] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff, "Model conformance for cyber-physical systems: A survey," *Trans. Cyber- Phys. Syst.*, vol. 3, no. 3, 2018, Art. no. 30.
[35] M. Koschi and M. Althoff, "Set-based prediction of traffic participants considering occlusions and traffic rules," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 249–265, Jun. 2021.
[36] S. Bak, S. Bogomolov, B. Hencey, N. Kochdumper, E. Lew, and K. Potomkin, "Reachability of Koopman linearized systems using random fourier feature observables and polynomial zonotope refinement," in *Proc. Int. Conf. Comput. Aided Verification*, 2022, pp. 490–510.
[37] I. E. Grossmann and S. Lee, "Generalized convex disjunctive programming: Nonlinear convex hull relaxation," *Comput. Optim. Appl.*, vol. 26, no. 1, pp. 83–100, 2003.
[38] X. Chen, S. Sankaranarayanan, and E. Ábrahám, "Taylor model flowpipe construction for non-linear hybrid systems," in *Proc. Real-Time Syst. Symp.*, 2012, pp. 183–192.
[39] N. Kochdumper and M. Althoff, "Reachability analysis for hybrid systems with nonlinear guard sets," in *Proc. 23rd Int. Conf. Hybrid Syst.: Comput. Control*, 2020, pp. 1–10.
[40] K. Makino and M. Berz, "Taylor models and other validated functional inclusion methods," *Int. J. Pure Appl. Math.*, vol. 4, no. 4, pp. 379–456, 2003.
[41] G. Dantzig, *Linear Programming and Extensions*. Princeton, NJ, USA: Princeton Univ. Press, 2016.
[42] E. Balas, "Disjunctive programming: Properties of the convex hull of feasible points," *Discrete Appl. Math.*, vol. 89, no. 1, pp. 3–44, 1998.
[43] L. Jaulin, M. Kieffer, and O. Didrit, *Applied Interval Analysis*. Berlin/Heidelberg, Germany: Springer, 2006.
[44] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2015, pp. 109–124.
[45] S. Bak, Z. Huang, F. A. T. Abad, and M. Caccamo, "Safety and progress for distributed cyber-physical systems with unreliable communication," *Trans. Embedded Comput. Syst.*, vol. 14, no. 4, 1–22, 2015.
[46] M. Althoff, "An introduction to CORA 2015," in *Proc. Int. Workshop Appl. Verification Continuous Hybrid Syst.*, 2015, pp. 120–151.
[47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[48] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," *Proc. Mach. Learn. Res.*, vol. 123, pp. 77–89, 2020.
[49] N. Kochdumper, F. Gruber, B. Schürmann, V. Gaßmann, M. Klischat, and M. Althoff, "AROC: A toolbox for automated reachset optimal controller synthesis," in *Proc. 24th Int. Conf. Hybrid Syst.: Computation Control*, 2021, pp. 1–6.
[50] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artif. Intell.*, vol. 128, no. 1-2, pp. 99–141, 2001.
[51] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. IEEE Intell. Veh. Symp.*, 2017, pp. 719–726.
[52] M. Koschi and M. Althoff, "SPOT: A tool for set-based prediction of traffic participants," in *Proc. Intell. Veh. Symp.*, 2017, pp. 1686–1693.
[53] X. Wang, H. Krasowski, and M. Althoff, "CommonRoad-RL: A configurable reinforcement learning environment for motion planning of autonomous vehicles," in *Proc. Int. Intell. Transp. Syst. Conf.*, 2021, pp. 466–472.
[54] Z. Yuan et al., "Safe-control-Gym: A unified benchmark suite for safe learning-based control and reinforcement learning in robotics," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 11142–11149, Oct. 2022.
[55] V. Gaßmann and M. Althoff, "Scalable zonotope-ellipsoid conversions using the Euclidean zonotope norm," in *Proc. Amer. Control Conf.*, 2020, pp. 4715–4721.
[56] C. Pek and M. Althoff, "Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization," in *Proc. Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1447–1454.
[57] N. Kochdumper, "Extensions of polynomial zonotopes and their application to verification of cyber-physical systems," Ph.D. dissertation, Tech. Univ. Munich, München, Germany, 2022.
[58] C. H. Papadimitriou, "On the complexity of integer programming," *J. ACM*, vol. 28, no. 4, pp. 765–768, 1981.
[59] B. Schürmann, N. Kochdumper, and M. Althoff, "Reachset model predictive control for disturbed nonlinear systems," in *Proc. Int. Conf. Decis. Control*, 2018, pp. 3463–3470.
[60] M. Wetzlinger, A. Kulmburg, and M. Althoff, "Adaptive parameter tuning for reachability analysis of nonlinear systems," in *Proc. 24th Int. Conf. Hybrid Syst.: Comput. Control*, 2021, pp. 1–11.
[61] M. Wetzlinger, N. Kochdumper, S. Bak, and M. Althoff, "Fully-automated verification of linear systems using inner-and outer-approximations of reachable sets," 2022, *arXiv:2209.09321*.

NIKLAS KOCHDUMPER received the B.Sc. degree in mechanical engineering, the M.Sc. degree in robotics, cognition and intelligence, and the Ph.D. degree in computer science from Technische Universität München, Germany, in 2015, 2017, and 2022, respectively. He is currently a Postdoctoral Researcher with Stony Brook University, Stony Brook, NY, USA. His research interests include formal verification of continuous and hybrid systems, reachability analysis, computational geometry, controller synthesis, and neural network verification.

HANNA KRASOWSKI received the B.Sc. degree in mechanical engineering from Technische Universität Darmstadt, Darmstadt, Germany, in 2017, and the M.Sc. degree in robotics, cognition and intelligence in 2020 from Technische Universität München, Munich, Germany, where she is currently working toward the Ph.D. degree. Her research interests include provably safe reinforcement learning and motion planning for cyber-physical systems.

XIAO WANG received the B.Eng. degree in vehicle engineering from Tongji University, Shanghai, China, in 2015, and the M.Sc. degree in mechanical engineering in 2018 from Technische Universität München, München, Germany, where she is currently working toward the Ph.D. degree. Her research interests include motion planning for autonomous vehicles, formal methods, and safe reinforcement learning.

STANLEY BAK received the B.Sc. degree in computer science from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2007, and the M.Sc. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2009 and 2013, respectively. He is currently an Assistant Professor in computer science with Stony Brook University in Stony Brook, NY, USA. His research interests include verification and testing methods for cyber-physical systems and neural networks.

MATTHIAS ALTHOFF received the Diploma Engineering degree in mechanical engineering and the Ph.D. degree in electrical engineering from Technische Universität München, München, Germany, in 2005 and 2010, respectively. He is currently an Associate Professor in computer science with Technische Universität München. From 2010 to 2012 he was a Postdoctoral Researcher with Carnegie Mellon University, Pittsburgh, PA, USA, and from 2012 to 2013 an Assistant Professor with Technische Universität Ilmenau, Ilmenau, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.

CC BY 4.0 DEED

Attribution 4.0 International

Canonical URL : https://creativecommons.org/licenses/by/4.0/

See the legal code

You are free to:

Share — copy and redistribute the material in any medium or format for any purpose, even commercially.

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation .

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Notice

This deed highlights only some of the key features and terms of the actual license. It is not a license and has no legal value. You should carefully review all of the terms and conditions of the actual license before using the licensed material.

Creative Commons is not a law firm and does not provide legal services. Distributing, displaying, or linking to this deed or the license that it summarizes does not create a lawyer-client or any other relationship.

Creative Commons is the nonprofit behind the open licenses and other legal tools that allow creators to share their work. Our legal tools are free to use.

- Learn more about our work
- **Learn more about CC Licensing**
- Support our work
- Use the license for your own material.
- Licenses List
- Public Domain List

Footnotes

appropriate credit — If supplied, you must provide the name of the creator and attribution parties, a copyright notice, a license notice, a disclaimer notice, and a link to the material. CC licenses prior to Version 4.0 also require you to provide the title of the material if supplied, and may have other slight differences.
- More info

indicate if changes were made — In 4.0, you must indicate if you modified the material and retain an indication of previous modifications. In 3.0 and earlier license versions, the indication of changes is only required if you create a derivative.
- Marking guide
- More info

technological measures — The license prohibits application of effective technological measures, defined with reference to Article 11 of the WIPO Copyright Treaty.
- More info

exception or limitation — The rights of users under exceptions and limitations, such as fair use and fair dealing, are not affected by the CC licenses.
- More info

publicity, privacy, or moral rights — You may need to get additional permissions before using the material as you intend.
- More info

A.3 Safe Reinforcement Learning for Autonomous Lane Changing using Set-based Prediction

Summary The control input space for autonomous vehicles is typically a continuous space. However, if a set of continuous actions can be easily summarized by one discrete action, this drastically reduces the complexity of an online verification approach. An example is lane changing when driving: There are many trajectories that conform with a lane change, but it is often more important if a lane change is conducted and not how exactly it is performed. Additionally, there are many controllers for autonomous vehicles that can produce drivable trajectories based on a rough reference path.

In this work, we develop an action masking approach for autonomous driving on highways. We regard three discrete high-level actions: keep driving in the current lane and change the lane to the left or right. We use a low-level planner to transform these high-level actions to drivable trajectories. The safety of these trajectories is verified by checking if they intersect with the potential future occupancy of other traffic participants and the road boundary. In particular, we regard legal safety by verifying that the autonomous vehicle keeps a safe distance to the leading vehicle of its current lane, and to the leading and following vehicle in the target lane of a lane change. In case no discrete action is safe, we employ a provably safe adaptive cruise controller. We also examine the effect of action masking on the applied RL algorithm.

We train and test our approach on recorded highway traffic data and compare different architectures for the policy neural network. Our proposed approach always guarantees legal safety, i.e., the RL agent does not cause any collision as it always keeps a safe distance to legally relevant traffic participants. We observe that including the safety verification of actions only slightly reduces the goal-reaching performance and that our action masking approach improves sample efficiency as it converges faster than standard RL.

Author contributions H.K., X.W., and M.A. developed the concept for provably safe RL on highways. H.K. implemented the RL approach and conducted the numerical experiments. X.W. cleaned and transformed the recorded traffic data into CommonRoad scenarios. H.K. and X.W. wrote the manuscript. M.A. provided feedback on the concept and helped improving the manuscript.

TUM Graduate School This publication is *not* a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

Safe Reinforcement Learning for Autonomous Lane Changing Using Set-Based Prediction

Hanna Krasowski*, Xiao Wang*, and Matthias Althoff

Abstract— **Machine learning approaches often lack safety guarantees, which are often a key requirement in real-world tasks. This paper addresses the lack of safety guarantees by extending reinforcement learning with a safety layer that restricts the action space to the subspace of safe actions. We demonstrate the proposed approach using lane changing in autonomous driving. To distinguish safe actions from unsafe ones, we compare planned motions with the set of possible occupancies of traffic participants generated by set-based predictions. In situations where no safe action exists, a verified fail-safe controller is executed. We used real-world highway traffic data to train and test the proposed approach. The evaluation result shows that the proposed approach trains agents that do not cause collisions during training and deployment.**

I. Introduction

Self-driving techniques have the potential to improve mobility in terms of safety and traffic efficiency. One of the most crucial tasks for autonomous vehicles is to plan their motion through traffic without harming other traffic participants. The recent development of motion planning techniques has become more data driven due to the advancement in computation power and the amount of available traffic data. Compared to rule-based methods, data-driven approaches require much less expert knowledge based on the ability to learn complex dependencies from data. Motion planning tasks can be modeled as Markov decision processes, for which reinforcement learning (RL) provides potential solutions. RL's core idea is that an agent learns to interact with the environment by exploring different actions and receiving the next state of the environment and a reward. The exploration process of RL impedes its applicability to real-world problems since unsafe actions are possibly executed.

Various approaches have been proposed to increase the safety of RL methods by modifying the optimality criterion [1], [2] or by verifying the exploration processes with external guidance [3]–[10]. By modifying the optimality objective, agents behave more cautious than those trained without a risk measure included in the objective; however, the absence of unsafe behaviors cannot be proven. In contrast, by verifying the safety of the action and excluding possible unsafe actions, we can ensure that the exploration process is safe. Therefore, we focus on verifying the safety of proposed actions and ensuring safety if the agent fails to find a safe action.

* The first two authors have contributed equally to this work.

All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.

hanna.krasowski@tum.de, xiao.wang@tum.de, althoff@in.tum.de

In this paper, we propose a safe RL framework for motion planning based on our previous work on autonomous lane changing [6]. Our contributions are threefold:

1) We benchmark state-of-the-art model-free RL algorithms to solve high-level behavior planning problems for highway driving.
2) We propose a framework to integrate RL methods in our developed safety layer for autonomous vehicles.
3) We evaluate the proposed approach using a real-world highway traffic dataset.

The remainder of this paper is organized as follows: Section II provides an overview of recent developments in safe RL and safe motion planning techniques. Section III introduces individual modules of our safe RL framework for motion planning. In Section IV, we evaluate the proposed method in real-world highway scenarios. Section V gives the conclusion.

II. Related Work

Safe RL approaches are distinguished in [11] by approaches that modify the optimization criterion and by approaches that modify the exploration process. As previously discussed, only approaches modifying exploration are verifiably safe; thus, we focus on this technique in the subsequent literature review.

A. Modification of Exploration Process

One method to alter the action selection process is to prioritize actions that are estimated to be safer [3]. However, this approach does not prove the nonexistence of unsafe behaviors. Another approach starts with a verified agent model and updates the agent only if the safety requirements are preserved [4], [5]. However, a verified agent model is not always available for complex tasks. A third alternative is to verify which actions are safe and to restrict the action space to safe actions [6]–[10]. However, if all actions are verified as unsafe, safety is no longer guaranteed. To guarantee safety, using the third method, we added a verified fail-safe planner, which holds available a safe action that is activated when the agent fails to identify a safe action.

B. Safety Verification for Autonomous Vehicles

Researchers have proposed various approaches to verify the safety of motion planners for autonomous vehicles. A common method is to predict the most likely motion of other traffic participants [12] or a probability distribution of their future behaviors [13]. The planned trajectories are executed if they do not collide with a traffic participant

according to its prediction. The limitation of this method is that collisions still happen if other traffic participants' behavior deviates from their prediction. In another approach, a minimum requirement for safe motion planning is that inevitable collision states are avoided [14]. A system is in an inevitable collision state if it collides with other traffic participants irrespective of the action taken. However, the calculation of inevitable collision states suffers from the curse of dimensionality. Another possibility is to apply logical reasoning, which uses deduction to prove correct behavior based on given rules [4], [5], [8]. However, logical reasoning is typically not appropriate for online verification, which is required in this work. Furthermore, logical reasoning requires human intervention.

Reachability analysis verifies the safety of planned trajectories by computing all possible future motions of obstacles and checking whether they intersect with the occupancy of the ego vehicle [15]. Since computing the exact reachable sets of nonlinear systems is impossible, reachable sets are over-approximated to ensure safety.

III. Reinforcement Learning with Safety Verification

A. Framework

We build a safe RL framework to tackle the safe lane-changing task by integrating a safety layer between the agent and the environment, as shown in Fig. 1. The safety layer guides the exploration process by restricting the action space to the safe subspace of actions. The task of the agent is to reach a goal area on a multilane highway safely. We define an agent's behavior as safe if it does not cause collisions with other traffic participants.

The safety layer receives the current state of the ego vehicle s_{ego} and the states $s_{\text{obstacles}}$ of surrounding obstacles. Using these states, we predict the possible occupancy areas of the surrounding obstacles. We generate trajectories from high-level actions to check for collisions with the predicted occupancies, thereby determining which high-level actions are safe, as described in Section III-C.

The safe action mask generated by the safety layer restricts the agent's actions to safe actions only, as presented in Section III-E.

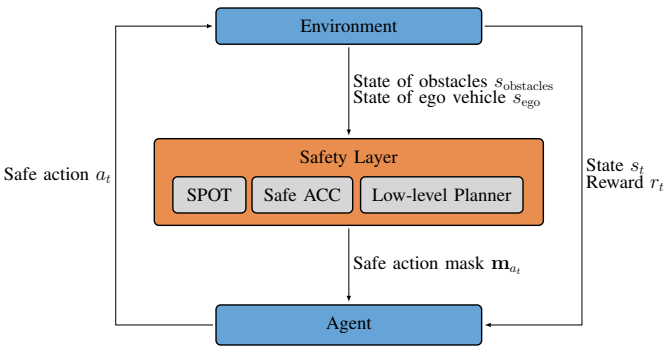


Fig. 1. Reinforcement learning with the safety layer.

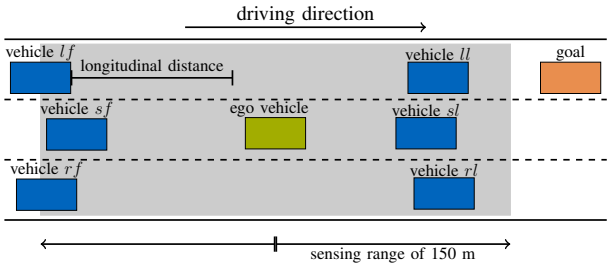


Fig. 2. Schematic representation of a three-lane road with the ego vehicle, surrounding vehicles ij, and the goal area. The gray area depicts the sensing field of the ego vehicle. The obstacle's lane is specified relative to the lane of the ego vehicle by i, i.e., l for the left lane, s for the same lane, and r for the right lane. The relative position of the surrounding vehicle to the ego vehicle is described by j, i.e., l for leading and f for following.

TABLE I

16-dimensional Continuous State Space

Dim.	State	Description
1-6	d_{ij}	The longitudinal distance of surrounding vehicle ij to the ego vehicle (Fig. 2)
7-12	v_{ij}	The relative velocity of surrounding vehicle ij to the ego vehicle
13	v_{ego}	Absolute velocity of the ego vehicle
14	a_{ego}	Absolute acceleration of the ego vehicle
15	d_{long}	Longitudinal distance from ego vehicle to the goal area
16	d_{lat}	lateral distance from ego vehicle to the goal area

B. Markov Decision Process for High-Level Planning

The discrete action space contains the high-level actions for lane-changing decisions: changing to the left lane, changing to the right lane, continuing in the current lane, and staying in the current lane by activating a safe adaptive cruise control (ACC) [16]. The safe ACC is only activated for fail-safe maneuvers since it ensures safety for an infinite time horizon.

The 16-dimensional continuous state space is shown in Tab. I. The agent is provided with the distance to the goal area, as well as the state of the ego vehicle and the surrounding vehicles, to reach a goal area on a highway safely. We consider the relative velocity and longitudinal distance of six surrounding vehicles in a sensing range of $150\,\text{m}$, as illustrated in Fig. 2. Binary variables are introduced below for further derivations:

- $\mathbf{1}_{\text{reach\_goal}} = 1$ when the ego vehicle reaches the goal area.
- $\mathbf{1}_{\text{goal\_lane}} = 1$ when the ego vehicle drives in the lane of the goal area.
- $\mathbf{1}_{\text{collision}} = 1$ if the ego vehicle collides with other vehicles.
- $\mathbf{1}_{\text{safe\_violation}} = 1$ if the ego vehicle violates the safe distance to the leading vehicle or during a lane change to the following vehicle.

We terminate an episode if the time horizon of the current traffic scenario is reached, the goal area is reached, or the ego vehicle collides with another vehicle. The reward function is defined as

$$r = r_{\text{reach\_goal}} + r_{\text{goal\_lane}} + r_{\text{closer}} + r_{\text{crash}} + r_{\text{safe\_dist}}, \quad (1)$$

where each term is further specified as

$$r_{\text{reach\_goal}} = 100 \cdot \mathbf{1}_{\text{reach\_goal}} \tag{2a}$$

$$r_{\text{goal\_lane}} = 5 \cdot \mathbf{1}_{\text{goal\_lane}} \tag{2b}$$

$$r_{\text{closer}} = d_{\text{long}}(t-1) - d_{\text{long}}(t) \tag{2c}$$

$$r_{\text{crash}} = -100 \cdot \mathbf{1}_{\text{collision}} \tag{2d}$$

$$r_{\text{safe\_dist}} = -10 \cdot \left(\frac{d_{\text{safe}}}{d_{ij}} - 1\right) \cdot \mathbf{1}_{\text{safe\_violation}}. \tag{2e}$$

The sparse rewards $r_{\text{reach\_goal}}$ and r_{crash} encourage goal-reaching or collision avoidance behaviors. Additionally, the positive rewards $r_{\text{goal\_lane}}$ and r_{closer} are provided if the ego vehicle gets closer to the goal area in a lateral or longitudinal direction. The penalty $r_{\text{safe\_dist}}$ encourages the agent to keep a safe distance. The safe distance [17] between two vehicles is calculated by

$$d_{\text{safe}} = \frac{1}{2a_{\max}} \left(v_f^2 - v_l^2\right) + v_f \, t_{\text{react}}, \tag{3}$$

where v_l and v_f are the the leading and following vehicles' current velocity, respectively, $t_{\text{react}} = 0.32\,\text{s}$ is the reaction time and $a_{\max} = 11.5\,\text{m/s}^2$ is the maximum deceleration of the vehicles. The value for the reaction time is taken from [17]. The maximum deceleration is based on common values for midsize cars like the BMW 320i [18].

C. Safety Layer

To check whether a high-level action might result in a collision, we compute the future occupancy of the ego vehicle and that of other traffic participants. If both occupancy sets do not intersect for all consecutive time intervals within a predefined time horizon and if the ego vehicle reaches an invariably safe set [19], a collision is impossible. Figure 3 shows an example of a traffic situation with trajectories and occupancies of two surrounding vehicles. If the occupancy of the ego vehicle intersects with the obstacles' occupancy at a certain time step, e.g., as shown in Fig. 3 c), the corresponding high-level action is regarded as unsafe. The occupancies of the surrounding traffic participants are obtained by using our tool SPOT [20]. SPOT considers the physical limits of surrounding traffic participants and constraints implied by traffic rules, e.g., vehicles are not allowed to drive backward on a highway.

The occupancy of the ego vehicle from high-level actions is obtained by a motion planner and the road network structure. For the go-straight action, the ego vehicle follows the center of its current lane. For lane-changing actions, we assume a duration of 2 s. The precise movement is obtained by a sampling-based trajectory planner [21], which requires the total time t_{total}, the final velocity of the trajectory v_{final}, and the lateral deviation from the given reference path $d_{\text{lat\_ref}}$. The intervals from which our planner samples are defined as

- $t_{\text{total}} \in [0.2\,\text{s}, t_{\max}]$,
- $v_{\text{final}} \in [v_{\min}, \max\left(v_{\min}, v_{\text{desired}} + 0.25\,t_{\max}a_{\max}\right)]$, where $v_{\min} = \max\left(0\,\text{m/s},\, v_{\text{desired}} - 0.125\,t_{\max}a_{\max}\right)$,
- $d_{\text{lat\_ref}} \in [-2\,\text{m}, 2\,\text{m}]$.

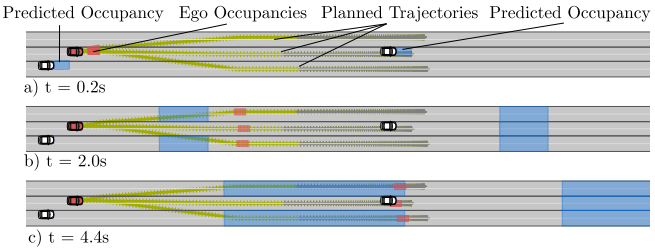


Fig. 3. Example situation for safety verification using set-based prediction [20] and sampling-based trajectory planning [21] with two traffic participants for three time steps. Blue polygons are the predicted occupancies of the surrounding vehicles at specified times. Green lines are the feasible trajectories of the ego vehicle and the gray lines are the appended braking trajectories. The red rectangles symbolize the occupancies of the ego vehicle at the specified time steps.

The desired velocity v_{desired} is defined by the mean velocity necessary for reaching the goal from the initial state in the considered dataset. The planning horizon $t_{\max}$ is set to 2.7 s for go-straight and lane-changing trajectories.

We first exclude meaningless high-level actions that would result in leaving the road, e.g., we exclude changing to the left when driving in the leftmost lane. For each combination of the sampling parameters, a trajectory is generated and checked with the vehicle's kinematic constraints. An optimal trajectory is selected according to the cost function in [21] after the exclusion of kinematically infeasible trajectories. Note that the proposed method can be extended to a continuous action space by converting a sequence of continuous actions to trajectories. We append a braking trajectory with maximum deceleration to the sampled trajectory (cf. Fig. 3). The ego vehicle never follows this braking trajectory but it is utilized to check if the vehicle is in an invariably safe state at the end of its driving trajectory.

If none of the calculated trajectories are considered safe, the fail-safe plan is executed. We utilize the safe ACC from [16] as a fail-safe planner to ensure safety beyond the planning horizon.

D. Selection of the Reinforcement Learning Algorithm

Before integrating the action masking in our policy model, we select the RL algorithm for the goal-reaching task on highways. We benchmark three state-of-the-art RL algorithms with a discrete action space, namely deep Q-network (DQN) [22], actor-critic with experience replay (ACER) [23], and proximal policy optimization (PPO) [24]. DQN [22] is a value-based method where the Q-value is represented by a neural network. The optimal policy is derived from the learned Q-value model. PPO [24] is a policy-gradient method where the policy is represented by a neural network and is directly sampled from the learned model. ACER [23] combines the idea of policy gradient and value-based methods.

We compare the performance of these three methods without the safety layer to exclude its effect on the learning algorithms. We perform a grid search for the hyperparameters for these three methods and select the best hyperparameters for each model. Table II shows that policies trained with PPO

TABLE II

RESULTS OF PRELIMINARY ALGORITHM COMPARISON

Parameter	PPO	ACER	DQN
Reached goal frequency	93.5 %	91.8 %	89.7 %
Elapsed training time	5.76 h	6.41 h	11.71 h
Multiprocessing	True	True	False

reached the goal most often on the test dataset and have the shortest computation time. Based on the implementation in OpenAI [25], PPO and ACER support multiprocessing to decrease the training time, while the DQN algorithm does not support parallelization. Therefore, we select PPO as our learning algorithm.

To be able to differentiate the PPO objective function for discrete action spaces, we apply the Gumbel noise from [26] to the output of the policy network:

$$a(t) = \underset{a_i(t)}{\arg\max} \big[\log(y_i(s_t)) \underbrace{- \log(-\log(u_i))}_{\text{Gumbel noise}} \big], \tag{4}$$

where $\log(y_i(s_t))$ is the output of the policy network corresponding to action a_i, and u_i is a random variable sampled from a uniform distribution $u_i \sim \mathcal{U}[0,1]$. We use $\log(y_i(s_t))$ as the output of the policy network instead of $y_i(s_t)$ because we use a hyperbolic tangent as the activation function.

The optimization objective of PPO $J_{\mathrm{PPO}}(\theta)$ [24] is

$$J_{\mathrm{PPO}}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{\mathrm{C}}(\theta) - v_1 L_t^{\mathrm{VF}}(\theta) \right], \text{with} \tag{5a}$$

$$L_t^{\mathrm{C}}(\theta) = \min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \tag{5b}$$

$$L_t^{\mathrm{VF}}(\theta) = (V_\theta(s_t) - V_t^{\mathrm{targ}})^2, \tag{5c}$$

where the probability ratio $r_t(\theta) = \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}$, $\pi_{\theta_{\mathrm{old}}}$ is the old policy before the update, v_1 and ϵ are scalar hyperparameters, V_θ is the estimated value function, and V_t^{targ} is the target value function collected through Monte Carlo simulations. The operator $\mathrm{clip}()$ limits the first argument to the range which is defined by the following two arguments, and $\hat{\mathbb{E}}_t[...]$ is the empirical mean over a finite batch of samples. We denote by θ the trainable parameters of the network. The main term of the objective is the clipped objective $L_t^{\mathrm{C}}(\theta)$, which is easier to implement than using the Kullback Leibler divergence while showing stability comparable to trust-region-based methods. The value loss $L_t^{VF}(\theta)$ is necessary because parameters between policy network and value function network are shared. The advantage function $\hat{A}_t$ is estimated by a general advantage estimator [27].

The gradient of the PPO objective $J_{\mathrm{PPO}}(\theta)$ is obtained by differentiation with respect to the trainable parameters θ, which is derived as in [27]:

$$\nabla_\theta J_{\mathrm{PPO}}(\theta) = \hat{\mathbb{E}}_t \big[\nabla_\theta \log\big(\pi_\theta(a_t|s_t)\big) J_{\mathrm{PPO}}(\theta) \big], \tag{6}$$

where ∇_θ denotes the gradient with respect to the trainable parameters.

E. Action Masking

The safety layer generates a mask for the action to restrict the action space to the safe subspace. The safety mask $\mathbf{m}_{a_i}$ is a binary vector defined as

$$\mathbf{m}_{a_i}(t) = \begin{cases} 1, & \text{if } a_i(t) \text{ is verified safe} \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

When we insert the safety mask (7) in (4), we obtain

$$\begin{aligned} a(t) &= \underset{a_i(t)}{\arg\max} \left[\log(y_i(s_t)) \mathbf{m}_{a_i(t)} - \log(-\log(u_i \, \mathbf{m}_{a_i(t)})) \right] \\ &= \underset{a_i(t) \in \mathcal{A}_{\mathrm{safe}}}{\arg\max} \big[\log(y_i(s_t)) - \log(-\log(u_i)) \big], \end{aligned} \tag{8}$$

where $\mathcal{A}_{\mathrm{safe}}$ is the set of actions which are verified as safe by the safety layer. The second line of (8) is derived from the fact that $0 - \log(-\log(0)) = -\infty$, i.e., unsafe actions can never be selected by the $\arg\max$ operator.

In the following, we show that masking does not affect the PPO objective and its gradient. All variables in (5) associated to masking are denoted by $\square^{\mathrm{m}}$. First, we obtain the objective with masking $J_{\mathrm{PPO}}^{\mathrm{m}}(\theta)$ based on (5a):

$$J_{\mathrm{PPO}}^{\mathrm{m}}(\theta) = \hat{\mathbb{E}}_t \big[L_t^{\mathrm{C,m}}(\theta) - v_1 L_t^{\mathrm{VF}}(\theta) \big], \tag{9}$$

where $L_t^{\mathrm{C,m}}(\theta)$ is the clipped objective, and $L_t^{\mathrm{VF}}(\theta)$ is the value function loss (see (5c)). The masking of actions does not alter the value function loss term $L_t^{\mathrm{VF}}(\theta)$ because the value function describes the value of a specific state independent of the actions. The clipped objective $L_t^{\mathrm{C,m}}(\theta)$ is based on (5b) and is specified as

$$L_t^{\mathrm{C,m}}(\theta) = \min(r_t^{\mathrm{m}}(\theta)\hat{A}_t, \mathrm{clip}(r_t^{\mathrm{m}}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \tag{10}$$

where

$$r_t^{\mathrm{m}}(\theta) = \frac{\pi_\theta^{\mathrm{m}}(a_t, \mathbf{m}_{a_t}|s_t)}{\pi_{\theta_{\mathrm{old}}}^{\mathrm{m}}(a_t, \mathbf{m}_{a_t}|s_t)} = \frac{\pi_\theta^{\mathrm{m}}(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}^{\mathrm{m}}(a_t|s_t)}. \tag{11}$$

The policy values π_θ^{m} and $\pi_{\theta_{\mathrm{old}}}^{\mathrm{m}}$ for safe actions are not modified as the masking yields that a_t is always a safe action. Therefore, $r_t^{\mathrm{m}}(\theta)$ stays the same as the initial (see r_t). The estimated advantage $\hat{A}_t$ depends on the value function and is not affected by the masking. Thus, the action masking does not affect the clipped objective $L_t^{\mathrm{C,m}}(\theta)$ and the PPO objective $J_{\mathrm{PPO}}^{\mathrm{m}}(\theta)$. Consequently, the gradient $\nabla_\theta J_{\mathrm{PPO}}^{\mathrm{m}}(\theta)$ also remains the same as defined in (6).

IV. EXPERIMENTS

We demonstrate the proposed approach using a real-world highway dataset. To show the safety layer's effect on an RL agent, we train two groups of agents with and without the safety layer, respectively. Furthermore, we investigate the impact of different neural network structures. We evaluate the agents' performance by comparing their learning curves during training as well as the collision rate and goal-reaching rate on a test set.

A. Simulation Environment

a) Dataset: We utilize the highD dataset [28] to train and test the proposed approach. This dataset includes real traffic scenarios of German highways from six different locations with three-lane and two-lane roads. The dataset includes $5.1\,\mathrm{h}$ of recorded traffic with a time step size of $0.04\,\mathrm{s}$. The scenarios' duration ranges from $12.45\,\mathrm{s}$ to $12.67\,\mathrm{s}$ in the $95\,\%$ confidence interval. The scenarios' duration is similar because the observed road length is the same for the six locations, and the scenarios were generated from the original data. We generated tasks by removing a vehicle from the recorded data and using its start and the final state as the initial state and the center of the goal region, respectively. In particular, the goal area is the occupancy of the removed vehicle at its final position. The goal is reached if the ego vehicle intersects with the goal area. We randomly split the dataset into $80\,\%$ training set and $20\,\%$ test set.

b) Policy Network: We conducted experiments with two different types of policy networks, namely, a multi-layer perceptron (MLP) network [29] and a long short-term memory (LSTM) network [30]. The hyperparameters of the policy networks are determined using a grid search. The hyperparameter search compares the convergence and final rewards on the training set. The MLP network consists of three hidden layers, with 128 neurons in each layer. We choose an LSTM network as the second type because the task is time-sequential and recurrent networks are well suited to solve sequential tasks. However, training an MLP network is more stable as it converges for a larger variety of hyperparameters. The LSTM network consists of 128 neurons, and layer normalization is applied. Furthermore, we compute the running mean and standard deviation of the states to normalize the state space for both policy networks.

c) Training Mode: We conducted the training in two modes:

- Safe mode: we train the agent as proposed in Fig. 1.
- Non-safe mode: we exclude the safety layer.

In both modes, we restrict the action space to the corresponding high-level lane change action in case a lane change is currently conducted. Thereby, we ensure that a lane change cannot be prematurely aborted. A lane change is considered finished when the orientation of the ego vehicle differs at most by $0.2\,\mathrm{rad}$ from the orientation of the target lane, and the center of the ego vehicle is closer than one-fourth of the lane width to the centerline of the target lane.

B. Results

We trained the MLP and LSTM agents in safe and non-safe mode, resulting in four different agents. We compared these agents with respect to training performance, safety during training, and testing performance. Furthermore, we evaluated the effect of the safety layer on the agent using the test dataset.

a) Training Performance: Figure 4(a) shows the reward curves, which reveals that training in safe mode leads to faster convergence. Notably, training the agents in the non-safe mode required three million training steps; in contrast,

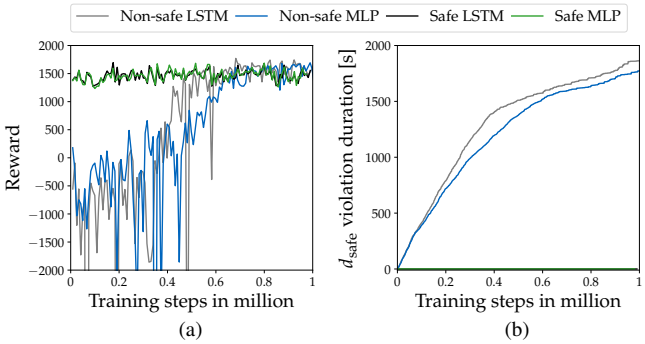


Fig. 4. Training results: (a) Reward curves for trained agents, (b) Safe distance violation for trained agents

one million steps were enough to train the agents in safe mode. The high negative rewards for non-safe mode agents originate from scenarios where the agent maintains a safe distance to the leading vehicle or changes lane close to the following vehicle on the target lane. The penalty for a safe distance violation with respect to surrounding vehicles is computed using (2e). Therefore, the penalty reaches high values if the agent almost collides with a traffic participant ahead.

Another indicator of how well the agents are exploring the action space is the number of lane changes per traffic scenario. Initially, the lane change frequency for training is about one lane change per traffic scenario for the safe agents and five for non-safe agents. The lower lane change frequency for the safe agents at the beginning of the training is due to the restriction of the action space. During training, the lane change frequency converges to about 0.2 lane changes per scenario, which means that the agent performs one lane change in every five scenarios with an average scenario duration of $12\,\mathrm{s}$. This convergence is significantly faster for the agents trained in safe mode. Note that the original data in the highD dataset has 0.1 lane changes per scenario, potentially caused by the low traffic density. Thus, lane change behaviors are not necessary in most scenarios.

Comparing the network types on the training set, the performance of MLP agents is almost identical to the corresponding LSTM agents. In particular, for the non-safe agents, the training converges marginally faster for the LSTM agent than for the MLP agent. For safe agents, there exists no visible difference in training convergence. For all agents, utilizing an MLP network leads to reaching the goal marginally more often.

b) Safety during Training: We have to differentiate between collisions for which the ego vehicle is responsible and collisions that occur because no interaction between traffic participants was considered due to prerecorded data. We exclude scenarios with collisions not caused by the ego vehicle from our evaluation, e.g., another vehicle colliding with the rear of the ego vehicle. Furthermore, the ego vehicle considers the safe distance to the leading vehicle and the following vehicle after a lane change.

During training, the non-safe agents caused collisions with

TABLE III

FINAL EPISODE STATUS ON THE TEST DATASET

Agent	Collision	Reached goal
Non-safe LSTM	1.3 %	95.0 %
Non-safe MLP	0.8 %	**97.1 %**
Safe LSTM	**0.0 %**	87.5 %
Safe MLP	**0.0 %**	75.4 %

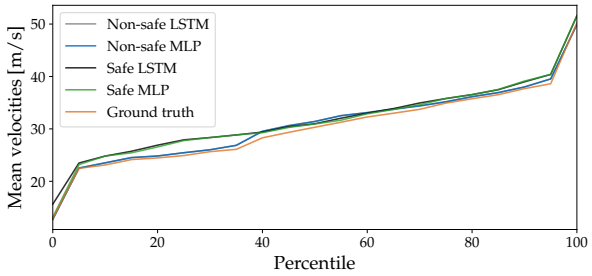


Fig. 5. Percentile curve for mean velocities on test scenarios

other traffic participants, while the safe agents did not cause any collisions. Moreover, the non-safe agents reached the goal more frequently than safe agents. However, the safe agents reach the goal in about 80 % of the scenarios in the last 500 000 training steps.

We measure safety using the duration for which the agents violated the safe distance. Figure 4(b) shows the duration of safe distance violations during training. The non-safe agents violate the safe distance while the safe agents never violate the safe distance.

c) Testing Performance: Table III summarizes the results of testing the agents. The test dataset's performance is similar to the performance of the training dataset, indicating that the agents are not overfitted. In the test set, the non-safe agents reach the goal more frequently than the safe agents. Moreover, the non-safe agents cause collisions in contrast to the safe agents. The performance of both policy networks is very similar in general. Based on the frequency of reaching the goal, the safe LSTM agent performs better than the safe MLP agent. This performance might be due to the ability of LSTM to store temporal features, e.g., acceleration of surrounding vehicles, providing additional information for planning a safe motion.

Furthermore, to show that the safe models do not drive too conservatively and impede the traffic flow, we evaluated the behavior of the agents against the original human driver trajectory from which the initial state and goal area of the task were generated. In general, the trained agents show a behavior similar to the original driver. Figure 5 depicts the percentile curve of the mean velocity. The mean velocities for the original human driver and all trained agents are almost identical. By comparing the safe distance violations to the leading vehicle, we observe that the original driver violated the safe distance, and the non-safe agents violated the safe distance even more. In contrast, safe agents did not violate any safe distances during testing.

d) Effect of Safety Layer on Learning: We tested the agents trained in safe mode also in non-safe mode to detect if training with the safety layer leads to better-performing agents. The comparison of the agents' performance in non-safe mode shows that the agents trained in safe mode perform worse than the agents trained in non-safe mode. The goal-reaching rate is 25 % less for the safe LSTM and 62 % less for the safe MLP agent than for agents trained in non-safe mode. Moreover, the agents trained in safe mode collided in more test scenarios (23 % for LSTM and 27 % for MLP). In contrast, the agents trained in non-safe mode caused collisions in about 1 % of the test scenarios. This performance is because the agents trained in safe mode did not experience dangerous situations with high penalties during training and cannot solve them in the non-safe test setting. Thus, the safety layer is necessary during deployment to ensure safety.

If training is conducted in a simulation setting and not in the real world, safety guarantees for real-world deployment would often suffice. Therefore, we tested the non-safe agents in safe mode and compared them to the agents trained and tested in safe mode. The performance of agents trained in non-safe mode and the safe LSTM agent on the test set is almost identical because all the agents reached the goal in 87 % of the scenarios. The safe MLP agent performs marginally worse as it only reaches the goal in 75 % of the scenarios. Due to the safety layer, none of the agents cause collisions in the test set. The result shows that the agents can be trained in non-safe mode and deployed in safe mode without causing performance loss and safety reduction. However, training in safe mode converges faster, which is a reason for training in safe mode.

C. Discussion

Although the proposed approach guarantees safety in all scenarios, the agent drives more cautiously than normal drivers, especially in dense traffic. In such scenarios, the predicted occupancy leads to a comparably small free space for the agent to drive. In this type of scenario, the interaction between traffic participants is essential. A traffic simulator can predict interactions between the agent and traffic participants to a certain degree.

Moreover, accurate modeling of physical parameters is crucial for set-based predictions. Too large physical bounds lead to significant over-approximation errors, limiting model applicability. Simultaneously, too small physical bounds cause inaccurate prediction that does not enclose the actual behavior, leading to unsafe behaviors. To check whether the modeled physical parameters over-approximate the real behavior, one can perform a reachset conformance test as shown for a pedestrian model in [31].

Due to the computational overhead for determining safe actions, the computation time for training safe agents is 16 times higher than for the non-safe agents. The average training step for safe agents takes 5.46 s and 0.112 s for non-safe agents. This significant increase in the training time is mainly because instead of one trajectory for the selected

action, all possible trajectories are generated and compared to the predicted occupancies of traffic participants. Optimizing the current implementation is necessary to benefit from the faster convergence of safe agents in order to safeguard machine learning in real vehicles.

V. Conclusions

In this paper, we present a framework for safeguarding an RL agent using a safety layer to verify whether the proposed actions are safe and provide a provably safe fail-safe controller. Safe actions are determined by set-based prediction, which considers all possible motions of traffic participants. We evaluated the proposed approach using a real-world highway dataset. The result of the evaluation shows that the trained policy does not cause any collisions. Furthermore, the safe agent's ability to reach the goal region is comparable to that of non-safe agents. The proposed approach only requires an additional navigation system to realize basic motion planning on highways.

Acknowledgment

The authors gratefully acknowledge the partial financial support of this work by the German Research Foundation Grant AL 1185/3-2 and the research training group CONVEY funded by the German Research Foundation under grant GRK 2428.

References

[1] B. Lütjens, M. Everett, and J. P. How, "Safe reinforcement learning with model uncertainty estimates," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8662–8668.

[2] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2034–2039.

[3] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Safe reinforcement learning with scene decomposition for navigating complex urban environments," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1469–1476.

[4] N. Fulton and A. Platzer, "Verifiably safe off-model reinforcement learning," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2019, pp. 413–430.

[5] S. Pathak, L. Pulina, and A. Tacchella, "Verification and repair of control policies for safe reinforcement learning," *Applied Intelligence*, vol. 48, no. 4, pp. 886–908, 2018.

[6] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2156–2162.

[7] D. Isele, A. Nakhaei, and K. Fujimura, "Safe reinforcement learning on autonomous vehicles," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–6.

[8] G. R. Mason, R. C. Calinescu, D. Kudenko, and A. Banks, "Assured reinforcement learning for safety-critical applications," in *Doctoral Consortium at the 10th International Conference on Agents and Artificial Intelligence*, 2017.

[9] A. Akametalu, S. Kaynama, J. Fisac, M. Zeilinger, J. Gillula, and C. Tomlin, "Reachability-based safe learning with Gaussian processes," in *IEEE Conference on Decision and Control*, 2015, pp. 1424–1431.

[10] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[11] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 42, pp. 1437–1480, 2015.

[12] F. Altché and A. de La Fortelle, "An LSTM network for highway trajectory prediction," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 353–359.

[13] T. Gindele, S. Brechtel, and R. Dillmann, "Learning driver behavior models from traffic observations for decision making and planning," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 69–79, 2015.

[14] T. Fraichard and H. Asama, "Inevitable collision states a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[15] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[16] M. Althoff, S. Maierhofer, and C. Pek, "Provably-correct and comfortable adaptive cruise control," *IEEE Transactions on Intelligent Vehicles*, 2020.

[17] M. Althoff and R. Lösch, "Can automated road vehicles harmonize with traffic flow while guaranteeing a safe distance?" in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 485–491.

[18] M. Althoff, "CommonRoad: Vehicle models." [Online]. Available: https://commonroad.in.tum.de/

[19] C. Pek and M. Althoff, "Efficient computation of invariably safe states for motion planning of self-driving vehicles," in *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 3523 – 3530.

[20] M. Koschi and M. Althoff, "SPOT: A tool for set-based prediction of traffic participants," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1686–1693.

[21] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét frame," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 987–993.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," in *Twenty-seventh Conference on Neural Information Processing Systems – Workshop on Deep Learning*, 2013.

[23] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," in *International Conference on Learning Representations (ICLR)*, 2017.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[25] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "OpenAI baselines," 2017. [Online]. Available: https://github.com/openai/baselines

[26] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-softmax," in *International Conference on Learning Representations (ICLR)*, 2016.

[27] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *4th International Conference on Learning Representations, ICLR*, 2016.

[28] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2118–2125.

[29] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, 2007.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts and London, England: MIT Press, 2016.

[31] S. B. Liu, H. Roehm, C. Heinzemann, I. Lütkebohle, J. Oehlerking, and M. Althoff, "Provably safe motion of mobile robots in human environments," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep 2017.

RightsLink

Sign in/Register

Safe Reinforcement Learning for Autonomous Lane Changing Using Set-Based Prediction

IEEE Requesting permission to reuse content from an IEEE publication

Conference Proceedings:
2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)

Author: Hanna Krasowski

Publisher: IEEE

Date: 20 September 2020

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

A.4 Safe Reinforcement Learning for Urban Driving using Invariably Safe Braking Sets

Summary The majority of provably safe RL research that considers the application of autonomous driving is developed for highway driving situations with no intersections. Yet, to leverage the full potential of autonomous driving, urban traffic situations have to be considered as the majority of car travel is short distance. Whereas for highway driving a discrete action space only needs to contain few high-level decisions, a discrete action space for urban traffic situations has to consider more actions.

In this work, we regard the task of provably safe urban intersection driving. We design a 63-dimensional action space based on the combination of three action types: routing actions, i.e., routing decisions at an intersection, acceleration actions, i.e., desired accelerations along a reference path, and lane actions, i.e., lane keeping and lane changing. The RL agent decides for one action of each action type and a low-level controller computes the corresponding drivable trajectory. We verify the legal safety of trajectories based on invariably safe braking sets and regard two safety factors: safety in lanes with the same driving direction and safety for passing intersections. Safety in lanes is specified as keeping a safe distance to the leading vehicle and to the following and leading vehicle in a target lane of a lane change. Safe passing of intersection is defined as entering intersections only if no other traffic participants could occupy the intersection areas that are also occupied by the trajectory of the autonomous vehicle. If we cannot verify any discrete action as safe, we employ a fail-safe planner that stops the vehicle outside of intersections or drives the vehicle off the intersection as fast as possible.

We evaluate our action masking approach on recorded traffic data from three German intersections. In contrast to standard RL, our approach does not cause collisions during training and deployment. Additionally, we conduct an ablation study for the safety factors by integrating either safety on intersections or safety in lanes. While safety in lanes improves the goal-reaching performance, safety on intersections leads to a significant drop mainly due to our conservative definition of safety on intersections, which does not regard right-of-way rules.

Author contributions H.K. and Y.Z. developed the concept for provably safe RL for intersection driving. Y.Z. implemented the RL approach. H.K. and Y.Z. designed, conducted, and evaluated the numerical experiments. H.K. and Y.Z. wrote the manuscript. M.A. provided feedback improving the manuscript.

TUM Graduate School This publication has been declared a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

Safe Reinforcement Learning for Urban Driving using Invariably Safe Braking Sets

Hanna Krasowski*,[1], Yinqiang Zhang*,[2], and Matthias Althoff[1]

***Abstract*— Deep reinforcement learning (RL) has been widely applied to motion planning problems of autonomous vehicles in urban traffic. However, traditional deep RL algorithms cannot ensure safe trajectories throughout training and deployment. We propose a provably safe RL algorithm for urban autonomous driving to address this. We add a novel safety layer to the RL process to verify the safety of high-level actions before they are performed. Our safety layer is based on invariably safe braking sets to constrain actions for safe lane changing and safe intersection crossing. We introduce a generalized discrete high-level action space, which can represent all high-level intersection driving maneuvers and various desired accelerations. Finally, we conducted extensive experiments on the inD dataset containing urban driving scenarios. Our analysis demonstrates that the safe agent never causes a collision and that the safety layer's lane changing verification can even improve the goal-reaching performance compared to the unsafe baseline agent.**

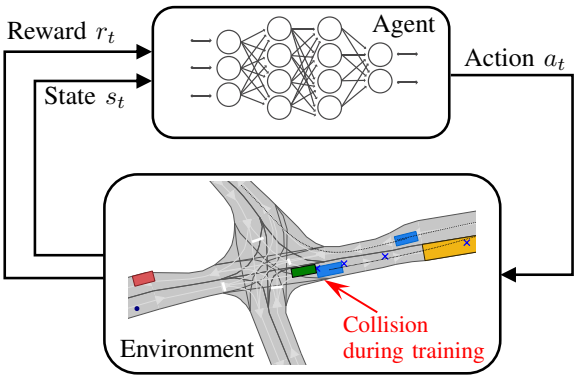


Fig. 1. Conventional RL process including an example collision situation.

I. Introduction

Motion planning in urban areas is challenging because of different road geometries and frequent interactions with traffic participants. A method suited explicitly for solving such tasks is reinforcement learning (RL) [1], [2]. With deep RL algorithms, vehicles can learn to control their motion for different tasks, such as lane-keeping and changing [3], [4], path tracking [5], [6], ramp merging [7], [8], navigating through intersections [9]–[12], and emergency braking [13], [14]. However, most deep RL approaches only focus on one simplified driving sub-tasks. Moreover, learning a driving policy with conventional deep RL is inherently unsafe. As shown in Fig. 1, the stochastic exploration process possibly guides the vehicle to unsafe states, where causing a collision cannot be avoided anymore. Furthermore, frequent visits to unsafe and meaningless states can decrease learning efficiency. To mitigate this problem, the exploration of RL agents can be directed or constrained. Lu et al. [15] designed risk networks that can guide a safe policy optimization. However, their approach cannot guarantee safety during driving. Another method is using control barrier functions [16], [17]; however, finding suitable control barrier functions for complex tasks, for example, urban autonomous driving, is not trivial.

To efficiently guarantee safety for an RL agent, we propose a safe RL algorithm for autonomous driving in urban scenarios, where safe actions are identified by a safety layer and the action selection of the RL agent is constrained such that only safe actions can be selected. Our work generalizes our pre-study on highway driving [18] so that it is also applicable in an urban setting. Notably, our contributions are as follows:

- By combining invariably safe braking sets and conflict zones, we introduce a safety layer that can verify the safety of junction crossing.
- We propose a generalized high-level action space to solve various driving tasks in urban scenarios.
- We conducted extensive numerical experiments and an ablation study to show the validity and efficiency of our implementation.

The remainder of this paper is structured as follows: Section II shows the related literature on RL algorithms for autonomous driving and the safety guarantees of RL algorithms. Section III describes the details of our proposed algorithms, particularly the safety layer. Section IV records the experimental settings, results of conducted experiments, and an ablation study. Finally, we conclude in Section V.

II. Related Work

RL research on motion planning for autonomous driving mainly differs in tasks and action specifications. Usually either high-level actions [19]–[21] or direct control inputs [5], [9], [17], [22] are learned, which differ depending on the regarded driving scenario. Furthermore, only some researchers tried to incorporate safety measures. We review current research on RL for autonomous driving and methods for extending RL to safe RL.

*The first two authors have contributed equally to this work.

[1]Hanna Krasowski and Matthias Althoff are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany, {hanna.krasowski, althoff}@tum.de

[2]Yinqiang Zhang is with the Department of Computer Science, University of Hong Kong, Hong Kong, China zyq507@connect.hku.hk

A. Reinforcement Learning for Autonomous Driving

To solve various driving tasks in urban scenarios, the action space should be appropriately designed. One RL approach is applying learned actions directly to the ego vehicle. With this end-to-end approach, the agent chooses an action value from a continuous action space, such as a speed set-point and steering angle [5], a velocity [22], an acceleration [9], or a yaw rate [4]. For these continuous action spaces, the agent often needs more training steps to learn the optimal policy because of the infinite number of action values that can be explored.

Other RL approaches use a discrete high-level action space, where actions typically represent different maneuvers. For instance, maneuvers [19], [23] are a commonly used action representation for lane keeping and changing tasks. A three-layer architecture for the lane-changing and left-turning tasks was recently proposed by Qiao et al. [20]. The top-level policy chooses a maneuver. An optimal trajectory is then created and tracked by a PID controller. Isele et al. [21] proposed three discrete action spaces for driving at intersections. They evaluated their approach on simulated traffic and found that the action space with a creep action (i.e., moving slowly) performs the best with occlusions near the intersection area. Li et al. [12] proposed a hierarchical framework with a high-level action space consisting of reference speeds and low-level controllers for intersection and round-about driving. Their evaluation shows that their approach can achieve high completion rates but causes more collisions than more conservative approaches. Many maneuvers such as lane following, lane changing, and intersection crossing have to be regarded in urban areas. A discrete action space can be used to efficiently learn in such a complex environment.

B. Provably Safe Reinforcement Learning

To guarantee safety, the agent's exploration must be limited to the safe state space. For that, two approaches are most relevant: advising the agent after the action selection with a possibly adapted safe action or constraining actions to safe actions before the agent can choose one [24]. For the first approach, usually, a penalty is given in case a correction is necessary [16], [17], [25], [26]. Saunders et al. [26] proposed a trained human-like supervisor in their RL algorithm to intervene in the agent's behavior when it tends to go into unsafe or risky states. Cheng et al. [16] presented an end-to-end safe RL algorithm, where control barrier functions restrict exploration and deployment. Similarly, Wang [17] proposed control barrier functions to achieve end-to-end safe RL for autonomous highway driving.

The second approach removes unsafe and meaningless actions in advance [9], [18], [27]–[29]. Only actions that entirely satisfy safety specifications are accessible to the agent. For instance, the methods in [9], [28] ensure safe intersection navigation by verifying safety with linear temporal logic and differential dynamic logic. Additionally, Q-masking removes meaningless and unsafe actions for Q-learning [23], [29]. For example, Mirchevska et al. [29] used the safe braking distance to decide, which actions are unsafe and need to be masked out. Krasowski et al. [18] built on this work and present a safety layer for highway driving, which generates safe action masks for the proximal policy optimization (PPO) algorithm [30]. They use set-based predictions [31] for the other traffic participants to identify safe actions. In this work, we build on the masking approach, which allows us to ensure safety by identifying safe actions in advance of their execution.

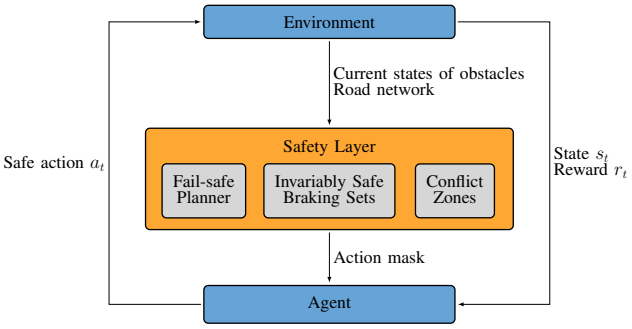


Fig. 2. Implementation overview of RL framework.

III. Safe Reinforcement Learning in Urban Environments

RL problems can be formulated as Markov decision processes, which is illustrated in Fig. 1. Our safe RL framework is extended by a safety layer (see Fig. 2). This safety layer generates an action mask that indicates the safe actions and removes unsafe and meaningless actions, e.g., actions that would lead off-road or actions that violate safe distances to other traffic participants. As a result, the agent can explore only safe actions. We use PPO with action masking as our learning approach and refer the interested reader to [18], [32] for implementation and theoretical details. The observation space, action space, and reward function employed in this work are first introduced in the following parts. The safety layer, which incorporates the concept of conflict zones [33] and invariably safe sets [34], is then thoroughly explained.

A. Observation Representation

Our 40-dimensional continuous state space consists of 26 observations from CommonRoad-RL [35] and 14 new intersection-related observations (see Table I). To define the intersection-related observations, we need to specify the intersection area, which is the area that is mutually accessible to vehicles arriving at the intersection from different entries. Furthermore, intersection-entering vehicles are those for which the following hold:

- the position is at most the longitudinal distance $s_{\text{intersection}}$ away from the intersection,
- the vehicle is not a lane-based surrounding vehicle [35], i.e., not surrounding the ego vehicle on the ego vehicle's lane or on adjacent lanes in the same driving direction,
- the vehicle is driving toward the intersection.

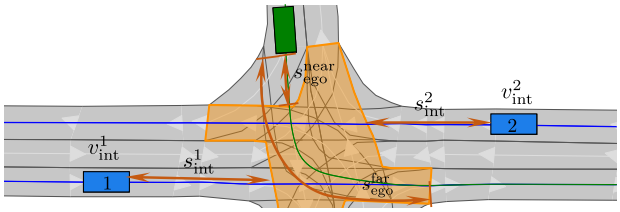


Fig. 3. Visualization of intersection-related observations: Relative distances $s_{\text{int}}^1, s_{\text{int}}^2$ between the vehicle $1, 2$ and the orange intersection, absolute velocities v_{int}^1 and v_{int}^2 of vehicle $1, 2$, and ego vehicle distances to intersection $s_{\text{ego}}^{\text{near}}$ and $s_{\text{ego}}^{\text{far}}$. The driving directions are indicated by light gray arrows.

The intersection-related observations are illustrated in Fig. 3. The first intersection-related observations are the absolute velocities v_{int}^i and relative distances s_{int}^i to the intersection for the intersection-entering vehicles $i \in 1, ..., n_{\text{intersection}}$. The observations are sorted based on the vehicles' distances to the intersection so that only the $n_{\text{intersection}}$ closest vehicles are considered. If fewer vehicles are detected, we set the relative distance and velocity to the predefined values $s_{\text{intersection}}$ (here $50\,\text{m}$) and $0\,\text{m}\,\text{s}^{-1}$, respectively. The remaining intersection-related observations are the longitudinal distances along the reference lane between the ego vehicle position and the intersection (cf. $s_{\text{ego}}^{\text{near}}$ and $s_{\text{ego}}^{\text{far}}$ in Fig. 3).

B. Action Representation

Driving in urban traffic requires various maneuvers. We used a two-level framework to represent this. The policy chooses the maneuver on the higher level, and the sampling-based motion planner from [36] concretizes the maneuver to a drivable trajectory on the lower level. The high-level action space consists of three action types. The first action type a_{lane} indicates maneuvers restricted to the current and adjacent lanes, i.e., change to left ($a_{\text{lane}} = 0$), or right lane ($a_{\text{lane}} = 2$), and keep driving in the current lane ($a_{\text{lane}} = 1$). The second action type a_{dir} can take three values and describes the driving directions at the next intersection, for example, turn left, right, or go straight. Note that if there is just one possible direction for driving, only $a_{\text{dir}} = 0$ is used; if there are two possible driving directions, $a_{\text{dir}} = 0$ corresponds to the left-most action and $a_{\text{dir}} = 1$ to the other. The third action type a_{acc} represents the desired longitudinal accelerations. Seven values can be selected: $\mathcal{A}_{acc} = \{0\,\text{m}\,\text{s}^{-2}, \pm 1.0\,\text{m}\,\text{s}^{-2}, \pm 2.0\,\text{m}\,\text{s}^{-2}, \pm 4.0\,\text{m}\,\text{s}^{-2}\}$. All possible combinations of the three action types ($a_{\text{lane}} \times a_{\text{dir}} \times a_{\text{acc}}$) lead to the action set $\mathcal{A}_{\text{regular}}$, which represent the regular maneuvers possible at an at most four-legged intersection. These 63 regular actions plus the fail-safe action lead to a 64-dimensional discrete action space $\mathcal{A}$. More complex intersections can be represented by extending the possible values for a_{lane} and a_{dir}.

TABLE I

40-DIMENSIONAL CONTINUOUS STATE SPACE

Dim.	Description
1-6	distance between ego vehicle and six lane-based surrounding traffic participants
7-12	velocity between ego vehicle and six lane-based surrounding traffic participants
13-14	velocity and acceleration of the ego vehicle
15-16	longitudinal distance and motion advance to goal area
17-18	lateral distance and motion advance to goal area
19-23	lateral distances from dynamically extrapolated ego vehicle positions to goal
24	remaining time steps to reach the goal area
25	orientation of goal area
26	remaining time steps in scenario
27-28	longitudinal distances to intersection area ($s_{\text{ego}}^{\text{near}}$, $s_{\text{ego}}^{\text{far}}$)
29-34	distance between intersection and six traffic participants ($s_{\text{int}}^{\text{i}}$ for $i = 1, ..., 6$)
35-40	velocity between ego vehicle and six lane-based surrounding traffic participants ($v_{\text{int}}^{\text{i}}$ for $i = 1, ..., 6$)

Note that the upper 26 observations are implemented as in [35] and the remaining 14 observations are derived in Sec. III-A.

C. Reward Function

We use sparse and dense components for the reward function. The sparse components are:

$$\begin{aligned} r_{\text{reach\_goal}} &= 50 \cdot \mathbf{1}_{\text{reach\_goal}}, \\ r_{\text{time\_out}} &= -10 \cdot \mathbf{1}_{\text{time\_out}}, \\ r_{\text{collision}} &= -50 \cdot \mathbf{1}_{\text{collision}}, \\ r_{\text{mask}} &= -10 \cdot \mathbf{1}_{\text{mask}}, \end{aligned}$$

where $\mathbf{1}_{\square}$ denotes binary variables that evaluate to 1 if the corresponding condition $\square$ is satisfied. Particularly, the reward r_{mask} is given when no regular action is verified as safe and the fail-safe planner is activated. The dense reward component guides the agent toward the goal at each time step:

$$r_{\text{goal\_guiding}} = \frac{-40 \cdot \Delta d_{\text{lat}}^t + 20 \cdot \Delta d_{\text{lon}}^t}{d_{\text{lon}}^{\text{total}}}, \tag{1}$$

where Δd_{lat}^t and Δd_{lon}^t are the position differences toward the goal in the longitudinal and lateral directions within a curvilinear coordinate system (introduced in Sec. III-D) at time step t compared to the previous time step. To reduce the influence of different distances between the initial state and the goal, we divide by the longitudinal distance $d_{\text{lon}}^{\text{total}}$ from the initial position to the goal. The final reward function is:

$$\begin{aligned} r = &r_{\text{reach\_goal}} + r_{\text{time\_out}} \\ &+ r_{\text{collision}} + r_{\text{mask}} + r_{\text{goal\_guiding}}. \end{aligned} \tag{2}$$

D. Preliminaries and Assumptions for the Safety Layer

The road network consists of lanelets [37], which are atomic, interconnected, and drivable road segments. We condensate the road network into a set of lanes $\mathcal{L}$. A lane is defined as a set of longitudinally adjacent lanelets from a lanelet that has no predecessor to a lanelet that has no successor [38]. We assign unique identifiers to the lanes and use $\mathcal{L}_k$ to specify the occupancy of the lane with the identifier k. In addition, $\mathcal{K}$ is the set of identifiers of all lanes in the scenario.

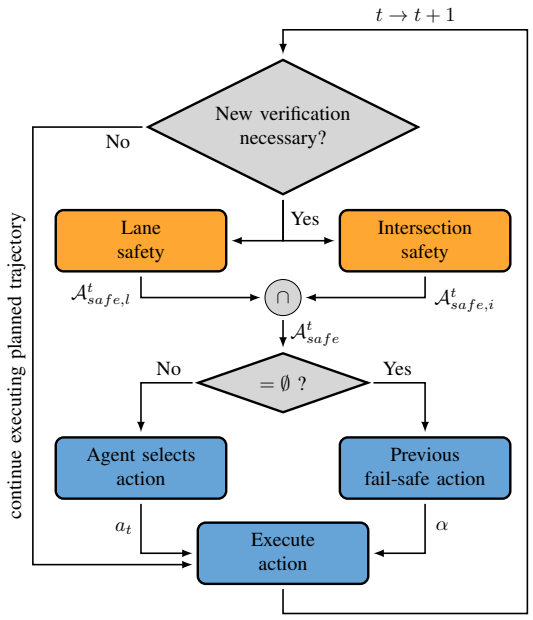


Fig. 4. Safety verification flowchart. Orange blocks belong to the safety layer and blue ones to the RL agent and environment.

We use a curvilinear coordinate system along the lanes such that the ego vehicle's state at each time step t is $x_t = (s, d, v)$, where s is the longitudinal position along the lane, d is the lateral position, and v is the velocity. The function $\text{proj}_s(x_t)$ returns the longitudinal position for a state x_t. The function plan creates the set of ego vehicle states $\{(x_{ego,0}, 0), ..., (x_{ego,t_p}, t_p), ..., (x_{ego,t_f}, t_f)\}$ for all time steps until the final time t_f. The function uses the sampling-based planner from [36] to generate the set of states for action $a = (a_{\text{lane}}, a_{\text{dir}}, a_{\text{acc}})$ until the planning horizon t_p and then attaches the fail-safe maneuver indicated by α until the final time t_f. The two types of fail-safe maneuvers considered in this work are braking with maximum deceleration $-a_{max}$ until standstill (i.e., $\alpha = 1$), or accelerating with maximum acceleration a_{max} until the ego vehicle fully left the intersection and then braking until standstill with maximum deceleration (i.e., $\alpha = 0$). To adequately address the computational demands of RL, we only consider on these two fail-safe maneuvers. Our verification is based on the assumptions that the absolute acceleration of all vehicles is less or equal to the maximum acceleration a_{max}. If traffic participants cause accidents by not respecting traffic rules, these collisions are considered to be not the fault of the ego vehicle.

E. Safety Layer

The safety layer (see Fig. 4) identifies the safe discrete action space $\mathcal{A}^t_{\text{safe}}$ when (a) the time step $(t_p - \Delta t)/\Delta t$ since the last verification cycle is reached, (b) the accessible road network for the ego vehicle changed since the last time step, or (c) a lane change finished. Note that we start the verification at least one time step before the planning horizon t_p is reached to simulate that for real-world experiments the verification calculations must be finished before the planning horizon is reached. If a verification is necessary, the trajectories for all regular actions $\mathcal{A}_{\text{regular}}$ are generated, and we check if safety can be verified in the two relevant safety dimensions: *lane safety* verification for distances to the leading vehicle and lane-changing maneuvers results in the safe action set $\mathcal{A}^t_{\text{safe,l}}$ and *intersection safety* verification for crossing intersections results in the safe action set $\mathcal{A}^t_{\text{safe,i}}$. Thus, the set of safe actions at time step t is:

$$\mathcal{A}^t_{\text{safe}} = \mathcal{A}^t_{\text{safe,l}} \cap \mathcal{A}^t_{\text{safe,i}} \ . \tag{3}$$

Subsequently, the RL agent selects an action from $\mathcal{A}^t_{\text{safe}}$ and the previously calculated fail-safe action. When no action from $\mathcal{A}_{\text{regular}}$ can be verified as safe, the fail-safe trajectory attached to the previously chosen action is executed.

a) Identifying meaningful actions: To minimize the verification effort, first, we identify if the action is meaningful with the predicate $\texttt{meaningful}(a_t, a_{t-1}, x_t)$ where a_t is the action to verify, a_{t-1} is the action of the previous time step, and x_t is the ego vehicle's state. This predicate evaluates to true if and only if for $a_t, a_{t-1} \in \mathcal{A}_{\text{regular}}$:

- the lane to change to exists for a_{lane}, and
- the driving direction of a_{dir} is permitted, and
- no lane change is currently conducted.

However, if the action verification determines that it is unsafe to proceed with the lane change, it will be aborted and the fail-safe plan will be executed instead.

b) Verifying safe actions: Only for meaningful maneuvers, trajectories for the desired accelerations are generated. To verify the safety of a trajectory, we use a subset of the invariably safe set $\mathcal{S}^t$ [34, Proposition 1] – the invariably safe braking set $\mathcal{S}^t_1$ [34, Algorithm 1, line 10]:

$$\begin{aligned}\mathcal{S}^t_1 \leftarrow \{(s,d,v)^T \in \mathcal{X} \mid \forall s_j \in \mathcal{O}_j(t) \colon s \leq s_j - \Delta^t_{\text{safe}}(v, b_j) \\ \wedge\, v \leq v_{\max} \wedge s \in \mathcal{C}_{b_i,b_j}\},\end{aligned} \tag{4}$$

where $\mathcal{X}$ is the state space, $\mathcal{O}_j(t)$ is the predicted occupancy for an obstacle b_j, s_j is its longitudinal position, $\Delta^t_{\text{safe}}(v, b_j)$ is its safe distance to the ego vehicle, $v_{\max}$ is the speed limit, and $\mathcal{C}_{b_i,b_j}$ is the part of the road network (e.g., a lane) regarded for the invariably safe braking set calculation of obstacles b_i and b_j. In a nutshell, driving in the invariably safe braking set $\mathcal{S}^t_1$ guarantees safety for a vehicle in a lane based on its current position and velocity, obstacle dynamics, and safe distance constraints. We define $\mathcal{S}^t_{\mathcal{L}_k}$ as the invariably safe braking set $\mathcal{S}^t_1$ of a lane k at time step t (Eq. (4) with $\mathcal{C}_{b_i,b_j} = \mathcal{L}_k$).

The verification of *lane safety* is depicted in Algorithm 1. For a given action a and ego vehicle state $x_{ego,0}$, the function $\texttt{get\_current\_lane}(a, x_{ego,0})$ returns an identifier e, which indicates the current lane of the ego vehicle and its driving direction (cf. line 4). Then, the invariably safe braking set for all vehicles in this lane in front of the ego vehicle is calculated in line 5. If $a_{lane} = 1$, then the action is a lane-keeping action. For these lane-keeping actions, we only verify the safety of the planned trajectories with respect

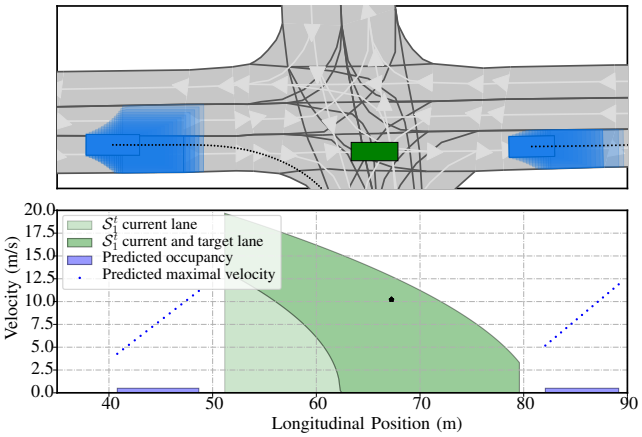


Fig. 5. Safety verification in a lane. Top: scenario with green ego vehicle and blue occupancy predictions for other vehicles; bottom: velocity and position for the invariably safe region in green, predicted maximal velocities and occupancies for traffic participants in blue, and ego state marked with a star.

to the leading vehicles (cf. line 7). For other actions, the function $\texttt{get\_target\_lane}(a, x_{ego,0})$ returns the identifier for the target lane of the lane change and we verify the safety with respect to all vehicles on the target lane and the leading vehicles on the current lane (cf. line 9-10). The study [39] describes a similar online verification for fail-safe planning of autonomous vehicles. In contrast to their work, we only considered two fail-safe maneuvers and limit the invariably safe set to the invariably safe braking set. This increases the efficiency of the implementation, which is necessary because of the learning setting of our work. Fig. 5 visualizes the invariably safe sets for an example lane safety situation.

For *intersection safety*, we verify the agent's actions at intersections such that the ego vehicle does not access an intersection in case another traffic participant could occupy it.

Algorithm 1 laneSafety()

Input: $\mathcal{S}^t_{\mathcal{L}_k}\ \forall k \in \mathcal{K}$, $x_{ego,0}$, a_{t-1}
Output: Safe lane actions $\mathcal{A}^t_{\text{safe,l}}$

1: $\mathcal{A}^t_{\text{safe,l}} := \emptyset$
2: **for all** $a \in \mathcal{A}_{\text{regular}} \wedge \texttt{meaningful}(a, a_{t-1}, x_{ego,0})$ **do**
3: **for all** $\alpha \in \{0, 1\}$ **do**
4: $e := \texttt{get\_current\_lane}(a, x_{ego,0})$
5: $\mathcal{S}^t_{\mathcal{L}_{e,lead}} := \{(s,d,v)^T \in \mathcal{S}^t_{\mathcal{L}_e} | s \geq \text{proj}_s(x_{ego,0})\}$
6: **if** $a_{lane} = 1$ **then**
7: $\mathcal{A}^t_{\text{safe,l}} \leftarrow \{(a,\alpha) | \text{plan}(a,\alpha) \subset \mathcal{S}^t_{\mathcal{L}_{e,lead}}\}$
8: **else**
9: $c := \texttt{get\_target\_lane}(a, x_{ego,0})$
10: $\mathcal{A}^t_{\text{safe,l}} \leftarrow \{(a,\alpha) | \text{plan}(a,\alpha) \subset \mathcal{S}^t_{\mathcal{L}_{e,lead}} \wedge \text{plan}(a,\alpha) \subset \mathcal{S}^t_{\mathcal{L}_c}\}$
11: **end if**
12: **end for**
13: **end for**
14: **return** $\mathcal{A}^t_{\text{safe,l}}$

Algorithm 2 intersectionSafety()

Input: $\mathcal{A}_{\text{regular}}$, $s_{i,start}$, $s_{i,end}$, $\mathcal{O}$, X, $\mathcal{L}$, $x_{ego,0}$, a_{t-1}
Output: Safe intersection actions $\mathcal{A}^t_{\text{safe,i}}$

1: $\mathcal{A}^t_{\text{safe,i}} := \emptyset$
2: **for all** $a \in \mathcal{A}_{\text{regular}} \wedge \texttt{meaningful}(a, a_{t-1}, x_{ego,0})$ **do**
3: **for all** $\alpha \in \{0, 1\}$ **do**
4: $\mathcal{CO} := \emptyset$
5: $e := \texttt{get\_current\_lane}(a, x_{ego,0})$
6: **for all** $o \in \mathcal{O}$ **do**
7: $\mathcal{C}_o := \texttt{get\_accessible\_lanes}(x_o) \cap \mathcal{L}_e$
8: $t_{cz} := \texttt{get\_t\_conflict}(x_o, \mathcal{C}_o)$
9: $\mathcal{CO} \leftarrow \{(s,t) | t \geq t_{cz} \wedge \min_s(\mathcal{C}_o) \leq s \leq \max_s(\mathcal{C}_o)\}$
10: **end for**
11: $\mathcal{A}^t_{\text{safe,i}} \leftarrow \{(a,\alpha) | \text{plan}(a,\alpha) \cap \mathcal{CO} = \emptyset\}$
12: **end for**
13: **end for**
14: **return** $\mathcal{A}^t_{\text{safe,i}}$

In contrast to other research on intersection safety [40], [41], our approach can deal with arbitrary real-world drivers and does not assume a cooperative setting. Algorithm 2 specifies the verification process. For all actions a and fail-safe actions α, we first identify the current lane e (cf. line 5). Then, we calculate the conflict zones $\mathcal{C}_o$ by intersecting the accessible lanes of each surrounding vehicle o with the lane $\mathcal{L}_e$, which corresponds to the regarded action (cf. line 7). For that, we use the obstacle set $\mathcal{O}$ containing identifiers for all obstacles within a circle around the ego vehicle's center with radius r_{int}. The current state of an obstacle x_o is obtained from the matrix $X = [x_1, ..., x_{\mathcal{O}}] \in \mathbb{R}^{N \times \mathcal{O}}$ where N is the number of state dimensions. The function $\texttt{get\_t\_conflict}(x_o, \mathcal{C}_o)$ returns the last time step t_{cz} before the surrounding obstacle o could reach its conflict zone with the ego vehicle $\mathcal{C}_o$ (cf. line 8). The reaching time is when the surrounding obstacle's occupancy (predicted using the SPOT [31] tool) intersects with the conflict zone $\mathcal{C}_o$. With the conflict zones $\mathcal{C}_o$ and the time step t_{cz}, we generate a collision object $\mathcal{CO}$ that describes the potential occupation of the conflict zones $\mathcal{C}_o$ for all surrounding obstacles o (cf. line 9). Finally, an action is safe if its corresponding trajectory, which includes the fail-safe trajectory, does not intersect with the collision object $\mathcal{CO}$ (cf. line 11).

IV. Experiments

We evaluated our implementation with recorded urban traffic data. First, we exhaustively specify the experimental setup. Then, we present the results, followed by an ablation study, and discuss our findings.

A. Experimental Setup

The inD dataset [42] contains recorded traffic data from four urban locations in Aachen, Germany. Particularly, two locations are at four-legged intersections (abbreviated by AAH_1 for Bendplatz and AAH_2 for Frankenburg) and

TABLE II

EXPERIMENTAL PARAMETERS

Parameter	Value	Parameter	Value
a_{max}	$8\,\mathrm{m\,s^{-2}}$	t_p	$0.4\,\mathrm{s}$
Δt	$0.04\,\mathrm{s}$	r_{int}	$50\,\mathrm{m}$
$s_{\mathrm{intersection}}$	$50\,\mathrm{m}$	$n_{\mathrm{intersection}}$	6

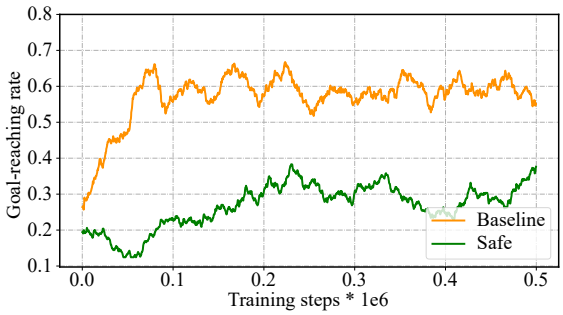
another at a T-junction (AAH_3 for Heckstraße). In this study, we excluded the data from the more complex T-junction at Neuköllner Strasse because without considering traffic signs and lights, the agent cannot reach the goal when the safety layer is activated and it mostly stops at an intersection. An open-source data converter[1] was used to convert the raw data was converted into CommonRoad scenarios [43]. Pedestrians and bicyclists were excluded for this study. Furthermore, we exactly detected the positions and velocities of the vehicles from the scenario data and no occlusions occured. To generate planning problems for the RL agent, one vehicle was removed from each scenario and its initial and finial state enlarged by the spatial dimensions of the vehicle were used as initial state and goal region for the planning problem. If the initial state of a generated scenario is not invariably safe, we did not use this scenario for learning. Additionally, we excluded scenarios where other vehicles appear in the scenario within the first planning cycle of the ego vehicle and close to the ego vehicle. Since these vehicles were absent for the first safety verification, they can lead to collisions due to the scenario data. Overall, we generated approximately 5000 traffic scenarios for the learning. Particularly, we used 1966 scenarios for AAH_1, 1904 for AAH_2, and 959 for AAH_3. The time step size for the scenarios is $0.04\,\mathrm{s}$ while the agent can decide on a new action every 10 time steps (i.e., $0.4\,\mathrm{s}$) in case a lane change did not finish before or the meaningful actions changed (see Fig. 4). All experimental parameters are specified in Table II.

We trained a safe and baseline agent without safety verification on each of the three inD locations and evaluated them on the test set. The safe agent uses the full safety layer for action verification. The baseline agent only uses the safety layer to eliminate meaningless actions (see Sec. III-E.a), thereby increasing the learning efficiency. For each experiment, we split the dataset into 70% training and 30% test sets. The implementation was based on the CommonRoad-RL[2] environment [35] and the stable baselines algorithm toolbox[3]. The PPO parameters and policy network architecture were identified by hyperparameter tuning. We trained every agent 500 000 training steps, which took approximately 24 hours for the safe agents with one thread on a machine with an AMD EPYC 7742 2.2 GHz processor and 1024 GB of DDR4 3200 MHz memory.

[1] commonroad.in.tum.de/dataset-converters

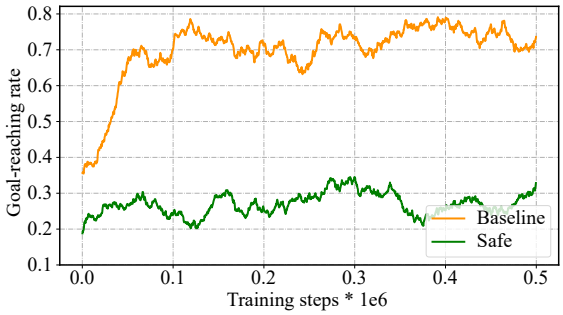
[2] We plan to release the exact implementation of this study with the next CommonRoad-RL release (commonroad.in.tum.de/commonroad-rl).

[3] https://github.com/hill-a/stable-baselines



(a) Goal-reaching rate of agents at location AAH_1.

(b) Goal-reaching rate of agents at location AAH_2.



(c) Goal-reaching rate of agents at location AAH_3.

Fig. 6. Goal-reaching rates for agents during training.

B. Results

The goal-reaching performance of the safe and baseline agent shows a approximately constant gap between 20 % and 40 % (see Fig. 6). The safe agent reached the goal for the AAH_1 location most often and the baseline agent for the AAH_3 location. The baseline agents still collided for 1.4 % to 7.2 % of the scenarios, whereas the safe agents did not cause any collision. Evaluation on the test set is similar to the training results indicating that the agents are not overfitted to the training set. The detailed training and testing results are shown in Table III.

C. Ablation Study

We conducted an ablation study to identify the benefits of the safety layer and its components. Therefore, we trained two additional safe agents: one restricts the actions only with the *intersection safety* (named safe int. agent) and

TABLE III

EVALUATION OF TRAINED AGENTS ON TRAINING AND TEST SETS FOR GOAL-REACHING RATE (COLLISION RATE).

Agent	AAH_1	AAH_2	AAH_3
Training Dataset			
Safe	30.4% (**0.0**%)	27.2% (**0.0**%)	29.1% (**0.0**%)
Safe lane	**73.0**% (3.1%)	**55.8**% (5.4%)	**73.9**% (1.5%)
Safe int.	42.1% (4.4%)	33.9% (4.8%)	52.4% (2.4%)
Baseline	65.1% (3.9%)	46.5% (6.8%)	72.6% (2.1%)
Test Dataset			
Safe	29.9% (**0.0**%)	25.3% (**0.0**%)	28.8% (**0.0**%)
Safe lane	**75.8**% (1.9%)	**54.6**% (4.6%)	71.2% (2.4%)
Safe int.	43.3% (4.5%)	30.4% (4.9%)	51.7% (2.4%)
Baseline	65.9% (4.1%)	44.4% (7.2%)	**75.0**% (1.4%)

Note: The collision rate is revised by collisions not caused by the ego vehicle, for example, another vehicle driving into the ego vehicle from behind, thus, violating the safe distance to the ego vehicle.

the other restricts the actions with the *lane safety* (named safe lane agent). The detailed evaluation results for the trained agents are shown in Table III. For the safe lane agent, the collision rate reduces to less than 5.5% for the training and test scenarios. Interestingly, at the same time the goal-reaching rate increased compared to that of the unsafe baseline agent. Thus, the agent learns better when guided by less and safer actions. For the agent whose actions are only restricted by intersection safety, the collision rate slightly increases compared to the baseline. Furthermore, the goal-reaching performance decreases on the training and test datasets compared to that of the baseline agent. However, only if the two concepts are combined in the safe agent, no collision caused by the ego vehicle occurred.

D. Discussion

The goal-reaching rate for the safe agent is comparably low. This is primarily due to the conservative setting of the parameters, which is necessary to guarantee safety with the current assumptions. However, integrating urban traffic rules in the verification of the safe actions could decrease conservative behavior in crowded intersections. This is supported by preliminary experiments on the data of the more complex T-junction at Neuköllner Strasse in Aachen. Furthermore, we plan to use our more holistic verification approach [44] in the future to alleviate conservativeness. This study has not realized this due to the RL's required low computation times.

Additionally, the current fail-safe planner is optimized for driving comfort and, thus, has limited capabilities to execute quick and uncomfortable reactions to maintain safety. Therefore, an advanced fail-safe planner [45] could be integrated. This is particularly important when human drivers break traffic rules since the autonomous agent needs to respond as quickly as possible to minimize the chances of an accident. However, the challenge is to decide for the correct time to use the fail-safe planner [46]. Additionally, formalized traffic rules could help to efficiently detect when and if a fail-safe planner should be activated.

To make our approach applicable to the real world, other traffic participants, such as pedestrians and cyclists, need to be included in the calculation of the invariably safe sets. Further, all urban traffic rules must be integrated into the verification process. Additionally, the current Python implementation would need to be computationally more efficient and possibly needs refactoring to C++. These issues are subject to future research.

V. CONCLUSIONS

We present a provably safe RL approach for urban driving that can simultaneously handle lane-changing and intersection crossing. Our general high-level action space can be applied to various intersection types. We show the capabilities of our approach on real-world traffic data from three intersections in Germany. These experiments demonstrate that our safety layer is inherently safe and provides safety guarantees for the ego vehicle. The ablation study indicates that compared to the unsafe baseline, adding the lane safety verification improves the performance while reducing collisions. To boost the provably safe RL agent's goal-reaching rate in the future, more traffic rules, a more complex fail-safe planner, better informed set-based prediction, and online verification of arbitrary maneuvers should be investigated.

ACKNOWLEDGMENT

The authors gratefully acknowledge the partial financial support of this work by the research training group ConVeY funded by the German Research Foundation under grant GRK 2428.

REFERENCES

[1] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 740–759, 2022.

[2] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2022.

[3] Z. Wang, Z. Yan, and K. Nakano, "Comfort-oriented haptic guidance steering via deep reinforcement learning for individualized lane keeping assist," in *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, 2019, pp. 4283–4289.

[4] P. Wang, C.-Y. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *Proc. of IEEE Intelligent Vehicles Symposium*, 2018, pp. 1379–1384.

[5] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *Proc. of IEEE International Conference on Robotics and Automation*, 2019, pp. 8248–8254.

[6] I.-M. Chen and C.-Y. Chan, "Deep reinforcement learning based path tracking controller for autonomous vehicle," *Proc. of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 235, no. 2-3, pp. 541–551, 2021.

[7] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-aware reinforcement learning for merging in dense traffic," in *Proc. of IEEE International Conference on Intelligent Transportation Systems*, 2019, pp. 3441–3447.

[8] S. Triest, A. Villaflor, and J. M. Dolan, "Learning highway ramp merging via reinforcement learning with temporally-extended actions," in *Proc. of IEEE Intelligent Vehicles Symposium*, 2020, pp. 1595–1600.

[9] D. Isele, A. Nakhaei, and K. Fujimura, "Safe reinforcement learning on autonomous vehicles," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1–6.

[10] M. Shikunov and A. I. Panov, "Hierarchical reinforcement learning approach for the road intersection task," in *Proc. of Biologically Inspired Cognitive Architectures Meeting*, 2019, pp. 495–506.

[11] Y. Guan, Y. Ren, H. Ma, S. E. Li, Q. Sun, Y. Dai, and B. Cheng, "Learn collision-free self-driving skills at urban intersections with model-based reinforcement learning," in *Proc. of IEEE International Intelligent Transportation Systems Conference*, 2021, pp. 3462–3469.

[12] J. Li, L. Sun, J. Chen, M. Tomizuka, and W. Zhan, "A safe hierarchical planning framework for complex driving scenarios based on reinforcement learning," in *Proc. of IEEE International Conference on Robotics and Automation*, 2021, pp. 2660–2666.

[13] H. Chae, C. M. Kang, B. Kim, J. Kim, C. C. Chung, and J. W. Choi, "Autonomous braking system via deep reinforcement learning," in *Proc. of IEEE International Conference on Intelligent Transportation Systems*, 2017, pp. 1–6.

[14] Y. Fu, C. Li, F. R. Yu, T. H. Luan, and Y. Zhang, "A decision-making strategy for vehicle autonomous braking in emergency via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5876–5888, 2020.

[15] L. Wen, J. Duan, S. E. Li, S. Xu, and H. Peng, "Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization," in *Proc. of IEEE International Conference on Intelligent Transportation Systems*, 2020, pp. 1–7.

[16] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. of AAAI Conference on Artificial Intelligence*, 2019, pp. 3387–3395.

[17] X. Wang, "Ensuring safety of learning-based motion planners using control barrier functions," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4773–4780, 2022.

[18] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *Proc. of IEEE International Conference on Intelligent Transportation Systems*, 2020, pp. 1–7.

[19] C. Hoel, K. Driggs-Campbell, K. Wolff, L. Laine, and M. J. Kochenderfer, "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 294–305, 2020.

[20] Z. Qiao, J. Schneider, and J. M. Dolan, "Behavior planning at urban intersections through hierarchical reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 2667–2673.

[21] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *Proc. of IEEE International Conference on Robotics and Automation*, 2018, pp. 2034–2039.

[22] D. Kamran, C. F. Lopez, M. Lauer, and C. Stiller, "Risk-aware high-level decisions for automated driving at occluded intersections with reinforcement learning," in *Proc. of IEEE Intelligent Vehicles Symposium*, 2020, pp. 1205–1212.

[23] T. Tram, I. Batkovic, M. Ali, and J. Sjöberg, "Learning when to drive in intersections by combining reinforcement learning and model predictive control," in *Proc. of IEEE International Conference on Intelligent Transportation Systems*, 2019, pp. 3263–3268.

[24] H. Krasowski, J. Thumm, M. Müller, X. Wang, and M. Althoff, "Provably safe reinforcement learning: A theoretical and experimental comparison," *arXiv preprint arXiv:2205.06750*, 2022.

[25] Z. Li, U. Kalabić, and T. Chu, "Safe reinforcement learning: Learning with supervision using a constraint-admissible set," in *Proc. of Annual American Control Conference*, 2018, pp. 6390–6395.

[26] W. Saunders, G. Sastry, A. Stuhlmüller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," in *Proc. of International Conference on Autonomous Agents and MultiAgent Systems*, 2018, p. 2067–2069.

[27] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Safe reinforcement learning with scene decomposition for navigating complex urban environments," in *Proc. of IEEE Intelligent Vehicles Symposium*, 2019, pp. 1469–1476.

[28] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods: Toward safe control through proof and learning," in *Proc. of AAAI Conference on Artificial Intelligence*, 2018, pp. 6485–6492.

[29] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *Proc. of IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 2156–2162.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[31] M. Koschi and M. Althoff, "SPOT: A tool for set-based prediction of traffic participants," in *Proc. of IEEE Intelligent Vehicles Symposium*, 2017, pp. 1686–1693.

[32] C.-Y. Tang, C.-H. Liu, W.-K. Chen, and S. D. You, "Implementing action mask in proximal policy optimization (PPO) algorithm," *ICT Express*, vol. 6, no. 3, pp. 200–203, 2020.

[33] N. Murgovski, G. R. de Campos, and J. Sjöberg, "Convex modeling of conflict resolution at traffic intersections," in *Proc. of IEEE Conference on Decision and Control*, 2015, pp. 4708–4713.

[34] C. Pek and M. Althoff, "Efficient computation of invariably safe states for motion planning of self-driving vehicles," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 3523–3530.

[35] X. Wang, H. Krasowski, and M. Althoff, "CommonRoad-RL: A configurable reinforcement learning environment for motion planning of autonomous vehicles," in *Proc. of IEEE International Conference on Intelligent Transportation Systems*, 2021, pp. 466–472.

[36] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenét frame," in *Proc. of IEEE International Conference on Robotics and Automation*, 2010, pp. 987–993.

[37] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of IEEE Intelligent Vehicles Symposium*, 2014, pp. 420–425.

[38] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 752–759.

[39] C. Pek and M. Althoff, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 798–814, 2021.

[40] H. Kowshik, D. Caveney, and P. R. Kumar, "Provable systemwide safety in intelligent intersections," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, p. 804–818, 2011.

[41] G. R. de Campos, F. D. Rossa, and A. Colombo, "Safety verification methods for human-driven vehicles at traffic intersections: Optimal driver-adaptive supervisory control," *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 1, pp. 72–84, 2018.

[42] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The inD dataset: A drone dataset of naturalistic road user trajectories at German intersections," in *Proc. of IEEE Intelligent Vehicles Symposium*, 2020, pp. 1929–1934.

[43] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.

[44] C. Pek, S. Manzinger, M. Koschi, and M. Althoff, "Using online verification to prevent autonomous vehicles from causing accidents," *Nature Machine Intelligence*, vol. 2, no. 9, pp. 518–528, 2020.

[45] S. Magdici and M. Althoff, "Fail-safe motion planning of autonomous vehicles," in *Proc. of IEEE International Conference on Intelligent Transportation Systems*, 2016, pp. 452–458.

[46] M. Althoff, S. Maierhofer, and C. Pek, "Provably-correct and comfortable adaptive cruise control," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 159–174, 2021.

Safe Reinforcement Learning for Urban Driving using Invariably Safe Braking Sets

Conference Proceedings:
2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)

Author: Hanna Krasowski

Publisher: IEEE

Date: 08 October 2022

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

B Reinforcement Learning with Safety Specifications via Temporal Logic

Legal safety specifications for autonomous vehicles performing motion planning tasks are often more complex than only avoiding collisions. If we solely regard collision avoidance, this can lead to a conservative safety specification, as empirically observed in Appendix A.4. Thus, including traffic rules in the safety verification of actions could help to reduce the conservatism for motion planning tasks. Temporal logic is suited to specify complex spatio-temporal properties common for traffic rules. However, integrating a safety specification formalized via temporal logic in verification approaches for autonomous vehicles is challenging because these systems operate in continuous space and a tight model often is only available for a part of the system.

The publications reproduced in this chapter formalize safety specifications via temporal logic and integrate these specifications in model-free RL algorithms to achieve probabilistic or hard safety guarantees. The safety specifications are more expressive than the avoid safety specification regarded in Appendix A and can be described by Safety Specification 1. Since correctly formalizing safety specifications from natural language to temporal logic is not trivial for real-world motion planning tasks, we first formalize maritime traffic rules in Appendix B.1. In Appendix B.2, we present a safe RL approach that achieves probabilistic guarantees without an explicit system model for an arbitrary temporal logic robustness measure. Finally, we present a provably safe RL approach for autonomous vessel navigation on the open sea in Appendix B.3 that always achieves rule compliance with our previously formalized maritime traffic rules.

B.1 Temporal Logic Formalization of Marine Traffic Rules

Summary Most vehicles are operated by humans. To achieve safe operation on roads, on water, or in the air, there are usually interaction rules, or more specifically, traffic rules, defined. These rules describe how to act predictably, reducing uncertainty for other humans operating other vehicles, and thereby avoid collisions. However, these rules are usually specified in natural language, which may be ambiguous. To incorporate these rules in autonomous vehicles, there needs to be a precise and unique specification.

In this work, we provide such a specification of maritime traffic rules for power-driven vessels on the open sea. We utilize MTL since it possesses the expressiveness necessary for formalization. First, we rigorously define the predicates that determine the spatial relations between two vessels and derive temporal logic specifications based on the predicates. Based on the predicates, we obtain six formalized rules, which represent the most important collision avoidance rules between power-driven vessels specified by the COLREGS. We use parameters within our formalization so that different instantiations of predicates and rules can capture different traffic situations or jurisdictions.

To obtain realistic traffic situations, in which our formalized rules apply, we filter maritime traffic data for US coastal areas for close encounters of vessels. We evaluate our specified rules for the vessel encounters and observe that the vast majority of vessels adhere to the formalized maritime traffic rules for our chosen open-sea parametrization of the rules. The parametrization allows for adjusting the rules to other maritime traffic situations, such as navigating in channels, without the need of re-formalization.

Author contributions H.K. and M.A. initiated the project of formalizing maritime traffic rules with temporal logic. H.K. identified the most relevant rules, translated the natural language into metric temporal logic specifications, implemented the traffic rule monitor, conducted the numerical experiments, and wrote the manuscript. M.A. provided feedback improving the temporal logic formalization and the manuscript.

TUM Graduate School This publication has been declared a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

Temporal Logic Formalization of Marine Traffic Rules

Hanna Krasowski and Matthias Althoff

***Abstract*— Autonomous vessels have to adhere to marine traffic rules to ensure traffic safety and reduce the liability of manufacturers. However, autonomous systems can only evaluate rule compliance if rules are formulated in a precise and mathematical way. This paper formalizes marine traffic rules from the Convention on the International Regulations for Preventing Collisions at Sea (COLREGS) using temporal logic. In particular, the collision prevention rules between two power-driven vessels are delineated. The formulation is based on modular predicates and adjustable parameters. We evaluate the formalized rules in three US coastal areas for over 1,200 vessels using real marine traffic data.**

I. Introduction

Human error is the main contributing factor to half of the 1,801 marine accidents between 2014 and 2019 analyzed by the European Maritime Safety Agency [1]. Autonomous vessels are a potential solution to decreasing the number of accidents caused by humans. These autonomous vessels will have to consider marine traffic rules and act accordingly. Thus, formalizing coherent marine traffic rules for machines is necessary.

The Convention on the International Regulations for Preventing Collisions at Sea (COLREGS[1]) [2] describes the marine traffic rules for preventing collisions. The COLREGS became effective in 1972 and consists of 38 rules grouped in five parts. In international waters, the COLREGS are the sole collision avoidance rules[2]. For the collision avoidance of autonomous vessels, only the second part, which considers steering and sailing regulation (i.e., COLREGS rules 4 - 19), is relevant. The rules of this part define different encounters and how the encountering vessels should react to prevent a collision. However, the rules specified in the COLREGS are formulated for humans and are not directly applicable to and verifiable for an autonomous vessel.

In this paper, we formalize the marine traffic rules of the COLREGS which consider collision avoidance between power-driven vessels. Our main contributions are:

- To the best of our knowledge, we present the first formalization of COLREGS using temporal logic.
- Our predicates and functions are parameterizable and usable for additional marine traffic rules.
- We evaluate our formalized rules on real marine traffic data for over 1,200 vessels.

The remainder of this paper is structured as follows: Section II presents an overview of the related literature. Section III describes methodical concepts for rule formalization. Sections IV and V introduce the formalized rules and associated predicates in detail. Section VI presents the evaluation of the formalized rules on scenarios generated from automatic identification system (AIS). Section VII provides conclusions.

II. Related work

The COLREGS has been mainly considered implicitly for motion planning problems so that motion planners comply with the COLREGS. In general, the COLREGS rules for steering and sailing are integrated in planners through geometric thresholds [3]–[8], virtual obstacles [9], or cost functions [10], [11]. Further, most research does not comply with all COLREGS rules for steering and sailing but instead focuses on the rules for the give-way vessel in crossing, head-on, and overtaking situations (see Section IV) [3]–[6], [11]. Some research additionally considers the safe distance [7], [8], stand-on vessel [7], [8], [10], and last-minute maneuver rule [7], [8]. However, all of these approaches are directly integrated in problem representation or motion planner, and are difficult to extend to include additional marine traffic rules.

As soon as a collision is possible, the COLREGS section describing steering and sailing rules for collision prevention has to be enforced. Therefore, detecting future collision possibilities is imperative. The simplest approach to detect potential collisions is by checking if a specified distance is kept in relation to other traffic participants [6]. This is easy to implement, however, this disregards the direction in which the vessel moves and results in many false-positive collision warnings. Another option is by calculating the closest point of approach [3], [4], which returns the time when the shortest distance between two vessels is reached assuming constant velocity and heading. However, defining thresholds [3] or constructing a virtual obstacle from the closest point of approach [4] to determine the risk of collision, which is not trivial, remains necessary. Another common approach is using a velocity obstacle [7], [12]. Here, heading and velocity are also assumed to be constant, but a virtual collision obstacle is constructed instead of calculating scalar values. From this object, the ego ship can anticipate which headings and velocities would lead to a collision without the necessity of specific thresholds. Further, for crowded scenarios, various velocity obstacles can be superimposed, and static obstacles

All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.
hanna.krasowski@tum.de, althoff@in.tum.de

[1]We use this acronym in this work for the Convention on the International Regulations for Preventing Collisions at Sea. However, in the literature, there is no consistent orthography, and it is unclear how the acronym is built.

[2]In national territory, additional rules have to be satisfied, e.g., in German coastal waters the German Traffic Regulations for Navigable Maritime Waterways (SeeSchStrO).

can be included [12]. The most general approach to detect potential collisions is through reachability analysis, where detailed kinematic models and measurement uncertainties can be considered to obtain an over-approximate occupancy [13]. However, in this study, we focus on formalizing marine traffic rules for open-sea situations; thus, collision detection with velocity obstacles is deemed sufficient.

Our approach uses temporal logic to model the COLREGS and explicitly checks rule compliance instead of only determining a contemporaneous traffic situation and rule-compliant actions. Temporal logic modeling allows for efficient extensions of rules, and implemented predicates can be reused. The COLREGS rules regarded in this work are the subset that regulates collision avoidance between power-driven vessels. The velocity obstacle concept is used to determine if a collision is possible.

III. Methodology

A. Metric temporal logic

Most marine traffic rules have preconditions; e.g., the possibility of a collision depends on the current relative position and speed of two vessels. These preconditions can be formalized by temporal logic. In particular, we use metric temporal logic (MTL) [14], which can define conditions that have to hold within a specified time interval.

In this work, MTL is interpreted over finite traces of predicates. The used fragment of MTL can specify propositions for the future. The temporal operators used in this work are G, F, X, and U. The future globally operator $\mathrm{G}(\phi)$ indicates that the proposition ϕ has to hold true for all future time steps, whereas for the future operator $\mathrm{F}(\phi)$, ϕ has to be true only for at least one future time step. The next operator $\mathrm{X}(\phi)$ specifies that ϕ holds true for the next time step. The until operator $\phi_1 \mathrm{U} \phi_2$ specifies that ϕ_1 holds true for all time steps until ϕ_2 holds true. The formal semantics of the temporal operators are described in detail in [15]. Given atomic propositions ϕ_i, an MTL formula Φ can be constructed as follows:

$$\Phi := \phi_i | \neg\Phi | \Phi_1 \wedge \Phi_2 | \Phi_1 \vee \Phi_2$$
$$\Phi := \mathrm{G}_I(\Phi) \,|\, \mathrm{F}_I(\Phi) \,|\, \mathrm{X}(\Phi) \,|\, \Phi_1 \mathrm{U} \Phi_2$$

The subscript I indicates the time interval for which the temporal operator is applied relative to the current time step. If no interval is specified, the operator is applied for the whole trace. The traces regarded in this work are finite. Boolean operators $\neg, \vee, \wedge$ are also used. The Boolean implication operator $\implies$ is modeled by $\Phi_1 \implies \Phi_2 \equiv \neg\Phi_1 \vee \Phi_2$.

B. Velocity obstacle

The velocity obstacle concept was first introduced in [16] to extract relevant objects for an autonomous mobile robot to avoid collisions. A velocity obstacle is a geometric object from which the currently feasible velocities of robots can be determined by testing if a velocity intersects with a velocity obstacle. The basic velocity obstacle approach assumes that the current position, spatial extensions, and speed of the obstacles are known and that they move with constant speed and heading; there are extensions that eliminate these assumptions [17]. However, the basic velocity obstacle approach is sufficient for this study as these assumptions are valid for checking potential collisions for vessels on the open sea, where course and speed are usually kept constant.

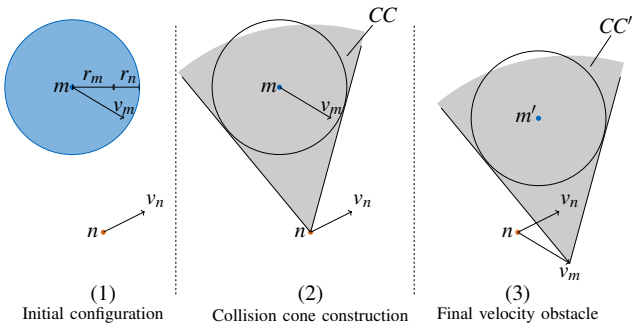


Fig. 1. Construction of a velocity obstacle. The robot n and obstacle m potentially collide as the velocity of the robot v_n intersects with the collision cone CC'

The construction of the basic velocity obstacle from [16] follows the three-step process visualized in Fig. 1. First, the shapes of the robot n and obstacle m are over-approximated by circles. Since the shape of the robot and obstacle does not change, it suffices to initialize the robot as a point n and add the radius of the robot r_n to the obstacle m with radius r_m. The collision cone CC is drawn by constructing tangent lines on the enlarged obstacle, which intersect at point n. Finally, the cone is translated by the obstacle velocity vector v_m. A collision is possible when the velocity vector of the robot v_n intersects with the translated collision cone CC'.

IV. Formalizing COLREGS in temporal logic

As mentioned in the introduction, the steering and sailing rules are relevant for specifying collision prevention maneuvers and are crucial for safe motion planning of autonomous vessels. We assume the following conditions for our formalization of the steering and sailing rules:

- The water depth is sufficient for all vessels and does not restrict the possible maneuvers.
- Fairways and marine traffic marks are absent.
- There is a good visibility.
- All vessels are power-driven.

However, our formalization can easily be extended to other vessel types by integrating a method (e.g., an automaton) that selects the rules applicable for the current vessel type combination as specified in rule 18. The assumptions limit the formalization to rules 4, 6, 8(d), 11, and 13 - 17.

We formulate the rules from the ego-ship perspective. The implementation is built on our previous work on road traffic rules [18], which developed a rule monitor for evaluating interstate traffic rules. The rule monitor first evaluates the predicates, which are described in detail in Section V, to create a finite predicate trace. This trace is then used to evaluate the specified MTL formulas.

TABLE I

OVERVIEW OF THE FORMALIZED MARINE TRAFFIC RULES

Rule	COLREGS reference	MTL formula
R_1	Rule 4, 8(d)	$\mathrm{G}\Big(\neg\text{collision\_possible}(x_{ego}, x_o, t^{coll}_{horizon})\Big)$
R_2	Rule 4, 6	$\mathrm{G}\Big(\text{safe\_speed}(x_{ego}, v_{max})\Big)$
R_3	Rule 11, 15, 16	$\mathrm{G}\Big(\big(\neg\text{crossing}(x_{ego}, x_o, *) \wedge \mathrm{G}_{[\Delta t, t_{react}]}\,(\text{crossing}(x_{ego}, x_o, *))\big) \implies \big(\mathrm{F}_{[0, t_{react}+t_{maneuver}]}(\text{maneuver\_crossing}(x_{ego}, x_o, *)) \wedge \mathrm{F}_{[t_{react}, t_{react}+2t_{maneuver}]}(\neg\text{crossing}(x_{ego}, x_o, *))\big)\Big)$
R_4	Rule 11, 14, 16	$\mathrm{G}\Big(\big(\neg\text{head\_on}(x_{ego}, x_o, *) \wedge \mathrm{G}_{[\Delta t, t_{react}]}\,(\text{head\_on}(x_{ego}, x_o, *))\big) \implies \big(\mathrm{F}_{[0, t_{react}+t_{maneuver}]}(\text{maneuver\_head\_on}(x_{ego}, x_o, *)) \wedge \mathrm{F}_{[t_{react}, t_{react}+2t_{maneuver}]}(\neg\text{head\_on}(x_{ego}, x_o, *))\big)\Big)$
R_5	Rule 11, 13, 16	$\mathrm{G}\Big(\big(\neg\text{overtake}(x_{ego}, x_o, *) \wedge \mathrm{G}_{[\Delta t, t_{react}]}\,(\text{overtake}(x_{ego}, x_o, *))\big) \implies \big(\mathrm{F}_{[0, t_{react}+t_{maneuver}]}(\text{maneuver\_overtake}(x_{ego}, x_o, *)) \wedge \mathrm{F}_{[t_{react}, t_{react}+2t_{maneuver}]}(\neg\text{overtake}(x_{ego}, x_o, *))\big)\Big)$
R_6	Rule 11, 15, 17	$\mathrm{G}\Big(\text{keep}(x_{ego}, x_o, *) \implies (\text{no\_turning}(x_{ego}, *)\mathrm{U}\,\neg\text{keep}(x_{ego}, x_o, *))\Big)$

Note: Additional arguments are abbreviated by $*$.

Table I shows an overview of the formalized rules and indicates the corresponding rules of the COLREGS. The state of vessel i is denoted as x_i. We use the subscript *ego* for the ego vessel and o for the other traffic participants. Additional arguments of the predicates are abbreviated by $*$ to ease readability and are fully specified in Section V. The textual description of the formalized rules is as follows:

a) Safe distance R_1: Vessels always have to keep a safe distance from one another. This distance depends on the current speed and traffic scene. Therefore, we determine if the current distance between two vessels is safe by checking with the velocity obstacle concept if no collision is possible within the time horizon $t^{coll}_{horizon}$.

b) Safe speed R_2: A vessel shall always maintain a safe speed depending on the state of visibility, traffic density, and technical equipment on board.

c) Crossing R_3: When (a) two vessels are sailing on crossing paths in sight of each other, (b) there is a risk of collision, and (c) the other vessel is on starboard (i.e., right side), the ego vessel is the give-way vessel in the crossing situation. Therefore, when the ego vessel detects this situation and it is maintained until the reaction time t_{react}, it has to significantly change its course to starboard within the sum of the reaction time t_{react} and maneuver time $t_{maneuver}$. Further, the situation has to be resolved after another maneuver time $t_{maneuver}$. The detection of the changed situation has to happen within the time step Δt.

d) Head-on R_4: When two vessels approach each other in sight on opposing or near-opposing courses and there is a risk of collision, both vessels have to give way. Thus, similar to rule R_3, both vessels have to significantly change their course to starboard to resolve the situation.

e) Overtake R_5: When the ego vessel is faster and approaching another vessel in sight from its stern (i.e., from behind), it is in the give-way position of the overtaking situation. Therefore, the ego vessel has to significantly change its course to any side to avoid collision with the other vessel while overtaking.

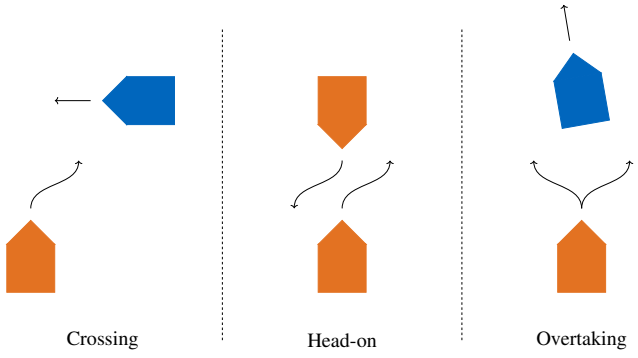


Fig. 2. Regulated situations of the COLREGS. Orange colors denote give-way vessels, and blue colors denote stand-on vessels. A give-way vessel has to evade the other vessel. A stand-on vessel has to keep its course during the maneuver of the other vessel.

f) Stand-on vessel R_6: When (a) two vessels are sailing in sight of each other, (b) the ego vessel has the other vessel on its port side (i.e., left side) or the ego vessel is overtaken, and (c) the risk of collision exists, the ego vessel is the stand-on vessel. Thus, the ego vessel has to keep its course until the situation is resolved.

Fig. 2 illustrates the situations and appropriate reactions for rules R_3 - R_6.

V. PREDICATES

Predicates are used to specify different conditions for marine traffic rules. We first specify necessary mathematical functions to model the predicates. Then, we group the predicates regarding position, velocity, and general conditions.

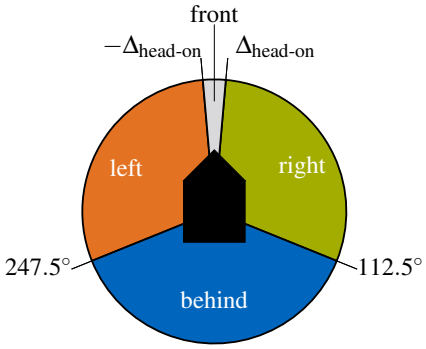


Fig. 3. Position regions relative to ego vessel.

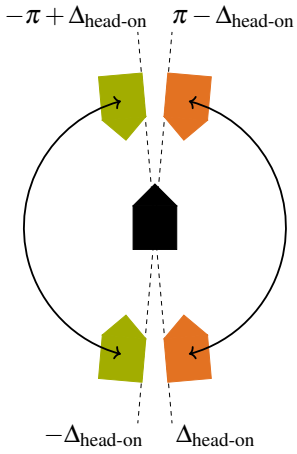


Fig. 4. Relative orientations to ego vessel. The black vessel is the ego vessel, the orange vessels indicate bounds for relative orientation toward left and the green vessels toward right.

A. Vessel movements and general functions

Vessel movements are specified through trajectories. Each trajectory consists of states at discrete time steps. The trajectory of vessel i is denoted as $\texttt{traj}_i$.The state x_i of vessel i consists of the position $p \in \mathbb{R}^2$, heading $h \in [0, 2\pi]$, velocity $v \in \mathbb{R}$, and yaw rate $\dot{\theta} \in \mathbb{R}$. The operator $\text{proj}_{\square}$ projects the state to the dimension specified by $\square$. We define a clock $\text{cl}(\texttt{traj}_i, x_i)$ which starts at the first time step of a trajectory and returns the passed time for a state x_i. In addition, we define a function $\text{state}(\texttt{traj}_i, t_k)$ which returns the state of a trajectory at time t_k. The Euclidean norm of a vector is denoted by $\|\cdot\|$ and the modulo operator $\text{mod}(a,b)$ returns the remainder of a/b for $a, b \in \mathbb{R}$ using floored division.

B. Position predicates

The important relative position regions for the formalized rules are visualized in Fig. 3. We use halfspaces to determine in which sector the other vessel is located. A position p_i is within a halfspace if:

$$d_{hs}^T p_i - b_{hs} \leq 0$$

where b_{hs} is the offset to the origin, and d_{hs} is the normal vector of the halfspace.

The predicate $\text{in\_front\_sector}(x_n, x_m)$ is true if and only if the center of vessel m is in the halfspace left of the $\Delta_{\text{head-on}}$ line and in the halfspace right of the $-\Delta_{\text{head-on}}$ line of Fig. 3 for vessel n. The parameter $\Delta_{\text{head-on}}$ specifies half of the front sector angle[3]. The predicates for the other three sectors can be analogously anticipated from Fig. 3. We consider the center of the other vessel instead of its entire occupancy because the spatial dimensions of vessels are negligibly small compared to the distance between the vessels.

In addition to the occupied sector, the relative orientation is also relevant. The predicate orientation_delta evaluates if the heading difference of two vessels is larger than a specified difference Δ_{orient}. Additionally, a constant offset c_{o} is integrated to evaluate the heading difference between two vessels in a head-on situation:

$$\text{orientation\_delta}(x_n, x_m, \Delta_{\text{orient}}, c_{\text{o}}) \iff$$
$$\text{mod}(\text{proj}_h(x_m) - \text{proj}_h(x_n) + c_{\text{o}}, 2\pi) \in [\Delta_{\text{orient}}, 2\pi - \Delta_{\text{orient}}].$$

Checking if the other vessel is heading toward right or left with respect to the ego vessel is necessary for evaluating the crossing situation. Thus, we define the predicate $\text{orientation\_towards\_right}(x_n, x_m, \Delta_{\text{head-on}})$ which is true when the relative orientation between the other vessel m and the ego vessel n is in $[-\pi + \Delta_{\text{head-on}}, -\Delta_{\text{head-on}}]$. Analogously, the predicate $\text{orientation\_towards\_left}(x_n, x_m, \Delta_{\text{head-on}})$ is true for $[\Delta_{\text{head-on}}, \pi - \Delta_{\text{head-on}}]$. Fig. 4 illustrates the ranges in which the two predicates are evaluated to true.

C. Velocity predicates

The predicate drives_faster evaluates if vessel n drives faster than vessel m:

$$\text{drives\_faster}(x_n, x_m) \iff \text{proj}_v(x_n) > \text{proj}_v(x_m).$$

For rule R_2, we need to determine if the ego vessel drives at a safe speed. As we assume the vessels to be on the open sea with a low traffic density, the maximal safe velocity v_{max} is the typical engine limit for power-driven vessels. Further, the minimal safe velocity is zero as backward driving is an unexpected and thus unsafe behavior on the open sea. Thus, the predicate that evaluates if a vessel is sailing at safe speed is as follows:

$$\text{safe\_speed}(x_n, v_{max}) \iff 0 \leq \text{proj}_v(x_n) \leq v_{max}.$$

D. General predicates

To determine if a collision between two vessels is possible, we use the velocity obstacle concept. Thus, we represent the velocity of a vessel i as vector v_i. The collision cone $CC'(x_n, x_m)$ for two vessels n and m is constructed as described in Section III-B and visualized in Fig. 1. Possible collisions are detected if the velocity v_n intersects with the collision cone $CC'(x_n, x_m)$ and if the lower bound of the time

[3]Note that the COLREGS do not specify the angle for the front sector compared to the behind sector. In the literature, $\Delta_{\text{head-on}}$ is usually set to 5 deg or 10 deg.

to collision is less than the time horizon t, which is greater or equal than the time step size Δt:

$$\begin{aligned}&\text{collision\_possible}(x_n, x_m, t) \iff \\ &\quad v_n \in CC'(x_n, x_m) \wedge \\ &\quad \|v_n - v_m\| \leq \|\text{proj}_p(x_n) - \text{proj}_p(x_m)\|/t.\end{aligned}$$

We use this predicate for two purposes. First, if $t = t_{horizon}^{coll}$, we check if the distance between vessels is sufficient, so that the ego vessel can still avoid a collision by changing the course or stopping even if the other vessel does not react properly. Second, if $t = t_{horizon}^{check}$, the vessels sail on courses that lead to potential collisions, and a COLREGS collision avoidance maneuver has to be applied.

For rules R_3 - R_6, we need to specify a collision avoidance maneuver. Therefore, we define a predicate that evaluates if the course has changed since a defined time:

$$\begin{aligned}&\text{change\_course}(x_n, \texttt{traj}_n, t_{start}, \Delta_{\text{course}}) \iff \\ &\quad \Big| \sum_{t_i = t_{start}}^{\text{cl}(\texttt{traj}_n, x_n)} \text{proj}_{\dot{\theta}}(\text{state}(\texttt{traj}_n, t_i))\, \Delta t \Big| \geq \Delta_{\text{course}},\end{aligned}$$

where t_{start} is the staring time of the maneuver, and Δ_{course} is the change of heading that should be achieved. Further, for head-on and crossing situations, the give-way vessel has to evade to starboard. Therefore, we specify a predicate that indicates the turning direction:

$$\begin{aligned}&\text{turning\_to\_starboard}(x_n, \texttt{traj}_n, t_{start}) \iff \\ &\quad \text{mod}\big(\text{proj}_h(\text{state}(\texttt{traj}_n, \text{cl}(\texttt{traj}_n, x_n))) - \\ &\quad \text{proj}_h(\text{state}(\texttt{traj}_n, t_{start})), 2\pi\big) \in (\pi, 2\pi).\end{aligned}$$

For both previous predicates, we need the starting time of a maneuver. Therefore, let us define the operator $t_s(\Psi)$ that returns the time of the last rising edge of a predicate Ψ relative to the initial time of the trajectory, which can be formulated as $\neg\Psi \wedge \text{X}(\Psi)$. The arguments of the predicate are omitted for better readability. If Ψ remained constant until the current time step, $t_s(\Psi)$ returns zero.

An overtaking situation as specified in the COLREGS is defined by (a) a potential collision, (b) the overtaken vessel m has the regarded vessel n in its behind sector, (c) the regarded vessel n has to be faster than the other vessel, and (d) the heading difference has to be less than 67.5 deg:

$$\begin{aligned}&\text{overtake}(x_n, x_m, t_{horizon}^{check}) \iff \\ &\quad \text{collision\_possible}(x_n, x_m, t_{horizon}^{check}) \wedge \\ &\quad \text{in\_behind\_sector}(x_m, x_n) \wedge \text{drives\_faster}(x_n, x_m) \wedge \\ &\quad \neg\text{orientation\_delta}(x_n, x_m, 67.5\,\text{deg}, 0).\end{aligned}$$

The angle of 67.5 deg is half of the behind sector angle and specified in rule 13 of the COLREGS. The appropriate maneuver for an overtaking situation is significantly turning, so that the overtaken vessel can detect the maneuver.

$$\begin{aligned}&\text{maneuver\_overtake}(x_n, x_m, \texttt{traj}_n, t_{horizon}^{check}, \Delta_{\text{large\_turn}}) \iff \\ &\quad \text{change\_course}(x_n, \texttt{traj}_n, t_s(\text{overtake}), \Delta_{\text{large\_turn}}) \wedge \\ &\quad \text{overtake}(x_n, x_m, t_{horizon}^{check}),\end{aligned}$$

where $\Delta_{\text{large\_turn}}$ is a turning angle that is sufficiently large to be detected by other vessels. For head-on situations, the specification is similar, however, there is no velocity condition, and the vessels have to be on opposing courses with deviation of at most $\Delta_{\text{head-on}}$:

$$\begin{aligned}&\text{head\_on}(x_n, x_m, t_{horizon}^{check}, \Delta_{\text{head-on}}) \iff \\ &\quad \text{collision\_possible}(x_n, x_m, t_{horizon}^{check}) \wedge \text{in\_front\_sector}(x_n, x_m) \wedge \\ &\quad \neg\text{orientation\_delta}(x_n, x_m, \Delta_{\text{head-on}}, \pi).\end{aligned}$$

The appropriate maneuver in a head-on situation is similar to the overtaking maneuver, but the vessel has to turn to the starboard side. Therefore, we define that the maneuver is conducted as follows:

$$\begin{aligned}&\text{maneuver\_head\_on}(x_n, x_m, \texttt{traj}_n, t_{horizon}^{check}, \Delta_{\text{large\_turn}}, \Delta_{\text{head-on}}) \\ &\quad \iff \text{change\_course}(x_n, \texttt{traj}_n, t_s(\text{head\_on}), \Delta_{\text{large\_turn}}) \wedge \\ &\qquad \text{turning\_to\_starboard}(x_n, \texttt{traj}_n, t_s(\text{head\_on})) \wedge \\ &\qquad \text{head\_on}(x_n, x_m, t_{horizon}^{check}, \Delta_{\text{head-on}}).\end{aligned}$$

A crossing situation is defined by (a) a collision possibility; (b) the regarded vessel n in the give-way position, i.e., the other vessel m is in its right sector; and (c) the heading of the other vessel points toward the left side of the regarded vessel. Thus, the predicate for the crossing situation is specified as follows:

$$\begin{aligned}&\text{crossing}(x_n, x_m, t_{horizon}^{check}, \Delta_{\text{head-on}}) \iff \\ &\quad \text{collision\_possible}(x_n, x_m, t_{horizon}^{check}) \wedge \text{in\_right\_sector}(x_n, x_m) \wedge \\ &\quad \text{orientation\_towards\_left}(x_n, x_m, \Delta_{\text{head-on}}).\end{aligned}$$

The appropriate maneuver in a crossing situation is identical to the head-on maneuver:

$$\begin{aligned}&\text{maneuver\_crossing}(x_n, x_m, \texttt{traj}_n, t_{horizon}^{check}, \Delta_{\text{large\_turn}}, \Delta_{\text{head-on}}) \\ &\quad \iff \text{change\_course}(x_n, \texttt{traj}_n, t_s(\text{crossing}), \Delta_{\text{large\_turn}}) \wedge \\ &\qquad \text{turning\_to\_starboard}(x_n, \texttt{traj}_n, t_s(\text{crossing})) \wedge \\ &\qquad \text{crossing}(x_n, x_m, t_{horizon}^{check}, \Delta_{\text{head-on}}).\end{aligned}$$

For the stand-on vessel, the predicate keep is introduced, which indicates if the vessel is in the stand-on position and, thus, has to keep its course. It either evaluates to true when the other vessel is in the left sector and driving toward the other vessel m or when the regarded vessel n is overtaken.

$$\begin{aligned}&\text{keep}(x_n, x_m, t_{horizon}^{check}, \Delta_{\text{head-on}}) \iff \\ &\quad \big(\text{collision\_possible}(x_n, x_m, t_{horizon}^{check}) \wedge \text{in\_left\_sector}(x_n, x_m) \wedge \\ &\quad \text{orientation\_towards\_right}(x_n, x_m, \Delta_{\text{head-on}})\big) \vee \\ &\quad \text{overtake}(x_m, x_n, t_{horizon}^{check})\end{aligned}$$

The maneuver of the stand-on vessel is keeping its course, which is described as follows:

$$\begin{aligned}&\text{no\_turning}(x_n, \texttt{traj}_n, \Delta_{\text{no\_turn}}) \iff \\ &\quad \neg\text{change\_course}(x_n, \texttt{traj}_n, t_s(\text{keep}), \Delta_{\text{no\_turn}}),\end{aligned}$$

where $\Delta_{\text{no\_turn}}$ is the maximal heading deviation from the original course.

VI. Numerical experiments

A. Dataset and preprocessing

Our dataset consists of recorded samples from the AIS, a radio system designed to improve the safety of marine traffic by providing information about surrounding vessels. AIS data consists of static vessel information (e.g., vessel name), dynamic information (e.g., current position), and voyage information (e.g., estimated time of arrival). All commercial vessels with gross tonnage over 300 and all passenger vessels are obligated to have AIS on board. As the AIS data providers track about 200,000 vessels per day, which is about four times the number of the worldwide merchant fleet, it is assumed that most larger vessels operating on the open sea are equipped with AIS.

Tu et al. [19] compare different AIS data sources with respect to their accessibility, time resolution, position precision, and live broadcasting possibility. For the purposes of this study, a high precision and time resolution is most important. Thus, we selected the Marine Cadastre dataset [20] for generating encounter scenarios. This dataset provides historic AIS data of US coastal waters from 2009 to 2020 of which we used data from January 2019. We preprocessed the data in two steps. First, we searched for encounters between vessels. Second, we generated the trajectories for the encountering vessels.

To ensure the validity of our assumptions in the chosen scenarios, we selected specific open-sea regions listed in Table II. For each of the locations, we search for vessels whose tracks have a distance lower than 0.03 degree of latitude or longitude, which is approximately 2000 m, within a 10 min time frame. As the rules have the implicit precondition that the vessels move, we excluded vessels that did not move.

We use the time stamp, longitudinal and lateral positions, speed over ground, and length and width of the vessel from the AIS data. For evaluation, each trajectory needs a fixed time step size, here 10 s, and the time steps between vessel have to be synchronous for evaluation. Therefore, we use linear interpolation to synchronize the time steps of different vessels. The longitudinal and lateral positions are converted in a metric coordinate system according to the Universal Transverse Mercator system. The generated scenarios are using the CommonRoad representation[4]. The scenarios include two, three, or four vessels and the duration is between 10 min and 106 min. The data and implementation are available online[5].

TABLE II

Selected locations of US coastal areas

Location	Degree latitude	Degree longitude	#Vessels
Florida	[27.51, 32.39]	[-80.18, -75.10]	364
Middle East Coast	[35.25, 38.89]	[-74.96, -73.92]	447
Upper West Coast	[37.32, 48.56]	[-126.91, -124.85]	487

TABLE III

User-defined parameters for traffic rules

Parameter	Value	Parameter	Value
v_{max}	20 m s^{-1}	Δt	10 s
$\Delta_{\text{head-on}}$	5 deg	$t_{horizon}^{check}$	420 s
$\Delta_{\text{no\_turn}}$	10 deg	$t_{horizon}^{coll}$	300 s
$\Delta_{\text{large\_turn}}$	20 deg	t_{react}	60 s
		$t_{maneuver}$	60 s

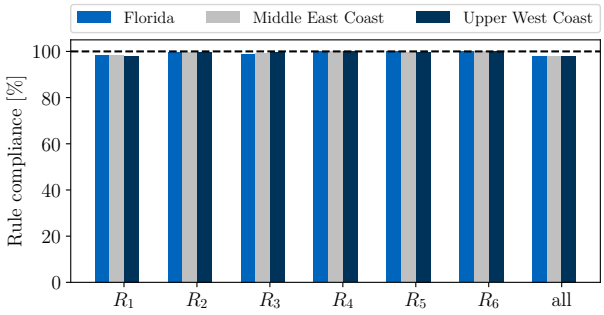


Fig. 5. Rule compliance for different geographical regions.

B. Results

For every location, we evaluate the number of vessels as indicated in Table II with the parameters specified by Table III. For each vessel, the rules are evaluated with respect to all other vessels in the scenario. Fig. 5 shows the rule compliance for different rules and regions. In general, the rule compliance at different locations is comparable. The individual rules R_1 - R_6 are mostly fulfilled with a compliance rate between 98 % and 100 %. On average, 98 % of all investigated vessels obey all formalized rules. As we investigate open-sea scenarios, where vessel often can easily keep a large distance to each other and have enough space to conduct collision avoidance maneuvers, the high rate of rule compliance confirms our expectations.

C. Discussion

The quantitative evaluation of the presented traffic rules has some limitations originating from the AIS data. In contrast to the vision-based reconstruction of scenarios, AIS data is less informative, and anomalies are difficult to detect. AIS data is asynchronously received, which makes interpolation necessary and, thus, can lead to deviations from the true path. Further, vessels without an AIS sender might sail in the area regarded as well, but are not visible in the data.

The user-defined parameters for evaluating the rules are based on the COLREGS and expert knowledge. For example, with the current value of $t_{horizon}^{check}$, the give-way vessel of two vessels sailing with 15 knots (i.e., 7.7 m s^{-1}) in a rectangular crossing situation would have to conduct a crossing maneuver when the distance between both vessels is approximately 2.5 nautical miles (i.e., 4500 m). This parameter setting might

[4]commonroad.in.tum.de/commonroad_io

[5]doi.org/10.24433/CO.8258454.v2

be too conservative for ship encounters close to shore but can be easily adapted if necessary.

We limited this study to encounters of power-driven vessels on the open sea. The data only includes encounters of two to four vessels, but the rules can be evaluated on any number of vessels. However, for these situations, the COLREGS are sometimes underspecified, e.g., a situation with three vessels where one vessel is overtaken by another one, whereas a third vessel crosses the path of the overtaken vessel from the left. Then, the overtaken vessel is in the stand-on position and the give-way position at the same time. Additionally, integrating the collision prevention hierarchy between vessel types (see rule 18 of COLREGS) would increase the applicability and could be done by switching between different rule sets depending on vessel types. Further, the last-minute-maneuver rule, which applies when one of the vessels violates the rules presented, highly depends on the situation (i.e., traffic, static obstacles, and weather conditions) and is insufficiently specified in the COLREGS to be readily formalizable. In general, MTL is very expressive and, thus, can be most likely used for further marine traffic rules as well. We will continuously update our formalized rule set and make it available online[6].

VII. Conclusions

We presented a temporal logic formalization of the COLREGS rules, which are essential for autonomous vessels. Without such a formalization, the rule compliance certification of autonomous vessels becomes much more difficult. The formalization is based on predicates and parameters that can be easily reused for specifying more marine traffic rules. The evaluation on real marine traffic data shows that most vessels in the investigated area and time obey the rules. The presented formalization allows the straightforward integration of marine traffic rules into motion planning of autonomous vessels. In addition, it can be used for the verification of marine traffic rule compliance independent of the used motion planner, thus demonstrating its general applicability.

Acknowledgment

The authors gratefully acknowledge the financial support of this work by the research training group CONVEY funded by the German Research Foundation under grant GRK 2428.

References

[1] European Maritime Safety Agency (EMSA), "Annual Overview of Marine Casualties and Incidents 2020," EMSA, Tech. Rep., 2020.

[2] "COLREGs: Convention on the International Regulations for Preventing Collisions at Sea," International Maritime Organization (IMO), 1972.

[3] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger, "Safe maritime autonomous navigation with COLREGS, using velocity obstacles," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 110–119, 2014.

[4] L. Zhao and M. I. Roh, "COLREGs-compliant multiship collision avoidance based on deep reinforcement learning," *Ocean Engineering*, vol. 191, pp. 106 436–106 450, 2019.

[6] gitlab.lrz.de/tum-cps/traffic-rules

[5] S. Guo, X. Zhang, Y. Zheng, and Y. Du, "An autonomous path planning model for unmanned ships based on deep reinforcement learning," *Sensors*, vol. 20, no. 2, 2020.

[6] X. Zhang, C. Wang, Y. Liu, and X. Chen, "Decision-making for the autonomous navigation of maritime autonomous surface ships based on scene division and deep reinforcement learning," *Sensors*, vol. 19, no. 18, 2019.

[7] M. Junmin, L. Mengxia, H. Weixuan, Z. Xiaohan, G. Shuai, C. Pengfei, and H. Yixiong, "Mechanism of dynamic automatic collision avoidance and the optimal route in multi-ship encounter situations," *Journal of Marine Science and Technology*, vol. 26, pp. 141–158, 2021.

[8] Y. He, Y. Jin, L. Huang, Y. Xiong, P. Chen, and J. Mou, "Quantitative analysis of COLREG rules and seamanship for autonomous collision avoidance at open sea," *Ocean Engineering*, vol. 140, pp. 281–291, 2017.

[9] H. T. L. Chiang and L. Tapia, "COLREG-RRT: An RRT-Based COLREGS-Compliant Motion Planner for Surface Vehicle Navigation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2024–2031, 2018.

[10] M. R. Benjamin and J. A. Curcio, "COLREGS-based navigation of autonomous marine vehicles," in *Proc. of the IEEE/OES Autonomous Underwater Vehicles*, 2004, pp. 32–39.

[11] T. A. Johansen, T. Perez, and A. Cristofaro, "Ship collision avoidance and COLREGS compliance using simulation-based control behavior selection with predictive hazard assessment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3407–3422, 2016.

[12] X. Geng, Y. Wang, P. Wang, and B. Zhang, "Motion plan of maritime autonomous surface ships by dynamic programming for collision avoidance and speed optimization," *Sensors*, vol. 19, no. 2, 2019.

[13] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

[14] R. Alur and T. A. Henzinger, "Real-Time Logics: Complexity and Expressiveness," *Information and Computation*, vol. 104, no. 1, pp. 35–77, 1993.

[15] P. Thati and G. Rou, "Monitoring Algorithms for Metric Temporal Logic Specifications," *Electronic Notes in Theoretical Computer Science*, vol. 113, pp. 145–162, 2005.

[16] P. Fiorini and Z. Shiller, "Motion Planning in Dynamic Environments Using Velocity Obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[17] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-Body Collision Avoidance," *Springer Tracts in Advanced Robotics*, vol. 70, pp. 3–19, 2011.

[18] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 752–759.

[19] E. Tu, G. Zhang, L. Rachmawati, E. Rajabally, and G. B. Huang, "Exploiting AIS Data for Intelligent Maritime Navigation: A Comprehensive Survey from Data to Methodology," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, pp. 1559–1582, 2018.

[20] "Marine Cadastre - Vessel Traffic Data," U.S. Coast Guard Navigation Center, Alexandria, USA, 2009 - 2020, https://marinecadastre.gov/ais/.

RightsLink

Sign in/Register

Temporal Logic Formalization of Marine Traffic Rules

Conference Proceedings: 2021 IEEE Intelligent Vehicles Symposium (IV)

Author: Hanna Krasowski

Publisher: IEEE

Date: 11 July 2021

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

B.2 Safe Reinforcement Learning with Probabilistic Guarantees Satisfying Temporal Logic Specifications in Continuous Action Spaces

Summary Temporal logic is capable of specifying complex notions of safety. While for discrete action and state spaces there are already powerful model checking methods that directly integrate temporal logic specifications with RL, the seamless integration with continuous spaces is an open research problem.

We use a probabilistically verified controller to guide the exploration of the RL agent whereby the agent employs a continuous action and observation space. We define the safety specification with STL and design a three-step process to achieve safe RL: verification of the controller on a system with stochastic disturbances from a compact set, safe RL with exploration constraint by the verified controller, and probabilistic verification of the learned RL agent. In particular, the RL exploration is bounded to the compact disturbance set used in the first step of verifying a controller. The probabilistic verification steps determine a probabilistic safety guarantee for the STL specification, whereas the RL agent improves other objectives such as efficiently fulfilling a task. We initially require a probabilistically verifiable controller for our approach. Such a controller is often available, e.g, a black-box controller learned from expert demonstrations.

We evaluate our approach on a proof-of-concept system, where a mobile robot has to evade a dynamic obstacle with unknown behavior in a specific manner and at the same time has to reach a goal area. We implement this system in a computer simulation and validate the learned RL agent in the physical world on mobile robots of the Robotarium. Our safety specification is a STL formula describing situations in which the safe evasion maneuver is necessary and how it needs to be executed. Results show that our approach maintains the probabilistic safety guarantees while improving the goal-reaching performance.

Author contributions H.K. and P.A. came up with the initial idea of combing probabilistic verification with safe RL and identified a proof-of-concept system. H.K. implemented the approach. H.K. and P.A. conducted the numerical experiments. H.K. wrote the majority of the manuscript with support from P.A. for Sec. II and the Corollaries 1 and 2. A.A. and M.A. provided feedback improving the manuscript.

TUM Graduate School This publication has been declared a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

Safe Reinforcement Learning with Probabilistic Guarantees Satisfying Temporal Logic Specifications in Continuous Action Spaces

Hanna Krasowski, Prithvi Akella, Aaron D. Ames, and Matthias Althoff

Abstract— **Vanilla Reinforcement Learning (RL) can efficiently solve complex tasks but does not provide any guarantees on system behavior. To bridge this gap, we propose a three-step safe RL procedure for continuous action spaces that provides probabilistic guarantees with respect to temporal logic specifications. First, our approach probabilistically verifies a candidate controller with respect to a temporal logic specification while randomizing the control inputs to the system within a bounded set. Second, we improve the performance of this probabilistically verified controller by adding an RL agent that optimizes the verified controller for performance in the same bounded set around the control input. Third, we verify probabilistic safety guarantees with respect to temporal logic specifications for the learned agent. Our approach is efficiently implementable for continuous action and state spaces. The separation of safety verification and performance improvement into two distinct steps realizes both explicit probabilistic safety guarantees and a straightforward RL setup that focuses on performance. We evaluate our approach on an evasion task where a robot has to reach a goal while evading a dynamic obstacle with a specific maneuver. Our results show that our safe RL approach leads to efficient learning while maintaining its probabilistic safety specification.**

I. Introduction

Reinforcement Learning (RL) has the potential to solve intricate tasks by learning complex policies efficiently. However, vanilla RL cannot provide (probabilistic) safety guarantees, which is essential for real-world applications. Formal methods can eliminate this problem when integrated into the learning process. The most prominent formal methods approaches achieving safety guarantees for RL with continuous action spaces are control-theoretic methods such as model predictive control [1], [2], control barrier functions [3], or reachability analysis [4]–[6]. However, all these methods only handle reach-avoid safety specifications since they either determine unsafe state sets and prohibit the RL agent from entering them or determine safe state sets and force the RL agent to remain within those. Other methods are needed whenever the safety specification is more complex and cannot be seamlessly translated into a reach-avoid problem.

One way of expressing more complex safety specifications is via temporal logic. Indeed, there has been significant work that combines RL with logical specifications for discrete action spaces [7]–[9]. For example, Alshiekh et al. [7] filter all unsafe actions proposed by the RL agent with a safety shield synthesized from linear temporal logic specifications. Hasanbeig et al. [9] include the temporal logic specifications in the RL process through a pessimistic and an optimistic learner where the pessimistic learner limits the exploration to low-risk actions. Still, applying approaches for discrete action spaces to real-world systems with continuous control inputs requires a low-level controller that converts the discrete actions to continuous inputs.

Other RL approaches realize continuous actions and guide the agent by a temporal logic specification that includes safety and performance objectives [10]–[12], *i.e.*, the temporal logic specification describes the entire task. For example, Cai et al. [12] transform linear temporal logic into a Büchi automaton, which is integrated into RL and yields probabilistic guarantees. Although specifying the task via temporal logic for RL is promising, safety and performance objectives included in the task are potentially not aligned. Possible solutions are to provide feedback to the user whenever the temporal logic specification becomes infeasible [11] or to trade off between safety and performance [12]. The first solution is usually not practical for autonomous real-world systems, and the second one does not provide an explicit probabilistic guarantee for the safety specification, which might be required. Instead, our approach separates safety and performance objectives such that we obtain probabilistic safety guarantees while the feasibility is ensured by utilizing a probabilistically verified safe controller.

To provide probabilistic safety guarantees, we leverage existing work in the probabilistic verification literature taking a scenario approach to risk-aware probabilistic verification [13], [14]. Here, the standard approach as described in [15] is to pose verification as an optimization problem minimizing a quantifiable satisfaction measure provided by either a temporal logic specification or another method.

Contribution: We propose a three-step safe RL approach that improves the performance of a probabilistically verified black-box controller and results in probabilistic safety guarantees for the learned agent. Our key idea is to separate safety and performance objectives in distinct steps (see Fig. 1 with probabilistic verification steps for safety and RL for performance), which leads to efficient RL while providing explicit safety guarantees. In contrast to existing methods, our approach is suited for complex real-world systems since it is tailored to continuous action spaces, can probabilistically verify arbitrary Signal Temporal Logic (STL) specifications,

The authors gratefully acknowledge the partial financial support of this work by the research training group ConVeY funded by the German Research Foundation under grant GRK 2428, by the project TRAITS funded by the German Federal Ministry of Education and Research, and by an IFI scholarship funded by the DAAD. Prithvi Akella was supported the Air Force Office of Scientific Research, grant FA9550-19-1-0302, and the National Science Foundation, grant 1932091.

H. Krasowski and M. Althoff are with the Technical University of Munich, Munich, Germany {hanna.krasowski, althoff}@tum.de

H. Krasowski, P. Akella and A. D. Ames are with the California Institute of Technology, Pasadena, USA {pakella, ames}@caltech.edu

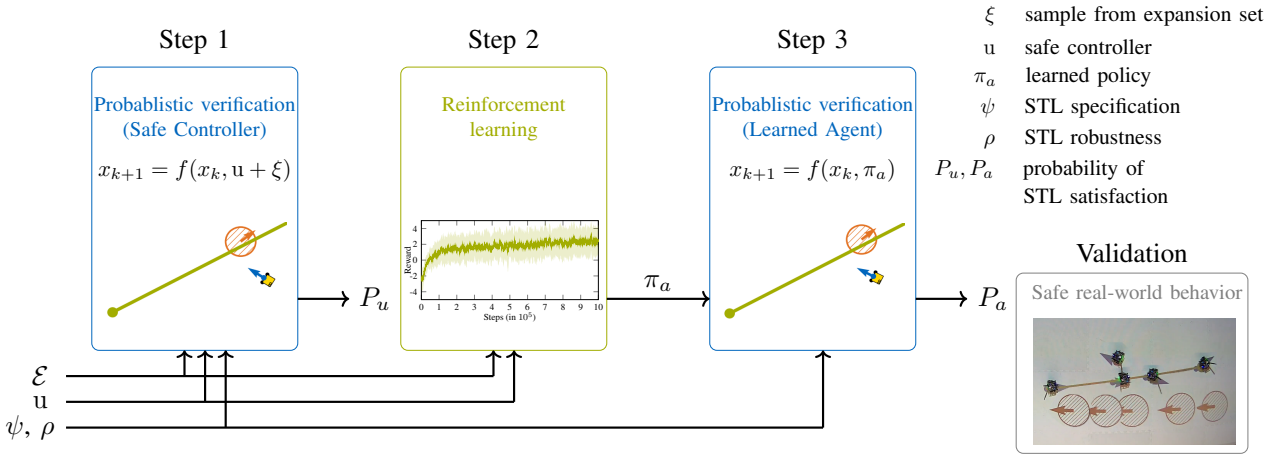


Fig. 1. Safe RL process for STL safety specification ψ with robustness measure ρ using a safe controller u and a system model $x_{k+1} = f(x_k, \cdot)$ where $\cdot$ can be replaced by any controller.

and does not require a system model to predict the safety of future actions. We validate our approach on an evasion task in a simulated dynamic environment and show that it can be translated to real-world systems as demonstrated through experiential results on the Robotarium [16].

Structure: The remainder of this paper is organized as follows. First, we introduce preliminary concepts in Sec. II. Second, we present our safe RL approach in Sec. III. Then, we explain the details of our safe evasion task and its experimental validation in Sec. IV. Finally, we discuss our approach in Sec. V and conclude in Sec. VI.

II. Preliminaries

Signal Temporal Logic STL is a language by which rich, time-varying system behavior can be expressed succinctly and concisely. STL is based on predicates τ which are Boolean-valued functions taking a truth value for each state $x \in \mathcal{X}$. Predicates τ and specifications ψ are defined in Backus-Naur notation [17, Section 2.1] with respect to predicate functions h_τ that define subsets of a state space $\mathcal{X}$ where τ evaluates to True:

$$\tau(x) = \text{True} \iff h_\tau(x) \geq 0,\ h_\tau : \mathcal{X} \to \mathbb{R}, \qquad (1)$$
$$\psi \triangleq \tau \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \psi_1 \,\mathrm{U}_{[a,b]}\, \psi_2,$$

where, $\psi \in \mathbb{S}$, and $a, b \in \mathbb{R}_{\geq 0} \cup \{\infty\},\ b \geq a$. Here, $\mathbb{S}$ is the set of all STL specifications which are evaluated over signals $s : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, and the space of all signals $\mathcal{S}^{\mathrm{n}} = \{s \mid s : \mathbb{R}_{\geq 0} \to \mathbb{R}^n\}$. Finally, we denote that a signal s satisfies ψ at time t via $(s, t) \models \psi$. Furthermore, every STL specification ψ has a *robustness measure* ρ [18]:

Definition 1. *A function $\rho : \mathcal{S}^{\mathrm{n}} \times \mathbb{R}_+ \to \mathbb{R}$ is a robustness measure for an STL specification ψ if it satisfies: $\rho(s, t) \geq 0 \iff (s, t) \models \psi$.*

Example 1. *Let $\psi = \neg(\text{True}\,\mathrm{U}_{[0,2]}\,|s(t)| > 2)$, then any real-valued signal $s : \mathbb{R}_{\geq 0} \to \mathbb{R}$ satisfies ψ at time t, i.e. $(s, t) \models \psi$ if $\forall\, t' \in [t, t{+}2]$, $|s(t')| \leq 2$. A possible robustness measure is $\rho(s, t) = \min_{t' \in [t, t+2]}\ 2 - |s(t')|$.*

Note that while defining a robustness measure as per Definition 1 aligns with prior works [19], [20] and our predicate definition in (1), it is not the only way of defining such a measure, *e.g.* see Definition 3 in [21] or Section 2.3 in [22].

Probabilistic Controller Verification As expressed in Section 3 in [15], STL provides a natural way of phrasing black-box controller verification as an optimization problem over a space of parameters $p \in \mathcal{P}$ affecting signal generation. More specifically, let $\mathcal{P}$ be a space of parameters denoting different environmental states in which we expect our closed-loop system to operate. For example, for warehouse robotics, these environmental states could be package and drop-off locations, the floor plan, *etc*. Additionally, for any specific environment parameter p, there may exist disturbances affecting system behavior. Thus, we expect the closed-loop trajectory ϕ_p, which is realized by our system for a specific environment parameter p, to be a sample of a p-parameterized random variable Φ_p with corresponding distribution π_p. To formulate the theorem used in this work, let us introduce $\mathrm{U}[\cdot]$ as uniform distribution and $\mathbb{P}_\triangle$ denoting a probability where $\triangle$ indicates the underlying distribution.

Theorem 1. *(Adapted from [13, Thm. 7]) Let $\mathcal{P}$ be a space that admits a uniform distribution and let $\mathcal{D} = \{r_i = \rho(\phi_{p_i})\}_{i=1}^N$ be a set of N closed-loop system robustnesses r_i, evaluating the robustness of one closed-loop trajectory sample ϕ_{p_i} per i.i.d. sample p_i drawn from the uniform distribution over $\mathcal{P}$. Furthermore, define $\rho_N^* = \min\{r_i \in \mathcal{D}\}$. For any $\epsilon \in [0, 1]$, the probability that ρ_N^* underperforms the $1-\epsilon$-th quartile of possible robustness values is bounded below by $1 - (1-\epsilon)^N$, i.e. with $\mu \triangleq \mathrm{U}[\mathcal{P}] \times \pi_p$,*

$$\mathbb{P}_\mu^N \left[\mathbb{P}_\mu[\rho(\phi_p) \geq \rho_N^*] \geq 1 - \epsilon\right] \geq 1 - (1-\epsilon)^N. \qquad (2)$$

For a more detailed derivation of this theorem and its implications on dimensional scaling, we refer the interested reader to [13].

Overarching Problem Statement: We assume that we have a safety specification ψ expressed in STL and an associated robustness measure ρ as per Definition 1. We also assume

that a model and black-box controller u are available:

$$x_{k+1} = f(x_k, u_k),\ x_k, x_{k+1} \in \mathcal{X} \subseteq \mathbb{R}^n,\quad u_k \in \mathcal{U} \subseteq \mathbb{R}^m,\ \mathrm{u} : \mathcal{X} \to \mathcal{U}, \tag{3}$$

where $\mathrm{u}(x_k) = u_k$ and the subscripts denote the time step. Additionally, we assume that the distribution of system behavior π_p is time-invariant. Note that we do not need to explicitly know the model f but can also employ a black-box simulation environment. The problem is to ensure probabilistic safety guarantees specified via STL for the given system while using RL for improving the performance.

III. Safe RL Process

Step One: Our safe RL concept consists of three steps as shown in Fig. 1. Our first step is to follow the probabilistic verification procedure outlined in Sec. II to verify whether the controller u realizes safe behavior when adding the uniformly sampled disturbance ξ:

$$x_{k+1} = f(x_k, \mathrm{u}(x_k) + \xi), \quad \xi \sim \mathrm{U}[\mathcal{E}],\ \mathcal{E} \subseteq \mathbb{R}^m. \tag{4}$$

Here, $\mathrm{U}[\mathcal{E}]$ corresponds to the uniform distribution over the set $\mathcal{E}$, which we assume to be fixed and independent of the system state $x \in \mathcal{X}$. Per Theorem 1, we can determine the following probabilistic lower bound on the robustness measure value achievable by the closed-loop system (4).

Corollary 1. *Let the system dynamics be as per* (4)*, the safety specification ψ have robustness measure ρ as per Definition 1, and $\mathcal{D} = \{r_i = \rho(\phi_{x_0^i})\}_{i=1}^N$ be the robustnesses of N trajectories $\phi_{x_0^i}$ where the initial conditions x_0^i were uniformly sampled over $\mathcal{X}$. Define $\rho_N^* = \min\{r_i \in \mathcal{D}\}$, then for some $\epsilon \in [0,1]$, ρ_N^* underperforms the $1-\epsilon$-th quartile robustnesses achievable by the stochastic closed-loop system in* (4) *with minimum confidence $1-(1-\epsilon)^N$, i.e. with $\mu = \mathrm{U}[\mathcal{X}] \times \mathrm{U}[\mathcal{E}] \times \mathrm{U}[\mathcal{E}] \times \ldots$,*

$$\mathbb{P}_\mu^N \left[\mathbb{P}_\mu \left[\rho(\phi_{x_0}) \geq \rho_N^* \right] \geq 1 - \epsilon \right] \geq 1 - (1-\epsilon)^N.$$

Proof: This is an application of Theorem 1. ∎

Following Corollary 1, we can identify the robustness set $\mathcal{D}$ by sampling N trajectories of the closed-loop system with controller u under bounded input disturbance from $\mathcal{E}$. Given the robustness set $\mathcal{D}$ and a specified ϵ, we can evaluate the probabilities in Corollary 1 and obtain ρ_N^*. We abbreviate this probabilistic verification process with the function $\mathrm{probv}(f, \mathrm{u}, \mathcal{E}, N, \epsilon)$. The first step concludes once we verified that a candidate controller u achieves safe behavior with a high probability, *i.e.* $\rho_N^* \geq 0$ with $1 - \epsilon \approx 1$.

Remark on Maximizing $\mathcal{E}$: The second step in our safe RL process will be to learn a policy that chooses disturbances within the expansion set $\mathcal{E}$. As such, maximizing the size of this set has a direct impact on the ability of our procedure to learn a performant policy. To that end, we propose a possible algorithm to expand $\mathcal{E}$ in Alg. 1.

In more detail, in line 2 of Alg. 1, the initial expansion set is verified. If this initial set is not verifiable, a smaller initial set must be provided. Otherwise, the expansion set $\mathcal{E}_{temp}$, increased iteratively through $\Delta\mathbf{f}$ (line 5), is subsequently verified (line 6). Once the verification is not successful anymore, the algorithm terminates and returns the largest verified expansion set $\mathcal{E}$ (line 10). Note that we use an interval for $\mathcal{E}$ for simplicity. However, other set representation such as zonotopes would be possible. To identify a more expressive expansion set, conformance checking [23] could be used.

Algorithm 1 findExpansionSet()

Input: executable system f, controller u, initial expansion interval set $\mathcal{E}_{init}$, vector of fractions to increase set $\Delta\mathbf{f} \in \mathbb{R}^m$, number of samples N, quartile parameter ϵ
Output: Final expansion set $\mathcal{E}$

1: $i = 1$
2: $\rho_N^* = \mathrm{probv}(f, \mathrm{u}, \mathcal{E}_{init}, N, \epsilon)$
3: **while** $\rho_N^* \geq 0$ **do**
4: $\quad \mathcal{E} = (\mathbf{1}_m + (i-1)\,\Delta\mathbf{f})\,\mathcal{E}_{init}$
5: $\quad \mathcal{E}_{temp} = (\mathbf{1}_m + i\,\Delta\mathbf{f})\,\mathcal{E}_{init}$
6: $\quad \rho_N^* = \mathrm{probv}(f, \mathrm{u}, \mathcal{E}_{temp}, N, \epsilon)$
7: $\quad i \leftarrow i + 1$
8: **end while**
9: **if** $i \neq 1$ **then**
10: $\quad$ **return** $\mathcal{E}$
11: **else**
12: $\quad$ **return** Reduce $\mathcal{E}_{init}$ as too large to verify
13: **end if**

Step Two: The second step in our safe RL process is to learn a controller that improves for performance of the safe controller u we verified in the prior step. To constrain the learning based on the safe controller, we define a state-dependent action space $\mathcal{A}(x)$ around the safe control input $\mathrm{u}(x)$ inspired by continuous action masking [24]:

$$\mathcal{A}(x) = \mathrm{u}(x) \oplus \mathcal{E}, \tag{5}$$

where $\oplus$ denotes the Minkowski sum. Based on Corollary 1, we can compute the probabilistic guarantee (usually ≈ 1) for the closed-loop system when perturbing the safe input $\mathrm{u}(x)$ with uniformly sampled noise within the expansion set $\mathcal{E}$. Therefore, if we continuously choose actions $a \in \mathcal{A}(x)$, our learned agent will with high probability yield safe behavior that also fulfills the probabilistic safety specification. Consequently, the agents can focus on optimizing for performance when learning within $\mathcal{E}$ around the safe controller. Thus, our three-step process delineates the safety (step one and three) and performance aspects (step two). This simplifies the reward and observation definitions as we only need to consider performance. Fig. 2 depicts this learning process and shows how the RL agent can effect a system trajectory in the state space by changing the control input within $\mathcal{A}(x)$.

Step Three: The third step is to verify the learned agent with a deterministic policy $\pi_a : \mathcal{X} \to \mathcal{U}$, to ensure the preservation of safety after optimizing for performance through learning. This is necessary since the learned agent (obtained by step two) will be different from the probabilistically verified safe controller with disturbance uniformly sampled from $\mathcal{E}$ (verified in step one). Thus, the probabilistic verification

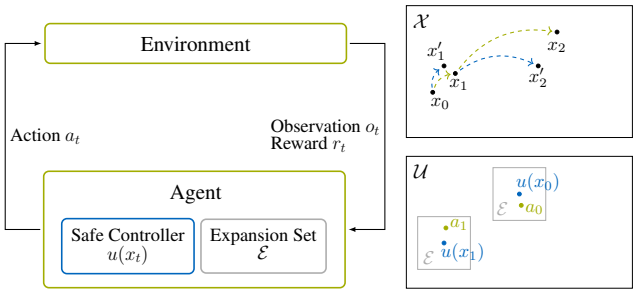


Fig. 2. RL within expansion set $\mathcal{E}$ around the safe controller $u(x)$ with trajectories for state space $\mathcal{X}$ and input space $\mathcal{U}$. The states x are reached by the RL actions, and x' are states that would be reached by the safe controller.

result is likely to hold but still needs to be verified for the learned agent. This verification, however, amounts to one more implementation of Theorem 1:

Corollary 2. *Let the system dynamics be as per* (3) *with a learned RL agent* $\pi_a : \mathcal{X} \to \mathcal{U}$ *as controller, let the system safety specification* ψ *have a robustness measure* ρ *as per Definition 1, and let* $\mathcal{D}$ *and* ρ_N^* *be as in Corollary 1. Then for some* $\epsilon \in [0, 1]$, ρ_N^* *underperforms the* $1-\epsilon$*-th quartile of robustnesses achievable by the learned system with minimum confidence* $1 - (1 - \epsilon)^N$.

$$\mathbb{P}^N_{\mathrm{U}[\mathcal{X}]}\left[\mathbb{P}_{\mathrm{U}[\mathcal{X}]}\left[\rho(\phi_{x_0}) \geq \rho_N^*\right] \geq 1 - \epsilon\right] \geq 1 - (1 - \epsilon)^N.$$

Proof: This is an application of Theorem 1. ∎

IV. Safe Evasion Task with Experimental Validation

To make our high-level approach more tractable, we define a problem with a safety specification that is more complex than a simple reach-avoid specification. Specifically, the mobile robot's task is to follow an optimal path to the goal while complying with a temporal logic safety specification – whenever a collision is possible with the dynamic obstacle of the environment within the next few time steps, the mobile robot has to evade in a specified manner. Relevant examples for real-world applications where only a specific evasion is safe are: autonomous vehicles that have to overtake another traffic participant in a specific lane [25] or autonomous vessels that have to perform specific collision avoidance maneuvers in order to be predictable for other ships [26]. We first describe the safety specification and RL problem. Our experiments are conducted on the Robotarium [16] and its simulation.

A. Safety Specification

The state of the robot is $r = [x_r, y_r, \theta_r, v_r] \in \mathbb{R}^4$ where x_r and y_r describe the position of the robot, θ_r is its orientation, and v_r is its velocity aligned with its orientation. There is always one dynamic obstacle present and it is described by the state $o = [x_o, y_o, \theta_o, v_o] \in \mathbb{R}^4$. The time step is Δt. We assume a unicycle model for the robot for which control inputs u are velocity and turning rate. Note that for the Robotarium, we provide the same control inputs and their robot controller generates the corresponding actuator signals.

Our safety specification is as follows: When the projected positions of the robot and any obstacle are closer than $0.4\,\mathrm{m}$ for any time steps within $1\,\mathrm{s}$, then the robot should evade in a specific manner that depends on the relative positions and orientations of the obstacle and agent. To formalize the specification with STL, we first need to define a few predicates and functions.

We can convert heading angles to unit vectors via $\mathrm{a2v}(\phi) = [\cos(\phi), \sin(\phi)]$. The rotation matrix for an angle α is

$$R_\alpha = \begin{bmatrix} \cos(-\alpha) & \sin(-\alpha) \\ -\sin(-\alpha) & \cos(-\alpha) \end{bmatrix}.$$

The function $\mathrm{sgn}(x)$ returns -1 if $x < 0$, 1 if $x > 0$ and 0 otherwise. The minimum distance function $\mathrm{MD}(r, o, \Delta t)$ is defined as:

$$\mathrm{MD}(r, o, \Delta t) = \min_{t \in \{0, \Delta t, \ldots, 1\,\mathrm{s}\}} \Bigg(\Bigg\| \left(\begin{bmatrix} x_r \\ y_r \end{bmatrix} + \mathrm{a2v}(\theta_r) v_r t \right) - \left(\begin{bmatrix} x_o \\ y_o \end{bmatrix} + \mathrm{a2v}(\theta_o) v_o t \right) \Bigg\|_2 \Bigg),$$

where the time step size is $\Delta t = 0.033\,\mathrm{s}$. The minimum distance prediction assumes that the robot and the obstacle will move in their current directions with their current speeds, which is often a reasonable prediction. The predicate $\mathrm{IH}(r, o)$ checks if the obstacle is in the halfspace in front of the vehicle, *i.e.* it evaluates to true iff $[x_o, y_o] \cdot \mathrm{a2v}(\theta_r) - b_r \geq 0$ where b_r is the offset. Safe evasion is specified by the predicate:

$$\mathrm{EV}(\dot\theta_r, \Delta\theta, sign) = \begin{cases} \text{True}, & \text{iff } (|\dot\theta_r| \leq 1.5\,\mathrm{rad\,s^{-1}} \\ & \wedge \mathrm{sgn}(\dot\theta_r) = sign) \vee \\ & ((\Delta\theta \geq 0\,\mathrm{rad} \\ & \vee |\Delta\theta| \leq 0.01\,\mathrm{rad}) \\ & \wedge 0.01\,\mathrm{rad\,s^{-1}} \geq |\dot\theta_r|) \\ \text{False}, & \text{otherwise,} \end{cases}$$

where $\Delta\theta \in [-\pi, \pi]$ is the orientation difference to the orientation perpendicular to the direction of the straight path between the initial state and goal in the direction of turning; $sign$ is -1 if the robot should turn to the left and 1 in case the robot should turn to the right. See Fig. 3 for a depiction of the case-by-case evasion situations. Note that for our task specification with one non-reactive dynamic obstacle, the four cases are exhaustive for identifying the direction of evasion. To give an intuition, a safe evasion maneuver is that the robot turns until its orientation is perpendicular to the straight path between the initial state and goal. If this orientation is reached, the robot is no longer required to turn[1] and continues driving away from the direct path between the initial state and goal to evade further.

With these predicates and functions, the STL formula is:

$$G\Big((\mathrm{IH}(r, o) \wedge \mathrm{MD}(r, o, \Delta t) \leq 0.4\,\mathrm{m}) \implies \mathrm{EV}(\dot\theta_r, \Delta\theta, sign) \Big). \tag{6}$$

[1]No turning would be an impossible requirement due to the stochasticity needed for the input space, which includes the turning rate.

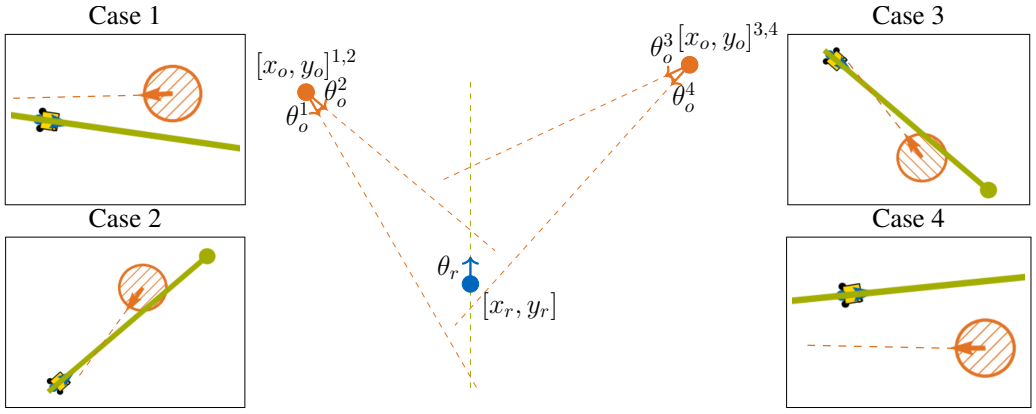


Fig. 3. Visualization of the four different relative orientations and position cases. The obstacle is depicted in orange and the robot in blue, and the case is indicated by the superscript. For cases 1 and 3, the robot should turn right (i.e., $sign = 1$) and for the cases 2 and 4, the robot should turn left (i.e., $sign = -1$).

The robustness measure for this STL formula is:

$$\rho = \begin{cases} -1, & \text{iff eq. (6)} = False \\ \sum_{k=1}^{K} \text{prfm}(r_K, r_0, goal, K), & \text{otherwise,} \end{cases}$$

where K is the length of the trajectory, $goal$ is the position of the goal, r_K is the position of the robot at the end of the trajectory, r_0 is the initial position of the robot, and the performance objective is defined by

$$\text{prfm}(r_K, r_0, goal, K) = \max\left(\left(1 - \frac{\|r_K - goal\|_2}{\|r_0 - goal\|_2}\right), 0\right) + \left(1 - \frac{K}{K_{max}}\right),$$

with the maximal length of a trajectory $K_{max} = 300$. Note that although the objective of this safety specification is collision avoidance, which can also be achieved with other formal methods such as control barrier functions, these methods cannot easily be used to guarantee a specific avoidance behaviour as defined in the predicate EV. Additionally, note that this particular safety specification can also be expressed with Linear Temporal Logic (LTL). However, as STL is more expressive than LTL, STL can be utilized to express more complex specifications, *e.g.* for marine traffic rules [26].

B. Reinforcement Learning Problem

We obtain the action space as described in (5). The reward function R is the difference between the achieved state with the RL control input and only with the safe control input

$$\text{R} = r_{\text{diff}} \cdot (\|goal - [x_{sc}, y_{sc}]\|_2 - \|goal - [x_r, y_r]\|_2),$$

where $r_{\text{diff}} \in \mathbb{R}_+$ scales the reward, and $[x_{sc}, y_{sc}]$ is the agent's position if the input from the safe controller would be used in the previous state, *i.e.* $a_{t-1} = u_{t-1}$. Our agent can observe its relative position to the goal, the relative position to the closest point on the optimal path, the orientation difference to the optimal orientation calculated from the initial position to the goal, and the relative position to the obstacle. Note that this observation differs from the state.

We used Proximal Policy Optimization (PPO) [27] as our RL algorithm and scaled the reward R with the factor r_{diff} such that the reward is approximately between -5 and 5 per episode. We identified the best hyperparameters for PPO with a small search of eight different configurations and three random seeds. However, all tested hyperparameters showed a converging behavior. The hyperparameters that differ from the default PPO configuration of stable-baselines3 [28] are the network architecture (two-layer multi-layer perceptron with 128 neurons in each layer), the value function coefficient (0.05), and the entropy coefficient (0.01). We train the agent for one million training steps.

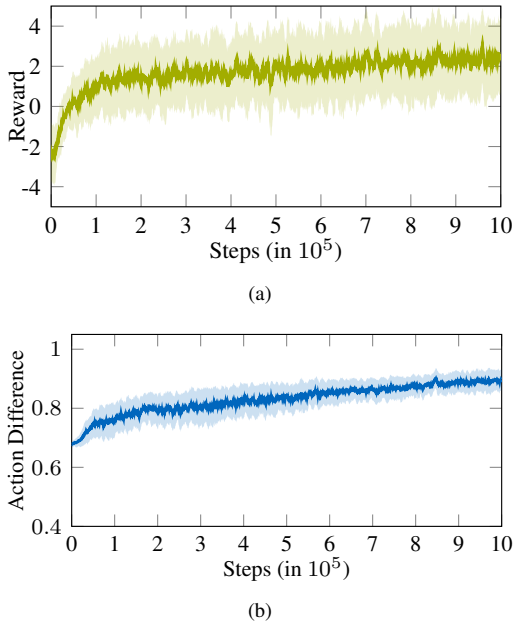


Fig. 4. Training curves for (a) reward function and (b) Euclidean norm difference between the learned agent's action and the safe control input scaled between 0 and 1. The action difference of a step is one if the agent selects an action on the border of the expansion set and zero if the agent does not alter the safe control input. The curves depict the mean and standard deviation for five different random seeds with the same hyperparameters.

Verification of Safe Controller (Step 1): We want to verify whether our safe controller with uniformly sampled

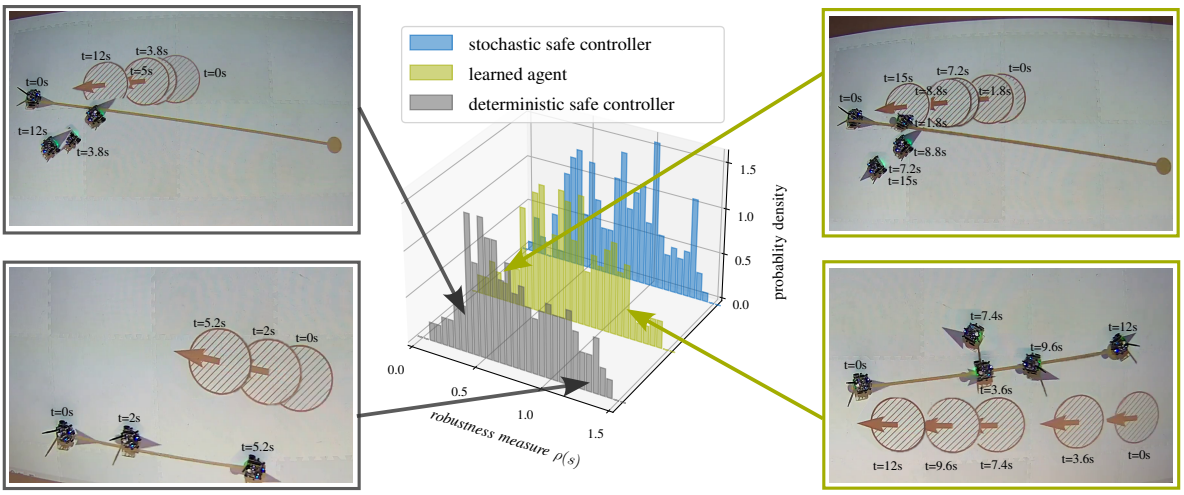


Fig. 5. Center: Robustness measure $\rho(s)$ histograms for 200 samples; Robotarium trajectories for different robustness values for learned agent (left top 0.114, bottom 1.62) and for safe controller (right top: 0.089, bottom: 1.21).

perturbations from the expansion set $\mathcal{E}$ achieves safe behavior with 95% probability ($\epsilon = 0.05$), *i.e.* has a probabilistic cutoff $\rho^*_N > 0$ as per Theorem 1 with $N = 50$ samples. We apply Alg. 1 with $\mathcal{E}_{init} = [-0.0002, 0.0002]\mathrm{m\,s^{-1}} \times [-0.005, 0.005]\mathrm{rad\,s^{-1}}$ and $\Delta\mathbf{f} = [10, 1.0]$ and obtain $\mathcal{E} = [-0.002, 0.002]\mathrm{m\,s^{-1}} \times [-0.01, 0.01]\mathrm{rad\,s^{-1}}$. Verifying the safe controller on the unicycle model with $N = 50$ samples, yielded a probabilistic cutoff $\rho^*_{50} = 0.276$ (see Fig. 5). This indicates that our chosen safe controller successfully exhibits safe behavior with 95% probability, and we are 92.3% confident in this statement — probabilities were calculated via substitution in (2).

Reinforcement Learning (Step 2): We expect that if we define our action space as in (5), the previous verification result will most likely hold for the learned agent. Fig. 4 depicts the episode reward and action difference to the safe control input over the training steps. A reward above zero indicates that the agent performs better with respect to goal reaching than the safe controller. This is the case after approximately 50000 training steps. Additionally, the difference between the safe control input and the agent's action increases over the training towards the agent mainly selecting actions close to its action space boundary.

Verification of Learned Agent (Step 3): Following the same verification procedure as for the stochastic safe controller yielded probabilistic cutoffs $\rho^*_{50} = 0.276$ for the safe controller without perturbations and $\rho^*_{50} = 0.221$ for the learned agent. Additionally, Fig. 5 shows the robustness measure histograms for 200 samples. For the learned agent, the distribution is shifted to slightly higher robustness values (robustness measure mean is 0.78 and standard deviation is 0.32) in comparison to the deterministic safe controller (robustness measure mean is 0.76 and standard deviation is 0.35). Since positive robustness measure values encode faster goal reaching, our numerical results imply that the learned agent improves the performance of the deterministic safe controller, while still exhibiting the same probabilistic level of STL specification compliance. The histogram for the stochastic safe controller has the same range and a similar shape as the histogram for the learned agent, which indicates that the verification result of the perturbed safe controller is, in fact, a good approximator of learned system behavior.

Testing on Robotarium Robot: Although our implementation is in Python and did not focus on efficiency, it was not necessary to improve the computational efficiency to be real-time capable for the Robotarium robot [16]. This is because we mainly run the forward evaluation of the policy network and safe controller online, which are computationally lightweight, and there is no need to predict and incorporate the safety of actions online which is often computation heavy. Fig. 5 shows example trajectories from the Robotarium experiments[2]. We observe that for high robustness values the robot only has to evade shortly or not at all. The main reason for low robustness values is that the robot has to evade and the final time for our experiment is reached before the robot can return to the optimal path and reach the goal.

V. Discussion

Our implementation is a proof of concept for our probabilistically safe RL approach and shows that for the safe evasion task probabilistic safety guarantees are not jeopardized by improving the performance with RL. Since other approaches using RL with temporal logic for safety specifications tackle a different problem (see Sec. I), we compare our approach only with the baseline of the safe controller's performance. A more complex problem would be autonomous driving on highways, where it is very hard to fulfill performance and safety specifications simultaneously during controller synthesis but often a safe controller exists. In the future such more complex problems should be investigated but are out of scope for this paper.

[2]Video of example trajectories: `https://youtu.be/8WWmKfh6WSM`

An important assumption of our approach is that a safe (black-box) controller is available. This assumption can be satisfied through the utilization of existing methods to synthesize safe controllers from temporal logic specifications [29], [30] or from expert data via imitation learning [31]. Note that, we only require this controller to satisfy a safety specification, *e.g.* avoid obstacles, and operate within safe bounds. Furthermore, as in the case of our experiment, it can be relatively easy to implement a safe controller. In particular, our safe controller consists of a high-level algorithm that provides waypoints to prevent a collision or track the optimal path as the situation requires. A low-level controller then tracks these waypoints and provides control inputs for the unicycle model. Note that we do not need to know the architecture of the safe controller as our approach regards it as a black-box.

VI. Conclusion

The proposed three-step safe RL approach achieves effective behavior that satisfies a desired probabilistic STL specification. Our results on a safe evasion task show that the separation of safety and performance leads to lean and efficient learning and the probabilistic STL guarantees can easily be verified. The architecture of our approach removes the necessity of online predictions for the safety of actions, which, as a consequence, reduces the computation time and allows us to effortlessly run the controller in real-time.

References

[1] S. Gros, M. Zanon, and A. Bemporad, "Safe reinforcement learning via projection on a safe set: How to achieve optimality?" *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8076–8081, 2020.

[2] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, no. 1, pp. 109 597–109 614, 2021.

[3] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. of the AAAI Conf. on Artificial Intelligence*, 2019, pp. 3387–3395.

[4] A. K. Akametalu, S. Kaynama, J. F. Fisac, M. N. Zeilinger, J. H. Gillula, and C. J. Tomlin, "Reachability-based safe learning with Gaussian processes," in *Proc. of the IEEE Conference on Decision and Control*, 2014, pp. 1424–1431.

[5] N. Kochdumper, H. Krasowski, X. Wang, S. Bak, and M. Althoff, "Provably safe reinforcement learning via action projection using reachability analysis and polynomial zonotopes," *IEEE Open Journal of Control Systems*, vol. 2, pp. 79–92, 2023.

[6] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan, "Reachability-Based Trajectory Safeguard (RTS): A safe and fast reinforcement learning safety layer for continuous control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3663–3670, 2021.

[7] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. of the AAAI Conf. on Artificial Intelligence*, 2018, pp. 2669–2678.

[8] B. Könighofer, F. Lorber, N. Jansen, and R. Bloem, "Shield synthesis for reinforcement learning," in *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*, 2020, pp. 290–306.

[9] M. Hasanbeig, A. Abate, and D. Kroening, "Cautious reinforcement learning with logical constraints," in *Proc. of the International Conference on Autonomous Agents and Multiagent Systems*, 2020, pp. 483–491.

[10] A. Lavaei, F. Somenzi, S. Soudjani, A. Trivedi, and M. Zamani, "Formal controller synthesis for continuous-space MDPs via model-free reinforcement learning," in *Proc. of the ACM/IEEE International Conference on Cyber-Physical Systems*, 2020, pp. 98–107.

[11] D. Gundana and H. Kress-Gazit, "Event-based signal temporal logic synthesis for single and multi-robot tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3687–3694, 2021.

[12] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7973–7980, 2021.

[13] P. Akella, A. Dixit, M. Ahmadi, J. W. Burdick, and A. D. Ames, "Sample-based bounds for coherent risk measures: Applications to policy synthesis and verification," *arXiv preprint arXiv:2204.09833*, 2022.

[14] B. Weng, G. A. Castillo, W. Zhang, and A. Hereid, "On safety testing, validation, and characterization with scenario-sampling: A case study of legged robots," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 5179–5186.

[15] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer, "A survey of algorithms for black-box safety validation of cyber-physical systems," *Journal of Artificial Intelligence Research*, vol. 72, pp. 377–428, 2021.

[16] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2017, pp. 1699–1706.

[17] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Nickovi ć, and S. Sankaranarayanan, *Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications*. Springer, 2018.

[18] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[19] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 96–101, 2018.

[20] ——, "Barrier function based collaborative control of multiple robots under signal temporal logic tasks," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1916–1928, 2020.

[21] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Proc. of the International Conference on Formal Modeling and Analysis of Timed Systems*, 2010, pp. 92–106.

[22] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[23] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff, "Model conformance for cyber-physical systems: A survey," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 3, pp. 1–26, 2019.

[24] H. Krasowski, J. Thumm, M. Müller, L. Schäfer, X. Wang, and M. Althoff, "Provably safe reinforcement learning: A theoretical and experimental comparison," *arXiv preprint arXiv:2205.06750*, 2022.

[25] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 752–759.

[26] T. R. Torben, J. A. Glomsrud, T. A. Pedersen, I. B. Utne, and A. J. Sørensen, "Automatic simulation-based testing of autonomous ships using Gaussian processes and temporal logic," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 2022.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[28] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[29] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 115–140, 2019.

[30] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. of the IEEE Conference on Decision and Control*, 2014, pp. 81–87.

[31] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys*, vol. 50, no. 2, 2017.

RightsLink

Sign in/Register

Safe Reinforcement Learning with Probabilistic Guarantees Satisfying Temporal Logic Specifications in Continuous Action Spaces

Conference Proceedings: 2023 62nd IEEE Conference on Decision and Control (CDC)

Author: Hanna Krasowski

Publisher: IEEE

Date: 13 December 2023

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

B.3 Provable Traffic Rule Compliance in Safe Reinforcement Learning on the Open Sea

Summary Most provably safe RL approach only regard safety as avoiding unsafe states, i.e., Safety Specification 2. However, legal safety for autonomous vehicles is often more complex and defined by traffic rule compliance. To integrate traffic rules into verification approaches, they need to be formalized, for which temporal logic is well suited.

To this end, we propose a safe RL approach that guarantees maritime traffic rule compliance during training and deployment for motion planning of autonomous vessels on the open sea. In order to integrate traffic rules in verification approaches, we first formalize them using temporal logic. To this end, we extend and adapt the maritime traffic rules formalized via temporal logic in Appendix B.1. We design a discrete action space and develop an online verification approach to determine all rule-compliant actions given the current system state. In particular, we introduce a statechart which entails the traffic rule specification of the COLREGS and design an efficient maneuver synthesis to obtain rule-compliant actions. Further, we introduce an emergency controller that reflects the last-minute maneuver rule of the COLREGS, i.e., the controller mitigates the collision risk by performing an evasive maneuver whenever other vessels do not take appropriate collision avoidance measures. We prove that our safe RL agent only selects verified actions that comply with the formalized legal safety specification of the COLREGS. Our verification approach is built on parameterized temporal logic rules and predicates. Thus, if parameters change due to traffic situations other than open-sea situations, they can be easily adapted.

We train and test the safe RL agent on critical traffic situations, which are either handcrafted or based on recorded maritime traffic data. We compare our safe RL agent with two baseline RL agents, which are informed about traffic rule compliance through the reward function only. We observe that the baseline RL agents violate safety and collide even after training, whereas our safe RL agent never collides and always complies with the formalized maritime traffic rules. Further, our results show that RL agents trained on handcrafted data generalize well to traffic situations based on recorded maritime traffic data.

Author contributions H.K. initiated the project of safe RL with maritime traffic rules formalized via temporal logic. H.K. formalized the last-minute maneuver rule and designed the emergency controller. H.K. developed the online verification approach, implemented the RL environment for autonomous vessels, conducted the numerical experiments, and wrote the article. M.A. provided feedback advancing the verification approach and the manuscript.

TUM Graduate School This publication is *not* a core publication in accordance with Article 7, section 3 TUM Doctoral Regulations (PromO).

Provable Traffic Rule Compliance in Safe Reinforcement Learning on the Open Sea

Hanna Krasowski and Matthias Althoff

***Abstract*—For safe operation, autonomous vehicles have to obey traffic rules that are set forth in legal documents formulated in natural language. Temporal logic is a suitable concept to formalize such traffic rules. Still, temporal logic rules often result in constraints that are hard to solve using optimization-based motion planners. Reinforcement learning (RL) is a promising method to find motion plans for autonomous vehicles. However, vanilla RL algorithms are based on random exploration and do not automatically comply with traffic rules. Our approach accomplishes guaranteed rule-compliance by integrating temporal logic specifications into RL. Specifically, we consider the application of vessels on the open sea, which must adhere to the Convention on the International Regulations for Preventing Collisions at Sea (COLREGS). To efficiently synthesize rule-compliant actions, we combine predicates based on set-based prediction with a statechart representing our formalized rules and their priorities. Action masking then restricts the RL agent to this set of verified rule-compliant actions. In numerical evaluations on critical maritime traffic situations, our agent always complies with the formalized legal rules and never collides while achieving a high goal-reaching rate during training and deployment. In contrast, vanilla and traffic rule-informed RL agents frequently violate traffic rules and collide even after training.**

***Index Terms*—Safe reinforcement learning, autonomous vessels, temporal logic, provable guarantees, collision avoidance.**

I. Introduction

Reinforcement learning (RL) has provided promising results for a variety of motion planning tasks, e.g., autonomous driving [1], [2], robotic manipulation [3], [4], and autonomous vessel navigation [5]–[7]. RL algorithms learn a capable policy through random exploration. As random exploration is inherently unsafe, RL agents are mainly trained and tested in simulation only. To transfer the capabilities of RL-based motion planning systems to the physical world, the agents have to be safe. Safe RL extends RL algorithms with safety considerations. Most safe RL approaches constrain the learning softly, e.g., by integrating risk measures in the reward function or by adapting the optimization problem for obtaining a policy considering constraints [8]. However, for safety-critical tasks, such as motion planning in the physical world, hard safety guarantees are necessary, which most safe RL approaches cannot provide.

Provably safe RL achieves hard safety guarantees during training and operation by combining RL with formal methods [8]. The safety specifications regarded in provably safe RL are so far mainly *avoid specifications*, i.e., it is ensured that unsafe areas and actions are always avoided. However, the notion of safety for real-world tasks is often more complex than avoiding unsafe sets. For autonomous vehicles, legal safety is usually required, meaning that vehicles do not cause collisions by obeying traffic rules [9], [10]. To apply formal methods, these traffic rules need to be formalized. Temporal logic is suited to formalize traffic rules [9], [11]–[15], as it can capture their spatial and temporal dependencies well. Still, efficient and generalizable integration of formalized traffic rules in motion planning approaches is an open research problem.

In this work, we propose a provably safe RL approach that ensures legal safety by complying with traffic rules formalized in temporal logic for the application of autonomous vessel navigation. Fig. 1 displays the concept of our approach. We develop a statechart that reflects the formalized traffic rules and their hierarchy. Regular collision avoidance rules are followed as long as there is no immediate collision risk, and an emergency operation that executes a last-minute maneuver is immediately activated once a collision becomes likely. For the regular collision avoidance rules, an application-specific maneuver synthesis method based on a search algorithm is developed to efficiently identify actions that are compliant with traffic rules. For emergency operation, we detect imminent collision of the vessels using set-based reachability analysis and design an emergency controller that aims to prevent collisions as much as possible. Rule-compliant actions for both regular and emergency operation are computed online based on our statechart and are used to constrain the RL agent so it can only select verified actions. Our main contributions are:

- We are the first to introduce a safe RL approach that ensures provable satisfaction of open-sea maritime collision avoidance rules, which are formally specified via temporal logic;
- We improve our previously formalized maritime traffic rules [15], newly formalize the last-minute maneuver rule from the Convention on the Convention on the International Regulations for Preventing Collisions at Sea (COLREGS), and develop a rule-compliant emergency controller;
- Our provably safe maneuver synthesis for discrete action spaces efficiently identifies safe actions online;
- We train provably safe RL and safety-informed RL agents on critical maritime traffic situations and evaluate their performance in different deployment configurations on handcrafted and recorded maritime traffic data.

The remainder of this article is structured as follows: We present and discuss related literature in Sec. II, introduce

All authors are with Technical University of Munich, Germany; TUM School of Computation, Information and Technology, Department of Computer Engineering; Munich Center for Machine Learning (MCML).
{hanna.krasowski, althoff}@tum.de

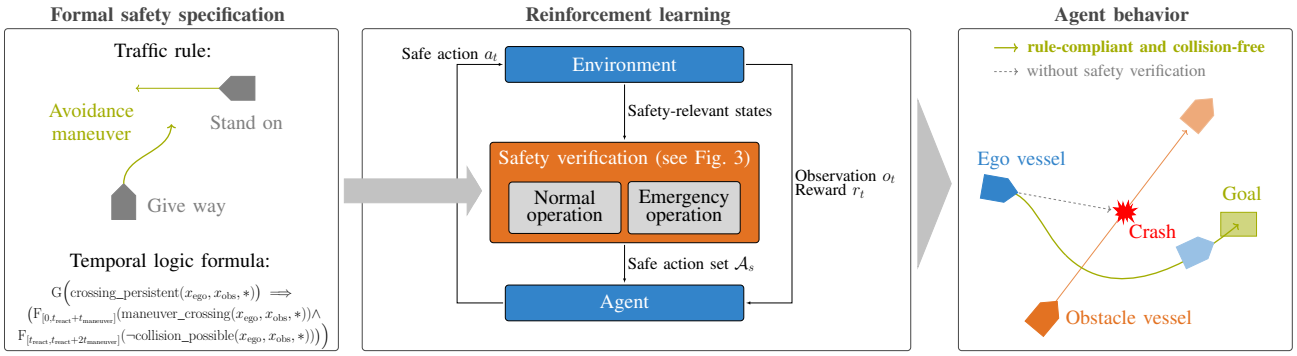


Fig. 1. Proposed provably safe RL approach for autonomous vessels. First, traffic rules for collision avoidance are formalized with temporal logic (see Sec. III). Based on the formal specification, the set of rule-compliant actions is identified (see Sec. IV and Sec. V), which are integrated in the RL process so that the agent can only select actions that are rule-compliant (see Sec. VI). Note that the statechart in Fig. 3 details the computation of verified rue-compliant actions and comprises two modes: normal operation and emergency operation. The resulting safe agent achieves rule compliance and collision avoidance during training and deployment, while agents without the safety verification of actions violate the formalized traffic rules and collide still after training (see Sec. VII).

relevant concepts published preliminarily to this article and state the problem in Sec. III. We present the formalized traffic rules and prove that a statechart models the traffic rules in Sec. IV. We describe our rule-compliant maneuver synthesis in Sec. V. The RL approach is detailed in Sec. VI. In Sec. VII, we discuss our experimental results on critical maritime traffic situations and conclude in Sec. VIII.

II. Related Work

We categorize related work into safety specifications for maritime motion planning, motion planning approaches for autonomous vessels, and provably safe RL.

a) Safety specification for maritime motion planning: The notion of safety in maritime motion planning is usually rule compliance with maritime traffic rules describing collision avoidance maneuvers [16]. The most relevant maritime traffic rules for collision avoidance are specified in the COLREGS [17]. Often, these traffic rules are indirectly integrated in the motion planning approach, e.g., through geometric thresholds [18]–[23], virtual obstacles [24], or cost functions [6], [25]–[29]. However, these approaches usually do not capture the temporal properties of collision avoidance rules, and the implemented interpretation of the COLREGS is often intransparent.

Another concept is to formalize the traffic rules and directly use them in motion planning. This is a more faithful consideration of traffic rules than the previously mentioned indirect integration. Additionally, the rule formalization is usually parameterized, which eases adaptions. Temporal logic is suited to formalize COLREGS since it captures temporal dependencies and thus can model sophisticated specifications of encounter situations. There are two relevant studies that formalize maritime traffic rules with temporal logic. Torben et al. [30] formalize COLREGS with signal temporal logic for automatic testing of autonomous vessels. This has the advantage that robustness measures specified through signal temporal logic formulas can be used as costs for motion planning approaches, since they quantify rule compliance. Krasowski et al. [15] formalize COLREGS with metric temporal logic and evaluate their compliance on real-world maritime traffic data. They discuss that the COLREGS are currently not well posed for more than two vessels, which needs to be addressed by regulators to make autonomous vessels admissible for commercial deployment in the real-world. How to best employ temporal logic formalizations for motion planning approaches as presented by [15], [30] is an open research question, for which we propose a solution in this work.

b) Motion planning for vessels: The motion planning literature can be categorized into single-agent and multi-agent motion planning problems [31], where multi-agent settings are often distinguished into cooperative [32]–[34] and non-cooperative [35], [36] settings. In this article, we regard single-agent motion planning. Maritime motion planners are often divided into three building blocks [37]: a guidance system generating reference trajectories, a control system for tracking reference trajectories, and a state observer[1]. For example, one line of single-agent motion planning research employs search-based algorithms based on motion primitives, e.g., rapidly-exploring random trees [29], [38], [39]. Other studies employ model predictive control (MPC) [26], [28], [40] to obtain an optimal control signal. In contrast to using search algorithms based on a finite amount of motion primitives, MPC directly optimizes the controller in the continuous state and input space. In particular, the studies [26], [28] show promising results on multi-obstacle scenarios and Kufoalor et al. [28] even evaluate their approach in real-world experiments with two obstacle vessels. However, for MPC an optimization problem must be solved repeatedly, which can be computationally costly.

RL is a well-suited machine learning approach to solve single-agent motion planning tasks in uncertain environments [6], [7], [41]–[44]. Regarded scenarios are usually on the open sea with other non-reactive dynamic obstacles [7], [42], [43] and static obstacles [6], [41], [44]. To achieve a behavior that adheres to maritime traffic rules, the reward function considers rule compliance to minimize risks, but does not guarantee compliance because the reward function is only maximized

[1]Often referred to as a navigation system in the maritime literature.

[6], [7], [41]–[44]. In contrast, provably safe RL approaches ensure safety [8].

c) Provably safe RL: Provably safe RL approaches ensure safety during training and operations. There are three conceptual approaches for provably safe RL [8]: action replacement, action projection, and action masking. In this article, we present an action masking approach, for which the agent can only choose actions that are verified as safe. Most research on action masking considers discrete action spaces; common applications are autonomous driving [45]–[50] and power systems [51]. Usually, the action verification is tailored to the specific application and, thus, cannot be directly transferred to other applications.

Another way to distinguish provably safe RL approaches is by the safety specification. Most approaches consider safety specifications that can be formalized as containment in a safe set or avoiding intersection with unsafe sets. A few works regard safety specifications based on temporal logic [52]–[54], which can additionally model temporal dependencies in safety specifications. The studies [52], [53] use model checking to determine whether a given action fulfills a linear temporal logic formula, which expresses the safety specification. Their approaches are transferable between applications but limited to discrete action and state spaces. In contrast, Li et al. [54] leverage linear temporal logic specifications to synthesize control barrier functions, which are used to project unsafe actions proposed by the agent to safe actions. This allows them to apply their approach to continuous action and state spaces. However, their approach cannot deal with dynamic obstacles that are not controllable, such as other traffic participants. To the best of our knowledge, we are the first to formulate a provably safe RL approach for the application of autonomous vessels and to include temporal safety specifications in the online safety verification of RL agents while operating in a continuous state space.

III. Preliminaries and Problem Statement

a) Notation and dynamics: We denote sets by calligraphic letters, vectors are boldfaced, and predicates are written in Roman typestyle. The Minkowski sum is defined as $\mathcal{Y}_1 \oplus \mathcal{Y}_2 = \{y_1 + y_2 \mid y_1 \in \mathcal{Y}_1, y_2 \in \mathcal{Y}_2\}$ and the set-based multiplication is defined as $\mathcal{Y}_1\mathcal{Y}_2 = \{y_1 y_2 | y_1 \in \mathcal{Y}_1, y_2 \in \mathcal{Y}_2\}$. A traffic rulebook $\langle \Phi, \leq \rangle$ is a tuple where Φ is the set of formalized rules and $\leq$ is the order [55]. We denote that the model Ξ and its initial state ξ entail the rulebook $\langle \Phi, \leq \rangle$ by $\Xi, \xi \models \langle \Phi, \leq \rangle$.

The state of a vessel $\mathbf{s} \in \mathbb{R}^4$ consists of the position $\mathbf{p} = [p_\mathrm{x}, p_\mathrm{y}] \in \mathbb{R}^2$ in the Cartesian coordinate frame as well as the orientation $\theta \in \mathbb{R}$, and the orientation-aligned velocity $v \in \mathbb{R}$. The operator $\mathtt{proj}_\square$ projects a state to the state dimensions indicated by $\square$ and $\mathrm{R}(\Upsilon) = \{\mathtt{R}(\upsilon) | \upsilon \in \Upsilon\}$ denotes the set of rotation matrices for the angles Υ with $\mathtt{R}(\upsilon)$ being the rotation matrix for the angle υ. To model the ego vessel (i.e., the autonomous vessel we control), we use a yaw-constrained model Ω_yc with orientation-aligned acceleration $\mathrm{a} \in \mathbb{R}$ and turning rate $\omega \in \mathbb{R}$ as control inputs:

$$\dot{\mathbf{s}} = \begin{bmatrix} \dot{p_\mathrm{x}} \\ \dot{p_\mathrm{y}} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \cos(\theta)\, v \\ \sin(\theta)\, v \\ \omega \\ \mathrm{a} \end{bmatrix}. \tag{1}$$

The control input is denoted as $\mathbf{u}(t) = [\mathrm{a}(t), \omega(t)]$ and the initial state as $\mathbf{s}_0$.

b) Set-based prediction of vessels: To obtain predictions that enclose all possible behaviors of a traffic participant, the concept of set-based predictions for road traffic participants [56] can be transferred to maritime traffic. The fundamental idea is to define abstract models and perform reachability analysis for them. We first specify the dynamics used for the prediction, and then introduce the reachable sets and occupancy sets. Finally, we discuss the special case of a closed-loop system.

For vessels, we assume that the abstract model is a point-mass model Ω_pm with velocity and acceleration constraints:

$$\begin{aligned} &\dot{p_\mathrm{x}}(t) = v_\mathrm{x}(t),\, \dot{p_\mathrm{y}}(t) = v_\mathrm{y}(t), \\ &\dot{v_\mathrm{x}}(t) = \mathrm{a}_\mathrm{x}(t),\, \dot{v_\mathrm{y}}(t) = \mathrm{a}_\mathrm{y}(t), \\ &\text{subject to} \quad \sqrt{v_\mathrm{x}(t)^2 + v_\mathrm{y}(t)^2} \leq v_\mathrm{pm,max} \\ &\qquad\qquad \sqrt{\mathrm{a}_\mathrm{x}(t)^2 + \mathrm{a}_\mathrm{y}(t)^2} \leq \mathrm{a}_\mathrm{pm,max}\,. \end{aligned} \tag{2}$$

The maximum velocity and maximum acceleration are denoted by $\mathrm{a}_\mathrm{pm,max}$ and $v_\mathrm{pm,max}$, respectively. To ensure formal safety of our approach, the two constraints must be chosen such that the point-mass model over-approximates the behavior of vessels using reachset conformance [57]. The state of the model Ω_pm is abbreviated by $\mathbf{x} = [p_\mathrm{x}, p_\mathrm{y}, v_\mathrm{x}, v_\mathrm{y}]$.

The time-point reachable sets for the model Ω_pm are calculated with set-based reachability analysis [58] based on the initial state $\mathbf{s}_0$, time step size Δt, and the time horizon t_pred. Note that the state $\mathbf{s}_0$ is transformed into $\mathbf{x}_0$ by using trigonometry to convert $[v, \theta]$ into $[v_\mathrm{x}, v_\mathrm{y}]$. The time-interval reachable sets are computed as in [58], [59] and are denoted by $\mathcal{R}_{\Delta t}(\mathbf{s}_0, \Omega_\mathrm{pm}, t_\mathrm{pred})$. To obtain the occupancy sets from the time-interval reachable sets, the reachable sets are projected to the position domain and enlarged by the spatial extensions of the vessel $\mathcal{V}$ rotated by all possible reachable orientations using the Minkowski sum:

$$\begin{aligned} &\mathcal{O}_\mathrm{pm}(\mathbf{s}_0, \Omega_\mathrm{pm}, t_\mathrm{pred}, \mathcal{V}) = \\ &\quad \mathtt{proj}_\mathbf{p}\left(\mathcal{R}_{\Delta t}(\mathbf{s}_0, \Omega_\mathrm{pm}, t_\mathrm{pred})\right) \oplus \\ &\quad \Big(\mathrm{R}\left(\mathtt{proj}_\theta\left(\mathcal{R}_{\Delta t}(\mathbf{s}_0, \Omega_\mathrm{pm}, t_\mathrm{pred})\right)\right)\, \mathcal{V}\Big). \end{aligned} \tag{3}$$

For a detailed derivation of the occupancy sets, we refer the interested reader to [56, Sec. V-A].

The occupancy sets $\mathcal{O}_\mathrm{pm}$ are calculated for the open-loop system Ω_pm since we do not have access to the control input of other traffic participants. However, for an ego vessel, we have a precise model Ω_yc and access to the control input. Thus, the forward simulation of our closed-loop system with the control input $\mathbf{u}(t)$ provides the time-point reachable sets. The occupancy is denoted by:

$$\mathcal{O}_\mathrm{traj}(\mathbf{s}_0, \Omega_\mathrm{yc}, t_\mathrm{pred}, \mathcal{V}, \mathbf{u}(t)). \tag{4}$$

c) Problem statement: The COLREGS specify the traffic rules for collision avoidance on the open sea for power-driven vessels in natural language. These traffic rules are satisfiable for two vessels. For more than two vessels, unsatisfiable traffic situations can occur, e.g., a vessel needs to keep its course and speed with respect to one vessel and perform an avoidance maneuver with respect to another vessel. The COLREGS do not specify how to adequately resolve such conflicting situations with more than two vessels. Due to the lack of legal specifications, we regard traffic situations with two vessels only. In particular, we assume:

1) The traffic situation is an open-sea situation without traffic signs, traffic separation zones, or static obstacles;
2) There is one traffic participant vessel obs and one autonomous vessel ego, which are both power-driven;
3) The dynamics of the autonomous vessel is modeled by (1);
4) The current state of the traffic participant vessel $\mathbf{s}_{\text{obs}}$ is observed without measurement errors;
5) In the initial state of the traffic situation, none of the collision avoidance rules specified in the COLREGS apply.

We define the traffic rulebook $\langle \Phi, \leq \rangle$ that describes the legally relevant collision avoidance rules of the COLREGS given our assumptions 1) and 2). The formal traffic rules are denoted by Φ and the hierarchy between them by $\leq$. Based on the traffic rules, we search for an RL approach, which ensures that the RL agent only selects safe, i.e., rule-compliant, actions leading to rule-compliant trajectories. Thus, the overall problem is to find

$$\begin{aligned} &\pi_s \colon \mathcal{S} \to \mathcal{A}_s \\ &\text{where} \quad \zeta_{\pi_s} \models \langle \Phi, \leq \rangle . \end{aligned} \tag{5}$$

The observation space of the RL agent is $\mathcal{S}$, the set of provably rule-compliant actions is $\mathcal{A}_s$, and the trajectories ζ_{π_s} are solutions of (1) when following the RL policy π_s. To address this problem, we first introduce the rulebook $\langle \Phi, \leq \rangle$, and prove that a statechart Γ entails the rulebook in Sec. IV. Then, we describe the synthesis of rule-compliant maneuvers and detail the safe-by-design action selection in Sec. V. Finally, we describe the RL specification in Sec. VI.

IV. Specification

Our previous work [15] formalizes the COLREGS rules specifying collision avoidance between two power-driven vessels on the open sea. The temporal operators used are G, F, and U, and if there is a subscript, the temporal operator is evaluated over the time interval indicated by the subscript. The operator $\mathrm{G}(\phi)$ evaluates to true iff ϕ is true for all future time steps. In contrast, for the operator $\mathrm{F}(\phi)$, ϕ only has to be true for at least one future time step. The until operator $\phi_1 \mathrm{U} \phi_2$ is true iff ϕ_1 holds true for all time steps until ϕ_2 holds true. In this section, we introduce the legal specification through a rulebook and detail the novel formalization of the emergency rule. Finally, we introduce the statechart Γ and show that it models the specification.

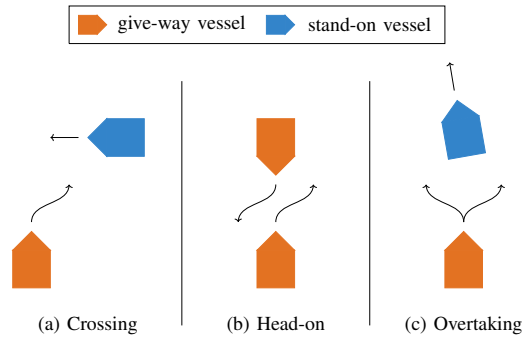


Fig. 2. Encounter situations and rule-compliant maneuvers specified in the COLREGS (adapted from [15]).

A. Traffic Rulebook

Table I lists all formalized rules considered in this work. While the predicates can be evaluated on any two vessels, the predicate arguments are set to be evaluated for the ego vessel with respect to an obstacle vessel according to the COLREGS. The traffic rule R_2 enforces a safe speed, which is trivially ensured through the ego vessel dynamics. Thus, we do not include this rule in the traffic rulebook.

Definition 1 (Rules Φ)**.** The rulebook consist of rules R_1 and $R_3 - R_6$ specified in Table I.

We introduce the emergency rule R_1 to reflect the COLREGS specification that if the other vessel does not take appropriate actions for collision avoidance, the ego vessel has to react and perform a last-minute maneuver for collision risk minimization.

COLREGS Requirement 1 (Rulebook order $\leq$)**.** Rule R_1 is always prioritized over rules R_3 - R_5, and R_6 has the lowest priority. Rules R_3 - R_5 are all of equal priority.

The predicates of rule R_1 are detailed in Sec. IV-B. Note that we use the emergency maneuver to describe the last-minute maneuver, through which the ego vessel minimizes the collision risk and thereby achieves legal safety. Yet, in the literature, the term failsafe planning is also frequently used [8], [60].

Rules R_3 - R_6 describe how vessels have to behave in a COLREGS encounter situation. In these encounter situations, the vessels are on a collision course meaning that the vessels would collide in the near future if no appropriate collision avoidance measures are taken. There are three different encounter situations specified in the COLREGS as illustrated in Fig. 2: overtaking (R_5, R_6), crossing (R_3, R_6), and head-on encounters (R_4). In an encounter, a vessel can be a give-way or a stand-on vessel. A give-way vessel is required to change course and perform a collision avoidance maneuver. A stand-on vessel has the obligation to keep its course and speed. The predicate for determining a stand-on vessel is keep (see Appendix-A). The stand-on rule R_6 has the lowest priority since whenever the other vessel changes its course so that the ego vessel becomes the give-way vessel, the give-way rules R_3 to R_5 are applied (see COLREGS Requirement 1).

TABLE I
OVERVIEW FORMALIZED MARINE TRAFFIC RULES INTEGRATED IN THE SAFETY VERIFICATION

Rule	Temporal logic formula
$R_1^{\ddagger}$	$\mathrm{G}(\text{is\_emergency}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *) \implies (\text{emergency\_maneuver U is\_emergency\_resolved}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *)))$
R_2	$\mathrm{G}\Big(\text{safe\_speed}(\mathbf{s}_{ego}, v_{\max})\Big)$
$R_3^{\dagger}$	$\mathrm{G}\Big(\text{persistent\_crossing}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *) \implies (\mathrm{F}_{[0, t_{\text{react}}+t_{\text{maneuver}}]}(\text{maneuver\_crossing}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *)) \wedge \mathrm{F}_{[t_{\text{react}}, t_{\text{react}}+2t_{\text{maneuver}}]}(\neg\text{collision\_possible}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, t_{\text{horizon}}^{\text{check}})))\Big)$
$R_4^{\dagger}$	$\mathrm{G}\Big(\text{persistent\_head\_on}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *) \implies (\mathrm{F}_{[0, t_{\text{react}}+t_{\text{maneuver}}]}(\text{maneuver\_head\_on}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *)) \wedge \mathrm{F}_{[t_{\text{react}}, t_{\text{react}}+2t_{\text{maneuver}}]}(\neg\text{collision\_possible}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, t_{\text{horizon}}^{\text{check}})))\Big)$
$R_5^{\dagger}$	$\mathrm{G}\Big(\text{persistent\_overtake}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *) \implies (\mathrm{F}_{[0, t_{\text{react}}+t_{\text{maneuver}}]}(\text{maneuver\_overtake}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *)) \wedge \mathrm{F}_{[t_{\text{react}}, t_{\text{react}}+2t_{\text{maneuver}}]}(\neg\text{collision\_possible}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, t_{\text{horizon}}^{\text{check}})))\Big)$
R_6	$\mathrm{G}\Big(\text{keep}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *) \implies (\text{no\_turning}(\mathbf{s}_{\text{ego}}, *)\mathrm{U}\,\neg\text{keep}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *))\Big)$

Note: Additional arguments are abbreviated by $*$, rules adapted from [15] are marked with †, and new rules are marked with ‡.

To formalize that a give-way encounter is persistent for at least the reaction time, we use the following temporal logic specification, where $\{\text{give\_way}\}$ can take the values from $\{\text{crossing, head\_on, overtake}\}$ (see Appendix-A) and $*$ denotes additional arguments for the predicates:

$$\begin{aligned}&\text{persistent\_}\{\text{give\_way}\}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *) = \\ &\quad \neg\{\text{give\_way}\}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *) \wedge \\ &\quad \mathrm{G}_{[\Delta \mathrm{t}, \mathrm{t}_{\text{react}}]}(\{\text{give\_way}\}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *)).\end{aligned}$$

We assume that both vessels keep their course and speed to obtain rule-compliant predictions for their future states. These predicted states allow us to evaluate ahead of time if the encounter situation will persist long enough so that the ego vessel has to perform a collision avoidance maneuver. The reaction time t_{react} does not indicate the minimum required reaction time of a human operator but instead specifies how much time the human operator would require to decide if the encounter situation persists. Given a give-way encounter is detected, a rule-compliant collision avoidance maneuver has to be conducted until $\neg$collision_possible evaluates to true (see Table I R_3 - R_5). The time interval for performing a rule-compliant maneuver is $t_{\text{react}} + 2t_{\text{maneuver}}$, where $2t_{\text{maneuver}}$ approximates the time required for the maneuvering.

COLREGS Requirement 2 (Maneuvering priority)**.** Given a rule R_i for $i \in \{3, ..., 5\}$ applies, rules R_j for $j \neq i \wedge j \in \{3, ..., 5\}$ are not applied until $\neg$collision_possible is true.

B. Emergency Rule Predicates

We use the predicate collision_possible to determine if two vessels are on a collision course for rules R_3 - R_6. Because the rules R_3 - R_6 assume a constant velocity, we use the velocity obstacle concept [61] for this predicate. However, the velocity obstacle concept is not sufficient for detecting imminent risk as necessary for R_1. Thus, we present four predicates in this section that are relevant for our formalization of rule R_1.

First, we define an auxiliary position predicate determining if vessel m is in a relative orientation sector of vessel l:

$$\begin{aligned}&\text{in\_sector}(\mathbf{s}_l, \mathbf{s}_m, \underline{\beta}, \overline{\beta}) \iff \\ &\quad \mathbf{h}_{\underline{\beta}}^T \mathtt{proj}_{\mathbf{p}}(\mathbf{s}_m) - b_{l,\underline{\beta}} \leq 0 \wedge \\ &\quad \mathbf{h}_{\overline{\beta}}^T \mathtt{proj}_{\mathbf{p}}(\mathbf{s}_m) - b_{l,\overline{\beta}} > 0,\end{aligned}$$

where the lower relative orientation is $\underline{\beta}$ and the upper relative orientation is $\overline{\beta}$ relative to the orientation of vessel l. The normal vector $\mathbf{h}_i$ is the unit vector in the direction $i - \pi/2$ and $b_{l,i}$ is the offset to the origin for a line through the position of vessel l in the direction i. We illustrate the sector predicate with two specific usages in Fig. 4.

Second, we use set-based prediction for rule R_1 to detect potential collisions in the near future. In particular, we predict the future occupancy of the obstacle vessel until the time horizon t_{pred} as described in (3) and that of the ego vessel as in (4), for the control sequence $\mathbf{u}_{\text{keep}}(t) = [0\,\mathrm{m\,s^{-2}}, 0\,\mathrm{rad\,s^{-1}}]$ to keep course and speed as demanded for stand-on vessels. If the ego occupancy and the predicted occupancy of the obstacle vessel intersect, the ego vessel is in an emergency situation:

$$\begin{aligned}&\text{is\_emergency}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, \mathcal{V}_{\text{ego}}, \mathcal{V}_{\text{obs}}, t_{\text{pred}}, \mathbf{u}_{\text{keep}}(t)) \iff \\ &\quad \exists t \in [t_0, t_0 + t_{\text{pred}}] : \mathcal{O}_{\text{pm}}(\mathbf{s}_0, \Omega_{\text{pm}}, t, \mathcal{V}_{\text{obs}}) \cap \\ &\quad \mathcal{O}_{\text{traj}}(\mathbf{s}_{\text{ego}}, \Omega_{\text{yc}}, t, \mathcal{V}_{\text{ego}}, \mathbf{u}_{\text{keep}}(t)) \neq \emptyset,\end{aligned}$$

where t_0 is the current time.

Third, the predicate emergency_maneuver describes a maneuver that minimizes the risk of collision for the specific traffic situation. We detail our interpretation of emergency_maneuver in Sec. V-A.

Fourth, an emergency situation is resolved when the obstacle vessel is behind the ego vessel, is moving away from the ego vessel, and the Euclidean distance between both is larger than

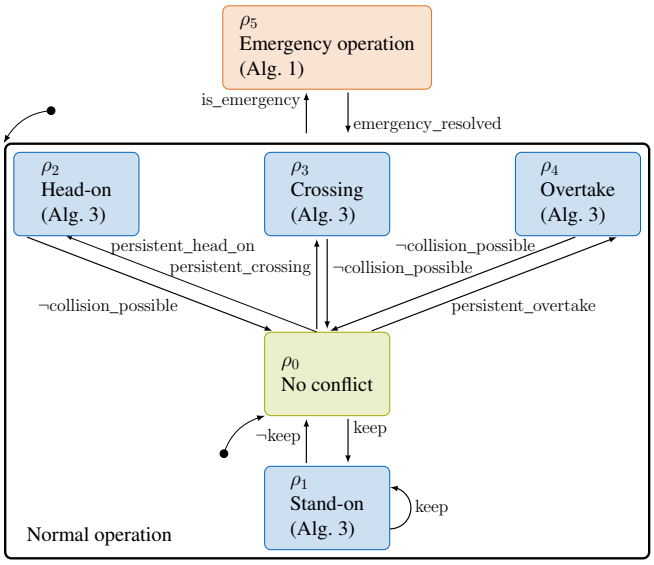


Fig. 3. Statechart Γ modeling the legal safety specification with predicates at the transitions. The states for the regular collision avoidance rules R_3 - R_6 are depicted in blue and the emergency operation state for rule R_1 in red. For safety verification of actions, the algorithms identifying the set of rule-compliant actions (indicated in brackets) are employed given the current state ρ_i of the statechart.

a specified minimum distance d_{resolved}:

$$\begin{aligned}&\text{is\_emergency\_resolved}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, d_{\text{resolved}}) \iff \\ &\underbrace{\text{in\_sector}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, 3\pi/2, \pi/2)}_{\textbf{obstacle is behind}} \wedge \\ &\underbrace{\texttt{unit\_v}(\mathbf{s}_{\text{obs}})^T\, \texttt{unit\_v}(\mathbf{s}_{\text{ego}}) \leq 0}_{\textbf{moving away}} \wedge \\ &\underbrace{\|\texttt{proj}_{\mathbf{p}}(\mathbf{s}_{\text{obs}}) - \texttt{proj}_{\mathbf{p}}(\mathbf{s}_{\text{ego}})\|_2 \leq d_{\text{resolved}}}_{\textbf{distance between vessels is large enough}}\end{aligned}$$

where the unit orientation vector of a state is $\texttt{unit\_v}(\mathbf{s}) = [\cos(\texttt{proj}_\theta(\mathbf{s})), \sin(\texttt{proj}_\theta(\mathbf{s}))]$.

C. Specification-compliant Statechart

The overall rule specification is modeled by the statechart Γ in Fig. 3. Due to assumption 5), the initial state in every traffic situation is the state ρ_0. There are two main states for normal operation and emergency operation. During normal operation, whenever the predicate collision_possible is true, the corresponding maneuver state for R_3 - R_6 (see blue states in Fig. 3) is entered and the collision avoidance maneuver is started.

Proposition 1. *For the states ρ_i, $\forall i \in \{1, ..., 4\}$, the predicate* collision_possible *is true.*

Proof: This follows directly from the definition of the predicates keep, head_on, crossing, and overtake (see Appendix-A), which are true for the states of the statechart $\rho_1 - \rho_4$, respectively. ■

Lemma 1. *For two specific vessels, at most one of the predicates* keep, head_on, crossing, *or* overtake *can be true at the same time.*

Proof: The predicates keep, head_on, crossing, and overtake cannot apply at the same time due to their mutually exclusive specification. The detailed proof is in Appendix-B. ■

If an emergency situation is detected, the statechart transitions to the emergency operation state until the emergency situation is resolved.

Theorem 1. *It holds that $\Gamma, \rho_0 \models \langle \Phi, \leq \rangle$ for the statechart Γ, its initial state ρ_0, and the rulebook $\langle \Phi, \leq \rangle$.*

Proof: The initial state ρ_0 fulfills the rulebook by assumption 5) (see Sec. III.c). We continue proving the compliance with each rule:

(I) R_1: If is_emergency is true, R_1 applies and R_3 - R_6 do not (see COLREGS Requirement 1), which is realized by transitioning to ρ_5 (see Fig. 3). The state ρ_5 can only be exited iff is_emergency_resolved evaluates to true. Thus, the transition to and from ρ_5 directly represents R_1.

If collision_possible $\wedge$ ¬is_emergency is true, then Γ has to represent rules R_3 - R_6. Whenever collision_possible becomes true, it can be deduced from Lemma 1 and Proposition 1 that the statechart transitions to a state ρ_i, $i \in \{1, ..., 4\}$.

(II) R_3 - R_5: Based on COLREGS Requirement 2, once a rule R_3 - R_5 applies, i.e., the statechart is in either of the states ρ_i, $i \in \{2, ..., 4\}$, the respective avoidance maneuver has to be conducted until ¬collision_possible $\vee$ is_emergency is true. For is_emergency, we showed in case (I) of this proof that Γ models $\langle \Phi, \leq \rangle$. For ¬collision_possible, the statechart Γ transitions to ρ_0.

(III) R_6: Once rule R_6 applies, i.e., keep is true, the statechart transitions to ρ_1 and stays there until ¬keep $\vee$ ¬collision_possible $\vee$ is_emergency. If ¬keep $\wedge$ collision_possible, an encounter of higher priority is present (see COLREGS Requirement 1) and R_3 - R_5 apply. In this situation, the statechart transitions to the states ρ_i for $i \in \{2, ..., 4\}$ and the remaining proof steps are stated in case (III). Identically to case (III), if ¬collision_possible is true, the statechart Γ transitions to ρ_0 and if is_emergency the statechart transitions to ρ_5. ■

V. Rule-Compliant Maneuver Synthesis

Given our specification-compliant statechart Γ, we need to identify rule-compliant actions for the individual states ρ_i of the statechart. Trivially, for the state ρ_0 all actions are rule-compliant since no rules apply. We introduce the synthesis of emergency maneuvers in Sec. V-A and of encounter maneuvers in Sec. V-B. Finally, we detail how we ensure a selection of only safe actions for the RL agent in Sec. V-C.

A. Emergency Maneuver

Once we detect an emergency situation, i.e., the statechart is in ρ_5, the ego vessel is legally required to evade the obstacle vessel in a manner that minimizes the risk of collision. In similar motion planning applications, such as autonomous driving [10], autonomous aerial traffic [62], or human-robot environments [63], states that are safe for infinite

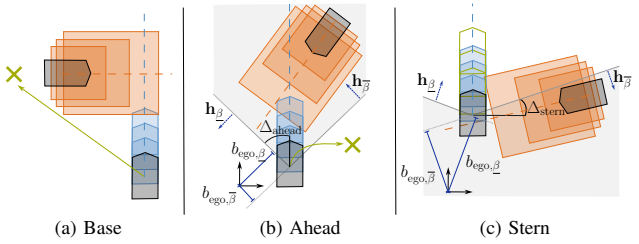


Fig. 4. Emergency controller modes with set-based occupancy prediction of obstacle vessel in orange and the occupancy of the ego vessel in blue for several time intervals. The orientation of the ego vessel and the obstacle vessel are indicated with dashed lines and emergency maneuver is depicted by green arrows or occupancies. The green cross indicates the target position for the base and ahead modes. The sectors, for which the predicate in_sector is true, are shown in gray for the ahead and stern mode. The visualization of the sectors includes the arguments of predicate in_sector in dark blue and the point of origin in black.

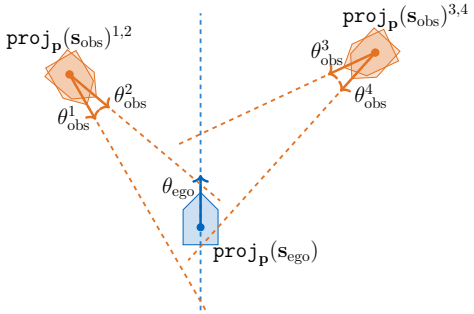


Fig. 5. Visualization of turning direction cases. The obstacle vessel is depicted in orange and the ego vessel in blue. Arrows indicate orientations and positions are marked with dots. The turning direction case is indicated by the superscript. For cases 1 and 3, the ego vessel should turn right and for the cases 2 and 4, the ego vessel should turn left.

time are used to identify a legally safe emergency maneuver. In contrast, the current COLREGS do not state specifically how to interpret "minimize risk" or the characteristics of an invariably safe state. Thus, we cannot provide a formal specification. Consequently, we cannot verify risk minimizing behavior. Nevertheless, we identify three situations in which different emergency maneuvers are appropriate: base mode, ahead mode, and stern mode (see Fig. 4).

In the ahead case (see Fig. 4b), the obstacle vessel is in the ahead sector in front of the ego vessel, and the orientation difference between the ego vessel orientation and the reversed orientation of the obstacle vessel is at most Δ_{ahead}. This can be formalized as:

$$\begin{aligned}&\text{ahead\_emergency}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, \Delta_{\text{ahead}}) \iff \\ &\quad \texttt{in}(\rho_5) \wedge \neg\text{orientation\_delta}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, \Delta_{\text{ahead}}, \pi) \wedge \\ &\quad \text{in\_sector}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, -\Delta_{\text{ahead}}, \Delta_{\text{ahead}}),\end{aligned} \tag{6}$$

where the predicate $\texttt{in}(\rho_5)$ evaluates to true if and only if the statechart Γ is in base state ρ_5. In this ahead situation, steering to the stern of the obstacle vessel would lead to an even more critical situation, as both vessels would encounter each other head-on, given the obstacle vessel approximately keeps its speed and course. Thus, we instead require the ego vessel to turn $90°$. The direction of turning is determined as presented in Fig. 5. Depending on the situation, turning $90°$ can be enough to resolve the emergency situations. Yet, if the emergency is not resolved and the traveled distance of the ego vessel from the start of the maneuver is larger than $d_{\text{min,ahead}}$, the emergency controller switches to the base mode (see Fig. 4a) and steers the ego vessel behind the stern of the obstacle vessel.

The stern case is necessary for situations where the obstacle vessel is almost astern of the ego vessel and still relatively far away (see Fig. 4c):

$$\begin{aligned}&\text{stern\_emergency}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, \Delta_{\text{stern}}, \mathbf{u}_{\text{acc}}(t), \mathcal{V}_{\text{ego}}, \mathcal{V}_{\text{obs}}, \\ &\qquad t_{\text{pred}}) \iff \\ &\quad \texttt{in}(\rho_5) \wedge \\ &\quad \text{in\_sector}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, 3\pi/2 + \Delta_{\text{stern}}, \pi/2 + \Delta_{\text{stern}}) \wedge \\ &\quad \neg\text{is\_emergency}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, \mathcal{V}_{\text{ego}}, \mathcal{V}_{\text{obs}}, t_{\text{pred}}, \mathbf{u}_{\text{acc}}(t)),\end{aligned} \tag{7}$$

with the control sequence $\mathbf{u}_{\text{acc}}(t) = [\text{a}_{\text{stern}}, 0\,\text{rad}\,\text{s}^{-1}]$, $\forall t \leq t_{\text{react}}$ and then $0\,\text{m}\,\text{s}^{-2}, 0\,\text{rad}\,\text{s}^{-1}]$, $\forall t \leq t_{\text{react}} < t \leq t_{\text{pred}}$. By using the set-based prediction within this predicate, we ensure that we only use this controller mode if it is certain that accelerating would resolve the situation. In such a situation, performing an emergency maneuver that navigates the ego vessel to the stern of the obstacle vessel would be an unnecessarily long detour, given that a short acceleration period would also resolve the emergency situation.

For the base case (see Fig. 4a), the emergency situation can be safely resolved by steering to a position behind the stern of the obstacle vessel. The base emergency situation is formalized by:

$$\begin{aligned}&\text{base\_emergency} \iff \\ &\quad \texttt{in}(\rho_5) \wedge \neg\text{ahead\_emergency} \wedge \neg\text{stern\_emergency} \wedge \\ &\quad \neg\text{is\_emergency\_resolved}.\end{aligned}$$

Alg. 1 summarizes the control mode selection when entering the emergency operation state (see Fig. 3) and is an instantiation of the predicate emergency_maneuver of rule R_1 in Table I for our problem statement. For base and ahead modes, the target positions are depicted in Fig. 4 and obtained with the functions `get_target_ahead` and `get_target_base`, respectively. Given the target position, a reachable desired position given the current state is identified and a control input toward this desired position is generated (for details on the controller design see Appendix-C). The controller is abbreviated by the function `tracking_controller`.

B. Encounter Maneuvers

Given a persistent give-way encounter is detected (i.e., the statechart in Fig. 3 transitions to one of the respective blue

Algorithm 1 emergency_maneuver$(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, *)$

Input: current state of ego vessel $\mathbf{s}_{\text{ego}}$, current state of obstacle vessel $\mathbf{s}_{\text{obs}}$, emergency mode $mode$, initial time t_0, time step size Δt, acceleration control sequence $\mathbf{u}_{\text{acc}}(t)$

Output: control input $\mathbf{u}(t_i)$

1: $\mathbf{s}_{\text{ego},0} = \texttt{proj}_{\mathbf{p}}(\mathbf{s}_{\text{ego}}), \mathbf{s}_{\text{obs},0} = \texttt{proj}_{\mathbf{p}}(\mathbf{s}_{\text{obs}}), t_i = t_0$
2: **while** ¬emergency_resolved **do**
3: **if** $\|\texttt{proj}_{\mathbf{p}}(\mathbf{s}_{\text{ego},0}) - \texttt{proj}_{\mathbf{p}}(\mathbf{s}_{\text{ego}})\|_2 > d_{\text{min,ahead}} \wedge mode = \text{ahead}$ **then**
4: $mode \leftarrow$ base
5: **end if**
6: **if** $mode =$ stern **then**
7: $\text{a}, \omega \leftarrow \mathbf{u}_{\text{acc}}(t_i)$
8: **else if** $mode =$ ahead **then**
9: $\mathbf{p}_{\text{target}} \leftarrow \texttt{get\_target\_ahead}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{ego},0}, \mathbf{s}_{\text{obs},0})$
10: $\text{a}, \omega \leftarrow \texttt{tracking\_controller}(\mathbf{s}_{\text{ego}}, \mathbf{p}_{\text{target}})$
11: **else**
12: $\mathbf{p}_{\text{target}} \leftarrow \texttt{get\_target\_base}(\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}})$
13: $\text{a}, \omega \leftarrow \texttt{tracking\_controller}(\mathbf{s}_{\text{ego}}, \mathbf{p}_{\text{target}})$
14: **end if**
15: **return** $\mathbf{u}(t_i) = [\text{a}, \omega]$
16: $\mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}} \leftarrow \texttt{step\_environment}(\text{a}, \omega)$
17: $t_i \leftarrow t_i + \Delta t$
18: **end while**

states $\rho_1, \ldots, \rho_4$), we identify safe actions that result in safe maneuvers resolving the encounter.

Set-based predictions are well suited to verify that no collisions can occur if not all vessels comply with the regular collision avoidance rules R_3 - R_6. Still, for the regular collision avoidance rules, the implicit assumption in the COLREGS is that both vessels comply with them. Thus, for identifying actions of the ego vessel that are rule-compliant with these rules, we can use a rule-compliant prediction for the obstacle vessel. For the three encounter situations specified (see Fig. 2), we differentiate between the ego vessel being the give-way (R_3 - R_5 apply) and stand-on vessel (R_6 applies). First, we detail the verification of actions given the ego vessel is the stand-on vessel, i.e., $\texttt{in}(\rho_1)$. Then, we describe the more intricate synthesis given that the ego vessel is the give-way vessel (ρ_i where $i \in \{2, ..., 4\}$), and finally, summarize our encounter action synthesis.

a) Stand-on maneuver synthesis for ρ_1: The trivial action for the predicate keep is $a_{\text{keep}} = [\text{a} = 0\,\text{m}\,\text{s}^{-2}, \omega = 0\,\text{rad}\,\text{s}^{-1}]$, i.e., keeping course and speed. Note that for this trivial action there is no explicit maneuver time and the action space needs to be restricted to this action until the ego vessel is not the stand-on vessel anymore or an emergency is detected (see Fig. 3).

b) Give-way maneuver synthesis for $\rho_2 - \rho_4$: For all give-way maneuvers, a significant change of orientation (i.e., $\Delta_{\text{large\_turn}}$) is required so that other traffic participants can identify give-way maneuvers (see Fig. 2). For head-on and crossing encounters, the give-way vessel is always obliged to turn toward the right. For the overtake encounter, the suited turning direction depends on the orientation of the obstacle vessel, but this is not further specified in the COLREGS. For our maneuver synthesis, the turning direction is to the left if the orientation of the obstacle vessel is more to the right than the orientation of the ego vessel, and otherwise turning direction is to the right.

Given the turning direction, we identify candidate actions, construct maneuvers based on them, and verify if a maneuver complies with the rules. Candidate actions lead to trajectories that already fulfill the minimal turning requirement within the maneuver segment time t_{m}. A maneuver is verified if the predicate collision_possible is false at the end of the maneuver and the occupancies of both vessels do not intersect during the maneuver:

$$\begin{aligned} &\text{maneuver\_verified}(\mathbf{u}_{\text{m}}(t), \mathbf{s}_{\text{ego}}, \mathbf{s}_{\text{obs}}, t_{\text{horizon}}^{\text{check}}, \mathbf{u}_{\text{keep}}(t), \\ &\qquad \mathcal{V}_{\text{ego}}, \mathcal{V}_{\text{obs}+}) \iff \\ &\neg\text{collision\_possible}(\mathbf{s}_{\text{ego},t_{\text{end}}}, \mathbf{s}_{\text{obs},t_{\text{end}}}, t_{\text{horizon}}^{\text{check}}) \wedge \\ &\quad \forall t \in [t_0, t_0 + t_{\text{end}}] : \mathcal{O}_{\text{traj}}(\mathbf{s}_{\text{ego}}, \Omega_{\text{yc}}, t, \mathcal{V}_{\text{ego}}, \mathbf{u}_{\text{m}}(t)) \cap \\ &\quad \mathcal{O}_{\text{traj}}(\mathbf{s}_{\text{obs}}, \Omega_{\text{yc}}, t, \mathcal{V}_{\text{obs}+}, \mathbf{u}_{\text{keep}}(t)) = \emptyset, \end{aligned} \tag{8}$$

where t_0 is the current time, $t_{\text{end}} \in n\, t_{\text{m}}$ is the time horizon of the maneuver with $n \in \mathbb{N}^+$, $\mathbf{s}_{\text{ego},t_{\text{end}}}$ is the final state of the maneuver, and $\mathbf{u}_{\text{m}}(t)$ is the control sequence for the maneuver trajectory. The predicted obstacle state at t_{end} is $\mathbf{s}_{\text{obs},t_{\text{end}}}$ and the set $\mathcal{V}_{\text{obs}+}$ is the spatial extensions of the obstacle enlarged by the safety factor $d_{\text{obs,safety}}$ for width and length. The occupancy of the obstacle vessel is based on the assumption that the obstacle vessel will keep its speed and course, i.e., the control sequence $\mathbf{u}_{\text{keep}}(t)$. This assumption is compliant with the COLREGS collision avoidance rules for the crossing and overtake encounter. In case of the head-on encounter, the predicted trajectory for the obstacle vessel is a conservative prediction since the obstacle vessel would also need to evade to the right to be rule-compliant. Assuming that the obstacle vessel will keep its course and speed leads to the fact that the ego vessel has to turn more to resolve the encounter situation.

With the turning direction and the maneuver verification predicate defined in (8), we want to determine all actions that lead to verified maneuvers. The generation of maneuvers based on candidate actions is computed by a breadth-first search with rule-compliant pruning. The search algorithm is detailed in Alg. 2. Note that to obtain a control sequence for multiple actions, we introduce the function a2u. For a maneuver segment trajectory, the control input corresponding to an action, is held constant for a maneuver segment time t_{m} while (1) is forward simulated. We initialize a search tree with a maneuver segment trajectory resulting from the candidate turning action a_{c}. A candidate action a_{c} ensures that the orientation of the ego vessel changes at least $\Delta_{\text{large\_turn}}$ within t_{m}. Potentially, this first maneuver segment trajectory results already in a verifiable maneuver (cf. Alg. 2, line 2–3). If not, the search tree is extended by (a) a maneuver segment trajectory based on the candidate action a_{c} (cf. Alg. 2, line 17–18), and (b) with maneuver segment trajectories for each action $a \in \mathcal{A}_{\text{acc}}$, which keep the speed or accelerate the ego vessel (cf. Alg. 2, line 19–21). If the action of the maneuver segment trajectory that should be extended (obtained with the function last) does not correspond to a_{c}, the maneuver is only

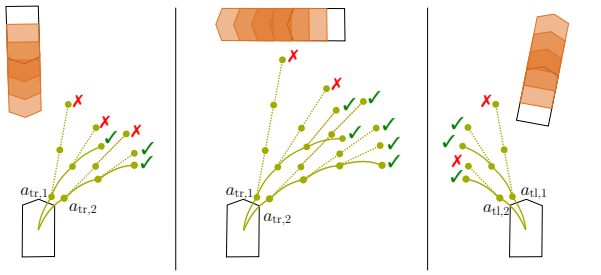
Algorithm 2 `build_st`

Input: candidate action a_c, accelerating actions $\mathcal{A}_\mathrm{acc}$, current state of obstacle vessel $\mathbf{s}_\mathrm{obs}$, current state of ego vessel $\mathbf{s}_\mathrm{ego}$, maneuver segment time t_m, maneuver horizon $t_\mathrm{max,m}$, control sequence $\mathbf{u}_\mathrm{keep}(t)$
Output: verified part of search tree $\mathcal{G}$

1: $t_\mathrm{end} \leftarrow t_\mathrm{m}, \mathcal{G} \leftarrow \{a_\mathrm{c}\}$
2: $\mathbf{u}_\mathrm{c}(t) = \mathtt{a2u}(a_\mathrm{c})$
3: **if** maneuver_verified$(\mathbf{u}_\mathrm{c}(t), ...)$ **then**
4: **return** $\mathcal{G}$
5: **else**
6: $\mathcal{U}_\mathrm{m} \leftarrow \{\mathbf{u}_\mathrm{c}(t)\}$
7: **while** $\neg$maneuver_verified$(\mathbf{u}_\mathrm{m}(t), ...)\ \forall \mathbf{u}_\mathrm{m}(t) \in \mathcal{U}_\mathrm{m}$ **do**
8: $\mathcal{U}_\mathrm{m} \leftarrow \emptyset$
9: $\mathcal{G}_\mathrm{temp} \leftarrow \emptyset$
10: **if** $t_\mathrm{end} < t_\mathrm{max,m}$ **then**
11: $t_\mathrm{end} \leftarrow t_\mathrm{end} + t_\mathrm{m}$
12: **else**
13: **return** $\mathcal{G} \leftarrow \emptyset$
14: **end if**
15: **for** $a' \in \mathcal{G}$ **do**
16: **if** $\mathtt{last}(a') = a_\mathrm{c}$ **then**
17: $\mathbf{u}_\mathrm{m}(t) \leftarrow \mathtt{a2u}(a') + \mathtt{a2u}(a_\mathrm{c})$
18: $\mathcal{U}_\mathrm{m} \leftarrow \mathbf{u}_\mathrm{m}(t), \mathcal{G}_\mathrm{temp} \leftarrow [a', a_\mathrm{c}]$
19: **for** $a_\mathrm{acc} \in \mathcal{A}_\mathrm{acc}$ **do**
20: $\mathbf{u}_\mathrm{m}(t) \leftarrow \mathtt{a2u}(a') + \mathtt{a2u}(a_\mathrm{acc})$
21: $\mathcal{U}_\mathrm{m} \leftarrow \mathbf{u}_\mathrm{m}(t), \mathcal{G}_\mathrm{temp} \leftarrow [a', a_\mathrm{acc}]$
22: **end for**
23: **else**
24: $\mathbf{u}_\mathrm{m}(t) \leftarrow \mathtt{a2u}(a') + \mathtt{a2u}(\mathtt{last}(a'))$
25: $\mathcal{U}_\mathrm{m} \leftarrow \mathbf{u}_\mathrm{m}(t), \mathcal{G}_\mathrm{temp} \leftarrow [a', \mathtt{last}(a')]$
26: **end if**
27: **end for**
28: $\mathcal{G} \leftarrow \mathcal{G}_\mathrm{temp}$
29: **end while**
30: **end if**
31: **return** $\mathcal{G}$

extended with the previously used action (cf. Alg. 2, line 24–25). This has the effect that the vessel does not switch between different accelerations during the maneuver. The expansion of the search tree is stopped (a) if at least one trajectory sequence is verified for the current search tree depth, i.e., for time horizon t_end, or (b) if the maneuver horizon $t_\mathrm{max,m}$ is reached. Note that $t_\mathrm{max,m}$ follows from the rule specification and is $t_\mathrm{react} + 2t_\mathrm{maneuver}$. The search tree generation is illustrated in Fig. 6 for three give-way encounters. Due to the rule-compliant pruning, our search algorithm has the time complexity $\mathcal{O}(n\, N_\mathrm{c}\, N_\mathrm{acc})$ for tree generation where $N_\mathrm{c} \in \mathbb{N}^+$ is the number of candidate actions a_c, and $N_\mathrm{acc} \in \mathbb{N}^+$ is the number of actions in $\mathcal{A}_\mathrm{acc}$.

c) Actions for encounter maneuvers: Alg. 3 summarizes the action verification to achieve rule-compliant maneuvers for rules R_3 - R_6 given the statechart Γ is in an encounter state (i.e., $\exists i \in \{1, ..., 4\}\colon \mathtt{in}(\rho_i)$). We denote the search tree generation with `build_st` (see Alg. 2) and the detection of



(a) Head-on (b) Crossing (c) Overtake

Fig. 6. Example search trees for the three give-way encounter situations, in which the ego vessel has to give way. The prediction of the obstacle vessel is depicted in orange and the maneuver segment trajectories in green with a dot for the final state. The trajectories based on actions from $\mathcal{A}_\mathrm{acc}$ are displayed as dashed line. Note that we display only one trajectory based on actions from $\mathcal{A}_\mathrm{acc}$ for visualization purposes. The candidate actions initializing the search trees are $a_{d,1}$ and $a_{d,2}$ where d is either tr for turning right and tl for turning left. The mark ✓ indicates that the maneuver is verified for the maneuver_verified predicate and ✗ indicates that the maneuver is not rule-compliant.

Algorithm 3 Encounter action verification

Input: stand-on action a_keep, turning to right actions $\mathcal{A}_{tr}$, turning to left actions $\mathcal{A}_{tl}$, accelerating actions $\mathcal{A}_\mathrm{acc}$, current state of obstacle vessel $\mathbf{s}_\mathrm{obs}$, current state of ego vessel $\mathbf{s}_\mathrm{ego}$, encounter predicate ψ_e
Output: set of safe actions $\mathcal{A}_s$, verified part of search tree $\mathcal{G}$

1: $\mathcal{A}_s \leftarrow \emptyset, \mathcal{G} \leftarrow \emptyset$
2: **if** $\psi_e = \mathrm{keep}$ **then**
3: $\mathcal{A}_s \leftarrow \{a_\mathrm{keep}\}$
4: **else**
5: **if** $\psi_e = \mathrm{head\_on} \vee \psi_e = \mathrm{crossing}$ **then**
6: $\mathcal{A}_\mathrm{temp} \leftarrow \mathcal{A}_{tr}$
7: **else**
8: $\mathcal{A}_\mathrm{temp} \leftarrow \mathtt{get\_turning\_act}(\mathbf{s}_\mathrm{ego}, \mathbf{s}_\mathrm{obs}, \mathcal{A}_{tr}, \mathcal{A}_{tl})$
9: **end if**
10: **for** $a \in \mathcal{A}_\mathrm{temp}$ **do**
11: $\mathcal{G}_\mathrm{temp} \leftarrow \mathtt{build\_st}(\mathbf{s}_\mathrm{ego}, \mathbf{s}_\mathrm{obs}, a, \mathcal{A}_\mathrm{acc}, t_\mathrm{m}, t_\mathrm{max,m})$
12: **if** $\mathcal{G}_\mathrm{temp} \neq \emptyset$ **then**
13: $\mathcal{G} \leftarrow \mathcal{G}_\mathrm{temp}, \mathcal{A}_s \leftarrow a$
14: **end if**
15: **end for**
16: **end if**
17: **return** $\mathcal{A}_s, \mathcal{G}$

actions in the correct turning direction for overtake situations is abbreviated by the function `get_turning_act`. The result of Alg. 3 is the safe action set $\mathcal{A}_s$ and the verified part of the search tree $\mathcal{G}$.

In an encounter situation, in which the ego vessel has to give way, a maneuver of the verified part of the search tree $\mathcal{G}$ is performed until there is no collision risk with respect to the obstacle vessel. In particular, the actions are conducted for at least the maneuver segment time t_m. At the end of a maneuver segment, the encounter situation is either resolved, or the action selection is constrained to the children of the

selected search tree node. If $\mathcal{G}$ is an empty set, the ego vessel is a stand-on vessel and the only selectable action is a_{keep}.

C. Safe-by-design Action Selection

We utilize a discrete action space for RL since this realizes efficient online safety verification and makes the encounter action verification feasible. In particular, we define an action set $\mathcal{A}$ of 49 discrete actions. One action represents the emergency action a_{em} and the others result from the combination of turning rates and accelerations:

$$\mathcal{A} = \{a_{\text{em}}, \mathcal{A}_{\text{regular}}\} \quad \text{where} \tag{9}$$
$$\mathcal{A}_{\text{regular}} = \{\mathrm{a} \times \omega \,|\, \mathrm{a} \in \mathcal{A}_{\mathrm{a}}, \omega \in \mathcal{A}_{\omega}\},$$

where $\mathcal{A}_{\mathrm{a}}$ is the finite set describing the allowed normal accelerations and $\mathcal{A}_{\omega}$ is the finite set describing the allowed turning rates.

In the previous sections, we derived the verification of rule-compliant actions. By constraining the RL agent to these rule-compliant actions, we ensure by design that only safe actions are executed, and consequently only safe trajectories are performed. Theorem 2 states the solution to our problem statement in (5).

Theorem 2. *Legal safety specified by $\langle \Phi, \leq \rangle$ can be ensured through constraining the action space of the RL agent to $\mathcal{A}_s(\hat{\rho})$ since all actions in $\mathcal{A}_s(\hat{\rho})$ are specification-compliant actions.*

Proof: To prove this statement, we derive the safe action set $\mathcal{A}_s$ for all states of the statechart Γ.

(I) Initial state ρ_0: Since no rules apply in this state as proven in Theorem 1, any action is compliant with the specification and $\mathcal{A}_s(\rho_0) = \mathcal{A}_{\text{regular}}$.

(II) Emergency state ρ_5: We constrain the actions of the RL agent to the emergency action a_{em} returned by Alg. 1, i.e., $\mathcal{A}_s(\rho_5) = a_{\text{em}}$.

(III) Encounter states $\rho_1 - \rho_4$: Based on Theorem 1 the maneuver predicates for the respective encounter situations must hold in these states to comply with the specification. Alg. 3 returns the synthesized rule-compliant maneuvers and respective actions $\mathcal{A}_s(\rho_i)$ where $i \in \{1, ..., 4\}$.

Given $\mathcal{A}_s(\rho)$, we can constrain the action selection of the RL agent to $\mathcal{A}_s(\rho)$ with standard action masking [8] to obtain the safe policy π_s. Since the safe policy π_s only allows rule-compliant actions from $\mathcal{A}_s$, the trajectories ζ_{π_s} are compliant with the legal safety specification $\langle \Phi, \leq \rangle$. ■

VI. Reinforcement Learning

For the task of autonomous vessel navigation on the open sea, we design a simulation environment based on CommonOcean benchmarks [64] and the yaw-constrained dynamics in (1). The CommonOcean benchmarks contain a planning problem which specifies the goal area and initial state of the ego vessel as well as a scenario which specifies the traffic situation, i.e., for this study the trajectory of the obstacle vessel and the navigational area. At the start of an episode, a CommonOcean benchmark is randomly selected from the training set and the agent is provided with the initial observation. Based on the observation, the agent selects an action from the action set and receives the corresponding reward and next observation of the environment (see Fig. 1). If the safety verification is activated, the agent can only select from the verified safe action set $\mathcal{A}_s$ as derived in Sec. V. We regard a setting with finite time horizon episodes and terminate the episode in specified situations (see Sec. VI-A). The observation space, termination conditions, action space, action selection constraints, and reward function are detailed in the following paragraphs.

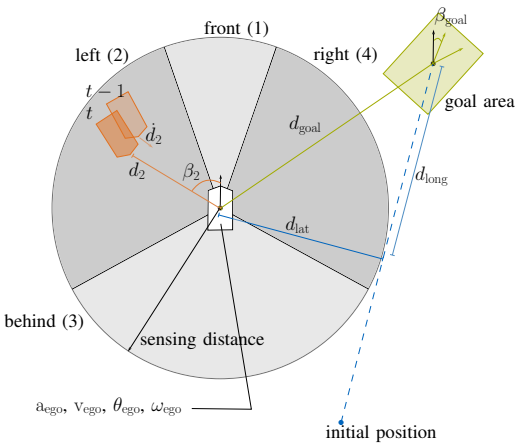


Fig. 7. Illustration of observations with sensing range and four sectors in gray, goal region in green, initial position with direct path to goal region in blue, and obstacle vessel for the previous time step $t-1$ and the current time step t in orange.

A. Observation Space and Termination

The observation space has 27 dimensions. We specify four types of observations: ego vessel observations, goal observations, surrounding traffic observations, and termination observations. Fig. 7 visualizes the ego vessel observations, goal observations, and surrounding traffic observations for the time step t.

The four ego vessel observations are the velocity v_{ego} and orientation θ_{ego} of the ego vessel state s_{ego}, the acceleration a_{ego}, and turning rate ω_{ego} corresponding to the ego vessel control input. The five continuous goal observations are the Euclidean distance to the goal d_{goal}, the remaining time steps until the maximal time step of the episode $k_{\max}$, the orientation difference to the goal orientation range β_{goal}, and the longitudinal d_{long} and lateral d_{lat} position with respect to the line from the initial state to the center of the goal state. The observations d_{long} and d_{lat} are relevant since they indicate the deviation of the ego vessel from the optimal path when no other vessels need to be avoided. Additionally, we provide one Boolean goal observation that evaluates to true whenever $\min(|d_{\text{lat}}|, |d_{\text{long}}|)$ is larger than the distance d_{hull}, i.e., the ego vessel is far away from the path between the initial state and goal area.

The surrounding traffic observations are the distance d_j, angle β_j and distance rate $\dot{d}_j$ for the detected vessel in the sector $j \in \{1, ..., J\}$, where J is the number of sectors. The

vessels are only detected if the Euclidean distance to the ego vessel is at most the sensing distance d_{sense}. For this study, we align the sectors with the sectors specified for the COLREGS collision avoidance rules. Thus, we obtain the four sectors front, left, right, and behind and twelve observation variables, as depicted in Fig. 7.

The five termination observations are Boolean observations and indicate if

- the maximal time step was reached $\mathbb{1}_{\text{time}} = 1$,
- the vessel is outside of the navigational area $\mathbb{1}_{\text{area}} = 1$,
- the vessel velocity is zero $\mathbb{1}_{\text{stopped}} = 1$,
- the vessel collided $\mathbb{1}_{\text{collision}} = 1$,
- the vessel reached the goal area $\mathbb{1}_{\text{goal}} = 1$.

We terminate the episode when the ego vessel stopped, as reverse driving is not meaningful on the open sea and the termination leads to the agent being reset to a much more meaningful initial state of another CommonOcean benchmark. The termination conditions follow directly form the termination observations, as we terminate the episode if one of these observations is present.

B. Reward

The reward is designed such that the vessel is reinforced in goal reaching behavior and penalized for unsafe or inefficient behavior. In particular, we design a reward function based on sparse and dense components. The sparse rewards are related to termination conditions and using the emergency planner:

$$\begin{aligned} r_{\text{sparse}} = & c_{\text{time}} \mathbb{1}_{\text{time}} + c_{\text{area}} \mathbb{1}_{\text{area}} + c_{\text{goal}} \mathbb{1}_{\text{goal}} + \\ & c_{\text{stopped}} \mathbb{1}_{\text{stopped}} + c_{\text{collision}} \mathbb{1}_{\text{collision}} + \\ & c_{\text{emergency}} \mathbb{1}_{\text{emergency}}, \end{aligned}$$

where c_{i} indicate the reward coefficients, which are all negative except for c_{goal}.

Additionally, we define four types of dense rewards for COLREGS compliance, advancing to the goal, keeping the velocity, and deviation from the path between initial state and goal. To incentivise behavior that is compliant with the collision avoidance rules specified in the COLREGS, we utilize a reward component specified in [41, Eq. (26)]:

$$r_{\text{colregs}} = -\frac{\alpha}{1 + \exp(\gamma_{\phi,\text{dyn}} |\phi|)} \exp((\zeta_v v_{\text{obs},\phi} - \zeta_{\text{obs,d}}) d_{\text{obs}}).$$

The angle $\phi \in [-\pi, \pi]$ specifies the relative angle between the ego orientation and the orientation toward the obstacle vessel, $v_{\text{obs},\phi}$ specifies the velocity component of the obstacle vessel velocity in the radial direction from the ego vessel to the obstacle vessel, and d_{obs} is the distance observed to the obstacle vessel, i.e., the respective d_j. The parameters $\alpha, \gamma_{\phi,\text{dyn}}, \zeta_v$, and $\zeta_{\text{obs,d}}$ are set to the same values as defined in [41].

Further, we define a reward component that supports the agent in learning how to reach the goal by providing a reward that is proportional to the advance or retreat from the goal since the previous time step:

$$r_{\text{goal}} = c_{\text{reach}} \left(\| \mathbf{p}_{\text{ego},t} - \mathbf{p}_{\text{goal}} \|_2 - \| \mathbf{p}_{\text{ego},t-1} - \mathbf{p}_{\text{goal}} \|_2 \right).$$

TABLE II
EXPERIMENTAL PARAMETERS

Parameter	Value	Parameter	Value
Safety verification			
Δ_{ahead}	$45\,\text{deg}$	Δ_{stern}	$20\,\text{deg}$
$v_{\text{pm,max}}$	$10\,\text{m}\,\text{s}^{-1}$	$\text{a}_{\text{pm,max}}$	$0.045\,\text{m}\,\text{s}^{-2}$
d_{resolved}	$2\ l_{\text{ego}}$	a_{stern}	$0.2\ \text{a}_{\text{max}}$
$d_{\text{obs,safety}}$	$2\ l_{\text{obs}}$	$d_{\text{min,ahead}}$	$3\ l_{\text{obs}}$
$\Delta_{\text{head-on}}$	$5\,\text{deg}$	$t_{\text{horizon}}^{\text{check}}$	$420\,\text{s}$
$\Delta_{\text{no\_turn}}$	$10\,\text{deg}$	t_{maneuver}	$70\,\text{s}$
$\Delta_{\text{large\_turn}}$	$20\,\text{deg}$	t_{react}	$60\,\text{s}$
t_{pred}	$180\,\text{s}$	t_{m}	$40\,\text{s}$
$t_{\text{max,m}}$	$200\,\text{s}$		
Ego vessel			
l_{ego}	$175\,\text{m}$	ω_{max}	$0.03\,\text{rad}\,\text{s}^{-1}$
a_{max}	$0.24\,\text{m}\,\text{s}^{-2}$	v_{max}	$9.5\,\text{m}\,\text{s}^{-1}$
Reinforcement learning			
v_{low}	$2.5\,\text{m}\,\text{s}^{-1}$	v_{high}	$8\,\text{m}\,\text{s}^{-1}$
c_{time}	-25	c_{area}	-5
c_{goal}	50	c_{stopped}	-40
$c_{\text{collision}}$	-50	$c_{\text{emergency}}$	-0.5
c_{reach}	1.5	c_v	-2
c_{deviate}	-0.001	d_{sense}	$8000\,\text{m}$
d_{hull}	$2000\,\text{m}$	J	4
$\mathcal{A}_{\text{a}} = \{-0.048, -0.032, -0.016, 0, 0.016, 0.032, 0.048\}\text{m}\,\text{s}^{-2}$			
$\mathcal{A}_{\omega} = \{-0.018, -0.012, -0.06, 0, 0.06, 0.012, 0.018\}\text{rad}\,\text{s}^{-1}$			

The center position of the goal area is $\mathbf{p}_{\text{goal}}$, and $\mathbf{p}_{\text{ego},t}$ is the current ego position, $\mathbf{p}_{\text{ego},t-1}$ is the ego vessel position at the previous time step, and c_{reach} is a scaling coefficient.

On the open sea, vessels typically navigate in a narrow speed range. To enforce this also for the RL agent, the reward component r_{velocity} provides a penalty proportional to the deviation from the desired speed range:

$$r_{\text{velocity}} = \begin{cases} c_v (v_{\text{ego}} - v_{\text{high}}) & \text{if } v_{\text{ego}} > v_{\text{high}} \\ c_v (v_{\text{low}} - v_{\text{ego}}) & \text{if } v_{\text{ego}} < v_{\text{low}} \\ 0 & \text{otherwise.} \end{cases}$$

The parameters v_{low} and v_{high} define the speed range bounds, and c_v is the reward coefficient.

The last reward component informs the agent about its deviation from the direct path between the initial state and the goal area:

$$r_{\text{deviate}} = c_{\text{deviate}} \min(|d_{\text{lat}}|, d_{\text{hull}}),$$

where the coefficient c_{deviate} scales the penalty proportional to the absolute lateral deviation $|d_{\text{lat}}|$, and $c_{\text{deviate}} d_{\text{hull}}$ is the maximum of the reward component r_{deviate}. Finally, the reward function is given by the sum of all components:

$$r = r_{\text{sparse}} + r_{\text{colregs}} + r_{\text{goal}} + r_{\text{velocity}} + r_{\text{deviate}}. \quad (10)$$

VII. Numerical Experiments

Critical encounter situations are rare in maritime traffic data. Thus, this data is not well suited for training RL agents that should learn how to handle encounter situations. Therefore, we construct random CommonOcean benchmarks [64] that represent critical encounters as a foundation of our simulation environment. In particular, we initialize the ego vessel and the other vessel approximately $2000\,\mathrm{m}$ - $3500\,\mathrm{m}$ away from their closest encounter position. The initial velocity range for both vessels is $[3\,\mathrm{m\,s^{-1}}, 7\,\mathrm{m\,s^{-1}}]$. For the obstacle vessel, we generate a trajectory that is close to constant velocity and speed, and disturb the initial orientation and velocity with values sampled uniformly from $[-0.05\,\mathrm{rad}, 0.05\,\mathrm{rad}]$ and $[-0.1\,\mathrm{m\,s^{-1}}, 0.1\,\mathrm{m\,s^{-1}}]$, respectively, to make the trajectory more realistic. The goal area is approximately $4500\,\mathrm{m}$ away from the initial position of the ego vessel. The goal area is $400\,\mathrm{m}$ long and $60\,\mathrm{m}$ wide. The time horizon for the scenario is $k_{\max} = 170$ time steps where the time step size is $\Delta t = 10\,\mathrm{s}$. In total, we constructed 2000 CommonOcean benchmarks [64] and randomly split them in a $70\,\%$ training and $30\,\%$ testing set. The model of the ego vessel is the yaw-constrained model in (1) and we use the parameters of a container vessel[2]. We reduce the maximum velocity specified in the vessel parameters to $9.5\,\mathrm{m\,s^{-1}}$ to better match a realistic velocity range for open sea maneuvering.

Next to the simulation environment, we need to specify values for the parameters of the safety verification approach, ego vessel, and reinforcement learning. Table II summarizes the parameters. Note that the emergency controller can use the full control input space specified for the ego vessel through the intervals $[-\,\mathrm{a_{max}}, \mathrm{a_{max}}]$ and $[-\omega_{\max}, \omega_{\max}]$. For normal operation, we reduce the control input limits to a more reasonable range for open sea maneuvering. This is reflected by the sets of allowable accelerations $\mathcal{A}_{\mathrm{a}}$ and turning rates $\mathcal{A}_{\omega}$ (see Table II). As model-free RL algorithm, we used proximal policy optimization (PPO) [65]. Our implementation is based on stable-baselines3 [66] and the action masking implementation in [8]. The agent networks are multi-layer perceptron networks with two layers and 64 neurons in each layer.

A. Evaluation concept

To comprehensively evaluate our approach, we introduce two benchmark agents next to our provably safe agent and compare different deployment setups. We train all three agents in our simulation environment, which is based on the training data of critical CommonOcean benchmarks [64]. The trained agents are:

1) the *baseline* agent with the reward function $r = r_{\mathrm{sparse}} + r_{\mathrm{goal}} + r_{\mathrm{velocity}} + r_{\mathrm{deviate}}$, i.e., $r_{\mathrm{colregs}} = 0$ in (10), and no safety verification,
2) the *rule-reward* agent, which is informed by the COLREGS reward r_{colregs}, i.e., reward function (10), and
3) the *safe* agent with safety verification and reward function (10).

The baseline agent represents a straightforward RL implementation for which the agent is informed about unsafe actions only sparsely with a collision penalty. The rule-reward agent models the state-of-the-art for traffic-rule-informed open-sea vessel navigation [6], [7], [41], [42], because the reward function includes a COLREGS reward r_{colregs}. For each agent type, we use ten random seeds and train an agent per seed for three million environment steps.

We evaluate the deployment performance of the trained agents on the testing set of the handcrafted critical scenarios and on scenarios from recorded traffic data[3]. For the rule-reward and baseline agent, we investigate performance without, i.e., as trained, and with safety verification enabled. Including the safety verification after training allows us to evaluate if guaranteeing traffic rule compliance after training is sufficient. Note that the action space of the two benchmark agents is $\mathcal{A}_{\mathrm{regular}}$, except for deployment with safety verification.

We consider critical scenarios from recorded traffic data to examine the generalization of the agents to real-world situations. To this end, we use marine traffic data from three large open-sea areas off the US coast from [15] and extract critical encounters. In particular, we only use scenarios where the distance between two vessels drops to $5000\,\mathrm{m}$ or lower. Further, we ensure that the paths of both vessels cross each other. Then, we replace one vessel by an ego vessel to generate the initial state and goal area. The initial state is part of the recorded trajectory and is selected about $2000\,\mathrm{m}$ before the closest encounter. The position of the goal area is also part of the recorded trajectory and is about $2000\,\mathrm{m}$ after the closest encounter. We use the same shape for the goal area as in our handcrafted scenarios. In total, we identify 49 critical scenarios in the three large open-sea areas off the US coast from traffic data of January 2019 (about 30 GB of raw Automatic Identification System (AIS) data).

We evaluate our agents based on the goal-reaching rate, reward, episode lengths, collisions, emergency controller usage and rule violations. Rule violations reflect how often per episode the regular collision avoidance rules are violated. For that, we count:

- every time step of violating the stand-on vessel position results;
- every crossing, overtaking and head-on encounter for which no proper collision avoidance maneuver is taken.

B. Results

a) Training evaluation: Fig. 8 shows the training curves for the three agent types. The average reward curves show similar convergence across agent types, although the baseline and rule-reward agents achieve slightly higher rewards after three million training steps. Note that for the displayed reward curves, the emergency penalty and COLREGS reward term r_{colregs} are subtracted for comparability. The goal-reaching

[2]The container vessel is the vessel type 1 from commonocean.cps.cit.tum.de/commonocean-models.

[3]All scenarios are publicly on the CommonOcean website with ids ZAM_AAA-1_20240121_T-[0,...,1999].

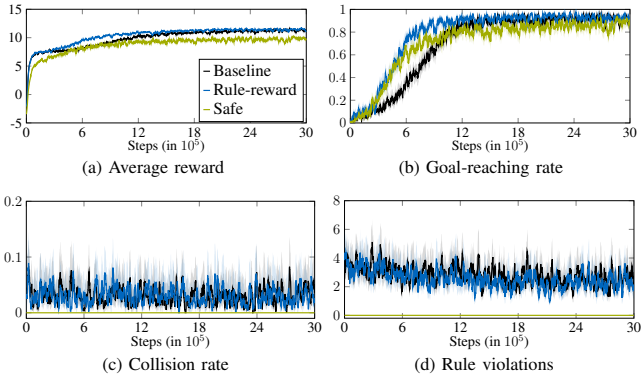


Fig. 8. Mean and bootstrapped 95 % confidence interval for training curves for baseline, rule-reward, and safe agents averaged over ten random seeds.

rate curves mirror the reward curves and the agents reach goals in about 90 % of all scenarios at the end of the training. We observe that the agent types without safety verification reach the goal slightly more often.

Importantly, there are no collisions and rule violations for the safe agent (see Fig. 8c and Fig. 8d). For the baseline and rule-reward agent, the collision rate is relatively stable around 5 % during the full training time. Rule violations for the baseline and rule-reward agent slightly decrease but never reach zero. This suggests that complying with the COLREGS effectively achieves collision avoidance.

b) Deployment evaluation: The results averaged over ten random seeds for each agent type are summarized in Table III. For the handcrafted scenarios, the rule-reward agents reach the goal for 90.7 % of the scenarios. This is about 5 % higher than for the baseline and safe agent. Yet, only the safe agent achieves zero collisions and no rule violations. The rule-reward agent collides and violates the rules fewer times than the baseline agent. If the safety verification is enabled for the baseline and rule-reward agent, the goal-reaching rate drops significantly by approximately 40 %. Additionally, for the safe agent, the emergency controller intervenes on average in 6 % of the time steps in an episode, whereas for the rule-reward and baseline agents with activated safety verification, the emergency controller is needed in approximately 10 % of the time steps in an episode.

Table III displays the testing results on the 49 critical recorded traffic scenarios for the different agent types. The rule-reward agent reaches the goal most often and exhibits the lowest average episode length. Interestingly, the goal-reaching rate for the baseline and rule-reward agent drops only by about 5 % when activating our safety verification approach. The collision rate and rule violation rate are smaller than for the handcrafted scenarios. With activated safety verification, we observe no collisions and no rule violations. Note that the differences in the reported means for goal-reaching rate and emergency steps between the agents with activated safety verification are statistically insignificant[4].

[4]The p-values for paired t-tests between the safe agent and the rule-reward and baseline agents for the goal-reaching rate are 0.248 and 0.951, respectively. The p-values for agents with respect to emergency steps are 0.211 and 0.381.

TABLE III
TESTING RESULTS ON 600 HANDCRAFTED AND 49 RECORDED SCENARIOS

Setup		**Efficiency**		**Safety**		
Agent	Verify	Goal-reach	Ep. length	Collided	Rules violated	Emerg. steps
Handcrafted testing scenarios						
Base	✗	86.8 %	566 s	3.13 %	2.65	–
RR	✗	**90.7 %**	**544 s**	2.85 %	2.24	–
Base	✓	44.0 %	678 s	**0.0 %**	**0**	10.06 %
RR	✓	47.6 %	702 s	**0.0 %**	**0**	9.96 %
Safe	✓	86.3 %	647 s	**0.0 %**	**0**	**6.18 %**
Recorded maritime traffic scenarios						
Base	✗	83.1 %	563 s	0.41 %	0.75	–
RR	✗	**84.7 %**	**550 s**	0.41 %	0.82	–
Base	✓	78.3 %	591 s	**0.0 %**	**0**	2.35 %
RR	✓	82.4 %	565 s	**0.0 %**	**0**	**1.84 %**
Safe	✓	78.2 %	630 s	**0.0 %**	**0**	2.98 %

Note: The rule-reward and baseline agents are abbreviated with RR and base. Ep. length is the average episode time horizon. Emerg. steps denote the percentage of steps for which the emergency controller intervened.

C. Discussion

a) Safety in handcrafted scenarios: The safety verification ensures that the encounter traffic rules are never violated and we empirically observe that no collisions occur. However, this results in a lower goal-reaching rate than for the soft-constrained rule-reward agent. One reason for this observation might be that with safety verification, the task is more difficult to solve since the agent is often constrained to avoidance maneuvers before it can maneuver freely again. Thus, the safe agent can explore less freely compared to the baseline and rule-reward agents. The drop in the goal-reaching rate when the safety verification is enabled after training is likely due to the distribution shift, as the baseline and rule-reward agents are probable led to states that they explored less frequently or not at all during training.

b) Safety on recorded scenarios: In contrast, testing the rule-reward and baseline agent with safety verification on the scenarios from recorded traffic data does not lead to such a significant drop. At the same time, the agent setups without safety verification exhibit fewer rule violations and fewer collisions on the recorded maritime traffic scenarios. Both observations indicate that the scenarios based on recorded data are less critical than the handcrafted situations and, thus, easier to solve for the agents that were not constrained to rule-compliant actions during training. Generally, the agents generalize well to the scenarios based on recorded data. Since identifying critical situations in recorded maritime traffic data is computation-heavy and critical situations are very rare, this small gap between realistic recorded and randomly handcrafted situations is compensated by being able to create many scenarios: The 49 critical situations resulted from one month of maritime traffic data at the coast of the US, whereas the 2000 handcrafted critical situations were generated in a matter of minutes. Yet, recorded scenarios are not fully representing the variety of the real world. Thus, future work should investigate

if our safe agent also performs well on a real-world test bed.

c) Requirements for multi-vessel traffic situations: Real-world traffic situations can include more than two vessel on a collision course. Our formalized traffic rules can be evaluated for these more complex traffic situations as demonstrated in [15]. Yet, the current version of the COLREGS does not provide a clear collision avoidance specification if more than two vessels are involved. Thus, a formal verification cannot be developed due to the lack of a clear specification. Future work should investigate extensions of the COLREGS to fill this specification gap and consequently realize provably rule-compliant motion planning in multi-vessel traffic situations.

d) Action space choice: The discrete action space makes it possible to efficiently identify rule-compliant actions. However, a continuous action space would allow the agent to explore all possible actions. This significantly increases the challenge of identifying safe actions, because there are infinitely many individual continuous actions in a continuous action space. Yet, one approach to investigate in future work could be obtaining rule-compliant state sets as proposed in [67] and correcting actions proposed by the agent to safe actions, e.g., with action projection as in [68].

e) Satisfiablity of rules: The parametrization of the temporal logic rules eases re-adjusting to regulation changes. Yet, these parameters must be manually tuned to ensure that the temporal logic rules are satisfiable. For example, it is important that the detection of an encounter situation happens early enough so that no emergency situation is detected during a give-way maneuver. For instance, theorem provers could help to verify that the chosen rule parameters guarantee that the rules are satisfiable. However, formulating this proof is challenging due to the continuous state and action space, and subject to future work.

VIII. Conclusion

We are the first to propose a provably safe RL approach for autonomous power-driven vessels on the open sea that achieves provable compliance with traffic rules formalized with temporal logic. For that, we introduced an online verification approach based on our formalized rules identifying the set of safe actions. Our formal emergency detection and emergency controller achieves collision avoidance for the regarded traffic situations even if other vessels do not comply with traffic rules. In critical maritime traffic situations, our safe RL agent achieves rule compliance, in contrast to state-of-the art agents that are informed about safety only through the reward. At the same time, all agents achieve a satisfactory goal-reaching performance on critical traffic situations. Our evaluation on recorded traffic situations shows that our safe RL agent generalizes beyond the distribution of training data. This study is a first step toward learning-based motion planning systems complying with traffic rules for autonomous vessel navigation.

Acknowledgment

The authors gratefully acknowledge the partial financial support of this work by the research training group ConVeY funded by the German Research Foundation under grant GRK 2428 and by the project TRAITS funded by the German Federal Ministry of Education and Research.

References

[1] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Sallab, S. Yogamani, and P. Perez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2022.

[2] F. Ye, S. Zhang, P. Wang, and C. Y. Chan, "A survey of deep reinforcement learning algorithms for motion planning and control of autonomous vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2021, pp. 1073–1080.

[3] M. El-Shamouty, X. Wu, S. Yang, M. Albus, and M. F. Huber, "Towards safe human-robot collaboration using deep reinforcement learning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 4899–4905.

[4] D. Han, B. Mulyana, V. Stankovic, and S. Cheng, "A survey on deep reinforcement learning algorithms for robotic manipulation," *Sensors*, vol. 23, no. 7, 2023.

[5] P. Sarhadi, W. Naeem, and N. Athanasopoulos, "A survey of recent machine learning solutions for ship collision avoidance and mission planning," *IFAC-PapersOnLine*, vol. 55, no. 31, pp. 257–268, 2022.

[6] A. Heiberg, T. N. Larsen, E. Meyer, A. Rasheed, O. San, and D. Varagnolo, "Risk-based implementation of COLREGs for autonomous surface vehicles using deep reinforcement learning," *Neural Networks*, vol. 152, pp. 17–33, 2022.

[7] X. Xu, P. Cai, Z. Ahmed, V. S. Yellapu, and W. Zhang, "Path planning and dynamic collision avoidance algorithm under COLREGs via deep reinforcement learning," *Neurocomputing*, vol. 468, pp. 181–197, 2022.

[8] H. Krasowski, J. Thumm, M. Müller, L. Schäfer, X. Wang, and M. Althoff, "Provably safe reinforcement learning: Conceptual analysis, survey, and benchmarking," *Transactions on Machine Learning Research*, 2023.

[9] B. Vanholme, D. Gruyer, B. Lusetti, S. Glaser, and S. Mammar, "Highly automated driving on highways based on legal safety," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 333–347, 2013.

[10] N. Mehdipour, M. Althoff, R. D. Tebbens, and C. Belta, "Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges," *Automatica*, vol. 152, 2023.

[11] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1555–1562.

[12] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scLTL motion planning for mobility-on-demand," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2017, pp. 1481–1488.

[13] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 752–759.

[14] K. Esterle, L. Gressenbuch, and A. Knoll, "Formalizing traffic rules for machine interpretability," in *Proc. of the IEEE Connected and Automated Vehicles Symposium*, 2020, pp. 1–7.

[15] H. Krasowski and M. Althoff, "Temporal logic formalization of marine traffic rules," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2021, pp. 186–192.

[16] X. Zhang, C. Wang, L. Jiang, L. An, and R. Yang, "Collision-avoidance navigation systems for maritime autonomous surface ships: A state of the art survey," *Ocean Engineering*, vol. 235, no. 109380, 2021.

[17] "COLREGs: Convention on the International Regulations for Preventing Collisions at Sea," International Maritime Organization (IMO), 1972.

[18] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger, "Safe maritime autonomous navigation with COLREGS, using velocity obstacles," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 110–119, 2014.

[19] L. Zhao and M. I. Roh, "COLREGs-compliant multiship collision avoidance based on deep reinforcement learning," *Ocean Engineering*, vol. 191, pp. 106 436–106 450, 2019.

[20] S. Guo, X. Zhang, Y. Zheng, and Y. Du, "An autonomous path planning model for unmanned ships based on deep reinforcement learning," *Sensors*, vol. 20, no. 2, 2020.

[21] X. Zhang, C. Wang, Y. Liu, and X. Chen, "Decision-making for the autonomous navigation of maritime autonomous surface ships based on scene division and deep reinforcement learning," *Sensors*, vol. 19, no. 18, 2019.

[22] M. Junmin, L. Mengxia, H. Weixuan, Z. Xiaohan, G. Shuai, C. Pengfei, and H. Yixiong, "Mechanism of dynamic automatic collision avoidance and the optimal route in multi-ship encounter situations," *Journal of Marine Science and Technology*, vol. 26, pp. 141–158, 2021.

[23] Y. He, Y. Jin, L. Huang, Y. Xiong, P. Chen, and J. Mou, "Quantitative analysis of COLREG rules and seamanship for autonomous collision avoidance at open sea," *Ocean Engineering*, vol. 140, pp. 281–291, 2017.

[24] H. T. L. Chiang and L. Tapia, "COLREG-RRT: An RRT-based COLREGS-compliant motion planner for surface vehicle navigation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2024–2031, 2018.

[25] M. R. Benjamin and J. A. Curcio, "COLREGS-based navigation of autonomous marine vehicles," in *Proc. of the IEEE/OES Autonomous Underwater Vehicles*, 2004, pp. 32–39.

[26] T. A. Johansen, T. Perez, and A. Cristofaro, "Ship collision avoidance and COLREGS compliance using simulation-based control behavior selection with predictive hazard assessment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3407–3422, 2016.

[27] B. O. H. Eriksen, M. Breivik, E. F. Wilthil, A. L. Flåten, and E. F. Brekke, "The branching-course model predictive control algorithm for maritime collision avoidance," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1222–1249, 2019.

[28] D. K. Kufoalor, E. Wilthil, I. B. Hagen, E. F. Brekke, and T. A. Johansen, "Autonomous COLREGs-compliant decision making using maritime radar tracking and model predictive control," in *Proc. of the European Control Conference*, 2019, pp. 2536–2542.

[29] P. Stankiewicz and M. Kobilarov, "A primitive-based approach to good seamanship path planning for autonomous surface vessels," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2021, pp. 7767–7773.

[30] T. R. Torben, J. A. Glomsrud, T. A. Pedersen, I. B. Utne, and A. J. Sørensen, "Automatic simulation-based testing of autonomous ships using Gaussian processes and temporal logic," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 237, no. 2, pp. 293–313, 2023.

[31] Y. Liu and R. Bucknall, "A survey of formation control and motion planning of multiple unmanned vehicles," *Robotica*, vol. 36, no. 7, pp. 1019—-1047, 2018.

[32] W. Wu, Z. Peng, L. Liu, and D. Wang, "A general safety-certified cooperative control architecture for interconnected intelligent surface vehicles with applications to vessel train," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 627–637, 2022.

[33] W. Wu and S. Tong, "Collision-free finite-time adaptive fuzzy output-feedback formation control for unmanned surface vehicle systems," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 1094–.1103, 2024.

[34] Y. Zhao, Y. Ma, and S. Hu, "USV formation and path-following control via deep reinforcement learning with random braking," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5468–5478, 2021.

[35] Y. Zhang, W. Wu, and W. Zhang, "Noncooperative game-based cooperative maneuvering of intelligent surface vehicles via accelerated learning-based neural predictors," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 3, pp. 2212–2221, 2023.

[36] G. Wen, X. Fang, J. Zhou, and J. Zhou, "Robust formation tracking of multiple autonomous surface vessels with individual objectives: A noncooperative game-based approach," *Control Engineering Practice*, vol. 119, 2022.

[37] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, 2011.

[38] J. Zhang, H. Zhang, J. Liu, D. Wu, and C. G. Soares, "A two-stage path planning algorithm based on rapid-exploring random tree for ships navigating in multi-obstacle water areas considering COLREGs," *Journal of Marine Science and Engineering*, vol. 10, no. 1441, 2022.

[39] T. T. Enevoldsen, C. Reinartz, and R. Galeazzi, "COLREGs-informed RRT* for collision avoidance of marine crafts," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2021, pp. 8083–8089.

[40] A. Tsolakis, D. Benders, O. De Groot, R. R. Negenborn, V. Reppa, and L. Ferranti, "COLREGs-aware trajectory optimization for autonomous surface vessels," *IFAC-PapersOnLine*, vol. 55, no. 31, pp. 269–274, 2022.

[41] E. Meyer, A. Heiberg, A. Rasheed, and O. San, "COLREG-compliant collision avoidance for unmanned surface vehicle using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 165 344–165 364, 2020.

[42] D.-H. Chun, M.-I. Roh, H.-W. Lee, J. Ha, and D. Yu, "Deep reinforcement learning-based collision avoidance for an autonomous ship," *Ocean Engineering*, vol. 234, 2021.

[43] W. Xie, L. Gang, M. Zhang, T. Liu, and Z. Lan, "Optimizing multi-vessel collision avoidance decision making for autonomous surface vessels: A colregs-compliant deep reinforcement learning approach," *Journal of Marine Science and Engineering*, vol. 12, no. 3, 2024.

[44] Y. Fan, Z. Sun, and G. Wang, "A novel intelligent collision avoidance algorithm based on deep reinforcement learning approach for usv," *Ocean Engineering*, vol. 287, 2023.

[45] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods: Toward safe control through proof and learning," in *Proc. of the AAAI Conf. on Artificial Intelligence*, 2018, pp. 6485–6492.

[46] ——, "Verifiably safe off-model reinforcement learning," in *Proc. of the Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, 2019, pp. 413–430.

[47] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *Proc. of the IEEE Int. Intelligent Transportation Systems Conference*, 2018, pp. 2156–2162.

[48] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2020.

[49] M. Brosowsky, F. Keck, J. Ketterer, S. Isele, D. Slieter, and M. Zöllner, "Safe deep reinforcement learning for adaptive cruise control by imposing state-specific safe sets," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2021, pp. 488–495.

[50] H. Krasowski, Y. Zhang, and M. Althoff, "Safe reinforcement learning for urban driving using invariably safe braking sets," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2022, pp. 2407–2414.

[51] D. Tabas and B. Zhang, "Computationally efficient safe reinforcement learning for power systems," in *Proc. of the American Control Conference*, 2022, pp. 3303–3310.

[52] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. of the AAAI Conf. on Artificial Intelligence*, 2018, pp. 2669–2678.

[53] B. Könighofer, F. Lorber, N. Jansen, and R. Bloem, "Shield synthesis for reinforcement learning," in *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*, 2020, pp. 290–306.

[54] X. Li, Z. Serlin, G. Yang, and C. Belta, "A formal methods approach to interpretable reinforcement learning for robotic planning," *Science Robotics*, vol. 4, no. 37, 2019.

[55] A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli, "Liability, ethics, and culture-aware behavior specification using rulebooks," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2019, pp. 8536–8542.

[56] M. Koschi and M. Althoff, "Set-based prediction of traffic participants considering occlusions and traffic rules," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 249–265, 2021.

[57] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff, "Model conformance for cyber-physical systems: A survey," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 3, pp. 1–26, 2019.

[58] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Dissertation, Technische Universität München, 2010.

[59] M. Wetzlinger, N. Kochdumper, S. Bak, and M. Althoff, "Fully automated verification of linear systems using inner and outer approximations of reachable sets," *IEEE Transactions on Automatic Control*, vol. 68, no. 12, pp. 7771–7786, 2023.

[60] S. Magdici and M. Althoff, "Fail-safe motion planning of autonomous vehicles," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2016, pp. 452–458.

[61] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[62] T. Schouwenaars, J. How, and E. Feron, "Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees," in *Proc. of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004.

[63] S. Bouraine, T. Fraichard, and H. Salhi, "Provably safe navigation for mobile robots with limited field-of-views in dynamic environments," *Auton. Robots*, vol. 32, no. 3, pp. 267–283, 2012.

[64] H. Krasowski and M. Althoff, "CommonOcean: Composable benchmarks for motion planning on oceans," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2022, pp. 1676–1682.

[65] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[66] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.
[67] E. Irani Liu and M. Althoff, "Specification-compliant driving corridors for motion planning of automated vehicles," *IEEE Transactions on Intelligent Vehicles*, 2023.
[68] N. Kochdumper, H. Krasowski, X. Wang, S. Bak, and M. Althoff, "Provably safe reinforcement learning via action projection using reachability analysis and polynomial zonotopes," *IEEE Open Journal of Control Systems*, vol. 2, pp. 79–92, 2023.

APPENDIX

A. Predicates specified [15]

In Table IV, we briefly recapitulate the predicates specified in [15]. We refer the interested reader to our previous work [15] for detailed explanations. Subsequently, the necessary notation that was not yet introduced in this article is introduced and the re-parametrization of the predicate $\mathrm{collision\_possible}$ is explained.

The trajectory of vessel i consists of states at discrete time steps and is denoted as $\mathcal{T}_i$. The velocity vector based on the state of the vessel is $\mathbf{v}_i = \mathtt{proj}_v(\mathbf{s}_i)\ \mathtt{unit\_v}(\mathbf{s}_i)$. We define a clock $\mathtt{cl}(\mathcal{T}_i, \mathbf{s}_i)$ that starts at the initial time step of a trajectory and returns the elapsed time for a state $\mathbf{s}_i$. Further, we require a function $\mathtt{state}(\mathcal{T}_i, t_k)$ which returns the state of a trajectory at time t_k. The modulo operator $\mathrm{mod}(a, b)$ returns the remainder of a/b for $a, b \in \mathbb{R}$ using floored division. The function $\mathtt{t}_s$ returns the time for a predicate trace where the respective predicates changed last from false to true. The collision cone CC' is based on the velocity obstacle concept [61] and the construction is detailed in [15, Fig. 1].

For this work, we made two re-parametrizations of the predicate $\mathrm{collision\_possible}$, which determines if two vessels l and m are on a collision course and, thus, could collide within the time t_{horizon}. First, we also want to detect a collision course if the vessels would pass each other with insufficient distance. Thus, we use $r_m = 3\, l_m$ for the collision cone CC' instead of $r_m = l_m$ in [15, Fig. 1]. This results in detecting a collision possibility if the vessels would not keep a safe distance of at least two lengths of the vessel m. Second, we evaluate the set of vessel velocities $\mathcal{V}_l$ with respect to their collision possibility instead of only the current velocity $\mathbf{v}_l$. In particular, we check the collision possibility for

$$\mathcal{V}_l = \{\lambda\, \mathtt{unit\_v}(\mathbf{s}_l) | \lambda \in [\mathtt{proj}_v(\mathbf{s}_l) - v_\epsilon, \mathtt{proj}_v(\mathbf{s}_l) + v_\epsilon]\}.$$

We set the velocity difference v_ϵ to $1\,\mathrm{m\,s^{-1}}$ for our numerical evaluations.

B. Proof of Lemma 1

Proof: To prove that only one predicate of keep, $\mathrm{crossing}$, $\mathrm{head\_on}$, and $\mathrm{overtake}$ can evaluate to true, we show for each combination that the conjunction is false when evaluated for two vessels l and m. For the combination of $\mathrm{crossing}$ and $\mathrm{head\_on}$, it directly follows that the predicates cannot be true at the same time from the relative position detected by the respective sector predicates.

(I) $\mathrm{crossing} \wedge \mathrm{head\_on}$:

$$\begin{aligned}
&\mathrm{crossing}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge \mathrm{head_on}(\mathbf{s}_l, \mathbf{s}_m, \cdot)\\
=&\,(\mathrm{in_right_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...) \wedge\\
&\,(\mathrm{in_front_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...)\\
=&\,\bot
\end{aligned}$$

For the combination of $\mathrm{crossing}$ and $\mathrm{overtake}$, let us assume that crossing predicate is true. Then, the vessel m is oriented towards left and in the right sector of vessel l (see Fig. 3 and Fig. 4 in [15]). Thus, it is geometrically impossible for vessel l to be in the behind sector of vessel m and $\mathrm{overtake}$ cannot be true.

(II) $\mathrm{crossing} \wedge \mathrm{overtake}$:

$$\begin{aligned}
&\mathrm{crossing}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge \mathrm{overtake}(\mathbf{s}_l, \mathbf{s}_m, \cdot)\\
=&\big(\mathrm{in_right_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge\\
&\,\mathrm{orientation_towards_left}(\mathbf{s}_l, \mathbf{s}_m, \Delta_{\text{head-on}}) \wedge ...\big) \wedge\\
&\,(\mathrm{in_behind_sector}(\mathbf{s}_m, \mathbf{s}_l) \wedge ...)\\
=&\,\bot
\end{aligned}$$

The predicates $\mathrm{head\_on}$ and $\mathrm{overtake}$ cannot be true simultaneously as the relative positions and orientations contradict each other similar to case (II). In particular, if the vessel m is in the front sector of vessel l and their relative orientation is in $[\pi - \Delta_{\text{head-on}}, \pi + \Delta_{\text{head-on}}]$, then vessel l cannot be in the behind sector of vessel m.

(III) $\mathrm{head\_on} \wedge \mathrm{overtake}$:

$$\begin{aligned}
&\mathrm{head_on}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge \mathrm{overtake}(\mathbf{s}_l, \mathbf{s}_m, \cdot)\\
=&\big(\mathrm{in_front_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge\\
&\,\neg\mathrm{orientation_delta}(\mathbf{s}_l, \mathbf{s}_m, \Delta_{\text{head-on}}, \pi) \wedge ...\big) \wedge\\
&\,(\mathrm{in_behind_sector}(\mathbf{s}_m, \mathbf{s}_l) \wedge ...)\\
=&\,\bot
\end{aligned}$$

The predicate keep is a disjunction of two cases in which the vessel has to keep its course and speed. Thus, we have to show that for both statements of the disjunction that they evaluate to false. The explanation for the equation steps are marked with small letters in round brackets, e.g., (a), and follow after the respective equations.

(IV) $\mathrm{overtake} \wedge \mathrm{keep}$:

$$\begin{aligned}
&\mathrm{overtake}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge \mathrm{keep}(\mathbf{s}_l, \mathbf{s}_m, \cdot)\\
\overset{(a)}{=}&\,(\mathrm{overtake}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge (\mathrm{in_left_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...)) \vee\\
&\,(\mathrm{overtake}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge \mathrm{overtake}(\mathbf{s}_m, \mathbf{s}_l, \cdot))\\
\overset{(b)}{=}&\Big(\big(\mathrm{in_behind_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...\big) \wedge\\
&\,\big(\mathrm{in_left_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...\big)\Big) \vee\\
&\,(\mathrm{overtake}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge \mathrm{overtake}(\mathbf{s}_m, \mathbf{s}_l, \cdot))\\
\overset{(c)}{=}&\,\bot \vee \bot\\
=&\,\bot
\end{aligned}$$

(a) We distribute the disjunction in keep over the conjunction with overtake.
(b) We insert the relevant parts of the predicates (see Table IV).
(c) For the first part of the disjunction, the vessels cannot be simultaneously in two sectors as in case (I). For the second part of the disjunction, the two overtake predicates cannot be true at the same time, as both vessels cannot overtake each other at the same time.

(V) crossing $\wedge$ keep:

$$
\begin{aligned}
&\text{crossing}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge \text{keep}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \\
&\overset{(a)}{=} (\text{crossing}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge (\text{in\_left\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...)) \vee \\
&\quad (\text{crossing}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge \text{overtake}(\mathbf{s}_m, \mathbf{s}_l, \cdot)) \\
&\overset{(b)}{=} \Big(\big(\text{in\_right\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...\big) \wedge \\
&\quad \big(\text{in\_left\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...\big) \Big) \vee \\
&\quad \Big(\big(\text{in\_right\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...\big) \wedge \\
&\quad \big(\text{in\_behind\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...\big) \Big) \\
&\overset{(c)}{=} \bot \vee \bot \\
&= \bot
\end{aligned}
$$

(a) We distribute the disjunction in keep over the conjunction with crossing.
(b) We insert the relevant parts of the predicates (see Table IV).
(c) For both parts of the disjunction, the vessels cannot be simultaneously in two sectors as in case (I).

(VI) head_on $\wedge$ keep:

$$
\begin{aligned}
&\text{head\_on}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge \text{keep}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \\
&\overset{(a)}{=} (\text{head\_on}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge (\text{in\_left\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...)) \vee \\
&\quad (\text{head\_on}(\mathbf{s}_l, \mathbf{s}_m, \cdot) \wedge \text{overtake}(\mathbf{s}_m, \mathbf{s}_l, \cdot)) \\
&\overset{(b)}{=} \Big(\big(\text{in\_front\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...\big) \wedge \\
&\quad \big(\text{in\_left\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...\big) \Big) \vee \\
&\quad \Big(\big(\text{in\_front\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...\big) \wedge \\
&\quad \big(\text{in\_behind\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge ...\big) \Big) \\
&\overset{(c)}{=} \bot \vee \bot \\
&= \bot
\end{aligned}
$$

(a) We distribute the disjunction in keep over the conjunction with head_on.
(b) We insert the relevant parts of the predicates (see Table IV).
(c) For both parts of the disjunction, the vessels cannot be simultaneously in two sectors as in case (I). ■

C. Position Tracking Controller

We design a Lyapunov controller to track a desired position $\mathbf{p}_{\text{des}}$ to realize the emergency maneuver control. This desired position is either the target position $\mathbf{p}_{\text{target}}$ to be reached or generated based on the current position $\mathbf{p}_t$ and the desired position so that the vessel approximately maintains the desired velocity v_{desired}. The Lyapunov function for turning rate V_ω and acceleration V_{a} are:

$$
\begin{aligned}
V_\omega &= 1 - ([\cos(\theta_t), \sin(\theta_t)]\mathbf{w}_{\text{des}}^T)^2, \\
V_{\text{a}} &= 0.5\,(\mathbf{p}_{\text{des}} - \mathbf{p}_t)(\mathbf{p}_{\text{des}} - \mathbf{p}_t)^T,
\end{aligned}
$$

with the desired orientation vector

$$
\mathbf{w}_{\text{des}} = (\mathbf{p}_{\text{des}} - \mathbf{p}_t) / \| \mathbf{p}_{\text{des}} - \mathbf{p}_t \|_2.
$$

With these Lyapunov functions, we obtain the control:

$$
\begin{aligned}
\omega &= -\lambda_1 \frac{V_\omega}{-2\,([-\sin(\theta), \cos(\theta)]\mathbf{w}_{\text{des}}^T)\,([\cos(\theta), \sin(\theta)]\mathbf{w}_{\text{des}}^T)}, \\
\text{a} &= -\lambda_2 \frac{V_{\text{a}}}{-(\mathbf{p}_{\text{des}} - \mathbf{p}_t)[v_t \cos(\theta_t), v_t \sin(\theta_t)]^T}.
\end{aligned}
$$

If V_ω is larger than a threshold Δ_{V_ω}, then the acceleration control is set to zero so that the vessel only turns. For our numerical evaluations, we use the following parameter values: $v_{\text{desired}} = 6\,\text{m}\,\text{s}^{-1}$, $\lambda_1 = 4$, $\lambda_2 = 0.04$, and $\Delta_{V_\omega} = 0.3$.

Hanna Krasowski is currently a Ph.D. candidate at the Technical University of Munich. She received her B.Sc. degree in mechanical engineering from Technical University of Darmstadt in 2017 and her M.Sc. degree in robotics, cognition and intelligence from Technical University of Munich in 2020. Her research interests include provably safe reinforcement learning and motion planning for cyber-physical systems.

Matthias Althoff received the Diploma Engineering degree in mechanical engineering and the Ph.D. degree in electrical engineering from Technical University of Munich, Germany, in 2005 and 2010, respectively. He is currently an Associate Professor in computer science with Technical University of Munich, Germany. From 2010 to 2012 he was a Postdoctoral Researcher with Carnegie Mellon University, Pittsburgh, PA, USA, and from 2012 to 2013 an Assistant Professor with Technische Universität Ilmenau, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.

TABLE IV
PREDICATES FOR TRAFFIC RULE SPECIFICATIONS FROM [15] WITH ADAPTIONS FOR THIS WORK

Predicate	**Arguments**	**Definition**	**Detects ...**
Position and orientation predicates			
in_front_sector	$\mathbf{s}_l, \mathbf{s}_m$	$\text{in\_sector}(\mathbf{s}_l, \mathbf{s}_m, -\Delta_{\text{head-on}}, \Delta_{\text{head-on}})$	relative position in front sector
in_left_sector	$\mathbf{s}_l, \mathbf{s}_m$	$\text{in\_sector}(\mathbf{s}_l, \mathbf{s}_m, -112.5°, -\Delta_{\text{head-on}})$	relative position in left sector
in_right_sector	$\mathbf{s}_l, \mathbf{s}_m$	$\text{in\_sector}(\mathbf{s}_l, \mathbf{s}_m, \Delta_{\text{head-on}}, 112.5°)$	relative position in right sector
in_behind_sector	$\mathbf{s}_l, \mathbf{s}_m$	$\text{in\_sector}(\mathbf{s}_l, \mathbf{s}_m, 112.5°, 247.5°)$	relative position in behind sector
orientation_delta	$\mathbf{s}_l, \mathbf{s}_m, \Delta_{\text{orient}}, c_{\text{o}}$	$\text{mod}(\text{proj}_\theta(\mathbf{s}_m) - \text{proj}_\theta(\mathbf{s}_l) + c_{\text{o}}, 2\pi) \in [\Delta_{\text{orient}}, 2\pi - \Delta_{\text{orient}}]$	if relative orientation is in defined range
orientation_towards_right	$\mathbf{s}_l, \mathbf{s}_m, \Delta_{\text{head-on}}$	$\text{mod}(\text{proj}_\theta(\mathbf{s}_m) - \text{proj}_\theta(\mathbf{s}_l), 2\pi) \in [-\pi + \Delta_{\text{head-on}}, -\Delta_{\text{head-on}}]$	if relative orientation of vessel m is toward right
orientation_towards_left	$\mathbf{s}_l, \mathbf{s}_m, \Delta_{\text{head-on}}$	$\text{mod}(\text{proj}_\theta(\mathbf{s}_m) - \text{proj}_\theta(\mathbf{s}_l), 2\pi) \in [\Delta_{\text{head-on}}, \pi - \Delta_{\text{head-on}}]$	if relative orientation of vessel m is toward right
Velocity predicates			
drives_faster	$\mathbf{s}_l, \mathbf{s}_m$	$\texttt{proj}_v(\mathbf{s}_l) > \texttt{proj}_v(\mathbf{s}_m)$	if vessel l is faster than vessel m
safe_speed	$\mathbf{s}_l, v_{\max}$	$0 \leq \texttt{proj}_v(\mathbf{s}_l) \leq v_{\max}$	safe speed of vessel l
General predicates			
collision_possible	$\mathbf{s}_l, \mathbf{s}_m, t_{\text{horizon}}$	$\mathcal{V}_l \in CC'(\mathbf{s}_l, \mathbf{s}_m) \wedge \|\mathbf{v}_l - \mathbf{v}_m\|_2 \geq \|\texttt{proj}_{\mathbf{p}}(\mathbf{s}_l) - \texttt{proj}_{\mathbf{p}}(\mathbf{s}_m)\|_2 / t_{\text{horizon}}$	if vessels l and m are on a collision course
change_course	$\mathbf{s}_l, \mathcal{T}_l, t_{\text{start}}, \Delta_{\text{course}}$	$\lvert \sum_{t_i=t_{\text{start}}}^{\texttt{cl}(\mathcal{T}_l, \mathbf{s}_l)} \texttt{proj}_\omega(\texttt{state}(\mathcal{T}_l, t_i))\, \Delta t \rvert \geq \Delta_{\text{course}}$	if course has changed significant since t_{start}
turning_to_starbord	$\mathbf{s}_l, \mathcal{T}_l, t_{\text{start}}$	$\text{mod}\big(\texttt{proj}_\theta(\texttt{state}(\mathcal{T}_l, \texttt{cl}(\mathcal{T}_l, \mathbf{s}_l))) - \text{proj}_\theta(\texttt{state}(\mathcal{T}_l, t_{\text{start}})), 2\pi\big) \in (\pi, 2\pi)$	if course has changed to starboard since t_{start}
overtake	$\mathbf{s}_l, \mathbf{s}_m, t^{\text{check}}_{\text{horizon}}$	$\text{collision\_possible}(\mathbf{s}_l, \mathbf{s}_m, t^{\text{check}}_{\text{horizon}}) \wedge \text{in\_behind\_sector}(\mathbf{s}_m, \mathbf{s}_l) \wedge \text{drives\_faster}(\mathbf{s}_l, \mathbf{s}_m) \wedge \neg\text{orientation\_delta}(\mathbf{s}_l, \mathbf{s}_m, 67.5°, 0)$	give-way vessel of overtaking encounter situation
maneuver_overtake	$\mathbf{s}_l, \mathbf{s}_m, \mathcal{T}_l, t^{\text{check}}_{\text{horizon}}, \Delta_{\text{large\_turn}}$	$\text{change\_course}(\mathbf{s}_l, \mathcal{T}_n, \mathbf{t}_s(\text{overtake}), \Delta_{\text{large\_turn}})$	correct maneuver of give-way vessel in overtaking encounter situation
head_on	$\mathbf{s}_l, \mathbf{s}_m, t^{\text{check}}_{\text{horizon}}, \Delta_{\text{head-on}}$	$\text{collision\_possible}(\mathbf{s}_l, \mathbf{s}_m, t^{\text{check}}_{\text{horizon}}) \wedge \text{in\_front\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge \neg\text{orientation\_delta}(\mathbf{s}_l, \mathbf{s}_m, \Delta_{\text{head-on}}, \pi)$	give-way vessel of head-on encounter situation
maneuver_head_on	$\mathbf{s}_l, \mathbf{s}_m, \mathcal{T}_l, t^{\text{check}}_{\text{horizon}}, \Delta_{\text{large\_turn}}, \Delta_{\text{head-on}}$	$\text{change\_course}(\mathbf{s}_l, \mathcal{T}_n, \mathbf{t}_s(\text{head\_on}), \Delta_{\text{large\_turn}}) \wedge \text{turning\_to\_starboard}(\mathbf{s}_l, \mathcal{T}_n, \mathbf{t}_s(\text{head\_on}))$	correct maneuver of give-way vessel in head-on encounter situation
crossing	$\mathbf{s}_l, \mathbf{s}_m, t^{\text{check}}_{\text{horizon}}, \Delta_{\text{head-on}}$	$\text{collision\_possible}(\mathbf{s}_l, \mathbf{s}_m, t^{\text{check}}_{\text{horizon}}) \wedge \text{in\_right\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge \text{orientation\_towards\_left}(\mathbf{s}_l, \mathbf{s}_m, \Delta_{\text{head-on}})$	give-way vessel of crossing encounter situation
maneuver_crossing	$\mathbf{s}_l, \mathbf{s}_m, \mathcal{T}_l, t^{\text{check}}_{\text{horizon}}, \Delta_{\text{large\_turn}}, \Delta_{\text{head-on}}$	$\text{change\_course}(\mathbf{s}_l, \mathcal{T}_n, \mathbf{t}_s(\text{crossing}), \Delta_{\text{large\_turn}}) \wedge \text{turning\_to\_starboard}(\mathbf{s}_l, \mathcal{T}_n, \mathbf{t}_s(\text{crossing}))$	correct maneuver of give-way vessel in crossing encounter situation
keep	$\mathbf{s}_l, \mathbf{s}_m, t^{\text{check}}_{\text{horizon}}, \Delta_{\text{head-on}}$	$\big(\text{collision\_possible}(\mathbf{s}_l, \mathbf{s}_m, t^{\text{check}}_{\text{horizon}}) \wedge \text{in\_left\_sector}(\mathbf{s}_l, \mathbf{s}_m) \wedge \text{orientation\_towards\_right}(\mathbf{s}_l, \mathbf{s}_m, \Delta_{\text{head-on}})\big) \vee \text{overtake}(\mathbf{s}_m, \mathbf{s}_l, t^{\text{check}}_{\text{horizon}})$	stand-on vessel
no_turning	$\mathbf{s}_l, \mathcal{T}_l, \Delta_{\text{no\_turn}}$	$\neg\text{change\_course}(x_n, \mathcal{T}_n, \mathbf{t}_s(\text{keep}), \Delta_{\text{no\_turn}})$	correct stand-on maneuver

Reuse Requests

This FAQ is an attempt to collect answers to your common questions surrounding reusing content from arXiv in your materials.

Can I reuse figures from an arXiv paper?

The short answer is "it depends". More specifically: - If the license applied to the work allows for remixing or reuse with citation, then yes. - If not, then the version is assigned one of the arXiv perpetual non-exclusive licenses, and you will need to contact the submitter or copyright holder (if published) to determine applicable permissions.

Do I need arXiv's permission to repost the full text?

Note: All e-prints submitted to arXiv are subject to copyright protections. arXiv is not the copyright holder on any of the e-prints in our corpus.

In some cases, submitters have provided permission in advance by submitting their e-print under a permissive Creative Commons license. The overwhelming majority of e-prints are submitted using the arXiv perpetual non-exclusive license, which does not grant further reuse permissions directly. In these cases you will need to contact the author directly with your request.

How can I determine what license the version was assigned?

All arXiv abstract pages indicate an assigned license underneath the "Download:" options.

The link may appear as just the text `(license)`, such as at arXiv:2201.14176. Articles between 1991 and 2003 have an assumed license. These are functionally equivalent to the arXiv non-exclusive license.

If the license applied by the submitter is one of the Creative Commons licenses, then a "CC" logo will appear, such as at arXiv:2201.04182.

I want to include a paper of mine from arXiv in my thesis, do I need specific permission?

If you are the copyright holder of the work, you do not need arXiv's permission to reuse the full text.

I want to include a paper of mine from arXiv in an institutional repository, do I need permission?

You do not need arXiv's permission to deposit arXiv's version of *your* work into an institutional repository. For all other institutional repository cases, see our help page on institutional repositories.

Can I harvest the full text of works?

Plase see our bulk data help page, and the API Terms of Use for specific options. Note that the license for the full text is not a part of the current search API schema. The license is, however, provided within arXiv's output from the OAI-PMH in either `arXiv` or `arXivRaw` formats.