# Realizable Continuous-Space Shields for Safe Reinforcement Learning

**Kyungmin Kim**[1,*], **Davide Corsi**[1,*], **Andoni Rodríguez**[2,3,*],
**JB Lanier**[1], **Benjami Parellada**[4], **Pierre Baldi**[1], **César Sánchez**[2], **Roy Fox**[1]
[1] *University of California, Irvine, USA*
[2] *IMDEA Software Institute, Madrid, Spain*
[3] *Universidad Politécnica de Madrid, Madrid, Spain*
[4] *Universitat Politècnica de Catalunya, Barcelona, Spain*

**Editors:** N. Ozay, L. Balzano, D. Panagou, A. Abate

## Abstract

While Deep Reinforcement Learning (DRL) has achieved remarkable success across various domains, it remains vulnerable to occasional catastrophic failures without additional safeguards. An effective solution to prevent these failures is to use a shield that validates and adjusts the agent's actions to ensure compliance with a provided set of safety specifications. For real-world robotic domains, it is essential to define safety specifications over continuous state and action spaces to accurately account for system dynamics and compute new actions that minimally deviate from the agent's original decision. In this paper, we present the first shielding approach specifically designed to ensure the satisfaction of safety requirements in continuous state and action spaces, making it suitable for practical robotic applications. Our method builds upon `realizability`, an essential property that confirms the shield will **always** be able to generate a safe action for *any* state in the environment. We formally prove that `realizability` can be verified for stateful shields, enabling the incorporation of non-Markovian safety requirements, such as loop avoidance. Finally, we demonstrate the effectiveness of our approach in ensuring safety without compromising the policy's success rate by applying it to a navigation problem and a multi-agent particle environment[1].

**Keywords:** Shielding, Reinforcement Learning, Safety, Robotics

## 1. Introduction

Deep Reinforcement Learning (DRL) has achieved impressive results in a wide range of fields, from mastering complex games like Go (Silver et al., 2016) and Dota 2 (Berner et al., 2019) to real-world applications in healthcare (Pore et al., 2021), autonomous driving (Tai et al., 2017), and robotics (Aractingi et al., 2023). However, even advanced DRL algorithms (Schulman et al., 2017) face considerable challenges when analyzed on specific corner cases, where they persist in demonstrating a proclivity to commit critical mistakes (Corsi et al., 2024a; Szegedy et al., 2013). Such limitations present a threat to the reliability of DRL systems, particularly when deployed in safety-critical applications, where even a single failure can have potentially catastrophic consequences (Srinivasan et al., 2020; Marvi and Kiumarsi, 2021; Katz et al., 2019; Corsi et al., 2021).

Traditional techniques to address safety concerns aim to embed this aspect as part of the learning process; some examples include reward shaping (Tessler et al., 2018), constrained reinforcement learning (Achiam et al., 2017; Ray et al., 2019), and adversarial training (Pinto et al., 2017). While these approaches can significantly enhance the overall reliability of the policy, their guarantees are

[*] These authors contributed equally to the work.
1. The full version of the paper, including the appendix, is available at https://arxiv.org/abs/2410.02038.

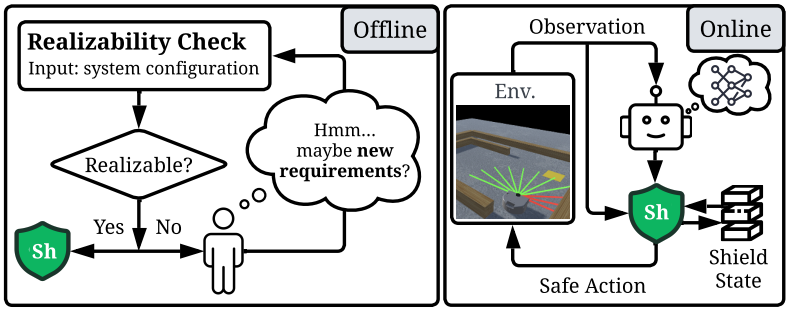


Figure 1: In the offline process, the realizability check verifies whether the system is realizable given the system configuration—i.e., the environment dynamics, safety requirements, and input domain. If not realizable, the system must be modified. Once proved realizable (i.e., a `Proper Shield`), the shield can be safely deployed in the environment for the online process.

typically empirical and their benefits are provided only in expectation (Srinivasan et al., 2020; He et al., 2023). Although these may be sufficient for many problems, they cannot guarantee the safety of a system, limiting their applicability in highly safety-critical contexts.

To face this limitation, a promising family of approaches provides formal safety guarantees through the adoption of an external component, commonly referred to as a shield (Garcıa and Fernández, 2015; Corsi et al., 2024b). A shield acts as a protective wrapper over the agent to ensure that its actions remain within safe boundaries, effectively preventing it from making dangerous or undesired decisions (Alshiekh et al., 2018). However, most shielding techniques in the literature face a fundamental challenge: there are states where no action satisfies all safety criteria simultaneously. This issue is particularly critical in robotics. When an agent encounters a scenario where no safe action is available from the shield, it is left uncertain about how to proceed, as a default safe action is not always feasible in complex and dynamic environments. Therefore, it is essential to develop a shield that can always provide an action satisfying the given specifications for *any* state of the system, which we define as a `Proper Shield`.

In the logic community, ensuring this property is equivalent to guaranteeing the realizability of the shield. Alshiekh et al. (2018) proposed a method to design a proper shield using Linear Temporal Logic (LTL). Their approach uses the synthesis of the shield as a formal tool to automatically guarantee the realizability of the system. However, this method is limited to discrete states and actions, as the LTL synthesizer is not designed to handle continuous spaces. This constraint is overly restrictive for real-world reinforcement learning applications such as robotics where the state and action spaces are often continuous, high dimensional and nonlinear. More in general, despite its necessity and importance, little has been explored regarding continuous-space proper shields in robotics. Compared to discrete spaces, ensuring a proper shield in continuous spaces is significantly more challenging, as theoretically demonstrated by Rodríguez et al. (2025).

In this work, we extend a proper shield to continuous spaces, addressing a gap in existing methods. Our approach builds on the theoretical foundations provided by Rodríguez and Sánchez (2023), which solve the realizability problem for specific fragments of **LTLt**, an extension of LTL that allows to include real values in the specification. While their work focuses purely on logical formulations with no direct connection to sequential decision-making, we adapt these ideas to develop a practical framework for generating a proper shield for continuous control tasks. As shown in Fig. 1, realizability checks are part of an offline procedure conducted prior to deployment. Crucially, work-

ing in continuous spaces allows us to encode a more accurate, model-based representation of the physical world within the specifications. Additionally, working with real values allows us to integrate optimization techniques to ensure that the shield not only returns safe actions but also provides decisions closely aligned with those proposed by the agent. This ensures that the agent's behavior remains both effective and optimal while adhering to strict safety constraints.

Additionally, we formally prove that realizability can be verified for shields with non-Markovian safety requirements. While the realizability of continuous non-Markovian requirements is typically undecidable, we address this challenge by introducing a so-called *anticipation* fragment of LTLt that eliminates operators requiring infinite memory, assuming deterministic dynamics. This fragment is expressive enough to capture relevant safety requirements under assumptions that are common in practical scenarios. For example, in our case study, we encode a rule to avoid loops within a specified time window (e.g., do not visit the same region for $i$ timesteps) to enhance success rates.

We demonstrate the effectiveness of our shielding approach in a mapless navigation domain, with additional experiments in a particle world multi-agent environment. In the mapless navigation domain, the shield effectively protects the agent from obstacles in an unknown environment, highlighting the shield's ability to enforce safety without relying on pre-defined maps or global knowledge. We guarantee the realizability of safety requirements, ensuring that the agent has always at least one safe action available. Furthermore, with the integration of optimization techniques and non-Markovian requirements, we achieve the safety of the system with a minimal impact on the policy performance. In the particle-world multi-agent environment, our shield prevents unsafe interactions among multiple agents in a shared, continuous space.

In summary, the contributions of this paper are as follows: (1) We introduce continuous-space proper shields for safe control. A detailed comparison with alternative shielding techniques is provided in Tab. 4 in the Appendix. (2) We propose a shield that enables formal realizability proofs for non-Markovian requirements by introducing the 'anticipation fragment' of LTLt, a decidable subset designed for stateful shields. This fragment effectively captures safety requirements under reasonable assumptions for practical use. (3) We demonstrate the merits of our method on robotic applications, including a mapless navigation domain and a particle-world multi-agent environment.

## 2. Preliminaries

We model interaction with an environment as a Partially Observable Markov Decision Process (POMDP), defined by a tuple $(X, A, O, P, R, \Omega, \gamma)$, with a state space of $X$, action space $A$, and observation space $O$. $P : X \times A \to \Delta(X)$ represents the state transition function, where $\Delta(\cdot)$ denotes the set of all probability distributions over the given set. $R : X \times A \to \mathbb{R}$ is the reward function. $\Omega : X \to \Delta(O)$ represents the observation function, and $\gamma \in (0, 1]$ is the discount factor. Our objective is to learn a policy $\pi : O \to \Delta(A)$ that maximizes the expected sum of discounted rewards over time. This is defined as $\mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(x_t, a_t)\right]$, where $x_t$ and $a_t$ are the state and action at time $t$ until the episode horizon $T$.

### 2.1. Concepts in Temporal Logic and Reactive Systems

**LTL.** Linear Temporal Logic (LTL) is a formal system for reasoning about the behavior of discrete-time systems over an infinite timeline. Concretely, it is a modal logic that extends propositional logic with temporal operators that allow the expression of properties about the future evolution of the system. Some key temporal operators in LTL include: (1) $\bigcirc$ (next): The property holds in the next

timstep; and (2) $\Box$ (always): The property holds at all future timsteps. Using these operators, LTL formulas can express a wide range of temporal properties, such as safety ("something bad never happens"). For instance, while in classic propositional logic we can express $v^1 \rightarrow v^2$, in LTL we can also express $\Box(v^1 \rightarrow \bigcirc v^2)$, which means that at every step, if $v^1$ holds, then $v^2$ must hold in the next timestep.

**Satisfiability and Realizability.** Given an LTL formula $\varphi$, the satisfiability problem determines if there is any possible execution that satisfies the specified temporal properties; i.e., we say $\varphi$ is satisfiable if there is a possible assignment of the variables in $\varphi$ such that $\varphi$ is satisfied. More formally, given variables $v = \{v^0, v^1, ...\}$ in $\varphi$, if $\exists v$ s.t. $\varphi(v)$ holds, then $\varphi$ is satisfiable. A reactive system is a type of system that continuously interacts with its environment by responding to inputs and adapting its behavior accordingly. To ensure such a system can meet the requirements under all possible conditions, we need a property stronger than satisfiability: `realizability`. In `realizability`, the variables of $\varphi$ are divided into an uncontrollable player (i.e., the inputs provided by the environment) and a controllable player (i.e., the outputs provided by the system). Then, a formula $\varphi$ is realizable if for *all* possible valuations of the input variables, the output variables can be assigned so that the $\varphi$ is not violated.

**Realizability for Continuous Domains.** LTL is a powerful tool for reasoning about temporal properties. However, it lacks the expressiveness needed to handle continuous values, which are essential for realistic robotics applications to accurately encode environment dynamics in continuous spaces. To address this, we use Linear Temporal Logic modulo theories (LTLt), an extension of LTL that allows formulas to include variables from a first-order theory $\mathcal{T}$. While this has a precise meaning in formal verification[2], for simplicity in this paper, it means that LTLt enables specifying more complex properties involving both the temporal behavior of a system and real values data. A recent development by Rodríguez and Sánchez (2024b) solved the problem of realizability for certain fragments of LTLt. This makes it feasible to create realizable continuous-space shields using LTLt to ensure such expressive properties; for example, $(v^1 > 2.5) \rightarrow \bigcirc(v^2 > v^1)$. Note that another powerful operator in LTLt is $\triangleleft v$, which, given $v$, allows access to the value of $v$ in the previous timestep. However, this expressivity comes at the expense of the LTLt not being decidable anymore (i.e., realizability checking procedures are not guaranteed to terminate).

## 3. Realizable Continuous Shields with Non-Markovian Requirements

A deep reinforcement learning shield is an external component that works on top of a policy to ensure compliance with a set of safety requirements $\varphi$. A `Proper Shield` is a special case of a shield that can provide safe alternatives to unsafe actions for *all* states in the system.

In this paper, to address the complexity of real-world robotic problems, our shield is designed to be *stateful*, enabling it to handle non-Markovian requirements. The shield maintains an internal state $h \in H$, which is updated after each time step and, by construction, represents only safe states (i.e., states where $\varphi$ has not been violated), as any violation in the past would contradict the safety specifications. Depending on the task, $h$ can range from being empty (for purely Markovian requirements) to capturing the full state-action history. In all cases, $h$ represents a subset of the complete history relevant for enforcing safety. Formally, we define a `Proper Shield` as follows:

2. We invite the reader to read the supplementary material (Appendix E) for further insights.

**Definition 1 (`Proper Shield`)**

*Let $A$ be the action space, $O$ the observation space, $H$ the set of shield states, and $\varphi : A \times O \times H \to \{\texttt{true}, \texttt{false}\}$ which encodes a set of safety specifications. We define a `Proper Shield` as a function $\zeta : A \times O \times H \to A$ such that for all $a \in A$, $o \in O$, and $h \in H$, the safety condition $\varphi(\zeta(a, o, h), o, h) = \texttt{true}$ holds.*

Intuitively, a `Proper Shield` is a function $\zeta$ designed to always return an action $a$ such that $\varphi$ holds when starting from the initial state. The shield $\zeta$ is combined with an external agent $\pi$ (in our case, an RL agent) to ensure that the composition $\zeta(\pi(o), o, h)$ never violates $\varphi$. At each time step, given the shield's internal state $h$: (1) $\pi$ receives observation inputs $o$ from the environment and produces an action $a$; (2) we check whether $\varphi(a, o, h)$ holds; and (3) if $\varphi(a, o, h)$ holds, the shield does not intervene, but if there is a violation, it overrides $a$ with a corrected action $\hat{a} = \zeta(a, o, h)$ such that $\varphi(\hat{a}, o, h)$ now holds for that step, as described in Fig. 1 (right).

Ensuring that $\varphi$ is realizable is essential, as unrealizability could lead to situations where no safe action is possible for a given state. In such cases, computing $\hat{a}$ is not possible because the formula is unsatisfiable (`UNSAT`). In our framework (see Fig. 1, left), we check the realizability of $\varphi$ as part of an offline process prior to deployment, treating the observation $o$ and shield state $h$ as uncontrollable variables, while considering the action $a$ as a controllable variable.

Alshiekh et al. (2018) showed that a proper shield can be constructed using LTL, but their approach is limited to discrete state and action spaces. This restricts its applicability to robotics tasks and fails to capture accurate environment dynamics. In contrast, our shield leverages the more expressive specification language, LTLt, enabling the construction of a continuous-space proper shield that incorporates accurate dynamics directly into the specification. Moreover, given a continuous space, we can minimize a distance metric between the correction $\hat{a}$ and the original candidate output $a$. This allows us to optimize the shield's safe-action correction by returning $\hat{a}$ such that it is the closest safe value to the agent's proposed action $a$.

### 3.1. Non-Markovian Encoding in LTLt

By exploiting recent advancements in LTLt synthesis as discussed in the previous sections, it becomes possible to encode Markovian properties that can be synthesized to create a proper shield:

**Example 1** *Consider a simple agent that moves along a one-dimensional line, where its state is represented by a single variable $x \in (0, 1]$. The action space consists of a single action $a \in [-1, 1]$, which specifies both the direction and magnitude of the step. The dynamics are given by $x_{t+1} = x_t + a_t$. Suppose we wish to enforce a simple safety requirement that ensures the agent never moves to a position lower than zero. This requirement can be encoded as follows:*

$$\square([0 < x \leq 1] \to \neg[a < -x])$$

Note that this requirement is highly simplified; the challenge for the solver arises from the exponential growth in the number and complexity of such requirements. Nevertheless, properties of this form are often sufficient to encode essential safety requirements, such as collision avoidance. However, for real-world problems, we often need to incorporate requirements that extend beyond simple input-output relationships, involving multiple steps in the environment and sequences of the agent's decisions. In such cases, it is necessary to encode non-Markovian requirements (or multistep properties) in a decidable fragment of LTLt, that guarantees an automatic realizability check. To better explain our approach we rely on a running example:

**Example 2** *Let us revisit the agent from Example 1. We may now want to ensure that, once the agent visits a specific region of the state space, it will never return to the same position within a finite time horizon $i$. Specifically, we subdivide the state space into a finite number of intervals $r \in R$, where $r = [r_l, r_u]$. This requirement requires a non-Markovian constraint as it involves consideration of multiple steps in the environment. We can formally encode the property using an LTLt formula $\varphi = A \rightarrow G$, where $A$ represents the assumptions, and $G$ specifies the guarantees the system must uphold under those assumptions:*

$$[A] \quad \square[x = \triangleleft a + \triangleleft x] \qquad\qquad [G] \quad \square[\bigwedge_{r \in R} [x \in r] \rightarrow \square_{[1,i]} \neg [x \in r]]$$

*where (i) $A$ (left) is an environment assumption that describes the dynamics of the **scenario** and allows to compute the next state given the current state and the action. Specifically, the operator $\triangleleft x$ represents a specific value that the variable $x$ has taken in the previous timestep; and (ii) $G$ (right) are the guarantees by the system, in this case, stating that if $x \in r$ in a given timestep, then $x \notin r$ throughout $i$ timesteps; i.e., the region $r$ will not be re-visited for $i$ timesteps.*

**Example 3** *Note that using the operator $\triangleleft$ is a very natural way to define A. However, this means that $\varphi$ is encoded in a fragment of LTLt that is not decidable, because $\triangleleft$ introduces unbounded memory to the program, as it allows an uncontrolled infinite nesting of the operator which at the same time may end up in an infinite recursion. Thus, realizability checking is not guaranteed to terminate and we need to rewrite $A$ in a fragment of LTL that does not include $\triangleleft$, specifically:*

$$[\tilde{A}] \quad \square[\bigwedge_{r \in R} [(x + a) \in r] \rightarrow \bigcirc [x \in r]],$$

*which means that when the result of applying an action to a state falls within the region $r$, the state itself must belong to $r$ in the next time step. This simple reformulation allows us to eliminate the operator $\triangleleft$, making realizability of the formula decidable while still preserving dynamic prediction in the subset of information relevant to the requirement $G$.*

An essential observation that makes this transformation possible is that we are (i) assuming to have access to a safety-relevant subset of the robot's dynamic, often called *safety-dynamics* in the literature (Yang et al., 2023); and (ii) we encoded the requirement for a finite number of regions $r \in R$ and a finite horizon $i$. In our example, we can *anticipate* the constraints-relevant components of the next state given the action and the current state of the system. Thus, we can *anticipate* any unsafe action of the agent, and potentially override it (whenever the specification realizable).

**Theorem 2** *If $\varphi$ is an LTLt formula where the subset of the next values of the environment variables that appear in $\varphi$ are isolated and can be fully determined from the current environment and system values, then $\varphi$ can be rewritten without the use of $\triangleleft$ (proof in Appendix D).*

**Definition 3 (Anticipation Fragment)** *The class of LTLt formula that can be translated to not using $\triangleleft$ is called the anticipation fragment of LTLt.*

**Corollary 4** *If $\varphi$ belongs to the anticipation fragment, then the realizability of $\varphi$ is decidable.*

## 4. Case Study: Mapless Navigation with Reinforcement Learning

To ground and demonstrate our theoretical framework in a realistic robotics scenario, we consider a *mapless indoor navigation* task that couples continuous control with partial observability, providing a challenging setting for provably safe decision making.

In this task, an agent must reach a target position while avoiding static obstacles in a previously unseen rectangular room. Each episode randomizes the agent's start pose, goal location, and placement of rectangular obstacles. Observations combine (i) a 23-ray lidar scan (360° field of view) that returns normalized distances to the nearest obstacles with (ii) the robot's planar position, heading, and goal coordinates. Actions are continuous linear and angular velocities. We train policies for 500 episodes with PPO (Schulman et al., 2017), which has proven effective on similar navigation benchmarks (Amir et al., 2023). Appendix B provides additional details on the environment setup, reward function, and hyperparameters. Fig. 2 plots the resulting learning curves. Critically, a trained unshielded policy succeeds in 98.1% of trials but still collides in 1.2% of episodes, a small but unacceptable failure rate that motivates the shielding specification introduced next.

### 4.1. Continuous Shield for Safe Navigation

In this section, we delineate the safety requirements we aim to enforce and how we specify them using LTLt to ensure robust and reliable performance. As discussed in the introduction, this section assumes access to the full system dynamics. However, in practice, this assumption can often be relaxed by relying solely on a partial model that captures the safety-relevant aspects of the dynamics.

**Markovian Requirements for Collision Avoidance.** In navigation tasks, ensuring collision avoidance is crucial under all circumstances. To achieve this, we encode specifications by leveraging the robot's dynamics, which our framework supports for continuous spaces. Given an action $a = [a^0, a^1]$, the robot first rotates by $a^1$ and then translates by $a^0$, where $a^0 \in [-L^0, L^0]$ and $a^1 \in [-L^1, L^1]$ and $L^N$ is a step size. Positive $a^0$ indicates forward motion and positive $a^1$ means a right turn. As the next pose of the robot is predictable through the dynamics, we can design requirements that strictly avoid collisions in the next timestep. Since robot rotation and translation are performed sequentially in the environment dynamics, requirements for each can be designed separately. Specifically, the red area in Fig. 3 (left) shows the robot's trajectory when it makes a maximum right turn of $L^1$. To conservatively prevent collisions from right turns, we prohibit turning right if any lidar value $l^i$ is below a certain threshold $T^i$. These thresholds are precomputed based on the robot's dynamics and maximum step size $L^1$ and they vary for each lidar (highlighted in blue). Formally, the specification for a right turn is: $\exists i$ s.t. $(l^i \leq T^i) \rightarrow a^1 \leq 0$. The same principle applies to left turns. For translation, we limit the maximum distance that we can translate based on the minimum distance of potential obstacles that any lidars sense in the direction of travel. Consider the $i^{th}$ lidar with value $l^i$ at an angular position $\theta^i$ relative to the robot's horizontal line, measured clockwise from the left. Note that $\theta^i$ is known by construction as part of the system. After the robot rotates by $a^1$, the angular position w.r.t. the new horizontal line becomes $\hat{\theta^i} = \theta^i - a^1$. If this lidar reveals that the obstacle is in the path of forward translation, the translation $a^0$ cannot exceed the distance to the obstacle; otherwise, a collision will occur. Formally, the requirement for forward translation w.r.t. the $i^{th}$ lidar is: $l^i|\cos\theta^{i'}| \leq \frac{W}{2} \rightarrow a^0 + H^f < l^i|\sin\theta^{i'}|$, where $W$ is the robot's width and $H^f$ is the lidar's forward offset. For forward translation, this requirement must

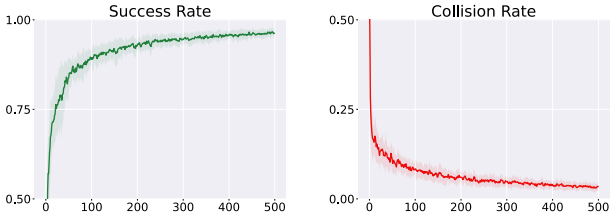


Figure 2: Average success rate and collision rate obtained during the DRL training process for 500 episodes (x-axis) without any shield applied (averaged over 5 different random seeds).

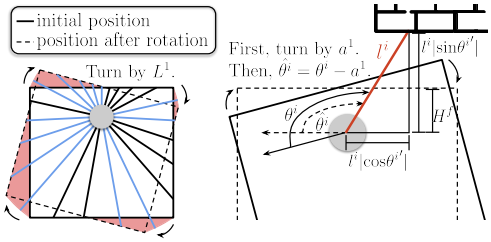


Figure 3: Description of collision avoidance requirements that include the dynamic of the robot as part of the formula.

hold for all lidars in front of the robot. Similar requirements apply for backward translation, with different signs, offsets, and considering lidars at the back.

**Non-Markovian Requirements for Loop Avoidance.** While safety remains paramount in autonomous systems, ensuring collision avoidance, etc. alone can lead to overly conservative behavior, often characterized by repetitive or looping actions. The agent then can be particularly inefficient, wasting time and energy. To address this issue, we introduce a set of important non-Markovian requirements. These prohibit repeating the same action in a state for a certain time window. This encourages the agent to explore new actions, potentially helping it escape from being stuck. We encode these requirements using a queue constructed from the shield state of length $L^Q$. Each queue element consists of the robot's pose (x/y position and rotation) and actions, represented as $(x, y, r, a^0, a^1)$. To effectively check for repeated tuples of continuous values, we quantize the pose and action space into grids of $G^P$ and $G^A$ cells, respectively. In this way, the shield does not allow an action $(a^0, a^1)$ in a state $(x, y, r)$ if any of the previous tuples in the queue falls into the same cell. Although these requirements incorporate stateful memory as an additional input, making them non-Markovian, our approach enables the realizability check with such inputs.

### 4.2. Realizability and Shielding Implementation

We specify the above requirements in LTLt and provide them to the offline `realizability` check process to guarantee that these requirements are always satisfiable. Our `realizability` check process follows the implementation established in Rodríguez and Sánchez (2023); Rodríguez and Sánchez (2024a,b); Rodríguez et al. (2024). After `realizability` for our requirements is guaranteed, at test-time, we deploy our agent with an online shield to provide safe alternative actions when the agent suggests an action that violates our constraints. The overall algorithm for deploying the online shield is presented in Appendix C.

| | | $G^A$ | | |
|---|---|---|---|---|
| | | 3 | 5 | 30 |
| $L^Q$ | 1 | 31 | 0 | 0 |
| | 13 | 288 | 6 | 0 |
| | 100 | 336 | 30 | 0 |

Table 1: Number of episodes where the shield returns `unsat` out of 500 tests, with realizable configurations in green and others in red. `realizability` checking identifies unsatisfiable specifications that empirical evaluation might overlook.

| | No Shield | | Collision Shield | | Collision & Loop Shield | | Optimizer | |
|---|---|---|---|---|---|---|---|---|
| | Success | Collision | Success | Collision | Success | Collision | Success | Collision |
| *Expert A* | $0.87 \pm 0.05$ | $0.03 \pm 0.03$ | $0.87 \pm 0.04$ | $0.00 \pm 0.00$ | $0.90 \pm 0.02$ | $0.00 \pm 0.00$ | $0.92 \pm 0.02$ | $0.00 \pm 0.00$ |
| *Expert B* | $0.88 \pm 0.03$ | $0.03 \pm 0.02$ | $0.87 \pm 0.03$ | $0.00 \pm 0.00$ | $0.90 \pm 0.02$ | $0.00 \pm 0.00$ | $0.90 \pm 0.02$ | $0.00 \pm 0.00$ |
| *Moderate A* | $0.77 \pm 0.04$ | $0.01 \pm 0.01$ | $0.76 \pm 0.04$ | $0.00 \pm 0.00$ | $0.79 \pm 0.04$ | $0.00 \pm 0.00$ | $0.80 \pm 0.05$ | $0.00 \pm 0.00$ |
| *Moderate B* | $0.85 \pm 0.02$ | $0.04 \pm 0.02$ | $0.85 \pm 0.02$ | $0.00 \pm 0.00$ | $0.87 \pm 0.02$ | $0.00 \pm 0.00$ | $0.87 \pm 0.02$ | $0.00 \pm 0.00$ |
| *Unsafe* | $0.22 \pm 0.03$ | $0.78 \pm 0.03$ | $0.40 \pm 0.05$ | $0.00 \pm 0.00$ | $0.47 \pm 0.02$ | $0.01^* \pm 0.01$ | $0.69 \pm 0.03$ | $0.01^* \pm 0.00$ |

Table 2: Comparison of success rate and collision rate with different settings of the shield on the mapless navigation environment. Mean scores over 5 seeds (100 runs per seed) with standard deviations are presented. *These collisions arose due to the partial observability in the environment and can be prevented by increasing the number of 1D lidar sensors.

## 5. Experimental Results

**Realizability Check.** A crucial aspect of our shield is that it guarantees a safe action in *any* state through a `realizability` check. Ensuring `realizability` by hand is challenging with complex requirements like ours, and statistics-driven safety checks are also unreliable. In Table 1, we evaluate a shielded agent for 500 episodes with multiple different queue lengths ($L^Q$) and action grid sizes ($G^A$) without performing a `realizability` check beforehand. We then tally the number of times we encountered a `unsat` output from the shield due to no safe action being available. Finally, we verify the `realizability` of each shield specification and highlight the realizable configuration in green. We see that performing a realizability check can warn us about potentially unsatisfiable and thus unsafe specifications even when empirical evaluations do not indicate a problem. For instance, no `unsat` situations occurred for our shield with evaluating $(L^Q, G^A) = (100, 30)$, however, our `realizability` check reveals that there still are potential situations where $(100, 30)$ is not satisfiable. For subsequent experiments, we use one of the verified realizable configurations, $(L^Q, G^A) = (30, 13)$.

**Online Shielding.** In Table 2, we report the analysis of 5 different RL agents with different capabilities. *Expert A* and *Expert B* are fully trained PPO models. *Moderate A* and *Moderate B* are checkpoints collected during an intermediate phase of learning and are more prone to making mistakes and failing the task. Finally, *Unsafe* is *Expert A* deployed without access to lidar data, simulating a dangerous agent oriented toward unsafe behavior and collisions. We believe that the addition of unsafe models is particularly valuable for our analysis to show that the shield can be effective with any model and guarantee safety independently of the input policy.

The first column shows the success rate and collision rate when each policy is executed without any external shielding component. Although the success rates for the well-trained agents could be considered satisfactory, all agents made some collisions with an obstacle. The second column shows each policy's performance with the collision avoidance shield added, clearly demonstrating the effectiveness of our shield, which reduces the number of collisions to zero. However, the shield alone leads to overly conservative behavior, resulting in the unsafe trajectories being converted into timeouts with oscillating behavior rather than successful episodes. Crucially, the third column shows the complete version of our shield (i.e., with collision and loop avoidance requirements), showing that the satisfaction of both properties allows the agent to recover from the conservative behavior and increase the success rate while ensuring the safety of the policy. Finally, the last column shows a small additional impact of optimizing the shield's choice of safe action. Since

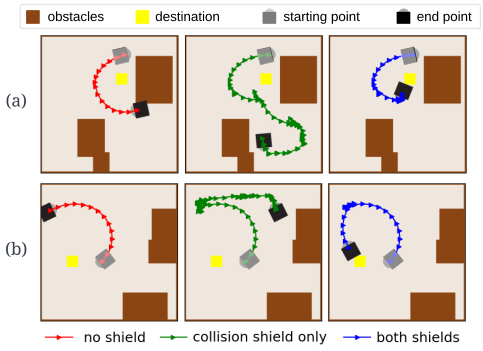


Figure 4: An unshielded agent collides with obstacles (brown) and avoids them when a collision shield is applied. By introducing non-Markovian requirements, the agent also avoids oscillations and reaches its target destination (yellow).

| | No Shield | | Safety Shield | |
|---|---|---|---|---|
| | Success | Collision | Success | Collision |
| *Model A* | $0.56 \pm 0.05$ | $0.44 \pm 0.05$ | $0.93 \pm 0.02$ | $0.00 \pm 0.00$ |
| *Model B* | $0.53 \pm 0.07$ | $0.47 \pm 0.07$ | $0.95 \pm 0.01$ | $0.00 \pm 0.00$ |
| *Model C* | $0.64 \pm 0.03$ | $0.36 \pm 0.03$ | $0.96 \pm 0.00$ | $0.00 \pm 0.00$ |
| *Model D* | $0.66 \pm 0.02$ | $0.34 \pm 0.02$ | $0.97 \pm 0.01$ | $0.00 \pm 0.00$ |

Table 3: Results on the `Particle World` environment. The *No Shield* agent operates without awareness of the safety requirements and, as a result, is not trained to avoid collisions. In this setting, a single collision is sufficient to terminate the episode unsuccessfully. Due to the full observability of the environment, it is possible to exploit the system's dynamics to account for all potential sources of collision, thereby reducing the collision rate to zero.

we operate in a continuous space, our shield can employ an optimizer to return a safe action that minimizes distance to the original policy output. In this navigation domain, we minimize $|\hat{a}^1 - a^1|$ where $a^1$ is the agent's original angular velocity and $\hat{a}^1$ is the angular velocity of the shielded action. Fig. 4 illustrates a policy's behavior both with and without a shield employed for collisions and loops. A detailed discussion on the broader relationship between shielding and other safety-oriented reinforcement learning methods is provided in Appendix F.

**Particle World Experiments** To further demonstrate the generalizability and robustness of our shielding approach, we also experiment in a multi-agent `Particle World` environment (Mordatch and Abbeel, 2017). Four agents are tasked with reaching target positions on the opposite side of the map while maintaining a safe distance from one another (a screenshot of this environment can be found in Fig. 5 of the Appendix). The primary safety requirement in this environment is to ensure that the agents always keep a specified minimum distance between each other, illustrated as circles around the agents. The results, summarized in Table 3, demonstrate that our safety shield can be seamlessly applied to this new environment with a continuous state and action space. Notably, our shielding technique in `Particle World` successfully eliminates all violations of the safety requirements, effectively preventing any unsafe actions. This highlights the versatility and effectiveness of our shielding approach, showcasing its potential for broader applications across various continuous-space environments.

## 6. Conclusion

In this paper, we introduce the concept of `Proper Shield` and propose a novel shielding approach for continuous action spaces with guaranteed realizability. Our results show that this technique effectively ensures the safety of a reinforcement learning agent in a navigation task. A key contribution is the theoretical and empirical integration of non-Markovian requirements into the shielding process, mitigating overly conservative behavior and enabling recovery from loops or repeated actions while maintaining efficient goal-directed progress. Limitations and future directions are discussed in Appendix G.

## Acknowledgments

Authors Kim and Lanier were supported by a Hasso Plattner Foundation Fellowship. This work was funded in part by the National Science Foundation (Award #2321786).

## References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, 2017.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. Ieee, 2019.

Guy Amir, Davide Corsi, Raz Yerushalmi, Luca Marzari, David Harel, Alessandro Farinelli, and Guy Katz. Verifying learning-based robotic navigation systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2023.

Michel Aractingi, Pierre-Alexandre Léziart, Thomas Flayols, Julien Perez, Tomi Silander, and Philippe Souères. Controlling the solo12 quadruped robot with deep reinforcement learning. *scientific Reports*, 2023.

Clark W. Barrett, Cesare Tinelli, Haniel Barbosa, Aina Niemetz, Mathias Preiner, Andrew Reynolds, and Yoni Zohar. Satisfiability modulo theories: A beginner's tutorial. In *Formal Methods - 26th International Symposium, FM 2024, Milan, Italy, September 9-13, 2024, Proceedings, Part II*, volume 14934 of *Lecture Notes in Computer Science*, pages 571–596. Springer, 2024. doi: 10.1007/978-3-031-71177-0_31. URL https://doi.org/10.1007/978-3-031-71177-0_31.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Aaron R. Bradley and Zohar Manna. *The Calculus of Computation: Decision Procedures with Applications to Verification*. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 3540741127.

Wonhyuk Choi, Bernd Finkbeiner, Ruzica Piskac, and Mark Santolucito. Can reactive synthesis and syntax-guided synthesis be friends? In Ranjit Jhala and Isil Dillig, editors, *43rd ACM SIGPLAN Int'l Conf. on Programming Language Design and Implementation (PLDI 2022)*, pages 229–243. ACM, 2022. doi: 10.1145/3519939.3523429. URL https://doi.org/10.1145/3519939.3523429.

George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition-preliminary report. *SIGSAM Bull.*, 8(3):80–90, 1974. doi: 10.1145/1086837.1086852. URL https://doi.org/10.1145/1086837.1086852.

Davide Corsi, Enrico Marchesini, and Alessandro Farinelli. Formal verification of neural networks for safety-critical tasks in deep reinforcement learning. In *Uncertainty in Artificial Intelligence*, 2021.

Davide Corsi, Raz Yerushalmi, Guy Amir, Alessandro Farinelli, David Harel, and Guy Katz. Constrained reinforcement learning for robotics via scenario-based programming. *arXiv preprint arXiv:2206.09603*, 2022.

Davide Corsi, Guy Amir, Guy Katz, and Alessandro Farinelli. Analyzing adversarial inputs in deep reinforcement learning. *arXiv preprint arXiv:2402.05284*, 2024a.

Davide Corsi, Guy Amir, Andoni Rodriguez, Cesar Sanchez, Guy Katz, and Roy Fox. Verification-guided shielding for deep reinforcement learning. *The 1st Reinforcement Learning Conference (RLC)*, 2024b.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 2015.

Luca Geatti, Alessandro Gianola, Nicola Gigante, and Sarah Winkler. Decidable fragments of ltlf modulo theories (extended version). *CoRR*, abs/2307.16840, 2023. doi: 10.48550/arXiv.2307.16840. URL https://doi.org/10.48550/arXiv.2307.16840.

Shangding Gu, Jakub Grudzien Kuba, Munning Wen, Ruiqing Chen, Ziyan Wang, Zheng Tian, Jun Wang, Alois Knoll, and Yaodong Yang. Multi-agent constrained policy optimisation. *arXiv preprint arXiv:2110.02793*, 2021.

Tairan He, Weiye Zhao, and Changliu Liu. Autocost: Evolving intrinsic cost for zero-violation reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

Andreas Katis, Grigory Fedyukovich, Huajun Guo, Andrew Gacek, John Backes, Arie Gurfinkel, and Michael W. Whalen. Validity-guided synthesis of reactive systems from assume-guarantee contracts. In *Proc. of the 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, (TACAS 2018)*, volume 10806 of *LNCS*, pages 176–193. Springer, 2018. doi: 10.1007/978-3-319-89963-3_10. URL https://doi.org/10.1007/978-3-319-89963-3_10.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification 2017*. Springer, 2017.

Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification 2019*, 2019.

Orna Lichtenstein, Amir Pnueli, and Lenore D. Zuck. The glory of the past. In *Logics of Programs, Conference, Brooklyn College, New York, NY, USA, June 17-19, 1985, Proceedings*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer, 1985. doi: 10.1007/3-540-15648-8_16. URL https://doi.org/10.1007/3-540-15648-8_16.

Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, Mykel J Kochenderfer, et al. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 2021.

Yongshuai Liu, Jiaxin Ding, and Xin Liu. Ipo: Interior-point policy optimization under constraints. In *Proceedings of the AAAI conference on artificial intelligence*, 2020.

Zahra Marvi and Bahare Kiumarsi. Safe reinforcement learning: A control barrier function optimization approach. *International Journal of Robust and Nonlinear Control*, 2021.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

Quan Nguyen and Koushil Sreenath. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. In *2016 American Control Conference (ACC)*, pages 322–328. IEEE, 2016.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International conference on machine learning*, 2017.

Ameya Pore, Davide Corsi, Enrico Marchesini, Diego Dall'Alba, Alicia Casals, Alessandro Farinelli, and Paolo Fiorini. Safe reinforcement learning using formal verification for tissue retraction in autonomous robotic-assisted surgery. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 2019.

Andoni Rodríguez and César Sánchez. Boolean abstractions for realizability modulo theories. In *International Conference on Computer Aided Verification*, pages 305–328. Springer, 2023.

Andoni Rodríguez, Guy Amir, Davide Corsi, César Sánchez, and Guy Katz. Shield synthesis for LTL modulo theories. In *Proc. of the 39th AAAI Conf. on Artificial Intelligence (AAAI 2025)*, pages 15134–15142. AAAI Press, 2025.

Andoni Rodríguez and César Sánchez. Adaptive Reactive Synthesis for LTL and LTLf Modulo Theories. In *Proc. of the 38th AAAI Conf. on Artificial Intelligence (AAAI 2024)*, 2024a.

Andoni Rodríguez and César Sánchez. Realizability modulo theories. *Journal of Logical and Algebraic Methods in Programming*, 2024b.

Andoni Rodríguez, Felipe Gorostiaga, and César Sánchez. Predictable and Performant Reactive Synthesis Modulo Theories via Functional Synthesis. In *Proc. of the 22nd International Symposium on Automated Technology for Verification and Analysis (ATVA 2024)*, 2024.

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

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

T Simão, S Tindemans, and M Spaan. Training and transferring safe policies in reinforcement learning. In *Sl: Adaptive and Learning Agents*, 2022.

Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Lei Tai, Giuseppe Paolo, and Ming Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2017.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2018.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 2021.

Wei Xiao, Calin Belta, and Christos G Cassandras. Adaptive control barrier functions. *IEEE Transactions on Automatic Control*, 67(5):2267–2281, 2021.

Wen-Chi Yang, Giuseppe Marra, Gavin Rens, and Luc De Raedt. Safe reinforcement learning via probabilistic logic shields. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 5739–5749, 2023.

Lijun Zhang, Lin Li, Wei Wei, Huizhong Song, Yaodong Yang, and Jiye Liang. Scalable constrained policy optimization for safe multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 37:138698–138730, 2024.