LANGUAGE-GUIDED SAFE POLICY OPTIMIZATION FOR MULTI-AGENT LOCOMOTION CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Can high-level language guidance improve safe coordination in multi-agent continuous control? We study this question in safety-constrained multi-agent reinforcement learning (MARL) for cooperative robotic locomotion, where multiple agents jointly control a shared body under an explicit safety budget. We introduce a framework that augments existing multi-agent policy optimisers with an episodic language-based meta-controller and reward penalisation. During training, the meta-controller periodically queries a large language model (LLM) using a compact textual summary of the system state and recent safety statistics to generate a cooperative strategy. The strategy is embedded and incorporated into each agent's observation through a learned gating module, while the underlying policy is trained under centralised training with decentralised execution (CTDE). Unlike Lagrangian constrained-MARL methods, our approach uses fixed reward penalisation and does not require additional cost critics or dual-variable updates.

Experiments across six Safe Multi-Agent MuJoCo locomotion tasks show that language-guided policy optimisation improves the reward–safety trade-off across multiple robotic locomotion settings. In particular, LLM-HAPPO achieves higher rewards than standard constrained MARL baselines and a matched HAPPO variant using the same reward shaping, while generally maintaining low safety costs. The framework is also compatible with different MARL backbones. Overall, our results demonstrate the potential of episodic language guidance as a lightweight mechanism for improving safety-aware multi-agent policy optimisation without replacing the learned decentralised policies.

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) has become a powerful framework for learning coordinated behavior in systems where multiple agents jointly control a shared environment. In continuous-control problems, especially robotic locomotion and embodied control, a widely adopted paradigm is centralized training with decentralized execution (CTDE), where policies are trained with access to global information while each agent acts only from its local observation at test time (Lowe et al., 2017; Sunehag et al., 2018; Rashid et al., 2020; Amato, 2024). Policy-gradient methods such as MAPPO (Yu et al., 2022) and HAPPO/HATRPO (Grudzien Kuba et al., 2022)have shown that CTDE can scale effectively to challenging cooperative control tasks. However, safe multi-agent robotic control remains difficult because agents must learn not only fine-grained motor behavior but also high-level coordination strategies under shared safety constraints (Gu et al., 2023).

Large language models (LLMs) offer a complementary source of high-level structure. Recent work shows that LLMs can support reasoning and planning (Wei et al., 2022) and can provide useful semantic priors for embodied decision making (Huang et al., 2022a; Ahn et al., 2022; Huang et al., 2022b; Brohan et al., 2023; Huang et al., 2023). In robotic locomotion settings, such high-level guidance is particularly appealing because unsafe or poorly coordinated actions can quickly lead to falls, unstable gaits, or unsafe exploration. However, it remains underexplored how language guidance can be integrated into CTDE-based actor-critic pipelines for safety-constrained continuous control without replacing low-level policies or destabilizing optimization. We address this problem with an LLM-guided safe MARL framework in which the LLM acts as a slow-timescale meta-controller. Every few episodes, it receives a natural-language summary of the current robot state, recent safety-cost trends, and task context and returns a concise cooperative strategy. This strategy is encoded into

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.

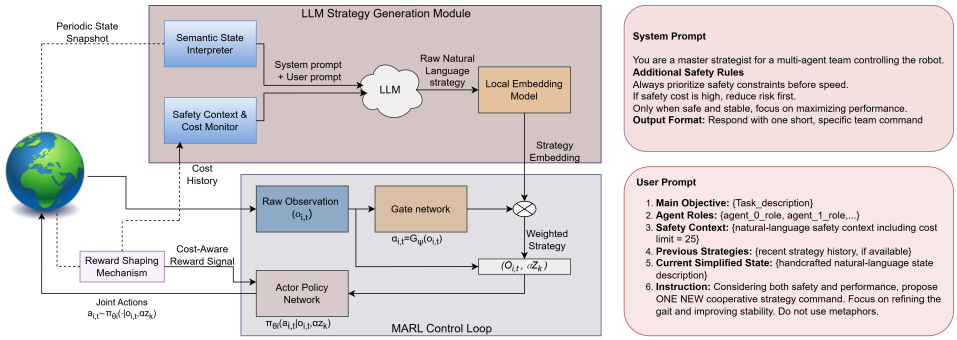

Figure 1: Overview of the LLM-guided framework. An LLM generates high-level strategies from environment summaries and safety costs, encoded as embedding z_k and integrated via a gating mechanism into MAPPO/HAPPO policies. The LLM does not produce actions; safety is enforced through cost signals and reward shaping.

a fixed-dimensional embedding and injected into each agent's observation through a learned gating mechanism, while the underlying MARL backbone remains responsible for low-level control. Safety is enforced through a simple penalized-reward formulation, avoiding per-minibatch dual updates and additional cost critics. We evaluate the framework on Safe Multi-Agent MuJoCo robotic locomotion tasks and show that episodic language guidance improves the empirical reward-safety trade-off while remaining compatible with standard MARL backbones under CTDE.

2 Related Work

Multi-Agent Reinforcement Learning, CTDE, and Safety. Multi-agent reinforcement learning (MARL) has made substantial progress through the CTDE paradigm, in which policies are trained with access to additional global information while remaining decentralized at execution time (Amato, 2024). Representative approaches include centralized-critic methods such as MADDPG (Lowe et al., 2017), value-decomposition methods such as VDN (Sunehag et al., 2018) and QMIX (Rashid et al., 2020), and policy-gradient methods such as MAPPO (Yu et al., 2022) and HAPPO/HATRPO (Grudzien Kuba et al., 2022). While these methods have demonstrated strong performance in cooperative decision-making and continuous-control settings, they primarily optimize task return and do not explicitly account for safety. Safe reinforcement learning is commonly formulated using constrained Markov decision processes, where a policy must maximize reward while satisfying explicit cost limits (Altman, 1999; García & Fernández, 2015). Classical approaches include constrained optimization and reward-penalization methods such as CPO (Achiam et al., 2017), RCPO (Tessler et al., 2018), and responsive Lagrangian methods (Stooke et al., 2020). In the multi-agent setting, safety becomes more challenging because constraint violations often arise from poor coordination under coupled dynamics rather than isolated agent-level mistakes. Prior work extends safe RL to cooperative multi-agent continuous control, introducing formulations and methods such as MAPPO–Lag and MACPO (Gu et al., 2022) on the Safe Multi-Agent MuJoCo benchmark (Gu et al., 2023). More recently, projection-based approaches such as MACUP (Liu et al., 2025) avoid explicit dual updates by directly enforcing constraints during policy updates. However, existing methods still require additional cost critics or careful hyperparameter tuning, which can complicate training and often lead to conservative behavior.

LLMs for Embodied and Multi-Agent Decision Making. Large language models (LLMs) have recently emerged as useful sources of high-level reasoning and semantic abstraction for decision making (Wei et al., 2022). In embodied and robotic settings, prior work has used LLMs for task decomposition, planning, feedback-driven reasoning, and grounding language in robot affordances, as in Language Models as Zero-Shot Planners, SayCan, Inner Monologue, RT-2, and VoxPoser (Huang et al., 2022a; Ahn et al., 2022; Huang et al., 2022b; Brohan et al., 2023; Huang et al., 2023). These works show that language models can provide structured priors that complement low-level control, but they mainly focus on single-robot planning or language-conditioned embodied agents. In parallel, LLMs have also been explored in multi-agent settings as communication agents, role assigners, or

centralized planners for cooperative interaction and agent societies, as illustrated by CAMEL and AutoGen (Li et al., 2023; Wu et al., 2024). However, this line of work has largely focused on dialogue, task allocation, or symbolic collaboration rather than decentralized continuous control. As a result, the integration of LLM-generated guidance into CTDE-based MARL remains limited.

Gap to This Work. Taken together, prior work suggests two complementary strengths: CTDE-based MARL provides an effective framework for decentralized continuous control, while LLMs provide high-level semantic priors for reasoning, planning, and constraint interpretation. Recent work has begun to study language in safe MARL, for example by using natural-language constraints to describe unsafe behaviors. However, existing language-based safe MARL methods primarily focus on interpreting or embedding textual constraints, rather than using an LLM as an episodic strategy generator for safety-aware coordination in continuous-control CTDE pipelines. Existing safe MARL methods such as MACPO and MAPPO-Lag provide explicit constraint-handling mechanisms, but do not incorporate semantic strategy guidance. Our work addresses this gap by injecting periodically generated LLM strategies into CTDE-based multi-agent policy optimization through a learned gated embedding, while enforcing safety using a simple penalty-based reward-shaping mechanism. Our work addresses this gap by incorporating periodically generated LLM strategies into CTDE-based multi-agent policy optimization through a learned gated embedding, while accounting for safety through a simple penalty-based reward-shaping mechanism.

3 Problem Formulation

Safe multi-agent reinforcement learning as a *constrained Markov game* defined by the tuple $(\mathcal{N}, \mathcal{S}, \mathcal{A}, P, \mu, \gamma, R, C, \mathbf{b})$, where $\mathcal{N} = \{1, \ldots, N\}$ is the set of agents, $\mathcal{S}$ is the global state space, and the joint action space is $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$. The transition kernel $P(s' \mid s, \mathbf{a})$ governs environment dynamics, μ is the initial-state distribution, and $\gamma \in [0, 1)$ is the discount factor. The team reward function is $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Each agent i is additionally associated with m_i safety cost functions $C_i = \{c_{i,1}, \ldots, c_{i,m_i}\}$, with $c_{i,j} : \mathcal{S} \times \mathcal{A}_i \to \mathbb{R}$. Every cost dimension has a corresponding limit $\mathbf{b}_i = (b_{i,1}, \ldots, b_{i,m_i})$. These constraints give rise to a multi-agent CMDP structure. We consider a cooperative multi-agent reinforcement learning (MARL) setting modeled as a *constrained Markov game*. The environment consists of N agents interacting over discrete time $t = 0, 1, \ldots$ with continuous states, actions, and safety constraints. At each timestep, agent i receives a *local observation* $o_{i,t}$ sampled from an observation function $O_i(o_{i,t}|s_t)$, and selects an action according to its decentralized policy $\pi_i(a_{i,t}|o_{i,t})$. The global joint action $\mathbf{a}_t = (a_{1,t}, \ldots, a_{N,t})$ is executed, producing a reward $r_{i,t}$, a safety cost $c_{i,t}$, and the next state s_{t+1}. Since the setting is fully cooperative, all agents optimize a common cumulative reward:$J_r(\pi) = \sum_{i=1}^{N} \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_i(s_t, \mathbf{a}_t)]$. Safety requires each agent's long-term discounted cost to remain below its limit: $J_c^i(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t c_i(s_t, \mathbf{a}_t)] \leq b_i, \qquad \forall i.$ A feasible policy set is $\Pi_{\text{safe}} = \left\{\pi_\theta \in \Pi_\theta \;\middle|\; J_c^i(\pi_\theta) \leq b_i,\ \forall i\right\}$. The goal of the constrained Markov game is to identify an optimal cooperative policy: $\pi^\star = \arg\max_{\pi_\theta \in \Pi_{\text{safe}}} J_r(\pi_\theta)$. Classical constrained RL approaches enforce these requirements via Lagrange multipliers or dual-gradient updates (Stooke et al., 2020; Wah et al., 2000). While theoretically sound, such methods are often brittle in multi-agent continuous control due to sensitivity to learning rates and oscillatory dual variables(Gu et al., 2023). In our experiments, this formulation is instantiated on Safe Multi-Agent MuJoCo velocity-control tasks, where the benchmark provides both the task reward and the safety cost signal.

4 Method

We propose an LLM-guided safe multi-agent reinforcement learning framework built on the centralized-training–decentralized-execution (CTDE) paradigm. The core idea is to use a large language model as a slow-timescale meta-controller that periodically generates high-level cooperative strategies from natural-language state summaries and recent safety information, while the underlying multi-agent policy optimizer remains responsible for low-level continuous control. Our framework consists of four main components: a natural-language state interpreter, an episodic LLM-based strategy generator, a gated strategy-embedding module that augments each agent's local observation, and a penalty-based policy optimization procedure. The proposed language-guidance module is optimizer-agnostic and can be seamlessly integrated with standard multi-agent policy optimization methods, including MAPPO, HAPPO and HATRPO. Figure 1 illustrates the framework, showing

Algorithm 1 LLM-Guided Multi-Agent Policy Optimization

Input: Multi-agent environment with N agents; language model $\mathcal{L}$; policy π_θ; critic V_ϕ; gate network G_ψ; LLM update interval K; penalty coefficient λ_c.
Output: Learned policy, critic, and gate parameters.

Initialize network parameters, replay buffers, default strategy text t_0, and embedding $z_0 \leftarrow \mathcal{L}.\mathrm{embed}(t_0)$.
for episode $k = 0, 1, 2, \ldots$ **do**
  if LLM guidance enabled and $k \bmod K = 0$ **then**
    Construct prompt from task description, agent roles, state summary, and recent safety context.
    Query $\mathcal{L}$ for strategy text t_k and encode $z_k \leftarrow \mathcal{L}.\mathrm{embed}(t_k)$.
  end if
  for $t = 1, \ldots, T$ **do**
    For all agents $i \in \{1, \ldots, N\}$, compute $\alpha_{i,t} = G_\psi(o_{i,t})$ and augmented observations $\tilde{o}_{i,t} = [o_{i,t}, \alpha_{i,t} z_k]$.
    Sample decentralized actions $a_{i,t} \sim \pi_\theta(\tilde{o}_{i,t})$ for all i.
    Execute $\mathbf{a}_t = (a_{1,t}, \ldots, a_{N,t})$, observe rewards $\{r_{i,t}\}_{i=1}^N$, costs $\{c_{i,t}\}_{i=1}^N$, and next observations.
    Compute shaped rewards $\hat{r}_{i,t} = r_{i,t} - \lambda_c c_{i,t}$ for all i.
    Store $\left(\tilde{o}_{i,t}, a_{i,t}, \hat{r}_{i,t}, c_{i,t}\right)_{i=1}^N$ in the replay buffers.
  end for
  Compute returns and advantages from the collected rollout.
  Update θ, ϕ, and ψ using the chosen multi-agent policy optimizer.
end for

Table 1: Safe MaMuJoCo task summary, including agent decomposition, controlled components, and penalty coefficient λ. Cost limit = 25 for all tasks.

Environment	Agents	Controlled Components	Penalty λ
Safety2x4AntVelocity-v0	2 agents (4 joints each)	Front legs / back legs	5.0
Safety4x2AntVelocity-v0	4 agents (2 joints each)	One leg group per agent	5.0
Safety2x3HalfCheetahVelocity-v0	2 agents (3 joints each)	Back joints / front joints	7.5
Safety6x1HalfCheetahVelocity-v0	6 agents (1 joint each)	One joint per agent	7.5
Safety2x3Walker2dVelocity-v0	2 agents (3 joints each)	Left leg / right leg	5.0
Safety3x1HopperVelocity-v0	3 agents (1 joint each)	Thigh / knee / torso	4.0

how language guidance, safety signals, and decentralized control are incorporated within the training pipeline. In the main experiments, we focus on MAPPO and HAPPO as representative backbones, while additional results for HATRPO are provided in the Appendix. The underlying optimizer updates remain unchanged, and Algorithm 1 summarizes the general training procedure.

LLM-Based Strategy Generation. A structured prompt is constructed from several components. At each update interval, the LLM receives a structured prompt containing four types of information: a natural-language task description, a decomposition of agent roles, a compact textual summary of the current physical state, and recent safety information. The task description specifies the overall objective, while the role decomposition describes each agent's control responsibility, enabling the LLM to reason about high-level multi-agent coordination rather than low-level actions. The state summary is produced by a handcrafted rule-based interpreter over the shared observation vector, using task-specific thresholds and heuristics to extract only the most relevant high-level variables, such as torso height, body orientation, balance status, and motion cues (details are provided in Appendix). To convey safety information, we compute the recent average episode cost C_k and map it to a small set of semantic safety levels ranging from low-risk regimes to emergency conditions. This safety context is included in the prompt as natural-language text rather than raw numerical costs, allowing the LLM to condition its strategy on the current safety regime. We include a short history of recent strategy outputs in the prompt to discourage repetitive responses and encourage more diverse guidance across updates. The resulting strategy text is encoded with the offline sentence encoder `intfloat/e5-small-v2` and then converted to a fixed 32-dimensional representation $z_k \in \mathbb{R}^{32}$ to match the policy input interface. Specifically, the actor consumes an augmented observation consisting of the original observation concatenated with a fixed-width language channel, so we use a constant 32-dimensional strategy vector for stable and lightweight conditioning. This shared embedding is broadcast to all agents and incorporated through a learnable gate. For agent

Table 2: Comparison of SafeMARL baselines and the proposed LLM-guided methods on Safe MaMuJoCo tasks with a cost limit of 25. Results are reported as mean ± standard deviation over 5 seeds (10 evaluation episodes each). LLM-HAPPO achieves strong reward performance while maintaining low evaluation costs across the evaluated tasks. Bold indicates the best safe method, and gray denotes constraint violations.

Method	Metric	2×4Ant	4×2Ant	2×3HalfCheetah	6×1HalfCheetah	3×1Hopper	2×3Walker2d
MAPPO-Lag	Reward	2345.22 ± 115.23	2193.12 ± 116.93	1553.74 ± 27.95	1390.52 ± 56.54	981.13 ± 7.27	1887.84 ± 65.29
	Cost	0.00 ± 0.00	0.00 ± 0.00	1.14 ± 1.86	0.24 ± 0.29	0.00 ± 0.00	0.02 ± 0.04
MACPO	Reward	1930.55 ± 90.01	2206.33 ± 34.98	2209.40 ± 57.82	1816.79 ± 57.06	484.04 ± 63.33	1132.80 ± 313.33
	Cost	6.28 ± 9.25	25.20 ± 16.04	98.22 ± 22.16	7.30 ± 8.09	25.82 ± 0.92	28.12 ± 26.05
MACUP	Reward	1828.90 ± 361.49	1886.64 ± 158.84	2084.89 ± 253.20	1762.60 ± 129.27	1023.01 ± 4.20	975.45 ± 630.26
	Cost	12.22 ± 11.19	8.60 ± 5.75	57.28 ± 74.30	32.00 ± 40.75	0.00 ± 0.00	4.48 ± 7.74
LLM-MAPPO	Reward	2051.89 ± 68.30	2039.52 ± 85.39	2579.03 ± 99.27	$\mathbf{2292.81 \pm 104.59}$	1517.89 ± 258.51	1878.16 ± 421.67
	Cost	0.00 ± 0.00	0.02 ± 0.04	15.52 ± 14.36	21.58 ± 42.46	0.40 ± 0.66	2.20 ± 3.41
LLM-HAPPO	Reward	$\mathbf{2643.87 \pm 68.13}$	$\mathbf{2521.42 \pm 69.41}$	$\mathbf{2624.41 \pm 80.61}$	2099.01 ± 290.24	$\mathbf{1522.08 \pm 190.80}$	$\mathbf{2064.41 \pm 148.74}$
	Cost	0.20 ± 0.27	0.98 ± 0.90	1.10 ± 2.80	15.24 ± 19.10	0.80 ± 1.66	2.28 ± 2.66

i, the gate network $G_\psi : \mathbb{R}^{d_o} \to [0,1]$ produces a scalar coefficient $\alpha_{i,t} = G_\psi(o_{i,t})$, which modulates the strategy embedding as $\tilde{z}_{i,t} = \alpha_{i,t} z_k$, and the resulting augmented observation is $\tilde{o}_{i,t} = [\, o_{i,t}, \alpha_{i,t} z_k \,] \in \mathbb{R}^{d_o+32}$. This mechanism allows each agent to adaptively regulate the influence of language guidance at each timestep. From a statistical perspective, the gate acts as a state-dependent shrinkage coefficient that balances the contribution of the external language embedding and the learned policy representation. In practice, this results in a dynamic behavior over training: during early, uncertain phases, the policy relies more heavily on language guidance, while as training progresses and the policy becomes more accurate, the gate reduces this reliance and shifts toward autonomous control. This behavior is consistent with the empirical trends shown in Figure 5, where the gate value decreases as task performance stabilizes.

Penalty-Based Policy Optimization. To enforce safety, we use a simple reward-penalization scheme: $r'_{i,t} = r_{i,t} - \lambda c_{i,t}$, where $r_{i,t}$ is the environment reward, $c_{i,t}$ is the safety cost, and $\lambda > 0$ is a fixed penalty coefficient. This provides a direct safety-aware shaping signal during training while avoiding per-minibatch dual updates and the additional cost critics commonly used in Lagrangian safe RL methods. In other words, although the environment provides both safety costs and a cost limit, we do not learn a separate value function for future cost; instead, the observed cost is incorporated directly into the shaped reward. Training follows the CTDE paradigm. During training, the critic receives the global state S_t, while each decentralized actor observes only its augmented local observation $\tilde{o}_{i,t}$. For clarity, we present the update in PPO-style form, while the same language-guidance module can also be combined with backbone-specific updates such as those used in HAPPO and HATRPO. The clipped surrogate objective is $L_\pi = \mathbb{E}_t[\min(\rho_t A_t,\ \mathrm{clip}(\rho_t, 1-\epsilon, 1+\epsilon) A_t)]$, where ρ_t is the policy ratio and A_t is the generalized advantage estimate. The actor parameters θ and gate parameters ψ are optimized jointly, while the critic parameters ϕ are trained with a standard value loss. The LLM-generated embeddings are treated as fixed inputs and are not updated by gradient descent.

4.1 Theoretical Properties

We provide a theoretical characterization of the proposed language-conditioned policy optimization framework. The main observation is that, because the strategy embedding is fixed within each episode, language conditioning can be represented through an augmented state without changing the underlying Markov structure.

Assumption 1 (Episode-wise exogeneity of the embedding)**.** For each episode k, the strategy embedding z_k is determined from pre-episode information and is held fixed throughout the episode. Conditional on z_k, within-episode randomness arises only from policy sampling and environment transitions.

Theorem 2 (Episode-wise extended-state Markovization)**.** *Under Assumption 1, define the extended state $x_t = (s_t, z_k)$. For a fixed episode embedding z_k, the process is a time-homogeneous Markov control process on the extended state space $\mathcal{X} = \mathcal{S} \times \mathcal{Z}$.*

The proof of Theorem is provided in Appendix G. This result shows that conditioning the policy on a fixed strategy embedding can be interpreted within the standard Markov-control framework. The

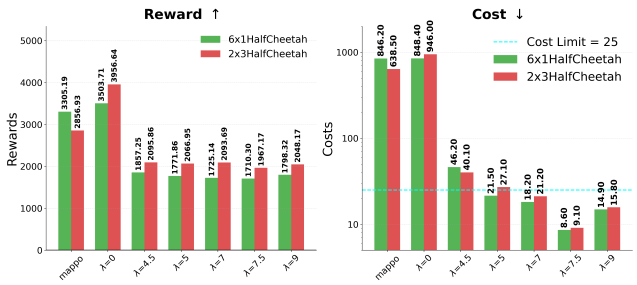

Figure 2: Effect of different values of penalty parameter λ. Comparison performance with baseline MAPPO, LLM without reward shaping ($\lambda = 0$), and LLM with reward shaping (with different values of penalty). Bars report the final episode reward (left) and safety cost (right).

appendix also provides additional analysis of the fixed-context policy gradient and the reward–cost trade-off induced by penalty-based optimization. These results provide structural characterizations of the method under their stated assumptions rather than formal safety guarantees for the neural policies used in our experiments.

This result shows that conditioning the policy on a fixed strategy embedding can be interpreted within the standard Markov-control framework. Additional analysis of the fixed-context policy gradient and the reward–cost trade-off induced by penalty-based optimization is provided in Appendix G. These results provide structural characterizations of the method under their stated assumptions rather than formal safety guarantees for the neural policies used in our experiments.

5 Experiments

Implementation Details We consider the Safe Multi-Agent MuJoCo benchmark built on Safety Gymnasium[1] and Safe Policy Optimization[2] (Ji et al., 2023), a suite of continuous-control tasks with varying numbers of agents. We instantiate the proposed framework with both MAPPO and HAPPO, integrating the LLM-driven meta-controller without modifying their underlying PPO-based optimization procedures. The benefit of language guidance is most consistent with the HAPPO backbone, whereas the MAPPO-based variant exhibits more task-dependent improvements. We select six representative velocity-based tasks from Safe MaMuJoCo, including *Safety2x4AntVelocity-v0, Safety4x2AntVelocity-v0, Safety2x3HalfCheetahVelocity-v0, Safety6x1HalfCheetahVelocity-v0, Safety3x1HopperVelocity-v0, and Safety2x3Walker2dVelocity-v0*. A summary of the environments and agent configurations is provided in Table 1. Each experiment is conducted with 5 random seeds, and all models are trained for 10 million environment steps, following the standard Safe policy optimization training protocol. We use Claude 3.5 Sonnet as our reasoning engine when using the LLM module. The MAPPO parameters remain fixed across the various tasks to allow for a fair comparison. The full breakdown of the prompt templates, LLM settings and training setups is provided in the Appendix. We also evaluate alternative LLM backbones and find that the proposed framework remains effective across a range of models, including GPT-4o-mini, LLaMA-3.1-8B, LLaMA-3.1-70B, and Gemini-2.0-Flash (details in Appendix).

Reward Shaping and Cost Penalty Selection. Safety-aware learning uses penalized reward shaping with a fixed penalty coefficient λ. We select λ via a lightweight validation sweep for each task family and reuse it across all experiments. The final choices are $\lambda = 5$ for Ant and Walker2d, $\lambda = 7.5$ for HalfCheetah, and $\lambda = 4$ for Hopper. Empirically, larger penalties were needed for task families where unsafe high-speed behavior was harder to suppress, while smaller penalties were sufficient when moderate shaping already produced stable constraint satisfaction. Figure 3 illustrate the evolution of episode reward and safety cost during training. We primarily compare our method against safety-aware baselines, including MACPO, MAPPO-Lag (Gu et al., 2022), and MACUP (Liu et al., 2025), which enforce constraints via dual optimization or projection-based

[1] https://github.com/PKU-Alignment/safety-gymnasium.git
[2] https://github.com/PKU-Alignment/Safe-Policy-Optimization.git

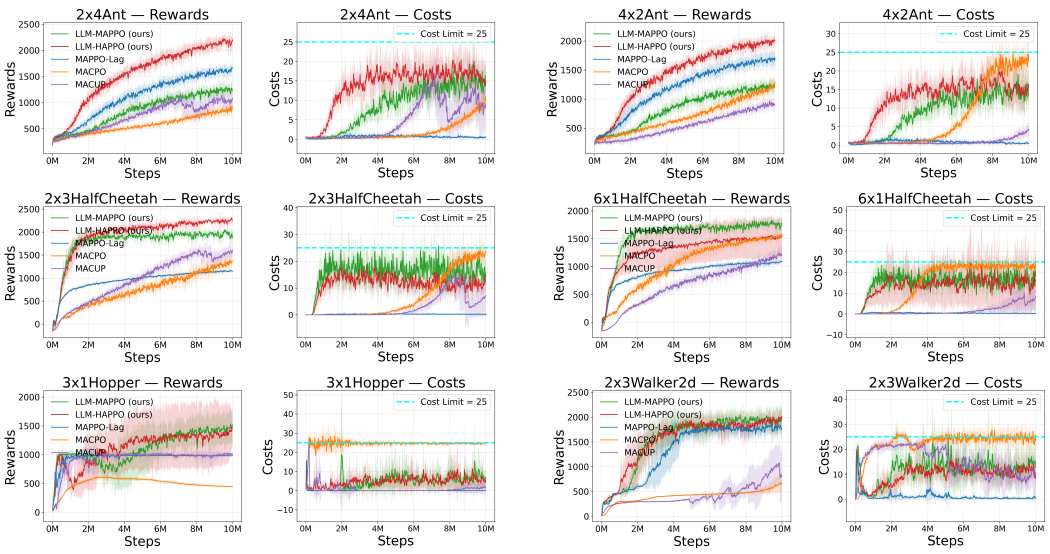

Figure 3: Training curves for safe MaMuJoCo tasks. Each plot shows episode reward (left) and safety cost (right) during training. Shaded regions indicate standard deviation across five random seeds.

updates. For MACUP, we implement the algorithm based on the paper due to the lack of public code. We also analyze the sensitivity of the penalty coefficient λ, which balances reward maximization and safety constraint satisfaction. As shown in Figure 2, the impact of λ varies across tasks, reflecting differences in environment dynamics and constraint difficulty. We observe that appropriate choices of λ lead to stable learning while maintaining safety costs within the specified limit. Based on this analysis, we select task-specific λ values that provide robust performance across seeds. The training time along with the LLM inference overhead during training (tokens and cost), are provided in the Appendix.

6 Results

The comparison with the constrained MARL baselines is presented in Table 2, with the corresponding training curves shown in Figure 3. Overall, the proposed methods achieve a favorable reward-cost trade-off across most of the evaluated tasks. LLM-HAPPO provides the most consistent improvements, achieving higher rewards across the evaluated tasks compared with the constrained MARL baselines. To separate the effect of language guidance from reward shaping, we additionally compare LLM-HAPPO with HAPPO in Table 3 using the same penalty-based reward shaping. LLM-HAPPO achieves higher mean rewards across all four evaluated tasks in this matched comparison, although the magnitude of the improvement varies across environments. These results suggest that the observed improvements cannot be attributed to reward shaping alone. The MAPPO-based variant exhibits a more task-dependent pattern. LLM-MAPPO improves reward over its reward-shaped MAPPO counterpart on some tasks, while the improvement is not consistent across all environments. We therefore consider LLM-HAPPO the primary empirical result, while LLM-MAPPO demonstrates the applicability of the proposed language-conditioning mechanism to another PPO-based MARL backbone. Across the evaluated tasks, the proposed methods generally maintain low safety costs, although the penalty-based formulation does not provide a hard constraint-satisfaction guarantee.

Comparison with reward-shaped HAPPO. To isolate the contribution of language guidance from reward shaping, we compare LLM-HAPPO against HAPPO+RS using matched training and evaluation protocols and comparable tuning budgets. As shown in Table 3, LLM-HAPPO achieves higher mean reward across all four evaluated tasks. Mean safety cost is reduced on Ant, HalfCheetah, and Walker2d, whereas Hopper exhibits a higher mean cost under LLM-HAPPO. These results indicate that language guidance provides additional empirical reward gains beyond reward shaping on the evaluated tasks, although improvements in safety cost are not uniform across environments.

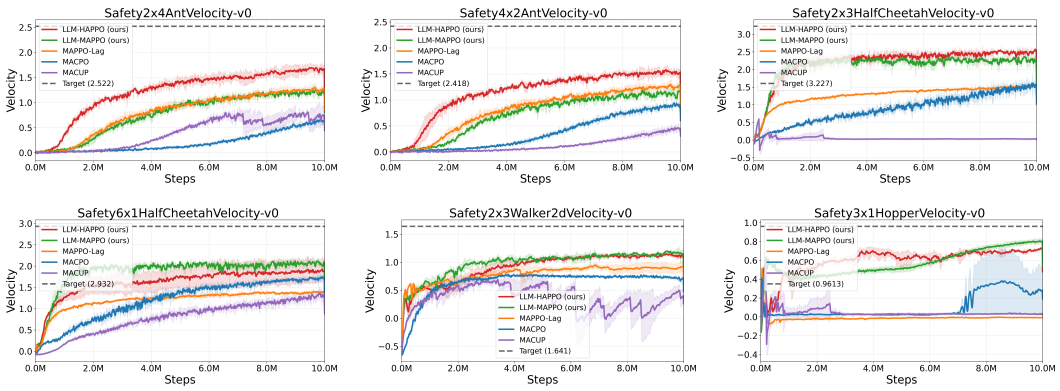

Figure 4: Velocity profiles during training on Safe MaMuJoCo tasks. LLM-guided methods achieve higher velocity in most environments, indicating improved coordination among agents. Dashed lines denote velocity thresholds used for safety cost computation.

Figure 4 further illustrates the velocity profiles during training across different tasks, the proposed methods show faster learning progress and higher velocity in most environments, indicating more effective coordination among agents. A cost is incurred when the velocity exceeds a predefined threshold. A notable observation is that in certain environments, LLM-MAPPO achieves competitive or even slightly higher early-stage performance, whereas LLM-HAPPO demonstrates more consistent and stable long-term improvements. We also observe that MAPPO-Lag tends to exhibit conservative behavior, maintaining very low cost throughout training but often converging to lower velocity. In contrast, MACPO shows suboptimal learning and less stable performance, which may limit its ability to reach higher velocity regimes within the same training budget. Importantly, the observed gains in velocity do not appear to come at the cost of safety. As shown in Table 2, both LLM-MAPPO and LLM-HAPPO maintain costs within the specified constraint threshold (cost ≤ 25) across all tasks.

These results suggest that LLM-guided policies can support improved coordination among agents, leading to higher performance while generally maintaining low safety costs. We also provide supplementary rollout videos for methods (LLM-MAPPO, LLM-HAPPO), safe baselines (MAPPO-Lag, MACPO, MACUP), and unsafe methods (MAPPO, HAPPO), enabling qualitative comparison of coordination patterns, safety behavior, and locomotion dynamics observed in the quantitative results. Videos further support these observations, In several environments, some baseline methods exhibit conservative or limited movement patterns, consistent with their lower achieved velocity and slower policy improvement during training. Figure 5 in Appendix illustrates how the learned gate modulates reliance on LLM-generated strategy guidance across different phases of training on Safety2x3HalfCheetahVelocity-v0. Higher gate activation appears during more difficult training phases, while lower activation is observed once locomotion becomes stable. The accompanying curves show that as success rate saturates and episode velocity improves, the gate value gradually decreases, suggesting reduced reliance on LLM-generated strategic guidance. The shaded regions highlight high gate-value phases, and the dashed line indicates the velocity threshold. Overall, the figure supports the view that the LLM provides adaptive strategic guidance that is most useful during challenging stages of learning.

Language versus non-language conditioning. To investigate whether the performance gains arise from language-based guidance or merely from adding an auxiliary conditioning pathway, we introduce a parameter-matched non-language baseline. This variant replaces the language-generation and sentence-encoding pipeline with a learned numerical encoder operating on the same structured state and safety features. The gating mechanism, HAPPO optimizer, reward shaping, and training budget remain unchanged. As shown in Table 4, LLM-HAPPO achieves higher mean evaluation reward on all six tasks. It also achieves lower mean cost on five of the six tasks, with the exception of 6×1 HalfCheetah, where the non-language baseline obtains a lower mean cost. These results suggest that the language-based pathway provides benefits beyond the inclusion of an auxiliary conditioning module. However, because the non-language results use three seeds and the reported

Table 3: Comparison of HAPPO with reward shaping (HAPPO+RS) and LLM-HAPPO on four Safe Multi-Agent MuJoCo tasks. Results are reported as mean ± standard deviation over 5 random seeds. Both methods use matched training and evaluation protocols and comparable RS tuning budgets.

Environment	Method	Reward ↑	Cost ↓
2×4 Ant	HAPPO+RS	2478.61 ± 67.06	0.38 ± 0.53
	LLM-HAPPO	$\mathbf{2643.87 \pm 68.13}$	$\mathbf{0.20 \pm 0.27}$
2×3 HalfCheetah	HAPPO+RS	2050.05 ± 499.13	2.72 ± 3.17
	LLM-HAPPO	$\mathbf{2624.41 \pm 80.61}$	$\mathbf{1.10 \pm 2.80}$
2×3 Walker2d	HAPPO+RS	2034.51 ± 218.14	3.31 ± 2.04
	LLM-HAPPO	$\mathbf{2064.41 \pm 148.74}$	$\mathbf{2.28 \pm 2.66}$
3×1 Hopper	HAPPO+RS	1391.51 ± 625.14	$\mathbf{0.12 \pm 0.24}$
	LLM-HAPPO	$\mathbf{1522.08 \pm 190.80}$	0.80 ± 1.66

Table 4: Comparison of LLM-HAPPO with a parameter-matched non-language conditioning baseline across six Safe Multi-Agent MuJoCo tasks. The non-language baseline replaces the language-generation and sentence-encoding pipeline with a learned numerical encoder while retaining the same structured inputs, gating mechanism, HAPPO optimizer, and reward-shaping configuration Results are reported as mean ± standard deviation.

Task	LLM-HAPPO		Non-language	
	Reward ↑	Cost ↓	Reward ↑	Cost ↓
2×4 Ant	$\mathbf{2643.87 \pm 68.13}$	$\mathbf{0.20 \pm 0.27}$	744.75 ± 551.45	61.88 ± 54.40
4×2 Ant	$\mathbf{2521.42 \pm 69.41}$	$\mathbf{0.98 \pm 0.90}$	2478.02 ± 107.23	5.53 ± 7.95
2×3 HalfCheetah	$\mathbf{2624.41 \pm 80.61}$	$\mathbf{1.10 \pm 2.80}$	220.02 ± 692.11	4.57 ± 5.88
6×1 HalfCheetah	$\mathbf{2099.01 \pm 290.24}$	15.24 ± 19.10	202.54 ± 670.95	$\mathbf{0.03 \pm 0.06}$
2×3 Walker2d	$\mathbf{2064.41 \pm 148.74}$	$\mathbf{2.28 \pm 2.66}$	1346.46 ± 751.33	11.07 ± 2.36
3×1 Hopper	$\mathbf{1522.08 \pm 190.80}$	0.80 ± 1.66	452.63 ± 503.30	$\mathbf{3.47 \pm 6.00}$

LLM-HAPPO results use five seeds, this comparison should be interpreted as indicative rather than as a matched-seed estimate of the language pathway's contribution.

7 Conclusion

In this work, we study the role of language guidance in safety-constrained multi-agent reinforcement learning using Safe Multi-Agent MuJoCo. We investigate whether high-level strategies generated by an LLM can provide useful coordination and safety guidance during training while leaving fine-grained continuous control to the underlying MARL policies. Across six locomotion tasks, our results show that language guidance can benefit cooperative policy learning, with the most consistent improvements observed with HAPPO. In particular, LLM-HAPPO achieves higher mean rewards than a matched HAPPO variant using the same reward shaping across all four evaluated tasks while generally maintaining low safety costs. The results with MAPPO are more task-dependent, suggesting that the effectiveness of language guidance also depends on the underlying MARL backbone. Overall, our findings suggest that language can serve as a useful high-level coordination signal during multi-agent policy learning, complementing rather than replacing learned decentralized control. The current penalty-based formulation does not provide formal safety guarantees, and studying language-guided coordination in more diverse and less structured multi-agent domains remains an important direction for future work.

AI use statement

Generative AI tools are an integral component of the proposed method in this work. We use a large language model to generate high-level cooperative strategies from structured task, state, and safety information during training. Generative AI tools were also used for language editing and improving the clarity of the manuscript. All AI-assisted text was reviewed and verified by the authors. The

authors take full responsibility for the methodology, experiments, analysis, claims, and final content of the paper.

ETHICS STATEMENT

This work studies language-guided safe multi-agent reinforcement learning entirely in simulated environments and involves no human subjects or sensitive personal data. We use publicly available benchmarks and external language models in accordance with their intended research use. While the framework is motivated by safety-aware control, it does not provide formal safety guarantees and should not be interpreted as validated for deployment in real-world safety-critical systems.

REPRODUCIBILITY STATEMENT

We provide the algorithmic details, model architectures, training hyperparameters, LLM integration settings, prompt design, and evaluation protocol in the main paper and appendices. The appendix additionally reports ablation studies, implementation details, and theoretical derivations needed to understand and reproduce the proposed framework. Experiments are reported over multiple random seeds, with the corresponding evaluation settings specified in the paper.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 22–31, 2017.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. URL https://arxiv.org/abs/2204.01691.

Eitan Altman. *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1999.

Christopher Amato. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2409.03052*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/abs/2307.15818.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge?, 2020. URL https://arxiv.org/abs/2011.09533.

Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015. URL http://jmlr.org/papers/v16/garcia15a.html.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022.

Shangding Gu, Jakub Grudzien Kuba, Munning Wen, Ruiqing Chen, Ziyan Wang, Zheng Tian, Jun Wang, Alois Knoll, and Yaodong Yang. Multi-agent constrained policy optimisation, 2022. URL https://arxiv.org/abs/2110.02793.

Shangding Gu, Jakub Grudzien Kuba, Yuanpei Chen, Yali Du, Long Yang, Alois Knoll, and Yaodong Yang. Safe multi-agent reinforcement learning for multi-robot control. *Artificial Intelligence*, 319: 103905, 2023.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022a. URL https://arxiv.org/abs/2201.07207.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022b. URL https://arxiv.org/abs/2207.05608.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. Safety gymnasium: A unified safe reinforcement learning benchmark. In *Advances in Neural Information Processing Systems*, volume 36, pp. 18964–18993, 2023.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 51991–52008. Curran Associates, Inc., 2023.

Yang Liu, Xiang Feng, and Huiqun Yu. A safe multi-agent reinforcement learning algorithm using constraint update projection approach. *Engineering Applications of Artificial Intelligence*, 161: 111929, 2025. ISSN 0952-1976. URL https://www.sciencedirect.com/science/article/pii/S0952197625019311.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9133–9143. PMLR, 2020.

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. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2085–2087, 2018.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12 (NeurIPS 1999)*, pp. 1057–1063. MIT Press, 1999.

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018. URL https://arxiv.org/abs/1805.11074.

Benjamin W. Wah, Tao Wang, Yi Shang, and Zhe Wu. Improving the performance of weighted lagrange-multiplier methods for nonlinear constrained optimization. *Information Sciences*, 124(1):241–272, 2000. ISSN 0020-0255. doi: https://doi.org/10.1016/S0020-0255(99)00081-X. URL https://www.sciencedirect.com/science/article/pii/S002002559900081X.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. In *ICLR 2024 Workshops: LLMAgents*, 2024. URL https://mlanthology.org/iclrw/2024/wu2024iclrw-autogen/.

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 *Advances in Neural Information Processing Systems*, volume 35, pp. 24611–24624, 2022.

A Appendix

B Large Language Model (LLM) Configuration

The strategic reasoning layer of our framework utilizes the *claude-3.5-sonnet* model via the OpenRouter API. To maintain consistency across rollout threads and ensure focused command generation, we utilized the following inference hyperparameters:

- **Temperature:** Set to 0.75 to balance exploratory strategy generation with consistent, physically grounded locomotion behaviors.
- **Max Tokens:** The maximum output length is restricted to 80 tokens to enforce concise, high-level directives and to control per-query inference cost and latency.
- **System Prompting:** A multi-tier system prompt is used, prioritizing safety constraints and stability over velocity-based performance.
- **Few-Shot Grounding:** When available, task-specific examples are injected into the context window to align the LLM outputs with the expected command structure and semantics.

C LLM-Generated Strategy Analysis

Analysis of LLM-generated strategy logs in Table 5 for the task 2×4Ant velocity reveals a clear progression in behavior over training. Early episodes are dominated by stabilization-focused commands emphasizing torso alignment, balance recovery, and standing up. During mid-training, the LLM frequently alternates between locomotion attempts and explicit reset or recovery strategies, indicating safety-aware fallback behavior when instability persists. In later stages, the LLM increasingly produces coordinated gait strategies characterized by rhythmic diagonal or front-back leg alternation with explicit phase offsets and increased thrust. Importantly, even at late training stages, the LLM continues to revert to low-force stabilization and posture-recovery commands when unsafe conditions are detected, demonstrating persistent safety awareness. Table 5 reports representative strategies generated by the LLM for selected episodes. Full LLM-generated strategy logs for all tasks are included with this supplementary files.

The commands are high-level natural-language strategies generated by the LLM and are not executed directly; they are embedded and incorporated into the policy through a gated observation mechanism. While the LLM-generated strategies are generally helpful, we observe occasional noisy or inconsistent suggestions. Importantly, the proposed gating mechanism allows the policy to regulate the influence of such strategies, preventing degradation in performance or safety.

Table 5: LLM-generated strategy commands for the 2x4 Ant Velocity task at different training stages.

Episode	LLM Strategy Output
0	All agents stabilize torso rotation by reducing roll and pitch angles to zero. (*Pure early stabilization*)
20	All agents level the torso by minimizing pitch angle first, followed by roll. (*Explicit posture correction (roll/pitch reasoning)*)
70	Alternate diagonal leg pairs with 50% phase offset for continuous forward thrust. (*First appearance of coordinated gait logic*)
120	STAND UP: Agents coordinate symmetrical leg forces at 30% power to push upright. (*Strong safety fallback (explicit STAND UP)*)
170	Front legs lift and push at 35% power while back legs drive at 45% power in alternating pairs. (*Mixed coordination + propulsion (transition phase)*)
210	Coordinate legs with 60% forward thrust, alternating front–back pairs at 0.15 s intervals. (*Sustained forward locomotion strategy*)
280	Front legs lift and push at 40% force while back legs push at 45% force with 0.15 s phase offset. (*Advanced gait with phase offsets*)
360	Front and back leg pairs alternate 30% force in diagonal sequence for a stable forward gait.(*Stable forward gait after learning*)
380	Reset to stable stance: all legs align vertically under body with zero force. (*Pure safety-driven reset emphasizing posture recovery with no propulsion*)
580	Synchronize diagonal legs at 0.3s phase offset with 80% power thrust. (*High-performance coordinated gait demonstrating aggressive but structured locomotion*)
620	Front-back pairs alternate at 0.4s offset with 70% thrust, maintain 3cm ground clearance. (*Mature coordinated gait with explicit environmental safety awareness*)
650	Coordinate legs with 60% thrust power, front pair leads by 0.3s delay. (*Advanced coordination with leader–follower structure*)
690	All agents apply 25% downward force in diagonal pairs to restore upright position. (*Safety-aware fallback triggered during late-stage training*)
750	Alternate diagonal legs with 40% forward thrust, 0.15s phase delay between pairs. (*Refined and stable diagonal gait after convergence*)
900	Alternate diagonal leg pairs at 16% force with 0.3s timing gap for stable forward propulsion. (*Late-stage conservative locomotion preserving stability*)

D Ablation Studies

We perform ablation studies to examine the contributions of reward shaping, language embeddings, LLM update frequency, and the gating mechanism. We first examine the effect of reward shaping (RS) and language guidance using matched HAPPO- and MAPPO-based comparisons. As shown in Table 6, LLM-HAPPO achieves higher mean reward than HAPPO+RS on all three shared tasks while maintaining low safety costs. For MAPPO, the effect is more task-dependent: LLM-MAPPO improves reward over MAPPO+RS on Ant and HalfCheetah, while MAPPO+RS achieves higher reward on Walker2d. Removing reward shaping from LLM-MAPPO leads to substantially higher safety costs across all three tasks. Overall, these results highlight the importance of reward shaping for controlling safety costs, while the benefit of language guidance depends on the underlying MARL backbone.

Effect of removing the gating mechanism. To further examine the contribution of the learned gating mechanism, we compare the full gated LLM-MAPPO with a No-Gate variant that directly concatenates the unweighted LLM embedding with each agent's observation. Table 7 reports the corrected five-seed evaluation results. The gated variant achieves higher mean reward on 2×3 HalfCheetah and 3×1 Hopper, whereas No-Gate obtains a slightly higher mean reward on 2×4 Ant. Gating also reduces mean evaluation cost on HalfCheetah and Ant, but No-Gate achieves lower mean cost on Hopper. The No-Gate variant exhibits larger across-seed reward standard deviations on all three tasks and larger cost standard deviations on HalfCheetah and Hopper.

Table 6: Effect of language guidance and reward shaping (RS) for HAPPO and MAPPO on three shared Safe MaMuJoCo tasks. Results are reported as mean ± standard deviation over five random seeds. Higher reward and lower cost are preferred.

Environment	Method	Reward ↑	Cost ↓
2×4 Ant	HAPPO+RS	2478.61 ± 67.06	0.38 ± 0.53
	LLM-HAPPO	$\mathbf{2643.87 \pm 68.13}$	$\mathbf{0.20 \pm 0.27}$
	MAPPO+RS	2003.15 ± 96.57	0.15 ± 0.21
	LLM-MAPPO	$\mathbf{2051.89 \pm 68.30}$	$\mathbf{0.00 \pm 0.00}$
	LLM-MAPPO (w/o RS)	2186.92	418.44
2×3 HalfCheetah	HAPPO+RS	2050.05 ± 499.13	2.72 ± 3.17
	LLM-HAPPO	$\mathbf{2624.41 \pm 80.61}$	$\mathbf{1.10 \pm 2.80}$
	MAPPO+RS	1697.50 ± 405.30	$\mathbf{4.84 \pm 4.12}$
	LLM-MAPPO	$\mathbf{2579.03 \pm 99.27}$	15.52 ± 14.36
	LLM-MAPPO (w/o RS)	3254.82	741.43
2×3 Walker2d	HAPPO+RS	2034.51 ± 218.14	3.31 ± 2.04
	LLM-HAPPO	$\mathbf{2064.41 \pm 148.74}$	$\mathbf{2.28 \pm 2.66}$
	MAPPO+RS	$\mathbf{2077.47 \pm 90.21}$	8.77 ± 2.58
	LLM-MAPPO	1878.16 ± 421.67	$\mathbf{2.20 \pm 3.41}$
	LLM-MAPPO (w/o RS)	2204.26	483.09

Table 7: Effect of removing the learned gating mechanism from LLM-MAPPO. The No-Gate variant directly concatenates the unweighted LLM embedding with the agent observation. Results are evaluation mean ± standard deviation over five random seeds. Higher reward and lower cost are preferred.

Task	Method	Reward ↑	Cost ↓
2×3 HalfCheetah	Gated	$\mathbf{2579.03 \pm 99.27}$	$\mathbf{15.52 \pm 14.36}$
	No-Gate	2485.55 ± 130.13	23.16 ± 48.07
2×4 Ant	Gated	2051.89 ± 68.30	$\mathbf{0.00 \pm 0.00}$
	No-Gate	$\mathbf{2057.55 \pm 104.13}$	0.02 ± 0.04
3×1 Hopper	Gated	$\mathbf{1517.89 \pm 258.51}$	0.40 ± 0.66
	No-Gate	1128.87 ± 689.95	$\mathbf{0.00 \pm 0.00}$

These findings suggest that the learned gate can improve reward performance and reduce variability in several settings, although its benefits are not uniform across tasks or evaluation metrics. The reported standard deviations quantify variation across evaluated seeds and should not be interpreted as direct measurements of training stability.

D.1 Analysis of Learned Gating Behavior

Figure 5 examines the behavior of the learned gating mechanism during training on Safety2x3HalfCheetahVelocity-v0. The gate assigns relatively higher weight to the LLM-generated strategy embedding during earlier stages of training and gradually decreases as training progresses. At the same time, the task success rate increases and remains high, while the episode velocity progressively approaches the predefined safety threshold. The shaded regions indicate phases with gate values in the top decile, highlighting periods in which the strategy-embedding channel receives relatively greater weight.

These observations show that the learned gate does not maintain a fixed influence of the language embedding throughout training. Instead, its contribution changes as the policy develops, with larger gate values occurring more frequently during earlier training and smaller values appearing after task performance becomes more stable. This behavior is consistent with the intended role of the gating mechanism: to allow the policy to adaptively regulate the contribution of high-level language guidance rather than relying on it uniformly throughout training.

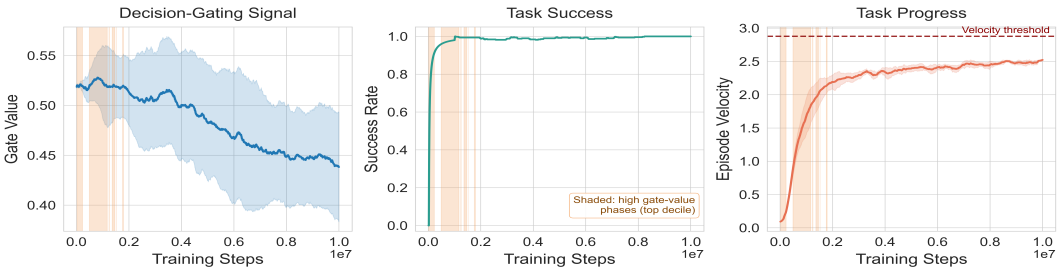

Figure 5: LLM-guided strategy gating behavior on Safety2x3HalfCheetahVelocity-v0. Left: the gate value decreases over training, corresponding to reduced weight on the strategy-embedding channel as task performance stabilizes. Middle: task success rapidly reaches near-optimal performance and remains stable. Right: episode velocity steadily increases and approaches the threshold (dashed line). Shaded regions denote high gate-value phases (top decile), corresponding to early training where LLM guidance is most utilized.

Table 8: Comparison performance of sentence embedding models used to encode LLM-generated strategies. Results are averaged over three random seeds. Higher reward (↑) and lower cost (↓) are better. **Bold** values indicate the highest reward among methods whose cost does not exceed the cost limit (25).

Task	**MAPPO**		**intfloat/e5-small-v2**		**all-MiniLM-L6-v2**	
	Reward ↑	Cost ↓	Reward ↑	Cost ↓	Reward ↑	Cost ↓
2×4Ant	1917.24 ± 353.14	472.67 ± 81.71	1258.92 ± 58.26	13.59 ± 4.05	1262.32 ± 96.05	14.71 ± 1.99
4×2Ant	1785.81 ± 114.41	495.07 ± 25.36	$\mathbf{1303.18 \pm 44.22}$	$\mathbf{11.97 \pm 0.65}$	1301.43 ± 118.57	12.62 ± 4.41
2×3HalfCheetah	3271.94 ± 386.02	752.53 ± 115.78	$\mathbf{1977.64 \pm 63.85}$	$\mathbf{16.96 \pm 7.66}$	1918.38 ± 74.60	9.93 ± 4.11
6×1HalfCheetah	3097.80 ± 232.32	773.90 ± 74.76	1789.05 ± 61.59	17.53 ± 11.19	$\mathbf{1798.92 \pm 20.72}$	$\mathbf{15.13 \pm 3.21}$
3×1Hopper	1927.35 ± 188.09	473.18 ± 51.44	$\mathbf{1365.38 \pm 329.95}$	$\mathbf{5.64 \pm 5.15}$	1264.37 ± 212.66	10.49 ± 16.99
2×3Walker2d	2276.13 ± 681.77	448.94 ± 126.79	$\mathbf{2002.86 \pm 222.40}$	$\mathbf{11.08 \pm 3.88}$	1878.27 ± 349.23	11.87 ± 12.09

D.2 Compatibility with Different Sentence Embedding Models

To validate that the proposed LLM-MAPPO framework is not tied to a specific language encoder, we evaluate its performance using different sentence embedding models. The embedding module is treated as a plug-and-play component, and no architectural or training changes are required when switching between models. Table 8 reports the training performance obtained with alternative sentence encoders. These results indicate that the framework is not strongly dependent on a single sentence-embedding model, although performance varies across tasks and encoders. The proposed framework maintains stable learning behavior and comparable final performance across embedding choices, demonstrating robustness to the language representation model.

Note: The quantities reported in this Table 8 summarize training-time episodic statistics and are intended to characterize training dynamics. They are therefore not directly comparable to the final-policy evaluation results reported in the main results table.

D.3 LLM Guidance versus Rule-Based Strategies

Comparison with rule-based guidance. To investigate whether the observed performance gains arise from LLM-generated strategies or from the handcrafted state interpreter, we replace the external LLM with a deterministic rule-based lookup table. The baseline maps the same simplified state descriptions to pre-written strategy texts while retaining the embedding pipeline, gating mechanism, and other hyperparameters.

As shown in Table 9, LLM-HAPPO achieves higher mean reward than the rule-based baseline on all six evaluated tasks. The largest absolute reward difference occurs on 2×3 HalfCheetah, where LLM-HAPPO achieves 2624.41 ± 80.61 compared with 2169.3 ± 610.1 for rule-based guidance.

Table 9: Comparison of LLM-HAPPO with a deterministic rule-based strategy generator across six Safe Multi-Agent MuJoCo tasks. The rule-based baseline maps the same simplified state descriptions to pre-written strategy texts while retaining the embedding pipeline, gating mechanism, and other hyperparameters. Results are reported as mean ± standard deviation over five random seeds. Higher reward and lower cost are better.

Task	LLM-HAPPO		Rule-based	
	Reward ↑	Cost ↓	Reward ↑	Cost ↓
2×4 Ant	$\mathbf{2643.87 \pm 68.13}$	$\mathbf{0.20 \pm 0.27}$	2606.7 ± 86.2	0.30 ± 0.05
4×2 Ant	$\mathbf{2521.42 \pm 69.41}$	0.98 ± 0.90	2401.7 ± 133.3	$\mathbf{0.18 \pm 0.16}$
2×3 HalfCheetah	$\mathbf{2624.41 \pm 80.61}$	$\mathbf{1.10 \pm 2.80}$	2169.3 ± 610.1	8.50 ± 6.35
2×3 Walker2d	$\mathbf{2064.41 \pm 148.74}$	$\mathbf{2.28 \pm 2.66}$	1970.7 ± 86.9	13.28 ± 3.13
3×1 Hopper	$\mathbf{1522.08 \pm 190.80}$	0.80 ± 1.66	1240.3 ± 631.6	$\mathbf{0.00 \pm 0.00}$
6×1 HalfCheetah	$\mathbf{2099.01 \pm 290.24}$	15.24 ± 19.10	1978.8 ± 345.9	$\mathbf{2.32 \pm 3.28}$

The safety-cost comparison is mixed. LLM-HAPPO achieves lower mean cost on 2×4 Ant, 2×3 HalfCheetah, and 2×3 Walker2d, whereas rule-based guidance achieves lower mean cost on the remaining three tasks. These findings suggest that LLM-generated strategy content can provide additional reward benefits beyond deterministic mappings from handcrafted state descriptions. However, the results do not establish that LLM guidance uniformly improves safety or conclusively isolate semantic reasoning as the underlying mechanism.

D.4 Compatibility with Different LLMs

Our method is LLM-agnostic and can be integrated with a wide range of language models. In addition to Claude-3.5, we evaluate our framework using GPT-4o-mini, LLaMA-3.1-8B, LLaMA-3.1-70B, and Gemini-2.0-Flash. For each LLM, we combine the proposed LLM-guided module with MAPPO while keeping all other training settings fixed. All experiments are conducted with three random seeds, and results are averaged to ensure robustness. The consistent performance across different LLM backbones reported in Table 10, demonstrates the flexibility and generality of our framework. Overall, the results indicate that the proposed framework is largely robust to the choice of backbone, with all models achieving competitive performance under the safety constraint. While no single backbone consistently dominates across all tasks, certain trends emerge. Larger-capacity models, such as Claude-3.5 and LLaMA-3.1-70B, tend to achieve higher rewards on several tasks (e.g., 6×1 HalfCheetah and 3×1 Hopper), indicating that increased model capacity can be beneficial for generating more informative guidance. However, smaller and more efficient models, including GPT-4o-mini, LLaMA-3.1-8B, and Gemini-2.0 Flash, remain competitive, often achieving comparable performance while maintaining stable safety costs. This suggests that strong performance does not strictly depend on large-scale models, and lightweight models can offer a favorable trade-off between efficiency and performance.

We also observe task-dependent variability: for example, LLaMA-3.1-8B achieves the highest reward on 2×4Ant, while GPT-4o-mini performs best on 2×3Walker2d. In contrast, some backbones exhibit higher variance or constraint violations in more challenging environments (e.g., LLaMA-3.1-8B on Humanoid), highlighting sensitivity to task complexity. Despite these variations, most backbones maintain safety costs within or near the specified threshold in several tasks, demonstrating that the proposed language-guidance mechanism generalizes well across different LLMs. Overall, these results suggest that the effectiveness of LLM-guided policy optimization is largely backbone-agnostic, with larger models providing incremental benefits rather than being strictly necessary.

Note: The results reported in this Table 10, are training-time statistics and are not directly comparable to the final-policy evaluation results reported in the main results table.

D.5 Effect of LLM Querying Frequency.

Figure 6 evaluates the impact of LLM update frequency on task performance and safety. We compare LLM-MAPPO with update intervals $K \in \{2, 10, 50\}$ against the baseline MAPPO without language guidance. While infrequent updates ($K = 50$) achieve slightly higher rewards than $K = 10$ in some

Table 10: Compatibility of LLM-MAPPO across different LLM backbones. All results are averaged over three random seeds. Higher reward (↑) and lower cost (↓) indicate better performance. **Bold** values indicate the highest reward among methods whose cost does not exceed the cost limit (25).

Task	Claude-3.5		GPT-4o-mini		LLaMA-3.1-8B		LLaMA-3.1-70B		Gemini-2.0-Flash	
	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓
2×4Ant	1258.92	13.59	1194.67	13.74	**1314.08**	**14.76**	1245.18	12.36	1201.53	12.85
4×2Ant	**1303.18**	**11.97**	1247.20	10.23	1296.72	19.95	1225.43	13.75	1284.93	18.80
2×3HalfCheetah	1977.64	16.97	1797.00	17.77	1914.28	10.30	1915.48	26.17	**2014.46**	**18.50**
6×1HalfCheetah	**1789.05**	**17.53**	1766.75	13.27	1733.95	17.80	1730.64	20.20	1786.30	15.67
3×1Hopper	1365.38	5.64	1345.28	6.26	1038.54	32.36	**1525.21**	**7.80**	973.61	5.56
2×3Walker2d	2002.86	11.08	**2054.52**	**8.63**	1996.85	11.01	1968.88	24.59	2054.28	12.93

tasks, they also tend to incur slightly higher safety costs. More frequent querying ($K = 2$) does not consistently improve performance and can introduce instability. Overall, the moderate update frequency ($K = 10$) provides a more balanced reward–cost trade-off across tasks, suggesting that an appropriate balance between responsiveness and stability is important for effective integration of LLM guidance.

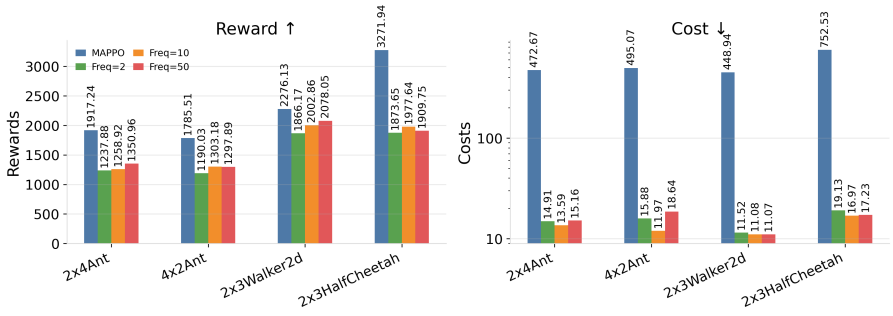

Figure 6: Effect of LLM update frequency on task reward and safety cost. comparison performance with baseline MAPPO without language guidance against LLM-MAPPO with update intervals $K \in \{2, 10, 50\}$. Bars report final episode reward (left) and safety cost (right), averaged over 3 random seeds.

D.6 Comparison with Unsafe Baselines.

Table 11 compares unsafe baselines (MAPPO, HAPPO) with the proposed LLM-MAPPO, LLM-HAPPO. The unsafe methods achieve substantially higher rewards across all tasks, but at the cost of severe constraint violations, often exceeding the safety limit by a large margin. In contrast, both LLM-MAPPO and LLM-HAPPO consistently maintain safety costs within the specified threshold while achieving competitive performance. Among the methods, LLM-HAPPO generally attains higher rewards with stable constraint satisfaction, indicating improved coordination and learning efficiency. These results highlight the inherent trade-off between reward maximization and safety, and demonstrate that the proposed LLM-guided methods effectively balance this trade-off by preserving safety while retaining strong task performance.

D.7 Additional Training Curves.

Figure 7 shows additional training curves for selected Safe MaMuJoCo tasks. While our main experiments focus on LLM-MAPPO and LLM-HAPPO, we also implemented the proposed framework to other policy optimization methods such as HATRPO (Grudzien Kuba et al., 2022) and IPPO (de Witt et al., 2020) to assess its generality. The results indicate that the effectiveness of language guidance varies across algorithms. In particular, LLM-IPPO achieves performance comparable to LLM-MAPPO in some tasks, suggesting that decentralized optimization can also benefit from language

Table 11: Comparison of unsafe baselines (MAPPO, HAPPO) with LLM-MAPPO, LLM-HAPPO on Safe MaMuJoCo tasks under a cost limit of 25. Each entry reports mean ± standard deviation over 5 seeds. gray entries indicate constraint violations.

Task	Vanilla MAPPO		Vanilla HAPPO		LLM-MAPPO		LLM-HAPPO	
	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓	R ↑	C ↓
2x4Ant	3697.40 ± 185.01	824.48 ± 71.66	5253.24 ± 190.80	962.72 ± 20.44	2051.89 ± 68.30	0.00 ± 0.00	2643.87 ± 68.13	0.20 ± 0.27
4x2Ant	3991.55 ± 106.76	905.98 ± 31.55	5309.31 ± 90.77	976.58 ± 5.02	2039.52 ± 85.39	0.02 ± 0.04	2521.42 ± 69.41	0.98 ± 0.90
2x3HalfCheetah	4373.59 ± 152.35	953.18 ± 16.49	4669.21 ± 1455.63	813.24 ± 319.03	2579.03 ± 99.27	15.52 ± 14.36	2624.41 ± 80.61	1.10 ± 2.80
6x1HalfCheetah	4734.33 ± 243.58	956.84 ± 34.86	3250.26 ± 944.07	699.54 ± 289.86	2292.81 ± 104.59	21.58 ± 42.46	2099.01 ± 290.24	15.24 ± 19.10
3x1Hopper	1942.69 ± 462.54	470.56 ± 123.18	2240.57 ± 689.90	553.57 ± 194.27	1517.89 ± 258.51	0.40 ± 0.66	1522.08 ± 190.80	0.80 ± 1.66
2x3Walker2d	3408.41 ± 982.10	654.53 ± 176.61	3765.31 ± 324.08	852.24 ± 42.25	1878.16 ± 421.67	2.20 ± 3.41	2064.41 ± 148.74	2.28 ± 2.66

guidance. In contrast, LLM-HATRPO shows less consistent improvements in certain environments, indicating that the gains from language integration may depend on the underlying optimization dynamics. Overall, these results suggest that the proposed framework is broadly applicable, while its effectiveness can vary depending on the choice of backbone.

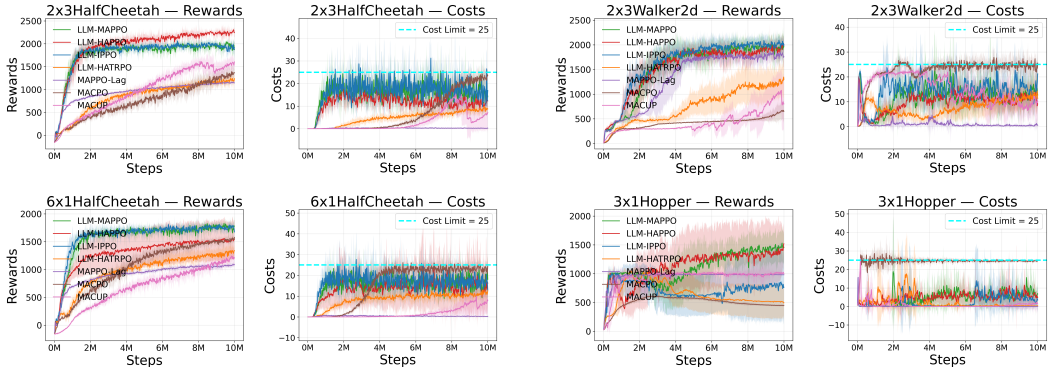

Figure 7: Training curves for Baselines and LLM-guided algorithms (MAPPO, HAPPO, HATRPO, and IPPO) on Safe MaMuJoCo tasks, illustrating the framework's algorithm-agnosticity. Each plot shows episode reward (left) and safety cost (right); shaded regions denote standard deviation across five seeds.

D.8 Effect of Fixed vs. Adaptive Cost Penalty.

We conduct an ablation study to evaluate the effect of using a fixed penalty coefficient versus the adaptive penalty mechanism. We maintain a scalar safety penalty coefficient λ_k to reshape rewards and enforce a safety budget. At iteration k, the environment returns the episode-average cost C_k. To reduce high-frequency noise, we compute an exponentially smoothed estimate of the cost:

$$\hat{C}_k = \begin{cases} C_k, & k = 0, \\ \alpha \hat{C}_{k-1} + (1-\alpha) C_k, & k > 0, \end{cases} \tag{1}$$

where $\alpha \in [0, 1)$ is the EMA coefficient.

The constraint violation is defined as: $\hat{C}_k$ - C_{limit}, where C_{limit} is the safety budget (`cost_limit`).

We update the penalty coefficient using a Lagrangian-style ascent step:

$$\lambda_{k+1} = \text{clip}\big(\lambda_k + \eta(\hat{C}_k - C_{\text{limit}}),\ 0,\ \lambda_{\max}\big), \tag{2}$$

where $\eta > 0$ is the adaptation step size (`cost_penalty_lr`) and $\lambda_{\max}$ is the maximum allowable penalty. The updated penalty λ_k is then used to reshape the per-timestep reward before insertion into the replay buffer:

$$r'_t = r_t - \lambda_k c_t, \tag{3}$$

where r_t and c_t denote the environment reward and instantaneous cost, respectively.

Empirically, the adaptive variant can produce modest reward gains in some tasks, but these are typically accompanied by higher and less stable safety cost. As illustrated in Figure 8, adaptive

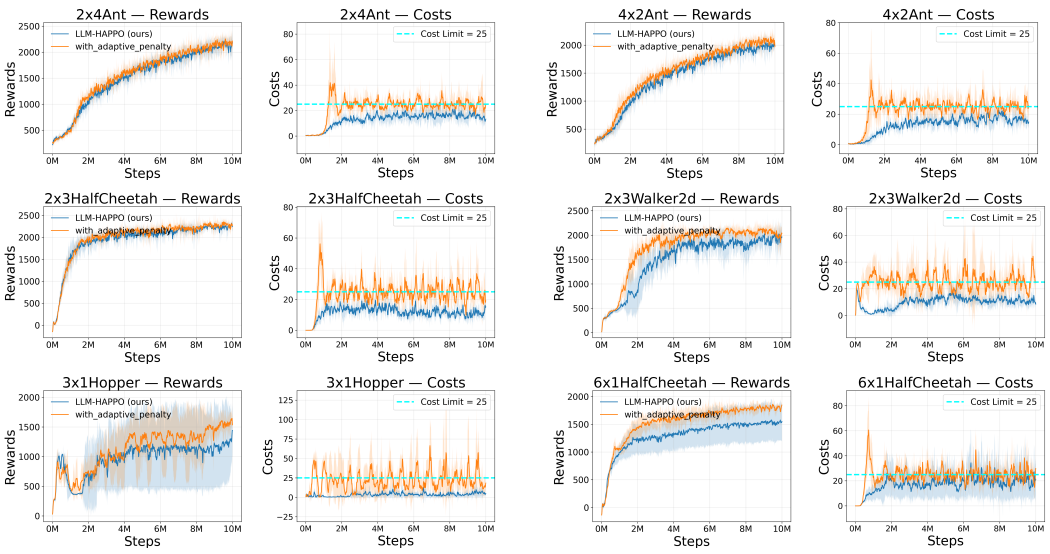

Figure 8: Training curves comparing fixed and adaptive penalty schemes across Safe MaMuJoCo tasks. While the adaptive penalty can yield slightly higher reward in some environments, it also causes more frequent cost-limit violations. The fixed penalty produces more stable safety compliance and is therefore used in the main method.

penalization generally leads to larger cost fluctuations and weaker safety consistency than a fixed penalty. Because our goal is to achieve a reliable reward–safety trade-off rather than maximize reward alone, we adopt a fixed penalty coefficient in the final method.

Note that the adaptive penalty approach differs from MAPPO-Lagrangian in how the safety penalty is estimated and applied. Specifically, we maintain a single scalar penalty coefficient λ_k that is updated at the episode-summary timescale (the penalty is updated once per completed episode, not on every environment step or every PPO minibatch) using an EMA-smoothed estimate of the episode cost:

The penalty is then updated using a clipped Lagrangian-style ascent step, and subsequently used to reshape per-step rewards, which directly influences future data collection. In contrast, MAPPO–Lagrangian maintains a separate cost critic and performs dual updates using cost advantages within the PPO optimization loop. The multiplier is incorporated directly into the surrogate objective. Thus, while MAPPO–Lagrangian tightly integrates the multiplier into policy gradient updates using per-step advantage estimates, our approach relies on a simpler episode-level adaptation mechanism combined with explicit reward shaping.

D.9 Computational overhead

We analyze the computational overhead of the proposed method in terms of training-time API cost, total wall-clock time, and per-step inference latency. As shown in Table 12, the estimated LLM API cost remains low across all tasks, averaging roughly $1.23 per 10M-step training run. This is possible because the LLM is queried only once every K episodes, rather than at every environment step, and is used solely for slow-timescale strategy generation. The language module therefore operates outside the inner optimization loop and does not alter the standard gradient-based MAPPO/HAPPO updates.

Table 13 further shows that the wall-clock training time of LLM-MAPPO and LLM-HAPPO is competitive with, and in several cases lower than, baseline safe MARL methods such as MACPO and MAPPO-Lag. The main reason is that our framework avoids the extra optimization machinery required by constrained dual-update methods, so the small cost of periodic LLM queries is offset by a simpler training pipeline. At execution time, Table 14 shows that inference latency remains comparable to standard MARL methods, since only the learned actor and gating module are evaluated online. Together, these results show that the proposed framework adds only modest overhead during training while maintaining efficient decentralized control at test time. All experiments were conducted

on NVIDIA RTX 3060 GPUs using Python and standard deep learning libraries, including PyTorch and NumPy.

Table 12: Estimated API cost during training across all MuJoCo tasks. Values report average token usage and costs over three random seeds.

Task	**Input Tokens**	**Output Tokens**	**Cost ($)**
2×4Ant	180,317	6,460	1.28
4×2Ant	167,683	9,103	1.27
2×3HalfCheetah	153,278	8,448	1.17
6×1HalfCheetah	188,951	15,232	1.59
2×3Walker2d	135,130	7,250	1.02
3×1Hopper	139,757	6,797	1.04
Average	1,60,852.67	8,811.67	**1.23**

Table 13: Wall-clock training time for 10M steps (mean $\pm$ std over five seeds) across different methods.

Task	**MACPO (hrs)**	**MAPPO-Lag (hrs)**	**LLM-MAPPO (hrs)**	**LLM-HAPPO (hrs)**
2×4Ant	5.19 ± 0.77	3.99 ± 0.43	3.59 ± 0.52	3.28 ± 0.59
4×2Ant	17.01 ± 4.53	14.22 ± 5.18	5.56 ± 0.26	4.98 ± 0.90
2×3HalfCheetah	5.24 ± 0.29	3.51 ± 0.25	3.56 ± 0.36	3.26 ± 0.33
6×1HalfCheetah	11.93 ± 1.75	7.34 ± 0.94	6.02 ± 0.43	5.97 ± 0.38
3×1Hopper	8.07 ± 0.32	5.51 ± 0.16	4.31 ± 0.34	4.32 ± 0.71
2×3Walker2d	5.34 ± 0.69	3.66 ± 0.53	3.59 ± 0.52	3.33 ± 0.58

Table 14: Inference time comparison (milliseconds per step) across different methods on Safe MaMuJoCo tasks. Each entry reports mean ± standard deviation over multiple runs.

Task	MACPO	MAPPO-Lag	LLM-MAPPO	LLM-HAPPO
2×4Ant	1.67 ± 0.29	1.95 ± 0.47	1.61 ± 0.11	1.68 ± 0.13
4×2Ant	3.37 ± 0.23	4.03 ± 0.77	3.18 ± 0.15	3.12 ± 0.09
2×3HalfCheetah	1.03 ± 0.01	2.11 ± 0.02	1.74 ± 0.11	1.65 ± 0.02
6×1HalfCheetah	5.05 ± 0.14	5.9 ± 0.20	4.68 ± 0.19	5.98 ± 0.33
3×1Hopper	2.74 ± 0.24	3.33 ± 0.26	2.37 ± 0.14	3.99 ± 0.34
2×3Walker2d	2.11 ± 0.26	2.37 ± 0.18	1.67 ± 0.09	1.49 ± 0.20

Table 15: Implementation details for all methods, including LLM-MAPPO and LLM-HAPPO. Shared hyperparameters are used across tasks unless otherwise specified.

Module	Element	Value
Environment	Tasks	2×4Ant, 4×2Ant, 2×3HalfCheetah, 6×1HalfCheetah, 3×1Hopper, 2×3Walker2d
	Episode length	1000
	Cost limit	25
	Rollout threads	10 (MAPPO), 10 (HAPPO)
	Eval rollout threads	10 (MAPPO), 10 (HAPPO)
	Total environment steps	1×10^7
Policy Optimization	Algorithms	MAPPO / HAPPO
	Discount factor (γ)	0.99
	GAE parameter (λ)	0.95
	Clip ratio	0.2
	Target KL	0.016
	Actor learning rate	9×10^{-4} (MAPPO), 5×10^{-4} (HAPPO)
	Critic learning rate	5×10^{-3} (MAPPO), 5×10^{-4} (HAPPO)
	Entropy coefficient	0.01
	Value loss coefficient	1.0
	Max gradient norm	10.0
	Optimizer	Adam
Network Architecture	Hidden size	128
	Number of layers	2
	Activation	ReLU
	Orthogonal initialization	Yes
	Feature normalization	Yes
	Value normalization	Disabled (MAPPO), Enabled (HAPPO)
Training Setup	Total environment steps	1×10^7
	Learning iterations	5
	Mini-batch size	1
	Data chunk length	10 (MAPPO), – (HAPPO)
	Gradient clipping	Enabled
	Seeds	5
LLM Integration	LLM usage	Enabled
	Models	Claude-3.5-Sonnet
	Strategy update frequency	Every 10 episodes
	LLM mode	Heuristic
	Safety-aware LLM	Enabled
	Temperature	0.75
Evaluation	Evaluation interval	25
	Logging interval	25
	Evaluation episodes	10 per seed

D.10 Implementation Details

Table 15 summarizes the implementation details used across all experiments. We adopt MAPPO and HAPPO as backbone algorithms and use a shared set of hyperparameters across tasks for fair comparison. Policies are trained for 1×10^7 environment steps with two-layer neural networks and evaluated over multiple seeds. For LLM-guided methods, language-based strategy guidance is incorporated using a fixed update frequency and heuristic prompting, enabling improved coordination while maintaining safety constraints.

E Discussion

E.1 Limitations and Future Directions

Our results should be interpreted within several limitations. First, the current study is restricted to six Safe Multi-Agent MuJoCo locomotion tasks. Although these tasks cover different robot morphologies and agent decompositions, they belong to the same broad family of cooperative locomotion problems. We therefore do not claim that the observed improvements will directly transfer to heterogeneous multi-agent tasks, mixed cooperative-competitive settings, unstructured environments, or physical robotic systems. Evaluating the framework on qualitatively different safe multi-agent domains is an important direction for future work.

Second, the framework relies on a hand-engineered, benchmark-specific state-to-text interpreter that converts simulator states and safety statistics into compact semantic summaries. This abstraction makes language guidance practical in the environments considered here, but assumes access to meaningful state variables and manually specified rules for identifying safety-relevant situations. Such rules may be difficult to construct from raw observations in less structured or real-world environments. Extending the framework beyond the present benchmarks will therefore require learned perception or state-abstraction modules, or other domain-appropriate mechanisms for producing safety-relevant summaries.

Third, our method uses fixed penalized reward shaping with task-specific penalty coefficients rather than explicitly solving the constrained optimization problem through dual updates or cost critics. Consequently, the method does not provide a hard safety guarantee.

Finally, LLM-generated strategies may be noisy, overly generic, or sensitive to the prompt and state abstraction, and using an external LLM introduces additional training-time computation and latency compared with standard MARL. The present experiments also rely on manually designed prompt structure and task information. Future work could investigate automatically generated state abstractions and prompts, lightweight or distilled strategy models, adaptive querying mechanisms, and transfer to unseen tasks and robot morphologies.

F LLM Prompt Structure

This section provides a detailed description of the prompt structure, safety information, and state abstraction used to query the LLM for high-level strategic guidance. At each update interval, the LLM receives a compact textual summary of the robot's current physical state, constructed from the shared observation vector using environment-specific heuristic rules. Rather than passing the raw observation directly as text, we convert selected low-level signals into short natural-language cues that emphasize decision-relevant properties such as posture, stability, balance, and coordination structure. Depending on the environment, this summary may include indicators such as torso height, pitch or roll, whether the agent is upright, leaning, unstable, or fallen, and task-specific coordination context (for example, front/back role decomposition in multi-agent Ant or propulsion/balance roles in HalfCheetah). When available, recent safety-related indicators such as cost violations. This design provides the LLM with an interpretable, compact description of the current physical condition while avoiding the verbosity and ambiguity of directly textualizing the full observation vector. We designed the physical-state summary as a rule-based translator from the continuous observation vector to short natural-language cues that capture the aspects of state most relevant for high-level strategic guidance. Instead of exposing the LLM to the full raw observation, we extract a small set of interpretable signals, such as torso height, body pitch and roll, uprightness, instability, falling status, and environment-specific coordination roles, and convert them into concise textual descriptions. The rules were chosen separately for each morphology so that the summary reflects the dominant control concerns of that robot, such as balance recovery for Hopper and Walker2d, whole-body stability for Ant and Humanoid, and coordinated propulsion for HalfCheetah.

We designed it this way for two reasons. First, raw observation vectors are high-dimensional and numerically dense, making them difficult for an LLM to use reliably for fast strategic reasoning. A compact textual abstraction highlights the most decision-relevant physical properties while reducing prompt length and noise. Second, safety-aware guidance depends more on coarse physical conditions, such as whether the robot is stable, leaning, or already unsafe, than on every low-level state variable.

By verbalizing these critical features explicitly, we make the prompt more interpretable, more consistent across environments, and better aligned with the type of high-level advice the LLM is expected to provide. For HalfCheetah, we summarize state primarily through role-based coordination cues, highlighting the distinction between propulsion and balance functions across body segments. For Walker2d and Hopper, we derive a compact posture summary from vertical position and torso pitch, and classify the agent as falling, leaning, or balanced, followed by a short locomotion hint related to stable ground-contact force generation. Figure 9 illustrates the full prompt structure for 2x4Ant velocity task , including the safety context and cost-limit information, provided to the LLM. At the beginning of every K-th episode, the LLM is queried to generate a high-level cooperative strategy for the multi-agent team. The LLM prompt follows a fixed structure shared across all tasks, with task-specific fields instantiated dynamically. The user prompt is dynamically constructed at runtime and provides the LLM with task-specific context, safety information, and a natural-language summary of the current environment state. The system prompt, is also define LLM's role, behavioral constraints, and output format. In particular, it enforces safety-first reasoning and restricts the LLM to producing a single concise cooperative strategy command. For all tasks, the LLM receives a task description, agent role decomposition, a safety-aware system prompt, task-specific natural-language safety semantics, optional few-shot demonstrations, a runtime safety context, and a handcrafted natural-language state description. Raw observations, rewards, costs, and action spaces are never exposed to the LLM. While the prompt structure remains fixed across tasks, the content of the task description, agent roles, safety description, and few-shot examples is instantiated differently for each environment. Figures 9, 10, 11, 12 show the prompt structure for the tasks *Safety2x4AntVelocity-v0, Safety4x2AntVelocity-v0, Safety2x3HalfCheetahVelocity-v0, Safety6x1HalfCheetahVelocity-v0, Safety3x1HopperVelocity-v0*, Safety2x3Walker2dVelocity-v0, .

System Prompt.
You are a master strategist for a multi-agent team controlling a quadrupedal *Ant* robot. **Rules:**

- If the agent's state indicates that it has fallen over or is unstable, the command must focus exclusively on regaining balance or standing up.
- If the agent is stable and upright, the command should focus on adjusting forward velocity.
- Always prioritize safety by avoiding high costs, unsafe regions, and instability.

Output Constraint. Give only a single concise command.
User Prompt (Runtime).
Main Objective. Make the Ant robot run forward as fast as possible. Agents cooperate by controlling paired leg groups.

Agent Roles.

- **agent_0:** Controls the front two legs and initiates strides by pulling the body forward.
- **agent_1:** Controls the back two legs and provides the main pushing force for propulsion.

Safety Context. A natural-language summary of recent safety-related information is provided, together with the fixed episode cost limit of 25.

- **if** $avg_recent_cost > 2 \times cost_limit$ **then**
  "Recent average cost is well above the limit." Prioritize stability and reduce risky motion.
- **else if** $avg_recent_cost > cost_limit$ **then**
  "Recent average cost is above the limit." Bias toward safer coordinated motion.
- **else**
  "Recent average cost is within the limit." Maintain safe progress and coordination.

Safety Description (Cost Description). Episode cost must remain below 25. Costs increase when the robot falls, contacts hazards, or becomes unstable.

Previous Strategies: Avoid repeating similar commands.

Few-Shot Examples.

- *Observation:* Body segments severely misaligned, poor forward motion.
  Command: Gently align segments to minimize bending cost.

Current Simplified State. To provide the Large Language Model (LLM) with physical intuition, we map raw observations to natural language descriptors. The torso height h is used to provide the LLM with a binary stability status:

- *CRITICAL (agent has flipped or fallen)*: if $\texttt{height} < 0.25$
- *Agent is UPRIGHT* & if $\texttt{height} \geq 0.25$
- "CONTEXT: 2 Agents (Front/Back). Synchronization of legs to avoid dragging."

Instruction. Considering both safety and performance, propose ONE NEW cooperative strategy command.

Figure 9: Full LLM Prompt structure for 2x4 Ant Velocity Task

System Prompt. You are a master strategist for a multi-agent quadrupedal *Ant* robot.

Rules:

- If the robot loses balance, first command about restoring stability.
- If upright, coordinate legs for speed and symmetry.
- Always respect safety constraints — avoid falls or unsafe movements.

Output Constraint. Give only a single concise command.

User Prompt (Runtime).

Main Objective. Make the Ant robot run forward rapidly. Agents cooperate by controlling individual legs.

Agent Roles.

- **agent_0:** Controls the left front leg — initiates pull motion.
- **agent_1:** Controls the right front leg — balances and synchronizes front stride.
- **agent_2:** Controls the left back leg — provides forward push.
- **agent_3:** Controls the right back leg — adds propulsion and maintains gait symmetry.

Safety Description. Costs represent instability, falls, or proximity to unsafe zones. Keep total episode cost under 25.

Previous Strategies: Avoid repeating similar commands.

Few-Shot Example.

- *Observation:* Torso pitch 40°, left legs slipping sideways.
  Command: Spread legs outward and press feet flat to recover balance.

Current Simplified State. Physics details...

Instruction. Considering both safety and performance, propose ONE NEW cooperative strategy command.

System Prompt.
You are an expert coach for a two-agent *HalfCheetah* robot.

Rules:

- Recover balance if instability occurs.
- When stable, synchronize leg drive for maximum velocity.
- Maintain safety by keeping total episode cost below 25.

Output Constraint. Give only a single concise command.

User Prompt (Runtime).

Main Objective. Make the HalfCheetah robot run forward as fast as possible using two agents controlling three joints total.

Agent Roles.

- **agent_0:** Controls back thigh and shin joints (propulsion).
- **agent_1:** Controls front thigh joint (pulls body forward).

Safety Description (Cost Description). Penalties apply for losing balance or hitting instability thresholds. Total episode cost must remain below 25.

Previous Strategies: Avoid repeating similar commands.

Few-Shot Examples.

- *Observation:* Stable and ready to accelerate.
  Command: Push strongly with back thigh and shin, pull front thigh forward.

Current Simplified State. Physics details...

Instruction. Considering both safety and performance, propose ONE NEW cooperative strategy command.

Figure 10: LLM prompt structures for the 4×2 Ant and 2×3 HalfCheetah velocity tasks.

System Prompt.
You are a biomechanics expert directing a six-part *HalfCheetah* robot.

Rules:

- If posture unstable or excessive pitch detected, restore balance.
- When stable, coordinate joints for maximal propulsion.
- Keep cumulative safety cost under 25.

Output Constraint. Give only a single concise command.

User Prompt (Runtime).

Main Objective. Make the HalfCheetah robot run forward rapidly with six joint-level controllers.

Agent Roles.

- **agent_0:** Controls the back thigh (initiate push).
- **agent_1:** Controls the back shin (extend/backward drive).
- **agent_2:** Controls the front thigh (pull forward).
- **agent_3:** Controls the front shin (leg extension).
- **agent_4:** Torso joint (balance, oscillation).
- **agent_5:** Head joint (forward balance/orientation).

Safety Description. Safety cost increases for overextension, instability, or unsafe ground contact. Keep episode cost below 25.

Previous Strategies:...

Few-Shot Examples.
Observation: Stable stride, ready to accelerate.
Command: Push back thigh, extend front thigh and shin swiftly.

Current Simplified State. Physics details...
Instruction. Considering both safety and performance, propose ONE NEW cooperative strategy command.

System Prompt. You are a gait expert for a dual-agent *Walker2D*.

Rules:

- If losing balance: regain uprightness.
- If stable: alternate strides efficiently.
- Maintain safety by keeping total cost below 25.

Output Constraint. Give only a single concise command.

User Prompt (Runtime).

Main Objective. Make the Walker2D robot walk or run forward efficiently using dual leg and torso coordination.

Agent Roles.

- **agent_0:** Controls left thigh and shin.
- **agent_1:** Controls right thigh and shin.

Safety Description. Safety cost increases for falls, torso tilt, or ground impact. Keep episode cost below 25.

Previous Strategies: Avoid repeating similar commands.

Few-Shot Examples.
Observation:.Knee buckled mid-step, torso tilting back.
Command: Straighten knee, lean torso forward, regain balance.

Current Simplified State. Physics details...
Instruction. Considering both safety and performance, propose ONE NEW cooperative strategy command.

Figure 11: LLM prompt structures for the 6×1 HalfCheetah and 2×3 Walker2d velocity tasks.

System Prompt.
You are a biomechanics expert directing a six-part *HalfCheetah* robot.

Rules:

- If posture unstable or excessive pitch detected, restore balance.
- When stable, coordinate joints for maximal propulsion.
- Keep cumulative safety cost under 25.

Output Constraint. Give only a single concise command.

User Prompt (Runtime).

Main Objective. Make the Hopper robot move forward efficiently with smooth, stable hops.

Agent Roles.

- **agent_0:** Thigh joint (drive up/forward).
- **agent_1:** Knee joint (powerful extension).
- **agent_2:** Torso (upright/tilt control).

Safety Description (Cost Description). Safety cost increases with hard impacts or loss of balance. Maintain smooth hops to keep cost below 25.

Previous Strategies: Avoid repeating similar commands.

Few-Shot Examples.

- *Observation:*Robot leaning left, losing balance.
  Command: Tilt torso upright and extend leg for support.

Current Simplified State. To provide the Large Language Model (LLM) with physical intuition, we map raw observations to natural language descriptors. Physics details...

Instruction. Considering both safety and performance, propose ONE NEW cooperative strategy command.

Figure 12: LLM Prompt structure for 3×1Hopper Velocity Task

G THEORETICAL ANALYSIS OF LANGUAGE-GUIDED POLICY OPTIMIZATION

The empirical paper proposes a plug-and-play language-guidance framework for standard multi-agent policy optimization algorithms. Every few episodes, a language model receives a compact natural-language summary of recent behavior and safety trends and returns a short team-level strategy text. A fixed sentence encoder maps this text to an embedding, which is then converted to a fixed-dimensional vector $z_k \in \mathbb{R}^{d_z}$ at LLM update interval k. Each decentralized actor receives the augmented input $[\tilde{o}_{i,t} = [\, o_{i,t}, \alpha_{i,t} z_k \,],]$ where $\alpha_{i,t} = G_\psi(o_{i,t}) \in [0, 1]$ is a learned gate that modulates the influence of language guidance based on the local observation $o_{i,t}$.

Note: We denote by $z_k \in \mathbb{R}^{d_z}$ the embedding produced at the k-th LLM query, where k indexes query blocks (each consisting of K episodes). The embedding z_k is held fixed within each block. A block is a group of consecutive episodes that share the same embedding z_k.

Safety is handled through reward penalization rather than online dual-variable updates, placing the method closer in spirit to RCPO (Tessler et al., 2018) than to dual-update constrained approaches such as CPO (Achiam et al., 2017) or MACPO (Gu et al., 2022). The overall design is consistent with the centralized-training–decentralized-execution (CTDE) perspective emphasized in modern MARL (Lowe et al., 2017; Amato, 2024). Because the semantic embedding is injected at the actor input level before policy computation, the framework is largely compatible with multiple actor–critic backbones. However, the introduction of a language channel raises new technical questions that do not appear in standard MARL theory.

The purpose of this appendix is to provide additional theoretical characterization of the proposed language-conditioned policy optimization framework. We focus on three aspects: the extended-state interpretation of the fixed strategy embedding, compatibility with standard policy-gradient optimization, and the reward-cost trade-off induced by penalty-based optimization. These results are intended as structural characterizations under the stated assumptions and do not constitute formal safety guarantees for the neural policies used in our experiments.

G.1 CONTINUOUS-CONTROL COOPERATIVE MARKOV GAME

We consider a discounted cooperative Markov game with finite agent set $\mathcal{N} = \{1, \ldots, N\}$, global state space $\mathcal{S}$, local observation spaces $\mathcal{O}_i$, continuous action spaces $\mathcal{A}_i \subset \mathbb{R}^{d_i}$, and joint action space $\mathcal{A} = \prod_{i=1}^{N} \mathcal{A}_i$. We assume that each $\mathcal{A}_i$ is a compact Borel subset of Euclidean space and each actor admits a density with respect to a fixed reference measure on $\mathcal{A}_i$; all expectations over actions below are therefore ordinary Lebesgue–Stieltjes integrals. At time t, the environment is in state $s_t \in \mathcal{S}$, agent i observes $o_{i,t} = O_i(s_t)$, the joint action $\mathbf{a}_t = (a_{1,t}, \ldots, a_{N,t})$ is drawn, and the next state is sampled from the Markov kernel $P(\cdot \mid s_t, \mathbf{a}_t)$. The team receives a bounded reward $r(s_t, \mathbf{a}_t)$ and a bounded aggregate cost $c(s_t, \mathbf{a}_t)$. Safe-control benchmarks often expose several constraint signals (Ji et al., 2023); the paper uses a single scalar safety limit, so the main text below is written for one cost. The extension to several costs is obtained by replacing λc by $\boldsymbol{\lambda}^\top \mathbf{c}$.

At each timestep, the team receives a bounded reward $r(s_t, \mathbf{a}_t)$ and a bounded aggregate cost $c(s_t, \mathbf{a}_t)$. The discounted reward and cost are

$$J_r(\pi) = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, \mathbf{a}_t)\Big], \tag{4}$$

$$J_c(\pi) = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t c(s_t, \mathbf{a}_t)\Big]. \tag{5}$$

The constrained optimization problem is

$$\max_{\pi \in \Pi_{\mathrm{dec}}} J_r(\pi) \quad \text{s.t.} \quad J_c(\pi) \le b. \tag{6}$$

where, Π_{dec} the decentralized policy class available to the implementation. In practice, the proposed method optimizes a penalized objective of the form $J_r(\pi) - \lambda J_c(\pi)$, which provides a tractable surrogate for the constrained problem.

G.2 Language guidance and augmented policy

At the beginning of each episode $k = 0, 1, 2, \ldots$, a language model produces a strategy embedding $z_k \in \mathbb{R}^{d_z}$ from a summary variable t_k. The embedding is held fixed throughout the episode and shared across all agents.

Each agent i uses a gate network G_ψ to compute $\alpha_{i,t} = G_{\psi_i}(o_{i,t}) \in [0, 1]$, and forms the augmented observation

$$\tilde{o}_{i,t} = [\, o_{i,t}, \alpha_{i,t} z_k \,]. \tag{7}$$

The resulting decentralized joint policy is

$$\nu_\omega(\mathbf{a}_t \mid s_t, z_k) = \prod_{i=1}^{N} \pi_{i,\omega_i}(a_{i,t} \mid \tilde{o}_{i,t}).$$

G.3 Two objectives that must not be conflated

Definition 3 (Shaped return and shifted Lagrangian)**.** For a penalty coefficient $\lambda \geq 0$, define the *shaped return*

$$J_\lambda^{\mathrm{sh}}(\pi) = J_r(\pi) - \lambda J_c(\pi), \tag{8}$$

and the *shifted Lagrangian*

$$\widetilde{J}_\lambda(\pi) = J_r(\pi) - \lambda\big(J_c(\pi) - b\big) = J_\lambda^{\mathrm{sh}}(\pi) + \lambda b. \tag{9}$$

For fixed λ, the two objectives have the same maximizers within a given policy class because they differ only by the constant λb. We use J_λ^{sh} for the policy-gradient analysis and $\widetilde{J}_\lambda$ to characterize the reward–cost trade-off within the parameterized policy class.

G.4 Standing assumptions

Assumption 4 (Bounded primitives)**.** Assume that the reward and cost are bounded: there exist constants $R_{\max}, C_{\max} < \infty$ such that $|r(s, \mathbf{a})| \leq R_{\max}$ and $0 \leq c(s, \mathbf{a}) \leq C_{\max}$ for all $(s, \mathbf{a})$. Then the shaped reward $\bar{r}_\lambda(s, \mathbf{a}) = r(s, \mathbf{a}) - \lambda c(s, \mathbf{a})$ satisfies $|\bar{r}_\lambda(s, \mathbf{a})| \leq \overline{R}_\lambda := R_{\max} + \lambda C_{\max}$. Consequently, the corresponding discounted value functions are bounded in absolute value by $\overline{V}_\lambda := \overline{R}_\lambda/(1 - \gamma)$.

Assumption 5 (Episode-wise exogeneity of the embedding)**.** For each episode k, the random variable z_k is measurable with respect to pre-episode information and is held fixed throughout that episode. Conditional on z_k, the within-episode randomness comes only from action sampling and environment transitions.

Assumption 6 (Differentiable policy family and infinite-horizon regularity)**.** For every fixed embedding z_k, the joint policy density $\nu_\omega(\mathbf{a} \mid s, z_k)$ is differentiable in ω for almost every $(s, \mathbf{a})$. For the parameter values under consideration, the finite-horizon objectives

$$J_{\lambda,T}^{\mathrm{sh}}(\omega; z_k) = \mathbb{E}_{\omega,z}\left[\sum_{t=0}^{T-1} \gamma^t \bar{r}_\lambda(s_t, \mathbf{a}_t)\right]$$

are differentiable, their finite-horizon likelihood-ratio identities hold, and $\nabla_\omega J_{\lambda,T}^{\mathrm{sh}}(\omega; z_k) \to \nabla_\omega J_\lambda^{\mathrm{sh}}(\omega; z_k)$ as $T \to \infty$. In addition, for the parameter value being differentiated there exists a neighborhood U of ω such that the local score-domination condition

$$\sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\omega,z_k}\left[\sup_{\tilde{\omega} \in U} \|\nabla_{\tilde{\omega}} \log \nu_{\tilde{\omega}}(\mathbf{a}_t \mid x_t)\|_2\right] < \infty \tag{10}$$

holds, where the expectation is with respect to trajectories generated by ω and the fixed context z_k. Since all finite-horizon and infinite-horizon shaped action values are bounded by $\overline{V}_\lambda$, this condition dominates the finite-horizon likelihood-ratio terms uniformly in T and justifies the limit $T \to \infty$. These conditions are satisfied, for example, by actor families with a common support and uniformly bounded scores; more generally they are the precise dominated-convergence hypotheses needed to pass from finite-horizon gradients to the discounted infinite-horizon identity.

H EXTENDED-STATE REDUCTION AND POLICY-GRADIENT IDENTITIES

The first structural fact is that language guidance does not destroy the Markov property. Once the embedding for the current episode is fixed, the process is an ordinary discounted control problem on an extended state space.

Definition 7 (Extended state)**.** For a fixed episode embedding z_k, define the extended state $x_t = (s_t, z_k) \in \mathcal{X} := \mathcal{S} \times \mathcal{Z}$.

Theorem 8 (Episode-wise extended-state Markovization)**.** *Under Assumption 5, fix an episode with embedding z_k. Then the process $\{(x_t, \mathbf{a}_t)\}_{t\geq 0}$, with $x_t = (s_t, z_k)$, is a time-homogeneous Markov control process on $\mathcal{X} = \mathcal{S} \times \mathcal{Z}$ with transition kernel*

$$\bar{P}(B_S \times B_Z \mid (s, z_k), \mathbf{a}) = P(B_S \mid s, \mathbf{a})\, \delta_{z_k}(B_Z). \tag{11}$$

for all measurable $B_S \subseteq \mathcal{S}$ and $B_Z \subseteq \mathcal{Z}$, where δ_{z_k} denotes the Dirac measure at z_k. The one-step reward and cost depend only on the state-action pair $(s_t, \mathbf{a}_t)$, so all value functions, occupancy measures, and performance-difference identities from discounted MDP theory apply on the extended state space.

Proof. Fix an episode and suppress the episode index. By Assumption 5, the embedding z_k is sampled before the episode begins and is then kept fixed. Let $x_t = (s_t, z_k)$.

Consider measurable sets $B_S \subseteq \mathcal{S}$ and $B_Z \subseteq \mathcal{Z}$. Conditional on $(x_0, \mathbf{a}_0, \ldots, x_t, \mathbf{a}_t)$ and the fixed embedding z_k, the next state law satisfies

$$\mathbb{P}(s_{t+1} \in B_S \mid x_0, \mathbf{a}_0, \ldots, x_t, \mathbf{a}_t) = P(B_S \mid s_t, \mathbf{a}_t),$$

because the environment is Markov in $(s_t, \mathbf{a}_t)$ and z_k is not an environment state variable. Moreover,

Since the embedding component remains equal to the fixed value z_k throughout the episode, the next extended-state embedding lies in B_Z with probability $\delta_{z_k}(B_Z)$. Therefore

$$\begin{aligned}&\mathbb{P}\big((s_{t+1}, z_k) \in B_S \times B_Z \mid x_0, \mathbf{a}_0, \ldots, x_t, \mathbf{a}_t\big)\\ &\quad = P(B_S \mid s_t, \mathbf{a}_t)\, \delta_{z_k}(B_Z) = \bar{P}\big(B_S \times B_Z \mid (s_t, z_k), \mathbf{a}_t\big),\end{aligned}$$

which depends only on $(x_t, \mathbf{a}_t)$.

The augmented local observation $\tilde{o}_{i,t}$ is a deterministic measurable function of (s_t, z_k) because $o_{i,t} = O_i(s_t)$ and the language componenet is $\alpha_i(o_{i,t})z_k$. Hence the policy is a Markov control law on the extended state space and all standard discounted-MDP constructions apply. □

Corollary 9 (Blockwise querying is the same construction at block level)**.** *If the language module is queried every K episodes and the embedding is held constant inside each block, then each episode inside a block still satisfies 8 with the same fixed embedding z_m^{blk}. Hence all within-episode statements below apply to the practical "query every K episodes" implementation used in the experiments.*

H.1 CORRECT POLICY-GRADIENT FORMULA

The policy-gradient expression must include the discount factor γ^t in the score-function form. The correct discounted formulation is as follows (Sutton et al., 1999).

Definition 10 (Fixed-context shaped values)**.** Fix $z_k \in \mathcal{Z}$ and parameters ω. On the extended state space $x = (s, z_k)$, define the shaped reward

$$\bar{r}_\lambda(x, \mathbf{a}) := r(s, \mathbf{a}) - \lambda c(s, \mathbf{a}).$$

The state value is

$$V_{\omega,\lambda}^{z_k}(x) = \mathbb{E}_\omega\left[\sum_{t=0}^{\infty} \gamma^t \bar{r}_\lambda(x_t, \mathbf{a}_t) \,\middle|\, x_0 = x,\ z_k\right],$$

the action value is

$$Q_{\omega,\lambda}^{z_k}(x, \mathbf{a}) = \bar{r}_\lambda(x, \mathbf{a}) + \gamma \mathbb{E}_{x' \sim \bar{P}(\cdot|x,\mathbf{a})}\big[V_{\omega,\lambda}^{z_k}(x')\big],$$

and the advantage is

$$A_{\omega,\lambda}^{z_k}(x, \mathbf{a}) = Q_{\omega,\lambda}^{z_k}(x, \mathbf{a}) - V_{\omega,\lambda}^{z_k}(x).$$

By Assumption 4, these quantities are well-defined and bounded.

Theorem 11 (Policy gradient for fixed language context)**.** *Under Assumptions 4, 5, and 6, the shaped return for a fixed context z_k satisfies*

$$\nabla_\omega J_\lambda^{\mathrm{sh}}(\omega; z_k) = \mathbb{E}_\omega \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\omega \log \nu_\omega(\mathbf{a}_t \mid x_t) Q_{\omega,\lambda}^{z_k}(x_t, \mathbf{a}_t) \;\middle|\; z_k \right]. \tag{12}$$

Equivalently, with normalized discounted occupancy measure

$$d_\omega^{z_k}(B) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_\omega(x_t \in B \mid z_k), \qquad B \subseteq \mathcal{X} \text{ measurable}, \tag{13}$$

one has

$$\nabla_\omega J_\lambda^{\mathrm{sh}}(\omega; z_k) = \frac{1}{1-\gamma} \mathbb{E}_{x \sim d_\omega^{z_k},\, \mathbf{a} \sim \nu_\omega(\cdot|x)} \big[\nabla_\omega \log \nu_\omega(\mathbf{a} \mid x) Q_{\omega,\lambda}^{z_k}(x, \mathbf{a}) \big]. \tag{14}$$

Replacing $Q_{\omega,\lambda}^{z_k}$ by $A_{\omega,\lambda}^{z_k}$ leaves the gradient unchanged.

Proof. Fix z_k and, for $T \geq 1$, define the finite-horizon return

$$R_{\lambda,T}(\tau) = \sum_{u=0}^{T-1} \gamma^u \bar{r}_\lambda(s_u, \mathbf{a}_u), \qquad J_{\lambda,T}^{\mathrm{sh}}(\omega; z_k) = \mathbb{E}_\omega[R_{\lambda,T}(\tau) \mid z_k].$$

For finite T, the trajectory likelihood contains only finitely many policy factors up to time $T-1$. Thus

$$\nabla_\omega J_{\lambda,T}^{\mathrm{sh}}(\omega; z_k) = \sum_{t=0}^{T-1} \mathbb{E}_\omega[\nabla_\omega \log \nu_\omega(\mathbf{a}_t \mid x_t) R_{\lambda,T}(\tau) \mid z_k]. \tag{15}$$

Decompose

$$R_{\lambda,T}(\tau) = \sum_{u=0}^{t-1} \gamma^u \bar{r}_\lambda(s_u, \mathbf{a}_u) + \gamma^t G_{t,T}^\lambda,$$

where

$$G_{t,T}^\lambda = \sum_{u=t}^{T-1} \gamma^{u-t} \bar{r}_\lambda(s_u, \mathbf{a}_u).$$

The past term has zero contribution since

$$\mathbb{E}\big[\nabla_\omega \log \nu_\omega(\mathbf{a}_t \mid x_t) \mid x_t, z_k\big] = 0.$$

Thus

$$\nabla_\omega J_{\lambda,T}^{\mathrm{sh}}(\omega; z_k) = \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_\omega \big[\nabla_\omega \log \nu_\omega(\mathbf{a}_t \mid x_t) G_{t,T}^\lambda \;\big|\; z_k\big].$$

Define

$$Q_{\omega,\lambda,T}^{z_k}(x_t, \mathbf{a}_t) = \mathbb{E}_\omega \big[G_{t,T}^\lambda \mid x_t, \mathbf{a}_t, z_k\big].$$

Then

$$\nabla_\omega J_{\lambda,T}^{\mathrm{sh}}(\omega; z_k) = \sum_{t=0}^{T-1} \gamma^t \mathbb{E}_\omega \left[\nabla_\omega \log \nu_\omega(\mathbf{a}_t \mid x_t) Q_{\omega,\lambda,T}^{z_k}(x_t, \mathbf{a}_t) \;\middle|\; z_k\right].$$

Taking $T \to \infty$ using dominated convergence yields

$$\nabla_\omega J_\lambda^{\mathrm{sh}}(\omega; z_k) = \mathbb{E}_\omega \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\omega \log \nu_\omega(\mathbf{a}_t \mid x_t) Q_{\omega,\lambda}^{z_k}(x_t, \mathbf{a}_t) \;\middle|\; z_k \right].$$

The occupancy form follows by exchanging sum and expectation. Replacing Q by A does not change the gradient since the baseline term integrates to zero. □

I PENALTY METHOD AND CONSTRAINED OPTIMALITY

This section addresses the core safety question. The shaped reward objective is not identical to the original constrained problem for arbitrary λ, but on the convex benchmark problem over discounted occupancy measures one can prove an exact dual certificate via a supporting-hyperplane argument. This avoids potential gaps that arise when applying minimax arguments such as Sion's theorem without verifying the required conditions.

Proposition 1 (Shaped and shifted objectives have the same maximizers for fixed λ)**.** *For every policy class* Π *and every fixed* $\lambda \geq 0$*,*

$$\arg\max_{\pi\in\Pi} J_\lambda^{\mathrm{sh}}(\pi) = \arg\max_{\pi\in\Pi} \widetilde{J}_\lambda(\pi), \tag{16}$$

because $\widetilde{J}_\lambda(\pi) = J_\lambda^{\mathrm{sh}}(\pi) + \lambda b$*.*

Proof. For fixed λ and b, the quantity λb is constant with respect to π. Therefore, for any $\pi_1, \pi_2 \in \Pi$,

$$J_\lambda^{\mathrm{sh}}(\pi_1) \geq J_\lambda^{\mathrm{sh}}(\pi_2) \quad \Longleftrightarrow \quad J_\lambda^{\mathrm{sh}}(\pi_1) + \lambda b \geq J_\lambda^{\mathrm{sh}}(\pi_2) + \lambda b.$$

Using $\widetilde{J}_\lambda(\pi) = J_\lambda^{\mathrm{sh}}(\pi) + \lambda b$, this is equivalent to

$$\widetilde{J}_\lambda(\pi_1) \geq \widetilde{J}_\lambda(\pi_2).$$

Hence the two objectives induce the same ordering over Π, and therefore have the same maximizers. □

J PENALTY-BASED REWARD–COST TRADE-OFF

The proposed method optimizes a fixed penalty-based objective rather than solving the constrained problem through online dual-variable updates. We therefore characterize the reward-cost trade-off induced by this objective within the parameterized decentralized policy class used in practice. The following result should not be interpreted as a guarantee of exact constraint satisfaction.

Definition 12 (Restricted-class approximation gap)**.** For fixed λ, define

$$\varepsilon_{\mathrm{cls}}(\lambda) = \sup_{\pi\in\Pi_{\mathrm{dec}}} \widetilde{J}_\lambda(\pi) - \sup_{\pi\in\Pi_{\mathrm{par}}} \widetilde{J}_\lambda(\pi). \tag{17}$$

This quantity is nonnegative and measures the expressivity loss from restricting the search to the parameterized class.

Theorem 13 (Restricted-class reward certificate and violation tradeoff)**.** *Fix* $\lambda > 0$ *and suppose that* $\hat{\pi} \in \Pi_{\mathrm{par}}$ *satisfies*

$$\widetilde{J}_\lambda(\hat{\pi}) \geq \sup_{\pi\in\Pi_{\mathrm{par}}} \widetilde{J}_\lambda(\pi) - \varepsilon_{\mathrm{opt}} \tag{18}$$

for some optimization error $\varepsilon_{\mathrm{opt}} \geq 0$*. Then for every feasible comparator* $\pi_f \in \Pi_{\mathrm{dec}}$ *with* $J_c(\pi_f) \leq b$*,*

$$J_r(\pi_f) - J_r(\hat{\pi}) \leq \varepsilon_{\mathrm{cls}}(\lambda) + \varepsilon_{\mathrm{opt}} + \lambda\big(b - J_c(\hat{\pi})\big). \tag{19}$$

Equivalently, the positive budget violation of $\hat{\pi}$ *obeys*

$$\big[J_c(\hat{\pi}) - b\big]_+ \leq \frac{\big[J_r(\hat{\pi}) - J_r(\pi_f)\big]_+ + \varepsilon_{\mathrm{cls}}(\lambda) + \varepsilon_{\mathrm{opt}}}{\lambda}. \tag{20}$$

Proof. Let π_f be any feasible comparator, so $J_c(\pi_f) \leq b$. Then

$$J_r(\pi_f) \leq J_r(\pi_f) - \lambda\big(J_c(\pi_f) - b\big) = \widetilde{J}_\lambda(\pi_f).$$

By definition of $\varepsilon_{\mathrm{cls}}(\lambda)$,

$$\widetilde{J}_\lambda(\pi_f) \leq \sup_{\pi\in\Pi_{\mathrm{par}}} \widetilde{J}_\lambda(\pi) + \varepsilon_{\mathrm{cls}}(\lambda).$$

Using the $\varepsilon_{\mathrm{opt}}$-optimality of $\hat{\pi}$ in equation 18,

$$\sup_{\pi \in \Pi_{\mathrm{par}}} \widetilde{J}_\lambda(\pi) \le \widetilde{J}_\lambda(\hat{\pi}) + \varepsilon_{\mathrm{opt}}.$$

Combining the previous displays gives

$$J_r(\pi_f) \le \widetilde{J}_\lambda(\hat{\pi}) + \varepsilon_{\mathrm{cls}}(\lambda) + \varepsilon_{\mathrm{opt}}.$$

Expanding $\widetilde{J}_\lambda(\hat{\pi})$ proves equation 19:

$$J_r(\pi_f) \le J_r(\hat{\pi}) - \lambda\big(J_c(\hat{\pi}) - b\big) + \varepsilon_{\mathrm{cls}}(\lambda) + \varepsilon_{\mathrm{opt}}.$$

Rearranging yields

$$\lambda\big(J_c(\hat{\pi}) - b\big) \le J_r(\hat{\pi}) - J_r(\pi_f) + \varepsilon_{\mathrm{cls}}(\lambda) + \varepsilon_{\mathrm{opt}}.$$

If the left-hand side is nonpositive then equation 20 is trivial. If it is positive, divide by $\lambda > 0$ and use $[u+v]_+ \le [u]_+ + v$ for $v \ge 0$ to obtain equation 20. □

K Structural role of the gate

The gate is one of the main components of the method. It allows the policy to use language guidance when helpful and suppress it when it is uninformative or misleading. This section gives three formal justifications: representational fallback, shrinkage optimality in a noisy embedding model, and robustness under embedding perturbations.

K.1 Representational fallback

Proposition 2 (The gated policy class contains the unguided class)**.** *Assume that:*

1. *For each agent $i \in \mathcal{N}$, the gate family contains the zero map $G_\psi(o_{i,t}) \equiv 0$.*
2. *The actor architecture with augmented input $\tilde{o}_{i,t} = [o_{i,t}, \alpha_{i,t} z_k]$ contains as a subfamily policies that ignore the language-input vector.*

Then the gated policy class contains the unguided decentralized policy class. Consequently, the supremum of the shaped return J_λ^{sh} over the gated class is at least the supremum over the unguided decentralized class. In particular, there exists a gate–actor parameter setting that exactly reproduces any unguided decentralized actor policy.

Proof. The proof is constructive. Consider any unguided decentralized policy $\pi_i(a_{i,t} \mid o_{i,t})$ from the base class. By the first assumption, choose gate parameters ψ such that $G_\psi(o_{i,t}) = 0$ for all $o_{i,t}$. Then $\alpha_{i,t} = 0$, and the augmented observation becomes

$$\tilde{o}_{i,t} = [\, o_{i,t}, \mathbf{0} \,].$$

By the second assumption, there exists a choice of actor parameters that ignores the language-input coordinates and reproduces the original mapping $o_{i,t} \mapsto \pi_i(a_{i,t} \mid o_{i,t})$. Hence every unguided decentralized policy is realizable within the gated architecture. Taking suprema of J_λ^{sh} over the two classes gives the claimed lower bound. □

Proposition 3 (Bounded Influence of Gated Language Guidance)**.** *Assume that the strategy embedding is bounded as $\|z_k\|_2 \le B_z$ and that the gate output satisfies $\alpha_{i,t} \in [0,1]$. Let the augmented observation be $\tilde{o}_{i,t} = [\, o_{i,t}, \alpha_{i,t} z_k \,]$. Then the contribution of the language-guidance component to the actor input is bounded by $\|\alpha_{i,t} z_k\|_2 \le B_z$. Moreover, let $f_{\theta_i}(\tilde{o}_{i,t})$ denote the vector-valued output of the actor network before action sampling, for example the action mean or policy logits. If f_{θ_i} is L_f-Lipschitz with respect to its input, then for any two strategy embeddings z_k and z'_k with corresponding gate values $\alpha_{i,t}$ and $\alpha'_{i,t}$, the change induced in the actor input satisfies*

$$\|\alpha_{i,t} z_k - \alpha'_{i,t} z'_k\|_2 \le \|z_k - z'_k\|_2 + B_z |\alpha_{i,t} - \alpha'_{i,t}|,$$

and therefore the corresponding change in actor output is bounded by

$$\|f_{\theta_i}([o_{i,t}, \alpha_{i,t} z_k]) - f_{\theta_i}([o_{i,t}, \alpha'_{i,t} z'_k])\|_2 \le L_f \Big(\|z_k - z'_k\|_2 + B_z |\alpha_{i,t} - \alpha'_{i,t}| \Big).$$

Proof. Since $\alpha_{i,t} \in [0,1]$ and $\|z_k\|_2 \le B_z$, we have

$$\|\alpha_{i,t} z_k\|_2 = \alpha_{i,t} \|z_k\|_2 \le B_z.$$

For the second claim, apply the triangle inequality:

$$\|\alpha_{i,t} z_k - \alpha'_{i,t} z'_k\|_2 \le \|\alpha_{i,t}(z_k - z'_k)\|_2 + \|(\alpha_{i,t} - \alpha'_{i,t}) z'_k\|_2.$$

Using $\alpha_{i,t} \le 1$ and $\|z'_k\|_2 \le B_z$ gives

$$\|\alpha_{i,t} z_k - \alpha'_{i,t} z'_k\|_2 \le \|z_k - z'_k\|_2 + B_z |\alpha_{i,t} - \alpha'_{i,t}|.$$

If the actor output map f_{θ_i} is L_f-Lipschitz in its input, then the induced change in actor output is at most L_f times the change in the input, yielding the result. □

Overall, the analysis provides a theoretical characterization of the proposed framework under idealized assumptions. In particular, it shows that fixed language embeddings can be incorporated through an extended-state formulation without altering the standard policy-gradient structure, and characterizes the reward-cost trade-off induced by penalty-based optimization. These results are intended to provide structural insight into the proposed method rather than formal safety guarantees for the non-convex neural policies used in our experiments.