Shield Decentralization for Safe Multi-Agent Reinforcement Learning

Daniel Melcer
Northeastern University
Boston, MA 02115
melcer.d@northeastern.edu

Christopher Amato*
Northeastern University
Boston, MA 02115
c.amato@northeastern.edu

Stavros Tripakis*
Northeastern University
Boston, MA 02115
stavros@northeastern.edu

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 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 environments with decentralized training and execution for the first time.

1 Introduction

Recently, the advent of deep reinforcement learning has produced solutions in highly complex domains such as Atari [17], and the game of Go [20]. 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 [10]. Such methods assume instantaneous communication with a centralized coordination algorithm, which is not 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.

*Equal Advising

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

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 value in the environment, and that, as is to be expected from the theoretical guarantees of our framework, decentralized shielding achieves 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 [6], from the Formal Methods community. In the single-agent case, shielding has been extended to environments with continuous action spaces [8], stochastic or adversarial environments [15], and partially-observable environments [7]. As with all shielding methods, these extensions require a model of the environment, with a varying degree of abstraction. This differs slightly from contract-based decomposition [5]; instead of defining a contract first, and then constructing systems whose interfaces satisfy this contract, shielding involves constraining an agent to act safely within an environment whose dynamics are fixed.

Other proposed methods specify algorithms for enforcing safety in the presence of communication constraints or delays for pathfinding agents [23], or for general LTL specifications [12], 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, 14]. 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 several relevant structures, and a formalization of the centralized shielding problem. Many elements of this framework have been described in prior formulations [1, 6], and extended to the multi-agent setting [10].

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, 10]; 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.*

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* $\mathcal{C}^{\phi^s} : Q^s \times L \to \mathcal{P}(\mathbb{A}) = (q, l) \to \{a \in \mathbb{A} | \delta^s(q, (l, a)) \in F^s\}$; 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 [6]; 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 [10] 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 iterating to a fixpoint [9].

A complete 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 $\mathcal{C}^{\phi^g}(g, l)$.

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

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 without water flowing during a given step due to chance. 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 [18, 21]. 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

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

shield that enforces ϕ^s, this trivial solution does not allow the agents any freedom to explore actions and optimize for expected returns.

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 in the general case. We therefore present a procedure which produces a maximally permissive decentralized shield when the input centralized shield is Cartesian and unambiguous, and in all other 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.

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
    Pick arbitrary $a = (a_1, a_2, \ldots, a_n) \in \mathcal{C}(s, l)$ // 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

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 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. Since the starting joint action and the order of candidate individual actions are arbitrary, this algorithm's output is not unique. The details of this step are further illustrated in Figure 2, and described more specifically in Algorithm 1.

Algorithm 1 performs most of the work in this sequence of algorithms; it takes a centralized shield as input, and outputs a set of safe actions for every state-label combination, as well as the information necessary to determine the next state of the environment.

Algorithm 2 (in appendix) projects the actions found by the previous step into an input-output state machine for each agent, which we call a *transient-state individual shield*. This structure has two categories of states: label states, which receive an observation from the environment (shown as black dots in Figure 1); and action states, which show the allowed actions of the agent (shown as white circles). Lastly, Algorithm 3 (in appendix) performs some mild post-processing so that this structure conforms to the shield interface. Additional information about these structures and algorithms can be found in Section A.

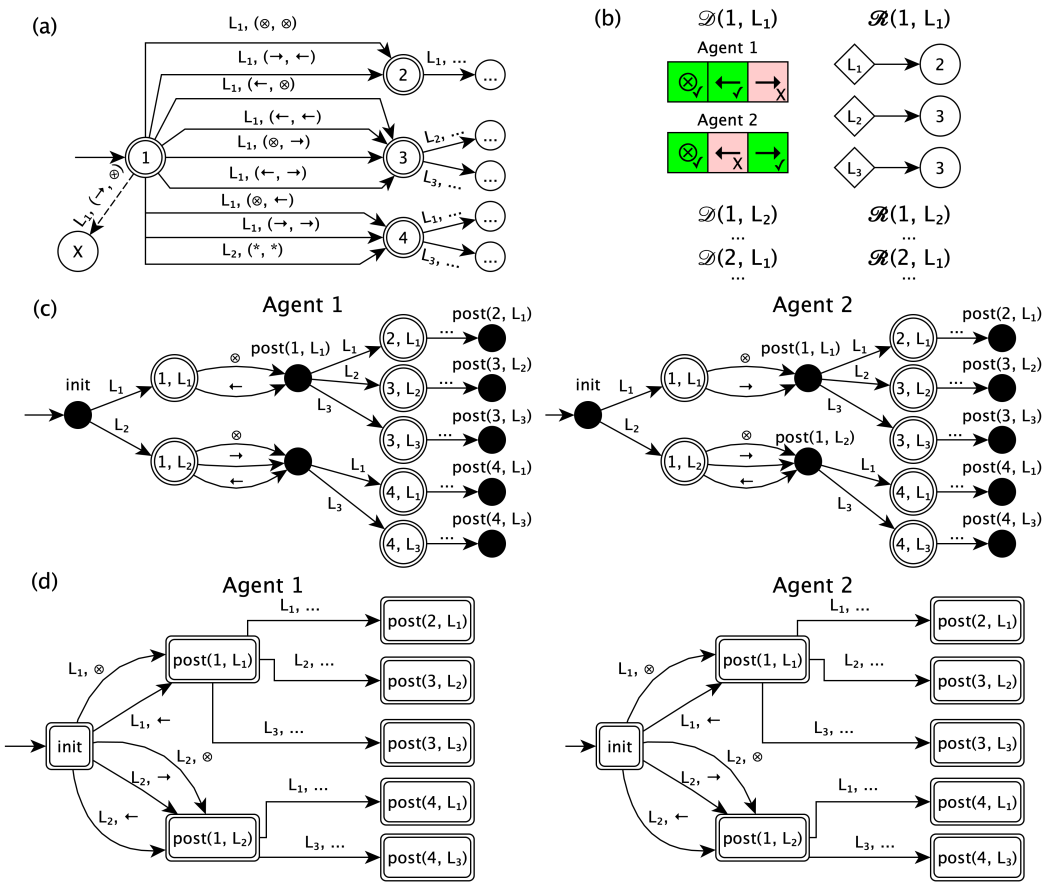

Figure 1: An overview of our algorithm sequence, in a hypothetical environment where $\mathbb{D} = \{1, 2\}, \mathbb{A}_1 = \mathbb{A}_2 = \{\bigotimes, \leftarrow, \rightarrow\}, \{L_1, L_2, L_3\} \in L$. (a) The input centralized shield, showing $\mathcal{L}(1) = \{L_1, L_2\}, \mathcal{L}(2) = \{L_1\}, \mathcal{L}(3) = \{L_2, L_3\}, \mathcal{L}(4) = \{L_1, L_3\}$, as well as several transitions. (b) The output of Algorithm 1: $\mathcal{D}_{1..n}$, which denotes the allowed actions for each agent, and $\mathcal{R}$, which maps labels seen in the next state to the state number. Here, for shield state 1 and label L_1, it is safe for agent 1 to take actions $\bigotimes$ or $\leftarrow$, while agent 2 may take $\bigotimes$ or $\rightarrow$. (c) The output of Algorithm 2: transient-state individual shields for each agent, constructed using $\mathcal{D}_{1..n}$ and $\mathcal{R}$. (d) The output of Algorithm 3: individual shields, constructed by transforming the transient-state individual shields. These can be used by agents in communication-free environments to ensure safety.

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 as the following theorem, and proven in Section B.2 of the appendix:

Theorem 1 *Given a MMDP $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$ where $\forall s \in \mathbb{S}, \mathcal{E}(s) = \mathbb{A}$, and a centralized shield ϕ^g for $\mathcal{M}$, the result of Algorithms 1, 2, and 3 is a decentralized shield for $\mathcal{M}$ which enforces ϕ^g.*

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 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.

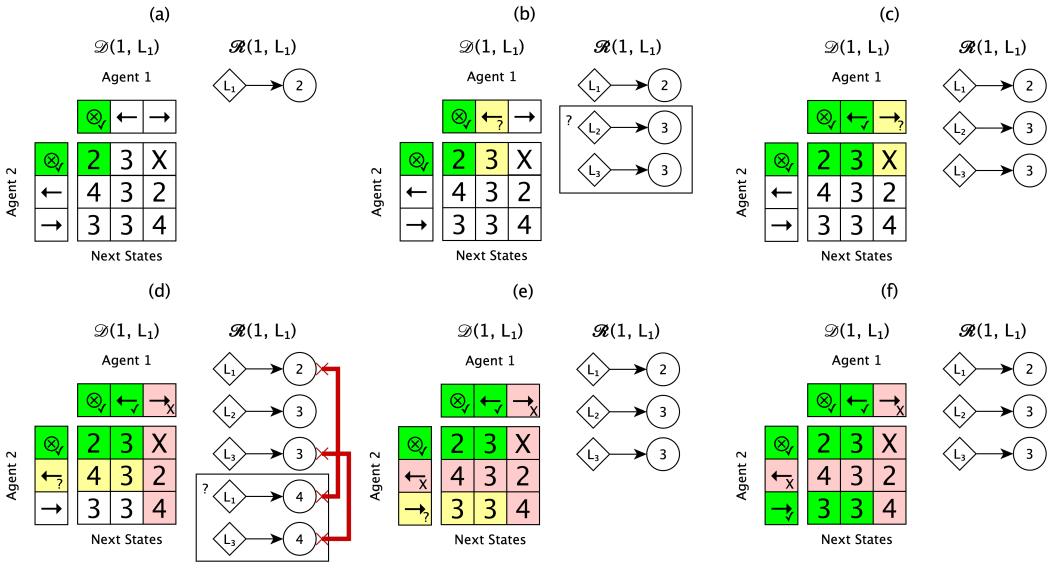

Figure 2: A detailed view of Algorithm 1, which finds a Cartesian set of safe, unambiguous actions. The shield from Figure 1 is used as input. This figure focuses on the process of finding safe actions for state 1, after L_1 has been observed. The 3x3 grid of squares represents the joint actions allowed by the centralized shield for a given state-label pair. The 3x1 rectangles on the top and left represent the individual actions which the decentralized shield will allow. Green with a check mark represents that the individual action is allowed, red with an "X" represents that the action is not allowed, and yellow with "?" represents that the action is under consideration. The mapping from diamonds to circles represents knowledge of what the next state will be, given the next shield. (a) A single safe joint action—$(\bigotimes, \bigotimes)$—is chosen arbitrarily. $\mathcal{R}$ reflects that if the agents see L_1 next, state 2 comes next. (b) The agent considers adding agent 1, action $\leftarrow$; it is safe so this action is allowed. (c) The algorithm considers agent 1, action $\rightarrow$. It may lead to an unsafe state, so this individual action is rejected. (d) The algorithm considers agent 2, action $\leftarrow$. This would add two joint actions. Adding state 4 as a possible next state means that the states are ambiguous if L_1 or L_3 is observed, so it is rejected. (e) The algorithm considers agent 2, action $\rightarrow$. Both joint actions which this would add are safe, so it is allowed. We don't consider the effects of action $(\rightarrow, \rightarrow)$, as agent 1 is prevented from selecting $\rightarrow$. (f) The finished result, used for the next step of the shield decentralization procedure.

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 may 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 [16] 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 [10]. 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

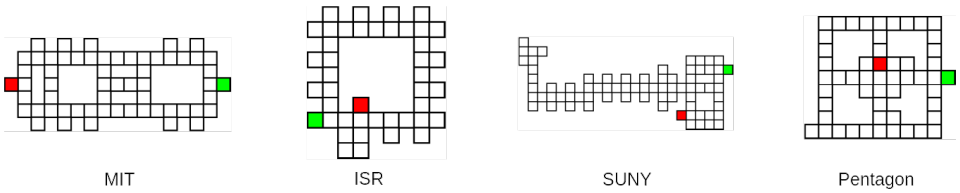

Figure 3: The four gridworld environments adapted from Melo and Veloso [16]. One agent each starts in the red and green squares; the goal is to switch positions.

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 [9]. 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 five 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 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 [10], 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 directly lead to an unsafe state, is that a shielded agent can avoid states which do not immediately violate a specification, but where all future paths end in an unsafe state. This can occur in domains with momentum; if a car is driving towards a cliff, it must slow down well before it reaches the edge. Therefore, we would like to evaluate agents which use our shield in momentum-based environments.

Table 1: Agent performance after 2.5 million environment steps, using the same shield for training and testing. Results show average evaluation return 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)

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)

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 task [17]. 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 reactive agents do not have sufficient information about the environment, and thus behave in a wildly unsafe manner. In contrast, 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 algorithms presented in this work represent progress towards the more general goal of safe reinforcement learning. Like most provably correct methods, shielding suffers from scalability issues such as the well-known state explosion problem [3]. For example, prior to deduplication 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. The size of a centralized shield tends to scale exponentially with the number of agents. 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, obtaining an environment abstraction is difficult—we wrote several iterations of our abstractions, each containing subtle but serious bugs, before finally obtaining precise and accurate models of the environments. If our algorithm is given an inaccurate centralized shield, the decentralized shield which the algorithm outputs cannot be expected to perform well. We therefore caution against overreliance on an abstraction which may not accurately represent the environment.

Furthermore, there exist some environments where it is difficult, or even impossible, to determine a shared label set. For example, an environment with highly asymmetric observations would not satisfy the assumptions of the shielding framework we use. However, there may still be sufficient information to ensure safety in these challenging environments; we are working on developing methods to synthesize a decentralized shield without a shared label set.

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 [17], and Centralized-Critic Decentralized-Actor methods [11]. 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. Our algorithm makes a number of arbitrary choices which do not impact safety, but which could affect shield performance. It may be useful to define additional metrics for shield performance and permissiveness that do not necessarily involve training and testing agents over thousands of episodes, and to use these metrics to improve the decisions that our algorithm makes. We would like to eventually develop a method which obtains a maximally permissive decentralized shield, no matter what properties hold for the input shield. Finally, although similar problems in formal methods are undecidable [18, 21], we would like to explore heuristic methods for direct decentralized shield search and synthesis.

Acknowledgments and Disclosure of Funding

We would like to thank Ingy ElSayed-Aly for her assistance with shield synthesis. We would also like to thank the reviewers for their thoughtful comments and suggestions.

This work has been supported by NSF SaTC Award CNS-1801546, NSF CAREER Award IIS-2044993, and Army Research Office Award W911NF2010265. This work was completed in part using the Discovery cluster, supported by Northeastern University's Research Computing team.

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] BENVENISTE, A., CAILLAUD, B., NICKOVIC, D., PASSERONE, R., RACLET, J., REINKEMEIER, P., SANGIOVANNI-VINCENTELLI, A. L., DAMM, W., HENZINGER, T. A., AND LARSEN, K. G. Contracts for system design. *Found. Trends Electron. Des. Autom. 12*, 2-3 (2018), 124–400.

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

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

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

[9] 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.

[10] ELSAYED-ALY, I., BHARADWAJ, S., AMATO, C., EHLERS, R., TOPCU, U., AND FENG, L. Safe multi-agent reinforcement learning via shielding. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (Richland, SC, 2021), AAMAS '21, International Foundation for Autonomous Agents and Multiagent Systems, p. 483–491.

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

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

[13] 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.

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

[15] KÖNIGHOFER, B., RUDOLF, J., PALMISANO, A., TAPPLER, M., AND BLOEM, R. Online shielding for reinforcement learning. *Innovations in Systems and Software Engineering* (2022), 1–16.

[16] 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.

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

[18] PNUELI, 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.

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

[20] 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.

[21] TRIPAKIS, S. Undecidable problems of decentralized observation and control on regular languages. *Information Processing Letters 90*, 1 (Apr. 2004), 21–28.

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

[23] 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] We 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 is 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]

A Additional Algorithms for Shield Decentralization

Algorithm 2 accepts information about allowed actions, and outputs a *Transient-State Individual Shield* for each agent.

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 $\delta_2(q)$. Most states in Q_1, aside from s_{init}, can be thought of as "post-states" of states which are in Q_2; Figure 1 is reflective of this.

Algorithm 2 Step 2: Projection onto Individual Automata

Input
  $\phi^g = (G, (L \times \mathbb{A}), \delta^g, s_0^g, F^g)$ // A centralized shield
  $\mathcal{D}_i, \mathcal{R}$ // Outputs from Algorithm 1
Output
  $\phi_i^t = (Q_1, L, \delta_1, Q_2, \mathbb{A}_i, \mathcal{A}, \delta_2, s_0)$ // Transient-state individual shields
procedure PROJECTINDIVIDUALSHIELDS($\phi^g, \mathcal{D}_i$)
  for $i \in \mathbb{D}$ **do**
    $Q_1 \leftarrow (G \times L)_{Q_1} \cup s_{init}$
    $s_0 \leftarrow s_{init}$
    $Q_2 \leftarrow (G \times L)_{Q_2}$
    for $s \in G, l \in \mathcal{L}(s)$ **do**
      for $[l' \to s'] \in \mathcal{R}(s, l)$ **do**
        $\delta_1((s, l)_{Q_1}, l') \leftarrow (s', l')_{Q_2}$
      end for
      $\delta_2((s, l)_{Q_2}) \leftarrow (s, l)_{Q_1}$
      $\mathcal{A}((s, l)_{Q_2}) \leftarrow \mathcal{D}_i(s, l)$
    end for
    for $l \in \mathcal{L}(s_0)$ **do**
      $\delta_1(s_{init}, l) \leftarrow (s_0^g, l)_{Q_2}$
    end for
  end for
end procedure

B Decentralized shield safety

B.1 Definitions

We informally described how a DFA composes with a MMDP in the main body of the paper. We re-state this composition more precisely.

$\mathbb{1}_x$ is the indicator function, taking a value of 1 when x is true, and 0 when false. $\mathcal{C}^{\phi^g} : G \times L \to \mathcal{P}(\mathbb{A})$ is the set of joint actions in a centralized shield which lead to an accepting state, as defined on line 8 of Algorithm 1.

Definition 5 (DFA-MMDP Composition) *Given an MMDP $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$, label set L, an abstraction function f over $\mathcal{M}$ and L, and DFA $\phi^g = (G, (L \times \mathbb{A}), \delta^g, s_0^g, F^g)$ where $s_0^g \in F^g$, the* composition *of $\mathcal{M}$ and ϕ^g is a new MMDP denoted as $\mathcal{M} || \phi^g = (\mathbb{D}^c, \mathbb{S}^c, \mathbb{A}^c, \mathcal{E}^c, T^c, \gamma^c, R^c, b_0^c)$, where*

- $\mathbb{D}^c = \mathbb{D}$

Algorithm 3 Step 3: Eliminate Transient States

Input
  $\phi_i^t = (Q_1, L, \delta_1, Q_2, \mathbb{A}_i, \mathcal{A}, \delta_2, s_0)$ // From Algorithm 2.
Output
  $\phi^{s_i} = (Q^{s_i}, (L \times \mathbb{A}_i), \delta^{s_i}, s_0^{s_i}, F^{s_i})$ // Individual shield for agent i
procedure ELIMTRANSIENTSTATES(ϕ_i^t)
  $Q^{s_i} \leftarrow Q_1, s_0^{s_i} \leftarrow s_0, F^{s_i} \leftarrow Q_1$
  for $s \in Q_1$ **do**
    for $l \in L, a \in \mathbb{A}_i$ **do**
      for $a \in \mathcal{A}(\delta_1(s, l))$ **do**
        $\delta^{s_i}(s, (l, a)) \leftarrow \delta_2(\delta_1(s, l))$
      end for
    end for
  end for
end procedure

Algorithm 4 Subroutine: Determine Unambiguous Actions

Input
  $\phi^g = (G, (L \times \mathbb{A}), \delta^g, s_0^g, F^g)$ // (Implicit input) A centralized shield
  $\mathcal{R} : [L \to \mathbb{S}]$ // The mapping of next states, given the next label
  $s : \mathbb{S}, l : L$ // The state and label to check for ambiguity
  $A : \mathcal{P}(\mathbb{A})$ // A set of actions to check for ambiguity
procedure UNAMBIGUOUSACTIONS($\mathcal{R}, s, l, A$)
  for $a \in A$ **do**
    $ns \leftarrow \delta^g(s, (l, a))$
    for $l \in \mathcal{L}(ns)$ **do**
      if NOTEMPTY($\mathcal{R}[l]$) **and** $\mathcal{R}[l] \neq ns$ **then**
        return false
      end if
      $\mathcal{R} \leftarrow \mathcal{R} \cup [l \to ns]$
    end for
  end for
  return true
end procedure

- $\mathbb{S}^c = \mathbb{S} \times G$
- $\mathbb{A}^c = \mathbb{A}$
- $\mathcal{E}^c((s, q^g)) = \mathcal{E}(s) \cap \mathcal{C}^{\phi^g}(q^g, f(s))$
- $T^c((s', q'^g)|(s, q^g), a) = T(s'|s, a) \cdot \mathbb{1}_{q'^g = \delta^g(q^g, (f(s), a))}$
- $\gamma^c = \gamma$
- $R^c((s, q^g), a) = R(s, a)$
- $Pr[b_0^c = ((s, q^g))] = Pr[b_0 = s] \cdot \mathbb{1}_{q^g = s_0^g}$

Algorithm 1 outputs two objects; an "Allowed Action Function" $\mathcal{D}_i$ for each agent i, and a "Reverse State Mapping" $\mathcal{R}$. It is possible to compose a MMDP with these objects directly, as follows:

Definition 6 (MMDP Composition with Allowed Action Functions and Reverse State Mappings) *Given an MMDP $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$, label set L, an abstraction function f over $\mathcal{M}$ and L, centralized shield $\phi^g = (G, (L \times \mathbb{A}), \delta^g, s_0^g, F^g)$, a set of allowed action functions for each agent $\mathcal{D}_1, \ldots, \mathcal{D}_n$, and a Reverse State Mapping $\mathcal{R}$, the composition of $\mathcal{M}$, ϕ^g, $\mathcal{R}$, and each $\mathcal{D}_i$ is a new MMDP denoted as $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R}) = (\mathbb{D}^*, \mathbb{S}^*, \mathbb{A}^*, \mathcal{E}^*, T^*, \gamma^*, R^*, b_0^*)$ where*

- $\mathbb{D}^* = \mathbb{D}$
- $\mathbb{S}^* = \mathbb{S} \times ((G \times L) \cup s_{init})$
- $\mathbb{A}^* = \mathbb{A}$
- $\mathcal{E}^*((s, s_{init})) = \mathcal{E}(s) \cap (\mathcal{D}_1(s_0^g, f(s)) \times \ldots \times \mathcal{D}_n(s_0^g, f(s)))$
- $\mathcal{E}^*((s, (q^g, l_{t-1}))) = \mathcal{E}(s) \cap (\mathcal{D}_1(\mathcal{R}(q^g, l_{t-1})[f(s)], f(s)) \times \ldots \times \mathcal{D}_n(\mathcal{R}(q^g, l_{t-1})[f(s)], f(s)))$
- $T^*((s', (q'^g, l_t))|(s, s_{init}), a) = T(s'|s, a) \cdot \mathbb{1}_{q'^g = s_0^g} \cdot \mathbb{1}_{l_t = f(s)}$
- $T^*((s', (q'^g, l_t))|(s, (q^g, l_{t-1})), a) = T(s'|s, a) \cdot \mathbb{1}_{q'^g = \mathcal{R}(q^g, l_{t-1})[f(s)]} \cdot \mathbb{1}_{l_t = f(s)}$
- $\gamma^* = \gamma$
- $R^*((s, \cdot), a) = R(s, a)$
- $Pr[b_0^* = ((s, x))] = Pr[b_0 = s] \cdot \mathbb{1}_{x = s_{init}}$

Algorithm 2 likewise outputs a *transient-state decentralized shield* per agent, defined in Section 5. We define the composition of a MMDP with a set of transient-state decentralized shields as follows:

Definition 7 (Transient-State-Shield-MMDP Composition) *Given an MMDP* $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$*, label set* L*, an abstraction function* f *over* $\mathcal{M}$ *and* L*, and a set of* $|\mathbb{D}|$ *transient-state individual shields* $\phi_i^t = (Q_{1,i}, L, \delta_{1,i}, Q_{2,i}, \mathbb{A}_i, \mathcal{A}_i, \delta_{2,i}, s_{0,i})$*, the composition of* $\mathcal{M}$ *and* $(\phi_1^t, \ldots, \phi_n^t)$ *is a new MMDP denoted as* $\mathcal{M}||_t(\phi_1^t, \ldots, \phi_n^t) = (\mathbb{D}^t, \mathbb{S}^t, \mathbb{A}^t, \mathcal{E}^t, T^t, \gamma^t, R^t, b_0^t)$ *where*

- $\mathbb{D}^t = \mathbb{D}$
- $\mathbb{S}^t = \mathbb{S} \times (Q_{1,1} \times \ldots \times Q_{1,n})$
- $\mathbb{A}^t = \mathbb{A}$
- $\mathcal{E}^t((s, (q_1, \ldots, q_n))) = \mathcal{E}(s) \cap (\mathcal{A}_1(\delta_{1,1}(q_1, f(s))) \times \ldots \times \mathcal{A}_n(\delta_{1,n}(q_n, f(s))))$
- $T^t((s', (q'_1, \ldots, q'_n))|(s, (q_1, \ldots, q_n)), (a_1, \ldots, a_n)) = T(s'|s, (a_1, \ldots, a_n)) \cdot \prod_{i \in \mathbb{D}} \mathbb{1}_{q'_i = \delta_{2,i}(\delta_{1,i}(q_i, f(s)))}$
- $\gamma^t = \gamma$
- $R^t((s, \cdot), a) = R(s, a)$
- $Pr[b_0^t = ((s, (q_1, \ldots, q_n)))] = Pr[b_0 = s] \cdot \prod_{i \in \mathbb{D}} \mathbb{1}_{q_i = s_{0,i}}$

Finally, the output of Algorithm 3 is an individual shield for each agent; collectively, these form a decentralized shield. As stated in Section 5, the product of a set of individual shields can be taken to obtain a centralized shield. Unlike a standard automaton product, we synchronize on the labels:

Definition 8 (Action Product) *Given 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 label set* L*, two agents* $i, j \in \mathbb{D}$*, and two individual shields* $\phi^{s_i} = (Q^{s_i}, (L \times \mathbb{A}_i), \delta^{s_i}, s_0^{s_i}, F^{s_i}), \phi^{s_j} = (Q^{s_j}, (L \times \mathbb{A}_j), \delta^{s_j}, s_0^{s_j}, F^{s_j})$*, the* Action Product *of* ϕ^{s_i} *and* ϕ^{s_j} *is a new DFA denoted as* $\phi^{s_i} \boxtimes \phi^{s_j} = (Q^{s_p}, (L \times \mathbb{A}_p), \delta^{s_p}, s_0^{s_p}, F^{s_p})$ *where*

- $Q^p = Q^{s_i} \times Q^{s_j}$
- $\mathbb{A}_p = \mathbb{A}_i \times \mathbb{A}_j$
- $\delta^{s_p}((q_i, q_j), (l, a_i, a_j)) = (\delta^{s_i}(q_i, (l, a_i)), \delta^{s_j}(q_j, (l, a_j)))$
- $s_0^{s_p} = (s_0^{s_i}, s_0^{s_j})$
- $F^{s_p} = F^{s_i} \times F^{s_j}$

This results in a regular DFA which can be composed with a MMDP.

We will also use the notion of an *underlying environment trace*.

Definition 9 *Given an environment trace* $(((s_0, \cdot), \ldots, (s_t, \cdot)), (a_0, \ldots, a_{t-1}))$ *for* $\mathcal{M}||\phi^g$*, the* underlying environment trace *for* $\mathcal{M}$ *is* $((s_0, \ldots, s_t), (a_0, \ldots, a_t))$.

Definition 10 *Given an environment trace* $(((s_0, \cdot), \ldots, (s_t, \cdot)), (a_0, \ldots, a_{t-1}))$ *for* $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$*, the* underlying environment trace *for* $\mathcal{M}$ *is* $((s_0, \ldots, s_t), (a_0, \ldots, a_t))$.

Definition 11 *Given an environment trace* $(((s_0, \cdot), \ldots, (s_t, \cdot)), (a_0, \ldots, a_{t-1}))$ *for* $\mathcal{M}||_t\phi^t_{1..n}$*, the* underlying environment trace *for* $\mathcal{M}$ *is* $((s_0, \ldots, s_t), (a_0, \ldots, a_t))$.

B.2 Proof of correctness

We restate the following theorem from Section 5:

Theorem 1 *Given a MMDP* $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$ *where* $\forall s \in \mathbb{S}, \mathcal{E}(s) = \mathbb{A}$*, and a centralized shield* ϕ^g *for* $\mathcal{M}$*, the result of Algorithms 1, 2, and 3 is a decentralized shield for* $\mathcal{M}$ *which enforces* ϕ^g*.*

Note the assumption that the underlying MMDP allows all joint actions. Without this assumption, it is possible to construct a MMDP, abstraction function, and centralized shield for which there does not exist any deadlock-free decentralized shield as follows:

- $\mathbb{D} = \{1, 2\}$
- $\mathbb{S} = \{s_0, s_1\}$
- $\mathbb{A} = \{a_0, a_1\} \times \{a_0, a_1\}$
- $\mathcal{E}(s_0) = \{(a_0, a_1)\}$
- $\mathcal{E}(s_1) = \{(a_1, a_0)\}$
- $T(s_0|\cdot) = T(s_1|\cdot) = 0.5$
- $L = \{l_0\}, f(\cdot) = l_0$
- $G = \{q_0, q_1\}$
- $\delta^g(\cdot, (l_0, (a_0, a_1))) = \delta^g(\cdot, (l_0, (a_1, a_0))) = q_0$
- $\delta^g(\cdot, (l_0, (a_1, a_1))) = \delta^g(\cdot, (l_0, (a_0, a_0))) = q_1$
- $F^g = \{q_0\}$

In short, the environment only allows either the joint action (a_0, a_1) in state s_0, or the the joint action (a_1, a_0) in state s_1, but not both at the same state. However, because the labels in both s_0 and s_1 are the same, it is impossible for a shield to distinguish between the two cases. A decentralized shield cannot allow both of these joint actions in a single shield state, but not (a_0, a_0) or (a_1, a_1), because the agents could not communicate in advance to determine which joint action to choose—the set is not Cartesian. Therefore, any decentralized shield can allow at most one of these actions, meaning that the composition of the MMDP with the decentralized shield may deadlock.

This assumption is used for the rest of the proof. We note that many standard RL problems follow this assumption.

To prove Theorem 1, we must show:

Lemma 1 *Given a MMDP* $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$*, a centralized shield* ϕ^g *for* $\mathcal{M}$*, and the decentralized shield* $D = (\phi^{s_1} \times \ldots \times \phi^{s_n})$ *resulting from the applications of Algorithms 1, 2, and 3 to* ϕ^g*, all possible underlying environment traces allowed by* $\mathcal{M}||D$ *are also allowed by* $\mathcal{M}||\phi^g$*.*

Lemma 2 *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* $\mathcal{M}$*, the decentralized shield* $D = (\phi^{s_1} \times \ldots \times \phi^{s_n})$ *resulting from the applications of Algorithms 1, 2, and 3 to* ϕ^g *does not contain any deadlocks.*

Lemma 1 is sufficient to show that $\mathcal{M}||D \vDash \phi^g$; for every underlying environment trace which could be produced by $\mathcal{M}||\phi^g$, the corresponding label trace must be accepted by ϕ^g. The combination of both lemmas is sufficient to show that D is a shield for $\mathcal{M}$ that enforces ϕ^g.

We proceed by analyzing each algorithm individually.

Lemma 3 *Given a MMDP $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$, a centralized shield ϕ^g for $\mathcal{M}$, and the outputs $\mathcal{D}_{1..n}, \mathcal{R}$ of Algorithm 1 with ϕ^g as the input, all underlying environment traces which are possible in $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ are also possible in ϕ^g.*

We use an inductive proof to relate the environment traces of the MMDPs:

The initial state of $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ is always of the form (s_0, s_{init}). The initial state of $\mathcal{M}||\phi^g$ is likewise (s_0, s_0^g), with the same distribution over possible values of s_0.

Lemma 4 *Given $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ with an initial state of (s_0, s_{init}) and $\mathcal{M}||\phi^g$ which is in state (s_0, s_0^g), all joint actions which are allowed in $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ are also allowed in $\mathcal{M}||\phi^g$.*

Proof: The allowed actions in $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ are $\mathcal{E}(s) \cap (\mathcal{D}_1(s_0^g, f(s)) \times \ldots \times \mathcal{D}_n(s_0^g, f(s)))$, while the allowed actions in $\mathcal{M}||\phi^g$ are $\mathcal{E}(s) \cap \mathcal{C}^{\phi^g}(s_0^g, f(s))$. It is therefore sufficient to show $(\mathcal{D}_1(s_0^g, f(s)) \times \ldots \times \mathcal{D}_n(s_0^g, f(s))) \subseteq \mathcal{C}^{\phi^g}(s_0^g, f(s))$. Because we assume ϕ^g is a centralized shield, it is guaranteed that $f(s) \in \mathcal{L}(s_0^g)$, and so we can use the following lemma.

Lemma 5 *Given a centralized shield $\phi^g = (G, (L \times \mathbb{A}), \delta^g, s_0^g, F^g)$ and the outputs $\mathcal{D}_{1..n}$ of Algorithm 1 applied to ϕ^g, $\forall q^g \in G, l \in \mathcal{L}(q^g), (\mathcal{D}_1(q^g, l) \times \ldots \times \mathcal{D}_n(q^g, l)) \subseteq \mathcal{C}^{\phi^g}(q^g, l)$.*

Proof: This invariant is established on line 10 of Algorithm 1, when $\mathcal{D}_1(q^g, l) \times \ldots \times \mathcal{D}_n(q^g, l)$ is set to an arbitrary element of $\mathcal{C}^{\phi^g}(q^g, l)$. $\mathcal{D}_1(q^g, l) \times \ldots \times \mathcal{D}_n(q^g, l)$ only expands on line 16; however, the check on line 15 means that all new elements of this set are also known to be in $\mathcal{C}^{\phi^g}(q^g, l)$.

We return to the base case of the inductive proof. Every possible second state of $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ is of the form $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ $(s_1, (s_0^g, f(s)))$. Every second state of $\mathcal{M}||\phi^g$ after the same action is taken has the form of (s_1, q_1), where s_1 has the same distribution. Using line 17 of Algorithm 1, we establish the invariant that $q_1 = \mathcal{R}(s_0^g, f(s_0))[f(s_1)]$. The check on line 15 ensures that there is only a single possible q_1 for any a_0.

For the inductive case, we would like to prove the following:

Lemma 6 *If $((s_t, (q_{t-1}, f(s_{t-1}))), a_t, (s_{t+1}, (q_t, f(s_t))))$ is a possible transition of $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ where $q_t = \mathcal{R}(q_{t-1}, f(s_{t-1}))[f(s_t)]$, then $((s_t, q_t), a_t, (s_{t+1}, q_{t+1}))$ is a possible transition of $\mathcal{M}||\phi^g$ such that $q_{t+1} = \mathcal{R}(q_t, f(s_t))[f(s_{t+1})]$.*

First, we need to show that if a_t is allowed by $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$, then it is also allowed by $\mathcal{M}||\phi^g$.

The allowed actions in $\mathcal{M}||\phi^g$ are $\mathcal{E}(s) \cap \mathcal{C}^{\phi^g}(q^g, f(s))$, similar to as in the base case. The allowed actions in $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ are $\mathcal{E}(s_t) \cap (\mathcal{D}_1(\mathcal{R}(q_{t-1}, f(s_{t-1}))[f(s_t)], f(s_t)) \times \ldots \times \mathcal{D}_n(\mathcal{R}(q^g, f(s_{t-1}))[f(s_t)], f(s_t)))$. Using the invariant of $q_t = \mathcal{R}(q_{t-1}, f(s_{t-1}))[f(s_t)]$, the same application of lemma 5 suffices here.

Likewise, the check on line 15 and the assignment on line 17 again ensure that $q_{t+1} = \mathcal{R}(q_t, f(s_t))[f(s_{t+1})]$.

Lemma 7 *Given a MMDP $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$, a centralized shield ϕ^g for $\mathcal{M}$, and the outputs $\mathcal{D}_{1..n}$ and $\mathcal{R}$ of Algorithm 1 with ϕ^g as the input, $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ does not deadlock.*

For each reachable state q and observable label l, line 10 of Algorithm 1 initializes $(\mathcal{D}_1(q, l) \times \ldots \times \mathcal{D}_n(q, l))$ to contain some arbitrary element of $\mathcal{C}(q, l)$. Because ϕ^g is a centralized shield and thus does not deadlock, $\mathcal{C}(q, l)$ always contains at least one element. Therefore, $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ always has a joint action is always available, so it does not deadlock either.

Now that we have proven that Algorithm 1 maintains the safety and deadlock-free properties of the input centralized shield, we must do the same for Algorithms 2 and 3.

Lemma 8 *Given an MMDP $\mathcal{M}$, an abstraction function f over $\mathcal{M}$ and L, a centralized shield ϕ^g for $\mathcal{M}$, the output of Algorithm 1 $\mathcal{D}_i, \mathcal{R}$ given ϕ^g, and the output of Algorithm 2 ϕ_i^t given $\phi^g, \mathcal{D}_i, \mathcal{R}$: the same set of underlying environment traces are allowed by $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ as in $\mathcal{M}||_t\phi_{1..n}^t$.*

Note that this lemma implies that $\mathcal{M}||_t\phi_{1..n}^t$ is deadlock-free, as it says that the same set of underlying environment traces are allowed, not just a subset. We again use an inductive proof here.

For the base case, we must show that all possible first transitions $((s, (s_{init}, \ldots, s_{init})), a, (s', ((s_0^g, f(s)), \ldots, (s_0^g, f(s)))))$ allowed by $\mathcal{M}||_t\phi_{1..n}^t$ are also allowed as the first transition $((s, s_{init}), a, (s', (s_0^g, f(s))))$ in $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$, and vice-versa.

There are two parts to this: showing that the enabled actions are the same in both constructions; and showing that after taking the same action, the possible next states correspond.

By definition, the enabled actions in $\mathcal{M}||_t\phi_{1..n}^t$ are $(\mathcal{A}_1(\delta_{1,1}(s_{init}, f(s))) \times \ldots \times \mathcal{A}_n(\delta_{1,n}(s_{init}, f(s))))$; in $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$, the enabled actions are $(\mathcal{D}_1(s_0^g, f(s)) \times \ldots \times \mathcal{D}_n(s_0^g, f(s)))$.

Therefore, we must show that

$$(\mathcal{A}_1(\delta_{1,1}(s_{init}, f(s))) \times \ldots \times \mathcal{A}_n(\delta_{1,n}(s_{init}, f(s)))) = (\mathcal{D}_1(s_0^g, f(s)) \times \ldots \times \mathcal{D}_n(s_0^g, f(s)))$$

It suffices to show that for all i,

$$\begin{aligned} \mathcal{A}_i(\delta_{1,i}(s_{init}, f(s))) &= \mathcal{D}_i(s_0^g, f(s)) \\ \mathcal{A}_i((s_0^g, f(s))_{Q_2}) &= \mathcal{D}_i(s_0^g, f(s)) \qquad \text{(Algorithm 2, line 19)} \end{aligned}$$

This final equality is true due to Algorithm 2, line 16.

The next state in $\mathcal{M}||_t\phi_{1..n}^t$ is $(s', ((\delta_{2,1}(\delta_{1,1}(s_{init}, f(s))), f(s)), \ldots, (\delta_{2,n}(\delta_{1,n}(s_{init}, f(s))), f(s))))$, with s' chosen from the distribution $T(s'|s, a)$. Due to Algorithm 2, lines 19 and 15, this is equal to $(s', ((s_0^g, f(s)), \ldots, (s_0^g, f(s))))$, with s' chosen from the same distribution.

Correspondingly, the possible next states of $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ are of the form $(s', (s_0^g, f(s))$, where s' follows the same distribution.

For the inductive step, we first need to show that the available actions are the same for state $(s_t, (q_{t-1}, f(s_{t-1}))$ of $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$, and state $(s_t, ((q_{t-1}, f(s_{t-1})), \ldots, q_{t-1}, f(s_{t-1})))$ of $\mathcal{M}||_t\phi_{1..n}^t$.

In $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$, the possible actions are $\mathcal{E}(s_t) \cap (\mathcal{D}_1(\mathcal{R}(q_{t-1}, f(s_{t-1}))[f(s_t)], f(s_t)) \times \ldots \times \mathcal{D}_n(\mathcal{R}(q_{t-1}, f(s_{t-1}))[f(s_t)], f(s_t)))$. The enabled actions in $\mathcal{M}||_t\phi_{1..n}^t$ are $\mathcal{E}(s_t) \cap (\mathcal{A}_1(\delta_{1,1}((q_{t-1}, f(s_{t-1})), f(s_t))) \times \ldots \times \mathcal{A}_n(\delta_{1,n}((q_{t-1}, f(s_{t-1})), f(s_t))))$.

It is thus sufficient to show that $\forall i \in \mathbb{D}$

$$\begin{aligned} \mathcal{D}_i(\mathcal{R}(q_{t-1}, f(s_{t-1}))[f(s_t)], f(s_t))) &= \mathcal{A}_i(\delta_{1,i}((q_{t-1}, f(s_{t-1})), f(s_t))) \\ &= \mathcal{A}_i(\mathcal{R}(q_{t-1}, f(s_{t-1}))[f(s_t)]) \qquad \text{(Lines 12 and 13)} \end{aligned}$$

Line 16 proves this equality.

Finally, we must show that after taking the same action, the possible next states of $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ and $\mathcal{M}||_t\phi_{1..n}^t$, respectively are $(s_{t+1}, (q_t, f(s_t))$ and $(s_{t+1}, ((q_t, f(s_t)), \ldots, q_t, f(s_t)))$ for some value of q_t, and with s_{t+1} picked from the same distribution; this re-establishes the state at the start of the inductive step.

The next state in $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ is $(s_{t+1}, (\mathcal{R}(q_{t-1}, f(s_{t-1}))[f(s_t)], f(s_t)))$ with s_{t+1} picked from the distribution $T(s_{t+1}|s_t, a)$. The next state in $\mathcal{M}||_t\phi_{1..n}^t$ is $(s_{t+1}, ((\delta_{2,1}(\delta_{1,1}((q_{t-1}, f(s_{t-1})), f(s_t)))), \ldots, \delta_{2,n}(\delta_{1,n}((q_{t-1}, f(s_{t-1})), f(s_t)))))$, with s_{t+1} also picked from the same distribution.

Now, we just need to show that $\forall i \in \mathbb{D}$,

$$(\mathcal{R}(q_{t-1}, f(s_{t-1}))[f(s_t)], f(s_t)) = \delta_{2,i}(\delta_{1,i}((q_{t-1}, f(s_{t-1})), f(s_t)))$$
$$= \delta_{2,i}((\mathcal{R}(q_{t-1}, f(s_{t-1}))[f(s_t)], f(s_t))) \quad \text{(Lines 12 and 13)}$$

Algorithm 2 line 17 proves this identity, concluding the proof that this algorithm preserves safety and deadlock-freedom.

Lemma 9 *Given an MMDP $\mathcal{M}$, an abstraction function f over $\mathcal{M}$ and L, a centralized shield ϕ^g for $\mathcal{M}$, the outputs $\mathcal{D}_{1..n}, \mathcal{R}$ of Algorithm 1 given ϕ^g, the outputs $\phi^t_{1..n}$ of Algorithm 2 given $\phi^g, \mathcal{D}_{1..n}, \mathcal{R}$, and the outputs $\phi^{s_{1..n}}$ of Algorithm 3 given $\phi^t_{1..n}$, the same set of underlying environment traces are possible in $\mathcal{M}||(\phi^{s_1} \boxtimes \ldots \boxtimes \phi^{s_n})$ as are possible in $\mathcal{M}||_t\phi^t_{1..n}$.*

We again use an inductive proof here. However, as the first transition is treated no differently than the others for both $\mathcal{M}||_t\phi^t_{1..n}$ and $\mathcal{M}||(\phi^{s_1} \boxtimes \ldots \boxtimes \phi^{s_n})$, we are able to induct over states, rather than transitions.

For the base case, all possible starting states in both $\mathcal{M}||_t\phi^t_{1..n}$ and $\mathcal{M}||(\phi^{s_1} \boxtimes \ldots \boxtimes \phi^{s_n})$ are of the form $(s_0, (s_{init}, \ldots, s_{init}))$ such that, in the underlying MDP $\mathcal{M} = (\mathbb{D}, \mathbb{S}, \mathbb{A}, \mathcal{E}, T, \gamma, R, b_0)$, $\Pr[b_0 = s_0] > 0$.

For the inductive case, $\mathcal{M}||_t\phi^t_{1..n}$ and $\mathcal{M}||(\phi^{s_1} \boxtimes \ldots \boxtimes \phi^{s_n})$ begin in the same state $(s_t, (q^1_t, \ldots, q^n_t))$. If the same label is observed in the environment, we would like to show that same set of actions are enabled in both constructions, and after a given enabled action is taken, the set of possible next states are the same.

The enabled actions in $\mathcal{M}||_t\phi^t_{1..n}$ are $\mathcal{E}(s_t) \cap (\mathcal{A}_1(\delta_{1,1}(q^1_t, f(s_t))) \times \ldots \times \mathcal{A}_n(\delta_{1,n}(q^n_t, f(s_t))))$. The enabled actions in $\mathcal{M}||(\phi^{s_1} \boxtimes \ldots \boxtimes \phi^{s_n})$ are $\mathcal{E}(s_t) \cap \mathcal{C}^{\phi^{s_1} \boxtimes \ldots \boxtimes \phi^{s_n}}((q^1_t, \ldots, q^n_t), f(s_t)) = \mathcal{E}(s_t) \cap (\mathcal{C}^{\phi^{s_1}}(q_t, f(s_t)) \times \ldots \mathcal{C}^{\phi^{s_n}}(q_t, f(s_t)))$.

Therefore, it is sufficient to show that $\forall i \in \mathbb{D}, \mathcal{C}^{\phi^{s_i}}(q_t, f(s_t)) = \mathcal{A}_i(\delta_{1,i}(q_t, f(s_t)))$. This equality holds due to Algorithm 3, lines 9 and 10.

In $\mathcal{M}||_t\phi^t_{1..n}$, after an enabled action a_t is taken, the next state is $(s_{t+1}, (\delta_{2,1}(\delta_{1,1}(q^1_t, f(s_t))), \ldots, \delta_{2,n}(\delta_{1,n}(q^n_t, f(s_t)))))$, with s_{t+1} picked from the distribution $T(s_{t+1}|s_t, a_t)$. After the same action is taken in $\mathcal{M}||(\phi^{s_1} \boxtimes \ldots \boxtimes \phi^{s_n})$, the next state is $(s_{t+1}, \delta^{\phi^{s_1} \boxtimes \ldots \boxtimes \phi^{s_n}}((q^1_t, \ldots, q^n_t), (f(s_t), a_t))) = (s_{t+1}, (\delta^{\phi^{s_1}}(q^1_t, (f(s_t), a^1_t)), \ldots, \delta^{\phi^{s_n}}(q^n_t, (f(s_t), a^n_t))))$, with s_{t+1} picked from the same probability distribution.

Therefore, we just need to show that $\forall i \in \mathbb{D}, \delta^{\phi^{s_i}}(q^i_t, (f(s_t), a^i_t)) = \delta_{2,i}(\delta_{1,i}(q^i_t, f(s_t)))$. This equality also holds due to Algorithm 3, lines 9 and 10.

By combining Lemmas 3, 8, and 9, Lemma 1 is proven. By combining Lemmas 7, 8, and 9, Lemma 2 is proven.

C Decentralized Shield is Maximally Permissive for Cartesian & Unambiguous Shields

We should more formally specify what it means for a shield to be Cartesian and unambiguous:

Definition 12 (Cartesian Actions) *A set of joint actions A is Cartesian iff $\exists(A_1, \ldots, A_n), A = A_1 \times \ldots \times A_n$.*

Intuitively, when a set of joint actions A is *Cartesian*, it can be decomposed into the Cartesian product of $A_1 \times \ldots \times A_n$. As long as for all $i \in \mathbb{D}$, agent i picks an action in A_i, the selected joint action will be in A. A shield is Cartesian iff $\forall s \in G, l \in \mathcal{L}(s), \mathcal{C}(s, l)$ is Cartesian.

Definition 13 (Unambiguous Actions) *Given 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 label set L, an abstraction function $f : \mathbb{S} \to L$, and centralized shield $\phi^g = (G, (L \times \mathbb{A}), \delta^g, s^g_0, F^g)$, a set of joint actions A is* unambiguous *at state $s \in G$ and label $l \in \mathcal{L}(s)$ iff*

$\forall a^1 = (a_1^1, \ldots, a_n^1) \in A, a^2 = (a_1^2, \ldots, a_n^2) \in A, \exists i \in \mathbb{D}, a_i^1 = a_i^2 \implies (\delta^g(s,(l,a_1)) = \delta^g(s,(l,a_2))) \vee \mathcal{L}(\delta^g(s,(l,a_1))) \cap \mathcal{L}(\delta^g(s,(l,a_2))) = \emptyset$.

This means that as long as *some* joint action in A was taken, it is possible to deduce the next shield state upon observing the next label, without knowing which specific joint action was taken. A shield is unambiguous iff $\forall s \in G, l \in \mathcal{L}(s), \mathcal{C}(s,l)$ is unambiguous. Algorithm 4 is a straightforward incremental check of Definition 13; if the shield is unambiguous, and only safe actions are given as input to this function, it will always return true.

If the input centralized shield is both Cartesian and unambiguous, we argue that the same set of behaviors are allowed by the resulting decentralized shield as are allowed by the input centralized shield; therefore, the decentralized shield is maximally permissive.

We modify Lemma 5 to the following:

Lemma 10 *Given a Cartesian and unambiguous centralized shield $\phi^g = (G, (L \times \mathbb{A}), \delta^g, s_0^g, F^g)$ and the outputs $\mathcal{D}_{1..n}$ of Algorithm 1 applied to ϕ^g, $\forall q^g \in G, l \in \mathcal{L}(q^g), (\mathcal{D}_1(q^g,l) \times \ldots \times \mathcal{D}_n(q^g,l)) = \mathcal{C}^{\phi^g}(q^g,l)$.*

Because the shield is Cartesian, there must exist some $\mathcal{D}_1(q^g,l), \ldots, \mathcal{D}_n(q^g,l)$ that satisfies this property; our algorithm just needs to find this set.

One invariant established during the proof lemma 5 is that at all times, $\mathcal{D}_1(q^g,l) \times \ldots \times \mathcal{D}_n(q^g,l) \subseteq \mathcal{C}(s,l)$. Because we assume $\mathcal{C}$ is Cartesian, we can expand this invariant to $\mathcal{D}_1(q^g,l) \times \ldots \times \mathcal{D}_n(q^g,l) \subseteq A_1 \times \ldots \times A_n$, where $A_1 \times \ldots \times A_n = \mathcal{C}(s,l)$. This is equivalent to $\forall i \in \mathbb{D}, \mathcal{D}_i(q^g,l) \subseteq A_i$.

Therefore, at every loop, when line 15 is evaluated, $a_i' \subseteq A_i \leftrightarrow \{a_i'\} \times \prod_{j \in \mathbb{D}, j \neq i} \mathcal{D}_j(s,l) \subseteq \mathcal{C}(s,l)$. If the shield is unambiguous as well, the conditional passes, and a_i' is added to $\mathcal{D}_i(s,l)$. Since all actions of each agent are evaluated, this means that all elements of A_i are eventually added to $\mathcal{D}_i(s,l)$, and thus $\mathcal{D}_i(s,l) = A_i$, so $\mathcal{D}_1(q^g,l) \times \ldots \times \mathcal{D}_n(q^g,l) = \mathcal{C}(s,l)$.

With Lemma 10 proved, the following lemma holds, given Lemma 4.

Lemma 11 *Given $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ with an initial state of (s_0, s_{init}) and $\mathcal{M}||\phi^g$ which is in state (s_0, s_0^g), the same set of joint actions are allowed in $\mathcal{M}||^*(\phi^g, \mathcal{D}_{1..n}, \mathcal{R})$ as in $\mathcal{M}||\phi^g$.*

We already show in the previous section that Algorithms 2 and 3 maintain the same set of behaviors; therefore, the above is sufficient to show that the entire sequence of algorithms maintains the same set of behaviors when ϕ^g is Cartesian and unambiguous.

D Time Complexity Analysis

We analyze the time complexity of Algorithm 1 as follows:

The time complexity is therefore $O(|G| \times |L|^2 \times |\mathbb{A}| \times \sum_{i \in \mathbb{D}} |\mathbb{A}_i|)$: linear with respect to the size of the centralized shield. In practice, the coefficient is fairly small, as usually $|\mathcal{L}(s)| \ll |L|$ for most shield states.

E Shield Generation Times

On a Macbook Pro with a M1 Max processor, the shield for the gridworld-collision domain required 273 seconds for synthesis, and the shield for the particle-momentum domain required 326 seconds. The vast majority of this time was spent in centralized shield generation.

F Full agent performance in the gridworld-collision domain

Algorithm 5 Time Analysis of Algorithm 1

procedure CALCULATESAFEACTIONS(ϕ^g)
  for $s \in G, l \in \mathcal{L}(s)$ **do** // Loop runs $O(|G||L|)$ times
    $\mathcal{C}(s,l) \leftarrow \{a \in \mathbb{A} | \delta^g(s,(l,a)) \in F^g\}$ // $O(|\mathbb{A}|)$
    Pick arbitrary $a = (a_1, a_2, \ldots, a_n) \in \mathcal{C}(s,l)$ // $O(1)$
    $\forall i \in \mathbb{D} : \mathcal{D}_i(s,l) = \{a_i\}$ // $O(|\mathbb{D}|)$
    $\mathcal{R}(s,l) \leftarrow [\mathcal{L}(\delta^g((s,l),a)) \rightarrow \delta^g((s,l),a)]$ // $O(|L|)$
    for $i \in \mathbb{D}$ **do** // Loop runs $O(|\mathbb{D}|)$ times
      for $a'_i \in \mathbb{A}_i$ **do** // Loop runs $O(|\mathbb{A}_i|)$ times
        $A \leftarrow \{a'_i\} \times \prod_{j \in \mathbb{D}, j \neq i} \mathcal{D}_j(s,l)$ // $O(|\mathbb{D}|)$
        if $A \subseteq \mathcal{C}(s,l)$ **and** UNAMBIGUOUSACTIONS($\mathcal{R}(s,l), s, l, A$) **then** // $O(|L||\mathbb{A}|)$
          $\mathcal{D}_i(s,l) \leftarrow \mathcal{D}_i(s,l) \cup \{a'_i\}$ // $O(1)$
          $\forall a \in A, \mathcal{R}(s,l) \leftarrow \mathcal{R}(s,l) \cup [\mathcal{L}(\delta^g(s,(l,a))) \rightarrow \delta^g(s,(l,a))]$ // $O(|\mathbb{A}|)$
        end if
      end for
    end for
  end for
  return $\mathcal{D}_i \forall i \in \mathbb{D}, \mathcal{R}$ // $O(|\mathbb{D}|)$
end procedure

Table 3: Agent performance after 2.5 million environment steps, with fixed start positions. Rows denote the shield that the agent used for training; columns denote the shield during testing.

Map	Eval Shield Shield	Centralized	Decentralized	No Shield
ISR	Centralized	90.4 ± 0.6 (0)	89.5 ± 0.4 (0)	85.3 ± 1.5 (50.6)
	Decentralized	90.0 ± 0.4 (0)	89.9 ± 0.5 (0)	87.2 ± 1.3 (25.8)
	No Shield	88.8 ± 0.9 (0)	88.6 ± 0.8 (0)	90.0 ± 0.6 (12.3)
MIT	Centralized	72.0 ± 0.8 (0)	72.0 ± 0.8 (0)	72.0 ± 0.8 (0)
	Decentralized	71.6 ± 0.3 (0)	71.6 ± 0.3 (0)	71.6 ± 0.3 (0)
	No Shield	72.8 ± 0.5 (0)	72.8 ± 0.5 (0)	72.8 ± 0.5 (0)
Pentagon	Centralized	89.0 ± 0.4 (0)	87.8 ± 0.4 (0)	81.6 ± 2.5 (66.9)
	Decentralized	83.7 ± 4.0 (0)	82.0 ± 5.8 (0)	83.2 ± 4.4 (5.4)
	No Shield	83.3 ± 2.7 (0)	80.4 ± 4.9 (0)	89.7 ± 0.3 (4.8)
SUNY	Centralized	83.3 ± 1.4 (0)	83.3 ± 1.3 (0)	83.3 ± 1.4 (0)
	Decentralized	86.6 ± 0.6 (0)	86.6 ± 0.6 (0)	86.6 ± 0.6 (0)
	No Shield	86.2 ± 0.4 (0)	86.2 ± 0.4 (0)	86.2 ± 0.4 (0)

Table 4: Agent performance after 2.5 million environment steps, with randomized start positions.

Map	Eval Shield Shield	Centralized	Decentralized	No Shield
ISR	Centralized	78.6 ± 2.2 (0)	78.1 ± 1.8 (0)	81.3 ± 1.6 (31.7)
	Decentralized	73.7 ± 2.4 (0)	76.1 ± 2.0 (0)	83.6 ± 0.6 (7.4)
	No Shield	72.1 ± 3.2 (0)	75.3 ± 3.5 (0)	83.6 ± 1.4 (2.0)
MIT	Centralized	82.6 ± 0.2 (0)	82.2 ± 0.3 (0)	82.5 ± 0.3 (1.8)
	Decentralized	83.2 ± 0.2 (0)	83.2 ± 0.2 (0)	83.2 ± 0.2 (0)
	No Shield	81.5 ± 0.9 (0)	81.3 ± 0.8 (0)	81.4 ± 0.9 (0.4)
Pentagon	Centralized	88.4 ± 0.5 (0)	87.8 ± 0.7 (0)	86.9 ± 0.8 (16.4)
	Decentralized	83.5 ± 1.9 (0)	80.3 ± 3.5 (0)	83.2 ± 2.1 (1.2)
	No Shield	88.5 ± 0.4 (0)	88.4 ± 0.3 (0)	89.1 ± 0.3 (0.9)
SUNY	Centralized	78.1 ± 0.4 (0)	77.5 ± 0.7 (0)	76.4 ± 1.1 (10.4)
	Decentralized	73.5 ± 1.7 (0)	73.1 ± 1.7 (0)	72.1 ± 2.3 (17.3)
	No Shield	74.9 ± 1.2 (0)	74.4 ± 1.0 (0)	77.3 ± 0.4 (0.4)

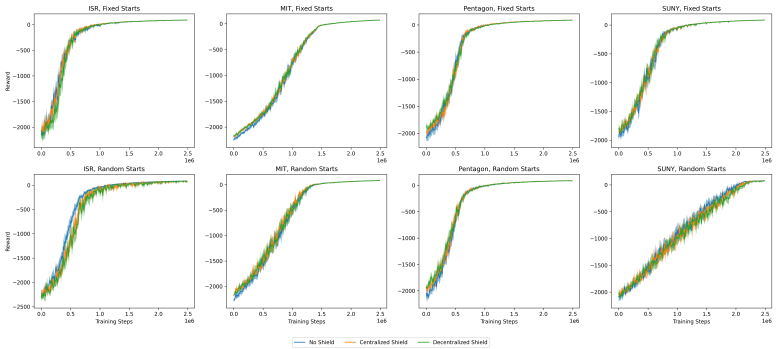

Figure 4: Training curves showing average agent returns and standard error during training, in all map configurations.

G Full agent performance in the particle-momentum domain

Table 5: Agent performance and standard error in the particle-momentum domain, using deep q-learning. The partially observable environments used 50 random seeds each, while 10 seeds were used for each fully observable case.

Obs.	Starts	Eval Shield / Shield	Centralized	Decentralized	No Shield
Partial	Fixed	Centralized	70.9 ± 5.8 (0)	62.2 ± 5.5 (0)	50.5 ± 8.0 (80.1)
		Decentralized	77.5 ± 2.8 (0)	82.8 ± 2.3 (0)	48.3 ± 12.0 (104.0)
		No Shield	76.5 ± 2.3 (0)	54.3 ± 3.9 (0)	67.8 ± 3.0 (34.3)
	Random	Centralized	70.8 ± 1.7 (0)	74.0 ± 1.6 (0)	20.7 ± 6.3 (191.6)
		Decentralized	71.8 ± 1.6 (0)	80.5 ± 1.4 (0)	7.0 ± 6.1 (235.1)
		No Shield	57.8 ± 1.9 (0)	61.6 ± 2.5 (0)	33.2 ± 2.4 (115.7)
Full	Fixed	Centralized	91.6 ± 0.0 (0)	91.0 ± 0.2 (0)	89.9 ± 0.4 (5.7)
		Decentralized	91.6 ± 0.0 (0)	91.6 ± 0.0 (0)	90.0 ± 0.3 (5.4)
		No Shield	91.6 ± 0.0 (0)	91.2 ± 0.2 (0)	90.7 ± 0.2 (3.0)
	Random	Centralized	94.6 ± 0.1 (0)	94.2 ± 0.1 (0)	93.4 ± 0.2 (4.1)
		Decentralized	94.6 ± 0.1 (0)	94.6 ± 0.1 (0)	93.5 ± 0.2 (3.8)
		No Shield	94.5 ± 0.1 (0)	94.2 ± 0.1 (0)	93.7 ± 0.2 (2.8)

H DQN Agent Architecture and Hyperparameters

Our DQN agents have a standard architecture of linear layers with ReLU nonlinearities [13]. The first linear layer is from the input (a one-hot encoding of position and momentum) to a layer size of 128. Successive linear layers are of size 128, 64, and 5, with the last layer's output corresponding to the Q-values for each action. There is no ReLU after the last layer.

We used a Double DQN architecture [22] with prioritized experience replay [19]. The agent is trained on a batch of 32 samples every 4 time steps. The target network is updated every 200 time steps. The replay buffer has a capacity of 10^5 transitions, and we use the prioritized experience replay hyperparameters $\alpha = 0.7, \beta = 0.9$. All other hyperparameters are the same as in the tabular Q-learning case. Due to the large variety of other configuration parameters—observability, starting configuration, and training and testing shields—we did not perform a formal hyperparameter search to arrive at these values.