Towards Smart Exploration: Reinforcement Learning for LLM Agents Guided by Meta-Actions

Hanwen Tong, Zhengqi Zhang, Ruixuan Lu, Jincheng Dai,
Bin Yang, Bo Zheng, Feng Wei*
MYbank, Ant Group
tonghanwen.thw@mybank.cn

Abstract

Reinforcement Learning (RL) has empowered Large Language Model (LLM) agents to tackle long-horizon tasks through multi-step interactions with the environment. However, current LLM agents still rely on inefficient trial-and-error exploration, leading to frequent task failures and redundant actions. In this paper, we propose *MetaAct*, an RL framework that incorporates a set of high-level meta-actions to explicitly guide the agent towards promising trajectories and facilitate goal-aware exploration. Specifically, the agent is required to predict one of four meta-actions (`FOCUS`, `GUESS`, `RESUME`, `UNDO`) to constrain the direction of its subsequent actions to minimize unproductive trials. More importantly, we employ group-based RL with episode-level and step-level advantages to fine-tune the agent, enabling it to identify strategic directions more accurately and generate effective task-specific actions. Experimental results demonstrate that our method consistently outperforms the state-of-the-art approaches on ALFWorld and WebShop, achieving higher success rates with fewer steps. Our code and data are publicly available at https://github.com/HavenTong/MetaAct.

1 Introduction

Building autonomous agents (Russell and Norvig, 1995; Franklin and Graesser, 1996) that accomplish goals without human intervention is a core objective of Artificial General Intelligence (AGI). Recent advances in Large Language Models (LLMs) (Jaech et al., 2024; Guo et al., 2025) have unlocked their remarkable reasoning capabilities, extending from single-step reasoning (Wei et al., 2022) to multi-step decision-making. Consequently, LLM-powered autonomous agents are being applied to a wide range of domains, including web navigation (Zheng et al., 2024; Zhou et al., 2024a), coding (Trivedi et al., 2024; Wei et al., 2026) and scientific discovery (Chen et al., 2025; Lou et al., 2025), posing the challenge of solving long-horizon tasks through multi-step interactions with real-world environments.

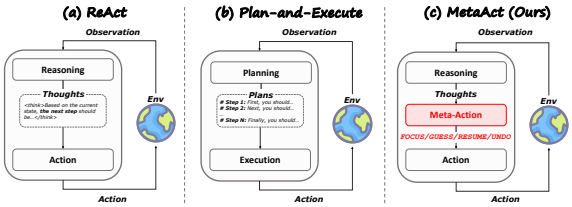


Figure 1: The comparison between ReAct, Plan-and-Execute and *MetaAct* (ours).

To tackle such tasks, existing approaches propose various prompting paradiagms, such as ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023), to encourage LLM agents to reason and reflect using environmental feedback. Further studies augment agents with planning (Erdogan et al., 2025), searching (Li et al., 2025b) and memory (Xu et al., 2026) to facilitate reasoning during multi-step decision-making. However, static prompts and hand-crafted workflows often lead to low success rates in real-world, long-horizon settings. Recent works adopt Reinforcement Learning (RL) (Wang et al., 2025; Feng et al., 2025; Kalyan and Andrews, 2025) to train LLM agents through direct interaction with the environment, enabling adaptation to dynamic environments. However, they still rely on inefficient trial-and-error exploration, leading to frequent task failures, redundant actions, and low adaptivity on unseen tasks and environments.

We argue that current LLM agents **lack explicit goal awareness**, which limits their ability to identify promising directions towards task completion. To overcome this limitation, inspired by Meta-Planning (Stefik, 1981), we introduce four meta-actions (`FOCUS`, `GUESS`, `RESUME`, `UNDO`), a set of high-level logical actions governing the strategic direction of subsequent task-specific actions. Their

*Corresponding author.

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 4369–4388
November 6-10, 2026 ©2026 Association for Computational Linguistics

Meta-action	When to use	Content
FOCUS	You are on a deterministic path.	Take a logically determined action with no uncertainty.
GUESS	It is insufficient to determine a unique action.	Make a heuristic choice to explore or try a plausible path.
RESUME	A previously paused sub-goal becomes feasible after acquiring new information.	Continue the previously paused sub-goal.
UNDO	You detect a conflict, mistake, or dead end.	Backtrack to correct errors or escape the dead end.

Table 1: The illustrations of four meta-actions.

definitions are in Table 1. Consider the household task of *"putting a cooled cup in the cabinet"*. FOCUS applies when the next action is logically deterministic (*"You arrive at a cabinet with a cooled cup."*). GUESS triggers when it is insufficient to determine a unique action (*"The cup might be located on a countertop, shelf, or table."*). RESUME applies when a previously paused sub-goal becomes feasible after acquiring new information (*"Now you have found a cup and should begin to find a way to cool it."*). UNDO triggers when there is a conflict, mistake, or dead-end (*"There is no cup on any table, but you keep finding a cup on a table."*). According to Stefik (1981), they form a minimal and functionally orthogonal set that covers all fundamental planning states via two complementary cycles: A deterministic cycle (FOCUS, RESUME) and an exploratory cycle (GUESS, UNDO). By identifying one of these meta-actions prior to task-specific actions, as illustrated in Figure 1, it encourages goal-aware exploration with stronger strategic foresight than ReAct agents. Unlike Plan-and-Execute agents that decouple planning from execution and often produce ungrounded plans, it flexibly adjusts the direction of subsequent actions during long-horizon tasks.

In this paper, we propose *MetaAct*, an RL framework that incorporates a set of meta-actions to explicitly guide LLM agents towards promising trajectories and facilitate goal-aware exploration. It consists of an LLM agent guided by meta-actions and a group-based RL algorithm with two-level advantage estimation to fine-tune the agent. Specifically, we augment the LLM agents with four meta-actions (FOCUS, GUESS, RESUME, UNDO) to direct subsequent task-specific actions. Given a task that requires multi-step interactions with the environment, for each time step, the agent first identifies one of the four meta-actions based on the current state and task description, and then generates the task-specific action guided by this meta-action. Upon executing the action, the environment returns the reward and next state. The process continues until the task is completed or the maximum episode budget is reached. More importantly, we fine-tune the agent through group-based RL with episode-level and step-level advantages. Specifically, we roll out multiple episodes based on the same task description and initial state to form an episode group. Then we estimate the episode-level advantage through group normalization to measure trajectory quality. Furthermore, we construct step-level groups by clustering the meta-actions and actions taken under the same state, and estimate their step-level advantages to capture their contribution to task completion. Finally, we combine the two-level advantages for policy optimization.

Our contributions are summarized as follows:

- We propose a novel LLM agent framework that leverages high-level meta-actions for goal-aware exploration in long-horizon tasks. The characteristic of our framework is that it requires the agent to identify its meta-action prior to task-specific action, constraining its direction towards promising trajectories.
- We propose a group-based RL algorithm with episode-level and step-level advantages to fine-tune the LLM agent with meta-actions, enabling it to identify strategic directions and generate effective task-specific actions.
- We conduct extensive experiments on two challenging benchmarks. Experimental results demonstrate that our method consistently outperforms the state-of-the-art approaches. Further analysis reveals that our framework ensures smart exploration, achieving higher success rates with fewer steps.

2 Related Work

Large Language Model Agents. Large Language Models (LLMs) have evolved from single-step reasoning (Wei et al., 2022) to agents that interact with external tools and environments (Schick et al., 2023), enabling applications in scientific research (Hubert et al., 2026; Li et al., 2025b), coding (Zhang et al., 2024), and simulation (Wang

et al., 2024; Xi et al., 2025; Ye et al., 2025). Most existing LLM agents rely on prompting paradigms such as ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023), often enhanced with planning and memory modules to handle more complex tasks (Li et al., 2025a; Xu et al., 2026). Recent studies further extend single agent to multi-agent systems, where multiple agents collaborate on tasks (Qian et al., 2025; Bo et al., 2024). Despite these advances, agents built on static prompts and predefined workflows remain brittle in long-horizon tasks with only partial observation. They often generalize poorly to unseen scenarios, leading to error accumulation and low success rates. To address this limitation, many studies adopt Reinforcement Learning (RL) (Sutton and Barto, 2018) to train robust agents for multi-step interactions.

Reinforcement Learning for LLM Agents. Applying RL to LLM agents for multi-step tasks introduces unique challenges (Zhang et al., 2025, 2026). Recent works primarily address the sparse and delayed rewards inherent in long-horizon problems and propose strategies for fine-grained credit assignment. These include designing reasoning-aware reward functions (RAGEN) (Wang et al., 2025), shaping rewards at the step level (Zeng et al., 2025), inferring implicit process rewards without explicit labels (Liu et al., 2026), and estimating hierarchical advantages (GiGPO) (Feng et al., 2025). Concurrently, another line of research aims to improve training efficiency and stability through hierarchical RL (Zhou et al., 2024b; Peng et al., 2026), flow-based optimization (Li et al., 2026) that makes on-policy training more tractable, and Meta-RL for rapid in-context adaptation across episodes (Jiang et al., 2026). The growth of this subfield is further supported by newly emerged training frameworks and practical guides (Wang and Ammanabrolu, 2025; Cheng et al., 2025; Xi et al., 2026). However, despite these advances in reward engineering and training algorithms, a critical challenge remains largely unresolved: inefficient exploration. Current LLM agents still exhibit trial-and-error exploration over long horizons, leading to repetitive action loops and task failures.

3 Overview

In this section, we first formalize the setting where an LLM agent interacts with the environment to complete a task. Then we overview our framework *MetaAct*, which is illustrated in Figure 2.

3.1 Problem Formulation

Given a task description $\boldsymbol{x} \in \mathcal{X}$, the goal of the LLM agent π is to interact with the environment over multiple steps to complete the task. Specifically, for each time step $t \in \{1, 2, ..., T\}$, the agent receives an observation $\boldsymbol{s}_t \in \mathcal{S}$ from the environment and generates an action $\boldsymbol{a}_t \in \mathcal{V}^n$, where $\mathcal{S}$ denotes the state space, $\mathcal{V}$ denotes the vocabulary space of the LLM and n denotes the sequence length. The behaviour of the LLM agent $\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{x})$ is modeled by a probability distribution over outputs conditioned on the state $\boldsymbol{s}_t$ and the task description $\boldsymbol{x}$, parameterized by θ. Upon executing the action $\boldsymbol{a}_t$, the environment returns a scalar reward r_t and the next state $\boldsymbol{s}_{t+1}$. Consequently, an episode is represented by the trajectory $\boldsymbol{\tau} = \{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t)\}_{t=1}^{T}$. In practice, the environment may provide sparse, delayed rewards at the end of the episode to indicate task success or weak process rewards in the intermediate time steps.

3.2 Framework

Our framework consists of an LLM agent guided by meta-actions and a group-based RL algorithm with two-level advantages for fine-tuning. Given a task description $\boldsymbol{x} \in \mathcal{X}$, we augment the agent with four meta-actions (`FOCUS`, `GUESS`, `RESUME`, `UNDO`). For each time step $t \in \{1, 2, ..., T\}$, the agent first produces one of four meta-actions $\boldsymbol{m}_t$ based on the current state $\boldsymbol{s}_t$ and task description $\boldsymbol{x}$, and then generates a task-specific action $\boldsymbol{a}_t$ guided by $\boldsymbol{m}_t$. Upon executing $\boldsymbol{a}_t$, the environment yields a reward r_t and the next state $\boldsymbol{s}_{t+1}$. An episode is represented by the trajectory $\boldsymbol{\tau} = \{(\boldsymbol{s}_t, \boldsymbol{m}_t, \boldsymbol{a}_t, r_t)\}_{t=1}^{T}$. We further fine-tune the agent through group-based RL using both episode-level and step-level advantages. Specifically, we roll out the agent π_θ for N times with the same task description $\boldsymbol{x}$ and initial state $\boldsymbol{s}_1$, yielding N trajectories $\{\boldsymbol{\tau}^{(i)}\}_{i=1}^{N}$, along with their returns $\{R(\boldsymbol{\tau}^{(i)})\}_{i=1}^{N}$. Consequently, we estimate the episode-level advantages $\{A^E(\boldsymbol{\tau}^{(i)})\}_{i=1}^{N}$ through group normalization. We further perform anchor state grouping following Feng et al. (2025) to cluster the meta-actions and actions under the same state $\tilde{\boldsymbol{s}}$, respectively. This enables the estimation of the step-level advantages $A^S(\boldsymbol{m}_t^{(i)})$ and $A^S(\boldsymbol{a}_t^{(i)})$ for meta-actions and actions through group normalization. Finally, we combine two-level advantages through weighted sum and use the aggregated advantage for policy optimization of the LLM agent.

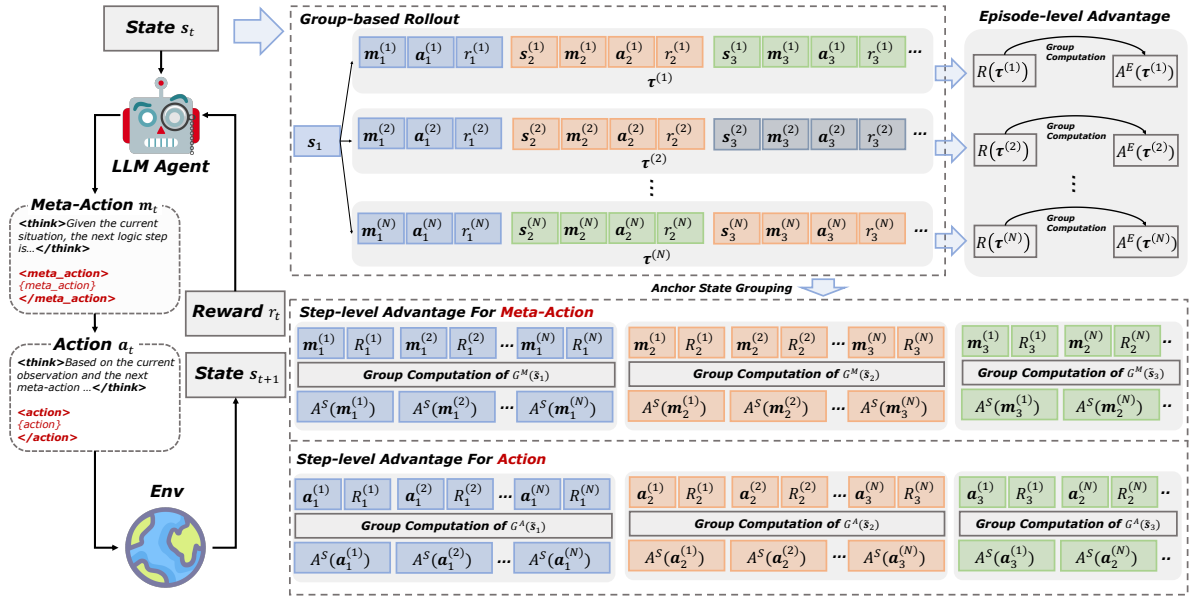


Figure 2: The overview of *MetaAct*. It consists of an LLM agent guided by meta-actions (left) and the group-based RL algorithm that fine-tunes the agent with both episode-level and step-level advantages (right).

4 Methodology

In this section, we detail our framework, including the LLM agent guided by meta-actions and the RL algorithm with two-level advantages.

4.1 LLM Agent Guided by Meta-Actions

The goal of the LLM agent guided by meta-actions is to complete multi-step tasks through goal-aware exploration. Current LLM agents often exhibit trial-and-error exploration, leading to task failures and redundant actions. To overcome this limitation, we propose that the agent generates one of four meta-actions (`FOCUS`, `GUESS`, `RESUME`, `UNDO`) prior to each task-specific action, guiding the agent towards promising trajectories. Specifically, given the task description $\boldsymbol{x} \in \mathcal{X}$, for each time step $t \in \{1, 2, ..., T\}$, the LLM agent π_θ first generates the meta-action $\boldsymbol{m}_t$ based on the current state $\boldsymbol{s}_t$ and task description $\boldsymbol{x}$, i.e., $\boldsymbol{m}_t = \pi_\theta(\cdot|\boldsymbol{s}_t, \boldsymbol{x})$. Next, the agent generates the task-specific action $\boldsymbol{a}_t$ guided by $\boldsymbol{m}_t$ based on $\boldsymbol{s}_t$ and $\boldsymbol{x}$, i.e., $\boldsymbol{a}_t = \pi_\theta(\cdot|\boldsymbol{m}_t, \boldsymbol{s}_t, \boldsymbol{x})$. After executing $\boldsymbol{a}_t$ in the environment, the agent receives a reward r_t and the next state $\boldsymbol{s}_{t+1}$. Ultimately, an episode is represented by the trajectory $\boldsymbol{\tau} = \{(\boldsymbol{s}_t, \boldsymbol{m}_t, \boldsymbol{a}_t, r_t)\}_{t=1}^{T}$. The prompts for the agent are illustrated in Appendix B.

4.2 RL for LLM Agent with Meta-Actions

The goal of fine-tuning the LLM agent with meta-actions is to enable it to identify the promising directions toward the goal from the current state and generate effective, task-specific actions to complete long-horizon tasks. Existing group-based RL algorithms such as GRPO (Shao et al., 2024) and RLOO (Ahmadian et al., 2024) assign a uniform episode-level advantage to all steps, failing to distinguish effective actions from ineffective ones and thus limiting performance on long-horizon tasks. Therefore, inspired by GiGPO (Feng et al., 2025), we incorporate two levels of advantages: (1) an episode-level advantage computed from a group of complete trajectories to measure overall trajectory quality; (2) a step-level advantage computed from a group of meta-actions or actions under the same state to capture their individual effectiveness. In the following sections, we first detail the episode-level advantage and the step-level advantage with the anchor state grouping mechanism, followed by the policy optimization of the LLM agent.

Episode-level Advantage. The goal of the episode-level advantage is to provide a global training signal that indicates whether the task is successfully completed. Specifically, we roll out the LLM agent for N times with the same task description $\boldsymbol{x}$ and initial state $\boldsymbol{s}_1$, yielding N complete trajectories $\{\boldsymbol{\tau}^{(i)}\}_{i=1}^{N}$. For each trajectory $\boldsymbol{\tau}^{(i)}$, we define its return as $R(\boldsymbol{\tau}^{(i)}) = \sum_t r_t^{(i)}$, where $r_t^{(i)}$ denotes the reward r_t in trajectory $\boldsymbol{\tau}^{(i)}$. In practice, the environment may only provide a sparse reward at the end of each episode to indicate whether the task is successfully completed. Therefore, a successful trajectory $\boldsymbol{\tau}^{(i)}$ has a return of $R(\boldsymbol{\tau}^{(i)}) = 1$, while

a failed trajectory has $R(\boldsymbol{\tau}^{(i)}) = 0$. Consequently, we organize the above N rollouts along with their returns into a group G^E described in equation (1):

$$G^E = \{(\boldsymbol{\tau}^{(i)}, R(\boldsymbol{\tau}^{(i)}))\}_{i=1}^N \tag{1}$$

Within G^E, we compute the episode-level advantage $A^E(\boldsymbol{\tau}^{(i)})$ of each trajectory $\boldsymbol{\tau}^{(i)}$ using group normalization, which is described in equation (2):

$$A^E(\boldsymbol{\tau}^{(i)}) = \frac{R(\boldsymbol{\tau}^{(i)}) - \text{mean}(\{R(\boldsymbol{\tau}^{(j)})\}_{j=1}^N)}{F_{\text{norm}}(\{R(\boldsymbol{\tau}^{(j)})\}_{j=1}^N)} \tag{2}$$

where $\text{mean}(\cdot)$ is the arithmetic mean and $F_{\text{norm}}(\cdot)$ is the normalization factor. Although GRPO (Shao et al., 2024) defines this factor as the standard deviation ($F_{\text{norm}} = \text{std}$), recent work (Liu et al., 2025) shows that it may cause the trajectories in a low-variance group (e.g., all trajectories are successful or failed) to receive higher weights during policy updates, leading to biased advantage estimation and unstable training. Therefore, we adopt both $F_{\text{norm}} = \text{std}$ and $F_{\text{norm}} = 1$ following GiGPO (Feng et al., 2025), and compare their performance in Section 5.2.

Step-level Advantage. The goal of the step-level advantage is to provide a fine-grained training signal that indicates which meta-actions guide towards promising trajectories and which actions contribute to successful outcomes. To achieve this, we build step-level groups to estimate advantages of meta-actions and actions. Since all N trajectories $\{\boldsymbol{\tau}^{(i)}\}_{i=1}^N$ are rolled out from the same task description and initial state, there are many shared states across trajectories or even within a single trajectory. Therefore, we follow GiGPO (Feng et al., 2025) and adopt an anchor state grouping strategy that groups the meta-actions and actions under the same state. Specifically, we first collect all the unique states that appear in the N trajectories to form the state set $\mathcal{U} = \{\tilde{\boldsymbol{s}}_i\}_{i=1}^U$. Then we treat each $\tilde{\boldsymbol{s}} \in \mathcal{U}$ as the *"anchor state"* to group the meta-actions and actions with corresponding rewards under $\tilde{\boldsymbol{s}}$, as described in equation (3) and (4):

$$G^M(\tilde{\boldsymbol{s}}) = \left\{(\boldsymbol{m}_t^{(i)}, r_t^{(i)}) \;\middle|\; \substack{\boldsymbol{s}_t^{(i)}=\tilde{\boldsymbol{s}}, \\ i\in[1,N], t\in[1,T]}\right\} \tag{3}$$

$$G^A(\tilde{\boldsymbol{s}}) = \left\{(\boldsymbol{a}_t^{(i)}, r_t^{(i)}) \;\middle|\; \substack{\boldsymbol{s}_t^{(i)}=\tilde{\boldsymbol{s}}, \\ i\in[1,N], t\in[1,T]}\right\} \tag{4}$$

where $G^M(\tilde{\boldsymbol{s}})$ and $G^A(\tilde{\boldsymbol{s}})$ denote the groups of meta-action and action under $\tilde{\boldsymbol{s}}$, respectively.

However, real-world environment may only provide a sparse reward at the end of each episode, i.e., $r_t^{(i)} = 0$ for intermediate steps. To mitigate this sparsity and capture the long-term effects of both meta-actions and actions, we compute their discounted returns instead. Specifically, since meta-actions and actions are generated alternatively, i.e., $\{\boldsymbol{m}_1, \boldsymbol{a}_1, \boldsymbol{m}_2, \boldsymbol{a}_2, ..., \boldsymbol{m}_T, \boldsymbol{a}_T\}$, we compute their discounted returns following equation (5) and (6):

$$R(\boldsymbol{m}_t^{(i)}) = \gamma R(\boldsymbol{a}_t^{(i)}) \tag{5}$$

$$R(\boldsymbol{a}_t^{(i)}) = r_t^{(i)} + \gamma R(\boldsymbol{m}_{t+1}^{(i)}),\; R(\boldsymbol{a}_T^{(i)}) = r_T^{(i)} \tag{6}$$

where $R(\boldsymbol{m}_t^{(i)})$ and $R(\boldsymbol{a}_t^{(i)})$ denote the discounted returns of the meta-action $\boldsymbol{m}_t^{(i)}$ and action $\boldsymbol{a}_t^{(i)}$, respectively. $\gamma \in (0, 1]$ is the discount factor. Consequently, the step-level groups of meta-action and action are described in equation (7) and (8):

$$G^M(\tilde{\boldsymbol{s}}) = \left\{(\boldsymbol{m}_t^{(i)}, R(\boldsymbol{m}_t^{(i)})) \;\middle|\; \substack{\boldsymbol{s}_t^{(i)}=\tilde{\boldsymbol{s}}, \\ i\in[1,N], t\in[1,T]}\right\} \tag{7}$$

$$G^A(\tilde{\boldsymbol{s}}) = \left\{(\boldsymbol{a}_t^{(i)}, R(\boldsymbol{a}_t^{(i)})) \;\middle|\; \substack{\boldsymbol{s}_t^{(i)}=\tilde{\boldsymbol{s}}, \\ i\in[1,N], t\in[1,T]}\right\} \tag{8}$$

Eventually, for each $\tilde{\boldsymbol{s}} \in \mathcal{U}$, we compute the step-level advantages of meta-action $A^S(\boldsymbol{m}_t^{(i)})$ and action $A^S(\boldsymbol{a}_t^{(i)})$ within $G^M(\tilde{\boldsymbol{s}})$ and $G^A(\tilde{\boldsymbol{s}})$, respectively, which is described in equation (9) and (10):

$$A^S(\boldsymbol{m}_t^{(i)}) = \frac{R(\boldsymbol{m}_t^{(i)}) - \text{mean}(\{R(\boldsymbol{m}_t^{(j)}) \mid (\boldsymbol{m}_t^{(j)}, R) \in G^M(\tilde{\boldsymbol{s}})\})}{F_{\text{norm}}(\{R(\boldsymbol{m}_t^{(j)}) \mid (\boldsymbol{m}_t^{(j)}, R) \in G^M(\tilde{\boldsymbol{s}})\})} \tag{9}$$

$$A^S(\boldsymbol{a}_t^{(i)}) = \frac{R(\boldsymbol{a}_t^{(i)}) - \text{mean}(\{R(\boldsymbol{a}_t^{(j)}) \mid (\boldsymbol{a}_t^{(j)}, R) \in G^A(\tilde{\boldsymbol{s}})\})}{F_{\text{norm}}(\{R(\boldsymbol{a}_t^{(j)}) \mid (\boldsymbol{a}_t^{(j)}, R) \in G^A(\tilde{\boldsymbol{s}})\})} \tag{10}$$

In this way, we evaluate meta-actions and actions within $G^M(\tilde{\boldsymbol{s}})$ and $G^A(\tilde{\boldsymbol{s}})$ using their respective step-level advantages, $A^S(\boldsymbol{m}_t^{(i)})$ and $A^S(\boldsymbol{a}_t^{(i)})$.

Policy Optimization for LLM Agent. The goal of policy optimization is to incorporate episode-level and step-level advantages to update the policy π_θ of the LLM agent guided by meta-actions, maximizing its expected returns. Specifically, we aggregate two-level advantages through a weighted sum, which is described in equation (11) and (12):

$$A(\boldsymbol{m}_t^{(i)}) = A^E(\boldsymbol{\tau}^{(i)}) + \omega^M A^S(\boldsymbol{m}_t^{(i)}) \tag{11}$$

$$A(\boldsymbol{a}_t^{(i)}) = A^E(\boldsymbol{\tau}^{(i)}) + \omega^A A^S(\boldsymbol{a}_t^{(i)}) \tag{12}$$

where $A(\boldsymbol{m}_t^{(i)})$ and $A(\boldsymbol{a}_t^{(i)})$ represent the final advantages of the meta-action $\boldsymbol{m}_t^{(i)}$ and action $\boldsymbol{a}_t^{(i)}$, respectively. $\omega^M \in \mathbb{R}^+$ and $\omega^A \in \mathbb{R}^+$ denote the weighting coefficients between the episode-level and step-level advantages of $\boldsymbol{m}_t^{(i)}$ and $\boldsymbol{a}_t^{(i)}$, respectively. Consequently, we optimize the policy π_θ

of the agent using a clipped policy optimization objective described in equation (13):

$$\mathcal{J}(\theta) = \mathbb{E}_{\substack{\boldsymbol{x}\sim\mathcal{X} \\ \{\boldsymbol{\tau}^{(i)}\}_{i=1}^{N}\sim\pi_{\theta_{\text{old}}}}} \left[\frac{1}{2NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\mathcal{L}_{clip}(\boldsymbol{m}_t^{(i)}) + \mathcal{L}_{clip}(\boldsymbol{a}_t^{(i)})\right)\right] - \beta\mathbb{D}_{\text{KL}}\left(\pi_\theta(\cdot \mid \boldsymbol{x}) \,\|\, \pi_{\text{ref}}(\cdot \mid \boldsymbol{x})\right) \tag{13}$$

where $\pi_{\theta_{\text{old}}}$ denotes the old policy, $\mathbb{D}_{\text{KL}}(\cdot \,\|\, \cdot)$ denotes the KL penalty term that keeps the policy π_θ from drifting too far from a reference policy π_{ref}, and β is the coefficient that controls the strength of penalty. The clipping loss items $\mathcal{L}_{clip}(\boldsymbol{m}_t^{(i)})$ and $\mathcal{L}_{clip}(\boldsymbol{a}_t^{(i)})$ are described in equation (14) and (15):

$$\mathcal{L}_{clip}(\boldsymbol{m}_t^{(i)}) = \min\Big(\rho_\theta(\boldsymbol{m}_t^{(i)})A(\boldsymbol{m}_t^{(i)}), \text{clip}\big(\rho_\theta(\boldsymbol{m}_t^{(i)}), 1\pm\epsilon\big)A(\boldsymbol{m}_t^{(i)})\Big) \tag{14}$$

$$\mathcal{L}_{clip}(\boldsymbol{a}_t^{(i)}) = \min\Big(\rho_\theta(\boldsymbol{a}_t^{(i)})A(\boldsymbol{a}_t^{(i)}), \text{clip}\big(\rho_\theta(\boldsymbol{a}_t^{(i)}), 1\pm\epsilon\big)A(\boldsymbol{a}_t^{(i)})\Big) \tag{15}$$

where $\rho_\theta(\boldsymbol{m}_t^{(i)}) = \frac{\pi_\theta(\boldsymbol{m}_t^{(i)}|\boldsymbol{s}_t^{(i)},\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{m}_t^{(i)}|\boldsymbol{s}_t^{(i)},\boldsymbol{x})}$ and $\rho_\theta(\boldsymbol{a}_t^{(i)}) = \frac{\pi_\theta(\boldsymbol{a}_t^{(i)}|\boldsymbol{m}_t^{(i)},\boldsymbol{s}_t^{(i)},\boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{a}_t^{(i)}|\boldsymbol{m}_t^{(i)},\boldsymbol{s}_t^{(i)},\boldsymbol{x})}$ denote the importance sampling ratio of $\boldsymbol{m}_t^{(i)}$ and $\boldsymbol{a}_t^{(i)}$, respectively. $\text{clip}(\cdot,\cdot)$ is the clipping function and ϵ is the clipping hyperparameter indicating how far new policy is allowed to go from the old. Consequently, we optimize the parameters θ of the LLM agent π_θ, aiming to maximize its policy optimization objective $\mathcal{J}(\theta)$.

5 Experiments

5.1 Experimental Setup

Benchmarks. We conduct experiments on two challenging benchmarks that require multi-step decision-making in an environment: ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022). ALFWorld is a text-based embodied environment for household tasks, where the agent must interact with the environment over multiple steps to complete them. It consists of six core task types: pick and place (Pick), examine in light (Look), clean and place (Clean), heat and place (Heat), cool and place (Cool), and pick two and place (Pick2). ALFWorld provides a binary reward $\in \{0, 1\}$ to indicate task success. WebShop is a web-based environment where the agent must navigate a simulated shopping website to locate and purchase specific items. It provides a dense reward $\in [0, 1]$ based on progress, and a task is considered successful only if the reward reaches 1. The statistics of the benchmarks are shown in Appendix A.1.

Baselines. We compare our method against three categories of baselines: (1) Closed-source LLMs, including GPT-4o (Achiam et al., 2023) and Gemini-2.5-Pro (Team et al., 2023). (2) Prompting-based methods, which design advanced strategies to prompt the LLMs to interact with the environments without parameter updates, including Zero-Shot, ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023). (3) RL-based methods, which adopt RL algorithm to fine-tune the LLM agent through environment feedback, including PPO (Schulman et al., 2017), RLOO (Ahmadian et al., 2024), GRPO (Shao et al., 2024) and GiGPO (Feng et al., 2025) (the state-of-the-art RL-based method).

Implementation Details. We adopt Qwen2.5-7B-Instruct (Yang et al., 2024) as our backbone. For RL training, we set the temperature to 1.0, group size N to 8, discount factor γ to 0.95, weighting coefficients of advantages ω^M and ω^A both to 1.0. We adopt $F_{\text{norm}} = 1$ unless otherwise specified. All RL-based methods are trained for 150 steps. More details are described in Appendix A.2.

Metrics. We report success rate for both benchmarks. For WebShop, we also report the task score provided by the environment.

5.2 Main Results

The overall performance is presented in Table 2. We conclude that: (1) *MetaAct* consistently outperforms the state-of-the-art approaches, achieving success rates of 95.7% on ALFWorld and 79.7% on WebShop, respectively. The improvements of 4.9% and 4.5% over the previous best results highlight the effectiveness of our framework. (2) *MetaAct* with different F_{norm} both outperform their GiGPO counterparts , indicating its robustness to the choice of F_{norm}. (3) All RL-based methods outperform both prompt-based methods and the state-of-the-art closed-source LLMs. It underscores the necessity of post-training LLM agents for long-horizon tasks.

5.3 Ablation Study

To assess the impact of key components, we conduct ablation studies on WebShop by removing individual meta-actions, advantage terms and the

Method	Type	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Closed-Source Model										
GPT-4o	Prompting	75.3	60.8	31.2	56.7	21.6	49.8	48.0	31.8	23.7
Gemini-2.5-Pro	Prompting	92.8	63.3	62.1	69.0	26.6	58.7	60.3	42.5	35.9
Qwen2.5-7B-Instruct										
Zero-shot	Prompting	33.4	21.6	19.3	6.9	2.8	3.2	14.8	26.4	7.8
ReAct	Prompting	48.5	35.4	34.3	13.2	18.2	17.6	31.2	46.2	19.5
Reflexion	Prompting	62.0	41.6	44.9	30.9	36.3	23.8	42.7	58.1	28.8
PPO (with critic)	RL Training	92.3	64.0	92.5	89.5	80.3	68.8	80.4	81.4	68.7
RLOO	RL Training	87.6	78.2	87.3	81.3	71.9	48.9	75.5	80.3	65.7
GRPO	RL Training	90.8	66.1	89.3	74.7	72.5	64.7	77.6	79.3	66.1
GiGPO$_{\text{w/ std}}$	RL Training	97.7	82.7	98.8	83.7	89.3	79.2	90.8	84.4	72.8
GiGPO$_{\text{w/o std}}$	RL Training	91.8	88.6	95.9	90.2	86.5	85.2	90.2	86.2	75.2
MetaAct$_{\textit{w/ std}}$	RL Training	**100.0**	**92.3**	**100.0**	**100.0**	**92.0**	83.3	95.0	85.8	74.2
MetaAct$_{\textit{w/o std}}$	RL Training	**100.0**	**92.3**	**100.0**	**100.0**	88.0	**91.7**	**95.7**	**88.2**	**79.7**

Table 2: Overall performance on ALFWorld and WebShop. For ALFWorld, we report success rates (%) for sub-tasks and the overall (All) performance. For WebShop, we report the score and success rate (%). "Succ." denotes the success rate. "w/ std" denotes $F_{\text{norm}} = \text{std}$, and "w/o std" denotes $F_{\text{norm}} = 1$. The best results are **bold**, and the second best results are underlined.

Method	Score	Succ.
MetaAct	88.2	**79.7**
w/o FOCUS	**88.8**	72.7
w/o GUESS	88.0	70.3
w/o RESUME	87.6	74.2
w/o UNDO	86.4	70.3
w/o A^E	88.4	73.4
w/o A^S	82.1	64.1
w/o γ $\left(\text{i.e.,} R(\boldsymbol{m}_t^{(i)}) = R(\boldsymbol{a}_t^{(i)})\right)$	87.4	71.1

Table 3: The results of ablation study on WebShop.

intra-step discount factor γ formulated in equation (5). The results are shown in Table 3.

We conclude that: (1) Removing any meta-action causes substantial performance drop , suggesting that all meta-actions are necessary. Notably, omitting FOCUS yields a slightly higher task score but a lower success rate. We suppose that the reason is that FOCUS helps the agent shift from exploration to exploitation. Without it, the agent keeps exploring without leveraging valuable information, resulting in higher progress rewards without completing the task. (2) Removing either advantage term decreases the performance, proving that training LLM agents with meta-actions requires both the trajectory-level signals for global supervision and the step-level signals for intermediate guidance. (3) Eliminating γ between meta-action and action under the same step leads to performance drop, suggesting the effectiveness of decoupling strategic intent from low-level execution.

Method	In-Distribution	Out-of-Distribution
GiGPO	90.8	86.6
MetaAct	**95.7**	**94.0**

Table 4: The success rates (%) on in-distribution and out-of-distribution tasks in ALFWorld.

5.4 Analysis

***MetaAct* generalizes better on unseen tasks.** To assess whether *MetaAct* achieves stronger generalization through smarter exploration, we compare the success rates of *MetaAct* and GiGPO (Feng et al., 2025) on the out-of-distribution test set of ALFWorld in Table 4. Our method outperforms GiGPO with a 7.4% higher success rate on unseen tasks, indicating superior generalization. Notably, *MetaAct* yields only a 1.7% performance drop when transitioning from seen to unseen tasks, while GiGPO shows a 4.2% decrease. This further demonstrates that *MetaAct* learns an exploration strategy that is decoupled from specific tasks, enabling it to generalize effectively to unseen tasks.

***MetaAct* boosts success rates with fewer steps.** In this section, we further investigate how *MetaAct* exhibits smart exploration and reduces unproductive trials. Specifically, we first categorize the tasks in ALFWorld into three difficulty levels: easy, medium, and hard, based on the number of steps GiGPO takes to complete them. Easy tasks are within 5 steps, medium tasks between 5 and 15 steps, and hard tasks require more than 15 steps.

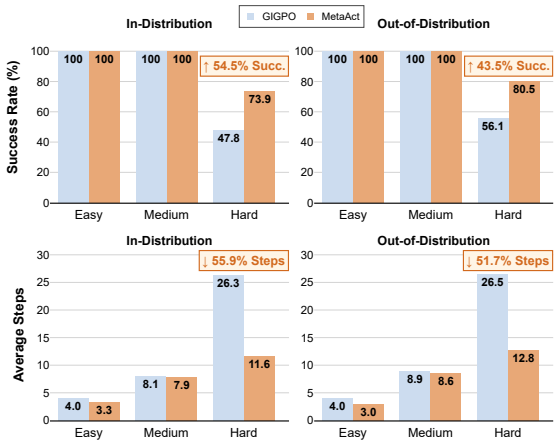


Figure 3: The success rates (Succ.) and average number of steps taken in the tasks in ALFWorld across all difficulty levels. The easy, medium and hard tasks refer to the tasks that are completed within 5 steps, between 5 and 15 steps and more than 15 steps, respectively. A step refers to an interaction with the environment.

Then we compare the success rates of *MetaAct* and GiGPO, along with the average number of steps on the tasks that are successfully completed by both methods. The results are illustrated in Figure 3.

We conclude that: (1) *MetaAct* significantly outperforms GiGPO in success rates on hard tasks under both in-distribution and out-of-distribution settings, with relative improvements of 54.5% and 43.5%. This suggests that our method effectively facilitates smart exploration to solve more challenging tasks. (2) *MetaAct* consistently requires fewer steps to complete tasks across all difficulty levels under both settings. This indicates that our method learns a goal-aware exploration strategy that generalizes well to both unseen tasks and varying difficulties. (3) *MetaAct* substantially reduces the number of steps on hard tasks, yielding relative improvements of 55.9% and 51.7% under both settings. This highlights that our method not only achieves higher success rates on hard tasks but also omits many unnecessary trials. Consequently, the agent navigates towards promising trajectories and avoids repetitive loops during long-horizon tasks.

Although *MetaAct* substantially reduces environmental interactions, it incurs additional computational overhead due to meta-action prediction. To measure this overhead, we compare the average token consumption of *MetaAct* and GiGPO on hard ALFWorld tasks (completed with more than 15 steps), following the setup in Figure 3. Detailed results are reported in Appendix A.3. Despite its higher token consumption, *MetaAct* will reduce end-to-end latency in real-world scenarios, where environmental interactions, such as tool calls and web searches, typically dominate execution time.

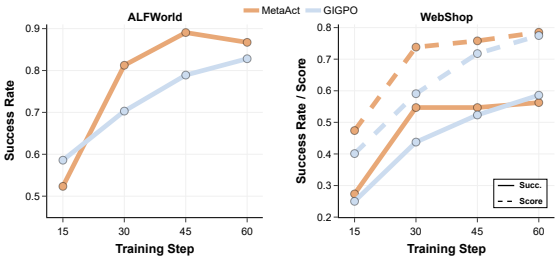


Figure 4: The performance comparison on ALFWorld and WebShop over the first 60 training steps.

	FOCUS	GUESS	RESUME	UNDO
Avg. step index	6.8	4.4	3.2	2.7

Table 5: The average step index at which each meta-action is triggered on in-distribution ALFWorld tasks.

***MetaAct* performs better in fewer training steps.** To investigate whether goal-aware exploration enhanced by *MetaAct* further improves sample efficiency in RL training, we compare the performance of *MetaAct* and GiGPO on the validation sets of ALFWorld and WebShop within the first 60 training steps in Figure 4. We conclude that *MetaAct* consistently outperforms GiGPO with fewer training steps. On ALFWorld, *MetaAct* surpasses a success rate of 80% in 30 training steps, achieving nearly 10% absolute improvement over GiGPO. A similar trend is also shown on WebShop. Although *MetaAct* reaches performance comparable to GiGPO at 60 training steps on WebShop, it still outperforms GiGPO after 150 training steps, as reported in Table 2. These highlight that *MetaAct* substantially improves sample efficiency in RL training, which is critical for training LLM agents in real-world environments where training samples are extremely difficult to acquire.

Analysis of Meta-Action Usage. We provide a comprehensive analysis of meta-action usage on the in-distribution test set of ALFWorld to demonstrate their learned behaviors.

First, we assess the validity of meta-actions and find that 99.7% of predictions produce one of four valid meta-actions, indicating that the agent reliably learns the meta-action space. Second, we report the average step index where each meta-action is trig-

gered in Table 5. It reveals a consistent behavioral pattern of "GUESS → UNDO/RESUME → GUESS → FOCUS", where the agent first gathers information and recovers from mistakes in the early stages and then switches to deterministic execution. Finally, we cluster the actions associated with each meta-action and find that only "FOCUS" is associated with task-completing actions such as *"move"*, *"clean"* and *"heat"*, while the others mostly involve intermediate actions such as *"go"*, *"take"* and *"open"*. These findings provide empirical evidence that the learned meta-actions exhibit meaningful behaviors.

Analysis of Hyperparameters. To prove the robustness of *MetaAct* to hyperparameters, we compare its performance against GiGPO across varying group sizes. More details are in Appendix A.4.

Case Study. We present qualitative examples to intuitively illustrate the goal-aware exploration of *MetaAct* in Appendix A.5.

6 Conclusion

In this paper, we propose *MetaAct*, an RL framework that incorporates four meta-actions (FOCUS, GUESS, RESUME, UNDO) to explicitly guide LLM agents towards promising trajectories through goal-aware exploration. It consists of an LLM agent augmented with meta-actions and a group-based RL algorithm to fine-tune the agent with two-level advantages. Specifically, the agent identifies one of four meta-actions to constrain the directions of its subsequent actions. More importantly, we fine-tune the agent with group-based RL using episode-level advantages that capture trajectory quality and step-level advantages that measure the contribution of each meta-action and action to task completion. Experimental results demonstrate that our method consistently outperforms state-of-the-art approaches, achieving higher success rates with fewer steps.

Limitations

Despite outperforming existing baselines, this work has several limitations. First, our framework adopts Qwen2.5-7B-Instruct as the backbone of the LLM agent with meta-actions. Future work could further reveal its potential by investigating other model architectures and scales. Second, requiring the agent to predict a meta-action prior to task-specific action introduces additional computational overhead, even though it achieves higher success rates with fewer environmental interactions. Future work could develop more advanced reasoning paradigms that integrate goal-aware exploration into the model's reasoning process, eliminating the need for explicit meta-action prediction. Finally, the step-level advantage in RL training of our framework relies on the anchor state grouping strategy that discovers identical states across trajectories, which is challenging in complex and noisy real-world environments. Future work could investigate more robust state grouping mechanisms to enable accurate advantage estimation under such conditions.

Ethical Considerations

In this paper, we aim to improve the goal-awareness of LLM agents and facilitate smart exploration during long-horizon tasks. Regarding transparency and fairness, all data sources used in our evaluation are publicly available, contain no personally identifiable information, and introduce no offensive or biased content at the data level. While there are many potential societal consequences of our work, none we feel must be specifically highlighted here. Furthermore, this research does not raise immediate ethical considerations regarding potential risks and does not involve human subjects.

Acknowledgements

We acknowledge the use of Claude Code (https://github.com/anthropics/claude-code) as a coding and writing assistant in this work. Specifically, it helped generate the code for visualizations in Figure 3 and Figure 4. Additionally, it helped polish the initial draft of the paper. However, all core ideas, framework design, and experimental design of this work were independently conceived and developed by the authors. This work was supported by MYbank, Ant Group.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the*

62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12248–12267.

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

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. 2025. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *The Thirteenth International Conference on Learning Representations*.

Mingyue Cheng, Jie Ouyang, Shuo Yu, Ruiran Yan, Yucong Luo, Zirui Liu, Daoyu Wang, Qi Liu, and Enhong Chen. 2025. Agent-r1: Training powerful llm agents with end-to-end reinforcement learning. *arXiv preprint arXiv:2511.14460*.

Lutfi Eren Erdogan, Hiroki Furuta, Sehoon Kim, Nicholas Lee, Suhong Moon, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Forty-second International Conference on Machine Learning*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Stan Franklin and Art Graesser. 1996. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *International workshop on agent theories, architectures, and languages*, pages 21–35. Springer.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, and 1 others. 2026. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Yulun Jiang, Liangze Jiang, Damien Teney, Michael Moor, and Maria Brbic. 2026. Meta-rl induces exploration in language agents. In *International Conference on Learning Representations*, volume 2026, pages 3279–3303.

Vivek Kalyan and Martin Andrews. 2025. Reinforcement learning for long-horizon multi-turn search agents. In *First Workshop on Multi-Turn Interactions in Large Language Models*.

Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. 2025a. Agent-oriented planning in multi-agent systems. In *International Conference on Learning Representations*, volume 2025, pages 19495–19517.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025b. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Y Zou, and Pan Lu. 2026. In-the-flow agentic system optimization for effective planning and tool use. In *International Conference on Learning Representations*, volume 2026, pages 50524–50570.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Jianbin Jiao, and Junge Zhang. 2026. Agentic reinforcement learning with implicit step rewards. In *International Conference on Learning Representations*, volume 2026, pages 129271–129291.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*.

Renze Lou, Hanzi Xu, Sijia Wang, Jiangshu Du, Ryo Kamoi, Xiaoxin Lu, Jian Xie, Yuxuan Sun, Yusen Zhang, Jihyun Janice Ahn, Hongchao Fang, Zhuoyang Zou, Wenchao Ma, Xi Li, Kai Zhang, Congying Xia, Lifu Huang, and Wenpeng Yin. 2025. AAAR-1.0: Assessing AI's potential to assist research. In *Forty-second International Conference on Machine Learning*.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. 2026. HiPER: Hierarchical plan–execute RL for multi-turn LLM agents. In *Forty-third International Conference on Machine Learning*.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, and 1 others. 2025. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*, volume 2025, pages 41488–41505.

Stuart J Russell and Peter Norvig. 1995. Artificial intelligence: a modern approach.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Mark Stefik. 1981. Planning and meta-planning (molgen: Part 2). *Artificial Intelligence*, 16(2):141–169.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT press.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Ruiyi Wang and Prithviraj Ammanabrolu. 2025. A practitioner's guide to multi-turn agentic reinforcement learning. *arXiv preprint arXiv:2510.01132*.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida Wang. 2026. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *Advances in Neural Information Processing Systems*, 38:78500–78525.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, and 1 others. 2025. Agentgym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27914–27961.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Jiaqi Liu, Honglin Guo, yajie yang, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, and 5 others. 2026. Agentgym-RL: An open-source framework to train LLM agents for long-horizon decision making via multi-turn RL. In *The Fourteenth International Conference on Learning Representations*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2026. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604.

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, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Xiaokang Ye, Jiawei Ren, Yan Zhuang, Xuhong He, Yiming Liang, Yiqing Yang, Mrinaal Dogra, Xianrui Zhong, Eric Liu, Kevin Benavente, Rajiv Mandya Nagaraju, Dhruv Vivek Sharma, Ziqiao Ma, Tianmin Shu, Zhiting Hu, and Lianhui Qin. 2025. Simworld: An open-ended simulator for agents in physical and social worlds. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, Yang Katie Zhao, and Mingyi Hong. 2025. Reinforcing multi-turn reasoning in LLM agents via turn-level credit assignment. In *ICML 2025 Workshop on Computer Use Agents*.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhong-Zhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita Velez, Yue Liao, Hongru WANG, and 6 others. 2026. The landscape of agentic reinforcement learning for LLMs: A survey. *Transactions on Machine Learning Research*. Survey Certification.

Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, and 1 others. 2025. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13643–13658.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. GPT-4v(ision) is a generalist web agent, if grounded. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2024a. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pages 15585–15606.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024b. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Forty-first International Conference on Machine Learning*.

A Experimental Details

A.1 The Statistics of Benchmarks

We conduct experiments on ALFWorld and WebShop and report their statistics in Table 6.

A.2 Training Details

Hyperparameters for ALFWorld. All experiments are conducted under identical hyperparameter settings. The maximum prompt and response lengths are set to 4096 and 1024 tokens, respectively. Each episode is limited to a maximum of 50 steps, with each step comprising one meta-action and one action. We set the actor's learning rate to $1e$-6 and the critic's to $1e$-5 (only for PPO). We employ a rule-based reward function that grants 10 for task success, 0 for failure, and a penalty of -0.5 for invalid actions. All group-based RL methods utilize a group size of 8 and sample 16 groups per rollout, totaling $16 \times 8 = 128$ parallel environments. During training rollouts, the sampling temperature is set to 1.0. For validation, we use a temperature of 0.7, with top-p sampling at 0.8 and top-k at 20. The mini-batch size is 256. For GiGPO, the weighting coefficient ω is set to 1.0. For *MetaAct*, ω^M and ω^A are both set to 1.0.

Hyperparameters for WebShop. All experiments are conducted under identical hyperparameter settings. The maximum prompt and response lengths are set to 8192 and 1024 tokens, respectively. Each episode is limited to a maximum of 15 steps, with each step comprising one meta-action and one action. We set the actor's learning rate to $1e$-6 and the critic's to $1e$-5 (only for PPO). We employ a rule-based reward function that grants 10 for task success, 0 for failure, and a penalty of -0.5 for invalid actions. All group-based RL methods utilize a group size of 8 and sample 16 groups per rollout, totaling $16 \times 8 = 128$ parallel environments. During training rollouts, the sampling temperature is set to 1.0. For validation, we use a temperature of 0.7, with top-p sampling at 0.8 and top-k at 20. The mini-batch size is 256. For GiGPO, the weighting coefficient ω is set to 1.0. For *MetaAct*, ω^M and ω^A are both set to 1.0.

A.3 Analysis of Computational Overhead

As shown in Figure 3, *MetaAct* substantially reduces environmental interactions, but incurs additional computational overhead from meta-action prediction. To measure this overhead, we compare the average token consumption of *MetaAct*

Benchmark	Train	Test (In-Distribution)	Test (Out-of-Distribution)
ALFWorld	3553	140	134
WebShop	1824	200	-

Table 6: The statistics of ALFWorld and WebShop.

Method	In-Distribution	Out-of-Distribution
All hard tasks		
GiGPO	4.7	4.4
MetaAct	14.8	12.6
Jointly solved hard tasks		
GiGPO	2.9	2.9
MetaAct	4.8	6.6

Table 7: Average token consumption ($\times 10^3$) per task on hard ALFWorld tasks, defined as tasks requiring more than 15 steps. Jointly solved hard tasks refer to tasks successfully completed by both methods.

Group Size N	Method	Score	Succ.
$N = 8$	GiGPO	86.2	75.2
	MetaAct	**88.2**	**79.7**
$N = 16$	GiGPO	84.5	74.2
	MetaAct	**88.4**	**78.1**

Table 8: The results of different group sizes N on WebShop. "Succ." denotes the success rate.

and GiGPO on hard ALFWorld tasks (completed with more than 15 steps), following the setup in Figure 3. The results are reported in Table 7.

As shown in Figure 3 and Table 7, despite consuming more tokens, *MetaAct* achieves substantially higher success rates on hard tasks: 73.9% versus 47.8% for GiGPO under the in-distribution setting, and 80.5% versus 56.1% under the out-of-distribution setting. This additional token cost is accompanied by substantial gains in task success. Moreover, on hard tasks successfully completed by both methods, *MetaAct* substantially reduces environmental interactions, from 26.3 steps to 11.6 steps (-55.9%) under the in-distribution setting and from 26.5 steps to 12.8 steps (-51.7%) under the out-of-distribution setting. Therefore, despite its higher token consumption, *MetaAct* will reduce end-to-end latency in real-world scenarios where environmental interactions such as tool calls and web searches typically dominate execution time.

A.4 Analysis of Group Size

To demonstrate that our method is robust to the choice of key hyperparameters, we conduct experiments on group size N by comparing the performance of GiGPO and *MetaAct* on WebShop with $N = 16$ (N = 8 by default). The results are shown in Table 8. It proves that *MetaAct* consistently outperforms GiGPO and exhibits stable performance across different group sizes.

A.5 Case Study

To illustrate how *MetaAct* achieves goal-aware exploration in a more intuitive way, we present a case study on the out-of-distribution test set of ALFWorld to compare how *MetaAct* and GiGPO perform on unseen hard tasks. The results are presented in Figure 5 and 6. *MetaAct* solves the tasks much more efficiently than GiGPO, reducing the number of steps from 44 to 13 on the clean-and-place task and from 36 to 8 on the heat-and-place task. These results demonstrate that *MetaAct* achieves more efficient exploration and switches to exploitation once sufficient information is available, while GiGPO incurs many unproductive trials.

B Prompts

We provide examples of prompts for each benchmark. Our method utilizes a two-stage prompting mechanism to guide the agent towards promising trajectories through meta-actions. There are two prompts: (1) the Meta-Action Prompt, where the agent identifies the high-level meta-action (i.e., `FOCUS`, `RESUME`, `GUESS`, `UNDO`) to direct subsequent task-specific action; (2) the Action Prompt, where the agent predicts the task-specific action strictly aligned with the determined meta-action.

The prompts contain variables (e.g., `{current_trajectory}`) and task-specific placeholders (e.g., `{init_observation}`). These are omitted in the descriptions below for clarity but are replaced with actual context during runtime.

In the responses, the agent is instructed to provide step-by-step reasoning followed by the final decision. We use the `<meta_action></meta_action>` block to indicate the meta-action and the `<action></action>` block to indicate the final task-specific action.

Case Study: Clean and Place Task in ALFWorld

Initial State & Task

Initial State: You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 3, a countertop 2, a countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Task: **clean some pan and put it in countertop.**

***MetaAct* (Ours) → Success in 13 steps**

(GUESS, go to countertop 1)
(GUESS, go to countertop 2)
(GUESS, go to shelf 1)
(GUESS, go to countertop 3)
(GUESS, go to garbagecan 1)
(GUESS, go to sinkbasin 1)
(GUESS, go to stoveburner 1)
(GUESS, go to stoveburner 2)
(FOCUS, take pan 1 from stoveburner 2)
(FOCUS, go to sinkbasin 1)
(FOCUS, clean pan 1 with sinkbasin 1)
(FOCUS, go to countertop 1)
(FOCUS, move pan 1 to countertop 1)

GiGPO (Baseline) → Success in 44 steps

go to countertop 1
go to countertop 3
go to countertop 2
go to fridge 1
go to countertop 1
go to countertop 3
go to stoveburner 1
go to countertop 1
go to sinkbasin 1
go to countertop 1
go to countertop 3
go to sinkbasin 1
go to countertop 3
go to stoveburner 1
go to fridge 1
go to countertop 2
go to sinkbasin 1
go to sinkbasin 1

... (20 unproductive steps omitted)

go to stoveburner 2
take pan 1 from stoveburner 2
go to sinkbasin 1
clean pan 1 with sinkbasin 1
go to countertop 1
move pan 1 to countertop 1

Figure 5: A comparative case study of Clean and Place task in ALFWorld. *MetaAct* completes the task in 13 steps by strategically switching between exploration (GUESS) and execution (FOCUS), while GiGPO takes 44 steps.

Case Study: Heat and Place Task in ALFWorld

Initial State & Task

Initial State: You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 3, a countertop 2, a countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Task: **put a hot cup in cabinet.**

***MetaAct* (Ours) → Success in 8 steps**

(GUESS, go to countertop 1)
(GUESS, go to fridge 1)
(GUESS, open fridge 1)
(FOCUS, take cup 3 from fridge 1)
(FOCUS, go to microwave 1)
(FOCUS, heat cup 3 with microwave 1)
(FOCUS, go to cabinet 1)
(FOCUS, move cup 3 to cabinet 1)

GiGPO (Baseline) → Success in 36 steps

go to countertop 1
go to countertop 2
go to fridge 1
go to sinkbasin 1
go to countertop 3
go to sinkbasin 1
go to countertop 3
go to sinkbasin 1
go to countertop 1
go to countertop 3
go to sinkbasin 1
go to countertop 1
go to countertop 2
go to sinkbasin 1

... (15 unproductive steps omitted)

go to fridge 1
open fridge 1
take cup 3 from fridge 1
go to microwave 1
heat cup 3 with microwave 1
go to cabinet 1
move cup 3 to cabinet 1

Figure 6: A comparative case study of Heat and Place task in ALFWorld. *MetaAct* completes the task in 8 steps by strategically switching between exploration (GUESS) and execution (FOCUS), while GiGPO takes 36 steps.

B.1 ALFWorld

The meta-action prompt and action prompt of ALFWorld are shown in Figure 7 and Figure 8.

B.2 WebShop

The meta-action prompt and action prompt of WebShop are shown in Figure 9 and Figure 10.

ALFWorld Meta-Action Prompt

You are an expert agent and meta-planner operating in the ALFRED Embodied Environment. Before taking a physical action, you must determine your current strategic state (Meta-Action).
`{init_observation}{current_trajectory}`

As a meta-planner, you must first select one of the following 4 Meta Actions for the current situation to guide your next move:

1. **FOCUS**: Use this when the next step is logically certain or mandatory based on current constraints. You are following a clear, deterministic path without needing to gamble.
2. **RESUME**: Use this when a previously suspended sub-goal or task (which was "waiting" for certain conditions) can now be continued because you have acquired new information or reached the right location.
3. **GUESS**: Use this when the logic is insufficient to determine a unique "correct" action. You must make a heuristic choice to explore the environment or try a plausible path.
4. **UNDO**: Use this when you detect a conflict, a mistake, or a dead-end. You need to backtrack, correct a previous "GUESS" or switch to a different high-level approach.

Now it's your turn to determine the Meta-Action.

- Your response should start with step-by-step reasoning. Analyze the current situation, your past failures (if any), and whether your next move is a certainty (FOCUS), a continuation (RESUME), an exploration (GUESS), or a correction (UNDO).
- Once you've finished your reasoning, you must output exactly one Meta-Action from the four options above within `<meta_action> </meta_action>` tags.

Figure 7: The meta-action prompt of ALFWorld.

ALFWorld Action Prompt

You are an expert agent operating in the ALFRED Embodied Environment.
You have already decided on a strategic direction (Meta-Action) for this step. Your task now is to execute a physical action that strictly aligns with that strategy.
`{init_observation}{current_trajectory}`

Based on the Meta-Action above, follow these guidelines to select your action:
- If **FOCUS**: Choose the most logically certain action. Avoid any exploratory or risky moves.
- If **RESUME**: Identify a previously paused task or a goal you were waiting to complete, and take the next step toward it.
- If **GUESS**: Since no certain path exists, select a plausible action to explore or gather more information.
- If **UNDO**: You have detected a mistake or a dead-end. Select an action that backtracks or tries an alternative path different from your previous failed attempts.

Your admissible actions of the current situation are:
[`{admissible_actions}`]

Now it's your turn to take an action.

- Your response should first be step-by-step reasoning about the current situation.
- Once you've finished your reasoning, you should choose an admissible action for current step and present it within `<action> </action>` tags.

Figure 8: The action prompt of ALFWorld.

WebShop Meta-Action Prompt

You are an expert autonomous agent and meta-planner operating in the WebShop e-commerce environment.
Before taking a physical action, you must determine your current strategic state (Meta-Action).
Your task is to: `{task_description}`.`{current_trajectory}`

As a meta-planner, you must first select one of the following 4 Meta Actions for the current situation to guide your next move:

1. **FOCUS**: Use this when the next step is logically certain or mandatory based on current constraints. You are following a clear, deterministic path without needing to gamble.
2. **RESUME**: Use this when a previously suspended sub-goal or task (which was "waiting" for certain conditions) can now be continued because you have acquired new information or reached the right location.
3. **GUESS**: Use this when the logic is insufficient to determine a unique "correct" action. You must make a heuristic choice to explore the environment or try a plausible path.
4. **UNDO**: Use this when you detect a conflict, a mistake, or a dead-end. You need to backtrack, correct a previous "GUESS" or switch to a different high-level approach.

Now it's your turn to determine the Meta-Action.

- Your response should start with step-by-step reasoning. Analyze the current situation, your past failures (if any), and whether your next move is a certainty (FOCUS), a continuation (RESUME), an exploration (GUESS), or a correction (UNDO).
- Once you've finished your reasoning, you must output exactly one Meta-Action from the four options above within `<meta_action> </meta_action>` tags.

Figure 9: The meta-action prompt of WebShop.

WebShop Action Prompt

You are an expert autonomous agent operating in the WebShop e-commerce environment.
You have already decided on a strategic direction (Meta-Action) for this step. Your task now is to execute a physical action that strictly aligns with that strategy.
Your task is to: `{task_description}`.`{current_trajectory}`

Based on the Meta-Action above, follow these guidelines to select your action:
- If **FOCUS**: Choose the most logically certain action. Avoid any exploratory or risky moves.
- If **RESUME**: Identify a previously paused task or a goal you were waiting to complete, and take the next step toward it.
- If **GUESS**: Since no certain path exists, select a plausible action to explore or gather more information.
- If **UNDO**: You have detected a mistake or a dead-end. Select an action that backtracks or tries an alternative path different from your previous failed attempts.

Your admissible actions of the current situation are:
[`{admissible_actions}`]

Now it's your turn to take one action for the current step.

- Your response should first be step-by-step reasoning about the current situation, then think carefully which admissible action best advances the shopping goal.
- Once you've finished your reasoning, you should choose an admissible action for current step and present it within `<action>` `</action>` tags.

Figure 10: The action prompt of WebShop.