Contract-Based Compositional Shielding for Safe Multi-Agent Reinforcement Learning

Omar Adalat[1], Edwin Hamel-De le Court[1,2], and Francesco Belardinelli[1]

[1] Imperial College London, London SW7 2AZ, UK
[2] University of Manchester, Manchester M13 9PL, UK
{o.adalat24,e.hamel.de-le-court,francesco.belardinelli}@imperial.ac.uk

Abstract. *Safe coordination* problems surface in multi-agent reinforcement learning when global safety cannot be enforced by any agent unilaterally: the admissibility of one agent's action may depend on the dynamics of other agents. Decentralised shields can enforce safety at runtime, but purely factorised permissions often exclude optimal team behaviour that is safe only through coordination. We study deterministic safety guarantees for agents trained and deployed under decentralised execution, recovering team-optimal safe behaviour without centralised runtime control. Agents have a shared global specification ϕ in the safety fragment of Linear Temporal Logic ($\mathsf{LTL}_{\mathsf{safe}}$), and select among tuples of local $\mathsf{LTL}_{\mathsf{safe}}$ obligations whose conjunction implies the global specification ϕ. Each agent may rely on the other agents' local obligations as assumptions because the whole contract tuple is certified simultaneously and allows projection into local action masks. At learning time, a non-stationary multi-armed bandit chooses among a library of local $\mathsf{LTL}_{\mathsf{safe}}$ obligations to select the tuple that optimises team reward, all without forgoing end-to-end safety. We evaluate the approach across 6 environments and 15 algorithmic variants.

Keywords: Safe Multi-Agent Reinforcement Learning · Decentralised Learning · Compositional Verification · Shielding

1 Introduction

Assuring the safety of learning cooperative agents requires reconciling two demands: agents should optimise a shared task objective, whilst satisfying safety constraints during training and deployment. In *safe coordination* problems [11, 23], the admissibility of one agent's action depends on the non-stationary policies of the other agents in the shared environment, so reasoning that treats teammate choices as arbitrary must discard behaviour that is safe only under coordinated actions. *Multi-Agent Reinforcement Learning* (MARL) offers a powerful framework for sequential decision-making under uncertainty in a shared environment, by leveraging sampling-based methods to iteratively refine policies in stochastic

Paper accepted to the 23rd European Conference on Multi-Agent Systems.

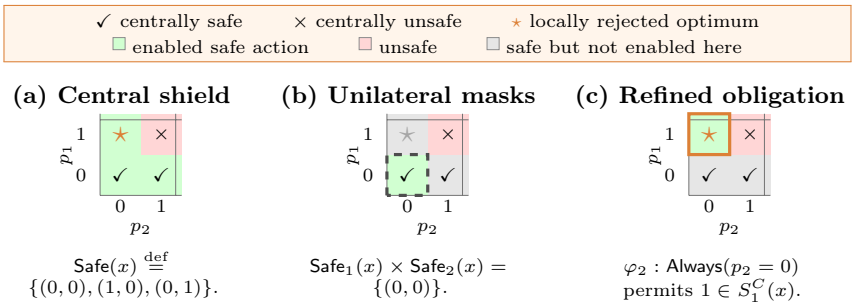

Fig. 1. A 2×2 safe-coordination instance in which unilateral local masks lose the safe optimal action, while a refined obligation recovers it through teammate commitment.

games [18]. Stochastic games are commonly studied under cooperative, competitive, and mixed strategic dynamics. Many multi-agent systems naturally involve cooperative tasks, such as rescue drones [9] and autonomous warehouses [28], which makes the cooperative setting a natural target for safe MARL. However, standard reward penalties are insufficient to verify that behaviour is safe once a policy is deployed [14]. From the formal methods standpoint, *shielding* [16] is a popular technique for enforcing safety during both training and deployment by pre-emptively masking out unsafe actions that may lead to a specification violation or post-posedly replacing unsafe actions [3].

Example 1. Figure 1 gives a two-agent safe-coordination instance. At state x, each agent chooses an action $p_i \in \{0,1\}$, and safety excludes only the simultaneous choice of 1: $\mathsf{Safe}(x) \stackrel{\text{def}}{=} \{(0,0),(1,0),(0,1)\}$. Suppose the team reward favours $(1,0)$. A central shield may admit the whole safe relation and therefore preserve this coordinated optimum. A purely unilateral decomposition, which permits an action only when it is safe against every teammate action, keeps only $\{0\} \times \{0\}$: $p_1 = 1$ is safe only when $p_2 = 0$, and $p_2 = 1$ is safe only when $p_1 = 0$. Thus unilateral (factorised) masking loses the optimal safe action. A certified local obligation, for example $\mathsf{Always}(p_2 = 0)$, records the missing commitment: under that contract, agent 1 can again be permitted to choose $p_1 = 1$, recovering $(1,0)$ without admitting the unsafe pair $(1,1)$.

This paper studies compositional decentralised shielding which allows the preservation of optimal safe coordination actions for cooperative MARL. We construct local shields from certified local temporal contracts, while preserving a single shared global safety objective. We ask whether decentralised shields can recover optimal team-return safe coordination without falling back to a conservative Cartesian decomposition of the central shield. The technical challenge is to let agents rely on one another's obligations without introducing an unsound proof cycle. We answer it by certifying whole contract tuples through a circular fixed-point construction and projecting the jointly certified action relation into decentralised runtime masks. Learning then selects among finite tuples of local temporal obligations whose conjunction entails the single globally

shared temporal objective, exposing high-return coordinated behaviours whilst upholding deterministic safety end-to-end.

Contributions. Our contributions are as follows:
(i) We show how a shared global safety specification in $\mathsf{LTL}_{\mathsf{safe}}$ can be decomposed through candidate local obligations and compiled into deterministic safety automata for shield synthesis.
(ii) We provide formal soundness guarantees for decentralised shielding based on circular assume-guarantee reasoning, local $\mathsf{LTL}_{\mathsf{safe}}$ contracts, and contract-product fixed points. Moreover, we also prove recoverability of the optimal safe deterministic policy under explicit library-representability conditions.
(iii) We show how bounded certified profile search and discounted upper-confidence contract selection can reduce conservatism while preserving certification.
(iv) Furthermore, we evaluate reward and safety performance across 6 cooperative multi-agent environments using an extensive set of 15 algorithmic variants.

2 Related Work

Multi-Agent Shielding. First covered in [11], multi-agent shielding has since been studied in both centralised and factorised (decentralised) forms. There is also the notion of *dynamic shielding* [29], which allows shields to split, merge, and recompute based on the current agent configuration, for example when interaction is local in space. In particular, Alshiekh et al. [3] compile safety specifications into automata and synthesise shields that are correct by construction and *minimally interfering*, in the sense that they forbid an action only when allowing it could lead to a violation. We adapt the same automata-theoretic safety-game perspective as [3, 17] for decentralised contracts.

Compositional Shielding. Melcer et al. [19] address shield decentralisation by compiling a centralised shield into per-agent local shields, which is a *shield decomposition* method or Cartesian factorisation. Their construction starts from a known-safe joint action for each shield-state/observation pair and incrementally enlarges each agent's locally permitted action set only when every induced joint action remains safe and the successor shield state remains locally identifiable. This preserves safety guarantees in a communication-free setting. The price of this decentralisation is that the safe joint-action relation must admit a Cartesian-style decomposition into independent local permissions. Jointly safe but non-factorisable coordinated actions are excluded, making the decentralised shield strictly more conservative than the original centralised shield in general.

Brorholt et al. [6] show that naively projecting a centralised shield can lose coordination information, and recover sound decentralised shields through *ordered assume-guarantee* reasoning. Their method requires an acyclic dependency structure: an agent may rely only on guarantees earlier in the order, and the associated cascading-learning procedure exploits this order to obtain in-distribution training and Pareto-style guarantees under local-optimality assumptions. This is the closest

precedent to our setting. Our contribution differs in both the proof principle and the learning style. Rather than fixing an acyclic dependency order, we certify an entire tuple of local obligations simultaneously by a contract-product fixed point. This permits circular coordination assumptions and therefore does not require cascading training or an acyclic training order. Moreover, algorithmic synthesis of local assume-guarantee properties is left unaddressed; we keep a single shared global $\mathsf{LTL}_{\mathsf{safe}}$ objective and search over finite tuples of local $\mathsf{LTL}_{\mathsf{safe}}$ obligations whose conjunction entails it. Learning then selects among pre-certified contracts that can expose high-return safe behaviours while preserving deterministic safety.

3 Preliminaries

Multi-Agent Reinforcement Learning. We model multi-agent RL problems as an n-agent *concurrent stochastic game* (a.k.a. a *Markov game* [18]) $\mathcal{M} \stackrel{\text{def}}{=} (\mathcal{S}, s_0, \mathcal{A}^1, \ldots, \mathcal{A}^n, d^1, \ldots, d^n, \mathcal{P}, \mathcal{R}^1, \ldots, \mathcal{R}^n, \gamma)$, where $\mathcal{S}$ is the finite *state space* and $s_0 \in \mathcal{S}$ is the *initial state*; $\mathsf{Ag} \stackrel{\text{def}}{=} \{1, \ldots, n\}$ is the set of *agents*; each $\mathcal{A}^i$ is the finite *action space* of agent $i \in \mathsf{Ag}$ and $\mathcal{A} \stackrel{\text{def}}{=} \prod_{i \in \mathsf{Ag}} \mathcal{A}^i$ is the *joint action space*; each $d^i : \mathcal{S} \to 2^{\mathcal{A}^i} \setminus \{\emptyset\}$ is the *action-availability protocol* of agent i, with *legal joint-action* set $d(s) \stackrel{\text{def}}{=} \prod_{i \in \mathsf{Ag}} d^i(s)$ at state s, so a is *legal* at s iff $a \in d(s)$. If the environment has no state-dependent feasibility restrictions, then $d^i(s) = \mathcal{A}^i$ for all i and s; $\mathsf{Dom}(d) \stackrel{\text{def}}{=} \{(s, a) \in \mathcal{S} \times \mathcal{A} \mid a \in d(s)\}$ is the *legal state-action domain*; $\mathcal{P} : \mathsf{Dom}(d) \to \Delta(\mathcal{S})$ is the *transition function*, with $\Delta(\mathcal{S})$ being the set of *probability distributions* over $\mathcal{S}$; each $\mathcal{R}^i : \mathsf{Dom}(d) \times \mathcal{S} \to \mathbb{R}$ is the (Markovian) *reward function* of agent i; and γ is the *discount factor*. The action protocol captures state-dependent primitive-action feasibility before shielding.

A (stochastic) *policy* π^i (also known as a *strategy*) of agent i is a function $\pi^i : \mathcal{S} \to \Delta(\mathcal{A}^i)$ such that $\pi^i(a^i \mid s) = 0$ whenever $a^i \notin d^i(s)$. A *joint policy* $\pi = (\pi^1, \ldots, \pi^n)$ induces the decentralised product distribution $\pi(a \mid s) \stackrel{\text{def}}{=} \prod_{i=1}^{n} \pi^i(a^i \mid s)$, where $a = (a^1, \ldots, a^n)$. The joint policy of other agents is $\pi^{-i}(a^{-i}|s) \stackrel{\text{def}}{=} \prod_{j \neq i} \pi^j(a^j|s)$, where a^{-i} is the joint action except agent i's action. In the *cooperative* strategic setting, the agents are evaluated by a common team objective. We use the standard team-return criterion:

$$J_{s_0}(\pi) \stackrel{\text{def}}{=} \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \sum_{i=1}^{n} \mathcal{R}^i(s_t, a_t, s_{t+1})\right],$$

where $\pi = (\pi^1, \ldots, \pi^n)$ is the joint policy. A joint policy $\pi^\star$ is *team-optimal* iff $\pi^\star \in \arg\max_\pi J_{s_0}(\pi)$, that is, no other joint policy achieves higher expected discounted total reward from the initial state.

Throughout, for a distribution $\mu \in \Delta(Z)$, write $\mathrm{Supp}(\mu) \stackrel{\text{def}}{=} \{z \in Z \mid \mu(z) > 0\}$ for its support, i.e. the set of values that can occur with non-zero probability. An *execution* from s is an alternating sequence $s_0 a_0 s_1 a_1 \cdots$ with $s_0 = s$, $a_t \in d(s_t)$, and $s_{t+1} \in \mathrm{Supp}(\mathcal{P}(s_t, a_t))$ at every step; finite execution prefixes are defined

similarly. The execution is consistent with π if $\pi(a_t \mid s_t) > 0$ at every step. We write $\mathsf{IPath}^{\pi}(s)$ for the set of state paths $s_0 s_1 \cdots$ induced by executions from s that are consistent with π. The same joint policy induces a finite *Markov chain* [4] $\mathcal{M}^{\pi} \stackrel{\text{def}}{=} (\mathcal{S}, s_0, \mathcal{P}^{\pi})$, where $\mathcal{P}^{\pi}(s, s') \stackrel{\text{def}}{=} \sum_{a \in d(s)} \pi(a \mid s)\, \mathcal{P}(s, a)(s')$.

Safe LTL. We express the desired global behaviour as a trace property in the safety fragment of *Linear Temporal Logic*, denoted $\mathsf{LTL}_{\mathsf{safe}}$. We consider a *labelled stochastic game* obtained by extending a Markov game $\mathcal{M}$ with a finite global alphabet Ap and a global labelling function $\mathcal{L} : \mathcal{S} \to 2^{\mathsf{Ap}}$. The shared safety objective is interpreted over this global labelling. For decentralised contracts we also fix local alphabets $\mathsf{Ap}_i \subseteq \mathsf{Ap}$ which specify the global propositions that agent i's local obligations may mention. Along a path, local observations are obtained by projecting the global labels to Ap_i. The syntax of the safety fragment is generated by the following grammar:

$$\phi ::= \top \mid \bot \mid p \mid \neg p \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{X}\phi \mid \phi \mathsf{W} \phi,$$

where $p \in \mathsf{Ap}$, and $\top, \bot$ denote the Boolean constants. We read X as *next* and W as *weak until*: $\phi \mathsf{W} \psi$ requires ϕ until ψ, or ϕ forever if ψ never occurs. Thus the fragment captures invariance-style properties: negation is restricted to atoms and temporal progression is expressed through next and weak until. This presentation is equivalent to the standard negation-normal Safety LTL fragment using X, R, and W [24], since $\phi \mathsf{R} \psi$ can be written as $\psi \mathsf{W}(\phi \wedge \psi)$. We use the always operator $\mathsf{G}\phi \stackrel{\text{def}}{=} \phi \mathsf{W} \bot$. The connectives $\rightarrow$ and $\leftrightarrow$ are used only as standard abbreviations with their expansions normalised to the safety grammar, since negation is only allowed on atomic propositions.

Let $\mathcal{T} \stackrel{\text{def}}{=} (2^{\mathsf{Ap}})^{\omega}$ be the set of traces over Ap. Each path $\rho = s_0 s_1 s_2 \cdots \in \mathsf{IPath}^{\pi}(s)$ induces the global trace $\mathcal{T}_{\mathcal{L}}(\rho) \stackrel{\text{def}}{=} \mathcal{L}(s_0)\, \mathcal{L}(s_1)\, \mathcal{L}(s_2) \cdots \in \mathcal{T}$ and, for each agent i, the local trace $\mathcal{T}_{\mathcal{L}}(\rho) \restriction_{\mathsf{Ap}_i} \stackrel{\text{def}}{=} (\mathcal{L}(s_0) \cap \mathsf{Ap}_i)\, (\mathcal{L}(s_1) \cap \mathsf{Ap}_i)\, (\mathcal{L}(s_2) \cap \mathsf{Ap}_i) \cdots \in (2^{\mathsf{Ap}_i})^{\omega}$. We write $(\tau, k) \models \phi$ when formula ϕ holds at position k of trace τ, and $\tau \models \phi$ for satisfaction at position 0. The inductive trace semantics are provided in Appendix A as standard. In the main text we use the induced-chain satisfaction relation $\mathcal{M}^{\pi} \models \phi$ iff $\mathcal{T}_{\mathcal{L}}(\rho) \models \phi$ for every $\rho \in \mathsf{IPath}^{\pi}(s_0)$. Thus satisfaction is universal over paths: a policy satisfies the safety formula only when all possible executions that it can generate satisfy the formula.

Problem Statement. In the labelled cooperative stochastic game defined above, fix a shared safety specification $\Phi_{\mathsf{safe}} \in \mathsf{LTL}_{\mathsf{safe}}(\mathsf{Ap})$. For a joint policy $\pi = (\pi^1, \ldots, \pi^n)$, the safe policy set is $\Pi_{\mathsf{safe}} \stackrel{\text{def}}{=} \{\pi \mid \mathcal{M}^{\pi} \models \Phi_{\mathsf{safe}}\}$. The formal target is to find a team-return-optimal safe joint policy, that is:

$$\pi^{\star} \in \arg \max_{\pi \in \Pi_{\mathsf{safe}}} J_{s_0}(\pi).$$

Thus the objective is not merely to rule out unsafe executions: among policies whose induced traces all satisfy the temporal safety property, the learner should

preserve as much discounted team return as possible. As in Example 1, the optimum may require coordinated joint actions that a Cartesian decomposition of the central safe-action relation would discard. We target decentralised execution: each shield restricts one agent to locally admissible actions, while certification of the active contract ensures every joint execution admitted by the distributed masks satisfies Φ_{safe}.

4 Contract Shielding

Contract shield synthesis constructs decentralised action masks whose Cartesian product is safe under a globally certified contract. Rather than deploying the central safety-game action relation directly, the construction compiles candidate local obligations into automata and certifies local action rectangles through a contract-product fixed point. We use a finite *safety abstraction*, obtainable by bisimulation [10], that is exact for the contract shielding problem. Each contract supplies one temporal obligation per agent. Using the finite bad-prefix characterisation of safety properties [17], each local $\mathsf{LTL}_{\mathsf{safe}}$ obligation induces a finite deterministic safety monitor: its bad states recognise exactly the finite prefixes from which the obligation is irrecoverably violated. The shield is synthesised from their product with the environment and certifying non-reachability of unsafe states; the resulting fixed point certifies decentralised action interfaces as opposed to a central joint-action controller.

Figure 2 summarises our approach. The labelled game and global safety specification are compiled into contract-specific safety products with local obligations providing the assume-guarantee interface of Section 4.1; resulting contract products are solved and projected to decentralised masks in Section 4.2. The certified contracts then form the library used for learning-time selection in Section 5.

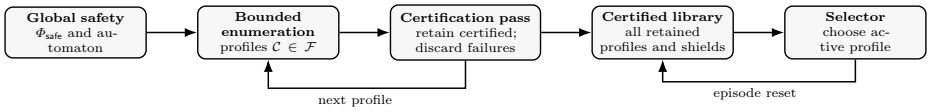

Fig. 2. High-level pipeline of contract-based compositional shielding.

4.1 Assume-Guarantee Contracts

Assume-guarantee reasoning is a technique from the toolkit of *compositional verification*. In the multi-agent setting we use it to control how much coordination information each local shield may rely on. We write $\langle A \rangle C \langle G \rangle$ to mean that component C guarantees G under assumption A. A local contract can express more than "avoid a violation": it can also encode coordination promises that make globally safe and high-return behaviour available under decentralised execution.

We begin by defining the local action rectangles used as shield interfaces:

Definition 1 (Local Action Rectangle). *At an environment state $s \in \mathcal{S}$, a local action rectangle (or simply rectangle) is a non-empty Cartesian product*

$$R \stackrel{\text{def}}{=} R^1 \times \cdots \times R^n \qquad \textit{with} \qquad \emptyset \neq R^i \subseteq d^i(s) \quad \textit{for every } i \in \mathsf{Ag}.$$

Thus $R \subseteq d(s)$, and each factor R^i is the local action mask exposed to agent i.

We write traces over the global alphabet as words in $(2^{\mathsf{Ap}})^\omega$, finite prefixes as words in $(2^{\mathsf{Ap}})^*$, and agent-i observations as their pointwise projections to $(2^{\mathsf{Ap}_i})^\omega$. Next, we introduce local obligations and contracts.

Definition 2 (Local Obligation). *A local $\mathsf{LTL}_{\mathsf{safe}}$ obligation for agent i is a formula $\varphi_i \in \mathsf{LTL}_{\mathsf{safe}}(\mathsf{Ap}_i)$ over that agent's local alphabet.*

Definition 3 (Contract). *A $\mathsf{LTL}_{\mathsf{safe}}$ contract is a tuple of local obligations, one for each agent,*

$$\mathcal{C} \stackrel{\text{def}}{=} \langle \varphi_i \rangle_{i=1}^n \in \prod_{i=1}^n \mathsf{LTL}_{\mathsf{safe}}(\mathsf{Ap}_i).$$

A contract can certify safety only when the conjunction of its local obligations entails the shared global safety objective. Together, Definitions 2 and 3 fix the local syntactic objects used below. Their semantics is lifted to global traces by projection: for each component, define $[\![\varphi_i]\!] \stackrel{\text{def}}{=} \{\tau \in \mathcal{T} \mid \tau \restriction_{\mathsf{Ap}_i} \models \varphi_i\}$, where $\tau \restriction_{\mathsf{Ap}_i}$ is the pointwise projection of τ to Ap_i. For a contract $\mathcal{C} = \langle \varphi_i \rangle_{i=1}^n$, write $[\![\mathcal{C}]\!] \stackrel{\text{def}}{=} \bigcap_{i=1}^n [\![\varphi_i]\!]$. Equivalently, for every global trace $\tau \in \mathcal{T}$, $\tau \in [\![\mathcal{C}]\!] \iff \forall i \in \{1, \ldots, n\},\ \tau \restriction_{\mathsf{Ap}_i} \models \varphi_i$. This lets local obligations be interpreted on global traces by ignoring propositions outside the corresponding agent alphabet. For a global formula $\Psi \in \mathsf{LTL}_{\mathsf{safe}}(\mathsf{Ap})$, write $\mathcal{C} \models_\uparrow \Psi$ when $\forall \tau \in \mathcal{T}.\ \big(\forall i \in \mathsf{Ag},\ \tau \restriction_{\mathsf{Ap}_i} \models \varphi_i\big) \implies \tau \models \Psi$. Semantic entailment involving local obligations is always understood through this lifted interpretation. Thus a contract is a decomposition of the global requirement into local promises.

Definition 4 (Finite Contract Search Space). *Fix a formula-depth bound D and, for each agent i, a finite local formula language $\mathsf{Form}_i^{\leq D} \subseteq \mathsf{LTL}_{\mathsf{safe}}(\mathsf{Ap}_i)$. The finite contract search space is $\mathcal{C}_D \stackrel{\text{def}}{=} \prod_{i=1}^n \mathsf{Form}_i^{\leq D}$.*

Definition 5 (Circular Contract Assumptions). *For a candidate contract $\mathcal{C} = \langle \varphi_i \rangle_{i=1}^n$, the local assume-guarantee obligation of agent i is $\langle A_i^{\mathcal{C}} \rangle i \langle \varphi_i \rangle$, where $A_i^{\mathcal{C}} \stackrel{\text{def}}{=} \bigwedge_{j \neq i} \varphi_j$. The conjunction in $A_i^{\mathcal{C}}$ is semantic notation using the lifted interpretation above: it denotes the intersection $\bigcap_{j \neq i} [\![\varphi_j]\!]$ over global traces, and may therefore combine obligations whose syntax ranges over different local alphabets. When the index set is empty, the conjunction is $\top$.*

Thus agent i may rely on the other agents satisfying their local obligations, while it must satisfy its own obligation. This is circular assume-guarantee reasoning: the obligations in $\mathcal{C}$ justify one another as a simultaneous contract rather than through a fixed linear order. Certification below checks the whole tuple at once:

each agent may rely on the other obligations only because the fixed point proves that the projected shields preserve all obligations simultaneously, which enables sound circularity. The automata used for this certification are defined next.

Definition 6 (Contract Safety Automata). *For a contract $\mathcal{C} = \langle \varphi_i \rangle_{i=1}^n$, each local obligation φ_i is compiled into a contract safety automaton $\mathcal{A}_i^{\mathcal{C}} \stackrel{\text{def}}{=} (Q_i, q_i^0, \Sigma_i, \delta_i, B_i)$, where Q_i is a finite set of states, $q_i^0 \in Q_i$ is the initial state, $\Sigma_i \stackrel{\text{def}}{=} 2^{\mathsf{Ap}_i}$, $\delta_i : Q_i \times \Sigma_i \to Q_i$ is the total transition function, and $B_i \subseteq Q_i$ is an absorbing set of bad states. Writing δ_i^* for the finite-word extension of δ_i, the finite words that reach B_i are exactly the bad prefixes of φ_i:*

$$\delta_i^*(q_i^0, w) \in B_i \quad \Longleftrightarrow \quad \forall \tau_i \in \Sigma_i^\omega .\, w\tau_i \not\models \varphi_i.$$

Thus a local trace satisfies φ_i precisely when no finite prefix reaches B_i. For compactness, write $\ell_i(s) \stackrel{\text{def}}{=} \mathcal{L}(s) \cap \mathsf{Ap}_i$ for the projection of the global label at s to agent i's alphabet.

Definition 7 (Contract Product). *For a candidate contract $\mathcal{C} = \langle \varphi_i \rangle_{i=1}^n$ with contract safety automata $\mathcal{A}_i^{\mathcal{C}}$ from Definition 6, the contract product has state space $X^{\mathcal{C}} \stackrel{\text{def}}{=} \mathcal{S} \times Q_1 \times \cdots \times Q_n$. A product state is written $x^{\mathcal{C}} = (s, q_1, \ldots, q_n)$, where q_i is the current automaton state for agent i's obligation. The initial contract product state is $x_0^{\mathcal{C}} \stackrel{\text{def}}{=} (s_0, \delta_1(q_1^0, \ell_1(s_0)), \ldots, \delta_n(q_n^0, \ell_n(s_0)))$. A contract product state is locally safe if $q_i \notin B_i$ for every agent. For a legal joint action $a = (a^1, \ldots, a^n) \in d(s)$, the qualitative successor set is*

$$\mathrm{Post}_{\mathcal{C}}(x^{\mathcal{C}}, a) \stackrel{\text{def}}{=} \{(s', \delta_1(q_1, \ell_1(s')), \ldots, \delta_n(q_n, \ell_n(s'))) \mid s' \in \mathrm{Supp}(\mathcal{P}(s, a))\}.$$

A contract-product execution from $x_0^{\mathcal{C}}$ is an alternating sequence $\langle x_0^{\mathcal{C}} a_0 x_1^{\mathcal{C}} a_1 \cdots \rangle$ such that, writing $x_t^{\mathcal{C}} = (s_t, \bar{q}_t)$, for every $t \geq 0$,

$$a_t \in d(s_t) \qquad \text{and} \qquad x_{t+1}^{\mathcal{C}} \in \mathrm{Post}_{\mathcal{C}}(x_t^{\mathcal{C}}, a_t).$$

The contract product in Definition 7 is the natural Markov state space for shielded controllers. In decentralised learning, agent i uses the active contract identifier together with an augmentation sufficient to recover its projected mask.

4.2 Contract Product Shields

We now turn a candidate contract into actual local shields. For a fixed $\mathcal{C}$, the shield construction computes one shared assume-guarantee fixed point over $X^{\mathcal{C}}$ and then projects the resulting action relation to each agent.

Definition 8 (Contract Predecessor). *At a contract product state $x^{\mathcal{C}} = (s, q_1, \ldots, q_n)$, let $R(x^{\mathcal{C}})$ range over local action rectangles at s in the sense of Definition 1. For $W \subseteq X^{\mathcal{C}}$, such a rectangle is W-closed at $x^{\mathcal{C}}$ if*

$$\mathrm{Post}_{\mathcal{C}}(x^{\mathcal{C}}, a) \subseteq W \text{ for every } a \in R(x^{\mathcal{C}}).$$

Let $\mathsf{Rect}_{\mathcal{C}}(W, x^{\mathcal{C}})$ *be the set of* W*-closed local action rectangles at* $x^{\mathcal{C}}$*. The assume-guarantee predecessor operator is*

$$\mathcal{T}_{\mathcal{C}}(W) \stackrel{\text{def}}{=} \left\{ x^{\mathcal{C}} \in X^{\mathcal{C}} \;\middle|\; \begin{array}{l} x^{\mathcal{C}} \text{ is locally safe and} \\ \mathsf{Rect}_{\mathcal{C}}(W, x^{\mathcal{C}}) \neq \emptyset \end{array} \right\}.$$

Definition 9 (Rectangular Winning Region). *The rectangular winning region of the contract product is the greatest fixed point* $\mathsf{Win}_{\mathcal{C}}^{\square} \stackrel{\text{def}}{=} \nu W.\, \mathcal{T}_{\mathcal{C}}(W)$*. It is the largest locally safe region from which the shield can offer a non-empty Cartesian action interface closed under all induced successors. For* $x^{\mathcal{C}} \in \mathsf{Win}_{\mathcal{C}}^{\square}$*, write* $\mathsf{Rect}_{\mathcal{C}}(x^{\mathcal{C}}) \stackrel{\text{def}}{=} \mathsf{Rect}_{\mathcal{C}}(\mathsf{Win}_{\mathcal{C}}^{\square}, x^{\mathcal{C}})$ *for the winning rectangles available at* $x^{\mathcal{C}}$*.*

The operator $\mathcal{T}_{\mathcal{C}}$ is monotone. Since $X^{\mathcal{C}}$ is finite, the descending iteration $W_0 = X^{\mathcal{C}}$ and $W_{k+1} = \mathcal{T}_{\mathcal{C}}(W_k)$ stabilises at $\mathsf{Win}_{\mathcal{C}}^{\square}$, so $\mathsf{Win}_{\mathcal{C}}^{\square} = \mathcal{T}_{\mathcal{C}}(\mathsf{Win}_{\mathcal{C}}^{\square})$. This fixed point is the certificate used below.

Definition 10 (Certified Contract). *A candidate contract* $\mathcal{C} = \langle \varphi_i \rangle_{i=1}^{n} \in \mathcal{C}_D$ *is* certified *if:*

1. *the local obligations entail the global safety objective under lifted semantics,* $\mathcal{C} \models_{\uparrow} \Phi_{\mathsf{safe}}$*;*
2. *the initial contract product state* $x_0^{\mathcal{C}}$ *lies in* $\mathsf{Win}_{\mathcal{C}}^{\square}$*.*

The entailment condition makes the local promises sufficient for global safety. The winning-region condition makes them operational: from the initial state, a non-empty decentralised interface can keep the active contract outside all bad states. Thus certification fails only when the promises are too weak or not enforceable by a rectangular shield from the start. This is the initial-state condition used by Algorithm 1. For every $x^{\mathcal{C}} \in \mathsf{Win}_{\mathcal{C}}^{\square}$, choose any certified rectangle $R_{\mathcal{C}}(x^{\mathcal{C}}) \in \mathsf{Rect}_{\mathcal{C}}(x^{\mathcal{C}})$. This membership is the closure certificate: every joint action in $R_{\mathcal{C}}(x^{\mathcal{C}})$ is legal and has all contract-product successors in $\mathsf{Win}_{\mathcal{C}}^{\square}$. Where multiple rectangles satisfy this property, we use a deterministic tie-breaker which orders choices lexicographically maximising $(\sum_i |R^i|, \prod_i |R^i|, (R^1, \ldots, R^n))$, with each R^i ordered by the local action order. The safety results hold in generality, for any choice satisfying the closure certificate. The tie-breaker allows fixing a reproducible shield.

Definition 11 (Projected Local Shield). *For a certified contract* $\mathcal{C}$ *and product state* $x^{\mathcal{C}} \in \mathsf{Win}_{\mathcal{C}}^{\square}$*, the deployed local shield for agent* i *is the projection* $\mathcal{M}_{\shield}^{\mathcal{C},i}(x^{\mathcal{C}}) \stackrel{\text{def}}{=} R_{\mathcal{C}}^{i}(x^{\mathcal{C}})$*. The shield is defined only on* $\mathsf{Win}_{\mathcal{C}}^{\square}$*.*

Definition 12 (Local Mask Identifiability). *For a certified contract* $\mathcal{C}$*, the projected shield is locally mask-identifiable for agent* i *if some function of the active contract identifier and agent-*i*'s observation history returns* $\mathcal{M}_{\shield}^{\mathcal{C},i}(x_t^{\mathcal{C}})$ *for every reachable contract-product execution and time* t*.*

Definition 11 fixes the local mask exposed to agent i: the ith factor of a certified rectangle. Definition 12 is the corresponding information condition for decentralised execution; it requires that this factor be computable from the active contract identifier and the information available to agent i (for example, its augmented local contract-product state), without necessitating agent i to reconstruct the other agents' masks.

Definition 13 (Admitted Joint Actions). *For a certified contract $\mathcal{C}$ and product state $x^{\mathcal{C}} \in \mathsf{Win}^{\Box}_{\mathcal{C}}$, the joint-action relation admitted by the deployed local shields is* $\mathsf{Adm}_{\mathcal{C}}(x^{\mathcal{C}}) \stackrel{\text{def}}{=} \prod_{i \in \mathsf{Ag}} \mathcal{M}^{\mathcal{C},i}_{\shield}(x^{\mathcal{C}})$.

Theorem 1 (Circular Compositional Soundness). *Let $\mathcal{C} = \langle \varphi_i \rangle_{i=1}^{n}$ be a certified contract. For any contract-product execution $x_0^{\mathcal{C}} a_0 x_1^{\mathcal{C}} a_1 \cdots$ starting from $x_0^{\mathcal{C}}$, if $a_t \in \mathsf{Adm}_{\mathcal{C}}(x_t^{\mathcal{C}})$ at every time t, then every reachable contract product state remains in $\mathsf{Win}^{\Box}_{\mathcal{C}}$. Consequently, the local obligations hold along the execution, and the induced global trace satisfies Φ_{safe}.*

The proof is available in Appendix B.1. Informally, the theorem says that a certified circular contract behaves like a joint safety proof split across local masks. The invariant is simple: start inside the winning region, and every projected shield action keeps the next product state inside it. Thus, once the rectangle has been certified, the agents do not need any coordination — any joint action admitted by the local masks is still inside the certified successor region.

5 Learning with Contract Shields

Building on Section 4, this section gives the concrete pipeline: bounded offline construction of a certified library in Section 5.1, its permissiveness and optimality interpretation in Section 5.2, and online contract selection in Section 5.3.

5.1 Library Construction: Bounded Certified Profile Search

The finite contract search space $\mathcal{C}_D$ is finite, with candidates $\mathcal{C}$ drawn from the finite search space of Definition 4, so we enumerate candidate local-obligation tuples before learning and certify each tuple using the fixed-point construction above. For any finite ordered candidate family $\mathcal{F} \subseteq \mathcal{C}_D$, the certified library induced by $\mathcal{F}$ is $\mathcal{L}_{\mathsf{cert}}(\mathcal{F}) \stackrel{\text{def}}{=} \{\mathcal{C} \in \mathcal{F} \mid \mathcal{C} \text{ is certified}\}$, with the order inherited from $\mathcal{F}$. Since $\mathcal{F}$ is finite, so is $\mathcal{L}_{\mathsf{cert}}(\mathcal{F})$; moreover, membership in the library is exactly certification, so every retained contract satisfies the hypotheses of Theorem 1. The selector therefore uses this catalogue of safe shield configurations without constructing or validating new safety certificates online. The algorithm enumerates scoped candidate subsets in a deterministic prioritised order subject to the configured profile limit and formula depth limit; this order uses a temporal-form heuristic that prefers persistent $G\psi$ obligations before next-state $X\psi$ obligations.

Algorithm 1: Certified profile library construction

Input: ordered bounded profile family $\mathcal{F} \subseteq \mathcal{C}_D$, initial abstract states S_0, global safety formula Φ_{safe}
Output: initial certified profile $\mathcal{C}_0$ and certified library $\mathcal{L}_{\mathsf{cert}}$

$\mathcal{L}_{\mathsf{cert}} \leftarrow \emptyset$
foreach $\mathcal{C} \stackrel{\text{def}}{=} \langle \varphi_i \rangle_{i=1}^{n} \in \mathcal{F}$ *in search order* **do**
  if $\mathcal{C} \not\models_{\uparrow} \Phi_{\mathsf{safe}}$ **then**
    continue
  compile the local obligation automata $\mathcal{A}_1^{\mathcal{C}}, \ldots, \mathcal{A}_n^{\mathcal{C}}$
  construct the contract product $X^{\mathcal{C}}$ and greatest fixed point $\mathsf{Win}_{\mathcal{C}}^{\Box}$
  if *every initial contract state induced by* S_0 *lies in* $\mathsf{Win}_{\mathcal{C}}^{\Box}$ **then**
    derive the local shields from the certified rectangles $R_{\mathcal{C}}(x^{\mathcal{C}})$
    add $\mathcal{C}$ to $\mathcal{L}_{\mathsf{cert}}$
if $\mathcal{L}_{\mathsf{cert}} = \emptyset$ **then**
  return failure
$\mathcal{C}_0 \leftarrow$ the first profile in $\mathcal{L}_{\mathsf{cert}}$ in certification order
return $\mathcal{C}_0$ and $\mathcal{L}_{\mathsf{cert}}$

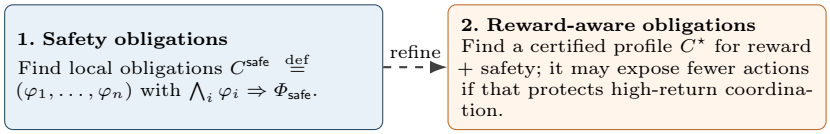

Fig. 3. Safety certification and reward-aware contract selection are distinct objectives.

Algorithm 1 summarises this finite preprocessing stage. Return-optimal behaviour does not necessarily arise from the first certified profile. A stricter contract can enable solving the decentralised learner's coordination problem, so the learning-time selector optimises over the retained library.

5.2 Permissiveness and Optimality

Definition 14 gives a statewise mask-size metric, obtained simply by counting the sizes of local action masks relative to the protocol-available action sets.

Definition 14 (Cardinality Permissiveness). *At a shield state* $x^{\mathcal{C}} = (s, \bar{q})$, *the cardinality permissiveness of agent* i*'s local mask is* $\rho_i(x^{\mathcal{C}}) \stackrel{\text{def}}{=} \frac{|\mathcal{M}_{\mathcal{C},i}(x^{\mathcal{C}})|}{|d^i(s)|}$.

Notice cardinality permissiveness is not the optimisation objective: $\rho_i(x^{\mathcal{C}}) = 1$ means no protocol-available primitive action of agent i is removed at $x^{\mathcal{C}}$, while smaller values indicate more intervention. A large mask can preserve many low-value actions while excluding coordinated high-return behaviours, and a stricter certified profile can make useful coordination easier to learn. Contract selection therefore optimises observed team return over certified profiles, not total mask

cardinality. The relevant completeness question is whether some certified contract preserves the globally optimal safe behaviour. We formalise this by comparing policies admitted by certified decentralised shields.

Definition 15 (Contract-Compatible Policies). *Let $\mathcal{C}$ be a certified contract. For a joint policy π, let $\mathsf{Reach}_{\mathcal{C}}(\pi)$ be the least subset of $\mathsf{Win}_{\mathcal{C}}^{\square}$ containing $x_0^{\mathcal{C}}$ and closed under admitted π-successors: if $x^{\mathcal{C}} = (s, \bar{q}) \in \mathsf{Reach}_{\mathcal{C}}(\pi)$, $a \in \mathrm{Supp}(\pi(\cdot \mid s)) \cap \mathsf{Adm}_{\mathcal{C}}(x^{\mathcal{C}})$, and $x' \in \mathrm{Post}_{\mathcal{C}}(x^{\mathcal{C}}, a)$, then $x' \in \mathsf{Reach}_{\mathcal{C}}(\pi)$. The policies compatible with $\mathcal{C}$ are*

$$\Pi(\mathcal{C}) \stackrel{\text{def}}{=} \left\{ \pi \;\middle|\; \forall x^{\mathcal{C}} = (s, \bar{q}) \in \mathsf{Reach}_{\mathcal{C}}(\pi),\ \mathrm{Supp}(\pi(\cdot \mid s)) \subseteq \mathsf{Adm}_{\mathcal{C}}(x^{\mathcal{C}}) \right\}.$$

For a deterministic Markov policy π, write $a_\pi(s)$ for its unique joint action at state s. Then $\pi \in \Pi(\mathcal{C})$ iff, for every $x^{\mathcal{C}} = (s, \bar{q}) \in \mathsf{Reach}_{\mathcal{C}}(\pi)$, we have $a_\pi(s) \in \mathsf{Adm}_{\mathcal{C}}(x^{\mathcal{C}})$. Equivalently, each local action $a_\pi^i(s)$ lies in the deployed local mask $\mathcal{M}_{\shield}^{\mathcal{C},i}(x^{\mathcal{C}})$ at every reachable contract-product state.

Theorem 2 (Optimal Safe Recovery). *Let $\mathcal{L}_{\mathsf{cert}}$ be a certified library, and let $\pi^\star \in \arg\max_{\pi \in \Pi_{\mathsf{safe}}} J_{s_0}(\pi)$ be a deterministic Markov policy. If some $\mathcal{C}^\star \in \mathcal{L}_{\mathsf{cert}}$ satisfies $\pi^\star \in \Pi(\mathcal{C}^\star)$, then $\max\{J_{s_0}(\pi) \mid \mathcal{C} \in \mathcal{L}_{\mathsf{cert}},\ \pi \in \Pi(\mathcal{C})\} = J_{s_0}(\pi^\star)$. Equivalently, when empty inner feasible sets are ignored,*

$$\max_{\mathcal{C} \in \mathcal{L}_{\mathsf{cert}}} \max_{\pi \in \Pi(\mathcal{C})} J_{s_0}(\pi) = J_{s_0}(\pi^\star).$$

The proof is available in Appendix B.2. In particular, suppose $\pi^\star$ is a deterministic Markov optimum in Π_{safe}, the local alphabets are action-separating (e.g. by folding the last action into the state or using a non-Markovian labelling function) for $\pi^\star$, and the depth bound D and candidate family are rich enough that $\mathcal{L}_{\mathsf{cert}}$ contains a certified contract $\mathcal{C}^\star$ with

$$\mathcal{M}_{\shield}^{\mathcal{C}^\star,i}(x^{\mathcal{C}^\star}) = \{a_{\pi^\star}^i(s)\} \qquad \text{for every reachable } x^{\mathcal{C}^\star} = (s, \bar{q}) \text{ and every } i \in \mathsf{Ag},$$

then $\pi^\star \in \Pi(\mathcal{C}^\star)$, so Theorem 2 applies. The local masks also enforce the globally optimal safe joint action on all reachable states. These statements concern deterministic Markov optima. For the finite discounted, fully observable cooperative concurrent stochastic games considered in our setting, policy randomisation is not required to attain the optimal team value: an optimal deterministic Markov joint policy exists, although stochastic policies remain useful during learning [22, 26].

5.3 Bandit Selector

The learning-time selector is an outer loop over the certified library: it chooses which pre-certified contract (shield) governs each training block, while the inner MARL algorithm continues updating its policies under the currently active shields. Each certified contract is treated as an arm. Pulling arm $\mathcal{C}$ means training for one dwell block under $\mathcal{C}$'s shield and measuring the resulting team return. The selector is parameterised by:

– the finite certified library $\mathcal{L}_{\mathsf{cert}}$ of candidate contracts;
– the initial certified contract $\mathcal{C}_0 \in \mathcal{L}_{\mathsf{cert}}$;
– the warm-up horizon H_0, measured in completed episodes;
– the dwell length H, measured in completed episodes per arm pull;
– the discount factor $\eta \in (0, 1]$ for historical block statistics; and
– the exploration coefficient $\beta \geq 0$.

Let E_b be the completed episodes in a dwell block b run under contract $\mathcal{C}_b$, and let R_e^i be the episode return of agent i in episode e. The selector uses the normalised team-return score $y_b \stackrel{\text{def}}{=} \overline{R}_b \stackrel{\text{def}}{=} \frac{1}{|E_b|\, n} \sum_{e \in E_b} \sum_{i=1}^{n} R_e^i$, which is equivalent to the usual team return up to the constant factor n. The resulting bandit is *non-stationary*: an arm's payoff changes as the inner policies continue to learn under the active shields. Discounting therefore estimates recent block performance rather than a fixed stationary arm mean, following discounted UCB methods for non-stationary bandits [13, 15]. For each certified contract $\mathcal{C}$, maintain discounted count and score statistics $N_{\mathcal{C}}$ and $S_{\mathcal{C}}$. When a block under $\mathcal{C}_b$ completes, all arms are first discounted by setting $N_{\mathcal{C}} \leftarrow \eta N_{\mathcal{C}}$ and $S_{\mathcal{C}} \leftarrow \eta S_{\mathcal{C}}$ for every $\mathcal{C} \in \mathcal{L}_{\mathsf{cert}}$, and the observed arm is then updated by

$$N_{\mathcal{C}_b} \leftarrow N_{\mathcal{C}_b} + 1, \qquad S_{\mathcal{C}_b} \leftarrow S_{\mathcal{C}_b} + y_b, \qquad \widehat{\mu}_{\mathcal{C}'} \stackrel{\text{def}}{=} S_{\mathcal{C}'} / N_{\mathcal{C}'},$$

where $\widehat{\mu}_{\mathcal{C}'}$ is defined for each $\mathcal{C}'$ with $N_{\mathcal{C}'} > 0$. After H_0 completed warm-up episodes under $\mathcal{C}_0$, the selector accumulates dwell blocks of H completed episodes, tries certified contracts whose identifiers have not yet been visited, then choosing

$$\mathcal{C}_{b+1} \in \arg \max_{\substack{\mathcal{C} \in \mathcal{L}_{\mathsf{cert}} \\ N_{\mathcal{C}} > 0}} \left(\widehat{\mu}_{\mathcal{C}} + \beta \sqrt{\frac{\log(1 + \max\{1, \sum_{\mathcal{C}'} N_{\mathcal{C}'}\})}{N_{\mathcal{C}}}} \right),$$

where $\beta \geq 0$ is the exploration coefficient. Ties are resolved deterministically, preferring to keep the current contract and otherwise using the certified-library order. Algorithm 2 in Appendix E gives the discounted-UCB update.

Theorem 3 (Safety of Certified Selection). *Let $\mathcal{L}_{\mathsf{cert}}$ be a certified library. Consider any sequence of episodes in which each episode is assigned a contract $\mathcal{C} \in \mathcal{L}_{\mathsf{cert}}$, starts from an initial product state covered by $\mathcal{C}$'s certification, the shield for $\mathcal{C}$ is deployed for the whole episode, every generated joint action lies in $\mathsf{Adm}_{\mathcal{C}}(x_t^{\mathcal{C}})$ at the current contract product state, and contract switches occur only at episode reset boundaries. Then no generated episode prefix is a bad prefix of the global safety objective. Under the standard per-episode infinite-trace semantics, every episode satisfies Φ_{safe}.*

The proof is available in Appendix B.3. Theorem 3 allows the selector to adapt reward estimates and switch contracts between episodes, but it never admits an uncertified tuple. Thus the outer loop can optimise team return over the finite space of certified contracts while preserving deterministic safety. Theorem 3 does not by itself give a convergence guarantee for the discounted bandit selector; the optimal safe team return is attainable only insofar as it is represented by some

certified contract in the library and the learning dynamics select the corresponding profile. Without that representability assumption, performance is limited by the expressive power of the contract language and certified library.

6 Empirical Evaluation

We report the experimental configuration, benchmark suite, certified search counts, and learning curves. Appendix C gives environment details, labelling, and safety objectives; Appendix D reports full curves. The implementation is open-source and available at https://github.com/nightly/contract-shielding.

Implementation Details and Hyperparameters. The formal development uses a single global alphabet Ap and global labelling function $\mathcal{L}$. For tractability, the implementation stores per-agent sub-alphabets $\mathsf{Ap}_i \subseteq \mathsf{Ap}$ and compiles each local automaton over $\Sigma_i = 2^{\mathsf{Ap}_i}$; automaton i receives label $\mathcal{L}(s) \cap \mathsf{Ap}_i$ at state s. Certification checks consistency with the global model: each $\mathsf{Ap}_i \subseteq \mathsf{Ap}$, each local label is the corresponding projection, and propositions shared by multiple local alphabets agree, see Definition 3. We bound candidate formula depth by $D = 2$ (interpreting atoms as $D = 1$). Other hyperparameters are provided in Appendix F.

Algorithms. We compare independent learners (`IPPO` [8], `IQL` [27]), centralised-critic or value-decomposition learners (`MAPPO` [30], `Joint-PPO`, `PQN-VDN` [12, 25]), Lagrangian variants of `IPPO` and `IQL`, `ICPO` [1], ordinary shielded variants, and `Contract-IPPO`/`Contract-IQL`. `MAPPO` uses a centralised critic with decentralised actors; `Joint-PPO` and `Shielded-Joint-PPO` use centralised actors and critics and serve as upper-end reward baselines although outside our decentralised-execution setting.

Environments. Evaluation covered six cooperative MARL benchmarks: Flatland [20], Connector [5], Level-Based Foraging [7], RWARE [21], Pressure Plate [2], and Car Platoon [6]. Example 2 illustrates one safety property and contract; Appendix C expands the rest.

Example 2. In Pressure Plate, 2 holder agents open doors by standing on assigned plates while a runner crosses to the goal. Let Z be goal achievement; let I_k, A_k, and B_k mean that the runner is in door k, after door k, and able to return before door k; and let H_k mean that holder k has not abandoned plate k before the goal. The holder obligations let the runner's mask rely on held doors when admitting crossings; factorised masks must be robust to holder departures, so they conservatively keep the runner out of the protected door regions. The default return-route objective and one certified local contract are

$$\Phi^{\mathsf{pp}}_{\mathsf{safe}} \stackrel{\text{def}}{=} \mathsf{G} \bigwedge_{k=0}^{1} \big((\neg Z \wedge (I_k \vee A_k)) \rightarrow B_k\big),$$

$$\mathcal{C}^{\mathsf{pp}} \stackrel{\text{def}}{=} \langle \mathsf{G}H_0,\ \mathsf{G}H_1,\ \Phi^{\mathsf{pp}}_{\mathsf{safe}} \rangle.$$

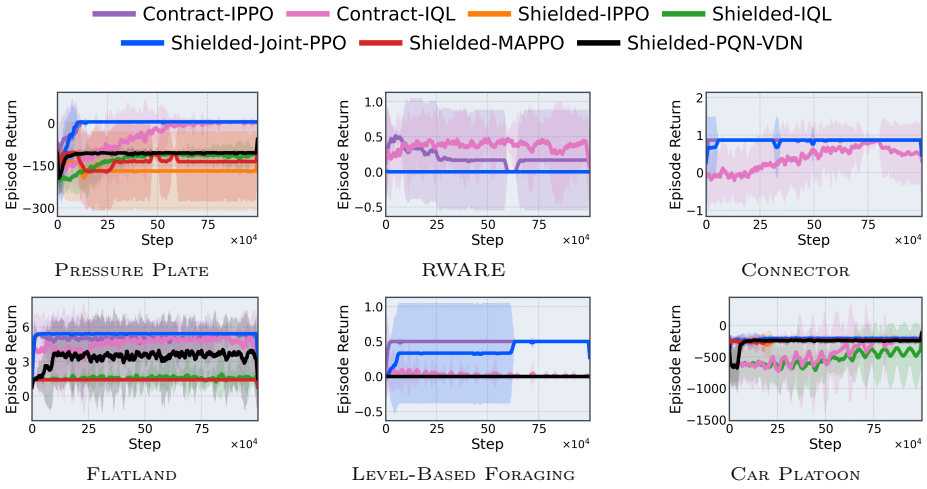

Fig. 4. Team-mean reward curves for all six benchmarks using shielded algorithms. Shaded regions are ± 95% confidence intervals across runs.

Analysis. Figure 4 shows the shielded and contract-aware reward curves. The plotted quantity is reward: safety is enforced structurally by certified masks and Theorems 1 and 3; the curves measure retained return under certified actions. Where ordinary factorised shields are deployable, contract shielding improves reward and converges toward `Shielded-Joint-PPO`. For RWARE, Pressure Plate, and Connector, the selected global specifications admit no factorised shield because masks must be conservative over other agents' actions. For Pressure Plate, shielded baselines use conservative local runner masks rather than a direct factorisation of $\Phi^{\mathsf{pp}}_{\mathsf{safe}}$ (Appendix C.1). In the largest environment instances, bounded search considered up to 12,001 profiles and certified up to 390 as safe; Table 1 in Appendix D gives all objectives and results.

7 Conclusion and Future Work

We presented contract-based decentralised shielding for cooperative MARL. Finite profiles of local $\mathsf{LTL}_{\mathsf{safe}}$ obligations are certified by a circular assume-guarantee fixed point, projected into local action masks, and selected during learning only from pre-certified libraries, separating reward adaptation from deterministic safety. Agents may rely on certified coordination promises while preserving one global safety objective. Across six benchmarks, contract-aware shields recover coordinated behaviours while retaining pre-emptive safety guarantees. Future work will study quantitative and probabilistic obligations, and heuristics for finding near-reward-optimal contracts without exhaustive enumeration.

Acknowledgments. The research described in this paper was partially supported by the EPSRC (grant number EP/X015823/1) and the UKRI Centre for Doctoral Training in Safe and Trusted AI (grant number EP/S0233356/1).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Bibliography

[1] Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained policy optimization. In: International conference on machine learning. pp. 22–31. Pmlr (2017)
[2] Ahmed, I.H., Brewitt, C., Carlucho, I., Christianos, F., Dunion, M., Fosong, E., Garcin, S., Guo, S., Gyevnar, B., McInroe, T., et al.: Deep reinforcement learning for multi-agent interaction. Ai Communications **35**(4), 357–368 (2022)
[3] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
[4] Baier, C., Katoen, J.P.: Principles of model checking. MIT press (2008)
[5] Bonnet, C., Luo, D., Byrne, D., Surana, S., Abramowitz, S., Duckworth, P., Coyette, V., Midgley, L., Tegegn, E., Kalloniatis, T., et al.: Jumanji: a diverse suite of scalable reinforcement learning environments in jax. In: International Conference on Learning Representations. vol. 2024, pp. 49264–49293 (2024)
[6] Brorholt, A.H., Larsen, K.G., Schilling, C.: Compositional shielding and reinforcement learning for multi-agent systems. arXiv preprint arXiv:2410.10460 (2024)
[7] Christianos, F., Schäfer, L., Albrecht, S.: Shared experience actor-critic for multi-agent reinforcement learning. Advances in neural information processing systems **33**, 10707–10717 (2020)
[8] De Witt, C.S., Gupta, T., Makoviichuk, D., Makoviychuk, V., Torr, P.H., Sun, M., Whiteson, S.: Is independent learning all you need in the starcraft multi-agent challenge? arXiv preprint arXiv:2011.09533 (2020)
[9] Drew, D.S.: Multi-agent systems for search and rescue applications. Current Robotics Reports **2**(2), 189–200 (2021)
[10] El Mqirmi, P., Belardinelli, F., León, B.G.: An abstraction-based method to check multi-agent deep reinforcement-learning behaviors. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems. pp. 474–482 (2021)
[11] ElSayed-Aly, I., Bharadwaj, S., Amato, C., Ehlers, R., Topcu, U., Feng, L.: Safe multi-agent reinforcement learning via shielding. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems. p. 483–491. AAMAS '21, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2021)
[12] Gallici, M., Fellows, M., Ellis, B., Pou, B., Masmitja, I., Foerster, J., Martin, M.: Simplifying deep temporal difference learning. In: International Conference on Learning Representations. vol. 2025, pp. 78148–78190 (2025)

[13] Garivier, A., Moulines, E.: On upper-confidence bound policies for switching bandit problems. In: International conference on algorithmic learning theory. pp. 174–188. Springer (2011)
[14] Ji, J., Zhang, B., Zhou, J., Pan, X., Huang, W., Sun, R., Geng, Y., Zhong, Y., Dai, J., Yang, Y.: Safety gymnasium: A unified safe reinforcement learning benchmark. Advances in Neural Information Processing Systems **36**, 18964–18993 (2023)
[15] Kocsis, L., Szepesvári, C.: Discounted ucb. In: 2nd PASCAL Challenges Workshop. vol. 2, pp. 51–134 (2006)
[16] Könighofer, B., Alshiekh, M., Bloem, R., Humphrey, L., Könighofer, R., Topcu, U., Wang, C.: Shield synthesis. Formal Methods in System Design **51**(2), 332–361 (2017)
[17] Kupferman, O., Vardi, M.Y.: Model checking of safety properties. Formal methods in system design **19**(3), 291–314 (2001)
[18] Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Machine learning proceedings 1994, pp. 157–163. Elsevier (1994)
[19] Melcer, D., Amato, C., Tripakis, S.: Shield decentralization for safe multi-agent reinforcement learning. Advances in Neural Information Processing Systems **35**, 13367–13379 (2022)
[20] Mohanty, S., Nygren, E., Laurent, F., Schneider, M., Scheller, C., Bhattacharya, N., Watson, J., Egli, A., Eichenberger, C., Baumberger, C., et al.: Flatland-rl: Multi-agent reinforcement learning on trains. arXiv preprint arXiv:2012.05893 (2020)
[21] Papoudakis, G., Christianos, F., Schäfer, L., Albrecht, S.V.: Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. arXiv preprint arXiv:2006.07869 (2020)
[22] Puterman, M.L.: Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons (2014)
[23] Raja, A., Barley, M., Zhang, X.S.: Towards safe coordination in multi-agent systems. In: Safety and Security in Multiagent Systems: Research Results from 2004-2006, pp. 1–7. Springer (2009)
[24] Sistla, A.P.: Safety, liveness and fairness in temporal logic. Formal Aspects of Computing **6**(5), 495–511 (1994)
[25] Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., et al.: Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:1706.05296 (2017)
[26] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)
[27] Tan, M., et al.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the tenth international conference on machine learning. pp. 330–337 (1993)
[28] Wurman, P.R., D'Andrea, R., Mountz, M.: Coordinating hundreds of cooperative, autonomous vehicles in warehouses. AI magazine **29**(1), 9–9 (2008)

[29] Xiao, W., Lyu, Y., Dolan, J.: Model-based dynamic shielding for safe and efficient multi-agent reinforcement learning. arXiv preprint arXiv:2304.06281 (2023)
[30] Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., Wu, Y.: The surprising effectiveness of ppo in cooperative, multi-agent games. Advances in neural information processing systems **35**, 24611–24624 (2022)

Appendix

The appendix is organised as follows:

Appendix A: $\mathsf{LTL}_{\mathsf{safe}}$ semantics are provided.
Appendix B: Technical results and proofs are provided.
Appendix C: Environment-level benchmark details are provided.
Appendix D: Complete experimental results are reported.
Appendix E: Technical remarks on contract selection are provided.
Appendix F: Hyperparameters are provided for replication.

A Safe LTL Semantics

Let $\mathcal{T} \stackrel{\text{def}}{=} (2^{\mathsf{Ap}})^{\omega}$ be the set of traces over Ap. Then, we can define the inductive semantics used in Section 3 as follows.

For any trace $\tau \in \mathcal{T}$, position $k \in \mathbb{N}$, and formulae ϕ, ψ:	
Trace semantics	
$(\tau, k) \models \top$	$\leftrightarrow$always
$(\tau, k) \models \bot$	$\leftrightarrow$never
$(\tau, k) \models p$	$\leftrightarrow p \in \tau[k]$ for $p \in \mathsf{Ap}$
$(\tau, k) \models \neg p$	$\leftrightarrow p \notin \tau[k]$ for $p \in \mathsf{Ap}$
$(\tau, k) \models \phi \wedge \psi$	$\leftrightarrow (\tau, k) \models \phi$ and $(\tau, k) \models \psi$
$(\tau, k) \models \phi \vee \psi$	$\leftrightarrow (\tau, k) \models \phi$ or $(\tau, k) \models \psi$
$(\tau, k) \models \mathsf{X}\phi$	$\leftrightarrow (\tau, k+1) \models \phi$
$(\tau, k) \models \phi \mathsf{W} \psi$	$\leftrightarrow \big(\forall m \geq k.\ (\tau, m) \models \phi\big)$ or $\big(\exists j \geq k.\ (\tau, j) \models \psi$ and $\forall m \in \{k, \ldots, j-1\}.\ (\tau, m) \models \phi\big)$
$(\tau, k) \models \mathsf{G}\phi$	$\leftrightarrow \forall j \geq k.\ (\tau, j) \models \phi$
Model satisfaction	
$\tau \models \phi$	$\leftrightarrow (\tau, 0) \models \phi$
$\mathcal{M}^{\pi} \models \phi$	$\leftrightarrow \mathcal{T}_{\mathcal{L}}(\rho) \models \phi$ for every $\rho \in \mathsf{IPath}^{\pi}(s_0)$

B Technical Results

This section provides all of the formal proofs of results stated in the main text.

B.1 Proof of Theorem 1. Circular Compositional Soundness

Theorem 1 (Circular Compositional Soundness (restated)). *Let $\mathcal{C} = \langle\varphi_i\rangle_{i=1}^n$ be a certified contract. For any contract-product execution $x_0^{\mathcal{C}} a_0 x_1^{\mathcal{C}} a_1 \cdots$ starting from $x_0^{\mathcal{C}}$, if $a_t \in \mathsf{Adm}_{\mathcal{C}}(x_t^{\mathcal{C}})$ at every time t, then every reachable contract product state remains in $\mathsf{Win}_{\mathcal{C}}^{\square}$. Consequently, the local obligations hold along the execution, and the induced global trace satisfies Φ_{safe}.*

Proof. We prove the invariant $x_t^{\mathcal{C}} \in \mathsf{Win}_{\mathcal{C}}^{\square}$ by induction on time (establishing the invariant for all $t \geq 0$). The base case is exactly the second certification condition, $x_0^{\mathcal{C}} \in \mathsf{Win}_{\mathcal{C}}^{\square}$. For the step case, assume $x_t^{\mathcal{C}} \in \mathsf{Win}_{\mathcal{C}}^{\square}$ and write $x_t^{\mathcal{C}} = (s_t, \bar{q}_t)$. Since the executed joint action satisfies $a_t \in \mathsf{Adm}_{\mathcal{C}}(x_t^{\mathcal{C}})$, Definitions 11 and 13 give

$$a_t \in \prod_{i \in \mathsf{Ag}} R_{\mathcal{C}}^i(x_t^{\mathcal{C}}) = R_{\mathcal{C}}(x_t^{\mathcal{C}}).$$

By the choice of the certified rectangle, $R_{\mathcal{C}}(x_t^{\mathcal{C}}) \in \mathsf{Rect}_{\mathcal{C}}(x_t^{\mathcal{C}}) = \mathsf{Rect}_{\mathcal{C}}(\mathsf{Win}_{\mathcal{C}}^{\square}, x_t^{\mathcal{C}})$. Thus $R_{\mathcal{C}}(x_t^{\mathcal{C}})$ is a $\mathsf{Win}_{\mathcal{C}}^{\square}$-closed local action rectangle at $x_t^{\mathcal{C}}$; in particular, a_t is legal at s_t and

$$\mathrm{Post}_{\mathcal{C}}(x_t^{\mathcal{C}}, a_t) \subseteq \mathsf{Win}_{\mathcal{C}}^{\square}.$$

Every possible next product state $x_{t+1}^{\mathcal{C}}$ is in this successor set, hence $x_{t+1}^{\mathcal{C}} \in \mathsf{Win}_{\mathcal{C}}^{\square}$.

Because $\mathsf{Win}_{\mathcal{C}}^{\square} = \mathcal{T}_{\mathcal{C}}(\mathsf{Win}_{\mathcal{C}}^{\square})$, every state in $\mathsf{Win}_{\mathcal{C}}^{\square}$ is locally safe. For each agent i and time t, the automaton component of $x_t^{\mathcal{C}}$ is

$$q_{i,t} = \delta_i^*(q_i^0, \ell_i(s_0)\ell_i(s_1)\cdots\ell_i(s_t)),$$

by the definition of the contract product. Since $q_{i,t} \notin B_i$ for every t, no finite prefix of the projected local trace is a bad prefix. By Definition 6, the projected local trace satisfies φ_i for every agent i. The lifted entailment condition $\mathcal{C} \models_{\uparrow} \Phi_{\mathsf{safe}}$ in Definition 10 then implies that the global trace satisfies Φ_{safe}.

B.2 Proof of Theorem 2. Optimal Safe Recovery

Theorem 2 (Optimal Safe Recovery (restated)). *Let $\mathcal{L}_{\mathsf{cert}}$ be a certified library, and let $\pi^\star \in \arg\max_{\pi \in \Pi_{\mathsf{safe}}} J_{s_0}(\pi)$ be a deterministic Markov policy. If some $\mathcal{C}^\star \in \mathcal{L}_{\mathsf{cert}}$ satisfies $\pi^\star \in \Pi(\mathcal{C}^\star)$, then $\max\{J_{s_0}(\pi) \mid \mathcal{C} \in \mathcal{L}_{\mathsf{cert}},\ \pi \in \Pi(\mathcal{C})\} = J_{s_0}(\pi^\star)$. Equivalently, when empty inner feasible sets are ignored,*

$$\max_{\mathcal{C} \in \mathcal{L}_{\mathsf{cert}}} \max_{\pi \in \Pi(\mathcal{C})} J_{s_0}(\pi) = J_{s_0}(\pi^\star).$$

Proof. Fix any certified $\mathcal{C} \in \mathcal{L}_{\mathsf{cert}}$ and any $\pi \in \Pi(\mathcal{C})$. We show that every contract-product execution consistent with π uses only admitted actions. The initial product state is in $\mathsf{Reach}_{\mathcal{C}}(\pi)$ by definition. If $x_t^{\mathcal{C}} = (s_t, \bar{q}_t) \in \mathsf{Reach}_{\mathcal{C}}(\pi)$, compatibility gives $\mathrm{Supp}(\pi(\cdot \mid s_t)) \subseteq \mathsf{Adm}_{\mathcal{C}}(x_t^{\mathcal{C}})$; therefore every action chosen with positive probability by π at s_t is admitted. For any successor $x_{t+1}^{\mathcal{C}} \in \mathrm{Post}_{\mathcal{C}}(x_t^{\mathcal{C}}, a_t)$, closure of $\mathsf{Reach}_{\mathcal{C}}(\pi)$ gives $x_{t+1}^{\mathcal{C}} \in \mathsf{Reach}_{\mathcal{C}}(\pi)$. By induction, all

actions along every execution consistent with π are admitted by the deployed shields.

Theorem 1 then implies that every such execution satisfies Φ_{safe}, so $\pi \in \Pi_{\mathsf{safe}}$. Thus every feasible pair $(\mathcal{C}, \pi)$ in the certified-shield optimisation has value at most $\max_{\pi \in \Pi_{\mathsf{safe}}} J_{s_0}(\pi) = J_{s_0}(\pi^\star)$. Conversely, by the hypothesis, the feasible pair $(\mathcal{C}^\star, \pi^\star)$ attains $J_{s_0}(\pi^\star)$. The two inequalities give the claimed equality.

B.3 Proof of Theorem 3. Certified Selection Safety

Theorem 3 (Safety of Certified Selection (restated)). *Let $\mathcal{L}_{\mathsf{cert}}$ be a certified library. Consider any sequence of episodes in which each episode is assigned a contract $\mathcal{C} \in \mathcal{L}_{\mathsf{cert}}$, starts from an initial product state covered by $\mathcal{C}$'s certification, the shield for $\mathcal{C}$ is deployed for the whole episode, every generated joint action lies in $\mathsf{Adm}_{\mathcal{C}}(x_t^{\mathcal{C}})$ at the current contract product state, and contract switches occur only at episode reset boundaries. Then no generated episode prefix is a bad prefix of the global safety objective. Under the standard per-episode infinite-trace semantics, every episode satisfies Φ_{safe}.*

Proof. Fix an arbitrary episode and let $\mathcal{C}$ be its assigned contract. Since $\mathcal{C} \in \mathcal{L}_{\mathsf{cert}}$, the definition of the certified library gives that $\mathcal{C}$ is certified. By hypothesis, the episode begins at an initial contract product state in $\mathsf{Win}_{\mathcal{C}}^{\square}$, and the shield for $\mathcal{C}$ is used for every step of the episode with generated actions in $\mathsf{Adm}_{\mathcal{C}}(x_t^{\mathcal{C}})$. Theorem 1 therefore applies throughout the episode: all reachable contract product states remain in $\mathsf{Win}_{\mathcal{C}}^{\square}$, no contract safety automaton reaches a bad state, and the global safety objective is satisfied. Hence no generated episode prefix is a bad prefix of the global safety objective. Because switches occur only between episodes, the same argument restarts independently for each selected contract.

C Environments

This section provides details for all of the environments considered in our empirical evaluation of Section 6. We use finite safety abstractions for contract and shield synthesis. For the safety game, the abstraction is required to preserve the successor support relevant to the automata and the truth of all propositions appearing in the global and local obligations.

Figure 5 shows renders of the six benchmark environments described below.

C.1 Pressure Plate

Pressure Plate is a role-structured grid task. Let $G \stackrel{\text{def}}{=} \{0, \ldots, H-1\} \times \{0, \ldots, W-1\}$ be the grid, let $m \stackrel{\text{def}}{=} n-1$ be the number of doors and plates, and let agent $n-1$ be the runner. An abstract state is

$$s \stackrel{\text{def}}{=} (p_0, \ldots, p_{n-1}, C, b_0, \ldots, b_{m-1}, o_0, \ldots, o_{m-1}, g, V_H, V_W, t),$$

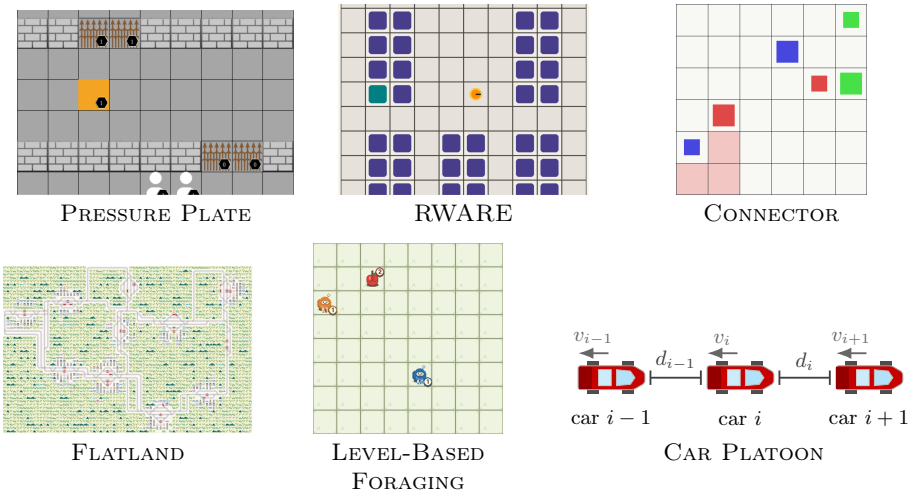

Fig. 5. Renders of the six benchmark environments used in the evaluation.

where $p_i \in G$ is an agent position, $C \subseteq \{0, \ldots, m-1\}$ is a one-step latch recording doors crossed by the runner on the preceding transition, $b_k, o_k, g \in \{0, 1\}$ record whether plate k is pressed, door k is open, and the goal is reached, V_H records door-holder protocol violations, V_W records runner wait-for-door violations, and t is the step count. Each agent has primitive actions $\{\mathsf{Up}, \mathsf{Down}, \mathsf{Left}, \mathsf{Right}, \mathsf{Noop}\}$. The transition samples a permutation of the agents, applies their proposed moves sequentially, and cancels moves into grid boundaries, walls, closed doors, or occupied cells. After movement, plate k is pressed iff the assigned holder k occupies it, and door k is open iff that plate is pressed. The latch C' contains door k exactly when the runner entered a cell of door k, left such a cell toward the far side, or crossed the door boundary during the transition; it is empty on non-crossing transitions. The episode terminates when some agent reaches the goal or the horizon is met.

Rewards are role-shaped: holders receive dense shaping toward their assigned plates, while the runner is shaped toward the goal. The reported experiments use distance normalisation $D_{\max} = 10$, step cost 0.01, and goal bonus 10.

For holder $i < m$, the local contract alphabet contains its door-holding proposition, door i open and plate-pressed propositions, holder-on-plate propositions, goal achievement, and the runner's waiting, crossing, in-door, and after-door phase propositions for door i. The runner's local alphabet contains, for every door, its wait-for-door, waiting, crossing, in-door, after-door, door-open, and return-route propositions, together with goal achievement. The global alphabet contains these propositions and the finite agent-cell, agent-at-goal, agent-before-door, and plate/door status propositions needed to interpret the benchmark labels.

The default benchmark instance uses the linear layout with $H = 7$, $W = 4$, $n = 3$, sensor range 3, and horizon 150. Let A_k, I_k, B_k, and Z respectively

abbreviate the runner-after-door, runner-in-door, runner-can-return before door, and goal-achieved propositions. The default safety formula is

$$\Phi^{\mathsf{pp}}_{\mathsf{safe}} \stackrel{\text{def}}{=} \mathsf{G} \bigwedge_{k\in\{0,1\}} \big((\neg Z \wedge (I_k \vee A_k)) \rightarrow B_k\big).$$

The return-route label B_k is computed by reachability from the runner's current cell to a cell before door k, treating walls and closed doors as blocked and ignoring teammate occupancy. A direct generic per-agent shield for this formula is not realisable from the initial state, so the ordinary shielded baseline uses the conservative runner-only formulas

$$\begin{aligned}
\varphi_0 &\stackrel{\text{def}}{=} \top,\\
\varphi_1 &\stackrel{\text{def}}{=} \top,\\
\varphi_2 &\stackrel{\text{def}}{=} \mathsf{G}\neg(I_0 \vee A_0 \vee I_1 \vee A_1),
\end{aligned}$$

while the global automaton still checks $\Phi^{\mathsf{pp}}_{\mathsf{safe}}$. Certified contract profiles can instead assign holder obligations that preserve the runner's return route while the runner carries the global return-route obligation.

C.2 RWARE

RWARE is a warehouse queue and delivery benchmark. A state is

$$s \stackrel{\text{def}}{=} (u_1, \ldots, u_n, \sigma_1, \ldots, \sigma_q, Q, Y, t, \iota),$$

where $u_i \stackrel{\text{def}}{=} (x_i, y_i, d_i, \ell_i, c_i, h_i)$ records robot position, direction, loaded status, whether the carried shelf is requested, and whether a delivery has occurred; σ_j records shelf identities and positions; Q is the request queue; Y records standing and requested shelf locations; and t, ι are the total and inactive step counts. Each robot has actions $\{\mathsf{NOOP}, \mathsf{FORWARD}, \mathsf{LEFT}, \mathsf{RIGHT}, \mathsf{TOGGLE\_LOAD}\}$. Turning updates d_i, forward movement requests the cell in direction d_i, and load toggling picks up or drops a shelf when the warehouse interaction is legal. The transition resolves collisions using separate robot and shelf occupancy layers; a loaded robot occupies both layers. When a requested shelf reaches a goal cell, the request is replaced by a new request according to the environment kernel.

Rewards are sparse and delivery based: in the default individual reward mode, robot i receives reward only when it delivers a requested shelf, with no movement reward or direct penalty for protocol violations.

The local contract alphabet for robot i contains the queue-yield proposition `agent_i_queue_yield_ok`. The proposition is false on a transition precisely when robot i is unloaded, occupies the next forward cell of a robot carrying a requested shelf in the queue, and remains in that blocking cell after joint-action execution and collision resolution. The global alphabet additionally contains finite robot-position propositions, highway and zone-membership

propositions, loaded and carrying-requested status, requested-shelf locations and at-requested-shelf facts, requested-carrier status, blocking-requested-carrier facts, and delivery-lane and requested-carrier-progress protocol propositions. Writing $q_i \stackrel{\text{def}}{=} \texttt{agent\_i\_queue\_yield\_ok}$, the default safety formula is the per-robot queue-yield invariant

$$\Phi^{\mathsf{rw}}_{\mathsf{safe}} \stackrel{\text{def}}{=} \bigwedge_{i=0}^{n-1} \mathsf{G} q_i.$$

The default queue-conflict instance uses $n = 2$, shelf columns 3, column height 8, one shelf row, sensor range 1, flattened observations, request queue size 2, individual rewards, no inactivity cutoff, and horizon 200.

Ordinary factorised shielding is not realisable for this default queue-yield objective. The proposition q_i records a joint transition fact. A factorised shield exposes independent local action sets, so every combination of individually permitted actions must remain safe against arbitrary teammate choices. In the queue-conflict layout, reachable states can require the blocker and carrier to take compatible same-step actions; no single local blocker action is certified safe against all carrier actions, and the projected local mask can become empty. Contract shields avoid this failure by first certifying teammate obligations, so local masks are checked against obligation-constrained teammates rather than the full adversarial action set.

C.3 Connector

Connector is a grid routing task. A state is

$$s \stackrel{\text{def}}{=} (B, p_0, \ldots, p_{n-1}, z_0, \ldots, z_{n-1}, V, t),$$

where $B \in \{0, \ldots, 3n\}^G$ is the grid of empty cells, trails, current positions, and targets; $p_i, z_i \in G$ are current and target cells; V records reservation violations; and t is the step count. Each agent has actions $\{\mathsf{Noop}, \mathsf{Up}, \mathsf{Right}, \mathsf{Down}, \mathsf{Left}\}$. The action mask permits a movement only into an empty cell or the agent's own target; illegal actions have no movement effect. Simultaneous proposed moves are computed from the current positions. Destination conflicts are resolved by a fixed agent-index priority rule: an agent whose proposed destination is also proposed by a later-indexed agent is cancelled, while successful moves leave a path marker behind, and reaching the target changes the target marker into the agent position marker. The episode terminates when all agents are connected or blocked, or when the horizon is met.

Dense rewards give each unconnected agent a connection bonus and a per-step cost until it reaches its target. The default constants are $r_{\mathsf{conn}} = 1$ and $r_{\mathsf{step}} = -0.03$.

The local contract alphabet for agent i contains its aggregate reservation-respect and route-clearance propositions, pairwise propositions stating that agent i keeps another agent's reserved route clear and respects that agent's reservation, and every agent's reserved-route-clear proposition. The global alphabet contains

finite current-position and target-position propositions, action-availability propositions, connection and blocked status, reserved-route-blocked and on-reserved-route propositions, and the aggregate and pairwise reservation-respect and route-clearance propositions. Let $r_i \stackrel{\text{def}}{=} \texttt{agent\_i\_reserved\_route\_clear\_ok}$. The default safety formula is

$$\Phi^{\mathsf{co}}_{\mathsf{safe}} \stackrel{\text{def}}{=} \bigwedge_{i=0}^{n-1} \mathsf{G} r_i.$$

The default reservation-conflict experiment uses the `reservation_priority_conflict` generator with fixed layout seed 0, a 5×5 grid, $n = 2$, dense rewards, the reserved cell $(2, 2)$ for agent 0, and horizon 12.

Ordinary factorised shielding is not realisable for this default reserved-route-clearance objective. The proposition r_i is owned by agent i but can be falsified by a teammate entering or remaining on agent i's reserved route. A local shield for the route owner must therefore be safe against all teammate actions, yet the owner has no action that can force the teammate off its reserved cells. Conversely, the teammate's factorised mask cannot express the conditional commitment "avoid the owner's reserved route while the owner is still unconnected" unless that commitment appears as part of a certified contract profile. The Cartesian product of ordinary local masks therefore cannot deploy a non-empty shield that preserves the global route-clearance invariant from the reservation-priority layout, whereas a contract profile can add explicit reservation-respect obligations.

C.4 Flatland

Flatland is a railway-routing benchmark. A state is $s \stackrel{\text{def}}{=} (\xi_0, \ldots, \xi_{n-1}, D, C, V, t)$, where each train state $\xi_i \stackrel{\text{def}}{=} (p_i, d_i, \kappa_i, z_i, p_i^0, d_i^0)$ contains current position, direction, Flatland train status, target, and initial placement; D is the set of deadlocked trains; C is the set of trains stopped by conflicts; V records pairwise yield violations; and t is the elapsed step count. The primitive actions are the five Flatland rail actions: do nothing, turn left, move forward, turn right, and stop. The transition is Flatland's `RailEnv` transition restricted by the rail graph, train status, malfunctions, and conflict resolution. In the single-track benchmark this transition is deterministic after the fixed scenario is generated.

Progress rewards give credit for reducing shortest-path distance to the target, with a per-step cost, arrival bonus, and deadlock penalty. The default constants are $c_{\mathsf{prog}} = 1$, $c_{\mathsf{step}} = 0.01$, $c_{\mathsf{arr}} = 5$, and $c_{\mathsf{dead}} = 5$.

The local contract alphabet for train i contains all trains' deadlock propositions and the pairwise yield propositions $\texttt{agent\_i\_yields\_to\_agent\_j\_ok}$ for $j \neq i$. The global alphabet contains finite train-position propositions, target and off-map status, Flatland train-status propositions, deadlock and conflict-stop propositions, and pairwise yield-protocol propositions. The default safety formula is

$$\Phi^{\mathsf{fl}}_{\mathsf{safe}} \stackrel{\text{def}}{=} \bigwedge_{i=0}^{n-1} \mathsf{G}\neg(\texttt{agent\_i\_deadlocked}).$$

The default instance is `single_track_meet` with $n = 2$, grid 7×3, unit-speed trains, tree depth 1, predictor depth 10, progress rewards, `team_done` termination, no malfunctions, and the horizon is 12. Candidate contracts can add pairwise yield obligations while preserving the global no-deadlock objective.

C.5 Level-Based Foraging

Level-Based Foraging models cooperative loading. A state is

$$s \stackrel{\text{def}}{=} (p_0, \ell_0, \ldots, p_{n-1}, \ell_{n-1}, F, V_F, V_S, V_L, V_C, t),$$

where $p_i \in G$ and ℓ_i are agent positions and levels; F is the finite set of food triples (x, y, ℓ); V_F, V_S, V_L, V_C record failed loads, successful loads, load attempts, and cooperative-load protocol violations; and t is the step count. Each agent has actions $\{\mathsf{NONE}, \mathsf{NORTH}, \mathsf{SOUTH}, \mathsf{WEST}, \mathsf{EAST}, \mathsf{LOAD}\}$. Invalid state-dependent actions become NONE, movement conflicts cancel all colliding moves, and a LOAD action affects a food item only when the loader is adjacent to that food. If the total level of adjacent loaders is below the food level, the load fails; otherwise the food is collected.

Rewards are assigned only by loading: failed load attempts are penalised, and successful loads give level-weighted credit to participating agents, normalised by the total spawned food level in the default configuration.

The local contract alphabet for agent i contains `agent_i_coop_load_ok` and every agent's failed-load proposition. The global alphabet contains food-availability, all-food-collected, joint-load-ready, food-position, and food-level propositions, together with finite agent-position propositions and per-agent load-attempt, successful-load, failed-load, cooperative-load, adjacent-food, needs-partner, partner-adjacent-same-food, and can-load-solo propositions. The default safety formula is

$$\Phi^{\mathsf{lbf}}_{\mathsf{safe}} \stackrel{\text{def}}{=} \bigwedge_{i=0}^{n-1} \mathsf{G}\neg(\texttt{agent\_i\_failed\_load}).$$

The default experiment uses a 5×5 field, $n = 2$, one level-2 food item, two level-1 agents, `force_coop=true`, normalised rewards, failed-load penalty 1, sight radius 5, fixed layout seed 0, vector observations, and horizon 25. Candidate contracts can add cooperative-load obligations using `agent_i_coop_load_ok` while preserving the global no-failed-load objective.

C.6 Car Platoon

Car Platoon has one uncontrolled lead car and $n = N - 1$ controlled followers, and is based on the environment described in [6]. Cars are ordered front to back. A state is

$$s \stackrel{\text{def}}{=} (v_0, \ldots, v_{N-1}, d_0, \ldots, d_{n-1}, \Delta_0, \ldots, \Delta_{N-1}, V_C, V_S, t),$$

where v_k is car velocity, d_i is the following gap between car i and car $i+1$, Δ_k records damage, V_C, V_S record conservative-follow and smooth-lead protocol violations, and t is the step count. Controlled agents choose brake, coast, or accelerate, mapped to accelerations $-2, 0, 2$, while the lead car follows a velocity-dependent stochastic policy. Velocities are clipped to $[v_{\min}, v_{\max}]$, and each gap is updated by trapezoidal integration of relative velocity:

$$d_i' = d_i + \frac{(v_i - v_{i+1}) + (v_i' - v_{i+1}')}{2}\Delta t.$$

Contact damages the adjacent cars, and a gap is unsafe when $d_i' \leq d_{\min}$ or $d_i' \geq d_{\max}$. Rewards penalise large following gaps and unsafe-gap violations. The default experiment uses $N = 3$, $\Delta t = 1$, velocity bounds $[v_{\min}, v_{\max}] = [0, 2]$, $d_{\min} = 0$, $d_{\max} = 20$, initial gap 10, safety-violation penalty 100, and termination on violation.

The local contract alphabet for controlled follower i contains its gap-safe, conservative-follow, and smooth-lead protocol propositions. The global alphabet contains discretised velocity, front-velocity, and following-distance propositions for each controlled follower, together with gap-safe, crashed, too-far, near-boundary, closing/opening-fast safe-to-accelerate-if-front-coasts, damage, and protocol propositions. Writing $g_i \stackrel{\text{def}}{=} \texttt{agent\_i\_gap\_safe}$, the default safety formula is

$$\Phi_{\mathsf{safe}}^{\mathsf{cp}} \stackrel{\text{def}}{=} \bigwedge_{i=0}^{n-1} \mathsf{G} g_i.$$

Candidate contracts can add conservative-follow or smooth-lead obligations while keeping this global gap-safety objective fixed.

D Extended Experimental Results

This appendix reports the extended empirical material omitted from the main paper. For each benchmark, the extended results record complete learning curves, safety-violation counts, shield-intervention statistics, and certified-contract selection traces when multiple certified profiles are available. Experiments were run on a Linux workstation with an AMD EPYC 7702P 64-core CPU and 64 logical cores and 203.48 GiB system RAM, and a single NVIDIA A16 GPU with 14.6 GiB memory. Table 1 reports the global safety objectives, bounded-search profile counts, and additional-obligation profile counts used in the main evaluation.

Tables 2, 3, and 4 report the training timings.

The extended plots are reported in Figures 6, 6, 7, 7, 8, and 8.

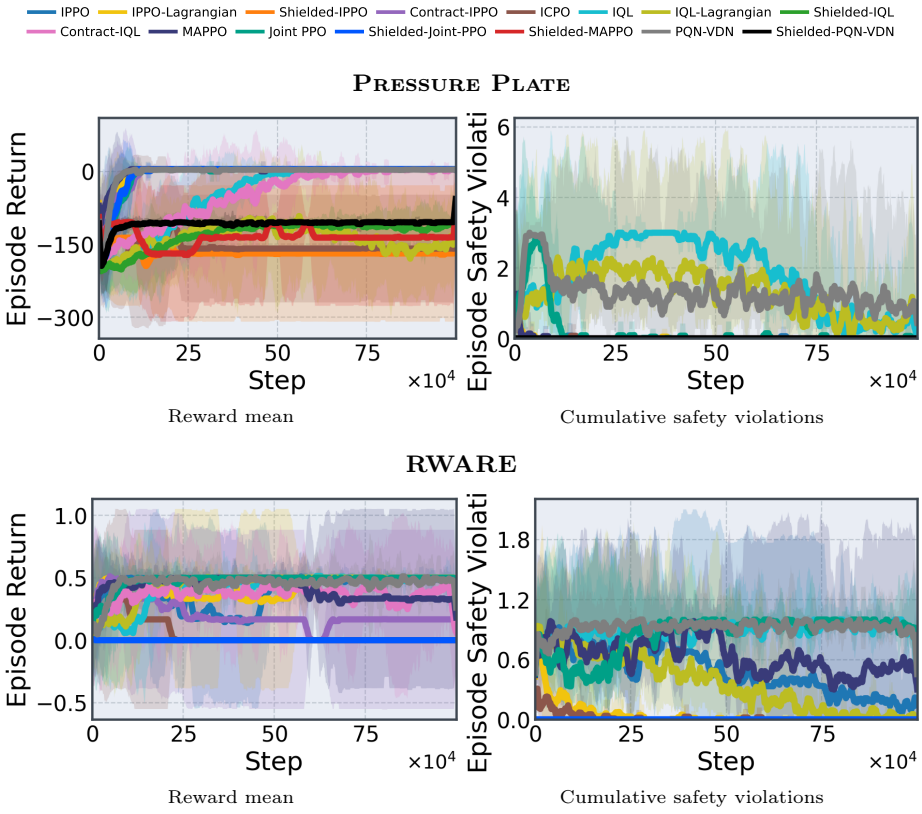

Fig. 6. Learning curves for Pressure Plate and RWARE. Shaded regions are 95% confidence intervals across runs.

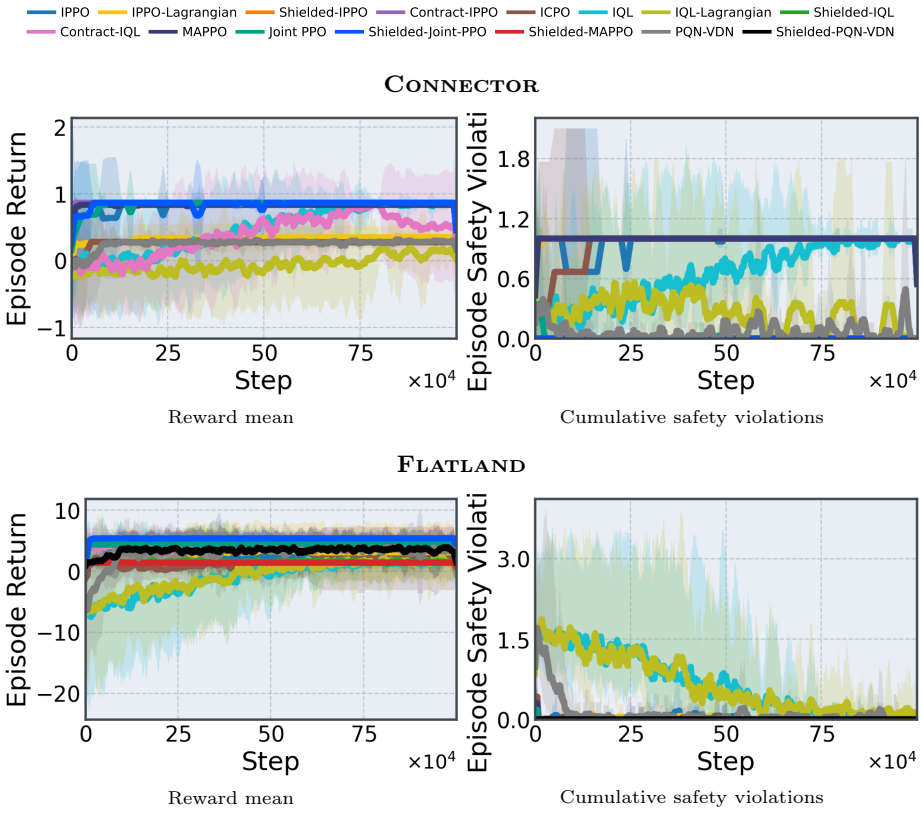

Fig. 7. Learning curves for Connector and Flatland. Shaded regions are 95% confidence intervals across runs.

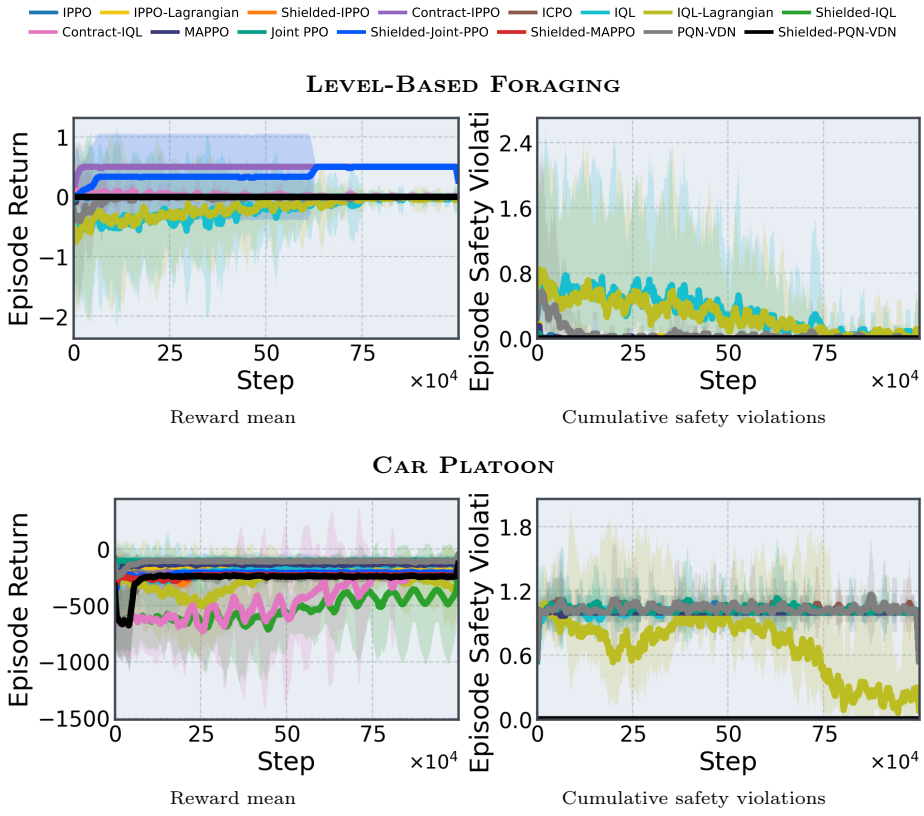

Fig. 8. Learning curves for Level-Based Foraging and Car Platoon. Shaded regions are 95% confidence intervals across runs.

Environment	Safety $\mathsf{LTL}_{\mathsf{safe}}$ formula	Profiles considered	Profiles certified safe	Additional-obligation profiles
Car Platoon	$\bigwedge_{i \in \text{all\_agents}} \mathsf{G}\,\text{agent}_i\text{\_gap\_safe}$	12,001	22	1
Connector	$\bigwedge_{i \in \text{all\_agents}} \mathsf{G}\,\text{agent}_i\text{\_reserved\_route\_clear\_ok}$	4,097	81	1
Flatland	$\bigwedge_{i \in \text{all\_agents}} \mathsf{G}\neg\text{agent}_i\text{\_deadlocked}$	4,097	337	2
Level-Based Foraging	$\bigwedge_{i \in \text{all\_agents}} \mathsf{G}\neg\text{agent}_i\text{\_failed\_load}$	4,097	390	1
Pressure Plate	$\mathsf{G} \bigwedge_{k \in \{0,1\}} ((\neg Z \wedge (I_k \vee A_k)) \rightarrow B_k)$	2	1	1
RWARE	$\bigwedge_{i \in \text{all\_agents}} \mathsf{G}\,\text{agent}_i\text{\_queue\_yield\_ok}$	1,369	16	0

Table 1. Safety objectives are shown as global $\mathsf{LTL}_{\mathsf{safe}}$ formulae alongside exported bounded-search profile counts. The final column counts retained profiles whose obligations are stricter than the global safety formula. For Pressure Plate, Z, I_k, A_k, and B_k abbreviate goal achievement, runner in-door, runner after-door, and runner return-route propositions.

Environment	IPPO	IPPO-Lagrangian	ICPO	MAPPO	Joint-PPO
Pressure Plate	0.81	0.84	0.85	0.85	0.88
RWARE	0.90	0.92	0.95	0.88	0.89
Connector	0.81	0.84	0.93	0.92	0.95
Flatland	1.04	1.04	1.09	1.12	1.14
Level-Based Foraging	0.73	0.73	0.78	0.86	0.82
Car Platoon	0.72	0.73	0.79	0.78	0.77

Table 2. Policy-gradient baseline training timings. Entries are mean wall-clock hours over 3 runs.

Environment	IQL	IQL-Lagrangian	PQN-VDN
Pressure Plate	0.53	0.55	0.67
RWARE	0.57	0.57	0.62
Connector	0.59	0.60	0.69
Flatland	0.82	0.83	0.90
Level-Based Foraging	0.47	0.46	0.60
Car Platoon	0.48	0.48	0.63

Table 3. Value-based training timings. Entries are mean wall-clock hours over 3 runs.

Environment	Shielded IPPO	Contract IPPO	Shielded IQL	Contract IQL	Shielded MAPPO	Shielded Joint-PPO	Shielded PQN-VDN
Pressure Plate	0.92	0.85	0.65	0.57	0.94	1.10	0.77
RWARE	N/A	0.96	N/A	0.64	N/A	1.23	N/A
Connector	N/A	0.88	N/A	0.62	N/A	1.07	N/A
Flatland	1.06	1.04	0.88	0.85	1.17	1.23	0.95
Level-Based Foraging	0.81	0.80	0.55	0.54	0.89	0.92	0.67
Car Platoon	0.77	0.73	0.52	0.50	0.83	0.82	0.68

Table 4. Shielded and contract-aware training timings. Entries are mean wall-clock hours over 3 runs; N/A marks intentionally omitted baselines.

Algorithm 2: Discounted-UCB contract selection

Input: initial certified contract $\mathcal{C}_0$, certified library $\mathcal{L}_{\mathsf{cert}}$, warm-up H_0, dwell time H, discount η, exploration coefficient β

deploy $\mathcal{C}_0$ for the first H_0 completed episodes
initialise discounted statistics $N_{\mathcal{C}}, S_{\mathcal{C}} \leftarrow 0$ for every $\mathcal{C} \in \mathcal{L}_{\mathsf{cert}}$
set $\mathcal{C} \leftarrow \mathcal{C}_0$ and visited set $V \leftarrow \emptyset$
while *training continues* **do**
  deploy $\mathcal{C}$ for one dwell block of H completed episodes
  observe per-agent episode returns and set $y \leftarrow (Hn)^{-1} \sum_{e=1}^{H} \sum_{i=1}^{n} R_e^i$
  foreach $\mathcal{C}' \in \mathcal{L}_{\mathsf{cert}}$ **do**
    update $N_{\mathcal{C}'} \leftarrow \eta N_{\mathcal{C}'}$ and $S_{\mathcal{C}'} \leftarrow \eta S_{\mathcal{C}'}$
  update $N_{\mathcal{C}} \leftarrow N_{\mathcal{C}} + 1$ and $S_{\mathcal{C}} \leftarrow S_{\mathcal{C}} + y$
  update $V \leftarrow V \cup \{\mathcal{C}\}$
  if *some* $\mathcal{C}' \in \mathcal{L}_{\mathsf{cert}}$ *has* $\mathcal{C}' \notin V$ **then**
    set $\mathcal{C}$ to the first unvisited certified contract in library order
  else
    set $\mathcal{C} \leftarrow \arg\max_{\mathcal{C}' \in \mathcal{L}_{\mathsf{cert}}} \left(S_{\mathcal{C}'}/N_{\mathcal{C}'} + \beta \sqrt{\log(1 + \max\{1, \sum_{\tilde{\mathcal{C}}} N_{\tilde{\mathcal{C}}}\})/N_{\mathcal{C}'}} \right)$

E Technical Remarks

Policies and shielded controllers. The set Π_{safe} in Section 3 records safety over environment traces. Once a contract is fixed, the current controller is naturally Markov on the contract product $x^{\mathcal{C}} = (s, \bar{q})$. However, viewed on the original environment, $\bar{q}$ is finite memory updated from the label stream. Definition 12 states when each agent can compute its own product-indexed mask without runtime communication. Thus safety claims for shielded policies mean that the product controller satisfies Φ_{safe}, equivalently that its finite-memory projection lies in Π_{safe}.

Algorithms. The learner suite includes independent policy-gradient and value-based baselines (`IPPO` [8], `IQL` [27]), centralised-critic cooperative baselines (`MAPPO` [30]), constrained baselines, ordinary shields, and contract-aware variants. `PQN-VDN` is a Parallelised Q-Network learner whose selected per-agent Q-values are summed through a VDN team value [12, 25]; `ICPO` is the independent `CPO` baseline, using cost critics, conjugate-gradient trust-region steps, and backtracking line search [1].

Synchronisation for contract selection. During learning, contract selection requires infrequent synchronisation among agents: they agree on the active certified contract and may exchange scalar performance summaries (to aggregate team reward where rewards are not homogeneous), used to test alternatives. This communication occurs only at episode or dwell-block boundaries and is infrequent relative to environment interaction. In particular, dwell block boundaries can be made extremely large, although this slows adaptation toward the best-performing

certified contract. It is not part of the runtime safety argument, which is carried by the deployed local certificates; at execution time no such communication is required once local mask identifiability holds.

Training overhead. The timing tables in Appendix D show no material overhead from contract selection relative to factorised shielded training, excluding time required to generate the initial library before training begins. Selection is evaluated only at dwell-block boundaries over a finite pre-certified library, so its cost is small compared with environment interaction and policy updates.

Decentralised common reward payoff. Although our evaluation uses decentralised-training-decentralised-execution algorithms, cooperative MARL optimises team reward. In our experimental configuration, individual rewards are generally aligned with team success, so independent optimisation can improve team return indirectly, even though rewards are neither a homogeneous common payoff nor broadcast to all agents. However, the bandit selector uses team return. All agents share the same task-level preference ordering, so the sum of individual returns is the natural cooperative objective rather than a centralised source of additional safety information.

F Hyperparameters

This appendix records the hyperparameters used for experimental replication. All runs used the same environment configuration, policy optimiser settings, shield-construction parameters, and contract-selection parameters reported here. Tables 5, 6, 7, and 8 collect the concrete settings.

Run budget and benchmark discounts. Each experiment setting is run three times with seeds $0, 1, 2$ and a budget of 10^6 environment steps per run. All runs use one parallel environment, rollouts of length 128, four update epochs, and four minibatches per update; for PPO-family methods this gives minibatches of size 32. The benchmark-level discount factor and episode cap are:

Benchmark	Safety formula	γ	Episode cap
Pressure Plate	`runner_requires_return_routes`	0.99	150
RWARE	`queue_yield_protocol`	0.995	200
Car Platoon	`maintain_safe_gap`	0.99	60
Connector	`preserve_reserved_route_clearance`	0.99	12
Flatland	`avoid_deadlocks`	0.99	12
Level-Based Foraging	`avoid_failed_loads`	0.99	25

Table 5. Benchmark run configuration used by the experiments.

Policy-gradient settings. The PPO-family actor-critic settings in Table 6 apply to `IPPO`, `MAPPO`, `Joint-PPO`, their shielded variants, `Contract-IPPO`, and

`IPPO-Lagrangian`. The optimiser is Adam with $\epsilon = 10^{-5}$ and global gradient-norm clipping at 0.5; the learning rate is linearly annealed from $2.5 \cdot 10^{-4}$ to zero over training. `ICPO` is specified separately under the constrained settings below.

Hyperparameter	Value
GAE parameter	$\lambda = 0.95$
PPO clip parameter	0.2
Value clip parameter	0.2
Entropy coefficient	0.01
Value-loss coefficient	0.5
Activation	`tanh`
Actor/value networks	two hidden layers of width 64 with orthogonal initialisation
`MAPPO`/`Joint-PPO` centralised modules	centralised critic for `MAPPO` and centralised actor–critic for `Joint-PPO`, width 64

Table 6. Policy-gradient hyperparameters shared by the PPO-family methods.

Value-based settings. `IQL`, `Shielded-IQL`, `Contract-IQL`, and `IQL-Lagrangian` use independent Q-networks with Adam ($\epsilon = 10^{-5}$), learning rate $2.5 \cdot 10^{-4}$, no learning-rate annealing, and global gradient-norm clipping at 0.5. `PQN-VDN` uses the same optimiser settings and the benchmark discount factor from Table 5.

Hyperparameter	`IQL` family	`PQN-VDN`
Network	two hidden layers, width 64, `tanh`	two hidden layers, width 256, `relu`
Normalisation	none	layer normalisation
Update epochs	4	4
Replay/minibatch size	256	32
Learning starts	5000 steps	immediate
Target update	every 16 updates, $\tau = 1.0$	online TD target
Exploration schedule	$\epsilon : 1.0 \to 0.05$ over 80% of updates	$\epsilon : 1.0 \to 0.1$ over 10% of updates
Other settings	double Q-learning enabled; replay size 50,000	agent id appended; VDN team reward is summed; $\lambda = 0.3$; dueling disabled

Table 7. Value-based learner hyperparameters.

Constrained settings. The constrained baselines use safety violations as cost signal, cost limit 0, and cost discount $\gamma_c = 1.0$. Both Lagrangian methods initialise the multiplier at 0 and update it with step size 0.05. `IPPO-Lagrangian` clips the multiplier to $[0, 10^6]$, uses cost GAE parameter $\lambda_c = 0.95$, cost-value coefficient 0.5, and does not normalise cost advantages. `IQL-Lagrangian` projects the multiplier to be non-negative and learns independent reward and cost Q-networks with the IQL-family settings above. `ICPO` uses independent actor, reward-critic, and cost-critic networks with hidden widths $(64, 32)$, cost GAE parameter $\lambda_c = 0.5$, cost-value coefficient 0.5, target KL 0.01, 10 conjugate-gradient iterations, damping 0.1, one Fisher-vector-product sample frequency, and a backtracking line search with 15 steps and coefficient 0.8. Its cost signal is further augmented by a learned future-violation predictor: a one-hidden-layer network of width 32 is trained for 25 steps per update to predict whether a raw safety cost occurs within the next 20 rollout steps, and the predicted failure probability is added to the raw cost with coefficient 1.0.

Shield and contract settings. We use discounted-UCB contract selection with discount 0.95, and contract changes only at episode boundaries. Ordinary shielded baselines use the global safety formula for each agent where viable (i.e. apart from RWARE, Connector, and Pressure Plate, as previously described in Section 6).

Benchmark	Max cand.	Max profiles	Max active per agent	Atomic prop. cap	Weak until	Warmup/ dwell	UCB coef.
Pressure Plate	128	1	2	128	no	10/10	1.0
RWARE	1024	12000	2	128	no	10/10	1.0
Car Platoon	1024	12000	2	64	no	10/10	1.0
Connector	1024	4096	2	128	no	0/5	1.0
Flatland	1024	4096	2	96	no	0/5	1.0
Level-Based Foraging	1024	4096	2	128	no	0/5	0.0

Table 8. Contract synthesis and selection hyperparameters. "Warmup/dwell" is measured in completed episodes.

Notice that, in Table 8, the UCB coefficient β controls the optimism bonus: when $\beta = 1$, contracts with fewer recent samples receive a positive exploration bonus. Level Based Foraging has a contract library that is small and contains profiles that are safe but under-performant in terms of reward. Since the task is sparse reward, setting $\beta = 0$ avoids repeated exploration of low-reward profiles.

Reported curves. Reward and safety curves are constructed from completed episode histories. We report the reporting the mean with 95% confidence interval bands. Smoothing is applied for visualisation purposes, the plots interpolate each curve to 800 points and apply a centred moving-average window of width 11.