Safe Multi-Agent Reinforcement Learning via Shielding

Ingy ElSayed-Aly
University of Virginia
Charlottesville, Virginia
ie3ne@virginia.edu

Suda Bharadwaj
University of Texas at Austin
Austin, Texas
suda.b@utexas.edu

Christopher Amato
Northeastern University
Boston, Massachusetts
c.amato@northeastern.edu

Rüdiger Ehlers
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
ruediger.ehlers@tu-clausthal.de

Ufuk Topcu
University of Texas at Austin
Austin, Texas
utopcu@utexas.edu

Lu Feng
University of Virginia
Charlottesville, Virginia
lu.feng@virginia.edu

ABSTRACT

Multi-agent reinforcement learning (MARL) has been increasingly used in a wide range of safety-critical applications, which require guaranteed safety (e.g., no unsafe states are ever visited) during the learning process. Unfortunately, current MARL methods do not have safety guarantees. Therefore, we present two shielding approaches for safe MARL. In *centralized shielding*, we synthesize a single shield to monitor all agents' joint actions and correct any unsafe action if necessary. In *factored shielding*, we synthesize multiple shields based on a factorization of the joint state space observed by all agents; the set of shields monitors agents concurrently and each shield is only responsible for a subset of agents at each step. Experimental results show that both approaches can guarantee the safety of agents during learning without compromising the quality of learned policies; moreover, factored shielding is more scalable in the number of agents than centralized shielding.

KEYWORDS

Safety; Multi-Agent Reinforcement Learning

ACM Reference Format:
Ingy ElSayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. 2021. Safe Multi-Agent Reinforcement Learning via Shielding. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) addresses sequential decision-making problems where multiple agents interact with each other in a common environment. In recent years, MARL methods have been increasingly used in a wide range of safety-critical applications from traffic management [20] to robotic control [24] to autonomous driving [19]. Existing MARL methods [12, 25] focus mostly on optimizing policies based on returns, none of which can guarantee safety (e.g., no unsafe states are ever visited) during the learning process. Nevertheless, learning with provable safety guarantees is necessary for many safety-critical MARL applications where the agents (e.g., robots, autonomous cars) may break during the exploration process and lead to catastrophic outcomes.

A recent work [1] developed a shielding framework for single-agent reinforcement learning (RL), which synthesizes a shield to enforce the correctness of safety specifications in linear temporal logic (LTL) [17]. The shield guarantees safety during learning by monitoring the RL agent's actions and preventing the exploration of any unsafe action that violates the LTL safety specification. In this paper, we adapt the shielding framework to the multi-agent setting. Guaranteeing safety for multiple agents with potentially competing goals is more challenging than the single-agent setting, because safety is an emergent property that concerns the coupling of all agents. In addition, the combinatorial nature of MARL (i.e., the joint state space and joint action space increase exponentially with the number of agents) poses scalability issues to the computation of shields.

We present in this paper the first work to provide safety guarantees (expressed as LTL specifications) for MARL. Our contributions are threefold. First, we develop a *centralized shielding* approach for MARL, where we synthesize a single shield to centrally monitor the joint actions of all agents. The shield determines that a joint action is safe if all agents satisfy the safety specification. We follow the *minimal interference* principle proposed in [1]; that is, a shield should restrict the agents as infrequently as possible and only corrects the actions that violate the safety specification. Moreover, we introduce an additional interpretation of minimal interference in the multi-agent setting: a shield should change the actions of as few agents as possible when correcting an unsafe joint action. The centralized shielding approach has limited scalability, because the computational cost of synthesizing shields depends on the number of MARL agents and the complexity of the safety specification.

Second, we develop a *factored shielding* approach for MARL to address the aforementioned scalability issues. The factored shielding offers a divide-and-conquer approach: multiple shields are computed based on a factorization of the joint state space observed by all agents. The set of factored shields monitors agents concurrently and each shield is only responsible for a subset of agents at each step. Agents can join or leave a factored shield at any time depending on their states. Factored shields enforce the correctness of safety specification by preventing unsafe actions similarly to the centralized shield. While each individual factored shield can only monitor a limited number of agents due to the restriction of shield computation, we can employ as many shields as needed; and together the set of factored shields can monitor a large number of MARL agents.

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

Third, we showcase the performance of the two shielding approaches via experimental evaluation on six benchmark problems in a grid world [16] and a cooperative navigation [23] environment. We used two MARL algorithms, CQ-learning [7] and MADDPG [15], in our experiments to demonstrate that the shielding approaches are compatible with different MARL algorithms. Experimental results show that the two shielding approaches can both guarantee the safety of agents during learning without compromising the quality of learned policies; moreover, factored shielding is more scalable in the number of agents than centralized shielding.

2 RELATED WORK

Safe reinforcement learning (RL) is an active research area, but existing results focus mostly on the single-agent setting [9], while safe MARL is still a relatively uncharted territory [25]. To the best of our knowledge, this paper presents the first safety-constrained MARL method. The survey in [9] classifies safe RL methods into two categories: (1) transforming the optimization criterion with a safety factor, such as the worst case criterion, risk-sensitive criterion, or constrained criterion; and (2) modifying the exploration process through the incorporation of external knowledge (e.g., demonstrations, teacher advice) or the guidance of a risk metric. Our shielding approaches fall into the second category. In particular, shields act similarly to a teacher who provides information (e.g., safe actions) to the learner when necessary (e.g., unsafe situations are detected). The concept of shielding was introduced to RL for the single-agent setting in [1]. In this work, we adapt the shielding framework for MARL via addressing challenges such as the coupling of agents and scalabilty issues in the multi-agent setting.

Different safety objectives for RL have been considered in the literature, such as the variance of the return, or limited visits of error states [9]. In this work, we synthesize shields that enforce safety specifications expressed in linear temporal logic (LTL) [17], which is a commonly used specification language in formal methods for safety-critical systems [2, 3]. For example, LTL has been used to express complex task specifications for robotic planning and control [13, 22]. Several recent works [6, 10, 11] have developed reward shaping techniques that translate logical constraints expressed in LTL to reward functions for RL. However, as we demonstrated in our experiments (Section 6), relying on reward functions only is not sufficient for MARL methods to learn policies that guarantee the safety (e.g., no collisions).

The shield synthesis technique based on solving two-player safety games was developed in [5] for enforcing safety properties of a system at runtime, and was adopted in [1] to synthesize shields for single-agent RL. We further adapt this technique to synthesize centralized and factored shields for MARL in this paper. There are a few recent works [4, 18] considering the shield synthesis for multi-agent (offline) planning and coordination, none of which are directly applicable for MARL.

3 BACKGROUND

A discrete probability *distribution* over a (countable) set S is a function $\mu : S \to [0, 1]$ such that $\sum_{s \in S} \mu(s) = 1$. Let $Distr(S)$ denote the set of distributions over S. We use $\mathbb{R}$ to denote the real numbers. Given an alphabet Σ, we denote by Σ^ω and Σ^* the set of infinite and finite words over Σ, respectively.

Multi-Agent Reinforcement Learning (MARL). We follow the Markov game formulation of MARL in [25]. A *Markov game* is a tuple $(N, S, \{A^i\}_{i \in N}, P, \{R^i\}_{i \in N}, \gamma)$ with a finite set $N = \{1, \cdots, n\}$ of agents, and a finite state space S observed by all agents; let $A := A^1 \times \cdots \times A^n$ be the set of joint actions for all agents, where A^i denotes the actions of agent $i \in N$; the probabilistic transition function $P : S \times A \to Distr(S)$ is defined over the joint states and actions of all agents; $R^i : S \times A \times S \to \mathbb{R}$ is an immediate reward function for agent i under the joint states and actions; $\gamma \in [0, 1]$ is the discount factor of future rewards. At time step t, each agent chooses an action $a_t^i \in A^i$ based on the observed state $s_t \in S$. The environment moves to state s_{t+1} with the probability $P(s_t, a_t, s_{t+1})$, where $a_t = (a_t^1, \cdots, a_t^n)$ is the joint action of all agents, and rewards agent i with $R^i(s_t, a_t, s_{t+1})$. The goal of an individual agent i is to learn a policy $\pi^i : S \to Distr(A^i)$ that optimizes the expectation of cumulative future rewards $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^i(s_t, a_t, s_{t+1})]$. The performance of individual agent i is not only influenced by its own policy, but also the choices of all other agents.

Depending on agents' goals, MARL algorithms can be categorized as fully cooperative (i.e., agents collaborate to optimize a common long-term return), fully competitive (i.e., zero-sum game among agents), or a mixed setting that involves both cooperative and competitive agents. In our experiments (Section 6), we used the following three mixed-setting algorithms. Independent Q-learning [21] is a baseline algorithm where agents learn Q-values over their own action set independently and do not use any information about other agents. CQ-learning [7] is an algorithm that allows agents to act independently most of the time and only accounts for the other agents when necessary (e.g., when conflict situations are detected). MADDPG [15] is a deep MARL algorithm featuring centralized training with decentralized execution, in which each agent trains models simulating each of the other agents' policies based on its observation of their actions.

Scalability is a key challenge of MARL, due to its combinatorial nature. For example, our experiments can only use two agents with CQ-learning, but more than four agents with MADDPG which applies deep neural networks for function approximation to mitigate the scalability issue. Another key challenge of MARL is the lack of convergence guarantees in general, except for some special settings [25]. As multiple agents learn and act concurrently, the environment faced by an individual agent becomes non-stationary, which invalidates the stationary assumption used for proving convergence in single-agent RL algorithms.

Safety Specifications and Safety Games. We use linear temporal logic (LTL) [17] to express safety specifications. In addition to propositional logical operators, LTL employs temporal operators such as $\bigcirc$ (next), U (until), $\Box$ (always), and $\Diamond$ (eventually). The set of words that satisfies an LTL formula ϕ represents a language $\mathcal{L}(\phi) \subseteq (2^{\mathsf{AP}})^\omega$, where AP is a given set of atomic propositions. LTL formulas can be used to express a wide variety of requirements. We focus on safety specifications, which are informally interpreted as "something bad should never happen". For example, the LTL formula $\Box\neg$unsafe expresses that "unsafe states should never be

visited". An LTL safe specification can be translated into a safe language accepted by a deterministic finite automaton (DFA) [14].

Formally, a *deterministic finite automaton* is a tuple $(Q, q_0, \Sigma, \delta, F)$ with a finite set of states Q, an initial state $q_0 \in Q$, a finite alphabet Σ, the transition function $\delta : Q \times \Sigma \to Q$, and a finite set of accepting states $F \subseteq Q$. Let $q_0\sigma_0 q_1\sigma_1 \cdots \in (Q \times \Sigma)^\omega$ be a run of the DFA. The word $\sigma_0\sigma_1 \ldots$ is in the safety language accepted by the DFA if the run only visits accepting states of the DFA, i.e., $q_i \in F$ for all $i \geq 0$.

We use Mealy machines to represent shields. Formally, a *Mealy machine* is a tuple $(Q, q_0, \Sigma_I, \Sigma_O, \delta, \lambda)$ with a finite set of states Q, an initial state $q_0 \in Q$, finite sets of input alphabet Σ_I and output alphabet Σ_O, the transition function $\delta : Q \times \Sigma_I \to Q$, and the output function $\lambda : Q \times \Sigma_I \to \Sigma_O$. For a given input trace $\sigma_0\sigma_1 \cdots \in \Sigma_I^\omega$, the Mealy machine generates a corresponding output trace $\lambda(q_0, \sigma_0)\lambda(q_1, \sigma_1) \cdots \in \Sigma_O^\omega$ where $q_{i+1} = \delta(q_i, \sigma_i)$ for all $i \geq 0$.

As we will describe later, we synthesize shields by solving two-player safety games. Formally, a *two-player safety game* is a tuple $(G, g_0, \Sigma_1, \Sigma_2, \delta, F)$ with a finite set of game states G, an initial state $g_0 \in G$, finite sets of alphabet Σ_1 and Σ_2 for Player 1 and Player 2 respectively, the transition function $\delta : G \times \Sigma_1 \times \Sigma_2 \to G$, and a set of safe states $F \subseteq G$ defines the *winning condition* such that a play $g_0 g_1 \ldots$ of the game is winning iff $g_i \in F$ for all $i \geq 0$. At each game state $g_i \in G$, Player 1 chooses an action $a_i^1 \in \Sigma_1$, then Player 2 chooses an action $a_i^2 \in \Sigma_2$, and the game moves to the next state $g_{i+1} = \delta(g_i, a_i^1, a_i^2)$. A memoryless strategy for Player 2 is a function $\kappa : G \times \Sigma_1 \to \Sigma_2$. A *winning region* $W \subseteq F$ is the set of states from which there exists a winning strategy (i.e., all plays constructed using the strategy satisfy the winning condition).

4 CENTRALIZED SHIELDING

We introduce a *centralized shield* (i.e., a single shield for all agents) into the traditional MARL process. In the following, we first describe how the centralized shield interacts with the learning agents and the environment to achieve safe MARL, then we present our method for synthesizing the centralized shield.

Figure 1 illustrates the interaction of the centralized shield, the MARL agents, and the environment. Algorithm 1 summarizes the centralized shield's behavior at time step t. The shield monitors the joint action $a_t = (a_t^1, \cdots, a_t^n)$ chosen by the MARL agents. If the shield detects that a_t is unsafe (i.e., violates the safety specification) at the agents' joint state $s_t \in S$, the shield substitutes a_t with a safe joint action $\bar{a}_t$; otherwise, the shield forwards a_t to the environment directly (i.e., $\bar{a}_t = a_t$). The environment receives the action $\bar{a}_t$ output by the shield, moves to state $s_{t+1} \in S$, and provides reward $R^k(s_t, \bar{a}_t, s_{t+1})$ for each agent k to update its policy. Meanwhile, the shield assigns a punishment ρ_t^k to agent k (where $\bar{a}_t^k \neq a_t^k$) to help the MARL algorithm learn about the cost of unsafe actions.

A centralized shield enforces the safety specification during the learning process (i.e., any unsafe action is corrected to a safe action before being sent to the environment). Moreover, we require the shield to restrict MARL agents as rarely as possible via the *minimal interference* criteria: (1) the shield only corrects the joint action a_t if it violates the safety specification, and (2) the shield seeks a safe joint action $\bar{a}_t$ that changes as few of the agents' actions as possible from a_t.

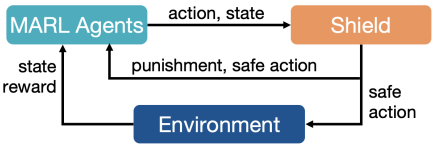

Figure 1: Safe MARL with centralized shielding.

Algorithm 1 Centralized shielding at time step t

Input: Shield $\mathcal{S}$, MARL agents' joint action $a_t = (a_t^1, \cdots, a_t^n)$ and joint state $s_t = (s_t^1, \cdots, s_t^n)$, a constant punishment cost c
Output: Safe joint action $\bar{a}_t$, punishment ρ_t

```
ρ_t ← 0
ā ← safe action output by the shield S
for all agent k such that ā^k ≠ a^k do
    ρ_t^k ← c
end for
return ā_t, ρ_t
```

Our approach synthesizes a centralized shield based on the safety specification and a coarse environment abstraction. Note that we do not require the environment dynamics to be completely known in advance. The shield can be synthesized based on a coarse abstraction of the environment that is sufficient to reason about the potential violations of safety specifications. For example, before deploying a team of robots for a disaster search and rescue mission, we may use some low-resolution satellite imagery to build a coarse, high-level abstraction about the terrain environment for shield synthesis. However, such a coarse environment abstraction is not sufficient for planning algorithms that rely on complete models of the environment. Therefore, MARL agents still need to learn about the concrete environment dynamics.

We describe how to synthesize centralized shields as follows. We assume some coarse environment abstraction has been given as a DFA $\mathcal{A}^e = (Q^e, q_0^e, \Sigma^e, \delta^e, F^e)$ with the alphabet $\Sigma^e = L \times A$, where an observation function $f : S \to L$ maps the MARL agents' joint state space S to some observation set L, and A is the joint action set of all agents. We translate the safety specification expressed as an LTL formula to another DFA $\mathcal{A}^s = (Q^s, q_0^s, \Sigma^s, \delta^s, F^s)$ with the same alphabet $\Sigma^s = L \times A$. We combine $\mathcal{A}^e$ and $\mathcal{A}^s$ into a two-player safety game $\mathcal{G} = (G, g_0, \Sigma_1, \Sigma_2, \delta^g, F)$ where $G = Q^e \times Q^s$, $g_0 = (q_0^e, q_0^s)$, $\Sigma_1 = L$, $\Sigma_2 = A$, $\delta^g((q^e, q^s), l, a) = (\delta^e(q^e, (l \times a)), \delta^s(q^s, (l \times a))$ for all $(q^e, q^s) \in G$, $l \in L$, and $a \in A$, and $F = Q^e \times F^s$. We solve the two-player safety game $\mathcal{G}$ and compute the wining region $W \subseteq F$ using the techniques described in [5]. We construct the centralized shield represented as a Mealy machine $\mathcal{S} = (Q, q_0, \Sigma_I, \Sigma_O, \delta, \lambda)$, where the state space is given by the game states $Q = G = Q^e \times Q^s$, the initial state $q_0 = g_0 = (q_0^e, q_0^s)$, the input alphabet $\Sigma_I = L \times A$, the output alphabet $\Sigma_O = A$; the transition function $\delta(g, (l, a)) = \delta^g(g, l, \lambda(g, (l, a)))$ for all $g \in G$, $l \in L$, and $a \in A$; the output function $\lambda(g, (l, a)) = a$ if $\delta^g(g, l, a) \in W$, and $\lambda(g, (l, a)) = \bar{a}$ if $\delta^g(g, l, a) \notin W$, where $\bar{a} \in A$ is a safe action with $\delta^g(g, l, \bar{a}) \in W$ and only differs from the unsafe action a in terms of the minimal number of agents' actions. We also define a (negative) constant c as punishment for unsafe actions. The computational cost of synthesizing centralized shields grows exponentially as the

Figure 2: Example grid map with two agents.

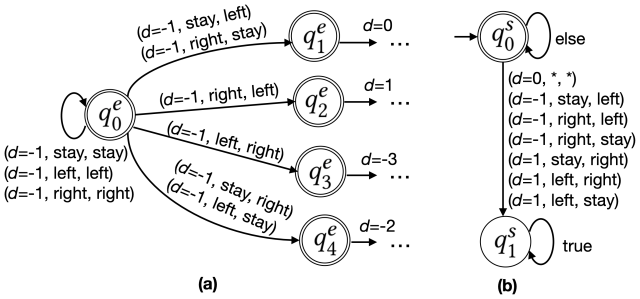

Figure 3: (a) An example environment abstraction DFA $\mathcal{A}^e$. (b) An example safety specification DFA $\mathcal{A}^s$. (Double circle denotes accepting states of DFAs. * refers to any action.)

number of agents increases, and also depends on the complexity of the safety specification and environment abstraction.

To exemplify the shield synthesis method, let us consider two agents (blue and orange) in the grid map shown in Figure 2. Each agent can move left or right, or stay in the same grid. An agent receives a reward of 10 if it reaches grid 1 or 6, and receives a negative reward of −1 if it collides with the other agent. The discount factor being $\gamma = 1$. Each agent tries to learn an optimal policy based on the observed rewards. However, the negative reward cannot completely prevent collisions during the learning process of traditional MARL algorithms. Because the agents need to explore different (even unsafe) actions to learn about states and rewards from the environment. Now we show how to construct a shield that can block unsafe actions and guarantee collision free. We use an observation set L that measures the distance d between blue and orange agents. For example, $d = -1$ for agents' positions shown in Figure 2. We build a coarse environment abstraction DFA $\mathcal{A}^e$ that captures the relation of agents' distances and joint actions. Figure 3(a) shows a fragment of $\mathcal{A}^e$. We can express the safety specification of collision avoidance using the following LTL formula:

$$\square\neg\Big((d=0)\vee\big((d=-1)\wedge((stay,left)\vee(right,left)\vee(right,stay))\big)$$
$$\vee\big((d=1)\wedge((stay,right)\vee(left,right)\vee(left,stay))\big)\Big)$$

which indicates that the following bad scenarios should never occur: two agents being in the same grid ($d = 0$), or taking certain unsafe joint actions that would make them collide into each other when $d = -1$ or $d = 1$. We can translate the LTL formula into the DFA $\mathcal{A}^s$ shown in Figure 3(b). We build a two-player safety game from the product of $\mathcal{A}^e$ and $\mathcal{A}^s$. Figure 4 shows a fragment of the safety game. For example, in the game state (q_0^e, q_0^s), the blue and orange agents should not choose a joint action (*stay*, *left*) that leads to an unsafe game state (q_1^e, q_1^s) where two agents collide into each other. The synthesized centralized shield prevents the collision by correcting the unsafe action (*stay*, *left*) with a safe action (*stay*, *stay*) and assigns a punishment cost of −1 to the orange agent.

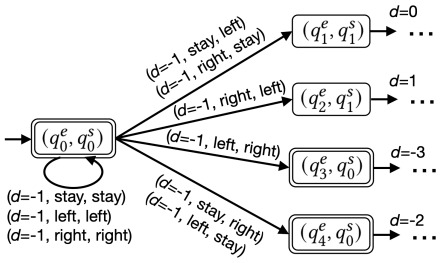

Figure 4: An example safety game given by the product of $\mathcal{A}^e$ and $\mathcal{A}^s$ shown in Figure 3. Double circles denote safe states.

Correctness. We show that the synthesized centralized shields can indeed enforce safety specifications for MARL agents as follows. Given a trace $s_0 a_0 s_1 a_1 \cdots \in (S \times A)^\omega$ jointly produced by MARL agents, the centralized shield, and the environment, there is a corresponding run $q_0 q_1 \cdots \in Q^\omega$ of the shield $\mathcal{S} = (Q, q_0, \Sigma_I, \Sigma_O, \delta, \lambda)$ such that $q_{i+1} = \delta(q_i, (f(s_i), a_i))$ and $a_i = \lambda(q_i, (f(s_i), a_i))$ for all $i \geq 0$, where $f : S \to L$ is the observation function. By the construction of the shield, we have $Q = Q^e \times Q^s$, where Q^e and Q^s are the state space of the environment abstraction DFA $\mathcal{A}^e$ and the safety specification DFA $\mathcal{A}^s$, respectively. Thus, we can project the run $q_0 q_1 \ldots$ of the shield onto a trace $q_0^s(f(s_0), a_0) q_1^s(f(s_1), a_1) \ldots$ on $\mathcal{A}^s$. The shield is constructed from the winning region of the safety game, which ensures that only safe states are ever visited along the trace $q_0^s(f(s_0), a_0) q_1^s(f(s_1), a_1) \ldots$ of $\mathcal{A}^s$ (i.e., $q_i^s \in F^s$ for all $i \geq 0$). Thus, the centralized shield $\mathcal{S}$ can guarantee that the safety specification $\mathcal{A}^s$ is never violated.

Impact on Learning Performance. The centralized shielding approach is agnostic to the choice of a MARL algorithm, because the shield interacts with the learner only via inputs and outputs, and does not rely on the inner-workings of the learning algorithm. As explained in Section 3, there is a lack of theoretical convergence guarantees for MARL algorithms in general. Thus, a full theoretical analysis of the shielding approach's impact on MARL convergence is out of scope for this paper. We show empirically in our experiments (Section 6) that (1) MARL with and without centralized shielding both converge; (2) centralized shielding can guarantee the safety in all examples, while MARL without shielding does not prevent agents' unsafe behavior; (3) centralized shielding learns more optimal policies with better returns than non-shielded MARL in some examples (e.g., due to the removal of unsafe actions that may destabilize learning).

5 FACTORED SHIELDING

The centralized shielding approach has limited scalability, because the computational cost of shield synthesis grows exponentially with the number of agents. To address this limitation, we develop a *factored shielding* approach that synthesizes multiple shields to monitor MARL agents concurrently, as illustrated in Figure 5.

Let us consider a finite set of factored shields $\{\mathcal{S}_1, \cdots, \mathcal{S}_m\}$ where each shield is synthesized based on a factorization of the joint state space observed by all agents. We can leverage problem-specific knowledge to achieve an efficient factorization scheme (e.g., how many shields to use, what is the state space covered by each

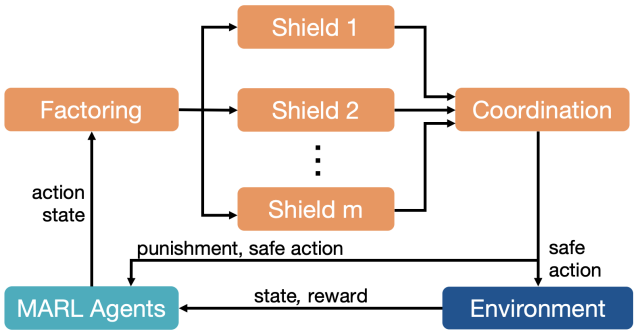

Figure 5: Safe MARL with factored shielding.

shield). For example, we synthesize two factored shields $\mathcal{S}_1$ and $\mathcal{S}_2$ for monitoring agents' behavior in grids 1-3 and 4-6 of Figure 2, respectively. A factored shield monitors a subset of agent actions at each time step. A shield is not tied to any specific agent; instead, an agent can request to join or leave a shield from border states at any time. For example, if the orange agent in Figure 2 wants to move from grid 4 to grid 3, it would request to join $\mathcal{S}_1$ and leave $\mathcal{S}_2$.

Algorithm 2 describes how the factored shielding works at each time step t. There are three phases: (1) factorization, (2) shielding, and (3) coordination. In the factorization phase (line 5-14), the algorithm identifies the factored shields that are responsible for monitoring each agent k in the current time step t, based on a mapping between the agent state s_t^k and the factored state space assigned to each shield $\mathcal{S}_i$. Thus, there must exist at least one factored shield monitoring each agent. If agent k happens to be in a border state s_t^k within the shield $\mathcal{S}_i$ and, by taking action a_t^k, the agent would cross the border to another shield $\mathcal{S}_j$, the algorithm relates agent k with both shields and renames its actions in shield $\mathcal{S}_i$ and $\mathcal{S}_j$ as *leave* and *join*, respectively. Next, in the shielding phase (line 16-33), each factored shield checks if the set of related agents act safely (i.e., not violating the safety specification within it) and substitutes any unsafe action with a default safe action (e.g., *stay* in our running example). In the coordination phase (line 35-47), the algorithm checks the output of all shields to make sure compatible decisions are made for each agent. For example, if an agent action a_t^k is translated to requests of leaving $\mathcal{S}_i$ and joining $\mathcal{S}_j$, then both requests need to be approved by the shields; however, if $\mathcal{S}_j$ considers *join* as unsafe at this time and substitutes with a default safe action *stay*, then the algorithm corrects the agent action a_t^k and output with safe action $\bar{a}_t^k = stay$, Finally, the algorithm assigns a punishment cost $\rho_t^k = c$ for any unsafe action a_t^k with $\bar{a}_t^k \neq a_t^k$.

We synthesize factored shields using a similar method as the synthesis of centralized shields. However, instead of building a safety game that accounts for the joint states S and joint actions $A = A^1 \times \cdots \times A^n$ of all MARL agents, we only consider a factorization of states and actions for the synthesis of each factored shield. Let $S_i \subseteq S$ be the factored state space of shield $\mathcal{S}_i$. We factor the coarse environment abstraction DFA $\mathcal{A}^e$ into a DFA $\mathcal{A}_i^e = (Q_i^e, q_{0,i}^e, \Sigma_i^e, \delta_i^e, F_i^e)$ with the alphabet $\Sigma_i^e = L_i \times A_i$, where an observation function $f : S_i \to L_i$ maps the factored states S_i to

Algorithm 2 Factored shielding at time step t

Input: A set of factored shields $\{\mathcal{S}_1, \cdots, \mathcal{S}_m\}$, MARL agents' joint action $a_t = (a_t^1, \cdots, a_t^n)$ and joint state $s_t = (s_t^1, \cdots, s_t^n)$, a default safe action b, a constant punishment cost c
Output: Safe joint action $\bar{a}_t$, punishment ρ_t

1: Initialize int array A2S : $n \times 2$ // related shield index
2: Initialize string array Act : $n \times 2$ // actions
3: Initialize Boolean array S2A : $m \times n$ // agents in each shield
4: // Factorization phase
5: **for all** agent $k \in \{1, \cdots, n\}$ **do**
6: find a factored shield $\mathcal{S}_i$ related to the agent state s_t^k
7: **if** (s_t^k, a_t^k) may leave shield $\mathcal{S}_i$ and join shield $\mathcal{S}_j$ **then**
8: A2S$[k][0] \leftarrow i$, A2S$[k][1] \leftarrow j$
9: Act$[k][0] \leftarrow$ "$leave$", Act$[k][1] \leftarrow$ "$join$"
10: S2A$[i][k] \leftarrow$ True, S2A$[j][k] \leftarrow$ True
11: **else**
12: A2S$[k][0] \leftarrow i$, Act$[k][0] \leftarrow a_t^k$, S2A$[i][k] \leftarrow$ True
13: **end if**
14: **end for**
15: // Shielding phase
16: **for all** shield $\mathcal{S}_i$ with $i \in \{1, \cdots, m\}$ **do**
17: $a \leftarrow \{\}$
18: **for all** k with S2A$[i][k]$ = True **do**
19: **if** A2S$[k][0] = i$ **then**
20: $a \leftarrow$ append Act$[k][0]$
21: **else**
22: $a \leftarrow$ append Act$[k][1]$
23: **end if**
24: **end for**
25: $\bar{a} \leftarrow$ safe action output by the shield $\mathcal{S}_i$
26: **for all** agent k such that $\bar{a}^k \neq a^k$ **do**
27: **if** A2S$[k][0] = i$ **then**
28: Act$[k][0] \leftarrow \bar{a}^k$
29: **else**
30: Act$[k][1] \leftarrow \bar{a}^k$
31: **end if**
32: **end for**
33: **end for**
34: // Coordination
35: **for all** agent $k \in \{1, \cdots, n\}$ **do**
36: **if** A2S$[k][1] \neq null$ **then**
37: **if** Act$[k][0] =$ "$leave$" and Act$[k][1] =$ "$join$" **then**
38: Act$[k][0] \leftarrow a_t^k$
39: **else**
40: Act$[k][0] \leftarrow b$
41: **end if**
42: **end if**
43: $\bar{a}_t^k \leftarrow$ Act$[k][0]$, $\rho_t^k \leftarrow 0$
44: **if** $\bar{a}_t^k \neq a_t^k$ **then**
45: $\rho_t^k \leftarrow c$
46: **end if**
47: **end for**
48: **return** $\bar{a}_t, \rho_t$

some observation set L_i, and $A_i = (A^1 \cup \cdots \cup A^n \cup \{join, leave\}) \times \cdots \times (A^1 \cup \cdots \cup A^n \cup \{join, leave\})$ is the joint action in shield $\mathcal{S}_i$ with $|A_i|$ determined by the maximum number of agents that shield $\mathcal{S}_i$ can monitor at once. Note that we need to translate the agent actions at border states of a shield to *join* or *leave* requests. Intuitively, since any agent may request to join or leave shield $\mathcal{S}_i$ at

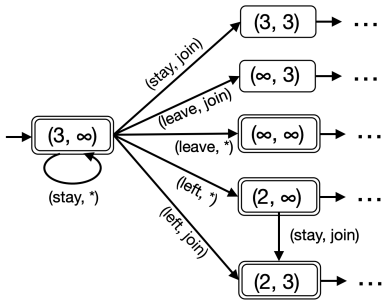

Figure 6: An excerpt of the safety game for constructing shield $\mathcal{S}_1$ of our running example. Double lines indicate safe states. To simplify the graphic notation, we put observations inside each state which should be labeled on all outgoing transitions from that state. The observations are about agents' grid positions, with ∞ denoting outside. $*$ refers to any action except "join".

any time, the joint action A_i needs to account for any possible combination of agents. This allows us to synthesize factored shields offline with a fixed alphabet, instead of re-computing shields for different agents at each step during learning. Similarly, we can factor the safety specification DFA $\mathcal{A}^s = (Q^s, q_0^s, \Sigma^s, \delta^s, F^s)$ into a DFA $\mathcal{A}_i^s = (Q_i^s, q_{0,i}^s, \Sigma_i^s, \delta_i^s, F_i^s)$ with the alphabet $\Sigma_i^s = L_i \times A_i$. We obtain the shield $\mathcal{S}_i$ as a Mealy machine by solving the two-player safety game $\mathcal{G}_i$ built from $\mathcal{A}_i^e$ and $\mathcal{A}_i^s$, in a similar way as described in Section 4.

Figure 6 shows an example safety game for synthesizing the shield $\mathcal{S}_1$ that monitors agents' actions in grid 1-3 of our running example. The initial game state observes that the blue agent is in grid 3 and the orange agent is outside the shield. If the blue and orange agents ask for a pair of actions (*stay*, *join*), then the game would move to an unsafe state where both agents collide into each other in grid 3. In this case, shield $\mathcal{S}_1$ substitutes (*stay*, *join*) with safe actions (*stay*, *stay*). Since the orange agent is involved in two shields $\mathcal{S}_1$ and $\mathcal{S}_2$, we need to coordinate the output of both shields. For example, if $\mathcal{S}_1$ rejects orange agent's *join* request but $\mathcal{S}_2$ accepts the same agent's *leave* request, then there is conflict among the output of $\mathcal{S}_1$ and $\mathcal{S}_2$. In such case, our coordination algorithm chooses the default safe action *stay* for the orange agent. Note that, if there is another agent in shield $\mathcal{S}_2$, then it should not be allowed to move to grid 4 before the orange agent successfully leaves $\mathcal{S}_2$ to avoid collision. Such safety constraints can be encoded in the safety game for synthesizing the shield $\mathcal{S}_2$.

Correctness. We show that the factored shielding algorithm can guarantee safety for MARL agents. Given a trace $s_0 a_0 s_1 a_1 \cdots \in (S \times A)^\omega$ jointly produced by MARL agents, the factored shielding, and the environment, we prove that the state-action pair (s_t, a_t) is safe at every time step t. There are several cases. First, suppose none of the agents requests to switch shields at time step t. By the construction of factored shields, each shield $\mathcal{S}_i$ monitors a subset of agents based on the factored state space $s_{t,i}$ and outputs a safe joint action $a_{t,i}$ that does not violate the safety specification. Thus, the joint state $s_t = s_{t,1} \cup \cdots \cup s_{t,m}$ and joint action $a_t = a_{t,1} \cup \cdots \cup a_{t,m}$ output by all shields are safe for all agents. Second, suppose there is some agent k requesting to leave shield $\mathcal{S}_i$ and join shield $\mathcal{S}_j$. If both shields accept agent k's requests, which means that agent k does not cause a violation of safety specification with either shield. So we still have s_t and a_t safe for all agents. If $\mathcal{S}_j$ rejects agent k's joining request and substitutes with a default safe action, then the factored shielding algorithm coordinates with the output of $\mathcal{S}_i$ and corrects agent k's leaving request with the default safe action as well. Such a correction does not affect the safety of other agents in shield $\mathcal{S}_i$, because by construction the shield accounts for the worst case scenario of leaving request being rejected. Therefore, we have the joint state-action pair (s_t, a_t) safe at every time step t for all agents.

Impact on Learning Performance. Similarly to centralized shielding, the factored shielding approach is agnostic to the choice of a MARL algorithm. We show empirically via our experiments that adding factored shields does not prevent MARL algorithms from converging. In addition, our experiments show that the factored shielding approach can be applied to examples where the synthesis of centralized shields is not feasible due to a large number of agents. While the two shielding approaches can both guarantee the safety during learning in all examples, factored shielding sometimes leads to less optimal policies than centralized shielding (e.g., due to the delay caused by agents switching shields).

6 EXPERIMENTS

We implemented both the centralized shielding and factored shielding approaches in Python and used the Slugs tool [8] to synthesize shields via solving two-player safety games. We applied our prototype implementation to six benchmark problems in the grid world (Figure 7) and a cooperative navigation environment (Figure 8). We used two MARL algorithms CQ-learning [7] and MADDPG [15] in experiments to show that our shielding approaches are agnostic to the choice of MARL algorithms. The experiments were run on a computer with Intel i5 CPU and 16 GB of RAM. Each experiment was split into training phase (linearly decreasing exploration) and evaluation phase (immediately following the training phase and with an exploration rate of 5%). All experiments were conducted for 10 independent runs whose results were averaged to reduce the impact of outliers. The shields in all examples were synthesized within two minutes.

Problem Setup. Figure 7 shows four maps of benchmark grid world examples adapted from [16]. Each map has two agents, where each agent aims to learn its own optimal policy for navigating from the start position to the target position while trying to avoid collisions. Each agent has five possible actions: *stay*, *up*, *down*, *left*, *right*. Once an agent reaches its target position, it stays there. A learning episode ends when both agents have reached their target positions. Both agents have the same reward function: -1 for a valid move, -10 for a collision with a wall, -30 for collision with the other agent, 100 for arriving at the agent's target position.

Figure 8 shows two benchmark cooperative navigation examples adapted from [23]. Each example has four agents represented as particles. The goal is for agents to cooperate and reach their designated target positions as fast as possible while avoiding collisions. We discretize the fully continuous environment in [23] by restricting agents only take positions with a precision of 0.1. An agent

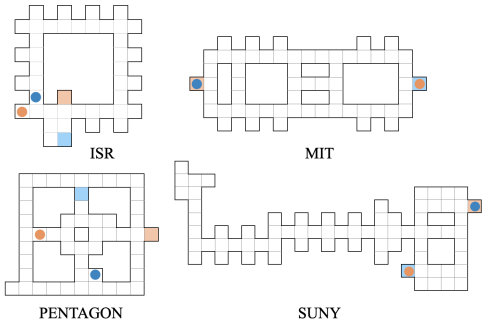

Figure 7: Maps of grid world examples adapted from [16]. In each map, blue and orange agents aim to learn optimal policies to navigate from start (circles) to target (squares) while avoiding collisions.

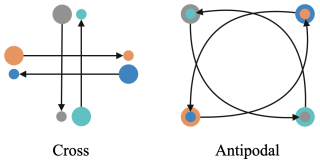

Figure 8: Visualisations of cooperative navigation examples adapted from [23]. Four agents (blue, orange, green, and grey) aim to learn optimal policies to navigate from start (large circles) to target (small circles) while avoiding collisions.

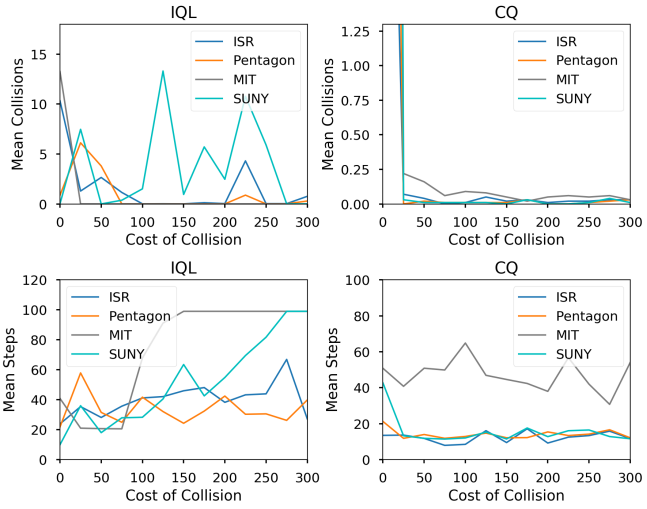

Figure 9: Collision variation experiments results (average of 10 evaluation episodes conducted after 1,000 training episodes, across 10 independent runs).

receives a higher reward when it gets closer to its target position (i.e., negation of the distance value), and a negative reward −1 for any collision.

Collision Variation Experiments. We conducted a set of experiments using the grid world examples to highlight why relying on the reward function only is not sufficient to achieve safety (i.e., collision avoidance in our examples). To prevent collisions, the traditional practice of reinforcement learning is to assign a negative reward (we refer to its absolute value as the cost of collision) whenever a collision occurs, and increase the cost until the probability of collision happening becomes negligible. Figure 9 shows the results of our experiments using the independent Q-learning [21] and CQ-learning[7]. The left side of the figure shows that, for the independent Q-learning, increasing the cost of collision cannot guarantee that the evaluation phase will be completely collision free; moreover, the increased cost of collision leads to a significant agent performance degradation measured by a larger number of steps to reach target positions. In the MIT and SUNY maps, agents even learn policies that give up the primary task of reaching target positions in order to avoid the high collision cost. The results of the CQ-learning (shown in the right side of the figure) are better than those of the independent Q-learning. The number of collisions drops quickly with a relatively low cost. However, CQ-learning cannot guarantee zero collision either (see Table 1).

Centralized Shielding Evaluation. We integrated CQ-learning with centralized shielding and applied it to the four grid world examples shown in Figure 7. The results in Table 1 show that centralized shielding can guarantee collision free learning in all cases. Moreover, in three out of four maps, CQ-learning with centralized shield obtained better policies with higher rewards and smaller number of steps to reach the target, compared to no shielding. Figure 10 shows that centralized shielding achieves the highest accumulated reward in most times; moreover, the blue shaded area (standard deviation of no shielding) tends to stretch lower than others, indicating that CQ-learning without shielding obtains lower

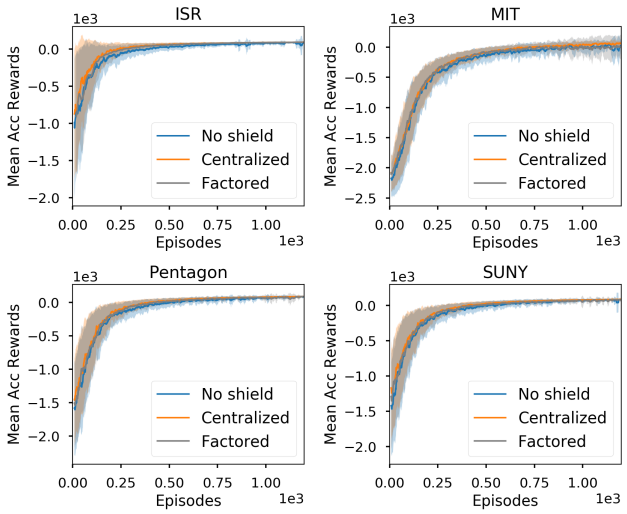

Figure 10: Comparison of CQ-learning without shielding, with centralized or factored shielding based on the accumulated rewards per episode (average and standard deviation over 1,000 training episodes, across 10 independent runs).

		IQL			CQ			CQ with centralized shield			CQ with factored shield		
Maps	Optimal Steps	Steps	Reward	Collisions	Steps	Reward	Collisions	Steps	Reward	Collisions	Steps	Reward	Collisions
ISR	5	30.35	-10.20	20.30	8.66	89.53	0.40	7.03	93.85	**0.00**	7.31	93.74	**0.00**
Pentagon	10	46.58	-19.17	11.60	10.96	88.96	0.20	12.08	88.44	**0.00**	13.20	84.88	**0.00**
MIT	18	20.84	77.33	0.00	42.93	30.38	0.90	28.38	73.94	**0.00**	29.96	37.96	**0.00**
SUNY	10	34.80	-160.175	72.60	13.97	84.78	0.30	11.97	88.44	**0.00**	14.02	83.77	**0.00**

Table 1: Results comparing the independent Q-learning, CQ-learning, CQ-learning with centralized and factored shields (average of 10 evaluation episodes conducted after 1,000 training episodes, across 10 independent runs).

rewards than with centralized shielding on average. The learning curves also show that the centralized shielding does not prevent the learner from converging across different examples. However, we failed to synthesize centralized shields with more than two agents in these grid maps, due to scalability issues of shield synthesis.

Factored Shielding Evaluation. First, we applied CQ-learning with factored shielding to the four grid world examples. We adopted a factorization scheme such that each shield monitors agent actions occurring within a 3×3 grid block in each map. Results in Table 1 show that CQ-learning with factored shielding can guarantee zero collisions in all examples, while learned policies have similar quality as those obtained from CQ-learning with centralized shielding. Figure 10 shows that factored shielding achieves similar performance in terms of the accumulated rewards per episode, compared to centralized shielding and without shielding. Due to the scalability limitation of CQ-learning, we can only consider two agents in these examples.

Additionally, we integrated a different algorithm MADDPG [15] with factored shielding and applied it to the cooperative navigation examples shown in Figure 8 with a 5×5 shield size where one unit of distance corresponds to 0.1 in the environment. There are four agents in each example, which is not feasible for centralized shielding approach to handle. Table 2 shows that MADDPG with factored shielding can guarantee zero collisions over the training period of 20, 000 episodes for both examples. By contrast, MADDPG without shielding leads to about 207 and 14, 419 occurrences of collisions for the cross and antipodal examples, respectively. Figure 11 shows that in the cross example, MADDPG without and with factored shielding have comparable learning performance in terms of the accumulated rewards per episode; in the antipodal example, MADDPG without shielding achieves higher rewards than MADDPG with factored shielding, though this comes at a trade-off of more collisions. The learning curves in Figure 11 also show that the factored shielding do not have negative impact on the learner's ability to converge.

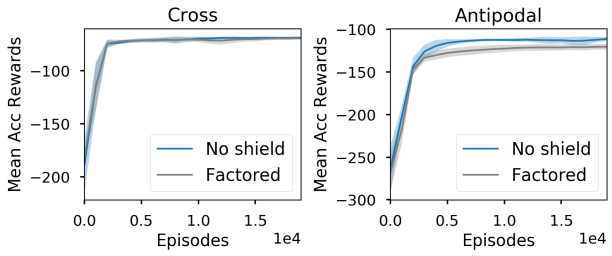

Figure 11: Comparison of MADDPG without and with factored shielding based on the accumulated rewards per episode (average and standard deviation over 20,000 training episodes, across 10 independent runs).

	MADDPG	MADDPG with Shield
Cross	207.20	**0.00**
Antipodal	14,419.20	**0.00**

Table 2: Total number of collisions over 20,000 training episodes for the cooperative navigation examples.

Summary. Our experiments demonstrate that the two shielding approaches can guarantee the safety, without compromising the learning performance in terms of the convergence rate and the quality of learned policies. Moreover, factored shielding is more scalable in the number of agents than centralized shielding.

7 CONCLUSION

In this paper, we present two shielding approaches that guarantee the safety specifications expressed in linear temporal logic (LTL) during the learning process of MARL. The centralized shielding approach synthesizes a single shield to centrally monitor the joint actions of all agents and only corrects any unsafe action that violates the LTL safety specification. However, the scalability of centralized shielding is restricted because the computational cost of shield synthesis grows exponentially with the number of agents. The factored shielding approach addresses this limitation by synthesizing multiple factored shields with each shield monitoring a subset of agents at each time step. Our experimental results show that both shielding approaches can guarantee the safety specification (e.g., collision avoidance) during learning, and achieve similar learning performance (e.g., convergence speed, quality of learned policies) as non-shielded MARL. We manually devise factorization schemes for the factored shielding approach in our experiments based on the problem-specific knowledge. In the future, we will explore the automated learning of efficient factorization schemes.

8 ACKNOWLEDGEMENTS

This work was supported in part by the Office of Naval Research Science of AI Program (grant N00014-18-1-2829). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research.

REFERENCES

[1] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe Reinforcement Learning via Shielding. In *AAAI-18: 32nd AAAI Conference on Artificial Intelligence*. 2669–2678.

[2] Rajeev Alur. 2015. *Principles of cyber-physical systems*. MIT Press.

[3] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*. MIT press.

[4] Suda Bharadwaj, Roderik Bloem, Rayna Dimitrova, Bettina Konighofer, and Ufuk Topcu. 2019. Synthesis of Minimum-Cost Shields for Multi-agent Systems. In *2019 American Control Conference (ACC)*. IEEE, 1048–1055.

[5] Roderick Bloem, Bettina Könighofer, Robert Könighofer, and Chao Wang. 2015. Shield synthesis. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 533–548.

[6] Alper Kamil Bozkurt, Yu Wang, Michael M Zavlanos, and Miroslav Pajic. 2020. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 10349–10355.

[7] Yann-Michaël De Hauwere, Peter Vrancx, and Ann Nowé. 2010. Learning multi-agent state space representations. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. 715–722.

[8] Rüdiger Ehlers and Vasumathi Raman. 2016. Slugs: Extensible gr (1) synthesis. In *International Conference on Computer Aided Verification*. Springer, 333–339.

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

[10] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. 2019. Omega-regular objectives in model-free reinforcement learning. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 395–412.

[11] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. 2020. Cautious Reinforcement Learning with Logical Constraints. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 483–491.

[12] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. 2019. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 33, 6 (2019), 750–797.

[13] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. 2009. Temporal-logic-based reactive mission and motion planning. *IEEE transactions on robotics* 25, 6 (2009), 1370–1381.

[14] Orna Kupferman and Moshe Y Vardi. 2001. Model checking of safety properties. *Formal Methods in System Design* 19, 3 (2001), 291–314.

[15] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*. 6379–6390.

[16] Francisco S Melo and Manuela Veloso. 2009. Learning of coordination: Exploiting sparse interactions in multiagent systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. 773–780.

[17] Amir Pnueli. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, 46–57.

[18] Dhananjay Raju, Suda Bharadwaj, and Ufuk Topcu. 2019. Decentralized runtime synthesis of shields for multi-agent systems. *arXiv preprint arXiv:1910.10380* (2019).

[19] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295* (2016).

[20] Arambam James Singh, Akshat Kumar, and Hoong Chuin Lau. 2020. Hierarchical Multiagent Reinforcement Learning for Maritime Traffic Management. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 1278–1286.

[21] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*. 330–337.

[22] Alphan Ulusoy, Stephen L Smith, Xu Chu Ding, Calin Belta, and Daniela Rus. 2013. Optimality and robustness in multi-robot path planning with temporal logic constraints. *The International Journal of Robotics Research* 32, 8 (2013), 889–911.

[23] Jiachen Yang, Alireza Nakhaei, David Isele, Kikuo Fujimura, and Hongyuan Zha. 2019. CM3: Cooperative Multi-goal Multi-stage Multi-agent Reinforcement Learning. In *International Conference on Learning Representations*.

[24] Chao Yu, Xin Wang, and Zhanbo Feng. 2019. Coordinated Multiagent Reinforcement Learning for Teams of Mobile Sensing Robots. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 2297–2299.

[25] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2019. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635* (2019).