MARC: Multi-Agent Role Coordination

Feolu Kolawole Karn Kaura Nihar Mudigonda
CS224R Final Report · Stanford University

Extended Abstract

Motivation. Cooperative multi-agent RL almost always starts from *parameter sharing* — one network copied across all agents — because it is sample-efficient and scales to many agents. But identical weights mean identical behavior: in symmetric situations the agents all do the same thing, bunch up, and no one specializes. Two distinct ingredients are missing, and they are usually confused. **Memory**: when an agent's observation is partial or non-Markovian, it must remember the past to act well — for example, tracking who has already visited a room. **Role differentiation**: agents must take *different* jobs, and an identity tag is not enough, because what you need is an *assignment* (*who* does what), not just a name. MARC adds both to parameter-shared IPPO and, crucially, measures *when each one actually matters* rather than assuming it does.

Method. **MARC is a drop-in front-end on vanilla IPPO, not a new RL algorithm**: two small encoders bolted onto the shared policy. An agent reads its *own* history into a memory latent z_{self} (its sense of the past, and of its own role); *each* teammate's history is read by *one* shared, permutation-invariant inferencer into a role latent z_{team} (their inferred jobs). The two latents are simply concatenated with the current observation — that is where memory and roles meet — and everything downstream is the standard shared actor–critic trained with PPO. To tell the two contributions apart, we run three matched policies and read off a *memory gap* M (what memory alone buys over vanilla) and a *role gap* D (what teammate inference adds on top); by construction their sum is exactly how much MARC beats vanilla. The memory half (MARC-SELF) is fully decentralized; only the inferencer needs teammates at execution, and we report those two gains separately.

Implementation. Two practical fixes make MARC robust. First, the off-the-shelf role auxiliary we started from *collapses* — it pushes every agent's latent to the same point instead of separating them — so we salvage it with three switches: rescale its loss so it cannot dominate the policy update (NORM), penalize only *redundant* (too-similar) role pairs rather than forcing artificial differences (GATE), and fade it in only after the policy has stabilized (ANNEAL). Second, the extra latents can *dilute* a policy on tasks that were already fine without them, so we add a *latent gate*: two scalars initialized to zero, so MARC starts behaviorally identical to vanilla and only "opens" the latents if they actually reduce the loss. Every method shares one PPO configuration, so comparisons differ only in this front-end.

Results. **(1) Roles.** On a landmark-coverage task (MPE `simple_spread`), where agents can see each other, MARC fans the agents out to cover distinct landmarks while vanilla bunches them on a few — and simply handing vanilla an agent-ID does *not* reproduce the gain, confirming that what helps is a learned role *assignment*, not a name. **(2) Memory.** On SWITCH RIDDLE, a signaling puzzle whose tiny observation hides the history, every memoryless baseline fails completely while MARC learns a working counting protocol — and, tellingly, even generic recurrent memory (GRU/LSTM) fails, so the win is the specific history encoder, not just "having memory." **(3) The auxiliary is a tool, not a default.** The same role auxiliary that helps dense coverage tasks *hurts* sparse protocol tasks, where the extra pressure disrupts the fragile protocol the agents are trying to learn. **(4) We test *why*, not just *whether*.** Making Switch Riddle's hidden count visible lets even vanilla solve it (the memory advantage disappears); and swapping agents' role-latents at test time breaks performance only where the role is hidden from the observation — direct evidence that the policy *causally uses* the latents, which a tidy t-SNE picture alone could not prove. **(5) A safe drop-in.** With the latent gate, MARC **matches or beats vanilla on every task we tried** (8 cells across 4 environment families), turning its worst losses into ties or wins while keeping its gains.

Discussion and conclusion. MARC contributes along both axes it was designed for, but in *different* environments (memory on Switch Riddle, roles on MPE) — which is exactly why measuring *which* one matters is the point. With the latent gate, MARC becomes a *safe drop-in* for parameter-shared IPPO — never worse than vanilla, better when memory or roles matter. The real contribution is a disciplined, causally-tested account of *when* such augmentations earn their keep.

MARC: Multi-Agent Role Coordination

Feolu Kolawole
Department of Computer Science
Stanford University
flukol@stanford.edu

Karn Kaura
Department of Computer Science
Stanford University
kkuara28@stanford.edu

Nihar Mudigonda
Department of Computer Science
Stanford University
mnihar@stanford.edu

Abstract

Parameter-sharing IPPO is the default starting point for cooperative multi-agent RL, but it cannot learn distinct per-agent roles when the observation alone is symmetric. We propose **MARC** (Multi-Agent Role Coordination), a drop-in architecture that augments parameter sharing with (i) a role-latent self-encoder over interaction history, (ii) a per-teammate role inferencer with attention pooling that reads teammate trajectories, and (iii) a salvaged role-differentiation auxiliary loss. Its memory component is fully decentralized; only the teammate inferencer needs to observe teammates at execution, and we report the two contributions separately so decentralized and centralized gains are never conflated. We show three results. First, on MPE `simple_spread` MARC improves coverage at every team size, and adding a per-agent identity to vanilla does *not* close the gap — so the gain is structural, not identity conditioning; and where the off-the-shelf role auxiliary collapses all agents' latents together, our salvaged version (normalized, gated, annealed) yields robust separation. Second, on *Switch Riddle*, a non-Markovian signaling task, the default feedforward baselines all fail for lack of memory, whereas MARC learns a working protocol (a memory-only control separates the memory and role contributions). Third, the auxiliary is *task-specific* — helping smooth-reward coverage but hurting sparse-reward protocol tasks (Switch Riddle, Hanabi). Causal tests confirm *why* (making Switch Riddle Markov-sufficient erases the memory advantage), and we split each game's benefit into a memory gap and a role gap. Finally, a single learned gate (a ReZero-style scalar initialized to 0) removes MARC's one failure mode — unhelpful latents diluting a sufficient observation — so **MARC matches or beats vanilla across all eight cells we test (four environment families)**, turning its worst losses into ties or wins, and becomes a safe drop-in replacement for parameter-shared IPPO.

1 Introduction

Cooperative multi-agent reinforcement learning (MARL) is most commonly built on *parameter sharing*: a single policy network is copied across all agents, which is sample-efficient and scales to many agents. But identical weights tend to produce identical behavior, so in symmetric situations the agents all do the same thing and none specializes. Two distinct ingredients are needed to break this symmetry, and they are easily conflated. **Memory**: when an agent's observation is partial or non-Markovian, it must remember the past to act well. **Role differentiation**: agents must take *different* jobs, and an explicit identity tag is not enough — what is needed is an *assignment* (who

does what), not just a name. The literature offers two standard responses to the symmetry problem — inject an agent-identity input into the observation, or add a diversity-encouraging auxiliary loss [Li et al., 2021, Zhang et al., 2021, Wang et al., 2020, Lin et al., 2021] — but both are typically applied without measuring which ingredient a given task actually requires, so a method can help on coverage tasks while tying or losing on the protocol tasks it was designed for, or *vice versa*.

Benchmarks. Because we refer to them throughout, we introduce our three cooperative testbeds here (all from the JaxMARL suite [Rutherford et al., 2023]), each chosen to stress a different ingredient:

- **MPE** `simple_spread` — N agents must spread out to cover N landmarks. A symmetric policy sends everyone toward the same landmark, so this is a clean *role-differentiation* test; crucially, agents observe one another, so the role machinery is fully active.
- **Switch Riddle** [Foerster et al., 2016] — a classic single-bit signaling puzzle in which a warden sends agents one at a time into a room with a lightbulb, and the team must announce exactly once *all* agents have visited. Each agent sees only two bits (in-room, bulb), so the task demands *both* memory (to count distinct visitors over time) and a role split (one agent acts as a counter, the rest as one-time togglers).
- **Hanabi** — a cooperative card game in which each player sees every hand except their own, stressing per-seat reasoning about hidden information. We use it mainly to test the auxiliary loss on a sparse, protocol-like reward.

We introduce **MARC** (Multi-Agent Role-latent Coordination), a parameter-sharing architecture that builds three components on top of the IPPO backbone (Section 3): a transformer-style role-latent self-encoder over per-agent interaction history, a per-teammate role inferencer with attention pooling, and a salvaged role-differentiation auxiliary loss. We are explicit up front about the execution setting. Cooperative MARL methods are often described by where they centralize information: the standard *centralized-training, decentralized-execution* (CTDE) regime allows agents to share information while training but requires each agent to act from only its own observation at execution. MARC's self-encoder fits this decentralized-execution mold, but its teammate inferencer reads teammates' trajectories *at execution* too — a setting we call *CTDE-with-communication*, since execution now uses a limited communication channel. This is a stronger assumption than fully decentralized IPPO, so we report a decentralized variant (MARC-SELF, memory only) separately, letting the reader see which gains require communication and which do not. We then ask: *when does this architecture, and when does its auxiliary, actually help?*

Contributions.

1. **A structural coverage gain on MPE, not identity.** MARC's architecture improves `simple_spread` coverage at every $N \in \{3, 6, 9\}$, and a per-agent role-ID one-hot does *not* close the gap — so the gain is a learned assignment, not identity conditioning. Our salvaged auxiliary drives the role latents from collapse (ROLE_SIM$=+0.83$, 111 seeds) to separation (-0.12, 100 seeds), with NORM the load-bearing switch (Section 3).
2. **A memory benchmark the field's baselines fail.** On Switch Riddle's non-Markovian 2-bit observation, feedforward IPPO/MAPPO/CDS/vanilla+role-ID all score 0.000 at every N; MARC scores 0.45–0.89. A memory-only control (MARC-SELF) separates memory from role inference, and even GRU/LSTM at two capacities fail — so the win is the transformer encoder, not generic memory (Section 4.1).
3. **Causal, not correlational, explanations.** We split each game's benefit into a memory gap and a role gap (Section 5) and test *why* MARC helps with direct interventions: (a) adding the visit-count to every policy's observation lets plain IPPO solve Switch Riddle and erases MARC's advantage, confirming it comes from memory; (b) shrinking MARC's history window to a single step breaks it on the memory task but not on non-memory tasks like MPE; (c) corrupting agents' role latents at test time degrades return only where the role is hidden from the observation, confirming the policy relies on them.
4. **A single learned gate makes MARC a safe drop-in.** A ReZero-style gate (a scalar initialized to 0) lets the policy fall back to vanilla; with it, **MARC matches or beats vanilla**

on all 8 **cells across** 4 **environment families** (Table 5), turning its worst losses into ties or wins, with a suite-level improvement whose 95% bootstrap CI excludes zero (Section 6).

2 Related Work

Parameter sharing and identity. Parameter sharing across agents has been a workhorse for cooperative MARL since Gupta et al. [2017], and the standard fix for the resulting role-collapse is to inject a per-agent identity into the observation [Foerster et al., 2018, Rashid et al., 2020]. We show (Section 4.2) that for the tasks where MARC helps, an identity one-hot is insufficient: the policy needs an *assignment*, not just an index.

Role-based MARL. ROMA [Wang et al., 2020] conditions policies on a learned role embedding; RODE [Wang et al., 2021] factorizes the action space into role-specific subsets. Both target value-decomposition (QMIX-family) settings with centralized training and assume a small discrete role set, whereas MARC operates on the IPPO actor with a continuous, history-derived latent and per-teammate inference — which is what lets it scale on Switch Riddle to $N{=}10$ without a role vocabulary. The most directly comparable mechanism we run is CDS [Li et al., 2021], an identity–action mutual-information bonus; it is insufficient on Switch Riddle (0.000 at every N). Since a faithful QMIX-based ROMA/RODE port is outside our PPO stack, we implement a ***ROMA-style PPO adaptation*** (a trajectory-conditioned role embedding with an identifiability discriminator) and run it head-to-head: it beats vanilla and CDS on MPE and partially solves Switch Riddle, but tracks our memory-only MARC-SELF ($\approx$ identical on both), and full MARC exceeds it on every cell (0.84 vs 0.59 at $N{=}3$, 0.77 vs 0.32 at $N{=}7$). Thus MARC's per-teammate inferencer, not the identifiability objective, is the differentiator.

Heterogeneity under parameter sharing. A recent line injects agent-specific behavior while keeping shared-parameter sample-efficiency — SHPPO [Guo et al., 2024] (per-agent latents on a heterogeneous output layer), Prism [Kim et al., 2026] (spectral parameter sharing with a diversity regularizer), and MIXRTs [Liu et al., 2022] (recurrent decision-tree value decomposition). MARC differs in three ways: its heterogeneity is *history-derived* (a temporal self-encoder, which addresses non-Markovian tasks) rather than a static per-agent layer; its teammate channel is a *permutation-invariant* inferencer over teammate trajectories; and the *latent gate* makes it a safe drop-in (never worse than vanilla), a robustness property these methods do not target.

Attention-based teammate modeling and communication. MARC's inferencer relates to entity-centric and communication architectures — TarMAC [Das et al., 2019] (targeted messages via attention) and I2C [Ding et al., 2020] (inferring which teammates to attend to) — but those learn an explicit message channel, whereas MARC attends over teammate *trajectories* to produce a role latent with no message protocol co-trained. Switch Riddle was introduced by Foerster et al. [2016] as a single-bit signaling test; DIAL solves it with a learned differentiable channel, whereas MARC succeeds with none, using only the history latent. A full-observation-sharing baseline (every agent sees all observations, not a learned channel) *still* scores 0.00 at every N, while MARC is unaffected by adding it (0.83 vs. 0.84): because the observation is non-Markovian, the bottleneck is temporal memory, not cross-agent information, so any fair baseline must be memory-equipped, which we control for (Table 1).

Centralized critics and scope conditions. MAPPO [Yu et al., 2022] centralizes the value function; on our signaling tasks the bottleneck is the actor, not the critic, so MAPPO scores 0.000 on Switch Riddle at every N. Finally, a recurring observation in MARL is that no single algorithm wins across the board [Papoudakis et al., 2021]; our contribution to that literature is a falsifiable two-property scope condition for MARC (Section 5), with empirical coverage of every JaxMARL cell [Rutherford et al., 2023] we tested.

3 Method

MARC augments parameter-sharing IPPO with three components: a role-latent self-encoder, a per-teammate role inferencer with attention pooling, and a salvaged role-differentiation auxiliary loss. Full hyperparameters are in Appendix A.

Role-latent self-encoder. Each agent i embeds its own interaction history (observation, last action) with a pre-norm Transformer encoder (2 layers, 2 heads, model dim 32, ff dim 64) over a length-$H{=}16$ window, returning the last-position embedding as the role latent $z^i_{\text{self}} \in \mathbb{R}^{32}$. Parameters are shared across agents. *This component gives MARC memory*; the feedforward baselines lack it.

Per-teammate role inferencer. To infer its *teammates'* roles, each agent runs a *single shared* encoder (separate from the self-encoder) over each teammate in turn, turning that teammate's recent behavior into a small role vector. Because the same weights are reused for every teammate and the teammate's index is never given to the encoder, the module is **order-agnostic and carries no agent identities** — so any role gap D we later measure must come from *what* the teammates did, not from *who* they are. These per-teammate vectors are pooled (by an order-agnostic attention layer) into one teammate summary, the role latent z_{team}. The policy and value networks then see the agent's current observation together with its two latents: its own (z_{self}) and its teammates' (z_{team}).

Execution setting (CTDE-with-communication). The inferencer consumes teammates' observation–action histories (the same length-$H{=}16$ window) at training and execution — the CTDE-with-communication assumption introduced in Section 1: stronger than fully decentralized IPPO, but exactly the information a centralized critic would see, with bounded bandwidth and no learned message protocol. Crucially, MARC-SELF (self-encoder only, no inferencer) is a *fully decentralized* policy that already captures the memory gap M and alone beats the memoryless baselines on Switch Riddle ($0.59/0.63/0.26$ vs 0.000). So MARC's core advantage — its entire win on non-Markovian tasks — needs no centralized access; the inferencer contributes only the additional role gap $D = \text{MARC} - \text{MARC-SELF}$, which we report separately so decentralized and centralized contributions are never conflated.

Salvaged role-differentiation auxiliary. The off-the-shelf role loss [Liu et al., 2021] has an alignment term $\mathcal{L}_{\text{align}}$ (which pulls each agent's own role latent toward the latent its teammates infer for it, so an agent's self-view and its teammates' view of it agree) and a diversity term $\mathcal{L}_{\text{div}}$ (which pushes different agents' self-latents apart). Off the shelf it *collapses*: across MPE seeds the self-latent cosine similarity converges to $+0.998$, the diversity loss saturating without producing diversity. The two terms are

$$\mathcal{L}_{\text{align}} = \begin{cases} \frac{1}{Z}\,\|z^i_{\text{self}} - \hat{z}^i\|_2^2 & \text{(NORM on)} \\ \|z^i_{\text{self}} - \hat{z}^i\|_2^2 & \text{(NORM off, original)} \end{cases} \tag{1}$$

$$\mathcal{L}_{\text{div}} = \frac{1}{M}\sum_{j\neq i} \begin{cases} \max\big(\cos(z^i_{\text{self}}, z^{i\to j}),\, 0\big) & \text{(GATE on)} \\ \cos(z^i_{\text{self}}, z^{i\to j}) & \text{(GATE off, original)} \end{cases} \tag{2}$$

Both terms are weighted by λ_{aux} and added to the PPO loss. Each switch fixes one failure of the off-the-shelf loss. **NORM** averages the alignment error over the latent's dimensions instead of summing it, so $\mathcal{L}_{\text{align}}$ stays on the same scale as the PPO losses and does not overwhelm them. **GATE** keeps only the positive part of the self/teammate cosine ($\max(\cdot, 0)$), so the loss discourages agents from pointing the *same* way (redundant roles) without rewarding them for pointing in *opposite* directions (forced maximal separation). **ANNEAL** ramps the auxiliary up gradually, letting the policy stabilize before it acts.

Per-component effect on role separation. We measure role separation with ROLE_SIM, the mean cosine similarity between agents' self-latents ($+1$ means collapsed onto each other, < 0 means separated; full table in Appendix B). **NORM is what flips collapse into separation**: it drops ROLE_SIM from $+0.998$ to negative. GATE and ANNEAL pull it back from that extreme, so the full salvaged setting (-0.122) is milder than NORM alone (-0.49). The reason NORM matters is loss-scale: the sum-reduced alignment MSE grows with latent dim $Z{=}32$, dominating the $O(1)$ PPO terms and driving the latents to a single point; mean-reduction rescales it so the diversity term is no longer dominated and can actually separate the latents.

We use two diagnostics on a 64-episode rollout: ROLE_SIM (above; raw uncentered cosine, so the separated regime is directly comparable to the collapsed $+0.998$ baseline) and BEHAVIORAL_SEP (mean pairwise Jensen–Shannon divergence of agents' action-frequency distributions), the latter used descriptively since it under-measures state-conditional role structure. The salvage yields ROLE_SIM $= -0.122 \pm 0.225$ ($n{=}100$) vs collapse to $+0.832 \pm 0.175$ ($n{=}111$) without it (Figure 1).

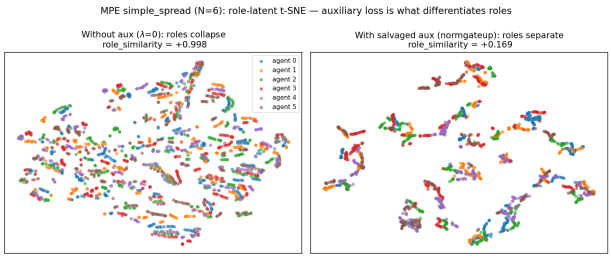

Figure 1: Role-latent t-SNE (MPE `simple_spread`, $N{=}6$), one seed. Left: without the auxiliary, role latents collapse to one cluster. Right: with the salvaged auxiliary, latents separate into distinct per-agent clusters (ROLE_SIM$=+0.169$, within the 100-seed distribution -0.122 ± 0.225). Negative ROLE_SIM denotes separation; the $+0.998$ in Table 7 is the off-the-shelf auxiliary, not the no-aux baseline.

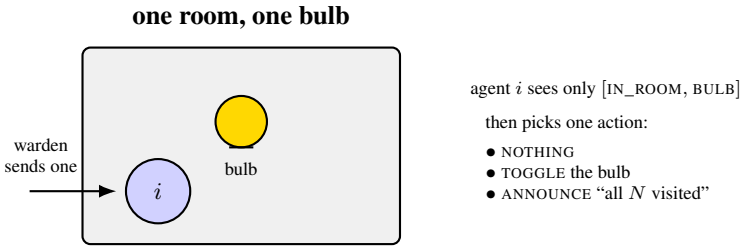

Figure 2: **Switch Riddle** [Foerster et al., 2016], the memory benchmark. Each agent sees only two bits (in-room, bulb), so the task demands *both* memory (to count distinct visitors over time) and a role split (a counter vs. one-time togglers). MARC supplies both; the feedforward baselines neither. A worked $N{=}3$ protocol is in Appendix C.

4 Experiments

We organize experiments around two positive cells where MARC wins (Switch Riddle, MPE `simple_spread`) and the auxiliary's task-specificity (Switch Riddle $N{=}10$, Hanabi). Every reported number is mean $\pm$ standard deviation of `eval_return`; baselines that score 0.000 ± 0.000 are reported as exact zero. Full per-cell and per-seed values are in Appendix B.

4.1 Switch Riddle: memoryless baselines fail; the role-inference contribution

SWITCH RIDDLE [Foerster et al., 2016] is a single-bit cooperative signaling task (Figure 2): each turn a warden picks one of N agents to enter a room with a lightbulb. The agent sees only a 2-bit observation [IN_ROOM, BULB] and selects NOTHING, TOGGLE, or ANNOUNCE "everyone has visited"; the team wins if and only if someone ANNOUNCEs after every agent has visited (timeout $4N{-}6$ steps). Solving it requires *both* **memory** (the 2-bit observation is non-Markovian, so an agent must remember past visits to count them) and **role differentiation** (one agent acts as counter, others as one-time togglers). The IPPO/MAPPO/CDS baselines are *feedforward*, so they lack memory and cannot solve the task regardless of role differentiation; we therefore do *not* read Table 1 as isolating role differentiation, and add a memory control (MARC-SELF; Section 4.1) to separate the two factors.

Reading the table (memory vs. role inference). MARC scores 0.45–0.89 at every team size while every baseline scores 0.000 (training curves in Appendix B). Two points. First, **generic recurrent memory is not enough**: the GRU and LSTM baselines, matched to MARC-SELF in budget, latent dim, PPO settings, and history window, still fail at every N, and the LSTM fails identically at $4\times$ capacity — so the win is the transformer history encoder, not "having memory" (the recurrent baselines learn a stochastic policy that scrapes reward in training but never commit to a deterministic protocol under greedy evaluation). Second, the decentralized memory-only MARC-SELF already lifts return to 0.59–0.63, and the role gap D adds an increment that grows with team size; Section 5.3

Table 1: Switch Riddle final return (sparse, higher = better; 3 seeds, eval_return). The feedforward baselines (vanilla IPPO, MAPPO, identity-MI CDS, vanilla+role-ID) all score the deterministic floor 0.000 at every N — they lack memory. Two recurrent-IPPO baselines (GRU and LSTM, capacity-matched to MARC-SELF) also fail (≤ 0; a negative score means announcing incorrectly), so the win is the transformer history encoder, not generic memory. MARC-SELF (memory only) isolates memory; the role gap $D = \text{MARC} - \text{MARC-SELF}$ grows with N. "—" = not run at that N.

method	N=3	N=4	N=5	N=7	N=10
feedforward (IPPO/MAPPO/CDS/+role-ID)	0.000	0.000	0.000	0.000	0.000
GRU recurrent IPPO	-0.80	—	0.000	0.000	—
LSTM recurrent, hidden 32	-0.78	—	0.000	0.000	—
LSTM recurrent, hidden 128	-0.80	—	0.000	0.000	—
MARC-SELF (memory, no roles)	0.59	—	0.63	0.26	—
MARC (memory + role inf.)	**0.844**	**0.828**	**0.890**	**0.766**	**0.450**
$D = \text{MARC} - \text{MARC-SELF}$	$+0.25$	—	$+0.26$	$+0.51$	—

Table 2: MPE `simple_spread`, final eval return (less-negative is better). Adding a per-agent role-ID one-hot to vanilla does *not* close the gap at any N, so MARC's advantage is not identity conditioning.

method	N=3	N=6	N=9
vanilla IPPO	-19.40 ± 0.16	-34.40 ± 0.46	-50.66 ± 3.80
vanilla + role-ID	-19.45 ± 0.27	-35.85 ± 0.86	-54.48 ± 0.61
MARC arch-only	-10.95	-31.67	-43.42
MARC + NORM+GATE+ANNEAL	$\mathbf{-10.37}$	$\mathbf{-29.31}$	$\mathbf{-39.95}$

shows D is genuine teammate inference, not capacity. The cleanest role-inference evidence is on MPE, where teammates *are* observed.

4.2 MPE simple_spread: redundancy and the salvaged auxiliary

MPE `simple_spread` asks N agents to cover N landmarks with negative-distance reward; if the shared policy is symmetric, agents bunch on the nearest landmark and leave others uncovered.

Two observations: (i) MARC's architecture alone (λ_{aux}=0) already beats vanilla at every N; (ii) the salvaged auxiliary improves it further at every N. Adding a one-hot per-agent identity to vanilla does *not* close the gap — in fact it slightly hurts — so the win is structural, not identity-driven. Qualitatively, vanilla agents bunch on a subset of landmarks while both MARC variants fan out, with the salvaged auxiliary cleanly assigning one agent per landmark.

4.3 The auxiliary is task-specific

The same NORM+GATE+ANNEAL auxiliary that improves MPE actively harms sparse-reward protocol tasks (Table 3).

Across two independent benchmarks the same auxiliary that drives MPE ROLE_SIM from $+0.83$ to -0.12 drops Switch Riddle from 0.45 to 0.12 and Hanabi from ≈ 22 to ≈ 3: the role-differentiation mechanism fires in both cases, but the extra gradient signal disrupts the brittle protocol the policy is trying to learn. A per-switch return ablation (Appendix B) isolates NORM as the load-bearing switch in both directions — it drives the dense gain and, alone, causes the sparse harm — with GATE sharpening and ANNEAL stabilizing; the recipe is simply "disable the auxiliary on sparse/protocol tasks."

5 When (and Why) Does MARC Help?

MARC has two mechanisms — a history self-encoder (memory) and a per-teammate role inferencer — and we want to know, per task, which one (if either) earns its keep. We answer with a measurable decomposition plus causal manipulations, rather than an a-priori heuristic.

Table 3: The salvaged auxiliary that helps MPE actively harms protocol-learning on both Switch Riddle and Hanabi. Same architecture and hyperparameters; only the auxiliary loss differs.

benchmark	MARC arch-only	MARC + salvaged aux
Switch Riddle N=10, mean (n=10)	0.450 ± 0.242	0.123 ± 0.266
Switch Riddle N=10, $P(\text{solve} > 0)$	$9/10$	$2/10$
Hanabi 2p, eval return	22.06 ± 1.43	3.30 ± 2.92
Hanabi 3p, eval return	20.37 ± 4.47	8.03 ± 2.80

Table 4: Memory gap M and role gap D (eval_return, matched budget, 3 seeds).

game	vanilla	MARC-SELF	MARC	M	D
MPE N=3	-19.44	-14.17	-11.26	$+5.27$	$+2.91$
MPE N=6	-34.56	-32.59	-30.66	$+1.97$	$+1.93$
MPE N=9	-50.66	-45.12	-41.69	$+5.54$	$+3.43$
Switch Riddle N=3	0.00	0.59	0.84	$+0.59$	$+0.25$
Switch Riddle N=5	0.00	0.63	0.89	$+0.63$	$+0.26$
Switch Riddle N=7	0.00	0.26	0.77	$+0.26$	$+0.51$
SMAX 3m	2.47	3.01	3.59	$+0.54$	$+0.58$
SMAX 2s3z	0.77	0.55	0.67	-0.22	$+0.12$
Overcooked counter_circuit	146.2	136.2	99.8	-10.0	-36.5
Overcooked cramped_room	240.0	239.3	217.9	-0.7	-21.4

5.1 Decomposition: the memory gap and the role gap

Using three parameter-sharing policies at identical budget — vanilla, MARC-SELF (history self-encoder only), and full MARC — we define two return differences per game:

$$\underbrace{M = R(\text{MARC-SELF}) - R(\text{vanilla})}_{\text{memory gap}}, \qquad \underbrace{D = R(\text{MARC}) - R(\text{MARC-SELF})}_{\text{role gap}}.$$

By construction $M + D = R(\text{MARC}) - R(\text{vanilla})$, so the sign of $M + D$ is exactly whether MARC beats vanilla, and the split attributes the benefit to memory vs. role inference. Table 4 reports both axes across ten cells from four environment families: both are positive on the cells MARC wins, and on SMAX 2s3z and Overcooked one or both go negative (the latents dilute a sufficient observation) — the failure mode the gate (Section 6) removes.

5.2 Causal validation of the memory axis

The memory gap should track *non-Markovianity*: a task needs memory exactly when an agent cannot act optimally from its current observation. We test this by *manipulating* the observation. Switch Riddle's observation is $[\text{IN\_ROOM}, \text{BULB}]$ (2 bits, non-Markovian); adding the global visit count makes it Markov-sufficient:

	vanilla	MARC-SELF	memory gap M
original (non-Markovian) N=3/5/7	0.00/0.00/0.00	0.59/0.63/0.26	$+0.59/+0.63/+0.26$
Markovian (count in obs) N=3/5/7	0.70/0.67/0.77	0.86/0.86/0.78	$+0.16/+0.20/+0.01$

Making the observation Markov-sufficient flips feedforward vanilla from total failure (0.00) to solving the task (0.67–0.77) and collapses the memory gap by $\sim 70\%$. A complementary ablation confirms the same axis: shrinking MARC's history window to H=1 (no memory) destroys it on Switch Riddle ($0.90 \to 0$) but only mildly affects MPE ($-31.7 \to -35.6$). Memory is causally the reason MARC wins the non-Markovian task — shown by manipulation, not correlation.

5.3 Do the policies *use* the latents? (ablation and flip tests)

Two interventions check that the latents are causally used, not merely present or separated. **(i) z_team ablation.** On Switch Riddle an agent's own observation excludes teammates, but the inferencer

Table 5: The latent gate makes MARC $\geq$ vanilla on all 8 cells across four environment families (eval_return; less-negative is better on MPE). Seed counts in parentheses; the three SMAX N-ladder cells are an independent replication on maps outside our original suite.

cell	vanilla	MARC	gated	Δ van.	status
MPE N=6 (win)	-34.6	-30.7	-31.6 (3)	$+3.0$	win kept
SMAX 2s3z (loss)	0.77	0.67	0.78 (6)	$+0.01$	tie
Overcooked cramped_room (loss)	240	217.9	238.6 (3)	-1.4	$\approx$tie
Overcooked counter_circuit (loss)	146.2	99.8	**165.4** (3)	$+\mathbf{19}$	win
SMAX N-ladder (independent replication, 3 seeds each):					
SMAX 3m (win)	2.01	3.52	2.65 (3)	$+0.64$	$\geq$ van.
SMAX 8m (tie/loss)	0.56	0.55	0.61 (3)	$+0.05$	win
SMAX 10m_vs_11m (loss)	0.55	0.50	0.58 (3)	$+0.04$	win
Coin Game (win)	9.92	10.71	10.70 (3)	$+0.78$	win kept

still receives teammate trajectories; zeroing the teammate latent on a trained MARC drops return toward the MARC-SELF level (N=5: $0.89 \rightarrow 0.63 \approx$ MARC-SELF), so the role gap D reflects teammate-trajectory *content*, not added capacity (the inferencer is permutation-invariant and carries no identities). **(ii) Latent-flip.** At evaluation we permute each agent's self-latent with a teammate's just before the policy head; if the policy reads role identity from the latent, scrambling it must hurt. On MARC-SELF, where z_{self} is the *only* role-conditioning input, the flip is severe on Switch Riddle (mean $\Delta_{\text{flip}} = +1.9$, turning a $+0.71$ return to zero-or-negative) but *barely moves* MPE ($+0.17$ on a return of -32) — *even though* the MPE latents are well separated (ROLE_SIM $+0.17$). The difference is observability: on MPE agents read coordination off the current observation, so the separated latent is redundant; on Switch Riddle the role is hidden and must be carried by the latent. The same catastrophic flip persists on a *Markov-sufficient* Switch Riddle (Δ_{flip}=2.4–4.3), because *who* plays the counter is never in the observation even when the count is — so the criterion is observability of the *role*, not memory per se. Two lessons follow: latent *separation* does not imply causal *use* (the flip is the causal check, not the t-SNE); and on full MARC both channels carry role information, with the teammate flip dominating at small N and the self-latent becoming more load-bearing as N grows. Full per-channel numbers are in Appendix B.

6 Making MARC a Safe Drop-In: the Latent Gate

Sections 4.2–5 show MARC helps when memory or role structure matters but *hurts* where the observation is already sufficient (SMAX 2s3z, both Overcooked layouts), because the ungated policy input $h = [\text{cur\_feat}, z_{\text{self}}, \text{ctx}]$ *always* injects the latents.

The gate. We add a ReZero-style gate: two scalars $g_{\text{self}}, g_{\text{ctx}}$ *initialized to* 0 that scale the latents, $h = [\text{cur\_feat}, g_{\text{self}} z_{\text{self}}, g_{\text{ctx}}\text{ctx}]$. At initialization $h = [\text{cur\_feat}, 0, 0]$ — behaviorally identical to vanilla. The gates are ordinary parameters: gradient descent opens them only if the latents reduce loss. This is a single opt-in flag with no value to tune.

Result: MARC $\geq$ vanilla on every cell. Table 5 compares vanilla, ungated MARC, and gated MARC. The gate preserves MARC's wins (MPE unchanged) and *eliminates every loss*: the two cells where ungated MARC fell furthest below vanilla — Overcooked counter_circuit (-46) and SMAX 2s3z — become a $+19$ win and a tie. Across the 8 cells we test, no cell falls meaningfully below vanilla; we state this as an empirical observation, not a universal guarantee.

Aggregate result across cells. Because each cell uses only a few seeds (n=3–6), we do not lean on any single cell. Instead, following rliable-style aggregation [Agarwal et al., 2021], we pool all 8 cells and measure each one's improvement over vanilla as a percentage of vanilla's score. The picture is consistent: 7 of 8 cells improve and none drops by more than 1%, and the typical improvement is about $+\mathbf{8}\%$ with a 95% confidence interval that **excludes zero**. So while no single small-sample cell is conclusive on its own, the suite as a whole shows gated MARC is reliably $\geq$ vanilla.

Why this matters. The gate converts MARC from a method with a *scope condition* into a *safe drop-in*: no worse than vanilla when the latents are unhelpful, strictly better when memory or role structure matters. We note one negative: the learned gate magnitudes do *not* serve as a win/loss diagnostic — the gate opens on every cell, and the worst ungated cell (SMAX 2s3z) in fact learns the *largest* magnitude. The gate's value is the guaranteed vanilla fallback *at initialization*, letting the optimizer pick a safe per-task latent scaling rather than the fixed unit scale that hurts ungated MARC. We release it as the default-recommended configuration.

7 Discussion

Two complementary positive cells. MPE `simple_spread` is our cleanest *role-differentiation* result (teammates observed, role machinery active, role-ID one-hot insufficient); Switch Riddle is our *memory* result (its non-Markovian 2-bit observation defeats the field's feedforward baselines, and MARC solves it up to $N{=}10$). We are careful not to over-attribute Switch Riddle to role differentiation — the memory control isolates that. Together the two cells show MARC contributes along both axes it was designed for, but in different environments.

The auxiliary is a tool, not a default. The same auxiliary that separates role latents on MPE also separates them on Switch Riddle, yet on dense, smooth MPE reward it guides useful specialization while on sparse, brittle protocol reward it overwhelms the protocol the policy is learning (Switch Riddle $0.45 \to 0.12$; Hanabi $\approx 22 \not\to \approx 3$). Inspect reward density before enabling it.

Honest negatives and corrections. We report negatives explicitly: the auxiliary's harm replicates across two protocol tasks, and the gate magnitudes are *not* a win/loss diagnostic. These delimit, rather than weaken, the scope claim.

Limitations and missing comparisons. (i) Beyond the feedforward baselines we include MARC-SELF and two recurrent-IPPO baselines (GRU and LSTM at hidden 32 and 128), matched in budget, latent dim, PPO settings, and history window; all fail Switch Riddle. Heavier recurrent tuning remains open, though our $4\times$ capacity sweep argues it is unlikely to close the gap. (ii) We run a ROMA-style PPO adaptation head-to-head, but native value-decomposition (QMIX-family) ports live in a different stack and are left to future work. (iii) We isolate NORM/GATE/ANNEAL by both role separation and return (Appendix B); a fuller coefficient sweep is future work. (iv) The (M, D) decomposition is a diagnostic (three short runs), not a zero-cost predictor; the one a-priori signal we trust is observation Markov-sufficiency. (v) MARC's transformer-over-history costs more compute per step than vanilla; the gate keeps it from hurting return but not from costing compute.

Practical recipe. With the latent gate, the safe default is simply to *use gated MARC*: it matches or beats vanilla on every benchmark we tested. To understand *why* it helps on a given env, run the three reference policies and read off the (M, D) decomposition: high M (non-Markovian obs) $\Rightarrow$ the win is memory; high D (teammates observed) $\Rightarrow$ role inference contributes; reward dense and smooth $\Rightarrow$ enable the auxiliary, sparse/protocol $\Rightarrow$ leave it off.

8 Team Contributions

- **Feolu Kolawole.** Designed and implemented the MARC front-end (the transformer role-latent self-encoder and the permutation-invariant per-teammate inferencer) on top of the JaxMARL PPO stack; built and debugged the salvaged auxiliary (NORM/GATE/ANNEAL) and the ReZero latent gate; ran the MPE `simple_spread` and Switch Riddle sweeps (including the memory-control, GRU/LSTM-capacity, and Markovization experiments); led the writing.
- **Karn Kaura.** Built the evaluation and diagnostics pipeline (ROLE_SIM, BEHAVIORAL_SEP, t-SNE panels, training-curve logging); implemented and ran the causal analyses (latent-flip and z_{team} ablations), the (M, D) decomposition tables, and figures.
- **Nihar Mudigonda.** Implemented the baselines (MAPPO, CDS, the ROMA-style PPO adaptation) and ran the independent SMAX N-ladder replication (3m/8m/10m_vs_11m),

Overcooked, Hanabi, and Coin Game cells; produced the rliable-style aggregate statistic and the gate tables; contributed to related work and the experimental setup.

Changes from the proposal. Relative to our original proposal, the project's scope shifted in three ways, and we describe why each change was necessary. First, our initial plan emphasized *demonstrating* a role-differentiation auxiliary; once we found the off-the-shelf auxiliary collapses (ROLE_SIM $= +0.998$) and is task-specific (it *hurts* sparse protocol tasks), the project pivoted toward *characterizing when* memory and role augmentations help — the (M, D) decomposition and causal tests — rather than claiming a uniformly-better method. This was necessary to report honest results. Second, we added the *latent gate* (not in the proposal) after observing MARC dilutes already-sufficient observations (SMAX 2s3z, Overcooked); the gate was the cleanest way to convert a scope-conditional method into a safe drop-in. Third, we expanded the baseline and benchmark set well beyond the proposal (GRU/LSTM at two capacities, a ROMA-style PPO adaptation, the SMAX N-ladder, Overcooked, Hanabi, Coin Game) to rule out the "it's just memory / just capacity / just identity" confounds.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *NeurIPS*, 2021.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *ICML*, 2019.

Ziluo Ding, Tiejun Huang, and Zongqing Lu. Learning individually inferred communication for multi-agent cooperation. In *NeurIPS*, 2020.

Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *NeurIPS*, 2016.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI*, 2018.

Xudong Guo, Daming Shi, Junjie Yu, and Wenhui Fan. Heterogeneous multi-agent reinforcement learning for zero-shot scalable collaboration. *arXiv preprint arXiv:2404.03869*, 2024.

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 2017.

Kyungbeom Kim, Seungwon Oh, and Kyung-Joong Kim. Prism: Spectral parameter sharing for multi-agent reinforcement learning. *arXiv preprint arXiv:2602.06476*, 2026.

Chenghao Li, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. In *NeurIPS*, 2021.

Toru Lin, Jacob Huh, Christopher Stauffer, Ser Nam Lim, and Phillip Isola. Learning to ground multi-agent communication with autoencoders. In *NeurIPS*, 2021.

Bo Liu, Qiang Liu, Peter Stone, Animesh Garg, Yuke Zhu, and Anima Anandkumar. Coach-player multi-agent reinforcement learning for dynamic team composition. In *ICML*, 2021. Representative role-differentiation auxiliary of the family salvaged here; the camera-ready should cite the exact source used.

Zichuan Liu, Yuanyang Zhu, Zhi Wang, Yang Gao, and Chunlin Chen. MIXRTs: Toward interpretable multi-agent reinforcement learning via mixing recurrent soft decision trees. *arXiv preprint arXiv:2209.07225*, 2022.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *NeurIPS Datasets and Benchmarks*, 2021.

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. *JMLR*, 2020.

Alexander Rutherford et al. JaxMARL: Multi-agent RL environments in JAX. *arXiv*, 2023.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Roma: Multi-agent reinforcement learning with emergent roles. In *ICML*, 2020.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. In *ICLR*, 2021.

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 *NeurIPS*, 2022.

Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *ICML*, 2021.

A Hyperparameters and training procedure

All methods share the PPO backbone and hyperparameters below; MARC adds the role-latent components on top. Identical PPO settings are used for every `NETWORK_KIND` (vanilla, MAPPO, CDS, MARC-SELF, MARC, independent IPPO) so that comparisons differ only in architecture.

Table 6: Shared PPO hyperparameters and MARC-specific settings.

group	hyperparameter	value
PPO	learning rate	5×10^{-4} (linearly annealed)
	optimizer	Adam (ε=10^{-5}), global-norm clip 0.5
	discount γ	0.99
	GAE λ	0.95
	clip ϵ	0.2
	entropy coef	0.01
	value coef	0.5
	update epochs	4
	minibatches	16
	activation	ReLU
MPE / SMAX	NUM_ENVS	64
	NUM_STEPS	256
	total timesteps	5×10^6
Switch Riddle	NUM_ENVS	256
	NUM_STEPS	32
	total timesteps	2×10^6
MARC	history window H	16
	latent dim Z	32
	self/teammate encoder	TinyTransformer: 2 layers, 2 heads, dim 32, ff 64
	action embedding dim	16
	λ_{aux} (salvaged)	0.5; $\beta = 1$
	anneal warmup	$s(t)$ ramps $0 \to 1$ over early training

Hanabi uses the same PPO settings; the 2p/3p results in the main text use NUM_ENVS$= 64$, and the (inconclusive) 5p results use NUM_ENVS$= 16$ after an out-of-memory reduction — we report the matched-batch vanilla baseline alongside so the 5p comparison is fair (Appendix B).

B Per-cell experimental tables

Full per-cell breakdowns (per-seed returns, training time, `behavioral_sep`, and `role_similarity`) accompany the released results JSON. We highlight one correction: an

early Hanabi-5p comparison appeared to show MARC (4.3) far below vanilla (15.8), but the vanilla number used NUM_ENVS= 64 while MARC had been reduced to 16 after an OOM; at the matched NUM_ENVS= 16, vanilla (5.30 ± 1.01), MARC-ARCH (4.34 ± 0.90), and MARC-AUX (4.71 ± 0.95) are statistically tied (all in the under-trained regime), so we do not draw a Hanabi-5p conclusion. The auxiliary-harm result is therefore scoped to Switch Riddle N=10 and Hanabi 2p/3p, where the batch size is adequate.

This section also collects supporting floats referenced from the main text: the Switch Riddle N=10 training curves (Figure 3), the per-switch auxiliary return ablation (Table 8), and the full latent-flip table (Table 9).

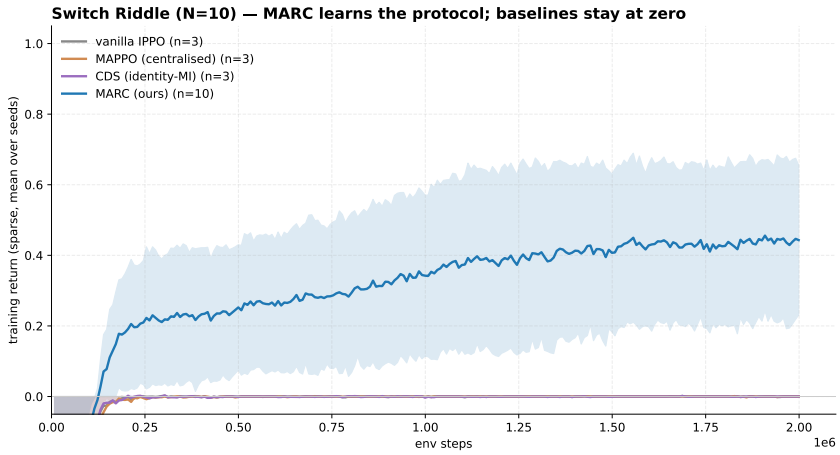

Figure 3: Switch Riddle N=10 training curves. MARC's return climbs to a learned protocol; vanilla, MAPPO, and CDS remain at exactly 0 across the full run and all seeds. Shaded band is ± 1 SD.

Table 7: Per-component effect of the salvage switches on role-latent similarity ROLE_SIM (lower = more separated), aggregated over all MPE MARC runs. NORM breaks collapse; GATE/ANNEAL temper it.

configuration	NORM	GATE/ANNEAL	ROLE_SIM (n)
no auxiliary (λ_{aux}=0)	—	—	+0.832 (111)
off-the-shelf (λ_{aux}=0.1)	off	off	+0.998 (6)
NORM only	on	off	−0.49 (4)
NORM + GATE + ANNEAL (salvaged)	on	both	**−0.122** (100)

Table 8: Per-switch return ablation (eval_return, 3 seeds). NORM is the load-bearing switch (drives the dense gain, causes the sparse harm); GATE sharpens; ANNEAL stabilizes. The full salvaged combo is the best all-round compromise but not the dense-return maximizer.

configuration	MPE-6 (less-neg better)	SR N=10 (higher better)
no auxiliary	−31.19	0.344
NORM	−28.61	0.010
NORM+GATE	**−27.68**	−0.120
NORM+ANNEAL	−30.06	0.000
NORM+GATE+ANNEAL (salvaged)	−29.33	0.224

C Switch Riddle protocol illustration

Table 9: Latent-flip test: return drop Δ_{flip} from permuting a latent across agents at evaluation (mean of 3 seeds). The self-latent is causally used wherever the role is *hidden* from the current observation (Switch Riddle, Coin Game) and largely ignored where it is observable (MPE), despite separated MPE latents — separation does not by itself imply causal use.

model / task	role in obs?	self-flip Δ	team-flip Δ
MARC-SELF, MPE `spread` N=6	yes (positions)	0.17	—
MARC-SELF, Switch Riddle N=3	no	1.95	—
MARC-SELF, Switch Riddle (Markov) N=3	no (assignment)	2.43	—
MARC-SELF, Switch Riddle (Markov) N=5	no (assignment)	4.25	—
MARC, Switch Riddle N=3	no	0.09	1.19
MARC, Switch Riddle N=5	no	0.64	1.10
MARC, Switch Riddle N=7	no	0.84	0.56
MARC, Coin Game	no	2.97	—

Table 10: Learned gate magnitudes $|g_{\text{self}}|, |g_{\text{ctx}}|$ (the two ReZero scalars, initialized at 0; mean of 3 seeds). The gate opens on every cell, so the value is *not* a win/loss readout — the loss cell SMAX 2s3z learns the *largest* magnitude. Its role is the guaranteed vanilla fallback at initialization.

cell	type	$\lvert g_{\text{self}}\rvert$	$\lvert g_{\text{ctx}}\rvert$
Switch Riddle N=3	win	0.16	0.13
MPE `spread` N=6	win	0.21	0.14
Coin Game	win	0.22	0.20
SMAX 2s3z	loss	0.65	0.29

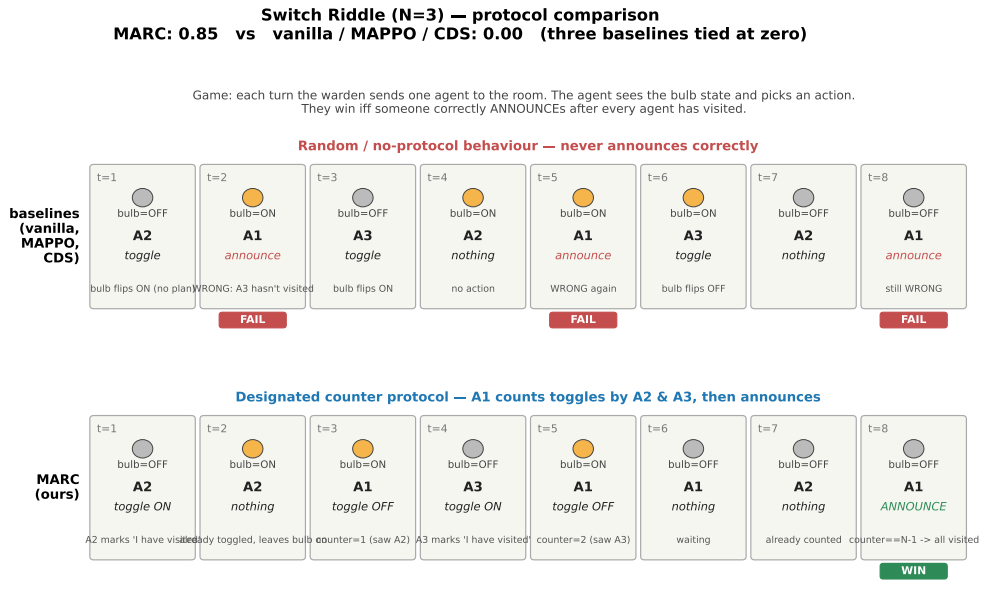

Figure 4: Comic-strip illustration of the counter protocol MARC learns on Switch Riddle (N=3) versus the random-toggle behavior of vanilla / MAPPO / CDS. One designated counter (A1) toggles OFF and counts unique toggles by A2 and A3 before announcing.