SLICR: STRATEGY LOGIC-GUIDED MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Specification-guided reinforcement learning (RL) translates formal specifications of desired behavior of agents into shaped rewards. This paper introduces SLICR, a significant paradigm shift that advances the state of the art to specification-guided RL in adversarial settings under partial observability. We employ *Strategy Logic* (SL), which allows explicit quantification over agents' strategies and uses the linear temporal logic (LTL) to specify the agents' objectives and constraints. This enables reasoning not only about how agents should act to satisfy a specification, but also about how they should respond to the strategies and behaviors of other agents. We evaluate SLICR on three popular benchmarks: Predator-Prey, Google Research Football, and our home-grown extension of SMAC-HARD, demonstrating its effectiveness, scalability, and ability to adapt to diverse adversarial strategies.

1 INTRODUCTION

Specification-guided reinforcement learning has recently gained significant attention by using desired agent behaviors expressed in formal languages to guide the learning process (Alur et al., 2026). Formal specification provides rigorous syntax and semantics for reasoning about sequences of events and states over time. Recently, approaches such as (Hsu et al., 2025; Rafieioskouei et al., 2026) have demonstrated that formal specifications not only provide precise objectives for multiple agents, but also improve learning performance. However, existing specification-guided multi-agent RL (MARL) approaches (ElSayed-Aly & Feng, 2022; Liu et al., 2024; León & Belardinelli, 2020) typically assume that other agents follow stationary strategies. More specifically, these approaches have two main limitations: (1) they learn a single strategy (i.e., policy) for a fixed task or game, without explicitly adapting to the strategies of other agents; and (2) adversarial formulations often focus on rational equilibrium behavior, such as Nash equilibria (Jothimurugan et al., 2022), without considering arbitrary or irrational opponent strategies. These limitations severely limit the ability of agents to respond to a changing collection of adversarial strategies.

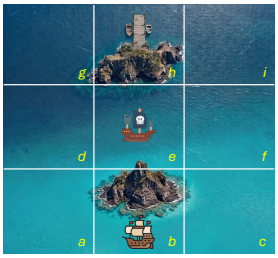


Figure 1: Voyage Scenario.

In this paper, we introduce SLICR (**S**trategy **L**ogic-**I**nformed **C**ontrol through **R**einforcement Learning), advancing the state of the art in specification-guided model-free MARL to adversarial settings under partial observability and non-stationary adversarial behavior. We employ the highly expressive *strategy logic* (SL) (Mogavero et al., 2014), which enables explicit quantification over multiple agents' strategies, along with the linear temporal logic (LTL) to specify the agents' objectives and constraints. This enables reasoning not only about how agents should act to satisfy a specification, but also about how they should respond to the strategies of other agents without making any assumptions about them. Consider the partially observable *Voyage* scenario shown in Fig. 1. The agents, ship () and pirate () observe only their own locations and can move between adjacent cells, except between b and e or h and e. SL formula $\exists y.\forall x(\mathsf{ship}, y)(\mathsf{pirate}, x).\big(\square\neg\mathsf{Inter}(\mathsf{pirate}) \wedge \Diamond\mathsf{h}(\mathsf{ship})\big)$ states that there exists a strategy for ship such that, for every strategy of pirate, ship reaches h without being intercepted. This formula cannot be satisfied because pirate can wait at d or f, depending on the route chosen by ship. By contrast, $\forall x.\exists y(\mathsf{pirate}, x)(\mathsf{ship}, y).\psi$ stipulates that for every strategy of pirate, there exists a strategy for ship to satisfy the objective. Thus, SL can express adversarial interactions by capturing dependencies between agents' strategies and specifying how agents should respond to one another.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

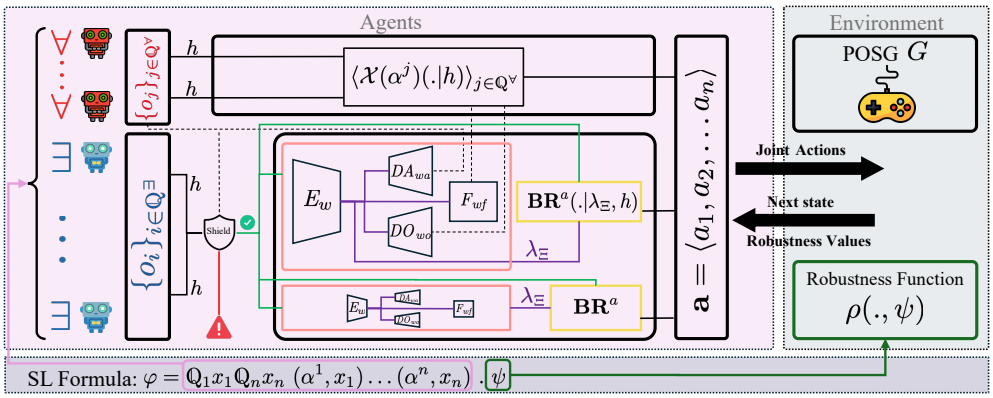


Figure 2: Overview of SLICR. Solid lines apply during training and evaluation; dashed lines apply only during training.

Contributions. We **(1)** formulate the strategy synthesis problem for a multi-agent system as a learning problem, where the goal is to maximize the probability of satisfying an SL specification under partial observability (Sec. 4); **(2)** solve this problem (see Fig. 2) by *(i)* transforming the temporal satisfaction problem into a robustness optimization problem, *(ii)* identifying irrational strategies assigned to universally quantified strategy variables and preventing them from misleading learning through a shielding mechanism, *(iii)* new opponent modeling based on a novel encoder-decoder technique, *(iv)* following the centralized training with decentralized execution (CTDE) paradigm to learn decentralized response functions under partial observation (Sec. 5), and **(3)** evaluate SLICR on three MARL benchmarks: Predator-Prey (Lowe et al., 2017), Google Research Football (GRF) (Kurach et al., 2020), and a home-grown extension of SMAC-HARD (Deng et al., 2024), using MARL algorithms to construct diverse opponent strategies. Our evaluation demonstrates the effectiveness of SLICR, with improvements of **up to 90**% over opponent-modeling baseline, and scalability to complex environments such as GRF and SMAC-HARD (Sec. 6).

We provide proofs, implementation and benchmark details, and additional results in App. A.

2 Related Work

Specification-Guided MARL. Specification-guided MARL has been extensively studied (ElSayed-Aly & Feng, 2022; Liu et al., 2024; Hsu et al., 2025; León & Belardinelli, 2020; Jothimurugan et al., 2022). Hammond et al. (2021) learn a joint policy that maximizes a weighted sum of the satisfaction probabilities of multiple LTL specifications. Bozkurt et al. (2024) instead consider a turn-based stochastic game and learn a controller policy that maximizes the probability of satisfying an LTL specification against a worst-case adversary. Unlike these settings, our framework uses SL under partial observability to express richer strategic relationships. In (Rafieioskouei et al., 2026), the authors use encoder-decoder models to reason about the dependencies represented by Skolem functions. Their encoder-decoder models reconstruct the states of universally quantified trace variables to guide cooperative MARL. In contrast, we use encoder-decoder models to reason over strategies of other agents for SL-guided adversarial MARL. We believe the work most closely related to SLICR is (Hahn et al., 2024), which studies RL in fully observable settings for strategic specifications expressed in probabilistic ATL (PATL), an extension of ATL (Alur et al., 2002). Unlike SLICR, their approach supports only a single outermost strategic quantifier without alternation.

Opponent Modeling. In SLICR, we reason about the strategies (behavior) of other agents, connecting our approach to opponent modeling. Rabinowitz et al. (2018) uses character and mental-state embeddings to represent agents' long-term and transient behavior, relational forward models to predict future actions or returns from global states (Tacchetti et al., 2019), and hierarchical sequential variational autoencoders to infer beliefs over other agents' strategies (Zintgraf et al., 2021). Policy representations can also be learned from observation-action trajectories, although representing unseen agents requires access to several of their previous trajectories (Grover et al., 2018). In (Yu et al., 2024), opponents' subgoals are inferred as embeddings of future states from historical trajectories under full observability and used to condition the learning agent's policy. Extending opponent modeling to partially observable settings, (Papoudakis et al., 2021a) reconstructs other agents' actions and observations from the controlled agent's local history, while (Odrowaz-Sypniewski et al., 2026) additionally models future states and returns and aggregates the resulting embeddings. In contrast, our learning is further guided by a specification and protected by a shield that excludes experiences associated with irrational strategies.

3 PRELIMINARIES

Let $\Delta(X) \triangleq \left\{p : X \to [0,1] \;\middle|\; \sum_{x \in X} p(x) = 1\right\}$ denote the probability simplex over a set X, and let $\mu(X)$ denote the uniform distribution, We abbreviate a tuple $\langle x_1, \ldots, x_n \rangle$ as $\langle x_i \rangle_{i \in \{1,\ldots,n\}}$.

3.1 PARTIALLY OBSERVABLE STOCHASTIC GAMES

We model the *environment* as an n-*agent partially observable stochastic game* (POSG) (Hansen et al., 2004). A POSG is defined as a tuple $G \triangleq \langle S, S_0, Ag, A, \mathbf{P}, O, \langle \Omega_{\alpha^i} \rangle_{i \in \{1,\ldots,n\}}, \mathsf{AP}, L \rangle$, where S is the set of environment *states*, and S_0 is the *initial-state* distribution. $Ag \triangleq \{\alpha^1, \ldots, \alpha^n\}$ is a finite set of n *agents*. We assume each agent α^i has the same *action space* A and we define $\mathcal{A} \triangleq \{\mathbf{a} = \langle a_1, \ldots, a_n \rangle \mid a_i \in A\}$ as the joint action space. The transition function $\mathbf{P} : S \times \mathcal{A} \to \Delta(S)$, where $\mathbf{P}(s' \mid s, \mathbf{a})$ denotes the probability of transitioning from $s \in S$ to $s' \in S$ under the joint action $\mathbf{a} \in \mathcal{A}$. In a POSG, agents do not have direct access to the full environment state. Instead, they receive a partial observation determined probabilistically by the state and the joint action. For simplicity, we assume each agent α^i has the same *observation space* O. For each agent α^i, the observation function $\Omega_{\alpha^i} : S \times \mathcal{A} \to \Delta(O)$, where $\Omega_{\alpha^i}(o_i \mid s, \mathbf{a})$ denotes the probability that α^i receives $o_i \in O$ in state $s \in S$ under the joint action $\mathbf{a} \in \mathcal{A}$. Lastly, AP denotes a finite set of *atomic propositions*, and $L : S \to 2^{\mathsf{AP}}$ is a *labeling* function.

Example. The POSG for the Voyage scenario in Fig. 1 is defined as follows. The environment is a 3×3 grid world with cells labeled $a, b, \ldots, i$. $Ag = \{\mathsf{ship}(\text{}), \mathsf{pirate}(\text{})\}$. Each state is of the form $(\mathtt{pos}_{\mathsf{ship}}, \mathtt{pos}_{\mathsf{pirate}})$, for the position of ship and pirate, respectively, where $\mathtt{pos}_{\mathsf{ship}}, \mathtt{pos}_{\mathsf{pirate}} \in \{a, b, \ldots, i\}$. Let (b, e) be the initial state. Both agents can move to adjacent cells using actions $A = \{U, D, R, L, S\}$, except between b and e or h and e. Each agent can only observe its own current position and not the position of the other agent (i.e., $O = \{a, b, \ldots, i\}$). For example, in state $s = (b, e)$, the observations of ship and pirate are b and e, respectively.

A *history* for an agent $\alpha^i \in Ag$ is an observation-action sequence $h_{\alpha^i} \triangleq o_{i_0} \xrightarrow{a_{i_0}} o_{i_1} \xrightarrow{a_{i_1}} \cdots$, where for all $j \geq 0$, $o_{i_j} \in O$ and $a_{i_j} \in A$. A *sub-history* $h_{\alpha^i[\ell:k]}$ is a finite segment $o_{i_\ell} \xrightarrow{a_{i_\ell}} \cdots \xrightarrow{a_{i_{k-1}}} o_{i_k}$, for $0 \leq \ell < k < |h_{\alpha^i}|$. We denote the set of all finite histories by $\mathcal{H}^*$. A *strategy* $\pi \colon \mathcal{H}^* \to \Delta(A)$, where $\pi(a \mid h)$ denotes the probability of choosing an action $a \in A$ after a finite history $h \in \mathcal{H}^*$. We denote the set of all strategies in G by Π. A *path* in a POSG is a sequence of environment states $\zeta \triangleq s_0 \xrightarrow{\mathbf{a}_0} s_1 \xrightarrow{\mathbf{a}_1} \cdots$, where $s_j \in S$ and $\mathbf{a}_j \in \mathcal{A}$ for all $j \geq 0$, and a *sub-path* $\zeta_{[\ell:k]}$ is a finite segment $s_\ell \xrightarrow{\mathbf{a}_\ell} \cdots \xrightarrow{\mathbf{a}_{k-1}} s_k$, for $0 \leq \ell < k < |\zeta|$. The *trace* t of a path ζ is a sequence of labels $\mathsf{Tr}(\zeta) \triangleq t(0)t(1)\cdots$, such that $t(j) = L(s_j)$ for all $j \geq 0$ (the trace of a sub-path, $\mathsf{Tr}(\zeta_{[\ell:k]})$, is defined similarly). Notice that, we only consider *valid* paths and sub-paths, where $\mathbf{P}(s' \mid s, \mathbf{a}) > 0$ for all transitions. We write $\mathcal{Z}^*$ to denote the set of all finite paths, and $\mathsf{Traces}(\mathcal{Z}^*)$ to denote the set of all finite traces. Furthermore, our setting is episodic, and each episode (a finite sequence of interactions from an initial state to termination) terminates after T steps, where $T \in \mathbb{Z}_{>0}$.

3.2 STRATEGIC LOGIC (SL)

Syntax. SL (Mogavero et al., 2014) is a temporal logic that allows explicit quantification over the strategies of agents. SL syntax is as follows:

$$\varphi ::= \exists x.\, \varphi \mid \forall x.\, \varphi \mid (\alpha, x).\varphi \mid \psi \qquad \psi ::= p \mid \neg\psi \mid \psi \vee \psi \mid \bigcirc\psi \mid \psi\, \mathcal{U}\, \psi \mid \psi \mathcal{T} \psi$$

where $p \in \mathsf{AP}$ is an atomic proposition, $\alpha \in Ag$ is an agent, and x is a *strategy variable*. We read $\exists x$ as "there exists a strategy assigned to x", and $\forall x$ as "for all strategies assigned to x". An *agent binding* (α, x) binds an agent α to the strategy associated with variable x. We write $Vars(\varphi)$ for the set of all strategy variables appearing in φ. The grammar of the inner sub-formula ψ yields a standard LTL formula with the Boolean connectives $\neg$ and $\vee$, as well as the temporal operators $\bigcirc$ ("next"), $\mathcal{U}$ ("until"), and $\mathcal{T}$ ("then"). Other Boolean and temporal operators are defined as syntactic sugar: $\mathsf{true} \triangleq p \vee \neg p$, $\mathsf{false} \triangleq \neg\mathsf{true}$, $\psi_1 \to \psi_2 \triangleq \neg\psi_1 \vee \psi_2$, $\Diamond\psi \triangleq \mathsf{true}\, \mathcal{U}\, \psi$, $\Box\psi \triangleq \neg\Diamond\neg\psi$, $\psi_1 \mathcal{U}_W \psi_2 \triangleq (\psi_1 \mathcal{U} \psi_2) \vee \Box\psi_1$, and $\psi_1 \mathcal{T}_W \psi_2 \triangleq (\psi_1 \mathcal{T} \psi_2) \vee (\Diamond\psi_1)$ where $\Diamond, \Box, \mathcal{U}_W$, and $\mathcal{T}_W$ denote 'eventually', 'always', 'weak until', and 'weak then', respectively. We say a formula φ is *agent-closed* if every agent $\alpha \in Ag$ occurring in the formula is bound to some variable $x \in Vars(\varphi)$, and is *variable-closed* if every variable $x \in Vars(\varphi)$ is quantified. A formula that is both variable-closed and agent-closed is called a *sentence*.

Semantics. Let $\varphi \triangleq \mathbb{Q}_1 x_1 \dots \mathbb{Q}_n x_n \, (\alpha^1, x_1) \dots (\alpha^n, x_n).\psi$ be an SL sentence, where each $\mathbb{Q}_i \in \{\forall, \exists\}$ and ψ is the inner LTL formula. The original semantics of SL in (Mogavero et al., 2014) is defined over deterministic concurrent games. Since we model RL environments by POSGs, we present the semantics of SL in two steps. The first step is concerned with the semantics of strategy quantifiers. The second step is related to the semantics of the inner LTL formula ψ for (sampled) paths from a POSG:

— An *assignment* is a partial mapping $\mathcal{X} \colon Ag \cup Vars(\varphi) \rightharpoonup \Pi$ that maps each agent and strategy variable in its domain to a strategy. Given an assignment $\mathcal{X}$, a strategy variable x_i, an agent α^i, and a strategy $\pi \in \Pi$, we write $\mathcal{X}[x_i \mapsto \pi]$ for the assignment that agrees with $\mathcal{X}$ except that x_i is mapped to strategy π. Similarly, $\mathcal{X}[\alpha^i \mapsto \pi]$ denotes the assignment that agrees with $\mathcal{X}$ except that α^i is mapped to π. We also write $\mathcal{X}[x_i, \alpha^i \mapsto \pi] \triangleq \mathcal{X}[x_i \mapsto \pi][\alpha^i \mapsto \pi]$. An assignment $\mathcal{X}$ is *complete* if it assigns a strategy to every agent and every variable. We use $\mathbb{X}_c$ to denote the set of all complete assignments and $\mathcal{X}_\emptyset$ for the empty assignment. The satisfaction relation is defined over a *model configuration* $(G, \Pi, \mathcal{X}, i)$ and φ, where $i \in \mathbb{Z}_{\geq 0}$ indicates the evaluation position.

$$
\begin{array}{lll}
(G, \Pi, \mathcal{X}, 0) & \models \exists x.\varphi & \text{iff} \quad \text{there exists } \pi \text{ in } \Pi \text{ st. } (G, \Pi, \mathcal{X}[x \mapsto \pi], 0) \models \varphi, \\
(G, \Pi, \mathcal{X}, 0) & \models \forall x.\varphi & \text{iff} \quad \text{for all } \pi \text{ in } \Pi \text{ st. } (G, \Pi, \mathcal{X}[x \mapsto \pi], 0) \models \varphi, \\
(G, \Pi, \mathcal{X}, 0) & \models (\alpha, x).\varphi & \text{iff} \quad (G, \Pi, \mathcal{X}[\alpha \mapsto \mathcal{X}(x)], 0) \models \varphi.
\end{array}
$$

— Now, for each complete assignment $\mathcal{X} \in \mathbb{X}_c$, let $G_{\mathcal{X}}$ denote the *distribution* over finite paths induced by the strategies assigned to the agents by $\mathcal{X}$. Observe that a complete assignment induces a *set* of paths in G, while the semantics of SL should be in terms of an individual induced path for each $\mathcal{X}$. Let $\zeta \sim G_{\mathcal{X}}$ denote a *sampled* path ζ from $G_{\mathcal{X}}$. Next, we define the semantics of the inner-LTL formula for $(\mathsf{Tr}(\zeta \sim G_{\mathcal{X}}), 0) \models \psi$:

$$
\begin{array}{lll}
(\mathsf{Tr}(\zeta), i) & \models p & \text{iff} \quad p \in \mathsf{Tr}(\zeta)(i), \\
(\mathsf{Tr}(\zeta), i) & \models \neg\psi & \text{iff} \quad (\mathsf{Tr}(\zeta), i) \not\models \psi, \\
(\mathsf{Tr}(\zeta), i) & \models \psi_1 \vee \psi_2 & \text{iff} \quad (\mathsf{Tr}(\zeta), i) \models \psi_1 \text{ or } (\mathsf{Tr}(\zeta), i) \models \psi_2, \\
(\mathsf{Tr}(\zeta), i) & \models \bigcirc\psi & \text{iff} \quad i+1 < |\mathsf{Tr}(\zeta)| \text{ and } (\mathsf{Tr}(\zeta), i+1) \models \psi, \\
(\mathsf{Tr}(\zeta), i) & \models \psi_1 \,\mathcal{U}\, \psi_2 & \text{iff} \quad \exists i' \geq i \text{ with } i' < |\mathsf{Tr}(\zeta)| \text{ such that } (\mathsf{Tr}(\zeta), i') \models \psi_2 \\
 & & \quad \text{and } \forall i'' \in [i, i'), \, (\mathsf{Tr}(\zeta), i'') \models \psi_1, \\
(\mathsf{Tr}(\zeta), i) & \models \psi_1 \mathcal{T} \psi_2 & \text{iff} \quad \exists i' \geq i \text{ with } i' < |\mathsf{Tr}(\zeta)| \text{ such that } (\mathsf{Tr}(\zeta), i') \models \psi_2 \\
 & & \quad \text{and } \exists i'' \in [i, i'), \, (\mathsf{Tr}(\zeta), i'') \models \psi_1.
\end{array}
$$

Roughly speaking, given the above semantics, our goal in this paper is to learn functions that map each existentially quantified strategy variable to a strategy in response to the choice of universally quantified strategy variables such that the probability of satisfaction of an SL specification in POSG is maximized. We will formally state this optimization problem in the next section.

4 Problem Statement

Let $\varphi = \mathbb{Q}_1 x_1 \dots \mathbb{Q}_n x_n \, (\alpha^1, x_1) \dots (\alpha^n, x_n). \, \psi$ be an SL sentence, where each $\mathbb{Q}_i \in \{\exists, \forall\}$ for $1 \leq i \leq n$. Intuitively, our objective is to identify a tuple of *best-response functions* that select a *strategy witness* for each $\mathbb{Q}_i = \exists$, such that the probability of induced paths satisfying ψ is maximized. We focus on the SL fragment with only one quantifier alternation (i.e., $\forall^*\exists^*$ and $\exists^*\forall^*$), as most adversarial RL problems can be expressed by this fragment.

We define $\mathbb{Q}^\exists \triangleq \{i \mid \mathbb{Q}_i = \exists\}$ and $\mathbb{Q}^\forall \triangleq \{j \mid \mathbb{Q}_j = \forall\}$ as the sets of indices of existentially and universally quantified strategy variables, respectively. Let $\mathbf{Pre} \triangleq \{j \in \mathbb{Q}^\forall \mid j < \min(\mathbb{Q}^\exists)\}$ denote the set of indices of universal quantifiers preceding the first existentially quantified variable. Given a complete strategy assignment $\mathcal{X}$, we define a *strategy context* $\Xi \triangleq \langle \mathcal{X}(x_j) \rangle_{j \in \mathbf{Pre}}$ as a tuple of strategies assigned to all the preceding universally quantified variables.

We now introduce our key objective. For each $i \in \mathbb{Q}^\exists$, we define a *response function* $\mathbf{RS}_{x_i} : \Pi^{|\mathbf{Pre}|} \to \Pi$, which maps the strategy context $\Xi \in \Pi^{|\mathbf{Pre}|}$ to a concrete strategy $\pi \in \Pi$. Whenever φ is of the form $\exists^*\forall^*$, then function $\mathbf{RS}_{x_i}$ becomes a constant function for all $i \in \mathbb{Q}^\exists$, because $\mathbf{Pre} = \{\}$ and $\Xi = \langle\rangle$. Finally, among all possible response functions, we denote the *best* response function by $\mathbf{BR}$; i.e., the one that gives *optimal* probability of satisfying φ.

Problem Statement

Given a POSG G and SL sentence $\varphi = \mathbb{Q}_1 x_1 \ldots \mathbb{Q}_n x_n\ (\alpha^1, x_1) \ldots (\alpha^n, x_n).\psi$ with at most one quantifier alternation our goal is to identify the tuple of best-response functions $\langle \mathbf{BR}_{x_i} \rangle_{i \in \mathbb{Q}^\exists}$ for all existentially quantified strategy variables $\langle x_i \rangle_{i \in \mathbb{Q}^\exists}$ such that:

$$\langle \mathbf{BR}_{x_i} \rangle_{i \in \mathbb{Q}^\exists} \in \left[\arg\max_{\langle \mathbf{RS}_{x_i} \rangle} \mathop{\mathbb{P}}_{\mathcal{X} \sim \mu(\mathbb{X}_c)} \Big[(G, \Pi, \mathcal{X}[x_i, \alpha^i \mapsto \mathbf{RS}_{x_i}(\Xi)], 0) \models \psi \Big] \right]_{i \in \mathbb{Q}^\exists} \tag{1}$$

where $\mathop{\mathbb{P}}_{\mathcal{X} \sim \mu(\mathbb{X}_c)}$ denotes the probability of satisfying ψ when $\mathcal{X}$ is sampled uniformly from $\mathbb{X}_c$.

Example. Consider the POSG in Fig. 1 and the SL formula φ_{exp} defined below.

$$\varphi_{\text{exp}} = \forall x. \exists y.\ (\mathsf{pirate}, x)(\mathsf{ship}, y)\ .\ \square(\mathtt{pos}_{\mathsf{ship}} \neq \mathtt{pos}_{\mathsf{pirate}})\ \wedge\ \Diamond(\mathtt{pos}_{\mathsf{ship}} = h)$$

Suppose that the strategy space is as follows (where $*$ denotes an arbitrary history prefix):

$$\Pi = \begin{Bmatrix} \pi_1 = \{(e,R),(*f,D),(*c,L),(*b,L)\}, & \pi_3 = \{(b,L),(*a,U),(*d,U),(*g,R),(e,S)\}, \\ \pi_2 = \{(e,L),(*d,S),(*a,R),(*b,R)\}, & \pi_4 = \{(b,R),(*c,U),(*f,U),(*i,L),(e,S)\} \end{Bmatrix}$$

Consider a response function $\mathbf{RS}^1_y = \{(\langle \pi_1 \rangle, \pi_3), (\langle \pi_2 \rangle, \pi_3), (\langle \pi_3 \rangle, \pi_4), (\langle \pi_4 \rangle, \pi_3)\}$. Essentially, $(\langle \pi_1 \rangle, \pi_3)$ means that when the pirate (i.e., x) is mapped to the π_1, then $\mathbf{RS}^1_y$ maps ship $\exists$-quantified variable y to π_3 (i.e., as the response strategy of ship against to pirate). The probability of satisfying the inner-LTL formula ψ_{exp} using $\mathbf{RS}^1_y$ is computed by considering the following four cases, each corresponding to a complete assignment: **Case 1.** For $(\langle \pi_1 \rangle, \pi_3)$, a path sampled from the distribution associated with this assignment is $(e,b) \xrightarrow{R,L} (f,a) \xrightarrow{D,U} (c,d) \xrightarrow{L,U} (b,g) \xrightarrow{L,R} (a,h)$, which satisfies the inner-LTL formula (i.e., $\models \psi_{\text{exp}}$). **Case 2.** For $(\langle \pi_2 \rangle, \pi_3)$, A path sampled from the distribution associated with this assignment is $(e,b) \xrightarrow{L,L} (d,a) \xrightarrow{S,U} (d,d) \xrightarrow{S,U} (d,g) \xrightarrow{S,R} (d,h)$, which does not satisfy the inner-LTL formula (i.e., $\not\models \psi_{\text{exp}}$, because ship is intercepted at cell d). **Case 3.** Similarly, for $(\langle \pi_3 \rangle, \pi_4)$, the induced path satisfies the inner-LTL formula. **Case 4.** Similarly, for $(\langle \pi_4 \rangle, \pi_3)$, the induced path also satisfies the inner-LTL formula. Hence, the probability of satisfying the SL formula φ_{exp} using $\mathbf{RS}^1_y$ is 0.75 .

However, suppose that in the next iteration we identify another response function $\mathbf{RS}^2_y = \{(\langle \pi_1 \rangle, \pi_3), (\langle \pi_2 \rangle, \pi_4), (\langle \pi_3 \rangle, \pi_4), (\langle \pi_4 \rangle, \pi_3)\}$. When the strategy context is $\langle \pi_2 \rangle$, $\mathbf{RS}^2_y$ returns π_4 for ship, and the induced path *satisfies* the inner-LTL formula. All remaining I/O in $\mathbf{RS}^2_y$ satisfy φ with the same analysis as $\mathbf{RS}^1_y$. Therefore, the probability of satisfying φ_{exp} using $\mathbf{RS}^2_y$ increases to 1 (as it remedies Case 2 which $\mathbf{RS}^1_y$ fails). Hence, $\mathbf{RS}^2_y$ is returned as the best-response function.

5 Algorithmic Details of SLICR

In this section, we describe the algorithmic details of SLICR (see Fig. 2). We first identify the challenges involved in solving the problem formulated in Sec. 4, and then present the components of SLICR designed to address each challenge.

Challenges and Overview of SLICR. First, recall that our objective in Sec. 4 is to find best-response functions that satisfy an SL specification under its binary semantics. We transform this temporal satisfaction problem into a robustness optimization problem using the quantitative semantics (Sec. 5.1). Second, universally quantified strategy variables may be assigned irrational strategies, resulting in noisy strategy contexts that can mislead the learning agents and degrade their performance. Therefore, SLICR identifies these strategies and prevents the corresponding experiences from affecting the learning process through a shielding mechanism (Sec. 5.2). Third, agents bound by existentially quantified strategy variables must respond to a wide range of strategies assigned to agents bound by preceding universally quantified variables. Thus, each response function takes as input a strategy context, which is a tuple of strategies. Since each strategy is itself a function, this tuple cannot be directly provided as input to a learnable response function. SLICR infers an embedding of the strategy context from the limited observations available to the agent (Sec. 5.3).

5.1 Quantitative Semantics of Inner-LTL Formulas

To obtain an optimization objective for SL satisfaction, we equip the inner-LTL formula with quantitative semantics (Li et al., 2017). Given a complete assignment $\mathcal{X} \in \mathbb{X}_c$ and those selected by $\mathbf{RS}$ for agents bound to existentially quantified variables, we sample a path with length T from the POSG G and quantitatively evaluate the inner-LTL formula ψ over this path. Let $\mathbb{R}$ denote the set of real numbers, Ψ the set of all LTL formulas, and $f : 2^{\mathsf{AP}} \to \mathbb{R}$ a *valuation function* that assigns a real value to each set of atomic propositions. For a state $s \in S$ in G, quantitative semantics are defined by predicates of the form $f\big(L(s)\big)$ (see App. C for full semantics). The robustness function $\rho : \mathsf{Traces}(\mathcal{Z}^*) \times \Psi \to \mathbb{R}$ assigns a real-valued *robustness value* to each finite trace and LTL formula, bounded below by ρ_{min} and above by ρ_{max}. Formally, given an inner-LTL formula ψ and a POSG G, we reformulate the problem statement introduced in Sec. 4 as follows:

$$\langle \mathbf{BR}_{x_i} \rangle_{i \in \mathbb{Q}^\exists} \in \left[\underset{\langle \mathbf{RS}_{x_i} \rangle}{\arg\max} \underset{\mathcal{X} \sim \mu(\mathbb{X}_c)}{\mathbb{P}} \Big[\rho\big(\mathsf{Tr}\big((\zeta_{[0:T]} \sim G_{\mathcal{X}[x_i, \alpha^i \mapsto \mathbf{RS}_{x_i}(\Xi)]})\big), \psi\big) \xrightarrow{\star} \rho_{max} \Big] \right]_{i \in \mathbb{Q}^\exists} \tag{2}$$

where $\xrightarrow{\star}$ denotes convergence. That is, the tuple of best-response functions $\langle \mathbf{BR}_{x_i} \rangle_{i \in \mathbb{Q}^\exists}$ maximizes the probability that the robustness of a path sampled from G under the complete assignment $\mathcal{X}$ converges to ρ_{max}, thereby optimizing the satisfiability of the inner-LTL formula ψ.

Lemma 5.1 *Under the order-preservation assumption, any tuple of best-response functions $\langle \mathbf{BR}_{x_i} \rangle_{i \in \mathbb{Q}^\exists}$ that maximizes Eq. (2) also solves the optimization problem stated in Sec. 4.*

5.2 Shielding from Noisy Strategies

Since universally quantified variables may be assigned any strategy from Π, the resulting strategy contexts may contain *irrational strategies* that hinder the learning of $\langle \mathbf{RS}_{x_i} \rangle_{i \in \mathbb{Q}^\exists}$. We define an irrational strategy as an observation-independent strategy that selects actions uniformly at random. To mitigate this issue, before the main learning phase, we train a shield $\mathbb{O} : \mathcal{H}^* \to \{ ⚠, ✅ \}$ that classifies the strategy context from the history of an agent bound to an existentially quantified variable as either rational (✅) or irrational (⚠). Though not a function, we use the inverse of $\mathbb{O}$ to denote the set of observation histories associated with rational strategies; i.e., $\mathbb{O}^{-1}(✅)$. Formally, $\mathbb{O}$ is trained on m-step histories $h_{\alpha^i_{[0:m]}} \in \mathcal{H}^*$, for each $i \in \mathbb{Q}^\exists$, collected under rational and irrational strategies assigned to universally quantified variables. Experiences labeled ⚠ are excluded from training, while those labeled ✅ are included. During the main learning phase, the shield is used only for inference. Further details on shield are provided in Appendix D.1.

5.3 Learning Best-Response Functions

We now address the optimization problem in Eq. (2). Our learning framework follows the *Centralized Training with Decentralized Execution (CTDE)* paradigm (Oliehoek et al., 2016). Conceptually, during centralized training, agents bound to existentially quantified variables have access to observations and actions of agents bound to preceding universally quantified variables. During execution, each agent has access only to its local information.

Robustness Values as Reward Signals. To solve Eq. (2), we derive reward signals that guide the learning of response functions. Given a $\mathcal{X} \in \mathbb{X}_c$, the reward for each α^i, where $i \in \mathbb{Q}^\exists$, at step k is computed as $\rho\big(\mathsf{Tr}\big((\zeta \sim G_{\mathcal{X}})_{[0:k+1]}\big), \psi\big)$. Given a discount factor $\gamma \in [0, 1)$, we learn best-response functions that maximize the expected cumulative discounted robustness:

$$\langle \mathbf{BR}_{x_i} \rangle_{i \in \mathbb{Q}^\exists} \in \left[\underset{\langle \mathbf{RS}_{x_i} \rangle}{\arg\max} \underset{\substack{\mathcal{X} \sim \mu(\mathbb{X}_c) \\ \zeta \sim G_{\mathcal{X}[x_i, \alpha^i \mapsto \mathbf{RS}_{x_i}(\Xi)]}}}{\mathbb{E}} \left[\sum_{k=0}^{T-1} \gamma^k \rho\big(\mathsf{Tr}\big(\zeta_{[0:k+1]}\big), \psi\big) \right] \right]_{i \in \mathbb{Q}^\exists} \tag{3}$$

Lemma 5.2 *If the objective in Eq. (3) strictly preserves the ordering induced by Eq. (2) over well-typed response-function tuples, then every maximizer of Eq. (3) also maximizes Eq. (2).*

Note that, in this setting, reward signals are derived from the inner-LTL formula, so all existentially quantified agents receive the same reward. However, more generally, a POSG may assign distinct rewards to individual agents. We can capture this setting by decomposing ψ into agent-specific subformulas, $\psi = \bigwedge_{i \in \mathbb{Q}^\exists} \psi_i$, where each ψ_i represents an objective or constraint associated with an agent α^i. The reward for agent α^i at step k is then individually computed as $\rho\big(\mathsf{Tr}\big((\zeta \sim G_{\mathcal{X}})_{[0:k+1]}\big), \psi_i\big)$. For simplicity, we present the remainder of our algorithm using a single full inner-LTL formula ψ.

Representing Strategy Context. As shown in Eq. (3), for each $i \in \mathbb{Q}^{\exists}$, the response function $\mathbf{RS}_{x_i}$ takes Ξ as an input. This raises two difficulties. First, $\Xi \in \Pi^{|\mathbf{Pre}|}$ is a $|\mathbf{Pre}|$-tuple of strategies, each of which is itself a function. Therefore, providing this tuple as input to a learnable response function requires a suitable finite-dimensional representation. Second, due to the partial observability of the POSG, each agent α^i, where $i \in \mathbb{Q}^{\exists}$, must infer this representation from its local history $h_{\alpha^i} \in \mathcal{H}^*$. We associate the strategy-context space $\Pi^{|\mathbf{Pre}|}$ with a latent context space $\Lambda \subseteq \mathbb{R}^d$, where $d \in \mathbb{Z}_{>0}$ denotes the dimension of the latent context embedding. Let $\lambda \in \Lambda$ denote a latent embedding. We call λ an *information-preserving latent context embedding* of Ξ (denoted λ_Ξ), if it represents Ξ without information loss, meaning that Ξ can be uniquely recovered from λ.

We are inspired by the agent-modeling approach proposed in (Papoudakis et al., 2021a), which uses an encoder-decoder architecture to learn embeddings of other agents from the controlled agent's local information under partial observability. We define *encoder* $E_w : \mathcal{H}^* \to \Lambda$, parameterized by w, that maps the history $h_{\alpha^i} \in \mathcal{H}^*$ of any agent α^i, where $i \in \mathbb{Q}^{\exists}$, to a latent embedding $\lambda \in \Lambda$. Next, we define two *decoders*: an *action-reconstruction* decoder $DA_{wa} : \Lambda \to \Delta(A)^{|\mathbf{Pre}|}$, and an *observation-reconstruction* $DO_{wo} : \Lambda \to O^{|\mathbf{Pre}|}$, parameterized by wa and wo, respectively. These decoders map a latent embedding $\lambda \in \Lambda$ to the reconstructed actions and observations of agents bound to the preceding universally quantified variables, respectively. In addition, we use the latent embedding to predict their actions over the next L-steps. This prediction encourages the latent embedding to capture information about their future behavior. Formally, we define a *future-action predictor* $F_{wf} : \Lambda \to \Delta(A)^{|\mathbf{Pre}| \times L}$, parameterized by wf, where $L \in \mathbb{Z}_{>0}$ denotes the prediction window. Further details on computing the latent embedding are provided in App. D.2.

Learning Response Functions. Recall that, as defined in Secs. 3 and 4, each complete assignment $\mathcal{X} \in \mathbb{X}_c$ induces a distribution $G_{\mathcal{X}}$ over finite paths, from which a path $\zeta \sim G_{\mathcal{X}}$ is sampled. For each $i \in \mathbb{Q}^{\exists}$, we seek a $\mathbf{RS}_{x_i}$ that maximizes the objective in Eq. (3). Given a Ξ, the $\mathbf{RS}_{x_i}(\Xi)$ returns a strategy for variable x_i and agent α^i, where each strategy is a mapping $\pi : \mathcal{H}^* \to \Delta(A)$. Hence, assigning a strategy π alone does not determine its action distribution during sampling, as π depends on the history observed by α^i. As described earlier, we use the latent embedding λ_Ξ to represent the strategy context Ξ. Hence, to explicitly represent how the assigned strategies generate actions during learning, we define its equivalent action-level response function $\mathbf{RS}^a_{x_i} : \Lambda \times \mathcal{H}^* \to \Delta(A)$, which maps a latent embedding and a history to a distribution over the actions.

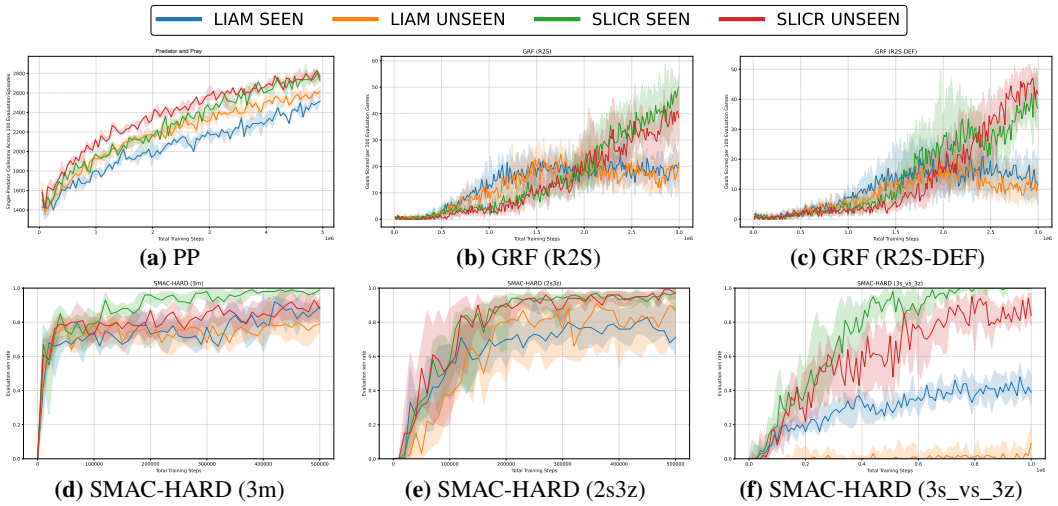
Now, let $\langle a_i \rangle_{i \in I}$ and $\langle a_j \rangle_{j \in J}$ be two action tuples, where $I \cup J = \{1, \ldots, n\}$ and $I \cap J = \emptyset$. To combine these tuples while preserving the order according to their indices, we use the operator $\oplus$, which returns a single tuple ordered by index. Given a complete assignment $\mathcal{X} \in \mathbb{X}_c$ and a tuple of action-level response functions $\langle \mathbf{RS}^a_{x_i} \rangle_{i \in \mathbb{Q}^{\exists}}$, a joint action $\mathbf{a}$ at step k is defined as:

$$\mathbf{a}_k = \langle a_{i_k} \sim \mathbf{RS}^a_{x_i}(.|E_w(h_{\alpha^i_{[0:k]}})_\Xi \,|_{\mathsf{O}^{-1}(\textcolor{green}{\checkmark})}, h_{\alpha^i_{[0:k]}}) \,|_{\mathsf{O}^{-1}(\textcolor{green}{\checkmark})} \rangle_{i \in \mathbb{Q}^{\exists}} \oplus \langle a_{j_k} \sim \mathcal{X}(x_j)(.|h_{\alpha^j_{[0:k]}}) \rangle_{j \in \mathbb{Q}^{\forall}}$$

Notice that we are restricting the inputs of E_w and $\mathbf{RS}^a_{x_i}$ by observation histories that yield a rational strategy by the O during training (and not execution). For simplicity, we write it as $\mathbf{a}_k \sim \langle \mathbf{RS}^a_{x_i}(.|\lambda_\Xi, h_{\alpha^i_{[0:k]}}) \rangle_{i \in \mathbb{Q}^{\exists}} \oplus \langle \mathcal{X}(x_j)(.|h_{\alpha^j_{[0:k]}}) \rangle_{j \in \mathbb{Q}^{\forall}}$, where $\lambda_\Xi = E_w(h_{\alpha^i_{[0:k]}})_\Xi \,|_{\mathsf{O}^{-1}(\textcolor{green}{\checkmark})}$. Accordingly, the next state is computed as $s_{k+1} \sim \mathbf{P}(.|s_k, \mathbf{a}_k)$, where $s_0 \sim S_0$. We now update the learning objective introduced in Eq. (3) to identify the action-level best-response functions, formally:

$$\langle \mathbf{BR}^a_{x_i} \rangle_{i \in \mathbb{Q}^{\exists}} \in \left[\underset{\substack{\langle \mathbf{RS}^a_{x_i} \rangle \\ \mathbf{a}_k \sim \langle \mathbf{RS}^a_{x_i}(.|\lambda_\Xi, h_{\alpha^i_{[0:k]}}) \rangle \oplus \langle \mathcal{X}(x_j)(.|h_{\alpha^j_{[0:k]}}) \rangle_{j \in \mathbb{Q}^{\forall}} \\ s_{k+1} \sim \mathbf{P}(.|s_k, \mathbf{a}_k)}}{\arg\max} \; \underset{\substack{\mathcal{X} \sim \mu(\mathbb{X}_c) \\ s_0 \sim S_0}}{\mathbb{E}} \left[\sum_{k=0}^{T-1} \gamma^k \rho\big(\mathsf{Tr}\big(s_0 \xrightarrow{\mathbf{a}_0} \cdots \xrightarrow{\mathbf{a}_k} s_{k+1}\big), \psi\big) \right] \right]_{i \in \mathbb{Q}^{\exists}}$$

Theorem 5.3 *Given a POSG G and an SL sentence φ, suppose that λ_Ξ is an information-preserving latent embedding of each Ξ and that the action-level response functions can represent every well-typed response function. Then any $\langle \mathbf{BR}^a_{x_i} \rangle_{i \in \mathbb{Q}^{\exists}}$ that maximizes the expected cumulative discounted robustness induces a $\langle \mathbf{BR}_{x_i} \rangle_{i \in \mathbb{Q}^{\exists}}$ that maximizes the objective in Eq. (3). By Lemma 5.2, this tuple also optimizes Eq. (2), and hence, by Lemma 5.1, optimizes the probability of satisfying φ in G as defined in Sec. 4.*



(a) PP **(b)** GRF (R2S) **(c)** GRF (R2S-DEF)

(d) SMAC-HARD (3m) **(e)** SMAC-HARD (2s3z) **(f)** SMAC-HARD (3s_vs_3z)

Figure 3: Learning curves for SLICR and LIAM, showing the median and 25th–75th percentiles.

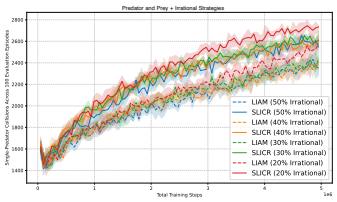


Figure 4: Irrational strategies(PP).

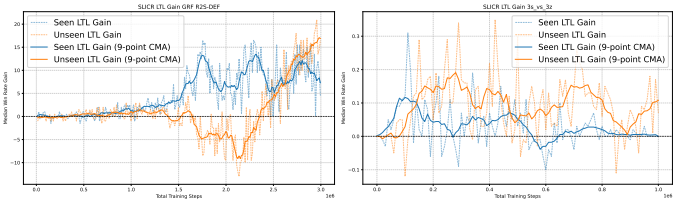


Figure 5: Effect of inner-LTL on R2S-DEF (left), `3s_vs_3z` (right).

6 Experiments and Results

Implementation Pipeline. We first construct Π using off-the-shelf MARL algorithms, including IPPO and MAPPO (Yu et al., 2022), VDN (Sunehag et al., 2017),QMIX (Rashid et al., 2018), and QTRAN (Son et al., 2019), together with irrational strategies that select actions uniformly at random. Let Π^{Train} and Π^{Test} denote the strategy sets used during training and evaluation, respectively. We evaluate performance in two settings: (1) `Seen`, where $\Pi^{\text{Train}} = \Pi^{\text{Test}}$, and (2) `Unseen`, where $\Pi^{\text{Train}} \cap \Pi^{\text{Test}} = \emptyset$. During training phase, we use the quantitative semantics (Sec. 5.1) to compute robustness-based rewards from the SL formula. For each $i \in \mathbb{Q}^{\exists}$, we then train the response function using h_{α^i} and λ with either A2C or PPO (Schulman et al., 2017), while the pretrained $\bigcirc$ excludes experiences associated with irrational strategies. We report results over ten independent runs with different random seeds. Further experimental and implementation details are provided in App. D.

Benchmarks. We evaluate SLICR on three benchmarks. (1) **Predator and Prey (PP)** (Lowe et al., 2017). We use the implementation provided by (Papoudakis et al., 2021a). PP consists of one existentially quantified prey that must be captured by exactly one of three universally quantified predators. We adopt the strategy sets from (Papoudakis et al., 2021b) and develop novel, diverse IPPO, MAPPO, and irrational predator strategies. (2) **Google Research Football (GRF)** (Kurach et al., 2020). We evaluate two scenarios. First, in "Run to Score" (R2S), a player (bound to an existentially quantified variable) starts at the midfield and must score against a goalkeeper while five opposing players (bound to universally quantified variables) are chasing from behind. Second, in "Run to Score DEF" (R2S-DEF), inspired by (Odrowaz-Sypniewski et al., 2026), two of the opposing players are positioned between the existentially quantified player and the goal, while the remaining three are chasing from behind. To construct a diverse strategy set, we train IPPO and MAPPO opponent teams with different tactical behaviors. (3) **SMAC-HARD** (Deng et al., 2024). We evaluated our extension with pretrained MARL opponent populations on three maps: `3m`, `2s3z`, and `3s_vs_3z`. The objective of the existentially quantified agents is to eliminate all universally quantified agents. To construct a diverse strategies set, we use VDN, QMIX, QTRAN, and built-in AI opponents. Further details on the environment settings, SL formulas, and opponent strategies are provided in App. E.

Experimental Organization. We assess SLICR along four dimensions: (1) Comparing SLICR with LIAM (Papoudakis et al., 2021a), an opponent-modeling baseline for partially observable environments. (2) Comparing the $\forall^*\exists^*$ and $\exists^*\forall^*$ fragments of the target SL formulas in terms of effectiveness and parameter efficiency. (3) Comparing SLICR and LIAM in the presence of irrational strategies. (4) Investigating how the inner-LTL formula guides and improves learning.

Table 1: Median performance of SLICR across SL fragments using the same inner-LTL specification and training budget. `S`, `U`, and `WR` denote `Seen`, `Unseen`, and win rate, respectively.

SLICR	**PP** (# Single-Collision)			**R2S** (# Goal)			**R2S-DEF** (# Goal)			`3m` (WR%)			`2s3z` (WR%)			`3s_vs_3z` (WR%)		
	#Prms	S	U	#Prms	S	U	#Prms	S	U	#Prms	S	U	#Prms	S	U	#Prms	S	U
($\forall^*\exists^*$)	250K	2765	2749	630K	47	37	630K	40	44	636K	99	89	748K	98	98	645K	100	86
($\exists^*\forall^*$)	111K	2239	2369	334K	12	14	334K	24	12	445K	96	87	496K	98	92	451K	98	87

(1) SLICR vs. LIAM. Since SLICR represents strategy context as a latent embedding, we compare it with LIAM, an opponent-modeling method for partially observable settings. Both methods use A2C on PP and PPO on GRF and SMAC-HARD. In PP (see Fig. 3a), SLICR achieves up to $\uparrow$400 and $\uparrow$200 more single-predator collisions than LIAM over 100 evaluation episodes in the `Seen` and `Unseen` settings, respectively. On the more complex GRF benchmark, SLICR significantly outperforms LIAM. In R2S (see Fig. 3b), SLICR surpasses LIAM after 2M training steps and ultimately scores $\uparrow$30 and $\uparrow$20 more goals over 100 evaluation episodes in the `Seen` and `Unseen` settings, respectively. Similarly, in the harder R2S-DEF scenario (see Fig. 3c), SLICR outperforms LIAM, scoring up to $\uparrow$30 and $\uparrow$35 more goals in the `Seen` and `Unseen` settings, respectively.

The superiority of SLICR extends to the SMAC-HARD as well. On `3m` (see Fig. 3d), SLICR achieves up to $\uparrow$25% and $\uparrow$15% higher win rates than LIAM in the `Seen` and `Unseen` settings, respectively. Notably, in the `Seen` setting, SLICR reaches $\approx$100% win rate after 400K training steps, whereas LIAM does not within the 500K-step training budget. On `2s3z` (see Fig. 3e), SLICR reaches $\approx$100% win rate after 400K steps, outperforming LIAM by up to $\uparrow$30% and $\uparrow$20% in the `Seen` and `Unseen` settings, respectively. The performance of SLICR is particularly substantial on `3s_vs_3z` (see Fig. 3f). In the `Seen` setting, SLICR reaches a 100% win rate after $\approx$700K steps, while LIAM reaches only $\approx$40% within the 1M-step training, yielding a gain of $\uparrow$60%. In the `Unseen` setting, SLICR achieves $\approx$90% win rate, whereas LIAM rarely wins, yielding a gain up to $\uparrow$90%. Further results are in App. F.1.

(2) SLICR $\forall^*\exists^*$ vs. $\exists^*\forall^*$. These two SL fragments induce different learning pipelines. Under $\forall^*\exists^*$, each response function is conditioned on the agent's history and latent embedding (i.e. strategy context) of the preceding universal agents' strategies, whereas under $\exists^*\forall^*$, it uses only the local information. Consequently, the former is more parameter-intensive, as quantified across benchmarks in Table 1. This is expected because solving the $\forall^*\exists^*$ problem is generally harder; however, it yields adaptive strategies. However, SLICR ($\forall^*\exists^*$) outperforms SLICR ($\exists^*\forall^*$) by up to $\uparrow$526 and $\uparrow$380 in the `Seen` and `Unseen` settings, respectively. This improvement is particularly substantial in the GRF scenarios. In R2S, SLICR ($\forall^*\exists^*$) achieves approximately $\times$4 and $\times$3 scores in the `Seen` and `Unseen` settings, respectively. Similarly, in R2S-DEF, it achieves approximately $\times$1.5 and $\times$4 scores, respectively. In contrast, in the SMAC-HARD scenarios, ($\forall^*\exists^*$) and ($\exists^*\forall^*$) perform similarly, although ($\forall^*\exists^*$) performs slightly better in 4/6 cases. This suggests that the heavier ($\forall^*\exists^*$) pipeline is not necessary for all cases, and ($\exists^*\forall^*$) can be sufficient. Additional analysis is provided in App. F.2.

(3) SLICR vs. LIAM Under Irrational Strategies Fig. 4 shows how the shield $\bigcirc$ protects SLICR when Π^{Train} contains irrational strategies. Across settings containing 20%, 30%, 40%, or 50% irrational strategies, SLICR (solid lines) achieves up to $\uparrow$300 more single-predator collisions than LIAM (corresponding dashed lines). A shield ablation study is provided in App. F.3.

(4) Inner-LTL vs. Shaped Reward. Fig. 5 compares inner-LTL guidance with shaped rewards. On GRF R2S-DEF (left), guidance from $\varphi_{\text{R2S-DEF}}$ enables SLICR to achieve up to $\uparrow$15 and $\uparrow$20 more goals in the `Seen` (blue line) and `Unseen` (orange line) settings, respectively. On `3s_vs_3z` (right), guidance from φ_{SMAC} improves the win rate by up to $\uparrow$30% in `Seen` and $\uparrow$35% in `Unseen`. In `Seen`, both variants eventually approach a 100% win rate, resulting in similar behavior at the end of training, while `Unseen` gain persists throughout. Additional ablations are provided in App. F.4.

7 Conclusion

SLICR is a specification-guided MARL framework for partially observable adversarial settings. By employing Strategic Logic, SLICR explicitly reasons about agents' strategies and their responses to adversaries. Our evaluation on Predator-Prey, Google Research Football, and SMAC-HARD demonstrates that SLICR is effective, scalable, and adaptable across adversarial strategies, improving opponent-modeling baselines by up to 90%.

Limitations. SLICR is limited to SL fragments with at most one quantifier alternation, which we believe are sufficient to express many adversarial RL problems. Also, its theoretical guarantees rely on information-preserving latent embedding, which may be hard to achieve under partial observability.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods and assist in the writing of proofs. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, interpret results, generate synthetic data sets, assist with translation, or lean and reformat dataset, support qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for brainstorming and to propose a title or keywords for a research paper. We reviewed all AI-assisted work. We used LLMs to assist with parts of the experimental implementation, and all generated code was tested and verified for correctness. We also used LLMs to review our proofs, provide feedback on their clarity, and edit and polish the manuscript. We take full responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

We provide the algorithmic details in Sec. 5, complemented by the experimental setup in Sec. 6. Complete implementation details and hyperparameters are provided in App. D, while benchmark configurations are described in App. E.

REFERENCES

Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, September 2002. ISSN 0004-5411. doi: 10.1145/585265.585270. URL https://doi.org/10.1145/585265.585270.

Rajeev Alur, Suguman Bansal, Osbert Bastani, and Kishor Jothimurugan. Specification-guided reinforcement learning. *Commun. ACM*, 69(2):80–87, January 2026. ISSN 0001-0782. doi: 10.1145/3744706. URL https://doi.org/10.1145/3744706.

Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. Learning optimal strategies for temporal tasks in stochastic games. *IEEE Transactions on Automatic Control*, 69 (11):7387–7402, 2024. doi: 10.1109/TAC.2024.3390848.

Yue Deng, Yan Yu, Weiyu Ma, Zirui Wang, Wenhui Zhu, Jian Zhao, and Yin Zhang. Smac-hard: Enabling mixed opponent strategy script and self-play on smac, 2024. URL https://arxiv.org/abs/2412.17707.

Ingy ElSayed-Aly and Lu Feng. Logic-based reward shaping for multi-agent reinforcement learning, 2022. URL https://arxiv.org/abs/2206.08881.

Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1802–1811. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/grover18a.html.

Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Multi-agent reinforcement learning for alternating-time logic. In Ulle Endriss, Francisco S. Melo, Kerstin Bach, Alberto José Bugarín Diz, Jose Maria Alonso-Moral, Senén Barro, and Fredrik Heintz (eds.), *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pp. 1680–1687. IOS Press, 2024. doi: 10.3233/FAIA240676. URL https://doi.org/10.3233/FAIA240676.

Lewis Hammond, Alessandro Abate, Julian Gutierrez, and Michael Wooldridge. Multi-agent reinforcement learning with temporal logic specifications, 2021. URL https://arxiv.org/abs/2102.00582.

Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In Deborah L. McGuinness and George Ferguson (eds.), *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pp. 709–715. AAAI Press / The MIT Press, 2004. URL http://www.aaai.org/Library/AAAI/2004/aaai04-112.php.

Tzu-Han Hsu, Arshia Rafieioskouei, and Borzoo Bonakdarpour. Hyprl: Reinforcement learning of control policies for hyperproperties. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 52298–52329. Curran Associates, Inc., 2025. doi: 10.52202/085713-1743. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/4b58601df13de0fab64698b6d01f13c3-Paper-Conference.pdf.

Kishor Jothimurugan, Suguman Bansal, Osbert Bastani, and Rajeev Alur. Specification-guided learning of nash equilibria with high social welfare. In *International Conference on Computer Aided Verification*, pp. 343–363. Springer, 2022.

Karol Kurach, Anton Raichuk, Piotr Stanczyk, Michal Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Google research football: A novel reinforcement learning environment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 4501–4510. AAAI Press, 2020. doi: 10.1609/AAAI.V34I04.5878. URL https://doi.org/10.1609/aaai.v34i04.5878.

Borja G. León and Francesco Belardinelli. Extended markov games to learn multiple tasks in multi-agent reinforcement learning, 2020. URL https://arxiv.org/abs/2002.06000.

Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3834–3839. IEEE, 2017.

Chanjuan Liu, Jinmiao Cong, Bingcai Chen, Yaochu Jin, and Enqiang Zhu. Guiding multi-agent multi-task reinforcement learning by a hierarchical framework with logical reward shaping, 2024. URL https://arxiv.org/abs/2411.01184.

Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf.

Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Logic*, 15(4), November 2014. ISSN 1529-3785. doi: 10.1145/2631917. URL https://doi.org/10.1145/2631917.

Mateusz Odrowaz-Sypniewski, Jasmine Bayrooti, Ajay Shankar, and Amanda Prorok. Generalized intention modeling in multi-agent reinforcement learning, 2026. URL https://arxiv.org/abs/2605.31318.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Georgios Papoudakis, Filippos Christianos, and Stefano Albrecht. Agent modelling under partial observability for deep reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 19210–19222. Curran Associates, Inc., 2021a. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/a03caec56cd82478bf197475b48c05f9-Paper.pdf.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021b. URL https://openreview.net/forum?id=cIrPX-Sn5n.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, S. M. Ali Eslami, and Matthew Botvinick. Machine theory of mind. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4218–4227. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/rabinowitz18a.html.

Arshia Rafieioskouei, Tzu-Han Hsu, Matthew Lucas, and Borzoo Bonakdarpour. HyPOLE: Hyperproperty-guided multi-agent reinforcement learning under partial observation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=EVoPYB6ss3.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4295–4304. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/rashid18a.html.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pp. 2186–2188, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450363099.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5887–5896. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/son19a.html.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning, 2017. URL https://arxiv.org/abs/1706.05296.

Andrea Tacchetti, H. Francis Song, Pedro A. M. Mediano, Vinicius Zambaldi, János Kramár, Neil C. Rabinowitz, Thore Graepel, Matthew Botvinick, and Peter W. Battaglia. Relational forward models for multi-agent learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJlEojAqFm.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and YI WU. The surprising effectiveness of ppo in cooperative multi-agent games. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24611–24624. Curran Associates, Inc., 2022. doi: 10.52202/068431-1787. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf.

Xiaopeng Yu, Jiechuan Jiang, and Zongqing Lu. Opponent modeling based on subgoal inference. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 60531–60555. Curran Associates, Inc., 2024. doi: 10.52202/079017-1936. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/6fb9ea5197c0b8ece8a64220fb82cdfe-Paper-Conference.pdf.

Luisa M. Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann. Deep interactive bayesian reinforcement learning via meta-learning. In Frank Dignum, Alessio Lomuscio, Ulle Endriss, and Ann Nowé (eds.), *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, pp. 1712–1714. ACM, 2021. doi: 10.5555/3463952.3464210. URL https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p1712.pdf.

A Appendix Outline

- Appendix B: Proofs
  - B.1: Proof of Lemma 5.1
  - B.2: Proof of Lemma 5.2
  - B.3: Proof of Theorem 5.3
- Appendix C: Full Quantitative Semantics of Inner-LTL Formulas
- Appendix D: Experimental Setup and Implementation Details
  - D.1: Details on the Shield
  - D.2: Encoder, Decoders, and Future-Action Prediction
  - D.3: Training the Action-Level Response Functions
- Appendix E: Benchmark Details
  - E.1: Predator-Prey (PP)
  - E.2: Google Research Football (GRF)
  - E.3: SMAC-HARD
- Appendix F: Additional Experimental Results
  - F.1: SLICR vs. LIAM
  - F.2: SLICR $\forall^*\exists^*$ vs. $\exists^*\forall^*$
  - F.3: Shield Ablation Study
  - F.4: Inner-LTL vs. Shaped Reward
- Appendix G: Computational Resources

B Proofs

B.1 Proof of Lemma 5.1

Under the order-preservation assumption, any tuple of best-response functions $\langle \mathbf{BR}_{x_i} \rangle_{i \in \mathbb{Q}^\exists}$ that maximizes Eq. (2) also solves the optimization problem stated in Sec. 4.

Proof 1 *We assume that the convergence objective is order-preserving with respect to satisfaction, meaning that a tuple of response functions with a higher probability of convergence to the maximum robustness value cannot have a lower probability of satisfying the specification. We now begin the proof by let, $\mathfrak{B} = \langle \mathbf{BR}_{x_i} \rangle_{i \in \mathbb{Q}^\exists}$ be a tuple of best-response functions that maximizes Eq. (2), and let $\mathsf{Tr}(\zeta_{\mathcal{X}}^{\mathfrak{B}})$ denote a trace sampled under a complete assignment $\mathcal{X}$ updated by a tuple $\mathfrak{B}$.*

Since ρ_{max} is the greatest positive robustness value, convergence to ρ_{max} implies positive robustness. By the soundness of the quantitative semantics Li et al. (2017), positive robustness implies satisfaction. Therefore,

$$\rho\big(\mathsf{Tr}(\zeta_{\mathcal{X}}^{\mathfrak{B}}), \psi\big) \xrightarrow{*} \rho_{max} \quad \Longrightarrow \quad \rho\big(\mathsf{Tr}(\zeta_{\mathcal{X}}^{\mathfrak{B}}), \psi\big) > 0 \quad \Longrightarrow \quad \mathsf{Tr}(\zeta_{\mathcal{X}}^{\mathfrak{B}}) \models \psi.$$

Thus, convergence to ρ_{max} guarantees satisfaction of ψ. By the order-preservation assumption, a tuple with a higher probability of convergence cannot have a lower probability of satisfaction. Since $\mathfrak{B}$ maximizes the probability of convergence in Eq. (2), it also maximizes the probability of satisfying ψ. Hence, maximizing this satisfaction probability is exactly the optimization problem stated in Sec. 4. Therefore, $\mathfrak{B}$ also solves the original optimization problem.

B.2 Proof of Lemma 5.2

If the objective in Eq. (3) strictly preserves the ordering induced by Eq. (2) over well-typed response-function tuples, then every maximizer of Eq. (3) also maximizes Eq. (2).

Proof 2 *Let $\mathcal{B}$ denote the set of well-typed response-function tuples, and let $J_2, J_3 : \mathcal{B} \to \mathbb{R}$ denote the objectives in Eq. (2) and Eq. (3), respectively. By assumption, J_3 strictly preserves the ordering induced by J_2; that is, for any $\mathbf{B}, \mathbf{B}' \in \mathcal{B}$, we have:*

$$J_2(\mathbf{B}) > J_2(\mathbf{B}') \quad \Longrightarrow \quad J_3(\mathbf{B}) > J_3(\mathbf{B}').$$

Let $\mathbf{B}^* \in \arg\max_{\mathbf{B}\in\mathcal{B}} J_3(\mathbf{B})$. Suppose, for contradiction, that $\mathbf{B}^*$ does not maximize J_2. Then there exists some $\widehat{\mathbf{B}} \in \mathcal{B}$ such that:

$$J_2(\widehat{\mathbf{B}}) > J_2(\mathbf{B}^*).$$

By strict order preservation, this implies:

$$J_3(\widehat{\mathbf{B}}) > J_3(\mathbf{B}^*),$$

which contradicts the assumption that $\mathbf{B}^*$ maximizes J_3. Therefore,

$$\arg\max_{\mathbf{B}\in\mathcal{B}} J_3(\mathbf{B}) \subseteq \arg\max_{\mathbf{B}\in\mathcal{B}} J_2(\mathbf{B});$$

hence, every maximizer of Eq. (3) also maximizes Eq. (2).

B.3 Proof of Theorem 5.3

Given a POSG G and an SL sentence φ, suppose that λ_Ξ is an information-preserving latent embedding of each Ξ and that the action-level response functions can represent every well-typed response function. Then any $\langle \mathbf{BR}^a_{x_i} \rangle_{i\in\mathbb{Q}^\exists}$ that maximizes the expected cumulative discounted robustness induces a $\langle \mathbf{BR}_{x_i} \rangle_{i\in\mathbb{Q}^\exists}$ that maximizes the objective in Eq. (3). By Lemma 5.2, this tuple also optimizes Eq. (2), and hence, by Lemma 5.1, optimizes the probability of satisfying φ in G as defined in Sec. 4.

Proof 3 *Let us write:*

$$\mathfrak{B}^a = \langle \mathbf{BR}^a_{x_i} \rangle_{i\in\mathbb{Q}^\exists}$$

as a tuple of action-level best-response functions maximizing the expected cumulative discounted robustness value. Since λ_Ξ is information-preserving for every strategy context Ξ, the latent representation retains all information required by a well-typed best-response function. Hence, for every $\mathbf{BR}^a_{x_i}$ in the optimal tuple $\mathfrak{B}^a$, there exists an induced best-response function $\mathbf{BR}_{x_i}$ on the original strategy context, such that:

$$\mathbf{BR}_{x_i}(\Xi)(h) = \mathbf{BR}^a_{x_i}(\lambda_\Xi, h),$$

and the two functions induce the same action choices given the same history h, and therefore the same distribution over trajectories. Consequently, for every complete assignment $\mathcal{X} \in \mathbb{X}_c$, the trajectory distribution induced by $\mathfrak{B}^a$ in the latent representation coincides with that induced by the corresponding best-response function tuple:

$$\mathfrak{B} = \langle \mathbf{BR}_{x_i} \rangle_{i\in\mathbb{Q}^\exists}$$

in the original representation. In particular, their expected cumulative discounted robustness values are equal:

$$J_a(\mathfrak{B}^a) = J_3(\mathfrak{B}),$$

where J_a denotes the action-level objective defined immediately before Theorem 5.3 and J_3 denotes the objective in Eq. (3).

Because the embedding is information-preserving and the action-level response functions can represent every well-typed response function, every well-typed response tuple has an equivalent action-level tuple that induces the same trajectory distribution and objective value. Therefore, if $\mathfrak{B}^a$ maximizes J_a, the induced tuple $\mathfrak{B}$ maximizes J_3; otherwise, a well-typed tuple with a strictly larger value of J_3 would induce a latent-level response tuple with a strictly larger value of J_a, contradicting the optimality of $\mathfrak{B}^a$.

Thus,

$$\mathfrak{B} \in \arg\max_{\mathfrak{B}\in\mathcal{B}} J_3.$$

By Lemma 5.2, every maximizer of Eq. (3) also maximizes the objective in Eq. (2). Hence,

$$\mathfrak{B} \in \arg\max_{\mathfrak{B}\in\mathcal{B}} J_2.$$

Finally, by Lemma 5.1, every maximizer of Eq. (2) solves the optimization problem of Sec. 4 (i.e., it maximizes the probability of satisfying the SL sentence ψ in G). Therefore, any tuple of optimal latent-level best-response functions $\mathfrak{B}^a$ induces a tuple $\mathfrak{B}$ that solves the original satisfaction-probability optimization problem formalized in Eq. (1).

C Full Quantitative Semantics of Inner-LTL formulas

Below, we present the full quantitative semantics of the inner-LTL formulas of SL, adapted from Li et al. (2017).

$$\begin{aligned}
\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \mathsf{true}\big) &= \rho_{max} \\
\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), f\big(L(s_\ell)\big)\big) &= f\big(L(s_\ell)\big) \\
\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \neg\psi\big) &= -\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \psi\big) \\
\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \bigcirc\psi\big) &= \rho\big(\mathsf{Tr}(\zeta_{[\ell+1:k]}), \psi\big) \ \text{ if } \ (k > \ell). \\
\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \square\psi\big) &= \min_{i\in[\ell,k)} \rho\big(\mathsf{Tr}(\zeta_{[i:k]}), \psi\big) \\
\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \diamondsuit\psi\big) &= \max_{i\in[\ell,k)} \rho\big(\mathsf{Tr}(\zeta_{[i:k]}), \psi\big) \\
\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \psi_1 \wedge \psi_2\big) &= \min\big(\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \psi_1\big), \rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \psi_2\big)\big) \\
\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \psi_1 \vee \psi_2\big) &= \max\big(\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \psi_1\big), \rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \psi_2\big)\big) \\
\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \psi_1 \,\mathcal{U}\, \psi_2\big) &= \max_{i\in[\ell,k)} \Big(\min\Big(\rho\big(\mathsf{Tr}(\zeta_{[i:k]}), \psi_2\big), \min_{j\in[\ell,i)} \rho\big(\mathsf{Tr}(\zeta_{[j:i]}), \psi_1\big)\Big)\Big) \\
\rho\big(\mathsf{Tr}(\zeta_{[\ell:k]}), \psi_1 \mathcal{T} \psi_2\big) &= \max_{i\in[\ell,k)} \Big(\min\Big(\rho\big(\mathsf{Tr}(\zeta_{[i:k]}), \psi_2\big), \max_{j\in[\ell,i)} \rho\big(\mathsf{Tr}(\zeta_{[j:i]}), \psi_1\big)\Big)\Big)
\end{aligned}$$

D Experimental Setup and Implementation Details

In this section, we describe the implementation details, architectures, hyperparameters, and loss functions of the models used in our experiments.

D.1 Details on Shield

We train a recurrent binary classifier to distinguish rational predator (agent bound to universally quantified variable) compositions from irrational tasks. At each time step, the classifier receives the prey's (agent bound to existentially quantified variable) current observation concatenated with its five previous actions. The previous action vector is zero at the beginning of each interaction sequence. The architecture consists of a single-layer LSTM with 128 hidden units, followed by a fully connected layer with 64 units, a ReLU activation, and a scalar output layer. The recurrent hidden and cell states are reset independently for each interaction sequence.

Data are collected using a uniform-random prey strategy across $m = 20$ training tasks, comprising of ten rational and ten irrational tasks. For each task, we collect 1,000 training interaction sequences and 200 independent calibration interaction sequences, each lasting 20 steps. Ten interaction sequences from the same task form a training bundle. Their final-step logits are averaged before applying a sigmoid to obtain the bundle's classification score. The classifier is trained using standard class-weighted binary cross-entropy; the balanced dataset gives equal weights to both classes. Interaction sequence are randomly regrouped within each task at every epoch, and the resulting bundles are shuffled. Training runs for 100 epochs using Adam with a learning rate of 0.001, minibatches of 64 bundles, and gradient-norm clipping at 1.0.

Shield in Deployment. During deployment (main training phase), each collected trajectory receives the stored classification of its opponent task. Experience classified as random is excluded from the actor, critic, and opponent-model losses, while the remaining losses are averaged over retained experience; if none remains in a batch, we skip the update.

D.2 Details on Encoder, Decoders and Future-Action Prediction

Encoder (E). SLICR uses a recurrent encoder to summarize the action-observation history of each agent bound to an existentially quantified variable into a latent context representation. At each time step, the encoder receives the agent's current local observation concatenated with its previous one-hot-encoded action. The encoder consists of a single-layer LSTM with 128 hidden units, followed by a fully connected layer with 128 units and a ReLU activation. A final linear layer produces a 32-dimensional ($d = 32$) embedding without an output activation.

Training uses five-step sequences with truncated backpropagation through time, storing the exact recurrent states at each sequence's start for replay. The encoder is optimized using Adam with a fixed learning rate of 0.0007 and gradient-norm clipping at 0.5.

Action-Reconstruction Decoder (DA). The action-reconstruction decoder predicts the current action of each agent bound to a preceding universally quantified variable from the encoder's latent context embedding ($\lambda \in \Lambda \subseteq \mathbb{R}^{32}$). It uses a shared feed-forward network with two fully connected hidden layers of 128 units, each followed by a ReLU activation. Separate linear outputs produce action logits for each agent bound to a preceding universally quantified variable, and a softmax converts these logits into categorical action probabilities. This decoder is trained jointly with the encoder using Adam with a learning rate of 0.0007 and gradient-norm clipping at 0.5.

Observation-Reconstruction Decoder (DO). The observation-reconstruction decoder predicts the agents bound to a preceding universally quantified variable observations from the encoder's latent context embedding ($\lambda \in \Lambda \subseteq \mathbb{R}^{32}$). It consists of two fully connected hidden layers with 128 units each, followed by ReLU activations, and a final linear layer that outputs the reconstructed observation vector without an output activation. This branch has separate parameters from the action-reconstruction branch. It is trained jointly with the encoder using Adam with a learning rate of 0.0007 and gradient-norm clipping at 0.5.

Future-Action Predictor (F). The future-action predictor estimates each agent bound to a preceding universally quantified variable action distribution over the next 5 steps from the encoder's latent embedding. It consists of two fully connected hidden layers with 128 units each and ReLU activations, followed by a linear output layer. The output is organized by prediction horizon and an agent bound to a preceding universally quantified variable, with a softmax producing a categorical action for each combination. The predictor is trained jointly with the encoder using Adam with a learning rate of 0.0007 and gradient-norm clipping at 0.5.

Training. We jointly train the encoder (E), observation-reconstruction decoder (DO), action-reconstruction decoder (DA), and future-action predictor (F) using samples collected. Samples are divided into five-step sequences, and the encoder is replayed from its stored hidden and cell states. Its latent embedding at each step is shared by all three prediction branches. Observation reconstruction (DO_{wo}) uses a standard squared-error loss. We sum the squared differences between predicted and recorded observation features, multiply by 0.5, and average over retained steps and environments. Action reconstruction (DA_{wa}) uses standard categorical cross-entropy against the opponents' recorded actions. The losses are summed across opponents and averaged over retained steps and environments. Future-action predictor (F_{wf}) uses standard categorical cross-entropy against recorded future actions. The loss is averaged over steps, environments, agents bound to a preceding universally quantified variables , and the five prediction horizons. Targets beyond the available rollout or across episode boundaries are excluded.

The combined loss for learning the latent context embedding is calculated as follows:

$$\mathcal{L}_\lambda = \mathcal{L}_{DO} + \mathcal{L}_{DA} + 0.1\,\mathcal{L}_F.$$

Gradients from all three terms update the shared encoder, while each prediction branch is updated through its own term. We use a shared Adam optimizer with a learning rate of 0.0007, clipping gradient norms to 0.5 separately for the encoder, decoders, and future-action predictor. Backpropagation is truncated at five-step boundaries, while recurrent states carry information across sequences.

Information-preserving latent context embedding Let w^* denote the encoder parameters obtained by minimizing the combined loss $\mathcal{L}_\lambda$. The resulting embedding is $\widehat{\lambda} = E_{w^\star}(h_{\alpha^i})$. By jointly reconstructing current actions and observations and predicting future actions, the encoder is encouraged to preserve the information needed to approximate λ. However, because the encoder has access only to the agent's local action-observation history, different strategy contexts may be observationally indistinguishable. Therefore, the learned embedding may only approximate the information-preserving latent context embedding.

D.3 Details on Training the Action-level Response Functions

Predator-Prey. SLICR uses A2C algorithm with a single-layer LSTM containing 128 hidden units. The LSTM processes the current observation and previous one-hot action. Its output is concatenated with the current observation and corrected 32-dimensional latent embedding, producing a 174-dimensional input to the actor and critic. Each network contains two independent fully connected hidden layers with 128 units and ReLU activations. The actor produces a categorical distribution over five actions, while the critic produces a scalar value estimate.

GRF. SLICR uses a recurrent PPO algorithm with a 128-unit LSTM. The current observation, 32-dimensional latent embedding, and history output form a 275-dimensional input to the actor and critic. Each network contains three independent fully connected hidden layers with 256 units and ReLU activations. The actor produces a categorical distribution over 19 actions, and the critic produces a scalar value estimate. PPO performs five optimization epochs per rollout update, using a clipping threshold of 0.2 for both the probability ratio and value prediction.

SMAC-HARD. SLICR uses the same 128-unit LSTM and separate actor and critic networks, each with three 256-unit ReLU hidden layers. Each controlled unit receives its own local observation, previous action, and locally computed latent embedding. Response functions parameters are shared across controlled units, while recurrent hidden and cell states are maintained separately for each unit. Observation and action dimensions depend on the map. Unavailable actions are masked before constructing the categorical action distribution. PPO uses five optimization epochs and a clipping threshold of 0.2.

Training. Across all three benchmarks, the actor and critic share the LSTM, which remains separate from the E. Training uses five-step recurrent sequences, with hidden and cell states carried across sequences and reset at episode boundaries. The response function's learning rate remains constant for the first 30% of training and then decreases linearly to one-tenth of its initial value. For PPO, behavior log probabilities, value estimates, and advantages are fixed before optimization begins; recurrent chunks are shuffled between epochs while steps order within each chunk is preserved. Details on the hyperparameters for training the response functions appear in Table 2.

E Benchmark Details

E.1 Predator and Prey

Environment. We used (Papoudakis et al., 2021a) implementation of predator and prey. The predator–prey environment is a two-dimensional arena containing one prey, three predators, and two stationary obstacles. All agents choose among five discrete actions: $\{Stay, left, right, Up, Down\}$. Episodes terminate after 50 steps. The prey has partial observation. It observes its own position and velocity, together with the relative positions of obstacles and predators within visible range.

SL Formula. We used the reward used in (Papoudakis et al., 2021a) for the guiding the LIAM for the baseline. For SLICR we use φ_{pp} as the SL formula in our experiments:

$$\begin{aligned}\varphi_{\text{pp}} \triangleq\ & \forall x_1\, \forall x_2\, \forall x_3\, \exists x_4\, (\text{pred_1}, x_1)(\text{pred_2}, x_2)(\text{pred_3}, x_3)(\text{prey}, x_4).\\ & \square\Diamond \mathsf{Exact_One}(\texttt{pos_pred_1}, \texttt{pos_pred_2}, \texttt{pos_pred_3}, \texttt{pos_prey}) \wedge\\ & \neg \mathsf{Two\&More_Eq}(\texttt{pos_pred_1}, \texttt{pos_pred_2}, \texttt{pos_pred_3}, \texttt{pos_prey}) \wedge\\ & \neg \mathsf{Out_of_Boundary}(\texttt{pos_prey})\end{aligned}$$

Table 2: Hyperparameters of response function of SLICR

Hyperparameter	PP	GRF	SMAC-HARD
Algorithm	A2C	PPO	PPO
LSTM layers	1	1	1
LSTM hidden units	128	128	128
Hidden layers per actor/critic	2	3	3
Units per hidden layer	128	256	256
Hidden activation	ReLU	ReLU	ReLU
Opponent embedding dimension	32	32	32
Optimizer	Adam	Adam	Adam
Initial learning rate	3×10^{-4}	3×10^{-4}	7×10^{-4}
Final learning rate	3×10^{-5}	3×10^{-5}	7×10^{-5}
Learning-rate decay start	30%	30%	30%
Learning-rate decay schedule	Linear	Linear	Linear
Maximum gradient norm	0.5	0.5	10.0
Recurrent sequence length	5	5	5
Discount factor γ	0.99	0.99	0.99
GAE parameter λ	0.95	0.95	0.95
Entropy coefficient	0.01	0.01	0.01
PPO optimization epochs	–	5	5
PPO probability-ratio clipping	–	0.2	0.2
PPO value clipping	–	0.2	0.2

Although $\square\Diamond$ conventionally expresses a Büchi recurrence condition over infinite traces, under our finite-prefix evaluation it checks whether Exact_One holds at the final step of each prefix. Because we evaluate the robustness of every prefix throughout an episode, this condition rewards repeated occurrences of contact with exactly one predator.

Learning set of Strategies. We have two sets of strategies. The first set adopted from (Papoudakis et al., 2021a) contains four heuristic strategies: pursuing the prey, following a randomly selected predator, approaching the nearest other agent, and moving toward the largest absolute relative-coordinate displacement among other agents. It also includes three pretrained IA2C (Papoudakis et al., 2021b) strategies, three pretrained MADDPG (Lowe et al., 2017), and observation-independent strategies (irrational strategies) that sample uniformly from the five available actions. Opponent tasks are formed by assigning a strategy to each of the three predator slots (Simulating the job of complete assignment $\mathcal{X} \in \mathbb{X}_c$).

Second, we construct a learned set of strategies with four training styles: direct, intercept, flank, and spread. Direct strategies use the environment reward without additional shaping. Intercept strategies are encouraged to approach the prey's projected position using its current velocity, flank strategies are encouraged to approach from different directions, and spread strategies are encouraged to maintain greater separation between predators. These behaviors are encouraged through reward shaping with a coefficient of 0.05 rather than prescribed action rules. The set of strategies comprises IPPO training strategies with direct, intercept, and flank styles, plus held-out strategies covering new instances, the unseen spread style, and MAPPO (Yu et al., 2022) opponents. Check more details on the opponent strategy sets in Table 3.

E.2 Google Research Football (GRF)

Environment. We use two scenarios in Google Research Football (Kurach et al., 2020): Academy Run to Score (R2S) with Keeper and a modified version of this scenario (R2S-DEF). In both, the agent controls an attacker who starts near midfield with the ball and attempts to score against five defenders and a goalkeeper. In (R2S), all five defenders initially approach from behind the attacker. In (R2S-DEF), two defenders start between the attacker and the goal, while the remaining three approach from behind. We cap episodes at 200 steps in both scenarios, with earlier termination following a goal, the ball leaving play, or a change in possession.

Table 3: Training and evaluation strategy sets in Predator–Prey. Each team contains three predators. D: direct pursuit; I: interception; F: flanking; S: spreading. Tactic counts denote distinct individual strategies.

Setting	`Seen`	`Unseen`
Novel Strategy	Π^{Train}:Sampled from 30 IPPO 12 D + 9 I + 9 F Π^{Test}: 30 fixed teams from IPPO	Π^{Train}:Sampled from 30 IPPO 12 D + 9 I + 9 F Π^{Test}: 21 IPPO + 12 MAPPO
Inspired by LIAM	Π^{Train}: 4 heuristic + 3 IA2C + 3 MADDPG Π^{Test}:4 heuristic + 3 IA2C + 3 MADDPG	Π^{Train}: 4 heuristic + 3 IA2C + 3 MADDPG Π^{Test}: New Composition from the Training set

Both scenarios use 19 discrete actions, including movement, passing, shooting, sprinting, dribbling, and tackling. Observations use the 115-dimensional embedding, containing player positions and movement directions, ball information, possession, the active player, and the game mode. In the partially observed setting, opponent positions and movement directions are available within a Euclidean distance of 0.6 in normalized coordinates; otherwise, these values are replaced with -5.

SL Formula. We use the default reward used in (Kurach et al., 2020) for the guiding the LIAM for the baseline. For SLICR we use φ_{R2S} and $\varphi_{\text{R2S-DEF}}$ as the SL formula in our experiments:

$$\begin{aligned}\varphi_{\text{R2S}} \triangleq\ & \forall x_1 \ldots \forall x_6\, \exists x_7\, (\text{def_1}, x_1)(\text{def_2}, x_2) \ldots (\text{def_5}, x_5)(\text{keeper}, x_6)(\text{atk}, x_7).\\ & \big(\mathsf{Reach_check_1}(\texttt{pos_atk})\ \mathcal{T}_W\ \mathsf{Reach_check_2}(\texttt{pos_atk})\mathcal{T}_W \ldots \mathcal{T}_W\\ & \mathsf{Reach_check_10}(\texttt{pos_atk})\mathcal{T}_W\ \mathsf{Shot_on_Target}(\texttt{atk})\ \mathcal{T}_W\ \mathsf{Score}(\texttt{atk})\big)\\ & \wedge \Box \neg \mathsf{Terminate}(\texttt{atk}) \wedge \Box(\bigwedge_{i=1}^{5} \neg \mathsf{Score}(\texttt{def_i}))\end{aligned}$$

$$\begin{aligned}\varphi_{\text{R2S-DEF}} \triangleq\ & \forall x_1 \ldots \forall x_6\, \exists x_7\, (\text{def_1}, x_1)(\text{def_2}, x_2) \ldots (\text{def_5}, x_5)(\text{keeper}, x_6)(\text{atk}, x_7).\\ & \big(\mathsf{Reach_check_1}(\texttt{pos_atk})\ \mathcal{T}_W\ \mathsf{Reach_check_2}(\texttt{pos_atk})\mathcal{T}_W \ldots \mathcal{T}_W\\ & \mathsf{Reach_check_10}(\texttt{pos_atk})\mathcal{T}_W\ \mathsf{Shot_on_Target}(\texttt{atk})\ \mathcal{T}_W\ \mathsf{Score}(\texttt{atk})\big)\\ & \wedge \Box(\bigwedge_{i=1}^{5} \neg \mathsf{Ball_Pos}(\texttt{def_i})) \wedge \Box(\bigwedge_{i=1}^{5} \neg \mathsf{Score}(\texttt{def_i}))\end{aligned}$$

Learning set of Strategies. We construct separate strategy sets for the R2S and R2S-DEF GRF scenarios by pretraining six-player teams against the built-in football AI. Each team contains five outfield players and a goalkeeper, with separate strategies for each player. Every strategy uses a feed-forward network with two 64-unit ReLU hidden layers, mapping a 115-dimensional observation to a categorical distribution over 19 actions. Teams train for one million environment steps with independent random seeds. IPPO uses a separate critic for each player, whereas MAPPO (Yu et al., 2022) uses a shared centralized critic during training.

All six players receive the same terminal reward: +1 for gaining possession, -1 for conceding a goal, and 0 for other episode endings. We add a small behavioral reward at each step, weighted by 0.001, to encourage different defensive styles. Low-block rewards favor positions near the penalty area and designated defensive locations. Medium-block rewards favor a defensive band farther from goal and discourage retreating into the penalty area. Pressing rewards encourage the three closest outfield players to approach the ball. Zonal rewards encourage players to follow assigned defensive regions that shift with the ball, together with a smaller incentive to apply pressure. For each scenario, the strategy set consists of IPPO teams exhibiting low-block, medium-block, and pressing tactics. Independently trained MAPPO teams (Yu et al., 2022), exhibiting low-block, medium-block, pressing, and zonal tactics, are reserved for evaluation. The exact opponent-strategy settings used in our experiments are shown in Table 4.

Table 4: Strategy set setting in GRF. LB: low block; MB: medium block; PR: pressing; ZN: zonal.

Maps	Seen	Unseen
R2S	Π^{Train}:4 LB + 3 MB + 3 PR, Trained using IPPO Π^{Test}:4 LB + 3 MB + 3 PR, Trained using IPPO	Π^{Train}:4 LB + 3 MB + 3 PR, Trained using IPPO, Π^{Test}: 3 LB + 2 MB + 2 PR + 3 ZN Trained using MAPPO
R2S-DEF	Π^{Train}:4 LB + 3 MB + 3 PR, Trained using IPPO Π^{Test}:4 LB + 3 MB + 3 PR, Trained using IPPO	Π^{Train}:4 LB + 3 MB + 3 PR, Trained using IPPO, Π^{Test}: 3 LB + 2 MB + 2 PR + 3 ZN Trained using MAPPO

E.3 SMAC-HARD

Environment. We use SMAC-HARD (Deng et al., 2024), an extension of the StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019). Whereas standard SMAC uses the built-in StarCraft II AI for the opposing team, SMAC-HARD provides multiplayer maps and an interface for externally controlled enemy strategies. It also introduces extensions for fog of war, unit abilities, terrain, and visibility. These features are inherited from SMAC-HARD. We extend SMAC-HARD by integrating PyMARL to train enemy-team strategies and deploy their frozen checkpoints as opponent populations. Our implementation supports learned, scripted, and built-in-AI opponents, with explicit task IDs defining controlled training and seen/unseen evaluation splits. We expose opponent action labels and physical features for auxiliary supervision while preserving the allied agents' original observations. We synchronize all participating clients so that each environment transition advances exactly eight frames. Each episode starts with fresh, living teams, and simulation step and reset failures raise errors rather than being recorded as battle losses. Games are restarted between episodes when needed to avoid the simulation time limit, while preserving training progress. Evaluation preserves the training random-number states and performs no parameter updates. In our experiments, we use the `3s_vs_3z` map, where the first team contains three Stalkers opposing three Zealots, the `3m` map, where each team contains three Marines, and the `2s3z`, where each team contains two Stalkers and three Zealots. Agents receive local SMAC observations and select available movement, stop, and targeted attack actions. The first-team agents are bound to existentially quantified variables, while the second-team agents are bound to universally quantified variables.

SL Formula. We use the shaped reward introduced in (Deng et al., 2024; Samvelyan et al., 2019) for the guiding the LIAM for the baseline. For SLICR we use φ_{SMAC} as the SL formula in our experiments:

$$
\begin{aligned}
\varphi_{\text{SMAC}} \triangleq & \forall x_1 \ldots x_l \exists y_1 \ldots y_m (\text{ene_1}, x_1) \ldots (\text{ene_l}, x_l)(\text{ally_1}, y_1) \ldots (\text{ally_m}, y_m) \\
& \Big[\mathsf{Diff}\Big(\mathsf{Diff}\big(\mathsf{Sum}(\texttt{Health_ene_1}, \ldots, \texttt{Health_ene_l}), \\
& \mathsf{Sum}(\texttt{Health_prev_ene_1}, \ldots, \texttt{Health_prev_ene_l}) \big), \\
& \mathsf{Diff}\big(\mathsf{Sum}(\texttt{health_ally_1}, \ldots, \texttt{health_ally_m}), \\
& \mathsf{Sum}(\texttt{health_prev_ally_1}, \ldots, \texttt{health_prev_ally_m}) \big) \Big] \ \mathcal{U}_W \\
& \mathsf{Eliminate}(\texttt{ene_1}, \ldots, \texttt{ene_l}) \ \mathcal{U}_W \ \mathsf{Victory}(\texttt{ally_1}, \ldots, \texttt{ally_m})
\end{aligned}
$$

Learning set of Strategies. We extend SMAC-HARD with an explicit opponent-task interface. Each task selects a complete enemy team at the beginning of an episode, and that team remains active throughout the battle. The interface supports scripted strategies, pretrained PyMARL (Samvelyan et al., 2019) teams, and native StarCraft II AI at different difficulty levels (i.e. 8, 9, 10). We additionally integrate PyMARL with the enemy side, enabling teams trained using VDN, QMIX, and QTRAN to be deployed as fixed opponents. The exact opponent-strategy settings used in our experiments are shown in Table 5.

Table 5: Strategy set setting in SMAC-HARD.

Maps	Seen	Unseen
3m	Π^{Train}: 2× VDN + 3× QMIX, Π^{Test}: 2× VDN + 3× QMIX	Π^{Train}: 2× VDN + 3× QMIX, Π^{Test}:5× QTRAN
2s3z	Π^{Train}: 2× VDN + 3× AI-8 Π^{Test}: 2× VDN + 3× AI-8	Π^{Train}: 2× VDN + 3× AI-8 Π^{Test}: 2× QMIX + 3× AI-10
3s_vs_3z	Π^{Train}: 3× VDN + 2× QMIX Π^{Test}: 3× VDN + 2× QMIX	Π^{Train}: 3× VDN + 2× QMIX Π^{Test}:5× QTRAN

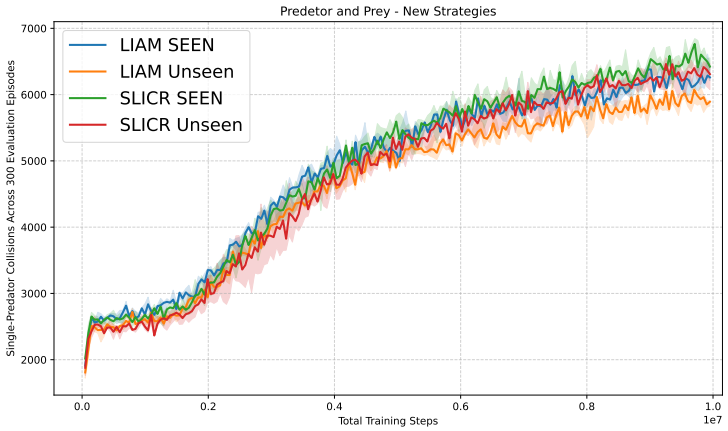


Figure 6: Learning curves for SLICR and LIAM on PP with updated population set of strategies for opponents, showing the median number of single-collision of prey and predator and the 25th–75th percentiles.

F Additional Experimental Results

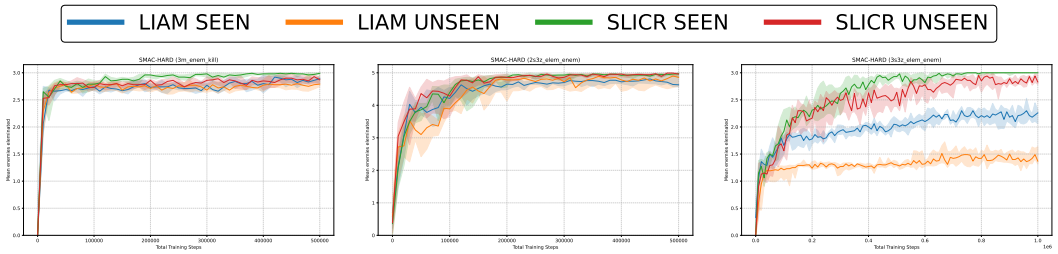
F.1 SLICR vs. LIAM

Fig. 6 compares SLICR and LIAM on PP using a new population of opponent strategies constructed with IPPO and MAPPO. In the Seen setting, SLICR (green line) performs similarly to LIAM (blue line) during most of the training process. However, during the final training steps, SLICR achieves up to ↑200 more single predator-prey collisions. In the Unseen setting, the advantage is more substantial, with SLICR (red line) achieving up to ↑500 more single predator-prey collisions than LIAM (orange line) during the final training steps.

We also report the number of eliminated agents bound to universally quantified variables in Fig. 7 and existentially quantified variables in Fig. 8. As shown in Fig. 7c in 3s_vs_3z map, SLICR (green line) eliminates all agents bound to universally quantified variables in the Seen setting. In the Unseen setting, SLICR (red line) eliminates nearly all such agents, whereas LIAM (orange line) eliminates only approximately half of them. Similarly, on the 3s_vs_3z map (see Fig. 8c), SLICR (green line) preserves two agents bound to existentially quantified variables in the Seen setting. In the Unseen setting, SLICR (red line) preserves approximately two agents, whereas LIAM (orange line) loses nearly all of them.

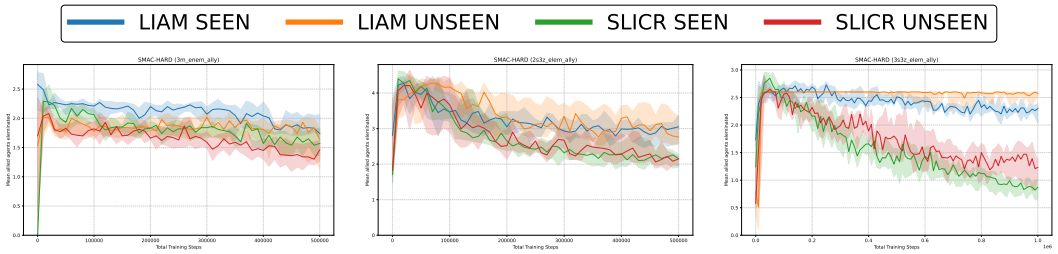
F.2 SLICR $\forall^*\exists^*$ vs. $\exists^*\forall^*$

As shown in Table 1, on PP, GRF R2S, and GRF R2S-DEF, SLICR ($\forall^*\exists^*$) (blue line), (orange line) performs significantly better than SLICR ($\exists^*\forall^*$) (green line), (red line) in both the Seen and Unseen settings. Although Table 1 shows that SLICR ($\forall^*\exists^*$) and SLICR ($\exists^*\forall^*$) achieve similar overall performance on the SMAC-HARD scenarios, their learning curves reveal important differences. As shown in Figs. 9d to 9f, SLICR ($\forall^*\exists^*$) (blue line), (orange line) converges faster and



(a) SMAC-HARD (3m) **(b)** SMAC-HARD (2s3z) **(c)** SMAC-HARD (3s_vs_3z)

Figure 7: Learning curves for SLICR and LIAM on SMAC-HARD, showing the median number of eliminated opponents (universally quantified) and the 25th–75th percentiles.



(a) SMAC-HARD (3m) **(b)** SMAC-HARD (2s3z) **(c)** SMAC-HARD (3s_vs_3z)

Figure 8: Learning curves for SLICR and LIAM on SMAC-HARD, showing the median number of eliminated Allied Agents (existentially quantified) and the 25th–75th percentiles.

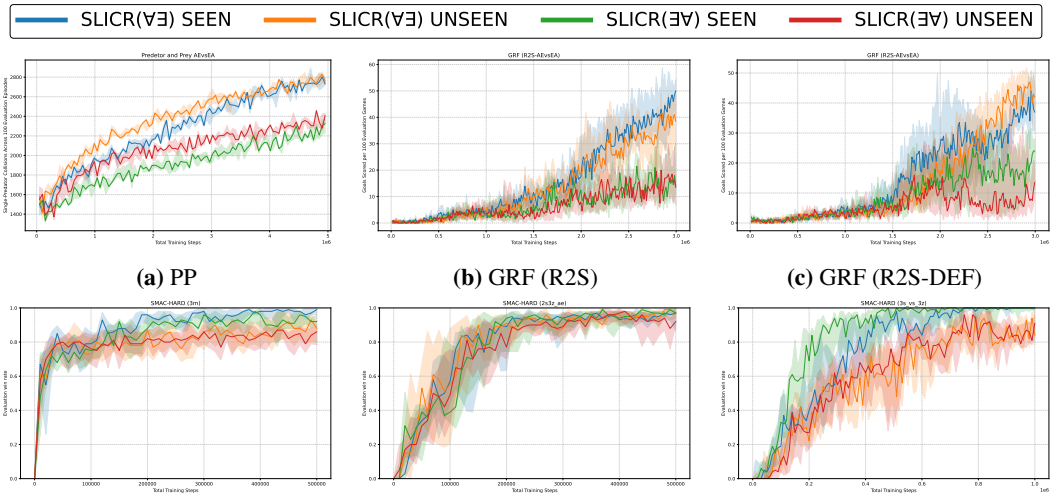
achieves better results than SLICR ($\exists^*\forall^*$) (green line), (red line) on the `3m` map in both the `Seen` and `Unseen` settings.

F.3 Ablation Study of the Shield

In addition to the experiments comparing SLICR with LIAM in the presence of irrational strategies, we conduct an ablation study comparing SLICR with SLICR without the shield $\mathbb{O}$, as shown in Fig. 10. The results show that the shield improves the performance of SLICR across all considered proportions of irrational strategies. In particular, when 50% of the strategies are irrational (blue line), the shield enables SLICR to achieve up to $\uparrow 250$ more single-predator collisions, with an average gain of approximately $\uparrow 100$. When the strategy set contains 40% (orange line) or 30% (green line) irrational strategies, the performance gains are smaller but still reach up to $\uparrow 150$ additional single-predator collisions. Finally, when 20% of the strategies are irrational (red line), the shield continues to provide a positive average performance gain over SLICR without the shield.

F.4 Inner-LTL vs. Shaped Reward.

In addition to GRF R2S-DEF, where we investigate the effect of inner-LTL guidance. In Fig. 11, we conduct the same analysis on the GRF R2S scenario. In the `Seen` setting (blue line), inner-LTL of formula φ_{R2S} guidance achieves up to $\uparrow 10$ more goals than the shaped reward. However, in the `Unseen` setting (orange line), the shaped reward performs better during most of training, with inner-LTL guidance outperforming it only between 2M and 2.5M training steps. We conduct similar experiments on the SMAC-HARD benchmarks. On the `3m` map (see Fig. 12), inner-LTL guidance achieves up to a $\uparrow 18\%$ higher win rate early in training in the `Seen` setting (blue line). However, the gain becomes negligible once both SLICR and SLICR without inner-LTL guidance reach nearly 100% win rates. Similarly, on the `2s3z` map (see Fig. 13), inner-LTL guidance achieves gains of up to $\uparrow 25\%$ and $\uparrow 20\%$ early in training in the `Seen` (blue line) and `Unseen` (orange line) settings, respectively. After approximately 200K training steps, both variants reach nearly 100% win rates, and their performance becomes similar.



(a) PP **(b)** GRF (R2S) **(c)** GRF (R2S-DEF)

(d) SMAC-HARD (3m) **(e)** SMAC-HARD (2s3z) **(f)** SMAC-HARD (3s_vs_3z)

Figure 9: Learning curves for SLICR $\forall^*\exists^*$ vs. $\exists^*\forall^*$, showing the median and 25th–75th percentiles.

G Computational Resources

The experimental platform consisted of two AMD EPYC 7742 processors with 128 physical cores and 256 threads, approximately 1 TiB of RAM, and three NVIDIA RTX PRO 6000 Blackwell Max-Q Workstation Edition GPUs with 96 GB of memory each.

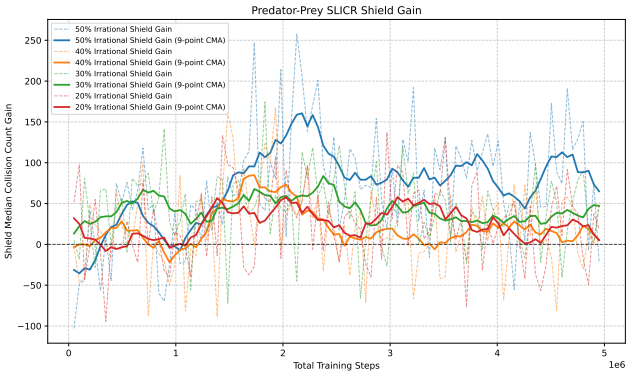


Figure 10: Performance gains from the shield under different proportions of irrational strategies.

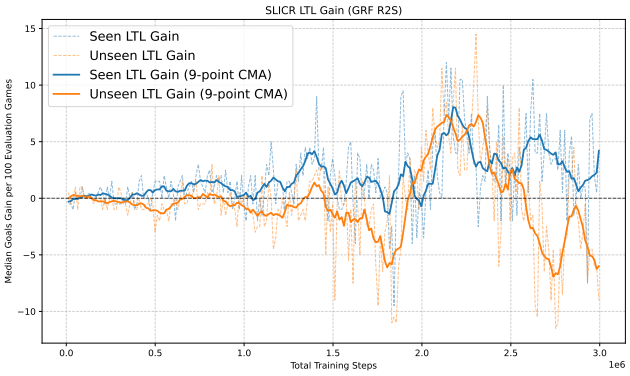


Figure 11: Gain from inner-LTL guidance over shaped rewards on GRF R2S.

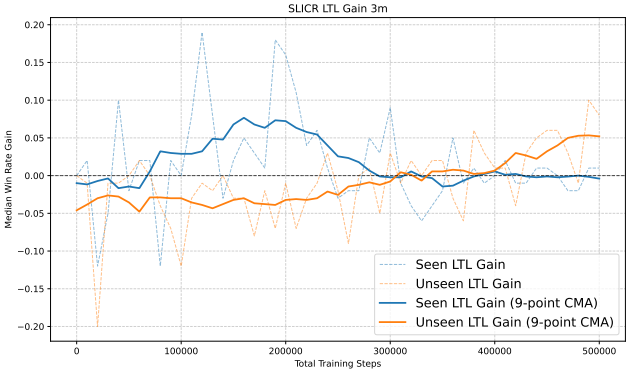


Figure 12: Gain from inner-LTL guidance over shaped rewards on `3m`.

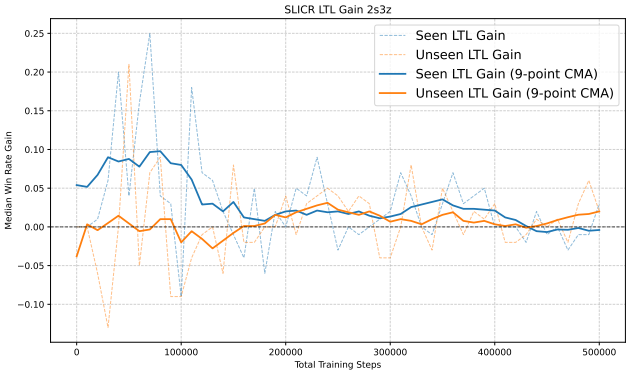


Figure 13: Gain from inner-LTL guidance over shaped rewards on SMAC-HARD `2s3z`.