Decoupling Local Validity and Global Effectiveness for Credit Assignment in Long-Horizon Agentic Learning

Anonymous ACL submission

Abstract

The natural-language action space of Large Language Model (LLM) agents creates a real risk of invalid outputs (e.g., API rejections, parsing errors). Consequently, in Reinforcement Learning (RL) for long-horizon LLM agents, learning to generate a locally valid action in each turn is as crucial as selecting a globally effective one. However, this requirement was overlooked by the prevailing additive paradigm for credit assignment in agentic RL. Specifically, it computes an action's advantage by summing an estimated local action score and the global trajectory score. This paradigm assigns a "contribution" score to all actions regardless of their validity, allowing invalid actions could be assigned positive advantages, especially in positive trajectories. To address this, we propose **Multiplicative Advantage via Decoupled Signals (MADS)**, a novel credit assignment framework. MADS computes the action-level advantage by decoupling two orthogonal signals: a local validity signal derived from fact-based checks, and a global effectiveness advantage from group-relative trajectory performance. Crucially, the final action-level advantage is then calculated by local signal to multiplicatively gate the global advantage. This ensures the action's validity is correctly considered in the credit assignment process. Experiments demonstrate that our method improves training stability and achieves SOTA performance on long-horizon LLM agent benchmarks. Code of MADS has been uploaded in the Supplementary Material.

1 Introduction

A key frontier for Large Language Model (LLM) agents lies in their ability to solve long-horizon tasks (Yao et al., 2022), often requiring a long chain of interactions to navigate complex environments (e.g., over 10 interactions). Reinforcement Learning (RL) has emerged as a standard for training these agents, enabling them to learn from environmental feedback. Unlike traditional RL agents with discrete action sets, language-based agents face a unique challenge in their vast, language-based action space. They generate actions as text sequences (e.g., code, API calls), creating a fundamental dual requirement for every action: it must be both **locally valid** (e.g., syntactically correct) to be executable, and **globally effective** to contribute to the final task goal. This yields a conditional dependency: $P(Effectiveness|action) \propto P(Valid) \cdot P(Effectiveness|Valid)$. As emphasized by (Zhao et al., 2025), the validity of each turn's action is crucial for long-horizon agents.

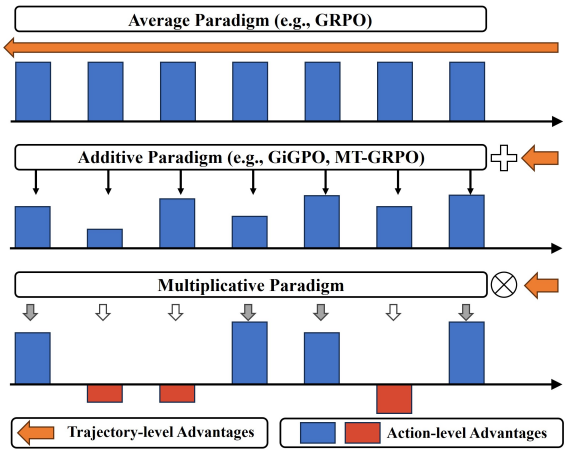

Figure 1: Illustration of Different Credit Assignment Paradigms in a Successful Trajectory.

However, we observe that this essential requirement for local action validity has been largely overlooked by the prevailing credit assignment methods in agentic RL, which we term the *additive paradigm* (as illustrated in Fig. 1). This paradigm computes an action's advantage by summing an estimated local action score with the global score of its parent trajectory, through learned progress estimators (Wu et al., 2025), reward decomposition (Zheng et al., 2025), or sub-trajectory (Feng et al., 2025) comparisons. This design is blind to actions' validity, and its core flaw is assigning

a “contribution" score to all actions regardless of whether they are executable or not. This leads to severe credit misassignment: an unequivocally invalid action may receive a positive advantage simply by appearing in a successful trajectory (shown in the second row of Fig. 1), violating the conditional dependency between validity and effectiveness. Such flawed signals fundamentally destabilize the learning process, forcing the agent to expend valuable samples learning basic syntax from ambiguous feedback.

To address this, we propose **Multiplicative Advantage via Decoupled Signals (MADS)**, a novel credit assignment framework that computes an action-level advantage by decoupling two orthogonal signals derived from environmental feedback. The first signal, from an action-level sign provider, determines the atomic correctness (a positive/negative sign) of an action. This signal is derived from direct environmental feedback and refined with the model temporal dynamics (e.g., error correction) and penalize unproductive behaviors (e.g., repetition). The second, from a trajectory-level magnitude provider, employs group-relative methods to evaluate the quality of an entire trajectory, providing the trajectory-level advantage.

These two components are fused via a multiplicative gating mechanism into the final action-level advantage. This design ensures that an action’s factual validity strictly governs the action-level advantage’s polarity. It guarantees that invalid actions receive negative scores even in highly successful trajectories, while valid actions within failed trajectories are selectively preserved as positive learning samples via a dynamic gate. Crucially, this mechanism also gives rise to an implicit curriculum: early in training, the agent is primarily rewarded for mastering foundational skills (i.e., producing valid actions); as it matures, the learning objective implicitly transitions to demanding the strategic composition of these actions to achieve high-quality trajectories. This multiplicative gating mechanism corrects the core flaw of the additive paradigm: an action’s factual invalidity can no longer be masked by overall trajectory success. This eliminates a critical vector of credit misassignment by design.

Our contributions are threefold:

- We identify the dual challenge in agentic learning: achieving both local action validity and global strategic effectiveness.
- We propose Multiplicative Advantage via Decoupled Signals (MADS), a new framework that uses a fact-based signal to gate the credit flowing from a trajectory’s global outcome.
- We demonstrate empirically that MADS leads to significant gains in training stability and sample efficiency, achieving SOTA performance on standard agentic benchmarks.

2 Related Work

The task of training multi-turn LLM agents has recently seen a surge of interest, with Reinforcement Learning (RL) emerging as a powerful paradigm for enhancing long-horizon decision-making. Our work builds upon and diverges from several key research threads in this area.

2.1 Trajectory-Level Reinforcement Learning

Initial applications of RL to LLMs often operated at the trajectory level. Methods like Proximal Policy Optimization (PPO) (Schulman et al., 2017) and its variants, including Group-Relative Policy Optimization (GRPO) (Shao et al., 2024), treat an entire sequence of actions as a single data point. The agent receives a sparse reward signal only upon task completion, and this signal is uniformly attributed to all actions within the trajectory. This approach suffers from severe credit misassignment noise; it invariably rewards erroneous or suboptimal actions within a successful trajectory and penalizes correct exploratory steps within a failed one. The work by (Shi et al., 2025) systematically identified this issue, noting that without fine-grained, reasoning-oriented feedback, agents can fall into “echo traps" and fail to develop robust, multi-step reasoning.

2.2 Fine-Grained Advantage Estimation

A category of methods refines the advantage calculation at a sub-trajectory level, but remains within the foundational additive framework of the Bellman equation. GiGPO (Feng et al., 2025) extends the group-relative concept to the step level by comparing actions from different trajectories at aligned “anchor states." ARPO (Dong et al., 2025) uses model uncertainty (entropy) as a heuristic to identify critical steps and attribute credit accordingly. SPO (Huang et al., 2025) proposes a segment-level advantage estimation as a compromise between turn-level and trajectory-level granularity. While these methods offer more sophisticated estimation techniques, they are still estimating a single, conflated score for each step that is implicitly added

to form the trajectory's total value. They do not possess a mechanism to enforce the binary validity of an action.

3 Method

As discussed in Section 1, the action space of LLM agents differs fundamentally from traditional RL. Therefore, the expected utility of an action a_t is not a monolithic scalar but follows a **conditional dependency structure**:

$$P(\text{Eff}|a_t) \propto P(\text{Valid}) \cdot P(\text{Eff}|\text{Valid}) \quad (1)$$

where Eff denotes the strategic effectiveness of action a_t.

However, the prevailing additive paradigm for credit assignment ($A_{\text{action}} = A_{\text{global}} + \lambda A_{\text{local}}$) conflates the prerequisite of validity with strategic utility. The violation of the conditional dependency structure leads to credit misassignment, as a large positive A_{global} from a successful trajectory can incorrectly assign positive credit to an invalid action.

To address this, our MADS framework aims to **recover the conditional structure** defined in Eq. 1. We operationalize this by mapping the probabilistic components to distinct advantage signals: the Local Validity Signal (S_{local}) and the Trajectory-level Advantage (A_{global}). Consequently, we formulate the final advantage not as a sum, but as a multiplicative interaction:

$$A_{\text{MADS}}(a_i) = \mathcal{G}(S_{\text{local}}(a_t) \times A_{\text{global}}(\tau)) \quad (3)$$

where $\mathcal{G}(\cdot)$ is a **gating function** to fuse the S_{local} and A_{global}. This formulation structurally enforces validity as a necessary precondition: $S_{\text{local}}(a_t)$ acts as a proxy for the probability that action a_t is valid, while $A_{\text{global}}(\tau)$ serves as a proxy for the strategic effectiveness of the entire trajectory τ. when the action is invalid, the total advantage A_{MADS} correctly collapses, preventing the misassignment of credit regardless of the global trajectory outcome.

3.1 Trajectory Segmentation and Reinforcement Learning

Our training process begins with collecting a set of m trajectories, $\{\tau_1, \tau_2, \ldots, \tau_m\}$, for each given task. A single trajectory τ_i of length T is a sequence of states and actions: $\tau_i = (s_1, a_1, s_2, a_2, \ldots, s_T, a_T)$. To form a training batch for our language model-based policy π_θ, we segment each trajectory into action-level samples. Each sample (p'_t, a_t) consists of the action a_t and its corresponding prompt p'_t, which comprises the entire history up to that point, i.e., $p'_t = (s_1, a_1, \ldots, s_t)$. This segmentation process can be visualized as shown in Eq. 2., where each action-level advantage $A_{\text{action},t}$ is computed at every step using our MADS.

These action-level advantages are then used to update the policy parameters θ using the Proximal Policy Optimization (PPO) algorithm. Following prior work (Chen et al., 2025a) that suggests removing the KL-divergence term can be beneficial for agentic learning, we optimize the following objective function:

$$\mathcal{L}^{\text{PPO}}(\theta) = \mathbb{E}_t \Bigg[\min \Bigg(\rho_t(\theta)\hat{A}_t, \text{clip}\left(\rho_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \Bigg) \Bigg] \quad (4)$$

where $\rho_t(\theta) = \frac{\pi_\theta(a_t|p'_t)}{\pi_{\theta_{\text{old}}}(a_t|p'_t)}$ is the probability ratio. The term $\hat{A}_t$ denotes the token-level advantage required by the PPO objective, which is derived from $A_{\text{action},t}$. Specifically, $A_{\text{action},t}$ serves as the primary turn-level credit signal used as a target for a critic model, which in turn computes the final, lower-variance token-level advantage $\hat{A}_t$ via Generalized Advantage Estimation (GAE) (Schulman et al., 2015). While GAE traditionally takes rewards r_t as input, our $A_{\text{action},t}$ can be interpreted as a pre-computed, single-step advantage estimate. This estimate is then refined by the critic and smoothed across the trajectory via GAE to produce the final, low-variance token-level advantage $\hat{A}_t$. This two-stage estimation process is central to our method's stability, ensuring both structural credit correctness and stable policy gradients.

3.2 Decouple Local Validity and Global Effectiveness

The core principle of MADS is the decomposition of the complex credit assignment problem into two orthogonal sub-problems, addressed by two distinct signal providers.

3.2.1 Action-Level Validity Estimation

This component generates a local signal $S_{\text{local},t}$ to evaluate the atomic correctness of action a_t, serving as the necessary condition for global utility.

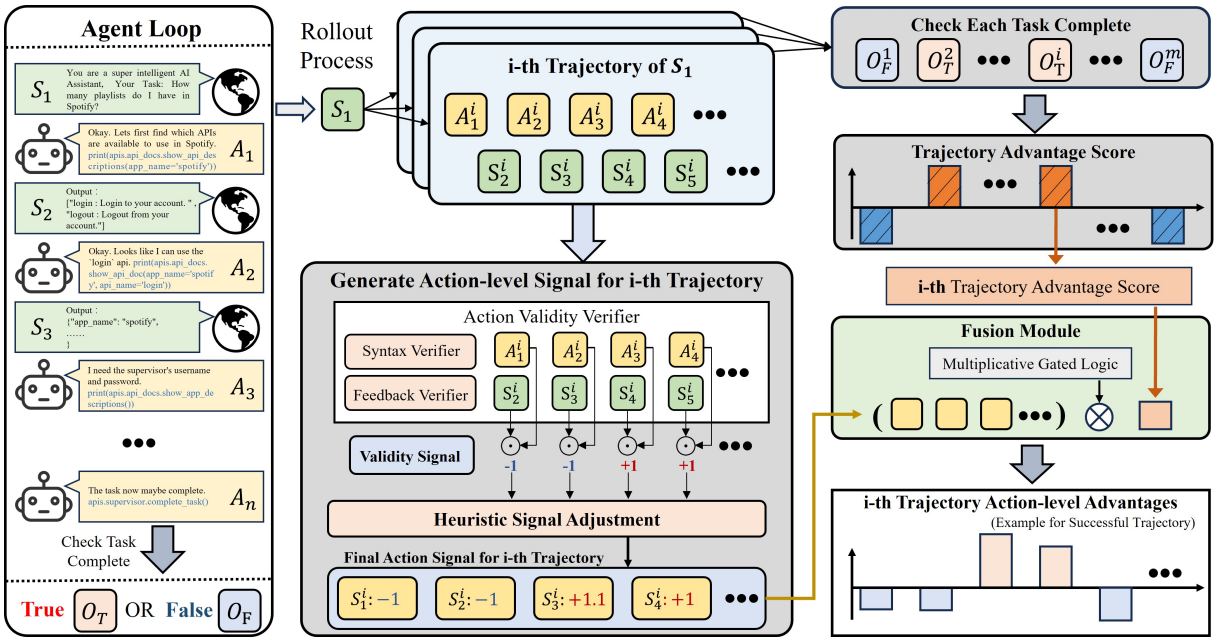

Figure 2: Example of **MADS (Multiplicative Advantage via Decoupled Signals)** calculation in a successful Trajectory. The example in the failure trajectory can be seen in App. A.1.

$$\tau = [s_1, a_1, \ldots, s_T, a_T] \quad \longrightarrow \quad \left\{ \begin{array}{ll} (\text{prompt: } s_1, & \text{response: } a_1) \rightarrow A_{\text{action},1} \\ (\text{prompt: } [s_1, a_1, s_2], & \text{response: } a_2) \rightarrow A_{\text{action},2} \\ \vdots & \vdots \\ (\text{prompt: } [s_1, a_1, \ldots, s_T], & \text{response: } a_T) \rightarrow A_{\text{action},T} \end{array} \right\} \quad (2)$$

We formulate the local signal as a composition of a fundamental validity check $\mathcal{V}$ and auxiliary regularization $\mathcal{R}$:

$$S_{\text{local},t} = \mathcal{V}(a_t, s_{t+1}) + \mathcal{R}_{\text{aux}}(\tau_{<t}, a_t) \quad (5)$$

where $\mathcal{V}(\cdot) \in \{-1, +1\}$ represents the **fundamental validity**. It is a binary indicator derived from immediate environmental feedback (e.g., execution errors, parser exceptions) and strict format constraints, detailed in App. A.7. Crucially, $\mathcal{V}(\cdot)$ exploits the objective nature of formal action spaces (e.g., code, API calls). Unlike semantic rewards which are sparse and task-specific, syntactic validity is dense and universally verifiable through standard execution protocols (e.g., standard error streams, exception codes). This ensures that our signal exhibits high transferability across diverse interactive domains with minimal adaptation.

The term $\mathcal{R}_{\text{aux}}$ incorporates **inductive biases** to regulate exploration and model temporal dynamics. Specifically, it includes (1) an *entropy regularization* term to penalize excessive action repetition, thereby preventing policy collapse into local minima; and (2) a *local consistency* term that rewards immediate recovery from errors, modeling the local Markovian dependency of self-correction. The specific structure for $\mathcal{R}_{\text{aux}}$ has been detailed in App. A.2.

3.2.2 Trajectory-Level Magnitude Provider

The effectiveness magnitude in our multiplicative paradigm is the **trajectory-level advantage**, A_{global}. This advantage is derived from the **trajectory-level effectiveness signal** (A_{global}) mentioned in our framework's overview, which fundamentally captures the strategic quality of an entire trajectory τ_i relative to its peers.

In practice, this signal is initially represented by a binary reward score, R_{τ_i}, based on task completion. Following recent advancements in training agents for long-horizon tasks (Chen et al., 2025a), we then transform these absolute scores into a normalized, zero-sum advantage that provides a more stable learning signal by focusing on relative performance. Specifically, the trajectory-level advantage for τ_i is calculated as:

$$A_{\text{global}}(\tau_i) = \frac{m}{m-1} \left(R_{\tau_i} - \frac{1}{m} \sum_{j=1}^{m} R_{\tau_j} \right) \quad (7)$$

$$A_{\text{action},t} = \begin{cases} S_{\text{local},t} \cdot |A_{\text{global}}(\tau(t))| & \text{if } \text{sign}(S_{\text{local},t}) = \text{sign}(A_{\text{global}}(\tau(t))) \\ \gamma \cdot S_{\text{local},t} \cdot A_{\text{global}}(\tau(t)) & \text{if } A_{\text{global}}(\tau(t)) > 0 \text{ and } S_{\text{local},t} < 0 \\ \gamma \cdot g_t \cdot S_{\text{local},t} \cdot |A_{\text{global}}(\tau(t))| & \text{if } A_{\text{global}}(\tau(t)) < 0 \text{ and } S_{\text{local},t} > 0 \end{cases} \quad (6)$$

3.3 Multiplicative Advantage Gating

The action-level advantage, $A_{\text{action},t}$, is computed via the conditional multiplicative process outlined in Eq. 3. The core logic is to use the action's local validity signal (S_{local}) to gate the magnitude of the trajectory-level advantage (A_{global}). This ensures the action-level advantage's sign is dictated by factual correctness, while its magnitude is scaled by strategic effectiveness. The gating logic is partitioned into three distinct scenarios based on the alignment of the local signal's sign and the trajectory's outcome:

1. **Concordant signals (sign(S_{local}) = sign(A_{global})):** When an action's validity aligns with the trajectory's result (e.g., both the action and the trajectory are either correct or incorrect.), the credit assignment is straightforward multiplication.

2. **Successful trajectory, invalid action ($A_{\text{global}} > 0, S_{\text{local}} < 0$):** The action must be penalized to discourage it, but the penalty is dampened. This is achieved by multiplying the signals and applying a scaling coefficient.

3. **Failed trajectory, valid action ($A_{\text{global}} < 0, S_{\text{local}} > 0$):** This is the most crucial and ambiguous case. We introduce a stochastic gating operator, g_t, which determines the sign of the final advantage.

This fusion logic is formally captured in the Eq. 6, where $\tau(t)$ denote the trajectory containing the action at turn t and $\gamma \in (0, 1]$ scales coefficients that reduce the magnitude in cases of sign conflict. More detail has shown in Alg. 1.

The key component is the gating operator $g_t \in \{-1, 1\}$, which is sampled based on the agent's overall performance (i.e., task success rate and action accuracy rate). The function of this gate is to dynamically control the balance of positive and negative training samples for locally correct actions, preventing the model from being rewarded for valid but strategically poor choices, especially in later stages of training. The mechanism of g_t is detailed in the next section.

3.4 Adaptive Signal Denoising via Stochastic Gating

As the model quickly learns to generate valid actions, we observed a severe data imbalance where positive local signals vastly outnumber negative ones (e.g., 90% vs 10%), even within unsuccessful trajectories. This hinders the learning of long-term strategy, as the model is not sufficiently penalized for valid actions in a failure trajectory.

Without dynamic regulation, this imbalance leads to a critical degradation in the **Signal-to-Noise Ratio (SNR)** of the reward signal. The overwhelming volume of positive rewards for valid actions begins to mask the penalty for the trajectory-level failure, causing the agent to stagnate in local optima (e.g., same valid actions) rather than improving global strategy.

The fundamental issue lies in the intrinsic **ambiguity** of valid actions within failed trajectories. They possess a dual nature depending on the training stage:

- **Early Stage (Signal):** When the agent is weak, these samples are critical *positive signals* for learning basic syntax, preventing policy collapse.

- **Late Stage (Noise):** As validity saturates, these samples become *false positive noise* relative to the strategic objective. High retention of such noise dilutes the gradient direction towards task completion.

To address this, we introduce the gating operator g_t to function as an **implicit curriculum** that manages this SNR transition. By probabilistically flipping the sign of these ambiguous samples, g_t effectively restores the **balance between positive and negative samples**.

For any given failed trajectory τ, a single probabilistic decision is made: either *all* locally valid actions within τ retain their positive sign, or they are *all* flipped to negative. This approach treats the trajectory's outcome as a holistic strategic failure, ensuring that all contributing actions receive a coherent learning signal.

$$g_t \sim \mathcal{B}_{\{\pm 1\}}(p), p = f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}}) \quad (8)$$

,where C_{batch} is the mean task completion rate, V_{batch} is the mean action validity rate for the current batch, and θ means the transformation threshold of p_{retain}. The primary objective of $f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}})$ is to guide the sample ratio towards a balanced state as training progresses, more detials are shown in A.5. This dynamic sample balancing naturally gives rise to an **implicit two-stage curriculum**:

- **Stage 1: Learning Action Validity.** In early training, when agent performance is low, p_{retain} is high. This allows the model to learn fundamental, locally valid actions from a sufficient pool of positive examples.

- **Stage 2: Learning Strategic Composition.** As the agent's proficiency increases, p_{retain} decreases. This forces the agent to focus on learning the correct sequencing and strategic use of those valid actions to achieve global success.

4 Experiment

4.1 Environments

To comprehensively evaluate the effectiveness of our MADS, we conduct experiments on two challenging benchmarks designed for multi-turn LLM agents: ALFWorld and AppWorld.

ALFWorld (Shridhar et al., 2020) is a benchmark that requires agents to perform complex, long-horizon tasks in a simulated text-based household environment. We report performance on three difficulty splits: L0, L1, and L2. Other settings are detailed in App. A.7.

AppWorld (Trivedi et al., 2024) is a recently proposed benchmark that evaluates agents on their ability to perform realistic, day-to-day digital tasks by interacting with a suite of simulated applications. We evaluate on its three official difficulty settings: Easy, Medium, and Difficult. Other settings are detailed in App. A.7.

4.2 Models and Baselines

All finetuning experiments are conducted on two powerful open-source models, **Qwen2.5-7B** (Qwen et al., 2025) and **Qwen3-8B** (Yang et al., 2025), to serve as the agent's backbone. To situate our results within the broader landscape, we also include performance metrics from the proprietary **GPT-4o** model and a previously reported strong system, **DS-R1** (DeepSeek-AI et al., 2025), as reference points for the SOTA. Our core comparisons are against a suite of finetuning baselines applied to the same models. We compare against several methods applied to the same base models for a fair and direct comparison: ReAct(Yao et al., 2022), SFT, GRPO (Shao et al., 2024), GiGPO (Feng et al., 2025), Loop (Chen et al., 2025a), detailed in App. A.6.

4.3 Implementation Details

We employ the AdamW optimizer with a constant learning rate of `5e-7`. The specific hyperparameters for our Multiplicative Advantage via Decoupled Signals (MADS) framework are set as described in the Method section. For the main hyperparameter, We set the scaling coefficient $\gamma = 1.0$. For the schedule function $f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}})$ shown in (8), we set the hyperparameters to maintain a 1:1 ratio of positive to negative samples, detailed in App. A.7. Other settings are also provided in App. A.7.

4.4 Results and Analysis

As presented in Table 1, our proposed method consistently achieves SOTA performance across all evaluated settings on both the ALFWorld and AppWorld benchmarks. This demonstrates the effectiveness and robustness of our MADS framework in training capable LLM agents for complex, multi-step interaction tasks.

On the ALFWorld benchmark, while some strong baselines perform well on simpler tasks, their effectiveness noticeably drops on the more challenging L2 split. This is because L2 tasks demand longer interaction trajectories, where the correctness of each action is critical. In contrast, MADS remains robust across all difficulty levels and improves substantially on L2, indicating that early reinforcement of syntactically valid, executable actions fosters deeper and more generalizable competence.

These advantages are decisive on AppWorld. Fine-grained credit-assignment methods such as GiGPO depend on structured per-turn state descriptions to compute granular advantages. AppWorld's complex, interactive interface does not expose such a well-parsed state, making these methods **directly inapplicable** and highlighting a broader challenge in agentic RL: reward designs frequently lack generality across realistic environments. MADS, by contrast, is broadly applicable. Its action-level va-

Table 1: Performance comparison across ALFWorld and AppWorld benchmarks. T/A means the signal comes from task completion and the action correction signal.

Model	Method	Signal Level	Signal Source	ALFWorld			AppWorld		
				L0	L1	L2	Easy	Medi	Diff
GPT-4o	-	-	-	57.3	66.0	68.8	73.7	32.1	20.1
DS-R1	-	-	-	68.8	70.2	67.3	85.6	51.2	29.3
Qwen2.5-7B	ReAct	-	-	23.1	28.5	27.0	20.2	6.1	0.0
	SFT	-	-	63.3	57.0	37.5	28.7	9.1	0.0
	GRPO	traj	Task	79.3	77.3	52.3	54.1	10.1	6.2
	GiGPO	action	Task	89.5	**90.2**	67.2	-	-	-
	Loop	token	Task	-	-	-	59.6	14.3	6.1
	MADS	action	T/A	**90.4**	89.7	**85.6**	**76.1**	**19.6**	**9.4**
Qwen3-8B	ReAct	-	-	28.1	32.5	29.2	35.1	8.4	0.0
	SFT	-	-	62.6	56.2	39.6	36.7	11.8	0.0
	GRPO	traj	Task	80.2	72.6	60.7	45.6	12.1	6.6
	GiGPO	action	Task	88.7	**91.2**	68.7	-	-	-
	Loop	token	Task	-	-	-	63.2	18.3	8.7
	MADS	action	T/A	**91.6**	90.8	**82.6**	**75.4**	**22.8**	**10.2**

lidity signal (S_{local}) is derived from generic, universally available feedback (e.g., execution errors or None responses) and lightweight syntactic checks, requiring no environment-specific state parsing. As a result, MADS can be applied seamlessly in unstructured environments like AppWorld, where it sets a new performance bar and significantly outperforms all baselines.

In summary, our empirical results strongly validate our central hypothesis: by strategically decoupling the signal into local validity and global effectiveness, MADS not only achieves superior and more generalizable performance but also offers a robust and universally applicable framework.

4.5 Analysis of Training Dynamics

To understand MADS's effectiveness, we analyze training dynamics by comparing MADS, GRPO, and Loop baselines on the AppWorld benchmark, tracking both action success rate and task success rate. Fig. 3 illustrates these learning curves, highlighting MADS's superior efficiency and stability.

The top plot of Fig. 3 depicts the task success rate. MADS (red curve) exhibits a stable and consistent upward trend, achieving a much higher final success rate. This stability stems from the agent's early mastery of valid actions, providing a robust foundation for strategic learning. Conversely, Loop (blue) and GRPO (purple) show high volatility and inconsistent progress throughout training. This instability is a direct consequence of credit misassignment, where the conflated signals lead to contradictory policy updates. MADS's design effectively mitigates this, resulting in more reliable and efficient learning of task-level objectives. The bottom plot of Fig. 3 shows the action success rate. MADS (orange curve) demonstrates rapid and significant improvement, quickly reaching over 80% validity. This reflects the efficacy of our explicit action-level validity signal (S_{local}), which enables efficient learning of correct action syntax. In contrast, Loop (green) and GRPO (brown) show minimal improvement, largely stagnating around 40%. Their reliance on a noisy, trajectory-level advantage makes it difficult to learn basic action correctness.

4.6 Ablation Studies

To dissect the contribution of each key component within our MADS framework, we conduct a series of ablation studies on the AppWorld using the Qwen3-8B model. The results are presented in Table 2.

Effect of the Gating Mechanism. In the first ablation, we remove the stochastic gating operator g_t from our fusion logic (Eqn. 6). This variant, labeled "- w/o Gating Mechanism", treats all valid actions in failed trajectories as positive learning samples (g_t is fixed to $+1$). As observed in Table 2, this leads to a significant performance drop. This result validates our hypothesis from Section 1: without the gate, the model is overwhelmed by a massive imbalance of positive rewards for actions that are locally correct but strategically poor. The gating mechanism, which forms the basis of our implicit

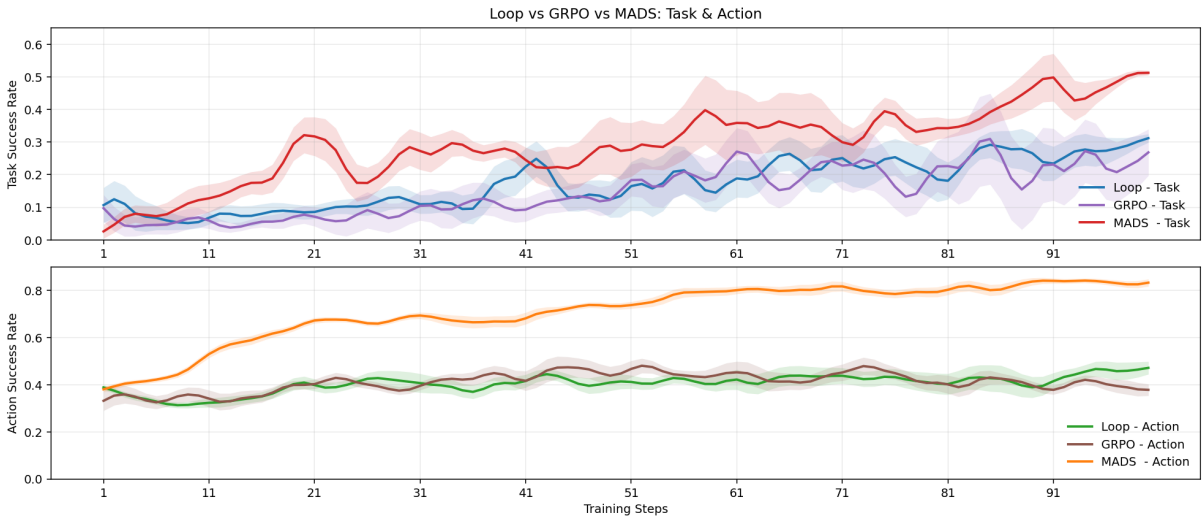

Figure 3: Training dynamics on AppWorld. Top plot: Task Success Rate. Bottom plot: Action Success Rate. MADS (red/orange) consistently outperforms Loop (blue/green) and GRPO (purple/brown) in both metrics, showing faster learning and greater stability.

Table 2: Ablation study of MADS on the AppWorld benchmark. Each row represents the removal of a specific component from our full model.

Method	Easy	Medium	Difficult	Avg.
MADS (Full Model)	**75.4**	**22.8**	**10.2**	**36.1**
- w/o Gating Mechanism (g_t)	60.1	10.3	5.2	25.2
- w/o Critic (GRPO-style Update)	71.1	19.6	8.9	33.2
- w/o Repetition Penalty	15.2	0.0	0.0	5.1
- w/ KL Divergence	72.6	19.3	10.1	34.0

curriculum, is crucial for compelling the agent to transition from learning syntax to learning strategy.

Effect of the Critic Model. Next, we assess the role of the critic and its token-level advantage estimation in PPO. We replace our critic model with a GRPO-style update rule for every token. The resulting performance degradation underscores the value of the critic in reducing the variance of policy gradients. The critic provides a more nuanced, token-level signal that stabilizes training, a benefit that is especially pronounced in the complex, high-variance environment of language-based agents.

Effect of the Repetition Penalty. Removing the repetition penalty results in a catastrophic performance collapse. Without the penalty, the agent quickly learns to exploit a flaw in its local optimization objective. It discovers that it can reliably accumulate positive rewards by repeatedly issuing a single, known-valid action, regardless of its strategic relevance, which lets the agent become a repeating machine.

Effect of KL Divergence. Our method, following prior work (Chen et al., 2025a), omits the KL divergence term commonly found in the PPO objective function. To verify this design choice, we reintroduce it in the "- w/ KL Divergence" variant. The results show a noticeable decline in performance. This suggests that the KL penalty, which constrains the policy from moving too far from the reference policy, can be overly restrictive for agentic finetuning.

Hyperparameter Sensitivity. We defer the detailed analysis of hyperparameters α, β, and γ to Appendix A.3 due to page limits.

5 Conclusion

We introduced Multiplicative Advantage via Decoupled Signals (MADS), a credit assignment framework that considers local action validity and global effectiveness and fuses them multiplicatively to prevent rewarding invalid actions. A fact-based local sign gates a group-relative global magnitude, yielding an implicit curriculum that first rewards producing valid actions and then prioritizes strategic composition. MADS delivers SOTA performance and markedly improved stability and sample efficiency.

6 Limitations

MADS relies on deterministic, parseable environment feedback (e.g., API execution results or error messages) to derive the local validity signal. While this covers a broad range of practical agent tasks, its applicability may be limited in environments where action correctness is ambiguous or cannot be reliably determined from immediate feedback.

References

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. 2025a. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*.

Yao Chen, Zhaofeng He, Geyu Lin, Zhidong Zhang, Hong-Ning L. Dai, Dong Li, and Wei Lo. 2025b. Spa-rl: Reinforcing llm agents via stepwise progress attribution. *Preprint*, arXiv:2505.20732.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, and 1 others. 2025. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*.

Tianle Gao, Jiatong Sun, Zishuai Geng, Wen-Ding Li, Xipeng Qiu, Yitao Liu, and Weizhe Yuan. 2025. Process reward models for llm agents: Practical framework and directions. *Preprint*, arXiv:2502.10325.

Jing-Cheng Huang, Tian-Shuo Li, Shiming Chen, Weijun Li, Pin-Yu Chen, and Chin-Yew Lin. 2025. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *Preprint*, arXiv:2505.23564.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Le Shi, Sijia Liu, Bowei He, Zhipeng Lucas Wu, Yuandong Tian, and Xiangjue Dong. 2025. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *Preprint*, arXiv:2504.20073.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076.

Zheyuan Wu, Zherui Li, Ziyi Lou, Chen Zhan, Xian-Ling Chen, and Chunyan Miao. 2025. SPA-RL: Reinforcing LLM agents via stepwise progress attribution. *arXiv preprint arXiv:2505.20732*.

Guofu Xie, Yunsheng Shi, Hongtao Tian, Ting Yao, and Xiao Zhang. 2025. Capo: Towards enhancing llm reasoning through generative credit assignment. *Preprint*, arXiv:2508.02298.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Tushar Avin, Jimmy Ba, and Quoc V Le. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Weize Zhao, Zeqi Liu, Zihan Wu, Shuyuan Xu, Yile Sun, Kai Zheng, and Yue Dong. 2025. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Han Zheng, Weize Chiang, Zeqi Liu, Zihan Wu, Kai Zheng, and Yue Dong. 2025. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. *arXiv preprint arXiv:2505.11821*.

A Appendix

A.1 MADS calculation in a failure Trajectory

To provide a comprehensive understanding of the MADS framework, we present an illustrative example of its advantage calculation process within a failed trajectory, as depicted in Fig. 4. This scenario serves as a direct contrast to the successful trajectory shown in Fig. 2 of the main paper, highlighting MADS's core mechanism for preventing credit misassignment for actions that are locally valid but strategically ineffective.

This example demonstrates how MADS's multiplicative gating and implicit curriculum work in synergy. The mechanism ensures that in early stages, the agent is not discouraged from exploring and learning correct syntax. However, as it matures, the learning objective implicitly shifts, forcing the agent to move beyond mere syntactic correctness and master the strategic composition of actions required for long-horizon success. This prevents the model from reinforcing "echo traps" of valid but useless actions.

A.2 Action-Level Sign Provider Implementation Details

(1) Fundamental Validity Check ($\mathcal{V}$): We first establish a binary signal, $\mathcal{V} \in \{-1, +1\}$, which serves as a hard constraint on action executability. Let s_{t+1} denote the environment execution feedback following action a_t. We define $\mathcal{V}(a_t, s_{t+1})$ as:

where Ω_{syntax} denotes the set of actions satisfying predefined structural and formatting constraints (e.g., correct code encapsulation), and $\mathcal{E}_{\text{runtime}}$ represents a set of universal error patterns inherent to the execution environment (e.g., `SyntaxError`, `TimeOut`, or empty responses). The function match$(\cdot)$ detects if the feedback contains any error pattern e. Crucially, this design leverages the standardized Input/Output nature of the most prominent LLM agent domains, such as coding, OS operation, and API interaction. Since these environments universally communicate execution failures via standard protocols (e.g., interpreter stderr, HTTP error codes), our validity check is inherently generalizable across these high-value settings without requiring task-specific state engineering.

(2) auxiliary regularization ($\mathcal{R}_{\text{aux}}$): The base signal is then refined with auxiliary regularization, Specifically:

- *entropy regularization:* We model the relationship between consecutive turns to encourage recovery from errors: $h_t^{\text{temporal}} = \beta$ if $v_t > 0$ and $v_{t-1} < 0$. Conversely, it receives a penalty for failing immediately after a success: $h_t^{\text{temporal}} = -\beta$ if $v_t < 0$ and $v_{t-1} > 0$.
- *local consistency:* To prevent reward hacking (e.g., repeatedly using same valid actions) and model stagnation, we apply a penalty for excessive repetition of the same action. This penalty, h_t^{repeat}, is only applied to locally valid actions ($v_t > 0$). Let $N(a_t, \tau)$ be the occurrence count of a specific action a_t among all valid actions within the trajectory τ up to the current turn. The penalty is activated when $N(a_t, \tau)$ exceeds threshold q, as shown in eq. 10

This ensures that once an action is deemed repetitive, each subsequent repetition receives an incrementally larger penalty. Final auxiliary regularization $\mathcal{R}_{\text{aux}} = h_t^{\text{temporal}} + h_t^{\text{repeat}}$.

The magnitudes of these adjustments are controlled by hyperparameters β and α.

A.3 More Ablation Study

Ablation on Repetition Penalty (α) To assess the necessity of explicit constraints against reward hacking, we evaluated the impact of the repetition penalty coefficient α. Table 3 reveals a critical insight: removing the penalty entirely ($\alpha = 0$) leads to a catastrophic collapse in performance (5.1%), as the agent degenerates into repeating a single locally valid action to exploit the reward signal. However, once the penalty is enabled ($\alpha \geq 0.5$), the performance stabilizes at a high level. While larger penalties (e.g., $\alpha = 2.0$) cause a minor drop by discouraging legitimate retries, the method remains effective across a wide range of non-zero values. This confirms that the presence of the penalty mechanism is a fundamental requirement for stability, but the framework is not brittle regarding its specific hyperparameter tuning.

Table 3: Impact of the repetition penalty hyperparameter α on AppWorld success rates.

Value of α	Success Rate (%)
0.0	5.1
0.5	**36.1**
1.0	34.2
2.0	34.5

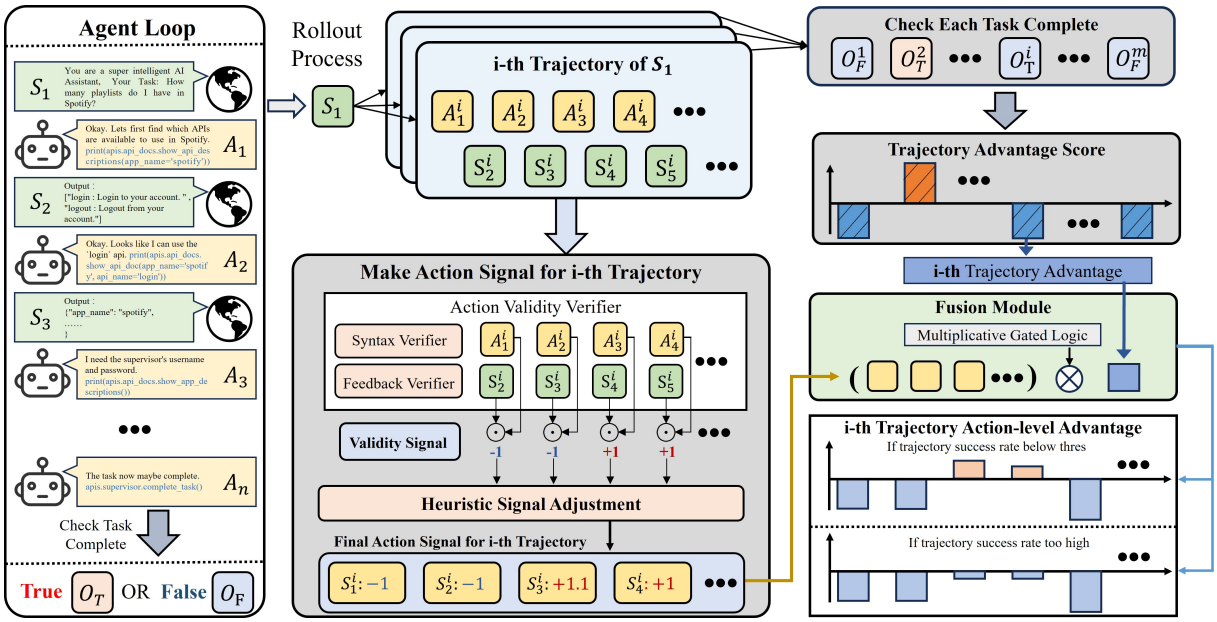

Figure 4: **Example of MADS Calculation in a Failure Trajectory.** (**Top row of final advantages**) In early training, g_t is likely $+1$, preserving a small positive advantage to encourage learning syntax. (**Bottom row**) In late training, g_t is likely -1, assigning a negative advantage to penalize strategically poor actions and enforce learning of the correct task flow.

$$\mathcal{V}(a_t, s_{t+1}) = \begin{cases} -1 & \text{if } a_t \notin \Omega_{\text{syntax}} \vee \exists e \in \mathcal{E}_{\text{runtime}}, \text{match}(s_{t+1}, e) \\ +1 & \text{otherwise} \end{cases} \tag{9}$$

Ablation on Scaling Coefficient (γ) We further investigated the robustness of the scaling coefficient, γ, which is used to dampen the penalty for invalid actions within successful trajectories (Scenario 2). As shown in Table 4, our method exhibits high robustness to variations in this parameter. Varying γ within the range of $[0.8, 1.4]$ results in less than a 1% fluctuation in the final success rate compared to our default setting of $\gamma = 1.0$. This stability indicates that the precise magnitude of the penalty is less critical than the structural logic of the sign alignment. As long as the mechanism correctly identifies and penalizes invalid actions (preserving the negative sign), the agent effectively learns to avoid them without being overly sensitive to the specific scale of the penalty.

Table 4: Sensitivity analysis of the scaling coefficient γ, measured by average success rate on AppWorld.

Value of γ	Success Rate (%)
0.8	35.4
1.0	**36.1**
1.2	35.9
1.4	35.2

Ablation on Temporal Bonus (β) To determine the optimal value for the temporal bonus hyperparameter, β, which is designed to encourage error recovery, we conducted a targeted ablation study on AppWorld. As demonstrated in Table 5, our approach is sensitive to this value. Removing the bonus entirely ($\beta = 0$) results in the lowest performance, confirming that an explicit incentive for self-correction is beneficial. A modest bonus of $\beta = 0.1$ achieves the peak success rate of 36.2%, striking an effective balance. However, we observed that larger bonus values ($\beta \geq 0.3$) lead to a slight performance degradation. This suggests that an overly strong local incentive might create a distracting signal, preventing the agent from focusing on the more crucial global strategic objective. This study empirically validates our choice of $\beta = 0.1$ for all main experiments, as it provides a useful learning signal for local error recovery while preserving the integrity of the global signal.

A.4 More Related Work

A.4.1 Learned Process Reward Models

Another significant line of work attempts to learn a model to predict turn-level rewards. These are of-

$$h_t^{\text{repeat}} = \begin{cases} -\alpha \cdot (N(a_t, \tau) - q) & \text{if } v_t > 0 \text{ and } N(a_t, \tau) > q \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Table 5: Ablation on the temporal bonus hyperparameter β, measured by average success rate on AppWorld.

Value of β	Success Rate (%)
0.0	32.6
0.1	**36.2**
0.3	36.0
0.5	34.6

ten referred to as Process Reward Models (PRMs) (Gao et al., 2025). For instance, SPA-RL (Chen et al., 2025b) trains a "progress estimator" to redistribute the final task reward back to intermediate steps. Similarly, CAPO (Xie et al., 2025) utilizes a powerful teacher LLM as a generative PRM to produce token-level feedback. These methods replace the sparse reward problem with a new, potentially complex estimation problem. The learned reward, which is a proxy for strategic contribution, is then typically added to the environment's reward stream. This still risks rewarding invalid actions if the estimator model incorrectly infers a positive contribution. Furthermore, they introduce significant computational overhead and a dependency on the quality and potential biases of the reward model itself.

Reinforcement Learning for LLM agents is fundamentally a credit assignment problem. Prevailing methods, which we classify under an *additive paradigm*, often conflate an action's intrinsic validity with its long-term strategic contribution. This leads to credit misassignment, where invalid actions in successful trajectories can erroneously receive positive rewards. To address this, we propose a novel *multiplicative framework* centered on the principle of **signal decoupling**. Instead of learning a single, entangled turn-level advantage, we decompose the learning signal into two orthogonal components: a turn-level sign representing factual validity and a trajectory-level magnitude representing strategic quality.

A.5 Adaptive Signal Denoising Implementation Details

The probability of retaining a positive sign, p_{retain}, is governed by the schedule function $f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}})$ shown in (11). **The hyperparameters within are set to achieve this functional goal; any function that successfully balances the sample distribution in later training stages would be suitable.**

The schedule is formalized as:

A.6 Baselines

We select the following baselines:

- **ReAct**: A zero-shot prompting baseline using the popular Reason-Act framework (Yao et al., 2022), demonstrating the base model's intrinsic agent capabilities without any parameter updates.
- **SFT (Supervised Fine-Tuning)**: Standard behavioral cloning where the model is finetuned on expert trajectories.
- **GRPO (Shao et al., 2024)**: The standard Group-Relative Policy Optimization algorithm, which applies a uniform positive or negative reward to all actions within a trajectory. This serves as our primary *trajectory-level* RL baseline.
- **GiGPO (Feng et al., 2025)**: A SOTA RL method that improves upon GRPO by incorporating an *additive turn-level* advantage estimation. This is our primary competitor representing the prevailing paradigm of credit assignment.
- **Loop (Chen et al., 2025a)**: A specialized agent method reported for the AppWorld benchmark, included as a strong contemporary baseline for that environment.

A.7 Implementation Details

Environment setting We utilize the ALFWorld benchmark (Shridhar et al., 2020), a text-based environment for long-horizon, embodied tasks. It is based on the ALFRED dataset and tests an agent's ability to reason, plan, and ground language in a sequence of actions to achieve high-level goals (e.g., "put a warm apple in the microwave"). We report performance on three difficulty splits—L0, L1, and L2:

$$p_{\text{retain}} = f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}}) = \begin{cases} 1 & \text{if } V_{\text{batch}} < \theta_V \text{ or } C_{\text{batch}} < \theta_{C1} \\ 1 - \delta \cdot C_{\text{batch}} & \text{if } \theta_{C1} \leq C_{\text{batch}} < \theta_{C2} \\ p_{\min} & \text{otherwise} \end{cases} \tag{11}$$

- **L0 :** Pick and Look.
- **L1 :** Clean and Heat.
- **L2 :** Cool and Pick2.

For ALFworld, we follow the setting of GiGPO (Feng et al., 2025), we use a group size of 8 and sample 16 different groups per rollout, resulting in a total of $16 \times 8 = 128$ environments. The rollout temperature is set to 1.0, the maximum response length is 512 tokens. Each episode allows up to 50 environment steps.

For AppWorld, we follow the setting of Loop (Chen et al., 2025a). We train on a subset of the AppWorld train set, excluding difficulty 3 tasks. This subset consists of 24 scenarios, with 3 minor variations (tasks) per scenario, which contain 72 tasks. Each iteration starts with the generation of K = 6 rollouts with temperature 1.0 for 40 randomly sampled tasks, for a total of 240 rollouts. We use only difficulty-1 and difficulty-2 tasks for training. The maximum response length is 2048 tokens. Each episode allows up to 35 environment steps. The task IDs we use are as follows:

07b42fd	229360a	27e1026
287e338	692c77d	82e2fac
aa8502b	b7a9ee9	c901732
ccb4494	ce359b5	e7a10f8
e85d92a	e3d6c94	d0b1f43
2a163ab	60d0b5b	6ea6792
29caf6f	cf6abd2	771d8fc
7d7fbf6	76f2c72	302c169

Action Validity Signal (R_{local}) Determination The core of our Multiplicative Advantage via Decoupled Signals (MADS) framework is the ability to decouple local action validity from global trajectory success. This is primarily achieved through the action-level signal, R_{local}, which begins with a **Fundamental Validity Signal** ($v_t \in \{-1, +1\}$). This signal provides an immediate, fact-based judgment on the correctness of each action a_t. An action is considered invalid (v_t is set to -1) if it fails any of the checks detailed below for each environment. Otherwise, it is considered valid (v_t is set to +1).

AppWorld In the AppWorld benchmark, agent actions consist of Python code designed to interact with a suite of simulated applications. An action is determined to be **invalid** if it meets one or more of the following criteria, categorized into environment feedback analysis and action format checks.

1. **Environment Feedback Analysis:** A negative validity signal ($v_t = -1$) is assigned if the environment's execution feedback contains any of the following error-indicating substrings. These keywords typically signify crashes, syntax errors, or execution timeouts:
   - `"Execution failed"`
   - `"Traceback:"`
   - `"SyntaxError"`
   - `"Exception"`
   - `"Error:"`
   - `"Maximum number of executions"`
   - `"timed out after"`
   - `"No code available to execute"`
2. **Action Format and Syntax Analysis:** Additionally, the raw action generated by the agent is preemptively checked for structural integrity. The action is considered invalid ($v_t = -1$) if it fails any of these checks:
   - **Invalid Formatting:** The generated code is not correctly encapsulated within a ```python ... ``` markdown block, making it unexecutable by the environment parser.
   - **Absence of API Call:** The executable code, once extracted, does not contain the `.api` substring. This rule is crucial to prevent the agent from receiving signals for syntactically correct but strategically useless code (e.g., `print("thinking")`) that does not contribute to task progression.

- **Excessive Reasoning:** The content within the <think> but without </think> tags is determined to be excessively long. This helps penalize instances where the model may be "stuck" or hallucinating, rather than producing a concise, relevant reasoning step.

ALFWorld In the text-based ALFWorld environment, agent actions are natural language commands. The validity of these commands is assessed based on the environment's textual response and the agent's output format.

1. **Environment Feedback Analysis:** An action is deemed invalid ($v_t = -1$) if the textual feedback from the ALFWorld environment matches any of the following regular expression patterns. These patterns capture a wide range of common failure scenarios, such as interacting with non-existent objects or attempting illogical actions:
   - `r"^nothing happens\.?$"` (The most common feedback for an ineffectual action)
   - `r"you don't see that"` (Object is not present)
   - `r"you can't see that"` (Object is out of sight)
   - `r"that command is not understood"` (Command is not recognized by the game parser)
   - `r"you haven't got"` (Agent is not holding the required item)
   - `r"you are not"` (Agent is in the wrong location)
   - `r"you need to"` (A prerequisite condition is not met)
   - `r"you must"` / `r"you have to"` (A required condition is not fulfilled)
   - `r"that's not"` / `r"not a valid"` / `r"not valid"` (Command is syntactically or logically invalid)
   - `r"you cannot"` / `r"you can not"` (The attempted action is impossible)
   - `r"not available"` (The target object or feature cannot be used)
2. **Action Format Analysis:** A strict format is enforced on the agent's output to ensure clear separation between reasoning and action. The generated output is marked as invalid ($v_t = -1$) if it does not precisely follow the required structure, which mandates that the reasoning and the action are respectively and completely enclosed within <think>...</think> and <action>...</action> tags.

Implicit Curriculum Hyperparameters For the schedule function $f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}})$ shown in Equation 8, the hyperparameters are set to $\theta_V = 0.4$, $\theta_{C1} = 0.1$, $\theta_{C2} = 0.6$, the decay factor $\delta = 1.5$, and the minimum retention probability $p_{\min} = 0.1$. This specific configuration is designed to counteract the inherent data imbalance in agentic learning, guiding the training process towards a healthier, near 1:1 ratio of positive to negative signals for strategically ambiguous actions. We also set the adjustment: $\beta = 0.1$ for temporal dynamics, $\alpha = 0.5$ with a threshold $q = 2$ for the repetition penalty.

A.8 Algorithm Pseudocode

To formalize the training procedure, we present the MADS algorithm in pseudocode (Algorithm 1). This algorithm details the end-to-end process, from trajectory collection to the final policy update, emphasizing the crucial steps of our framework.

The key aspects highlighted in the pseudocode are:

- **Trajectory Segmentation (Lines 10–12):** Before advantage calculation, each trajectory τ_i is segmented into a set of prompt-action pairs (p'_t, a_t), where the prompt p'_t comprises the entire history up to that point. This step correctly formats the data for training an autoregressive language model policy.
- **Decoupled advantage Calculation (Lines 14–15 & 19–22):** The algorithm computes advantages in a decoupled manner. The global, trajectory-level magnitude A_{global} is computed first (Line 15). Then, for each segmented sample, the local, action-level sign $R_{\text{local},t}$ is derived from the environmental feedback contained in the subsequent state s_{t+1} (Line 20).
- **Conditional Multiplicative Fusion (Lines 24–29):** The core multiplicative gating logic is shown. The final signal $A_{\text{action},t}$ is determined by the alignment of the local and global signals, with the stochastic gate g_t being the key component for the implicit curriculum.

- **Dynamic Gating (Lines 17–18 & 28):** The probability p_{retain} for the stochastic gate is dynamically calculated based on the agent's batch-level performance $(C_{\text{batch}}, V_{\text{batch}})$, ensuring the reward strategy adapts as the agent improves.

Algorithm 1 Multiplicative Advantage via Decoupled Signals (MADS) Training Loop

Require: Initial policy π_θ, Environment Env
Require: Hyperparameters: batch size m, learning rate η, scaling coeff. γ
Require: MADS Hyperparameters: β, α, q for heuristics; $\theta_v, \theta_{C1}, \theta_{C2}, \alpha_{\text{decay}}, p_{\text{min}}$ for curriculum

Initialize policy parameters θ
for each training iteration **do**
  Initialize a raw trajectory buffer $\mathcal{B}_{\text{raw}} \leftarrow \emptyset$
  # — Phase 1: Trajectory Collection —
  for $i = 1$ to m **do**
    Collect trajectory $\tau_i = (s_1, a_1, s_2, \ldots, s_{T_i}, a_{T_i})$ by executing π_θ in Env
    Store (τ_i, R_{τ_i}) in $\mathcal{B}_{\text{raw}}$, where R_{τ_i} is the task success signal

  # — Phase 2: Segmentation and advantage Calculation —
  Initialize a training batch buffer $\mathcal{D}_{\text{batch}} \leftarrow \emptyset$
  # Step 2a: Segment trajectories into (prompt, action) samples
  Compute C_{batch} (task completion rate) and V_{batch} (action validity rate) from $\mathcal{B}_{\text{raw}}$
  Compute positive sign retention probability $p_{\text{retain}} \leftarrow f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}})$ ▷ Equation 8
  for each trajectory $\tau_i \in \mathcal{B}_{\text{raw}}$ **do**
    Compute trajectory-level advantage $A_{\text{global}}(\tau_i) \leftarrow \frac{m}{m-1}\left(R_{\tau_i} - \frac{1}{m}\sum_{j=1}^{m} R_{\tau_j}\right)$ ▷ Equation 7
    for $t = 1$ to T_i **do**
      Define prompt $p'_t \leftarrow (s_1, a_1, \ldots, s_t)$ and action a_t from τ_i
      Get next state s_{t+1} from τ_i (or final state if $t = T_i$)
      # Step 2b: Compute action-level signals and fuse for final advantage
      $v_t \leftarrow \text{ComputeFundamentalValidity}(a_t, s_{t+1})$ ▷ Feedback is in s_{t+1}
      $h_t \leftarrow \text{ComputeHeuristicAdjustments}(a_{1..t}, v_{1..t}, \alpha, \beta, q)$ ▷ Temporal & Repetition
      $S_{\text{local},t} \leftarrow v_t + h_t$ ▷ Equation 5

      # Multiplicative Gating Fusion Logic
      if $\text{sgn}(S_{\text{local},t}) = \text{sgn}(A_{\text{global}}(\tau_i))$ **then**
        $A_{\text{action},t} \leftarrow S_{\text{local},t} \cdot |A_{\text{global}}(\tau_i)|$
      else if $A_{\text{global}}(\tau_i) > 0$ and $S_{\text{local},t} < 0$ **then** ▷ Successful traj, invalid action
        $A_{\text{action},t} \leftarrow \gamma \cdot S_{\text{local},t} \cdot |A_{\text{global}}(\tau_i)|$ ▷ Dampened penalty
      else ▷ Failed traj, valid action
        $g_t \sim \text{GatedSign}(p_{\text{retain}})$ ▷ Sample gate: $g_t \in \{+1, -1\}$
        $A_{\text{action},t} \leftarrow g_t \cdot \gamma \cdot R_{\text{local},t} \cdot |A_{\text{global}}(\tau_i)|$
      Add $(p'_t, a_t, A_{\text{action},t})$ to training batch $\mathcal{D}_{\text{batch}}$

  # — Phase 3: Policy Update —
  Compute advantages $\hat{A}_t$ for all samples in $\mathcal{D}_{\text{batch}}$ based on $A_{\text{action},t}$
  Update policy parameters $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}^{\text{PPO}}(\theta)$ using samples from $\mathcal{D}_{\text{batch}}$ ▷ Equation 4