Who Gets the Reward & Who Gets the Blame? Evaluation-Aligned Post-Training for Multi-LLM Agents

Anonymous Author(s)
Affiliation
Address
email

Abstract

Teams of LLM agents are increasingly used for complex tasks, yet post-training of these multi-agent systems (MAS) is often outcome-only and ad hoc: effective learning requires *localized* signals that specify *which agent* and *which action* to adjust. We introduce an *evaluation-aligned* framework that maps

$$\text{system evaluation} \rightarrow \text{agent credit} \rightarrow \text{response-level signals},$$

closing the loop between evaluation and training. In successful episodes, the system reward is attributed fairly to agents and then distributed across their steps using context-aware labels; in failures, we localize the first error in the interaction trace to construct preference comparisons. These signals drop into standard post-training (value-free policy gradients with KL on success; preference learning on failure) without bespoke algorithms. Progress is tracked by the same evaluator that defines the signals (e.g., success rates and scores), with optional surrogate judges to reduce cost. The result is a simple, task- and model-agnostic recipe that yields interpretable artifacts (agent contributions, step helpfulness) and turns routine operation into training data. As a conceptual paper, we formalize this mapping and protocol.

1 Introduction

Large language model *post-training* refers to refining a pretrained model with outcome- and process-level signals to better follow instructions, align with human preferences, and reason over intermediate steps—e.g., RLHF, DPO, and process supervision [1, 2, 3, 4, 5]. In parallel, LLMs are increasingly deployed as *teams* of specialized agents that plan, verify, and refine one another's outputs to solve complex tasks [6, 7, 8, 9, 10]. However, post-training for such multi-agent systems (MAS) often remains ad hoc: evaluations tend to score only final outcomes, while effective learning requires *localized* signals indicating *which* agent and *which* step to improve [10, 6, 7]. We take a cooperative-games view of multi-LLM systems, where a set of role-specialized policies forms a coalition to pursue a shared objective [11, 12]. Game-theoretic tools provide principled notions of contribution and fairness, and recent works begin to explore these ideas for LLM agents [13, 14, 15]. However, existing approaches largely *stop at attribution*: they do not connect system-level scores (success or failure) to *post-training* procedures that update agents at the granularity of roles and responses.

This gap matters in practice. Small local errors can cascade across agents and derail an otherwise competent team [6]; moreover, most agents are trained independently for narrow

skills rather than for cooperative dynamics [7, 16]. The central question we address is: *How can we transform system-level evaluation into agent-level and response-level supervision that directly drives post-training for multi-agent LLMs?* Our answer is an evaluation-aligned pipeline that, in **success** episodes, allocates the system reward to agents using Shapley-style credit with explicit counterfactual baselines and then distributes each agent's credit across its individual responses via process labels; and, in **failure** episodes, localizes the *first error* in the global interaction history via an OmegaPRM-style binary search and converts that step into less-/more-preferred pairs for preference learning [4, 17, 2]. This design deliberately uses *reward-based* updates when the scalar signal is informative (success) and *preference-based* updates when the scalar is uninformative (R_{sys}=0 in failure), matching established advantages of RLHF/DPO-style training under sparse or zero rewards [1, 2, 5].

Contributions. We propose a general, model-agnostic framework that maps evaluator outcomes to agent and response supervision and plugs directly into off-the-shelf post-training. It (i) unifies cooperative-game attribution with process supervision to yield *fair, incentive-compatible* agent credits that sum to the system reward and *context-aware* response scores; (ii) extends OmegaPRM-style first-error localization to *multi-agent* traces to construct preference pairs in failure; (iii) exposes a simple interface for value-free policy gradients (success) and DPO/GRPO-style preference learning (failure), without bespoke algorithms; and (iv) improves *interpretability* by producing explicit, auditor-friendly artifacts—per-agent contribution ratios and per-step helpfulness labels—while *naturally generating data* (episodes, coalition/evaluator outcomes, first-error pairs) that accumulates during normal operation and can seed future benchmarking or surrogate models.

2 Background and Related Work

Multi-agent LLM systems. Recent surveys document rapid progress in multi-LLM workflows, infrastructure, and evaluation challenges [10, 7]. Large-scale systems (e.g., MegaAgent) show that heterogeneous teams can operate without predefined SOPs but remain brittle under coordination errors [8, 6]. Methods for reflection, verification, and planning (e.g., multi-agent reasoning and tool use) improve robustness but do not directly translate outcomes into training signals [16, 9, 18].

Credit assignment and cooperation. In classical multi-agent reinforcement learning (MARL), credit assignment is a core challenge (e.g., Multi-Agent Deep Deterministic Policy Gradients, *MADDPG*, and other multi-agent actor–critic methods) [19]. For LLM agents, Shapley-based and language-guided credit assignment are emerging: Shapley-Coop attributes cooperative gains in self-interested settings [13]; language-model–generated dense rewards accelerate MARL convergence [20, 21]; and public-goods formulations study incentives in multi-LLM teams [14]. These approaches illuminate *how* to measure or shape credit, but generally do not *route* those signals into a unified post-training protocol that updates agents at the step level.

Process supervision and preference learning. Process reward models (PRMs) score intermediate steps for single-agent reasoning [17, 3, 22, 23, 24]. OmegaPRM automates data collection via a *binary search* to find the first incorrect step [4]. In outcome-level alignment, RLHF provides reward-driven updates [1], while DPO and KTO leverage preference/comparison data without explicit reward modeling [2, 5]. Our framework *extends* process supervision to *multi-agent* interaction traces and pairs it with cooperative-game credit so that success episodes yield reward-shaped updates and failure episodes yield preference-shaped updates within the same pipeline.

Gaps our framework addresses. First, existing MAS works often treat *attribution* and *training* separately: they measure credit or encourage cooperation but stop short of translating outcomes into per-agent *and* per-response updates. Second, success/failure are not handled symmetrically: reward-based RL relies on informative scalars (satisfied by success), whereas failure offers R_{sys}=0; preference learning thrives precisely when comparisons (better/worse steps) are available. Our pipeline formalizes this separation and provides a

single route from system outcome to localized supervision across both regimes. Finally, prior efforts typically evaluate on narrow settings; our design is task- and topology-agnostic and, as a byproduct of operation, *produces* a valuable dataset (episodes, coalition outcomes, step labels, first-error pairs) useful for interpretability, reproducibility, and future surrogate modeling [25, 26, 27, 18]. Moreover, because our mapping from system outcome → agent credit → response signal does not depend on a particular model architecture, it *could* carry over to foundation-model(FM) settings by instantiating "agents" as role-conditioned modules (e.g., prompts, adapters, or heads) atop a shared backbone.

3 Proposed Framework: System→Agent→Response Signals

We propose a general framework for evaluation-driven post-training of multi-agent LLM systems. Our goal is to transform *system-level outcomes* into *agent- and response-level learning signals*, thereby closing the loop between evaluation and training. Unlike prior work that focuses either on high-level credit assignment or on single-agent post-training, our framework integrates **game-theoretic attribution** and **process supervision** to design a principled and task-agnostic protocol. *Throughout, we use "response-level" to mean **per-step, context-conditioned** signals attached to individual agent emissions $m_{i,t}$ within a full episode of n collaborating agents (i.e., supervision at the granularity of $(H_{t-1}, m_{i,t})$ rather than only at the final system output).*

3.1 Setting

We consider a multi-agent LLM system composed of a finite set of agents

$$\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}, \qquad n \triangleq |\mathcal{A}|,$$

each instantiated as an LLM policy π_i specialized for a role. Examples of roles include *planner*, *coder*, and *summarizer*, analogous to role-specialized policies in cooperative MARL. The agents collaborate in a cooperative game: given a user input $x \in \mathcal{X}$ (e.g., a natural language instruction or problem description), they exchange intermediate messages and collectively produce a system output $y \in \mathcal{Y}$ (e.g., program, plan, or solution) intended to solve the task.

Communication proceeds asynchronously along a directed interaction graph $G = (V, E)$, where vertices correspond to agents $V = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ and edges $(\mathcal{A}_i \to \mathcal{A}_j) \in E$ specify which agents may pass messages to which others. At each interaction step t, an agent $\mathcal{A}_i$ receives an input state $s_{i,t}$ (consisting of x and prior messages) and produces an output message $m_{i,t}$. A full execution trace of the system is thus a sequence of agent–message pairs

$$\tau = \{(\mathcal{A}_i, m_{i,t})\}_{t=1}^{T},$$

where T is the number of interaction rounds until termination.

The system-level output $y = f(\tau)$ is verified by an external evaluator $\mathcal{E}$, which returns an outcome of the form

$$\mathcal{E}(x, y) = \begin{cases} \texttt{fail}, & \text{if the task is unsuccessful,} \\ \texttt{success}(r), & \text{if the task succeeds with scalar score } r \in \mathbb{R}_{\geq 0}. \end{cases}$$

Here r is the scalar score attached to a successful outcome (we will map outcomes to a scalar in Sec. 3.3). This evaluation establishes two regimes: (i) in failure, we must localize which agent actions contributed to the error; (ii) in success, we must divide the scalar reward fairly among agents, avoiding degenerate competition for credit. Formally, each system run yields a tuple

$$(x, \tau, \mathcal{E}(x, y)),$$

which serves as the training signal for our framework.

3.2 Overview

Given this setting, we propose a *pipeline* that, for both successful and failing episodes, transforms the evaluator's global outcome $\mathcal{E}(x, y)$ into localized supervision. Concretely, we

(i) compute *agent-level credits* via cooperative-game attribution and (ii) refine these into *response-level signals*—i.e., per-step, context-aware supervision on $(H_{t-1}, m_{i,t})$—via process supervision, yielding $\{r_{i,t}\}$. The remainder of this section specifies this mapping from system outcome to response-level feedback; the next section shows how to plug these signals into standard post-training objectives (preference- and RL-based) for updating the policies π_i.

3.3 Success Regime: System Evaluation → Agent Credit

Objective. Recall from Sec. 3 /Setting that the (multi-LLM-agent) system's evaluator outputs $\mathcal{E}(x, y) \in \{\texttt{fail}, \texttt{success}(r)\}$ with $r \geq 0$. Define a scalar extractor score$(\cdot)$ by score$(\texttt{success}(r)) = r$ and score$(\texttt{fail}) = 0$, and let the *system reward* for the *full* multi-agent system (all n agents participating) be

$$R_{\text{sys}} \triangleq \text{score}\big(\mathcal{E}(x, y)\big).$$

In the success regime, we design a credit-allocation mechanism such that (i) each agent's local objective is aligned with improving R_{sys}, and (ii) credit is divided *fairly* to discourage free-riding and encourage productive collaboration.

Cooperative game and Shapley value. Model the interaction as a transferable-utility cooperative game on $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$. For any coalition $S \subseteq \mathcal{A}$, consider the *counterfactual* run where only agents in S contribute (others are replaced by a baseline; see below). Let

$$\mathcal{E}_S(x, y_S) \in \{\texttt{fail}, \texttt{success}(r_S)\}$$

be the evaluator's outcome for this coalition, and define the coalition value

$$v(S) \triangleq \text{score}\big(\mathcal{E}_S(x, y_S)\big) = \begin{cases} r_S, & \text{if } \mathcal{E}_S = \texttt{success}(r_S), \\ 0, & \text{if } \mathcal{E}_S = \texttt{fail}. \end{cases} \tag{1}$$

In particular, for the grand coalition $S = \mathcal{A}$ we have $v(\mathcal{A}) = R_{\text{sys}}$.

Baselines. "Replacing by a baseline" implements the counterfactual where non-members do not contribute. We use one of: (i) *no-op* (the agent emits an empty/neutral message), (ii) a fixed *reference policy* π_{ref} (e.g., a small, general-purpose LLM frozen for evaluation), or (iii) *masking* that copies the last valid state without adding content. These choices keep comparisons well-defined and reproducible.

The *Shapley value* assigns agent $\mathcal{A}_i$ its expected marginal contribution:

$$\phi_i = \sum_{S \subseteq \mathcal{A} \setminus \{i\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \Big(v(S \cup \{i\}) - v(S)\Big), \tag{2}$$

satisfying *efficiency*, *symmetry*, *linearity*, and the *null-player* property. In particular, $\sum_{i=1}^{n} \phi_i = v(\mathcal{A}) = R_{\text{sys}}$, preventing free-riding.

Normalized contribution coefficients and agent rewards. Define the (system-normalized) agent contribution ratio

$$\alpha_i \triangleq \frac{\phi_i}{\sum_{j=1}^{n} \phi_j} = \frac{\phi_i}{R_{\text{sys}}}, \qquad \sum_{i=1}^{n} \alpha_i = 1, \quad \alpha_i \in [0, 1]. \tag{3}$$

We allocate agent rewards by

$$r_i \triangleq \alpha_i \cdot R_{\text{sys}} = \phi_i, \tag{4}$$

so that $\sum_{i=1}^{n} r_i = R_{\text{sys}}$ (by (3)) and each agent maximizes its payoff by increasing its marginal contribution to the system.

3.4 Agent → Response Decomposition via Binary Labels

To leverage standard post-training techniques, we further decompose the agent-level reward r_i into *response-level* rewards for trajectories produced by the *full* system (all n agents

participating). This provides fine-grained signals suitable for preference-based or RL-style updates.

Each agent $\mathcal{A}_i$ can emit multiple intermediate responses during a trajectory. Let the global message (chat) history up to step $t-1$ be

$$H_{t-1} \triangleq \{(\mathcal{A}_{i'}, m_{i',u})\}_{u=1}^{t-1},$$

and let the response from agent i at step t be $m_{i,t} \in \mathcal{M}$. Denote by

$$\mathcal{T}_i \triangleq \{\, t \in \{1, \ldots, T\} \;:\; \text{agent } i \text{ acts (produces a response) at time } t \,\}$$

the index set of timesteps at which agent i acts within the episode, and let

$$k_i \triangleq |\mathcal{T}_i|$$

be the *number of responses produced by agent i in that episode*. After system-level allocation (4), the per-agent data for the episode can be summarized as

$$\mathcal{D}_i = \Big\{ r_i, \;\; (H_{t-1},\, m_{i,t}) \text{ for all } t \in \mathcal{T}_i \Big\}.$$

In words: we first decompose the system reward into agent-level rewards via Shapley values, then further distribute each agent's reward across that agent's individual responses within the episode.

Relation to multi-step single-agent training. At this granularity, the setting reduces to step-wise credit assignment familiar from single-agent multi-step training, where response-level supervision guides post-training. We adopt a simple, intuitive scheme that is compatible with preference-based or RL-style updates (cf. *RLHF* and preference learning [28, 1, 2]).

Binary labels (context-aware). We use a judge (human or LLM-as-judge) that evaluates a response *in its context*:

$$\mathcal{J}(H_{t-1},\, m_{i,t}) = \begin{cases} 1, & \text{if } m_{i,t} \text{ is helpful/correct given } H_{t-1} \\ & \text{(moves the trajectory toward a correct final output),} \\ 0, & \text{if } m_{i,t} \text{ is harmful/irrelevant given } H_{t-1}. \end{cases} \tag{5}$$

We write $q_{i,t} \triangleq \mathcal{J}(H_{t-1}, m_{i,t}) \in \{0, 1\}$. The context H_{t-1} is explicitly considered so that an agent is judged on whether its action was *directionally* helpful given its inputs—not merely on its surface form.[1]

Per-response allocation weights and response rewards. Let the (response-level) allocation weight for agent i at step $t \in \mathcal{T}_i$ be

$$\omega_{i,t} \triangleq \frac{q_{i,t}}{\sum_{u \in \mathcal{T}_i} q_{i,u}} \quad \text{with the convention} \quad \sum_{u \in \mathcal{T}_i} q_{i,u} = 0 \Rightarrow \omega_{i,t} = \frac{1}{k_i}. \tag{6}$$

The response-level reward is then

$$r_{i,t} \triangleq \omega_{i,t}\, r_i. \tag{7}$$

By construction,

$$\sum_{i=1}^{n} r_i = R_{\text{sys}}, \qquad \sum_{t \in \mathcal{T}_i} r_{i,t} = r_i,$$

so the system reward decomposes into agent rewards, which further decompose into response rewards.

[1] Even in successful episodes, some intermediate responses can be unhelpful; later agents may correct the trajectory. To avoid penalizing an agent for faithfully acting on a flawed context, helpfulness is judged *conditional on* H_{t-1}.

Summary of the success pipeline. At a high level, we (i) run the full multi-agent system and obtain an evaluator outcome; (ii) convert the outcome into a scalar R_{sys}; (iii) attribute R_{sys} to agents by predicting $\hat{\boldsymbol{\alpha}} = G_\theta(\tau)$ (optionally seeded by a lightweight Shapley sampler during training); (iv) decompose each agent's reward into response-level signals using the learned judge J_ϕ; and (v) apply standard post-training (preference/RL) updates with these localized signals—thereby unifying *system-level evaluation*, *agent-level attribution*, and *response-level supervision* into a single, scalable pipeline.

3.5 Failure Regime: First-Error Localization and Preference Construction

Objective. When the evaluator returns $\mathcal{E}(x, y) = \texttt{fail}$, we set the system reward

$$R_{\text{sys}} \;=\; \text{score}(\mathcal{E}(x, y)) \;=\; 0,$$

so agent-level rewards (and therefore response scores $r_{i,t}$) vanish for that episode. We nevertheless extract *response-level* supervision by localizing the *first error* in the trajectory and converting it into *less-preferred* examples for preference-based post-training (consumed in Sec. 4).

Extending process supervision to multi-agent episodes. Process supervision rewards/penalizes *intermediate steps* rather than only final answers [17]. OmegaPRM [4] automates data collection by *binary searching* a Chain-of-Thought to find the first incorrect step, enabling efficient PRM training. We extend this idea from *single-agent* traces to *multi-agent* interaction traces by running the check over the *global* history prefixes and attributing the first detected error to the agent that acted at that step.

First-error via OmegaPRM-style binary search. Let $H_t = \{(\mathcal{A}_{i'}, m_{i',u})\}_{u=1}^{t}$ denote the global prefix through step t. Define a prefix-consistency judge $\mathcal{J}_\Omega$ that returns

$$\mathcal{J}_\Omega(H_t) \in \{\texttt{OK}, \texttt{ERR}\},$$

with an approximate *monotonicity* property: once a violation appears, all longer prefixes remain erroneous (as in 4). We locate the first failing index by bisection:

$$t^\star \;=\; \min\{\, t \in \{1, \dots, T\} \;:\; \mathcal{J}_\Omega(H_t) = \texttt{ERR} \,\}, \qquad \text{found with } \mathcal{O}(\log T) \text{ judge queries.} \quad (8)$$

Let $i^\star$ be the agent that acted at $t^\star$ (i.e., produced $m_{i^\star,t^\star}$). We set the response-level label

$$q_{i^\star,t^\star} \;=\; 0, \qquad \text{and (when available) mark passing steps as } q_{j,u} = 1 \text{ for } u < t^\star. \quad (9)$$

Here $q_{i,t} \in \{0, 1\}$ is the binary *helpfulness* indicator from (5): 1 if the response is locally correct/helpful given its context H_{t-1}, and 0 otherwise. These labels supply directional supervision even when scalar rewards are absent.

Preference construction (handoff to post-training). For training, treat the first-error message as the *less-preferred* candidate. Using $c_{i^\star,t^\star} = H_{t^\star-1}$ as the conditioning context, choose a *preferred* alternative y^+ via one of: (i) a successful episode that encountered a comparable context, (ii) a post-hoc correction (human/LLM edit) of $m_{i^\star,t^\star}$, or (iii) an N-best/beam sample that passes local checks. Form pairs

$$\big(c_{i^\star,t^\star},\; y^+,\; y^- \triangleq m_{i^\star,t^\star}\big) \;\;\mapsto\;\; \text{(preferred, less preferred)}, \quad (10)$$

optionally weighted by judge confidence. These pairs are fed to preference-based objectives (e.g., DPO/GRPO) in Sec. 4. No agent-level scalar rewards are required in the failure regime.

Unification. Together with the success regime (Sec. 3.3)—where we decompose $R_{\text{sys}} > 0$ into agent- and response-level rewards—this failure pipeline provides complementary supervision: *positive* credit when the system succeeds, and *negative* preferences anchored at the first detected error when it fails.

3.6 Closing the loop (evaluation ↔ training)

With the mappings in Secs. 3.3–3.4 in place, a full iteration proceeds as follows. Given current agent policies $\{\pi_i^{(k)}\}_{i=1}^n$, we run a rollout to obtain a trace $\tau^{(k)}$ and an evaluator outcome $\mathcal{E}(x, y^{(k)})$, which we convert to a scalar $R_{\text{sys}}^{(k)} = \text{score}(\mathcal{E})$.

1. **If success** ($R_{\text{sys}}^{(k)} > 0$): compute cooperative-game attributions $\{\phi_i^{(k)}\}$ via (2), set agent rewards $r_i^{(k)} = \phi_i^{(k)}$ by (4), and distribute them across steps as response-level scores $\{r_{i,t}^{(k)}\}$ using context-aware labels $q_{i,t}^{(k)}$ from (5) and weights (6)–(7).
2. **If failure** ($R_{\text{sys}}^{(k)} = 0$): run the OmegaPRM-style bisection over global prefixes to find the first error $t^\star$ via (8), mark $q_{i^\star,t^\star} = 0$ per (9), and form contrastive pairs for the implicated context using (10). (No scalar rewards are required in this regime.)

We aggregate the resulting tuples into a supervision buffer

$$\mathcal{S}^{(k)} = \big\{(i, t,\ c_{i,t}^{(k)},\ y_{i,t}^{(k)},\ q_{i,t}^{(k)},\ r_{i,t}^{(k)})\big\}, \quad c_{i,t}^{(k)} = H_{t-1}^{(k)},\ y_{i,t}^{(k)} = m_{i,t}^{(k)}.$$

Operationally, the buffer accumulates both success-derived response scores and failure-derived preference pairs, enabling a single interface for downstream updates. *Optionally* (see Sec. **??**), after multiple iterations one may train lightweight auxiliaries to amortize attribution and labeling; we defer all efficiency details and models to the Discussion.

In the next section (Sec. 4), we show how to *plug* $\mathcal{S}^{(k)}$ into standard objectives: value-free policy-gradient updates on success episodes (using response scores) and preference-based updates on failure episodes (using first-error pairs). We then roll out the updated policies $\{\pi_i^{(k+1)}\}$ and repeat, thus closing the evaluation→training loop across both regimes.

4 From Response-Level Signals to Post-Training

Having mapped system outcomes to response-level signals $\{r_{i,t}\}$ (Sec. 3), we now show how to *plug these signals into existing post-training methods* with minimal machinery. Our interface exposes, for each agent i and step t: the context $c_{i,t} = H_{t-1}$, the produced response $y_{i,t} = m_{i,t}$, a binary/soft helpfulness label $q_{i,t}$, and a response-level score $r_{i,t}$ (nonzero only in successful episodes). We use the same interface to feed (i) *RL-style* objectives on successful episodes (using $r_{i,t}$) and (ii) *preference-based* objectives on failing episodes (using first-error pairs; Sec. 3.5). The intent is practical: readers can drop these signals into their preferred post-training stack with little adaptation.

4.1 Supervision set

For each episode, agent i, and timestep $t \in \mathcal{T}_i$,

$$c_{i,t} \triangleq H_{t-1}, \qquad y_{i,t} \triangleq m_{i,t}, \qquad q_{i,t} \in \{0, 1\} \ \text{(or soft)}, \qquad r_{i,t} \ \text{ as in (7)}.$$

Aggregating across episodes yields the supervision multiset

$$\mathcal{S} \ = \ \big\{(i, t,\ c_{i,t},\ y_{i,t},\ q_{i,t},\ r_{i,t})\big\},$$

which we reuse across the two training routes below.

4.2 RL-style post-training (success episodes)

For episodes with $R_{\text{sys}} > 0$, we use the *response scores* $\{r_{i,t}\}$ directly in a simple, value-free policy-gradient update; no learned critic is required.

Value-free policy gradient (group-relative / REINFORCE). Define a centered signal by subtracting a baseline that does not depend on the sampled action:

$$\tilde{A}_{i,t} \ = \ r_{i,t} - b_{i,t}, \qquad b_{i,t} \in \big\{\, \bar{r}_i \ \text{(per-episode mean)},\ \bar{r}_{\text{batch}} \ \text{(mini-batch mean)} \,\big\}.$$

With KL regularization to a fixed reference policy π_{ref}, the update for agent i is

$$\mathcal{L}_i^{\text{RL}} = -\,\mathbb{E}_t\big[\,\tilde{A}_{i,t}\,\log \pi_i(y_{i,t}\,|\,c_{i,t})\,\big] \;+\; \eta\,\mathbb{E}_t\Big[\,\mathrm{KL}\big(\pi_i(\cdot\,|\,c_{i,t})\;\|\;\pi_{\text{ref}}(\cdot\,|\,c_{i,t})\big)\Big]. \tag{11}$$

This objective is "plug-and-play": it accepts the response scores from Sec. 3 and uses a lightweight scalar baseline $b_{i,t}$ solely for variance reduction, which is a *scalar control variate* (e.g., a running mean); it *is not* a policy and does not generate tokens. It is distinct from the *counterfactual baseline policy* (no-op/masking/π_{ref}) used earlier when computing coalition values for attribution. We do reuse π_{ref} as a fixed reference distribution for KL regularization.

4.3 Preference-based post-training (failure episodes)

For episodes with $R_{\text{sys}} = 0$, response scores vanish ($r_{i,t} = 0$), but Sec. 3.5 provides *first-error* pairs. Construct (approximately context-matched) tuples $(c,\ y^+,\ y^-,\ w)$ with $c = H_{t^\star-1}$, $y^- = m_{i^\star,t^\star}$, y^+ a corrective alternative, w (optional weight). A standard DPO-style objective for agent i is

$$\mathcal{L}_i^{\text{pref}} = -\,\mathbb{E}_{(c,y^+,y^-,w)}\Big[w\;\log\sigma\Big(\beta\big([\log \pi_i(y^+\,|\,c) - \log \pi_{\text{ref}}(y^+\,|\,c)] - [\log \pi_i(y^-\,|\,c) - \log \pi_{\text{ref}}(y^-\,|\,c)]\big)\Big)\Big], \tag{12}$$

with GRPO/KTO variants being drop-in replacements. This route requires *no* scalar rewards—only the less-/more-preferred responses around the localized first error.

4.4 Putting it together (plug-and-play)

Training mixes the two routes across agents:

$$\mathcal{L} \;=\; \sum_{i=1}^{n}\Big(\underbrace{\mathbb{1}[R_{\text{sys}} > 0]\;\mathcal{L}_i^{\text{RL}}}_{\text{success episodes}} \;+\; \underbrace{\mathbb{1}[R_{\text{sys}} = 0]\;\mathcal{L}_i^{\text{pref}}}_{\text{failure episodes}}\Big).$$

In practice: (i) build $\mathcal{S}$ from full multi-agent rollouts; (ii) on success episodes, apply the value-free update (11) using response scores; (iii) on failure episodes, apply (12) to first-error pairs from Sec. 3.5. This section's role is connective tissue: it shows how the *signals* produced by our framework drop cleanly into widely used post-training objectives. Practitioners can swap in their preferred losses (e.g., different KL schedules, pair weighting, temperature scaling) without changing the supervision interface.

5 Evaluation Protocol

We outline a concise evaluation protocol for practitioners who adopt our framework. The core idea is to *reuse the same evaluator* $\mathcal{E}$ and *process judge* $\mathcal{J}$ used during training so that metrics and training signals remain aligned. Users can evaluate on held-out tasks with fixed prompts, tools, and interaction graph G, averaging results across random seeds.

System level: Measure end-to-end success rate and mean system reward $\overline{R}_{\text{sys}}$, optionally normalized by tokens/time/cost. Track rounds-to-success as an efficiency indicator. Probe robustness by small team perturbations (e.g., drop or shuffle agents/edges) and report safety or constraint metrics when applicable.

Agent level: Track contribution ratios $\boldsymbol{\alpha}$ before and after post-training to assess fairness and reduced free-riding. Validate attribution with light counterfactuals such as leave-one-agent-out or replacing an agent with the reference policy π_{ref}; compare observed marginal impact against assigned credit.

Response level: Report the fraction of helpful steps ($q_{i,t}$=1), the judge's agreement/calibration against a human subset, and, for failures, the accuracy and cost of first-error localization (does $t^\star$ match human diagnosis with $\mathcal{O}(\log T)$ checks?). As a practical diagnostic, measure "correction efficiency" by swapping the first-error response with a preferred alternative and re-evaluating.

Generalization and reporting: Test transfer under distribution shifts (new domains/tools or harder task compositions) and topology changes in G. Provide concise ablations for key components (credit attribution, response labeling, failure-pair construction, KL regularization). Report 95% confidence intervals over seeds, disclose compute (tokens, time, and cost), and include prompts and hyperparameters to support reproducibility.

6 Discussion and Limitations

Limitation: evaluator cost. Exact cooperative-game attribution is expensive (coalitions grow exponentially), and first-error checks add judge calls. This is the primary bottleneck *early on* [25, 26, 27].

Our practical remedy: learn from operations. A key advantage of our framework is that episodes, scores, and labels are produced *while the system runs* (no dependence on static benchmarks). After a few passes, the collected buffer enables:

- *Lightweight attribution*: train a compact episode-to-credit model to predict normalized contribution ratios $\hat{\boldsymbol{\alpha}} = G_\theta(\tau)$ in a single forward pass, using a small number of sampled Shapley targets for bootstrapping [25, 27].
- *Learned process judge*: distill a binary/soft judge $J_\phi(H_{t-1}, m_{i,t})$ from human-verified labels for fast response scoring (building on [17, 4, 3]).

Additional engineering wins include caching repeated coalitions, bandit-style sampling of informative subsets, and incremental updates when roles/topology change.

What remains and why it is acceptable. Attribution fidelity can drift under distribution shift; periodic recalibration with a *small* batch of sampled coalitions keeps G_θ honest. Judge reliability is approximate; consensus judging and spot audits mitigate errors. Despite these caveats, the *operate* $\rightarrow$ *collect* $\rightarrow$ *distill* loop turns live interaction data into efficient surrogates, making the pipeline practical at scale while remaining model- and domain-agnostic (and complementary to emerging MAS credit-assignment and collaboration work [21, 20, 13, 9, 16, 18, 6]).

7 Conclusion

We presented a practical, task-agnostic framework that turns *system-level* evaluation of multi-LLM teams into *localized, response-level* supervision and feeds it back into post-training. The pipeline first maps evaluator outcomes to **agent credit** via Shapley-style cooperative-game attribution with explicit counterfactual baselines, then refines credit into **response-level signals** using context-aware (binary or soft) process labels. These signals plug directly into off-the-shelf objectives: *value-free policy gradients with KL regularization* for successful episodes (using per-step response scores) and *preference learning* (DPO/GRPO) for failures, where we construct pairs by *localizing the first error* in the global trace via an OmegaPRM-style binary search. The result is a simple, scalable way to close the loop between evaluation and training for multi-agent LLM systems.

Contributions. Our work (i) formalizes a unified *System* $\rightarrow$ *Agent* $\rightarrow$ *Response* mapping that converts global outcomes into fair, incentive-compatible agent credit and step-level scores; (ii) extends process supervision to multi-agent traces with context-conditioned labels and first-error localization, exposing a plug-and-play interface that routes success to value-free RL updates and failure to preference-based updates while clarifying the difference between counterfactual *policy* baselines for attribution and scalar *variance-reduction* baselines for learning; and (iii) outlines a practical efficiency path—operate$\rightarrow$collect$\rightarrow$distill—where lightweight surrogates (episode$\rightarrow$credit predictors and learned judges) amortize attribution and labeling without changing the supervision format. Because the mapping is architecture-independent, it can plausibly extend to foundation-model settings by instantiating "agents" as role-conditioned modules atop a shared backbone, enabling the same system$\rightarrow$agent$\rightarrow$response supervision to fine-tune a single FM.

References

[1] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[2] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[3] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

[4] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

[5] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. KTO: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

[6] Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent LLM systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

[7] Junda He, Christoph Treude, and David Lo. LLM-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–30, 2025.

[8] Qian Wang, Tianyu Wang, Zhenheng Tang, Qinbin Li, Nuo Chen, Jingsheng Liang, and Bingsheng He. MegaAgent: A large-scale autonomous LLM-based multi-agent system without predefined SOPs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4998–5036, 2025.

[9] Yang Zhang, Shixin Yang, Chenjia Bai, Fei Wu, Xiu Li, Zhen Wang, and Xuelong Li. Towards efficient LLM grounding for embodied multi-agent collaboration. *arXiv preprint arXiv:2405.14314*, 2024.

[10] Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024.

[11] Talal Rahwan, Tomasz Michalak, Michael Wooldridge, and Nicholas R Jennings. Anytime coalition structure generation in multi-agent systems with positive or negative externalities. *Artificial Intelligence*, 186:95–122, 2012.

[12] Stéphane Airiau. Cooperative games and multiagent systems. *The Knowledge Engineering Review*, 28(4):381–424, 2013.

[13] Yun Hua, Haosheng Chen, Shiqin Wang, Wenhao Li, Xiangfeng Wang, and Jun Luo. Shapley-Coop: Credit assignment for emergent cooperation in self-interested LLM agents. *arXiv preprint arXiv:2506.07388*, 2025.

[14] Yunhao Liang, Yuan Qu, Jingyuan Yang, Shaochong Lin, and Zuo-Jun Max Shen. Everyone contributes! Incentivizing strategic cooperation in multi-LLM systems via sequential public goods games. *arXiv preprint arXiv:2508.02076*, 2025.

[15] Kavindu Warnakulasuriya, Prabhash Dissanayake, Navindu De Silva, Stephen Cranefield, Bastin Tony Roy Savarimuthu, Surangika Ranathunga, and Nisansa de Silva. Evolution of cooperation in LLM-agent societies: A preliminary study using different punishment strategies. *arXiv preprint arXiv:2504.19487*, 2025.

[16] Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. Reflective multi-agent collaboration based on large language models. *Advances in Neural Information Processing Systems*, 37:138595–138631, 2024.

[17] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[18] Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Ivan Laptev, Philip HS Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. MALT: Improving reasoning with multi-agent LLM training. *arXiv preprint arXiv:2412.01928*, 2024.

[19] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[20] Muhan Lin, Shuyang Shi, Yue Guo, Vaishnav Tadiparthi, Behdad Chalaki, Ehsan Moradi Pari, Simon Stepputtis, Woojun Kim, Joseph Campbell, and Katia Sycara. Speaking the language of teamwork: LLM-guided credit assignment in multi-agent reinforcement learning. *arXiv preprint arXiv:2502.03723*, 2025.

[21] Kartik Nagpal, Dayi Dong, Jean-Baptiste Bouvier, and Negar Mehr. Leveraging large language models for effective and explainable multi-agent credit assignment. *arXiv preprint arXiv:2502.16863*, 2025.

[22] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. ReST-MCTS*: LLM self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

[23] Jiawei Li, Xinyue Liang, Junlong Zhang, Yizhe Yang, Chong Feng, and Yang Gao. PSPO*: An effective process-supervised policy optimization for reasoning alignment. *arXiv preprint arXiv:2411.11681*, 2024.

[24] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.

[25] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & operations research*, 36(5):1726–1730, 2009.

[26] Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. Bounding the estimation error of sampling-based Shapley value approximation. *arXiv preprint arXiv:1306.4265*, 2013.

[27] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[28] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer , , or .
- means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "" is generally preferable to "", it is perfectly acceptable to answer "" provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "" or "" is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: **Yes**

   Justification: The abstract and introduction state exactly what the paper contributes: a pipeline that maps system-level evaluation to agent- and response-level signals, with (i) Shapley-based, system→agent→response credit in success and (ii) OmegaPRM-style first-error localization for failure, and then plugging these signals into standard RL-/preference-based post-training (§3, §4). We make no experimental performance claims.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: **Yes**

   Justification: §6 explicitly discusses computational cost (evaluator/Shapley), dependence on judge quality and evaluator alignment, data/operational assumptions, and

offers mitigation routes (learned contribution-ratio predictor, learned judge with calibration).

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: **N/A**

   Justification: The paper proposes a framework and uses standard definitions (e.g., Shapley value) with citations; it does not introduce new theorems requiring formal proofs.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper (regardless of whether the code and data are provided or not)?

   Answer: **N/A**

   Justification: No experiments are reported in this submission; the paper is methodological.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results?

   Answer: **N/A**

   Justification: No datasets or experimental code accompany this conceptual framework in the current submission.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details necessary to understand the results?

   Answer: **N/A**

   Justification: There are no experiments in this submission.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars or other appropriate information about statistical significance?

   Answer: **N/A**

   Justification: There are no experimental results.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources needed?

   Answer: **N/A**

   Justification: There are no experiments in this submission.

9. **Code Of Ethics**

   Question: Does the research conform to the NeurIPS Code of Ethics?

   Answer: **Yes**

   Justification: The work is a conceptual framework; it does not involve human subjects or sensitive data. We discuss potential risks and mitigations at a high level in §6.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive and negative societal impacts?

    Answer: **Yes**

Justification: §6 outlines potential benefits (more interpretable/traceable multi-agent training) and risks (misuse, misattribution, evaluator bias), with suggested safeguards.

11. **Safeguards**

    Question: Does the paper describe safeguards for responsible release of high-risk assets?

    Answer: **N/A**

    Justification: No models or datasets are being released in this submission; the paper presents a methodology.

12. **Licenses for existing assets**

    Question: Are third-party assets properly credited with licenses and terms of use?

    Answer: **N/A**

    Justification: The paper uses no external assets beyond cited literature.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented?

    Answer: **N/A**

    Justification: No new datasets/models are released in this submission.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing/human-subjects research, are instructions and compensation details included?

    Answer: **N/A**

    Justification: The work does not involve crowdsourcing or human subjects.

15. **Institutional Review Board (IRB) Approvals or Equivalent**

    Question: Were risks disclosed and approvals obtained for human-subjects research?

    Answer: **N/A**

    Justification: No human-subjects research is included.