

# 000 001 002 003 004 005 006 007 BOTH LOCAL VALIDITY AND GLOBAL EFFECTIVENESS MATTER: DECOUPLED CREDIT ASSIGNMENT FOR LONG-HORIZON AGENTIC LEARNING 008 009 010 011 012

006 **Anonymous authors**  
007 Paper under double-blind review  
008  
009  
010  
011  
012

## ABSTRACT

013 The natural-language action space of Large Language Model (LLM) agents creates  
014 a real risk of invalid outputs (e.g., API rejections, parsing errors). Consequently,  
015 in Reinforcement Learning (RL) for long-horizon LLM agents, learning to  
016 generate a locally valid action in each turn is as crucial as selecting a globally  
017 effective one. However, this requirement was overlooked by the prevailing additive  
018 paradigm for credit assignment in agentic RL. Specifically, it computes an  
019 action’s credit by summing an estimated local score and the trajectory-level score.  
020 This paradigm assigns a “contribution” score to all actions regardless of their val-  
021 idity, allowing invalid actions to be assigned positive credit, especially in positive  
022 trajectories. To address this, we propose Multiplicative Gated Rewards (MGR),  
023 which decouples local action-level validity from global effectiveness. MGR uses  
024 a fact-based validity signal, derived from direct environment feedback and syn-  
025 tactic validity, to determine the action-level score (e.g.,  $\pm 1$ ). This score is then  
026 multiplied by the magnitude of the trajectory-level score. This ensures the ac-  
027 tion’s validity strictly governs the reward’s polarity, preventing credit misassign-  
028 ment. Experiments demonstrate that our method improves training stability and  
029 achieves SOTA performance on long-horizon LLM agent benchmarks. Code of  
030 MGR has been uploaded in the Supplementary Material.  
031  
032

## 1 INTRODUCTION

033 A key frontier for Large Language Model (LLM) agents lies in their ability to solve long-  
034 horizon tasks (Yao et al., 2022), often requiring  
035 a long chain of interactions to navigate com-  
036 plex environments (e.g., over 10 interactions).  
037 Reinforcement Learning (RL) has emerged as  
038 a standard for training these agents, enabling  
039 them to learn from environmental feedback.  
040 Unlike traditional RL agents with discrete ac-  
041 tion sets, language-based agents face a unique  
042 challenge in their vast, language-based action  
043 space. They generate actions as text sequences  
044 (e.g., code, API calls), creating a fundamental  
045 dual requirement for every action: it must be  
046 both **locally valid** (e.g., syntactically correct)  
047 to be executable, and **globally effective** to con-  
048 tribute to the final task goal. As emphasized by  
049 (Zhao et al., 2025), the validity of each turn’s  
050 action is crucial for long-horizon agents.

051 However, we observe that this essential require-  
052 ment for local action validity has been largely overlooked by the prevailing credit assignment meth-  
053 ods in agentic RL, which we term the *additive paradigm* (as illustrated in Fig. 1). Whether through  
learned progress estimators (Wu et al., 2025), reward decomposition (Zheng et al., 2025), or sub-

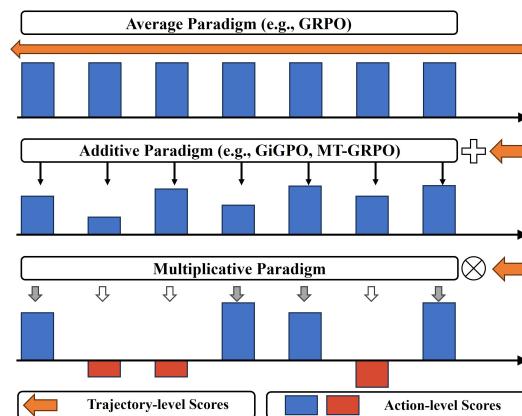


Figure 1: Illustration of Different Credit Assignment Paradigms in a Successful Trajectory.

054 trajectory (Feng et al., 2025) comparisons, this paradigm computes an action’s credit by summing  
 055 an estimated local score with the global score of its parent trajectory. This design is blind to intrinsic  
 056 validity, and its core flaw is assigning a “contribution” score to all actions regardless of whether they  
 057 are executable or not. This leads to severe credit misassignment: an unequivocally invalid action  
 058 may receive a positive reward simply by appearing in a successful trajectory (shown in the second  
 059 row of Figure 1). Such flawed signals fundamentally destabilize the learning process, forcing the  
 060 agent to expend valuable samples learning basic syntax from ambiguous feedback, a problem that  
 061 contributes to training inefficiencies.

062 To address this, we propose **Multiplicative Gated Rewards (MGR)**, a novel credit assignment  
 063 framework that decouples the reward signal into two orthogonal components. The first is an Action-  
 064 Level Sign Provider, which determines the atomic, factual correctness (a positive/negative sign)  
 065 of an action. This sign is derived from direct environmental feedback and refined with heuristics  
 066 that model temporal dynamics (e.g., error correction) and penalize unproductive behaviors (e.g.,  
 067 repetition). The second is a Trajectory-Level Magnitude Provider, which employs group-relative  
 068 methods to evaluate the quality of an entire trajectory, providing the trajectory-level scores.

069 These two components are fused via a multiplicative gating mechanism, where the final reward is  
 070 the product of the local signal and the global magnitude. This design ensures that an action’s fac-  
 071 tual validity strictly governs the reward’s polarity. It guarantees that invalid actions receive negative  
 072 rewards even in highly successful trajectories, while valid actions within failed trajectories are se-  
 073 lectively preserved as positive learning samples via a dynamic gate. Crucially, this mechanism also  
 074 gives rise to an implicit curriculum: early in training, the agent is primarily rewarded for mastering  
 075 foundational skills (i.e., producing valid actions); as it matures, the learning objective implicitly  
 076 transitions to demanding the strategic composition of these actions to achieve high-quality trajec-  
 077 tories. This multiplicative gating mechanism corrects the core flaw of the additive paradigm: an  
 078 action’s factual invalidity can no longer be masked by overall trajectory success. This eliminates a  
 079 critical vector of credit misassignment by design.

080 Our contributions are threefold:

- 081 • We identify the dual challenge in agentic learning: achieving both local action validity and  
 082 global strategic effectiveness.
- 084 • We propose Multiplicative Gated Rewards (MGR), a new framework that uses a fact-based,  
 085 action-level signal to gate the credit flowing from a trajectory’s global outcome.
- 087 • We demonstrate empirically that MGR leads to significant gains in training stability and  
 088 sample efficiency, achieving SOTA performance on standard agentic benchmarks.

## 090 2 RELATED WORK

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

### 097 2.1 TRAJECTORY-LEVEL REINFORCEMENT LEARNING

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

108 2.2 LEARNED PROCESS REWARD MODELS  
109

110 Another significant line of work attempts to learn a model to predict turn-level rewards. These are  
111 often referred to as Process Reward Models (PRMs) (Gao et al., 2025). For instance, SPA-RL (Chen  
112 et al., 2025b) trains a “progress estimator” to redistribute the final task reward back to intermediate  
113 steps. Similarly, CAPO (Xie et al., 2025) utilizes a powerful teacher LLM as a generative PRM to  
114 produce token-level feedback. These methods replace the sparse reward problem with a new, poten-  
115 tially complex estimation problem. The learned reward, which is a proxy for strategic contribution,  
116 is then typically added to the environment’s reward stream. This still risks rewarding invalid actions  
117 if the estimator model incorrectly infers a positive contribution. Furthermore, they introduce sig-  
118 nificant computational overhead and a dependency on the quality and potential biases of the reward  
119 model itself.

120  
121 2.3 FINE-GRAINED ADVANTAGE ESTIMATION  
122

123 A third category of methods refines the advantage calculation at a sub-trajectory level, but remains  
124 within the foundational additive framework of the Bellman equation. GiGPO (Feng et al., 2025)  
125 extends the group-relative concept to the step level by comparing actions from different trajectories  
126 at aligned “anchor states.” ARPO (Dong et al., 2025) uses model uncertainty (entropy) as a heuris-  
127 tic to identify critical steps and attribute credit accordingly. SPO (Huang et al., 2025) proposes a  
128 segment-level advantage estimation as a compromise between turn-level and trajectory-level granu-  
129 larity. While these methods offer more sophisticated estimation techniques, they are still fundamen-  
130 tally estimating a single, conflated score for each step that is implicitly added to form the trajectory’s  
131 total value. They do not possess a mechanism to strictly enforce the binary validity of an action.

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

140  
141 3 METHOD  
142

143 To decouple reward signals into local and global components, we introduce **MGR (Multiplicative**  
144 **Gated Rewards)**, a multiplicative framework that separates the reward into an action-level signal  
145 judging local validity and a trajectory-level magnitude evaluating global strategic contribution. The  
146 final reward,  $R_{\text{action}}$ , is conceptualized as:

$$R_{\text{action}} = \underbrace{\mathcal{G}(R_{\text{local}})}_{\text{Validity Signal}} \times \underbrace{R_{\text{global}}}_{\text{Effectiveness Magnitude}} \quad (1)$$

147 where  $R_{\text{local}}$  is the action-level signal,  $R_{\text{global}}$  is the trajectory-level score. And  $\mathcal{G}$  is a multiplica-  
148 tive gated module, which fuses the two signals before. This design ensures that credit is assigned  
149 correctly: all invalid actions are consistently penalized, while valid actions within failed trajectories  
150 can be selectively rewarded to facilitate learning. The algorithm is detailed in Alg. 1.

155  
156 3.1 TRAJECTORY SEGMENTATION AND REINFORCEMENT LEARNING  
157

158 Our training process begins with collecting a set of  $m$  trajectories,  $\{\tau_1, \tau_2, \dots, \tau_m\}$ , for each  
159 given task. A single trajectory  $\tau_i$  of length  $T$  is a sequence of states and actions:  $\tau_i =$   
160  $(s_1, a_1, s_2, a_2, \dots, s_T, a_T)$ . To form a training batch for our language model-based policy  $\pi_\theta$ ,  
161 we segment each trajectory into action-level samples. Each sample  $(p'_t, a_t)$  consists of the ac-  
162 tion  $a_t$  and its corresponding prompt  $p'_t$ , which comprises the entire history up to that point, i.e.,

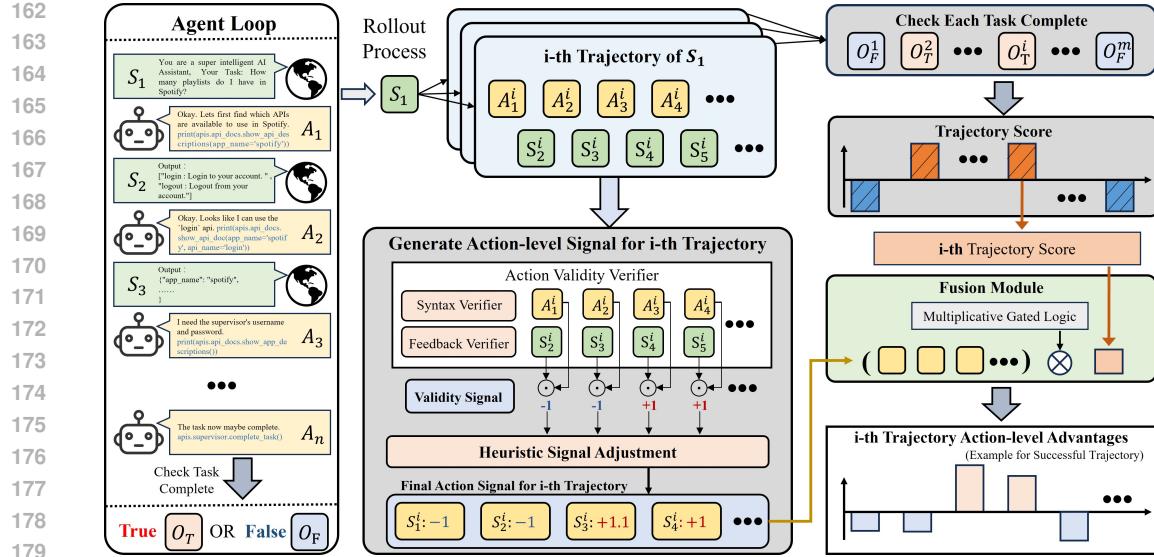


Figure 2: Example of **MGR (Multiplicative Gated Rewards)** calculation in a successful Trajectory. The example in the failure trajectory can be seen in App. A.2.

$p'_t = (s_1, a_1, \dots, s_t)$ . This segmentation process can be visualized as follows:

$$\tau = [s_1, a_1, \dots, s_T, a_T] \longrightarrow \left\{ \begin{array}{ll} (\text{prompt: } s_1, & \text{response: } a_1) \rightarrow r_1 \\ (\text{prompt: } [s_1, a_1, s_2], & \text{response: } a_2) \rightarrow r_2 \\ \vdots & \vdots \\ (\text{prompt: } [s_1, a_1, \dots, s_T], & \text{response: } a_T) \rightarrow r_T \end{array} \right\}$$

where each action-level reward  $r_t$  is computed by our MGR framework.

Based on MGR's rewards, we then update the policy parameters  $\theta$  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:

$$\mathcal{L}^{\text{PPO}}(\theta) = \mathbb{E}_t \left[ \min \left( \rho_t(\theta) \hat{A}_t, \text{clip}(\rho_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right] \quad (2)$$

where  $\rho_t(\theta) = \frac{\pi_\theta(a_t | s'_t)}{\pi_{\theta_{\text{old}}}(a_t | s'_t)}$  is the probability ratio, and  $\hat{A}_t$  denotes the token-level advantage in PPO, which is estimated from the MGR rewards  $r_t$  via GAE and PPO's critic model. The subsequent sections will provide an exposition of the MGR.

### 3.2 REWARD SIGNAL DECOUPLING

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

#### 3.2.1 ACTION-LEVEL SIGNAL PROVIDER ( $R_{\text{LOCAL}}$ )

This component is responsible for determining the local, factual validity of an action  $a_t$ . It generates a signal  $R_{\text{local},t}$ , that evaluates the atomic correctness of an agent's action at each turn. The process involves two stages:

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

216  
217  
218  
219

$$v_t = \begin{cases} -1 & \text{if } a_t \notin \Omega_{\text{syntax}} \vee \exists e \in \mathcal{E}_{\text{runtime}}, \text{match}(o_{t+1}, e) \\ +1 & \text{otherwise} \end{cases} \quad (3)$$

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

229 **(2) Heuristic Adjustments ( $h_t$ ):** The base signal is then refined with heuristic bonuses and pen-  
 230 alties,  $h_t$ , to capture more nuanced behaviors. Specifically:

- 232 • *Heuristic Bonus:* We model the relationship between consecutive turns to encourage re-  
 233 recovery from errors:  $h_t^{\text{temporal}} = \beta$  if  $v_t > 0$  and  $v_{t-1} < 0$ . Conversely, it receives a penalty  
 234 for failing immediately after a success:  $h_t^{\text{temporal}} = -\beta$  if  $v_t < 0$  and  $v_{t-1} > 0$ .
- 235 • *Repetition Penalty:* To prevent reward hacking (e.g., repeatedly using same valid actions)  
 236 and model stagnation, we apply a penalty for excessive repetition of the same action. This  
 237 penalty,  $h_t^{\text{repeat}}$ , is only applied to locally valid actions ( $v_t > 0$ ). Let  $N(a_t, \tau)$  be the  
 238 occurrence count of a specific action  $a_t$  among all valid actions within the trajectory  $\tau$  up  
 239 to the current turn. The penalty is activated when  $N(a_t, \tau)$  exceeds threshold  $q$ :

$$h_t^{\text{repeat}} = \begin{cases} -\alpha \cdot (N(a_t, \tau) - q) & \text{if } v_t > 0 \text{ and } N(a_t, \tau) > q \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

240 This ensures that once an action is deemed repetitive, each subsequent repetition receives  
 241 an incrementally larger penalty.

242 The magnitudes of these heuristic adjustments are controlled by hyperparameters  $\beta$  and  $\alpha$ . Final  
 243 heuristic adjustments  $h_t = h_t^{\text{temporal}} + h_t^{\text{repeat}}$ . The final action-level signal of  $a_t$  is given by:

$$R_{\text{local},t} = v_t + h_t \quad (5)$$

### 249 3.2.2 TRAJECTORY-LEVEL MAGNITUDE PROVIDER ( $R_{\text{GLOBAL}}$ )

250 This component evaluates the global, strategic quality of an entire trajectory  $\tau_i$ . For a given task  
 251  $\tau$ , we compare all  $m$  collected trajectories. First, each trajectory is assigned a binary score  $S(\tau_i)$   
 252 based on task completion signal. Following recent advancements in training agents for long-horizon  
 253 tasks (Chen et al., 2025a), we utilize the Eqn. (6) to transform these absolute scores into a normal-  
 254 ized, zero-sum advantage scores. This provides a more stable learning signal by focusing on relative  
 255 performance. The global magnitude for trajectory  $\tau_i$  is calculated as:

$$R_{\text{global}}(\tau_i) = \frac{m}{m-1} \left( S(\tau_i) - \frac{1}{m} \sum_{j=1}^m S(\tau_j) \right) \quad (6)$$

### 260 3.3 REWARD FUSION

261 The fusion of the local signal  $R_{\text{local},t}$  and the global magnitude  $R_{\text{global}}$  is performed via a conditional  
 262 multiplicative process. The core idea is that the final reward's sign should primarily be dictated by  
 263 the action's local validity ( $R_{\text{local}}$ ), while its magnitude should be scaled by the trajectory's overall  
 264 effectiveness ( $R_{\text{global}}$ ). The logic is partitioned into three distinct scenarios based on the alignment  
 265 of the signals' signs:

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

270  
 271    2. **Successful trajectory, invalid action** ( $R_{\text{global}} > 0, R_{\text{local}} < 0$ ): The action must be pe-  
 272    nalyzed to discourage it, but the penalty is damped. This is achieved by multiplying the  
 273    signals and applying a scaling coefficient.  
 274    3. **Failed trajectory, valid action** ( $R_{\text{global}} < 0, R_{\text{local}} > 0$ ): This is the most crucial and  
 275    ambiguous case. We introduce a stochastic gating operator,  $g_t$ , which determines the sign  
 276    of the final reward.

277    This fusion logic is formally captured in the following Equation. Let  $\tau(t)$  denote the trajectory  
 278    containing the action at turn  $t$ . The final reward  $R_{\text{final},t}$  is defined as (more detail in Alg. 1):  
 279

280  
 281    
$$R_{\text{final},t} = \begin{cases} R_{\text{local},t} \cdot |R_{\text{global}}(\tau(t))| & \text{if } \text{sign}(R_{\text{local},t}) = \text{sign}(R_{\text{global}}(\tau(t))) \\ \gamma \cdot R_{\text{local},t} \cdot R_{\text{global}}(\tau(t)) & \text{if } R_{\text{global}}(\tau(t)) > 0 \text{ and } R_{\text{local},t} < 0 \\ \gamma \cdot g_t \cdot R_{\text{local},t} \cdot |R_{\text{global}}(\tau(t))| & \text{if } R_{\text{global}}(\tau(t)) < 0 \text{ and } R_{\text{local},t} > 0 \end{cases} \quad (7)$$
  
 282  
 283

284    where  $\gamma \in (0, 1]$  is scaling coefficients that reduce the reward magnitude in cases of sign conflict.  
 285    The key component is the gating operator  $g_t \in \{-1, 1\}$ , which is sampled based on the agent’s  
 286    overall performance (i.e., task success rate and action accuracy rate). The function of this gate is to  
 287    dynamically control the balance of positive and negative training samples for locally correct actions,  
 288    preventing the model from being rewarded for valid but strategically poor choices, especially in later  
 289    stages of training. The mechanism of  $g_t$  is detailed in the next section.  
 290

### 291    3.4 DYNAMIC SAMPLE BALANCING VIA GATED SIGN FLIPPING

293    As the training progresses, the model rapidly masters the local validity of actions. We observe  
 294    a shift in the distribution of action-level samples: the ratio of valid to invalid actions transitions  
 295    from a balanced state (e.g., 40% vs 60%) to a severe imbalance (e.g., 90% vs 10%), even within  
 296    unsuccessful trajectories.

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

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

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

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

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

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

317  
 318    
$$p_{\text{retain}} = f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}}) = \begin{cases} 1 & \text{if } V_{\text{batch}} < \theta_V \text{ or } C_{\text{batch}} < \theta_{C1} \\ 1 - \delta \cdot C_{\text{batch}} & \text{if } \theta_{C1} \leq C_{\text{batch}} < \theta_{C2} \\ p_{\text{min}} & \text{otherwise} \end{cases} \quad (8)$$
  
 319  
 320

322    where  $C_{\text{batch}}$  is the mean task completion rate,  $V_{\text{batch}}$  is the mean action validity rate for the current  
 323    batch, and  $\theta$  means the transformation threshold of  $p_{\text{retain}}$ . The probability of retaining a positive  
 324    sign,  $p_{\text{retain}}$ , is governed by the schedule function  $f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}})$  shown in Eqn. (8). The

324 **primary objective** of  $f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}})$  is to guide the sample ratio towards a healthier state as  
 325 training progresses. *The hyperparameters within are set to achieve this functional goal; any function*  
 326 *that successfully balances the sample distribution in later training stages would be suitable.* This  
 327 dynamic sample balancing naturally gives rise to an **implicit two-stage curriculum**:

328

- 329 • **Stage 1: Learning Action Validity.** In early training, when agent performance is low,  
 330  $p_{\text{retain}}$  is high. This allows the model to learn fundamental, locally valid actions from a  
 331 sufficient pool of positive examples.
- 332 • **Stage 2: Learning Strategic Composition.** As the agent’s proficiency increases,  $p_{\text{retain}}$   
 333 decreases. This forces the agent to focus on learning the correct sequencing and strategic  
 334 use of those valid actions to achieve global success.

335

## 336 4 EXPERIMENT

337

### 338 4.1 ENVIRONMENTS

339

340 To comprehensively evaluate the effectiveness of our MGR, we conduct experiments on two chal-  
 341 lenging benchmarks designed for multi-turn LLM agents: ALFWorld and AppWorld.

342 **ALFWorld** (Shridhar et al., 2020) is a benchmark that requires agents to perform complex, long-  
 343 horizon tasks in a simulated text-based household environment. We report performance on three  
 344 difficulty splits: L0, L1, and L2. Other settings are detailed in App. A.5.

345 **AppWorld** (Trivedi et al., 2024) is a recently proposed benchmark that evaluates agents on their  
 346 ability to perform realistic, day-to-day digital tasks by interacting with a suite of simulated appli-  
 347 cations. We evaluate on its three official difficulty settings: Easy, Medium, and Difficult. Other  
 348 settings are detailed in App. A.5.

349

350

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

353 <b>Model</b>	354 <b>Method</b>	355 <b>Reward</b> 356 <b>Level</b>	357 <b>Reward</b> 358 <b>Signal</b>	359 <b>ALFWorld</b>			360 <b>AppWorld</b>		
				361 L0	362 L1	363 L2	364 Easy	365 Medi	366 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	<b>90.2</b>	67.2	-	-	-
	Loop	token	Task	-	-	-	59.6	14.3	6.1
	MGR	action	T/A	<b>90.4</b>	89.7	<b>85.6</b>	<b>76.1</b>	<b>19.6</b>	<b>9.4</b>
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	<b>91.2</b>	68.7	-	-	-
	Loop	token	Task	-	-	-	63.2	18.3	8.7
	MGR	action	T/A	<b>91.6</b>	90.8	<b>82.6</b>	<b>75.4</b>	<b>22.8</b>	<b>10.2</b>

### 368 4.2 MODELS AND BASELINES

369

370 All finetuning experiments are conducted on two powerful open-source models, **Qwen2.5-7B** (Qwen  
 371 et al., 2025) and **Qwen3-8B** (Yang et al., 2025), to serve as the agent’s backbone. To situate our  
 372 results within the broader landscape, we also include performance metrics from the proprietary **GPT-**  
 373 **4o** model and a previously reported strong system, **DS-R1** (DeepSeek-AI et al., 2025), as reference  
 374 points for the SOTA. Our core comparisons are against a suite of finetuning baselines applied to the  
 375 same models. We compare against several methods applied to the same base models for a fair and  
 376 direct comparison: ReAct(Yao et al., 2022), SFT, GRPO (Shao et al., 2024), GiGPO (Feng et al.,  
 377 2025), Loop (Chen et al., 2025a), detailed in App. A.3.

378  
379

## 4.3 IMPLEMENTATION DETAILS

380

381 We employ the AdamW optimizer with a constant learning rate of  $5e-7$ . The specific hyperparam-  
 382 eters for our Multiplicative Gated Rewards (MGR) framework are set as described in the Method  
 383 section. We set the heuristic adjustment:  $\beta = 0.1$  for temporal dynamics,  $\alpha = 0.5$  with a thresh-  
 384 old  $q = 2$  for the repetition penalty. For the schedule function  $f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}})$  shown in  
 385 Eqn. (8), we set the hyperparameters to maintain a 1:1 ratio of positive to negative samples, detailed  
 386 in App. A.5. Other settings are also provided in App. A.5.

387

388

## 389 4.4 RESULTS AND ANALYSIS

390

391 As presented in Table 1, our proposed method consistently achieves SOTA performance across all  
 392 evaluated settings on both the ALFWORLD and AppWorld benchmarks. This demonstrates the effec-  
 393 tiveness and robustness of our MGR reward framework in training capable LLM agents for complex,  
 394 multi-step interaction tasks.

395

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

402

403 These advantages are decisive on AppWorld. Fine-grained credit-assignment methods such as  
 404 GiGPO depend on structured per-turn state descriptions to compute granular advantages. App-  
 405 World’s complex, interactive interface does not expose such a well-parsed state, making these meth-  
 406 ods **directly inapplicable** and highlighting a broader challenge in agentic RL: reward designs fre-  
 407 quently lack generality across realistic environments. MGR, by contrast, is broadly applicable.  
 408 Its action-level validity signal ( $R_{\text{local}}$ ) is derived from generic, universally available feedback (e.g.,  
 409 execution errors or None responses) and lightweight syntactic checks, requiring no environment-  
 410 specific state parsing. As a result, MGR can be applied seamlessly in unstructured environments  
 411 like AppWorld, where it sets a new performance bar and significantly outperforms all baselines.

412

413 In summary, our empirical results strongly validate our central hypothesis: by strategically decou-  
 414 pling the reward signal into local validity and global effectiveness, MGR not only achieves superior  
 415 and more generalizable performance but also offers a robust and universally applicable framework.

416

## 417 4.5 ANALYSIS OF TRAINING DYNAMICS

418

419 To understand MGR’s effectiveness, we analyze training dynamics by comparing MGR, GRPO, and  
 420 Loop baselines on the AppWorld benchmark, tracking both action success rate and task success rate.  
 421 Figure 3 illustrates these learning curves, highlighting MGR’s superior efficiency and stability.

422

423 The top plot of Figure 3 depicts the task success rate. MGR (red curve) exhibits a stable and consis-  
 424 tent upward trend, achieving a much higher final success rate. This stability stems from the agent’s  
 425 early mastery of valid actions, providing a robust foundation for strategic learning. Conversely, Loop  
 426 (blue) and GRPO (purple) show high volatility and inconsistent progress throughout training. This  
 427 instability is a direct consequence of credit misassignment, where the conflated reward signals lead  
 428 to contradictory policy updates. MGR’s design effectively mitigates this, resulting in more reliable  
 429 and efficient learning of task-level objectives. The bottom plot of Figure 3 shows the action success  
 430 rate. MGR (orange curve) demonstrates rapid and significant improvement, quickly reaching over  
 431 80% validity. This reflects the efficacy of our explicit action-level validity signal ( $R_{\text{local}}$ ), which en-  
 432 ables efficient learning of correct action syntax. In contrast, Loop (green) and GRPO (brown) show  
 433 minimal improvement, largely stagnating around 40%. Their reliance on a noisy, trajectory-level  
 434 reward makes it difficult to learn basic action correctness.

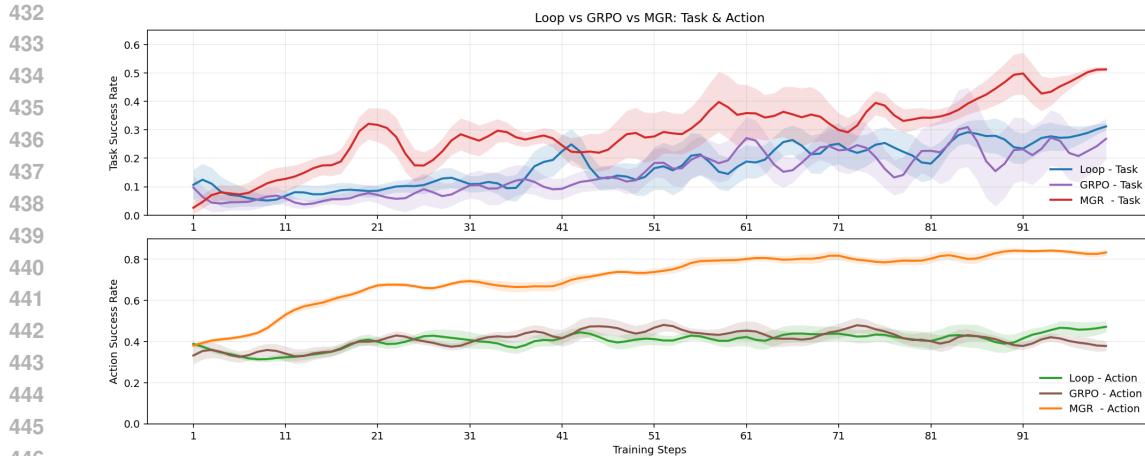


Figure 3: Training dynamics on AppWorld. Top plot: Task Success Rate. Bottom plot: Action Success Rate. MGR (red/orange) consistently outperforms Loop (blue/green) and GRPO (purple/brown) in both metrics, showing faster learning and greater stability.

#### 4.6 ABLATION STUDIES

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

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

Method	Easy	Medium	Difficult	Avg.
<b>MGR (Full Model)</b>	<b>75.4</b>	<b>22.8</b>	<b>10.2</b>	<b>36.1</b>
- 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

**Effect of the Gating Mechanism.** In the first ablation, we remove the stochastic gating operator  $g_t$  from our reward fusion logic (Eqn. 7). 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 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 update with a GRPO-style rule in which the advantage for every token in an action is set to the final action-level reward  $R_{final,t}$  (i.e., no critic model). 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

486 repeating machine. This failure becomes absolute on Medium and Difficult tasks, as they require  
 487 longer and more diverse interaction sequences.  
 488

489 **Effect of KL Divergence.** Our method, following prior work (Chen et al., 2025a), omits the KL  
 490 divergence term commonly found in the PPO objective function. To verify this design choice, we  
 491 reintroduce it in the "- w/ KL Divergence" variant. The results show a noticeable decline in perfor-  
 492 mance. This suggests that the KL penalty, which constrains the policy from moving too far from the  
 493 reference policy, can be overly restrictive for agentic finetuning. The complex exploration required  
 494 in these tasks benefits from the greater freedom afforded by removing this constraint, allowing the  
 495 policy to more aggressively adapt based on MGR's high-quality reward signals.

### 496 **Ablation on Hyperparameter Sensitivity**

497 **Sensitivity on Temporal Bonus ( $\beta$ )** To determine the  
 498 optimal value for the temporal bonus hyperparameter,  $\beta$ ,  
 499 which is designed to encourage error recovery, we con-  
 500 ducted a targeted ablation study on AppWorld. As demon-  
 501 strated in Table 3, our approach is sensitive to this value.  
 502 Removing the bonus entirely ( $\beta = 0$ ) results in the low-  
 503 est performance, confirming that an explicit incentive for  
 504 self-correction is beneficial. A modest bonus of  $\beta = 0.1$   
 505 achieves the peak success rate of 36.2%, striking an ef-  
 506 fective balance. However, we observed that larger bonus  
 507 values ( $\beta \geq 0.3$ ) lead to a slight performance degradation.

508 This suggests that an overly strong local incentive might  
 509 create a distracting signal, preventing the agent from focusing on the more crucial global strategic  
 510 objective. This study empirically validates our choice of  $\beta = 0.1$  for all main experiments, as it  
 511 provides a useful learning signal for local error recovery while preserving the integrity of the global  
 512 reward.

513 **More experiment can be seen in App. A.6**

## 516 5 CONCLUSION

517 We introduced Multiplicative Gated Rewards (MGR), a credit assignment framework that considers  
 518 local action validity and global effectiveness and fuses them multiplicatively to prevent reward-  
 519 ing invalid actions. A fact-based local sign gates a group-relative global magnitude, yielding an  
 520 implicit curriculum that first rewards producing valid actions and then prioritizes strategic composi-  
 521 tion. MGR delivers SOTA performance and markedly improved stability and sample efficiency on  
 522 ALFWorld and AppWorld.

## 525 6 ETHICS STATEMENT

526 All authors have read and adhere to the ICLR Code of Ethics. Our study does not involve human  
 527 subjects, private data, or potentially harmful applications; we foresee no ethical concerns beyond  
 528 standard academic practices.

## 531 7 REPRODUCIBILITY STATEMENT

532 We provide anonymized source code, complete proofs, and detailed data-processing descriptions in  
 533 the supplementary material to ensure full reproducibility. Hyper-parameters are listed in App. A.5.

## 536 REFERENCES

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

Table 3: Ablation on the temporal bonus hyperparameter  $\beta$ , measured by average success rate on AppWorld.

Value of $\beta$	Success Rate (%)
0.0	32.6
<b>0.1</b>	<b>36.2</b>
0.3	36.0
0.5	34.6

540 Yao Chen, Zhaofeng He, Geyu Lin, Zhidong Zhang, Hong-Ning L. Dai, Dong Li, and Wei Lo.  
 541 Spa-rl: Reinforcing llm agents via stepwise progress attribution, 2025b.  
 542

543 DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu,  
 544 Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu,  
 545 Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao  
 546 Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan,  
 547 Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao,  
 548 Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding,  
 549 Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang  
 550 Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai  
 551 Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang,  
 552 Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang,  
 553 Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang,  
 554 Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang,  
 555 R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuan Chen, Shengfeng  
 556 Ye, Shiyu Wang, Shuiping Yu, Shufeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing  
 557 Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen  
 558 Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong  
 559 Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu,  
 560 Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xi-  
 561 aoshua Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia  
 562 Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng  
 563 Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong  
 564 Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong,  
 565 Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou,  
 566 Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying  
 567 Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda  
 568 Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu,  
 569 Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu  
 570 Zhang, and Zhen Zhang. Deepseek-rl: Incentivizing reasoning capability in llms via reinforce-  
 571 ment learning, 2025. URL <https://arxiv.org/abs/2501.12948>.

572 Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia  
 573 Chen, Jiazen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization.  
 574 *arXiv preprint arXiv:2507.19849*, 2025.

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

577 Tian Gao, Jiatong Sun, Zishuai Geng, Wen-Ding Li, Xipeng Qiu, Yitao Liu, and Weizhe Yuan.  
 578 Process reward models for llm agents: Practical framework and directions, 2025.

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

582 Qwen, ;, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan  
 583 Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,  
 584 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin  
 585 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,  
 586 Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,  
 587 Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.  
 588 URL <https://arxiv.org/abs/2412.15115>.

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

591 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,  
 592 Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathe-  
 593 matical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.

594 Le Shi, Sijia Liu, Bowei He, Zhipeng Lucas Wu, Yuandong Tian, and Xiangjue Dong. Ragen:  
 595 Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025.  
 596

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

600

601 Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank  
 602 Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of  
 603 apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual*  
 604 *Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–  
 605 16076, 2024.

606

607 Zheyuan Wu, Zherui Li, Ziyi Lou, Chen Zhan, Xian-Ling Chen, and Chunyan Miao. SPA-RL:  
 608 Reinforcing LLM agents via stepwise progress attribution. *arXiv preprint arXiv:2505.20732*,  
 609 2025.

610

611 Guofu Xie, Yunsheng Shi, Hongtao Tian, Ting Yao, and Xiao Zhang. Capo: Towards enhancing llm  
 612 reasoning through generative credit assignment, 2025. URL <https://arxiv.org/abs/2508.02298>.

613

614 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu,  
 615 Chang Gao, Chengan Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint*  
 616 *arXiv:2505.09388*, 2025.

617

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

620

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

624

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

627

## 628 A APPENDIX

### 630 A.1 LLM USAGE DECLARATION

632 Authors prepared complete drafts of the manuscript independently. The large language model was  
 633 employed solely for language polishing and stylistic refinement; no content generation, data analy-  
 634 sis, or interpretation was performed by the LLM.

### 636 A.2 MGR CALCULATION IN A FAILURE TRAJECTORY

638 To provide a comprehensive understanding of the MGR framework, we present an illustrative exam-  
 639 ple of its reward calculation process within a failed trajectory, as depicted in Figure 4. This scenario  
 640 serves as a direct contrast to the successful trajectory shown in Figure 2 of the main paper, highlight-  
 641 ing MGR’s core mechanism for preventing credit misassignment for actions that are locally valid  
 642 but strategically ineffective.

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

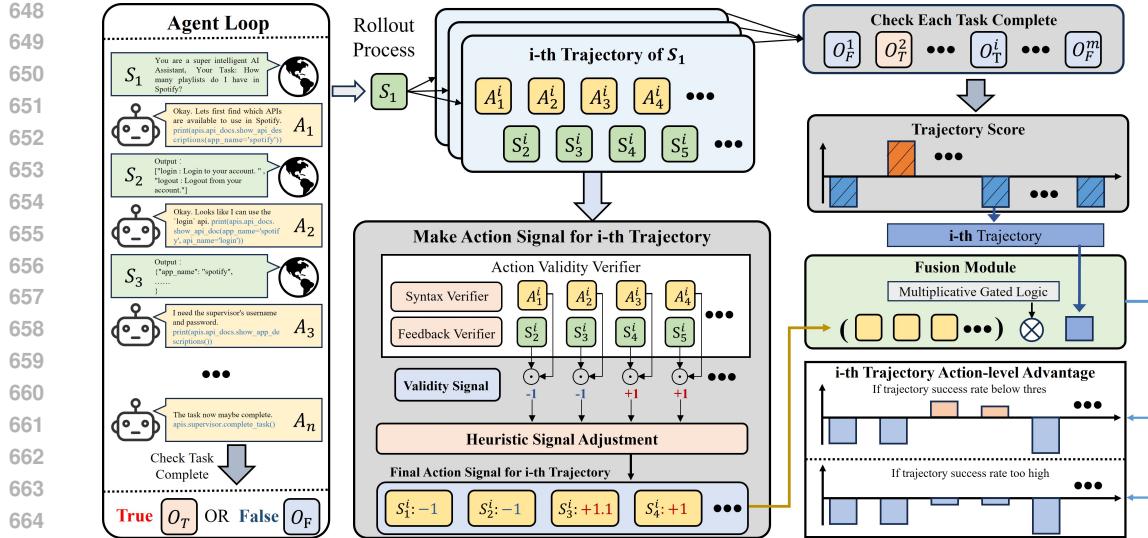


Figure 4: **Example of MGR Reward Calculation in a Failure Trajectory.** (Top row of final rewards) In early training,  $g_t$  is likely  $+1$ , preserving a small positive reward to encourage learning syntax. (Bottom row) In late training,  $g_t$  is likely  $-1$ , assigning a negative reward to penalize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

### A.3 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.4 ALGORITHM PSEUDOCODE

To formalize the training procedure, we present the MGR 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 reward calculation, each trajectory  $\tau_t$  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 Reward Calculation (Lines 14–15 & 19–22):** The algorithm computes rewards in a decoupled manner. The global, trajectory-level magnitude  $R_{\text{global}}$  is computed

702 first (Line 15). Then, for each segmented sample, the local, action-level sign  $R_{\text{local},t}$  is  
 703 derived from the environmental feedback contained in the subsequent state  $s_{t+1}$  (Line 20).  
 704

- 705 • **Conditional Multiplicative Fusion (Lines 24–29):** The core multiplicative gating logic  
 706 is shown. The final reward  $R_{\text{final},t}$  is determined by the alignment of the local and global  
 707 signals, with the stochastic gate  $g_t$  being the key component for the implicit curriculum.
- 708 • **Dynamic Gating (Lines 17–18 & 28):** The probability  $p_{\text{retain}}$  for the stochastic gate is dy-  
 709 namically calculated based on the agent’s batch-level performance ( $C_{\text{batch}}$ ,  $V_{\text{batch}}$ ), ensuring  
 710 the reward strategy adapts as the agent improves.

---

**Algorithm 1** Multiplicative Gated Rewards (MGR) Training Loop
 

---

711

712 **Require:** Initial policy  $\pi_\theta$ , Environment  $Env$   
 713 **Require:** Hyperparameters: batch size  $m$ , learning rate  $\eta$ , scaling coeff.  $\gamma$   
 714 **Require:** MGR Hyperparameters:  $\beta, \alpha, q$  for heuristics;  $\theta_v, \theta_{C1}, \theta_{C2}, \alpha_{\text{decay}}, p_{\text{min}}$  for curriculum

715 1: Initialize policy parameters  $\theta$   
 716 2: **for** each training iteration **do**  
 717 3: Initialize a raw trajectory buffer  $\mathcal{B}_{\text{raw}} \leftarrow \emptyset$   
 718 4: # — Phase 1: Trajectory Collection —  
 719 5: **for**  $i = 1$  to  $m$  **do**  
 720 6: Collect trajectory  $\tau_i = (s_1, a_1, s_2, \dots, s_{T_i}, a_{T_i})$  by executing  $\pi_\theta$  in  $Env$   
 721 7: Store  $(\tau_i, S(\tau_i))$  in  $\mathcal{B}_{\text{raw}}$ , where  $S(\tau_i) \in \{0, 1\}$  is the task success signal

722

723 8: # — Phase 2: Segmentation and Reward Calculation —  
 724 9: Initialize a training batch buffer  $\mathcal{D}_{\text{batch}} \leftarrow \emptyset$   
 725 10: # Step 2a: Segment trajectories into (prompt, action) samples  
 726 11: Compute  $C_{\text{batch}}$  (task completion rate) and  $V_{\text{batch}}$  (action validity rate) from  $\mathcal{B}_{\text{raw}}$   
 727 12: Compute positive sign retention probability  $p_{\text{retain}} \leftarrow f_{\text{schedule}}(C_{\text{batch}}, V_{\text{batch}})$  ▷ Equation 7  
 728 13: **for** each trajectory  $\tau_i \in \mathcal{B}_{\text{raw}}$  **do**  
 729 14: Compute trajectory-level magnitude  $R_{\text{global}}(\tau_i) \leftarrow \frac{m}{m-1} \left( S(\tau_i) - \frac{1}{m} \sum_{j=1}^m S(\tau_j) \right)$  ▷  
 730 Equation 5

731 15: **for**  $t = 1$  to  $T_i$  **do**  
 732 16: Define prompt  $p'_t \leftarrow (s_1, a_1, \dots, s_t)$  and action  $a_t$  from  $\tau_i$   
 733 17: Get next state  $s_{t+1}$  from  $\tau_i$  (or final state if  $t = T_i$ )  
 734 18: # Step 2b: Compute action-level signals and fuse for final reward  
 735 19:  $v_t \leftarrow \text{ComputeFundamentalValidity}(a_t, s_{t+1})$  ▷ Feedback is in  $s_{t+1}$   
 736 20:  $h_t \leftarrow \text{ComputeHeuristicAdjustments}(a_{1..t}, v_{1..t}, \alpha, \beta, q)$  ▷ Temporal & Repetition  
 737 21:  $R_{\text{local},t} \leftarrow v_t + h_t$  ▷ Equation 4

738 22: # Multiplicative Gating Fusion Logic  
 739 23: **if**  $\text{sgn}(R_{\text{local},t}) = \text{sgn}(R_{\text{global}}(\tau_i))$  **then**  
 740 24:  $R_{\text{final},t} \leftarrow R_{\text{local},t} \cdot |R_{\text{global}}(\tau_i)|$   
 741 25: **else if**  $R_{\text{global}}(\tau_i) > 0$  and  $R_{\text{local},t} < 0$  **then** ▷ Successful traj, invalid action  
 742 26:  $R_{\text{final},t} \leftarrow \gamma \cdot R_{\text{local},t} \cdot |R_{\text{global}}(\tau_i)|$  ▷ Dampened penalty  
 743 27: **else** ▷ Failed traj, valid action  
 744 28:  $g_t \sim \text{GatedSign}(p_{\text{retain}})$  ▷ Sample gate:  $g_t \in \{+1, -1\}$   
 745 29:  $R_{\text{final},t} \leftarrow g_t \cdot \gamma \cdot R_{\text{local},t} \cdot |R_{\text{global}}(\tau_i)|$   
 746 30: Add  $(p'_t, a_t, R_{\text{final},t})$  to training batch  $\mathcal{D}_{\text{batch}}$

747

748 31: # — Phase 3: Policy Update —  
 749 32: Compute advantages  $\hat{A}_t$  for all samples in  $\mathcal{D}_{\text{batch}}$  using GAE on rewards  $R_{\text{final},t}$   
 750 33: Update policy parameters  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}^{\text{PPO}}(\theta)$  using samples from  $\mathcal{D}_{\text{batch}}$  ▷ Equation 2

---

752

753 **A.5 IMPLEMENTATION DETAILS**

754

755 **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

756 agent's ability to reason, plan, and ground language in a sequence of actions to achieve high-level  
 757 goals (e.g., "put a warm apple in the microwave"). We report performance on three difficulty splits—  
 758 L0, L1, and L2:

760 • **L0** : Pick and Look.  
 761 • **L1** : Clean and Heat.  
 762 • **L2** : Cool and Pick2.

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

768 For AppWorld, we follow the setting of Loop (Chen et al., 2025a). We train on a subset of the  
 769 AppWorld train set, excluding difficulty 3 tasks. This subset consists of 24 scenarios, with 3 minor  
 770 variations (tasks) per scenario, which contain 72 tasks. Each iteration starts with the generation of K  
 771 = 6 rollouts with temperature 1.0 for 40 randomly sampled tasks, for a total of 240 rollouts. We use  
 772 only difficulty-1 and difficulty-2 tasks for training. The maximum response length is 2048 tokens.  
 773 Each episode allows up to 35 environment steps. The task IDs we use are as follows:

774  
 775 07b42fd 229360a 27e1026 287e338 692c77d 82e2fac  
 776 aa8502b b7a9ee9 c901732 ccb4494 ce359b5 e7a10f8  
 777 e85d92a e3d6c94 d0b1f43 2a163ab 60d0b5b 6ea6792  
 778 29caf6f cf6abd2 771d8fc 7d7fbf6 76f2c72 302c169  
 779

780  
 781 **Action Validity Signal ( $R_{local}$ ) Determination** The core of our Multiplicative Gated Rewards  
 782 (MGR) framework is the ability to decouple local action validity from global trajectory success.  
 783 This is primarily achieved through the action-level signal,  $R_{local}$ , which begins with a **Fundamental**  
 784 **Validity Signal** ( $v_t \in \{-1, +1\}$ ). This signal provides an immediate, fact-based judgment on the  
 785 correctness of each action  $a_t$ . An action is considered invalid ( $v_t$  is set to -1) if it fails any of the  
 786 checks detailed below for each environment. Otherwise, it is considered valid ( $v_t$  is set to +1).

787  
 788 **AppWorld** In the AppWorld benchmark, agent actions consist of Python code designed to interact  
 789 with a suite of simulated applications. An action is determined to be **invalid** if it meets one or more  
 790 of the following criteria, categorized into environment feedback analysis and action format checks.  
 791

792 1. **Environment Feedback Analysis:** A negative validity signal ( $v_t = -1$ ) is assigned if the  
 793 environment's execution feedback contains any of the following error-indicating substrings.  
 794 These keywords typically signify crashes, syntax errors, or execution timeouts:

795 • "Execution failed"  
 796 • "Traceback:"  
 797 • "SyntaxError"  
 798 • "Exception"  
 799 • "Error:"  
 800 • "Maximum number of executions"  
 801 • "timed out after"  
 802 • "No code available to execute"

803  
 804 2. **Action Format and Syntax Analysis:** Additionally, the raw action generated by the agent  
 805 is preemptively checked for structural integrity. The action is considered invalid ( $v_t = -1$ )  
 806 if it fails any of these checks:

807 • **Invalid Formatting:** The generated code is not correctly encapsulated within a  
 808 ``python ... `` markdown block, making it unexecutable by the environ-  
 809 ment 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 rewards 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 heuristic 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  $\alpha = 1.5$ , 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.

## A.6 HYPERPARAMETER SENSITIVITY ANALYSIS

**Sensitivity on Scaling Coefficient ( $\gamma$ )** We further investigated the robustness of the scaling coefficient,  $\gamma$ , 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  $\gamma$  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

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

Value of $\gamma$	Success Rate (%)
0.8	35.4
<b>1.0</b>	<b>36.1</b>
1.2	35.9
1.4	35.2

864  
 865 invalid actions (preserving the negative sign), the agent  
 866 effectively learns to avoid them without being overly sen-  
 867 sitive to the specific scale of the penalty.

868  
 869 **Ablation on Repetition Penalty ( $\alpha$ )** To assess the ne-  
 870 cessity of explicit constraints against reward hacking, we  
 871 evaluated the impact of the repetition penalty coefficient  
 872  $\alpha$ . Table 5 reveals a critical insight: removing the penalty  
 873 entirely ( $\alpha = 0$ ) leads to a catastrophic collapse in per-  
 874 formance (5.1%), as the agent degenerates into repeating  
 875 a single locally valid action to exploit the reward signal.  
 876 However, once the penalty is enabled ( $\alpha \geq 0.5$ ), the per-  
 877 formance stabilizes at a high level. While larger penalties  
 878 (e.g.,  $\alpha = 2.0$ ) cause a minor drop by discouraging legit-  
 879 imate retries, the method remains effective across a wide  
 880 range of non-zero values. This confirms that the presence  
 881 of the penalty mechanism is a fundamental requirement for stability, but the framework is not brittle  
 882 regarding its specific hyperparameter tuning.

883  
 884  
 885  
 886  
 887  
 888  
 889  
 890  
 891  
 892  
 893  
 894  
 895  
 896  
 897  
 898  
 899  
 900  
 901  
 902  
 903  
 904  
 905  
 906  
 907  
 908  
 909  
 910  
 911  
 912  
 913  
 914  
 915  
 916  
 917

Table 5: Impact of the repetition penalty hyperparameter  $\alpha$  on AppWorld success rates.

Value of $\alpha$	Success Rate (%)
0.0	5.1
<b>0.5</b>	<b>36.1</b>
1.0	34.2
2.0	34.5