Shield Decentralization for Safe Multi-Agent Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning safe solutions is an important but challenging problem in multi-agent reinforcement learning (MARL). Shielded reinforcement learning is one approach for preventing agents from choosing unsafe actions. Current shielded reinforcement learning methods for MARL make strong assumptions about communication and full observability. In this work, we extend the formalization of the shielded reinforcement learning problem to a truly decentralized multi-agent setting. We then present an algorithm for decomposition of a centralized shield, allowing shields to be used in such decentralized, communication-free environments. Our results show that agents equipped with decentralized shields perform comparably to agents with centralized shields in several tasks, allowing shielding to be used in decentralized environments for the first time.

1 Introduction

Recently, the advent of deep reinforcement learning has produced solutions to highly complex domains such as Atari [16], and the game of Go [19]. However, the neural networks which power these methods are opaque, and there is a significant risk of unintentional harm when a poorly understood reinforcement learning method is applied to a safety-critical system [2].

One common approach for safe RL is the idea of a shield—a reactive system which monitors a reinforcement learning agent and environment [1]. During each step, the agent proposes an action, the shield evaluates whether this action is safe, and then the shield potentially replaces the proposed action with a known-safe action. This method has been extended to multi-agent environments, but the shield remains centralized [9]. Such methods assume instantaneous communication with a centralized coordination algorithm, which may not be realistic in many environments.

In this work, we first discuss the centralized shield synthesis problem, and outline its proposed solutions. We describe several implicit communication assumptions present in the existing centralized shield definition, and define a novel framework for decentralized shielding. These decentralized shields enable each agent to act independently, without any coordination after shield synthesis.

Our main technical contribution is an algorithm which, using any centralized shield as input, produces a decentralized shield that guarantees safety. We also discuss various properties of input centralized shields, and the effect of these properties on the permissiveness of the output shield.

Lastly, we evaluate the performance of our shield decomposition algorithm on shields from several domains. We first analyze the structure of the shield itself, ensuring that the majority of actions allowed by the centralized shield are still allowed by the decentralized shield. We then train a shielded reinforcement learning agent to solve these tasks, and measure the performance of the learned policy. We find that all agents achieve a comparable reward in the environment, and that, as is to be expected from the theoretical guarantees of our framework, decentralized shielding achieves

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

100% safety. These results show that our shield decentralization approach can, for the first time, allow reinforcement learning methods to be provably safe in decentralized multi-agent settings.

2 Related Work

The concept of shielded reinforcement learning [1] draws from the idea of Runtime Enforcement [5], from the Formal Methods community. In the single-agent case, shielding has been extended to environments with continuous action spaces [7], stochastic or adversarial environments [14], and partially-observable environments [6]. As with all shielding methods, these extensions require a model of the environment, with a varying degree of abstraction.

Other proposed methods specify algorithms for enforcing safety in the presence of communication constraints or delays for pathfinding agents [22], or for general LTL specifications [11], but neither method eliminates communication entirely.

Several pieces of work focus on verifying a neural network or closed-loop reinforcement learning system after training is complete [4, 13]. These methods could potentially succeed in a multi-agent reinforcement learning system if successfully scaled up, but such approaches typically make no guarantees of safety during training.

3 Preliminaries

We first describe a formalization of the centralized shielding problem. Many elements of this framework have been described in prior formulations [1, 5], and extended to the multi-agent setting [9].

Let $\Delta(X)$ be the set of all probability distributions over set X. Let $\mathbb{R}$ denote the set of reals.

An environment with cooperative agents and full observability is often characterized as a *MMDP*:

Definition 1 (MMDP) *A* Multi-agent Markov Decision Process (MMDP) *is a tuple* $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$ *where* $\mathbb{D} = \{1, \ldots, n\}$ *is a set of agents,* $\mathbb{S}$ *is a set of states,* $\mathbb{A} = \prod_{i \in \mathbb{D}} \mathbb{A}_i$ *is a finite joint action space, factorizable into* n *individual action spaces,* $\mathcal{E} : \mathbb{S} \to \mathcal{P}(\mathbb{A})$ *represents the set of actions available to the agents at a given state,* $T : \mathbb{S} \times \mathbb{A} \to \Delta(\mathbb{S})$ *is the transition probability distribution function,* $\gamma \in [0, 1]$ *represents the discount factor for future rewards,* $R : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ *is the reward given after a transition, and* $b_0 \in \Delta(\mathbb{S})$ *is the distribution of initial states.*

We call a sequence of states and joint actions $((s_0, s_1, \ldots, s_{n+1}), (a_0, a_1, \ldots, a_n))$ an *environment trace* of MMDP $\mathcal{M}$ if every pair (s_i, a_i, s_{i+1}) is allowed by the transition function of the MMDP.

$\mathcal{M}$ is said to be in a *deadlock* at state $s \in \mathbb{S}$ if $\mathcal{E}(s) = \emptyset$. $\mathcal{M}$ is *deadlock-free* if there are no environment traces of $\mathcal{M}$ that end in a deadlock state.

We will often have some prior knowledge of how to abstract the states in a meaningful way for safety properties. For example, in a domain where we avoid collisions between agents, the relative positions of the agents may be useful, even if the absolute positions of all agents are not known. We call a function $f : \mathbb{S} \to L$ which translates MMDP states into members of a given *label set* an *abstraction function*. Note that some prior work refers to f as the *observer function* [1, 9]; we use a different name to avoid potentially conflicting terminology. Every environment trace through a MMDP has a corresponding *label trace* $((l_0, a_0), \ldots, (l_n, a_n))$, where $\forall i \in \{0, \ldots, n\}, l_i = f(s_i)$.[1] Several unique environment traces may correspond to the same label trace.

We note that an environment with partial observability or a more complex reward structure is acceptable; as long as each agent can independently calculate the current label, a shield can be constructed using this framework. However, for simplicity, we continue to use MMDPs in our description.

Definition 2 (DFA) *A* Deterministic Finite Automaton (DFA) *is a tuple* $\phi = (Q, \Sigma, \delta, s_0, F)$ *where* Q *is a set of states,* $s_0 \in Q$ *is the initial state,* Σ *is an alphabet,* $\delta : Q \times \Sigma \to Q$ *is the transition function, and* $F \subseteq Q$ *is the set of accepting states.*

[1] We intentionally omit $f(s_{n+1})$ to allow label traces to be a word of $(L \times \mathbb{A})^*$.

Given a word $(w_0, w_1, \ldots, w_n) \in \Sigma^*$, the corresponding *DFA trace* $(q_0, q_1, \ldots, q_{n+1}) \in Q^*$ is obtained by stepping through the transition function. A word is *accepted* by the DFA iff the last state of its corresponding DFA trace is a member of F.

For a given MMDP $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$ and label set L, we define a *safety specification* over $\mathcal{M}$ and L to be a DFA $\phi^s = (Q^s, (L \times \mathbb{A}), \delta^s, s_0^s, F^s)$ which accepts the empty word and does not contain any transitions from $Q^s \setminus F^s$ to F^s. $\mathcal{M}$ satisfies ϕ^s ($\mathcal{M} \vDash \phi^s$) iff all possible label traces of $\mathcal{M}$ are accepted by ϕ^s. ϕ^s induces the *accepting action function* $\mathcal{C}^{\phi^s} : Q^s \times L \rightarrow \mathcal{P}(\mathbb{A}) = (q, l) \rightarrow \{a \in \mathbb{A} | \delta^s(q, (l, a)) \in F^s\}$; given a DFA state and label, output the actions which transition the DFA to an accepting state.

We can compose a DFA ϕ^g and a MMDP $\mathcal{M}$ to form a new MMDP $(\mathcal{M}||\phi^g)$. The state space of the resulting MMDP is the product of the state spaces of $\mathcal{M}$ and ϕ^g, and the transition function steps through the DFA and MMDP's states in lockstep. At state (s, q), a given action is only enabled in the resulting MMDP if it is a member of both $\mathcal{E}(s)$ and $\mathcal{C}^{\phi^g}(q, f(s))$.

4 Centralized Shielding

We use a definition of a centralized shield based on the joint actions which are allowed at each shield state. Note that some prior formulations define the shield as a Moore machine which outputs a specific safe action in response to an input action [5]; the following definition is equivalent.

Definition 3 (Centralized Shield) *A DFA ϕ^g is a* centralized shield *for MMDP $\mathcal{M}$ that enforces safety specification ϕ^s iff $\mathcal{M}||\phi^g \vDash \phi^s$ and $\mathcal{M}||\phi^g$ is deadlock-free.*

Problem 1 (Centralized Shield Synthesis) *Given a MMDP $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$, a label set L, an abstraction function f over $\mathcal{M}$ and L, and a safety specification ϕ^s over $\mathcal{M}$ and L, synthesize a centralized shield for $\mathcal{M}$ that enforces ϕ^s.*

This problem has previously been explored in [1] for the single-agent case, and [9] for the multi-agent case; we do not aim to advance the state of the art in this domain. Briefly, centralized shield synthesis works by constructing a game with two players: the agents, and the environment. When constructing a shield for a multiagent environment, all agents are collectively considered one "player" whose action set equals the space of joint actions for all of the agents. The environment proposes a label, and the agents must respond with a joint action which does not violate the safety specification. The safety game is solved, typically by finding a fixpoint [8].

A detailed model of the environment is not necessary for this synthesis; an abstraction which captures all possible label traces is sufficient. For example, our test environments all involve avoiding collisions. We therefore use the relative positions of the agents as the label set. The environment abstraction only captures information about how the relative positions of the agents change with each joint action, not any information about rewards or walls.

When using a centralized shield, the agents never explicitly construct the full composition of $\mathcal{M}||\phi^g$, as $\mathcal{M}$ may have an infinite state space, and its exact dynamics may not be known. Instead, the agents maintain the current centralized shield state g, initialized to s_0^g. At every time step, they observe the label l from the environment. Using the shield's induced accepting action function, the agents compute the set of safe actions $\mathcal{C}^{\phi^g}(g, l)$.

At this point, there are two common design choices in how the shield gets used [1]. In *pre-posed shielding*, the agent's action space is restricted to $\mathcal{C}^{\phi^g}(g, l)$; it is unable to choose an unsafe action. In *post-posed shielding*, the agent is free to choose any action. However, if the action is not safe, the shield substitutes the action with an arbitrary action $a' \in \mathcal{C}^{\phi^g}(g, l)$. The agent observes, and is trained on the environment transition (s, a', r, s'). The agent is additionally trained on the synthetic transition $(s, a, r + r_p, s')$ where r_p is a penalty reward. For both designs, the shield's transition function δ^g is then used to obtain the next centralized shield state. We choose to use post-posed shielding for our experiments, as this method is generalizable to a greater variety of agents.

5 Decentralized Shielding

The use of a centralized shield requires agents to communicate with each other for two reasons:

First, the set of safe individual actions for a given agent may depend on the action choice of all other agents. For example, consider the safety specification where an even number of agents must pick a given action. Unless a single joint action is decided upon in advance, the agents must all communicate before action selection in order to choose a safe action.

Furthermore, even if the set of safe individual actions are independent, the centralized shield may transition to different states in response to different joint actions—even if both joint actions produce the same result in the environment. Alshiekh et al. [1] gives the example of an industrial valve controller, where the flow of water through the valve is stochastic when the valve is open. Consider a similar environment where multiple agents each control an independent valve, and are unable to observe other agents' valve states, but they do observe the flow of water through each valve. An agent would not be able to tell the difference between other agents' valves being closed, versus their valves being open but without flow. Depending on the shield structure, there may be no way to calculate what the next shield state is based on flow rates alone.

Therefore, in a no-communication environment, each agent must be equipped with a shield which has the following two properties: First, each agent must be able to independently choose actions. No matter what combination of actions are chosen by the various agents, the joint action must be safe. We call a shield with this property *Cartesian*, as the safe joint action set results from the Cartesian product of each agents' safe individual action sets. Second, each agent's local shield must be able to independently determine its next internal state, without direct knowledge of the actions that other agents have taken. We call a shield with this property *unambiguous*.

We call a DFA an *individual shield* for agent $i \in \mathbb{D}$ if the transition alphabet of this DFA is $L \times \mathbb{A}_i$. If the product[2] of a series of individual shields forms a centralized shield which enforces specification ϕ^s, we say that the individual shields collectively form a *decentralized shield* which enforces ϕ^s.

Problem 2 (Dec-A Shield Synthesis) *Given a MMDP $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$, a label set L, an abstraction function over $\mathcal{M}$ and L, and a safety specification ϕ^s, synthesize a decentralized shield which enforces ϕ^s.*

Problem 2 is generally hard to solve, and may even be undecidable in some cases, as it is related to decentralized controller synthesis problems [17, 20]. Therefore, instead of solving Problem 2 directly, we opt for the following approach. First, we attempt to synthesize a centralized shield (i.e., to solve Problem 1) using the methods described in Section 4. If this fails, i.e., if no centralized shield exists, than we can conclude that no decentralized shield exists either. If, on the other hand, we manage to synthesize a centralized shield, than we can *decentralize* it, thereby turning our decentralized shield synthesis problem into the following *shield decomposition* problem:

Problem 3 (Dec-A Shield Decomposition) *Given a MMDP $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$ and a centralized shield ϕ^g for $\mathcal{M}$, synthesize a decentralized shield for $\mathcal{M}$ which enforces ϕ^g.*

Problem 3 admits a trivial solution, where for every shield state and label, we prescribe a single safe joint action in advance. Such a joint action is guaranteed to exist for all reachable states due to the deadlock-free property of the centralized shield. While the result is technically a decentralized shield that enforces ϕ^s, this trivial solution does not allow the agents any freedom to explore actions and optimize for expected future rewards.

Instead, we would ideally like to synthesize a *maximally permissive* shield; i.e. a shield for which there does not exist an alternate shield that admits a strict superset of label traces. However, this is a fundamentally difficult problem. We therefore present a procedure which produces a maximally permissive decentralized shield when the input centralized shield is Cartesian and unambiguous, and in all cases produces a shield which is at least as permissive as the trivial solution. The steps of our algorithm are illustrated in Figure 1, and described in what follows.

First, for every centralized shield state, and for every label which can be observed from that state, find a subset of safe joint actions which is Cartesian and unambiguous. This step works by first choosing a single safe joint action, like in the trivial solution. The algorithm then attempts to add individual actions for each agent in turn. When considering a given individual action, the algorithm first calculates the set of all joint actions which would be enabled if this individual action were

[2] Unlike a standard automaton product, we synchronize on L, so the resulting automaton's alphabet is $L \times \mathbb{A}$.

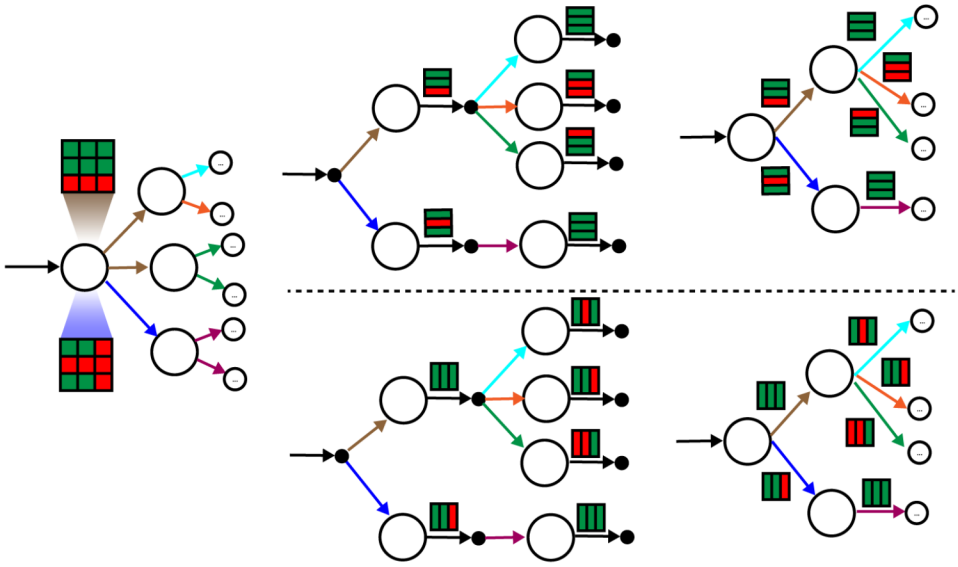

Figure 1: An overview of our proposed algorithm. Step 1 produces a set of safe individual actions for each state and label, step 2 projects these individual actions onto transient-state individual shields, and step 3 eliminates the transient states to obtain an individual shield for each agent.

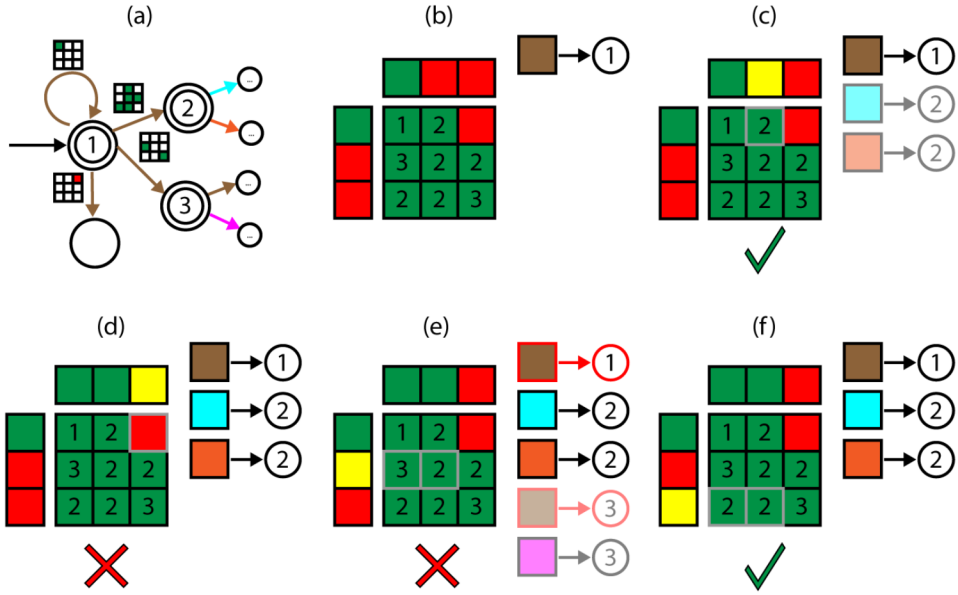

Figure 2: A detailed view of Algorithm 1, which finds a Cartesian set of safe, unambiguous actions. (a) An example input centralized shield. This figure focuses on the process of finding safe actions for state 1, when a brown label has been observed. (b) The grid represents the next state which will be seen after taking each joint action. A single safe joint action—both agents choosing action 1—is chosen arbitrarily. $\mathcal{R}$ is initialized with the information that if the next label is brown, the next state must be 1. (c) The algorithm considers adding agent 1, action 2. This would not enable any unsafe joint actions, and $\mathcal{R}$ remains unambiguous, so this individual action is allowed. (d) The algorithm considers agent 1, action 3. This would add an unsafe joint action and is rejected. (e) The algorithm considers agent 2, action 2. This would add two joint actions. Adding state 3 as a possible next state would cause $\mathcal{R}$ to become ambiguous if brown were observed, so the individual action is rejected. (f) The algorithm considers agent 2, action 3. This adds two joint actions—since neither joint action causes any issues, this individual action is added. The algorithm is now complete. Agent 1 is allowed to take actions 1 or 2, while agent 2 can take actions 1 or 3.

Algorithm 1 Step 1 of shield decomposition: Determining safe actions

Input
  $\phi^g = (G, (L \times \mathbb{A}), \delta^g, s_0^g, F^g)$ // A centralized shield where $\mathbb{A} = \prod_{i \in \mathbb{D}} \mathbb{A}_i$
Output
  $\mathcal{D}_i : G \times L \to \mathcal{P}(\mathbb{A}_i)$ // A set of safe individual actions for agent i
  $\mathcal{R} : G \times L \to [L \to G]$ // Uniquely determine the next state, given the observed label
procedure CALCULATESAFEACTIONS(ϕ^g)
  for $s \in G, l \in \mathcal{L}(s)$ **do** // $\mathcal{L}(s)$ = set of all labels which may be observed in shield state s
    $\mathcal{C}(s, l) \leftarrow \{a \in \mathbb{A} | \delta^g(s, (l, a)) \in F^g\}$ // All legal joint actions from s, l
    Choose arbitrary $a = (a_1, a_2, \ldots, a_n) \in \mathcal{C}^l$ // Guaranteed to exist, given that $l \in \mathcal{L}(s)$
    $\forall i \in \mathbb{D} : \mathcal{D}_i(s, l) = \{a_i\}$
    $\mathcal{R}(s, l) \leftarrow [\mathcal{L}(\delta^g((s, l), a)) \to \delta^g((s, l), a)]$
    for $i \in \mathbb{D}$ **do** // Optimization: randomly permute $\mathbb{D}$ at each time step
      for $a_i' \in \mathbb{A}_i$ **do**
        $A \leftarrow \{a_i'\} \times \prod_{j \in \mathbb{D}, j \neq i} \mathcal{D}_j(s, l)$ // New joint actions if a_i' were added to $\mathcal{D}_i$
        if $A \subseteq \mathcal{C}(s, l)$ **and** UNAMBIGUOUSACTIONS($\mathcal{R}(s, l), s, l, A$) **then**
          $\mathcal{D}_i(s, l) \leftarrow \mathcal{D}_i(s, l) \cup \{a_i'\}$
          $\forall a \in A, \mathcal{R}(s, l) \leftarrow \mathcal{R}(s, l) \cup [\mathcal{L}(\delta^g((s, l), a)) \to \delta^g((s, l), a)]$
        end if
      end for
    end for
  end for
  return $\mathcal{D}_i \forall i \in \mathbb{D}, \mathcal{R}$
end procedure

enabled. For example, if the algorithm has previously decided that agent 1 may take either actions a or b, then enabling agent 2's action b would require checking that both joint action candidates (a, b) and (b, b) are suitable. In this manner, the set of enabled joint actions remains Cartesian. If any joint action candidate is unsafe, or if adding any of them would cause ambiguity in the next state, then the individual action is rejected. The details of this step are further illustrated in Figure 2, and described more specifically in Algorithm 1.

Second, we make use of the following structure:

Definition 4 (Transient-State Individual Shield) *Given a label set L and a MMDP $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A} = \prod_{i \in \mathbb{D}} \mathbb{A}_i, \mathcal{E}, T, \gamma, R, b_0)$, a* transient-state individual shield *for agent $i \in \mathbb{D}$ is a tuple $\phi_i^t = (Q_1, L, \delta_1, Q_2, \mathbb{A}_i, \mathcal{A}, \delta_2, s_0)$ where Q_1 and Q_2 are sets of states, $\delta_1 : Q_1 \times L \to Q_2$ is the environment transition function, $\mathcal{A} : Q_2 \to \mathbb{A}_i$ is the allowed action function, $\delta_2 : Q_2 \to Q_1$ is the agent transition function, and $s_0 \in Q_1$ is the initial state.*

Q_1 represents states in which an agent has just taken an action, but the next environment label has not been observed; Q_2 represents states in which a label has been observed, and the agent is about to take an action. For a given state $q \in Q_2$, no matter what action in $\mathcal{A}(q)$ the agent takes, the next state is always $\delta_2(q)$.

Algorithm 2 (in appendix) projects the joint action set found by the previous step to a transient-state individual shield per agent. Lastly, Algorithm 3 (in appendix) performs some mild post-processing to eliminate the transient states, and obtain an individual shield for each agent.

If the centralized shield given as input to this sequence of algorithms enforces a given specification, the decentralized shield will also enforce that specification. This important safety property is formalized and proven in the appendix as Theorem 1.

As mentioned previously, given an input centralized shield which is already both unambiguous and Cartesian, the resulting decentralized shield will be maximally permissive. A shield is trivially unambiguous if each label only appears in a single state; many safety specifications, especially in fully observable environments, are enforceable by such a shield. This includes all *stateless* shields—shields where $F^g = \{s_0^g\}$. For input shields which are non-Cartesian, our algorithm gives one agent "right of way"; an agent with higher priority cannot put another agent in a situation for which there

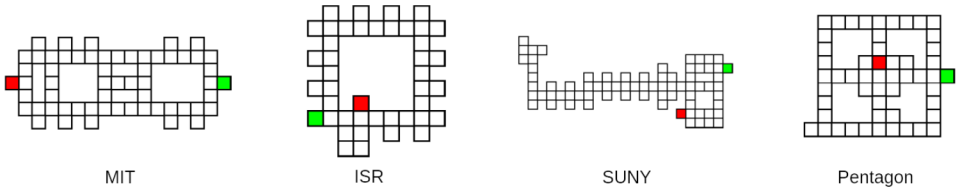

Figure 3: The four gridworld environments adapted from Melo and Veloso [15]. One agent each starts in the red and green squares; the goal is to switch positions.

are no safe actions, but the lower-priority agent must otherwise accommodate any potential actions that the higher-priority agent may take. A centralized shield is usually Cartesian in states where such a "right of way" is not necessary, i.e. when agents are not in contention with each other.

No matter what, the decentralized shield which our algorithm generates will always allow at least one safe joint action, in a manner similar to the trivial solution described above. In practice (c.f. Section 6), we find that the resulting decentralized shield tends to allow a large range of joint actions, even on input shields which are not fully unambiguous or Cartesian.

6 Experiments

We aim to show with our experiments that, as predicted by Theorem 1, decentralized shields correctly prevent the agents from taking unsafe actions, while an equivalent unshielded agent would take unsafe actions in the same situation. Furthermore, we hypothesize that agents which use our decentralized shielding method perform comparably to agents which use a centralized shield.

6.1 The Gridworld Collision Domain

We adapt the gridworld maps from Melo and Veloso [15] to test our method. These maps, shown in Figure 3, have previously been used to test multiagent shielding, with an assumption of instantaneous local communication [9]. Each agent has five actions available: movement in the four cardinal directions, or a no-op. If all agents reach their goal positions, they each receive a +100 reward. Any agents which hit a wall receive a -10 reward. The safety specification is that a collision between two agents should never occur (including agents crossing over each other). In an unshielded environment, if an agent violates this specification, all agents receive a -30 reward. Agents receive a -1 reward at all other time steps. We use the four maps "MIT," "ISR," "SUNY," and "Pentagon," both with and without randomized start positions. These maps range in size from 8x10 to 10x23.

We synthesized a centralized shield which avoids collisions, using a reactive synthesis tool [8]. Note that this shield is shared across all four environments—it is a coarse abstraction which operates solely on relative agent positions, ensuring collision-free behavior in any gridworld with similar dynamics. The synthesis tool outputs a shield with many redundant states; we remove duplicate and unreachable states. In the resulting shield, each environment label only appears in a single state, so the resulting shield is trivially unambiguous (though it is not fully Cartesian).

We then utilized the procedure described in Section 5 to obtain a decentralized shield which works for all of our gridworld environments. The resulting individual shields are more conservative when the agents are near each other, ensuring that no more than one agent moves into a potentially contested area. Centralized shield synthesis took approximately 10 minutes on a M1 Macbook Pro, and our decentralization algorithm ran in under 30 seconds.

We analyzed the structure of the shields to determine how much the decentralization procedure restricts the set of actions available. With the centralized shield, there are an average of 23.30 safe joint actions available to the agents for a given shield state. Using the trivial solution from Section 5, every shield state would only have 1 joint action available. Our decentralization algorithm produces a shield where there are an average of 22.07 joint actions available—not much more restrictive than the centralized shield.

We trained independent tabular Q-learning agents using ϵ-greedy exploration, with a linear ϵ annealing schedule from 1 to 0.05. The discount factor is 0.9. Agents were trained with a centralized

Table 1: Agent performance after 2.5 million environment steps, using the same shield for training and testing. Results show average evaluation reward and standard error over 10 seeds, as well as average total violations over 100 testing episodes in parentheses. Note that centralized shielding is only for comparison; in a no-communication environment, this method would not be feasible.

Start Type	Map Name	Centralized	Decentralized	No Shield
Fixed	ISR	90.4 ± 0.6 (0)	89.9 ± 0.5 (0)	90.0 ± 0.6 (12.3)
	MIT	72.0 ± 0.8 (0)	71.6 ± 0.3 (0)	72.8 ± 0.5 (0)
	Pentagon	89.0 ± 0.4 (0)	82.0 ± 5.8 (0)	89.7 ± 0.3 (4.8)
	SUNY	83.3 ± 1.4 (0)	86.6 ± 0.6 (0)	86.2 ± 0.4 (0)
Random	ISR	78.6 ± 2.2 (0)	76.1 ± 2.0 (0)	83.6 ± 1.4 (2.0)
	MIT	82.6 ± 0.2 (0)	83.2 ± 0.2 (0)	81.4 ± 0.9 (0.4)
	Pentagon	88.4 ± 0.5 (0)	80.3 ± 3.5 (0)	89.1 ± 0.3 (0.9)
	SUNY	78.1 ± 0.4 (0)	73.1 ± 1.7 (0)	77.3 ± 0.4 (0.4)

shield, a decentralized shield, and with no shield. When an agent attempts to take an action a which is not allowed by the shield, the penalty reward for the synthetic transition $r_p = -10$. These hyperparameters were chosen to closely match the parameters from [9], where possible.

Agents were trained for 2.5 million timesteps each, using 10 different random seeds for each configuration. Each individual run took approximately 5 minutes on a single core of an Intel server CPU; we trained 240 agents in total. This was sufficient for convergence in all variations. Results are shown with both fixed and random starting positions in Table 1. Extended tables and training curves are listed in the appendix.

As guaranteed by Theorem 1, this experiment shows that, because the centralized shield protects the agents from taking such unsafe actions, the agents which use a decentralized shield are also prevented from taking unsafe actions. The experiment also reveals that neither shielding method consistently performs better than the other method, as the highest-performing agent is generally within the margin of error. Therefore, our decentralized shielding approach allows safe, high-quality learning during training and execution in environments without communication. The training curves (Figure 4, in the appendix) tell the same story—no method works significantly faster during training.

6.2 The Relative Particle + Momentum Domain

One advantage of formally synthesizing a shield, rather than naively blocking actions which immediately lead to an unsafe state, is that a shield can avoid causing a situation in which there are no safe actions available. This frequently occurs in domains with momentum; if a car is driving towards a cliff, its shield must react well before the car reaches the actual edge. Therefore, we would like to evaluate how well our shield works in such momentum-based environments.

We introduce an environment as follows. There are two agents, representing particles on a discretized grid. Agent 1 begins 9 units above, and 9 units to the left of agent 2, with both agents at rest. Both agents obtain a reward of 100 when agent 1 gets to the position 9 units to the right of and below agent 2. If the agents get more than 10 units apart in any individual dimension, or collide or cross with each other, it is considered a safety violation (and the agents get a reward of -30). Otherwise, all agents get a reward of -1 at each time step. Each agent can move in any of the four cardinal directions, or do nothing. Additionally, agents have *momentum*; an agent's action is added to its previous movement to obtain its next movement in the environment. The relative velocity is capped at 2 units per time step in each direction. In the fully observable variant of this environment, both agents observe the relative positions and relative velocities. In the partially observable version, the agents only observe the relative positions.

We synthesized shields which only use the relative positions of the agents as the label set; such shields can be utilized in either variant of the environment. Using the same structure analysis as in Section 6.1, we calculated that the centralized shield allows an average of 21.67 actions, while the decentralized shield allows an average of 20.81 actions.

To show that our decentralized shielding algorithm is also effective regardless of the underlying reinforcement learning algorithm used by the agent, we trained reactive DQN agents to solve this

Table 2: Individual DQN Agent Performance in the Particle-Momentum domain.

Observability	Start Type	Centralized	Decentralized	No Shield
Partial	Fixed	70.9 ± 5.8 (0)	82.8 ± 2.3 (0)	67.8 ± 3.0 (34.3)
	Random	70.8 ± 1.7 (0)	80.5 ± 1.4 (0)	33.2 ± 2.4 (115.7)
Full	Fixed	91.6 ± 0.0 (0)	91.6 ± 0.0 (0)	90.7 ± 0.2 (3.0)
	Random	94.6 ± 0.1 (0)	94.6 ± 0.1 (0)	93.7 ± 0.2 (2.8)

task [16]. Full hyperparameters for this agent are located in the appendix. 10 random seeds were used in the fully observable case, and 50 seeds were used in the partially observable version of the environment. Each run took approximately 8 hours using 3 threads on a server CPU.

The results are shown in Table 2. Under full observability, we reach similar conclusions as in the gridworld-collision domain: the agents protected by centralized and decentralized shields both achieve comparable performance. As guaranteed by Theorem 1, the agent which uses the decentralized shield is as safe as the agent which uses a centralized shield. This is further emphasized under the partially observable variant of the environment, where the unshielded agents behave in a wildly unsafe manner. In contrast, armed with a description of the basic physics of the environment, the shields are able to infer the current momentum based on the incremental changes in relative agent positions, leading them to never take an unsafe action.

7 Broader Impact and Limitations

We believe that the algorithm presented in this work represents progress towards the more general goal of safe reinforcement learning. However, like most provably correct methods, shielding also suffers from scalability issues such as the well-known state explosion problem [3]. For example, prior to de-duplication and removal of unreachable states, the shield which we synthesized for the gridworld experiments contained 71130 states. The particle environment's shield, including momentum information, contained 170348 states. Our decentralization algorithm takes linear time with respect to the number of state-label pairs in the input shield, so we are well positioned to take advantage of methods which synthesize a centralized shield without such state explosion.

Additionally, writing an environment abstraction is extremely difficult—we wrote several iterations of our environment abstractions, each containing subtle but serious bugs, before finally obtaining precise and accurate models of the environment. If the centralized shield which our algorithm is given as input is a poor fit for the environment, the decentralized shield which it outputs cannot be expected to perform well. We therefore caution against overreliance on a model abstraction which is not proven to be completely correct.

As with any advance in machine learning, it may be possible to use our method to safely train an agent to perform an unethical task. However, we hope that as a whole, a process for safe reinforcement learning training and execution will reduce potential societal harms resulting from AI.

8 Conclusion & Future Work

Our experiments show that the shield decomposition method described in this paper results in shields that are safe and do not require any communication between agents. Furthermore, agents which are trained and evaluated with a decentralized shield perform comparably to agents which are trained and evaluated with a centralized shield, or without any shield.

We are currently working to extend our results in several ways. First, we are implementing additional agents and training schemes, including individual Deep Q Networks with convolutional layers [16], and Centralized-Critic Decentralized-Actor methods [10]. Second, we are investigating methods to scale our shield decomposition method to the more complicated shields and safety specifications necessary to describe complex continuous or partially-observable environments. Lastly, we would like to achieve ever more permissive decentralized shields, potentially obtaining a maximally permissive shield. This may require direct decentralized shield synthesis, without necessarily using a centralized shield as an intermediate step.

References

[1] ALSHIEKH, M., BLOEM, R., EHLERS, R., KÖNIGHOFER, B., NIEKUM, S., AND TOPCU, U. Safe reinforcement learning via shielding, 2017.

[2] AMODEI, D., OLAH, C., STEINHARDT, J., CHRISTIANO, P., SCHULMAN, J., AND MANÉ, D. Concrete problems in ai safety, 2016.

[3] BAIER, C., KATOEN, J., AND LARSEN, K. *Principles of Model Checking*. The MIT Press. MIT Press, 2008.

[4] BASTANI, O., PU, Y., AND SOLAR-LEZAMA, A. Verifiable reinforcement learning via policy extraction, 2019.

[5] BLOEM, R., KOENIGHOFER, B., KOENIGHOFER, R., AND WANG, C. Shield synthesis: Runtime enforcement for reactive systems, 2015.

[6] CARR, S., JANSEN, N., JUNGES, S., AND TOPCU, U. Safe reinforcement learning via shielding for pomdps, 2022.

[7] DALAL, G., DVIJOTHAM, K., VECERIK, M., HESTER, T., PADURARU, C., AND TASSA, Y. Safe exploration in continuous action spaces, 2018.

[8] EHLERS, R., AND RAMAN, V. Slugs: Extensible gr(1) synthesis. In *Computer Aided Verification* (2016), S. Chaudhuri and A. Farzan, Eds., Lecture Notes in Computer Science, Springer International Publishing, p. 333–339.

[9] ELSAYED-ALY, I., BHARADWAJ, S., AMATO, C., EHLERS, R., TOPCU, U., AND FENG, L. Safe multi-agent reinforcement learning via shielding. *CoRR abs/2101.11196* (2021).

[10] FOERSTER, J., FARQUHAR, G., AFOURAS, T., NARDELLI, N., AND WHITESON, S. Counterfactual multi-agent policy gradients, 2017.

[11] GALLAY, F., AND FALCONE, Y. Decentralized LTL enforcement. *Electronic Proceedings in Theoretical Computer Science 346* (sep 2021), 135–151.

[12] GLOROT, X., BORDES, A., AND BENGIO, Y. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (Fort Lauderdale, FL, USA, 11–13 Apr 2011), G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15 of *Proceedings of Machine Learning Research*, PMLR, pp. 315–323.

[13] IVANOV, R., WEIMER, J., ALUR, R., PAPPAS, G. J., AND LEE, I. Verisig: verifying safety properties of hybrid systems with neural network controllers, 2018.

[14] KÖNIGHOFER, B., RUDOLF, J., PALMISANO, A., TAPPLER, M., AND BLOEM, R. Online shielding for stochastic systems. In *NASA Formal Methods* (Cham, 2021), A. Dutle, M. M. Moscato, L. Titolo, C. A. Muñoz, and I. Perez, Eds., Springer International Publishing, pp. 231–248.

[15] MELO, F. S., AND VELOSO, M. Learning of coordination: Exploiting sparse interactions in multiagent systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2* (Richland, SC, 2009), AAMAS '09, International Foundation for Autonomous Agents and Multiagent Systems, p. 773–780.

[16] MNIH, V., KAVUKCUOGLU, K., SILVER, D., GRAVES, A., ANTONOGLOU, I., WIERSTRA, D., AND RIEDMILLER, M. Playing atari with deep reinforcement learning, 2013.

[17] PNEULI, A., AND ROSNER, R. Distributed reactive systems are hard to synthesize. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science* (1990), pp. 746–757 vol.2.

[18] SCHAUL, T., QUAN, J., ANTONOGLOU, I., AND SILVER, D. Prioritized experience replay, 2015.

[19] SILVER, D., SCHRITTWIESER, J., SIMONYAN, K., ANTONOGLOU, I., HUANG, A., GUEZ, A., HUBERT, T., BAKER, L., LAI, M., BOLTON, A., CHEN, Y., LILLICRAP, T., HUI, F., SIFRE, L., VAN DEN DRIESSCHE, G., GRAEPEL, T., AND HASSABIS, D. Mastering the game of Go without human knowledge. *Nature 550*, 7676 (Oct. 2017), 354–359.

[20] TRIPAKIS, S. Undecidable Problems of Decentralized Observation and Control on Regular Languages. *Information Processing Letters 90*, 1 (Apr. 2004), 21–28.

[21] VAN HASSELT, H., GUEZ, A., AND SILVER, D. Deep reinforcement learning with double q-learning, 2015.

[22] ZHANG, Y., KIM, K., AND FAINEKOS, G. Discof: Cooperative pathfinding in distributed systems with limited sensing and communication range. In *Distributed Autonomous Robotic Systems* (Tokyo, 2016), N.-Y. Chong and Y.-J. Cho, Eds., Springer Japan, pp. 325–340.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We present a method to decentralize a centralized shield, and verify experimentally that this method does not significantly affect performance in the gridworld domain.
   (b) Did you describe the limitations of your work? [Yes] See Section 7
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] This work is fairly abstract, but we lightly discuss the topic in Section 7
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Sections 4 and 5 discuss the relevant assumptions.
   (b) Did you include complete proofs of all theoretical results? [Yes] See appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We attach our code as supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 6, and the appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The tables and figures are all reflective of this.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The total compute time is mentioned in Section 6; the provider of the computation power will be mentioned in the acknowledgements section.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 6.1, where we discuss the origins of the environment maps.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]