SKILL IN THE LOOP: EXPLICIT SKILL DECLARATION AND ATTRIBUTION FOR AGENT TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agents increasingly rely on reusable skills distilled from historical experience to solve long-horizon decision-making tasks. However, existing methods primarily provide skills as contextual information, leaving the agent's skill selection implicit and offering limited mechanisms to assess and improve how skills are used. We propose **S**kill-**C**onditioned **P**olicy **O**ptimization (**SCPO**), a closed-loop framework for explicit skill use and continual skill evolution. Before each action, the agent explicitly declares the skill guiding its decision. A retrospective mechanism then evaluates the effectiveness of skill usage over multi-step interactions and provides fine-grained feedback, which jointly optimizes the policy via GRPO and updates the skill bank based on observed utility. Experiments on ALFWorld and WebShop show that SCPO achieves state-of-the-art performance and consistently improves both skill utilization and skill-bank quality.

1 INTRODUCTION

Large Language Model (LLM) agents are increasingly applied to long-horizon decision-making tasks Yao et al. (2022b); Shi et al. (2025); He et al. (2025), including embodied interaction Shridhar et al. (2020), scientific research Schmidgall et al. (2025), and web navigation Yao et al. (2022a). Unlike short-horizon tasks, these scenarios require agents to execute sequences of interdependent actions, where the consequence of an individual decision may only be revealed after completing the entire trajectory. Such delayed feedback makes it challenging to identify the contribution of each action and improve decision-making through immediate supervision. Meanwhile, long-horizon tasks often exhibit recurring objectives and shared action patterns across different instances, indicating that knowledge acquired from previous interactions can help reduce redundant exploration and guide future decisions. Therefore, effectively leveraging experience is crucial for enabling agents to learn from past interactions and continuously improve their performance.

A natural way to leverage such experience is to distill successful trajectories into reusable skills Ni et al. (2026). Building on this idea, recent methods further study how these skills can be organized, evolved, and utilized within a skill bank. SkillRL Xia et al. (2026) organizes experience into hierarchical skills and evolves the skill bank, D2Skill Tu et al. (2026) adopts dual-granularity skill organization, and Skill1 Shi et al. (2026) models skill selection, utilization, and distillation as a lifecycle. These skills summarize past experience into high-level decision rules (e.g., *where to search first*, *when to use an object*) and provide reusable guidance for future interactions. The central question, however, is not only how to retrieve relevant skills, but also how an agent actually uses them when making decisions. Existing methods generally provide several potentially relevant skills as contextual information and append them to the agent's input, leaving the agent to implicitly determine which, if any, should guide its next decision Lu et al. (2026); Zheng et al. (2026); Wang et al. (2026). This implicit interface makes skill use difficult to observe and failure difficult to diagnose.

Consider the example in Figure 1. At the current observation, the agent is holding an uncleaned apple, with the ultimate goal of placing it on the dining table. Two retrieved skills suggest different strategies: *go to the sink first* and *move directly to the destination*. If the agent outputs "go to the dining table", the failure is apparent, but its underlying cause remains unclear without an explicit skill declaration. It may have ignored the appropriate skill, followed an unsuitable one, or failed to use any skill at all. These failure modes require different corrections but cannot be distinguished

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

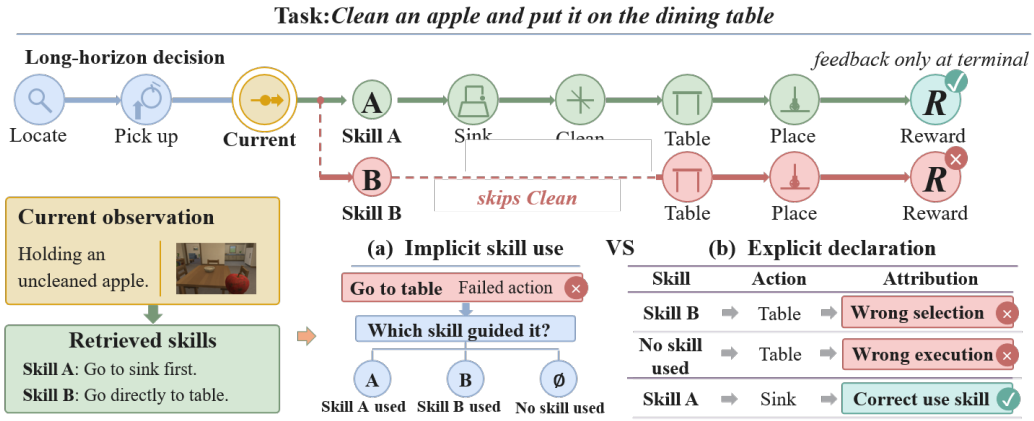


Figure 1: Motivating example of implicit skill usage versus explicit skill declaration.

when skills are provided only as contextual information. The skills available to the agent are visible, whereas the skill actually guiding its decision remains hidden.

Explicit skill declaration resolves this ambiguity by making skill selection an observable decision-making signal. Before generating reasoning and actions, the agent declares the skill it intends to apply. In the above example, declaring *move directly to the destination* reveals an inappropriate strategy for the current state, while declaring *go to the sink first* indicates an appropriate decision aligned with the task requirements. However, declaration alone does not guarantee effective skill use: an agent may select the wrong skill or select the right skill but fail to follow its principle. For instance, declaring *go to the sink first* while moving directly to the dining table reflects a skill execution failure rather than a selection failure. Therefore, reliable skill use requires step-level feedback that evaluates both whether a declared skill matches the current state and whether subsequent reasoning and actions follow that skill. Such feedback enables precise error attribution, reinforces effective skill usage, and provides a foundation for improving the agent policy [1].

Motivated by these observations, we propose **S**kill-**C**onditioned **P**olicy **O**ptimization (**SCPO**), a closed-loop framework that transforms skills from contextual information into explicit decision-making signals. Specifically, at each interaction step, SCPO requires the agent to declare the retrieved skill before generating its reasoning and action, making skill selection an observable policy output. After the interaction, a step-level retrospective attribution mechanism evaluates whether each declared skill is appropriate for the current state and whether subsequent behaviors are consistent with it. These evaluations provide fine-grained learning signals that jointly optimize the agent policy via GRPO Shao et al. (2024) and evolve the skill bank based on observed utility. SCPO thereby enables skill selection and usage to become observable, evaluable, and continually improvable components of agent decision-making.

We evaluate SCPO on two benchmarks for LLM agents, ALFWorld Shridhar et al. (2020) and WebShop Yao et al. (2022a). SCPO achieves state-of-the-art performance on both benchmarks. Further analyses show that SCPO progressively improves skill utilization and declaration accuracy during training, leading to more reliable skill use in long-horizon interactions. The contributions of this work are three-fold: (1) We identify that existing skill-based agents leave skill selection implicit, limiting the interpretability and improvement of skill usage. SCPO addresses this limitation by introducing explicit skill declaration and step-level feedback. (2) We propose a closed-loop optimization framework that retrospectively evaluates skill usage and jointly improves the agent policy and skill bank from the resulting feedback. (3) Extensive experiments on ALFWorld and WebShop demonstrate the effectiveness of SCPO across different model scales, with component analyses validating the contribution of each design choice.

[1] See Appendix D for another empirical motivating analysis.

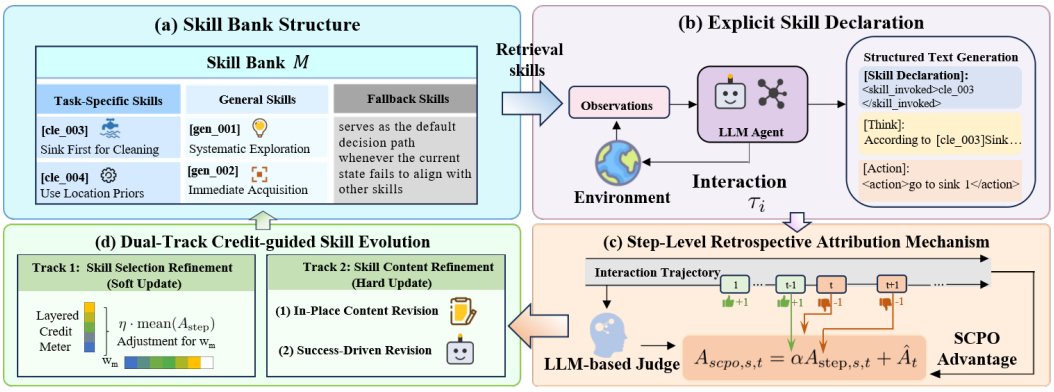


Figure 2: The overall framework of SCPO.

2 Related Work

Skill-Augmented Agents. Early experience-based agents leveraged raw historical experience for future decision-making Zhao et al. (2024); Wang et al. (2023), while later work distilled successful experience into reusable skills Ni et al. (2026). Subsequent studies scaled skill banks through better organization, evaluation, and continual evolution Xu & Yan (2026); Jiang et al. (2026b); Li et al. (2026a); Liang et al. (2026); Li et al. (2026b); Zhang et al. (2026); Yang et al. (2026); Jiang et al. (2026a). Recent methods further integrate skills with RL. SkillRL Xia et al. (2026) evolves hierarchical skills, D2Skill Tu et al. (2026) organizes skills at dual granularities, and Skill1 Shi et al. (2026) models skill selection and utilization as a lifecycle. However, skills remain largely contextual guidance, leaving step-level skill selection implicit.

Credit Assignment for Agents. Credit assignment is challenging in long-horizon tasks, as terminal rewards provide limited supervision for intermediate decisions Williams (1992); Arjona-Medina et al. (2019). Existing methods provide trajectory-level Shao et al. (2024), step-level Cobbe et al. (2021); Lightman et al. (2023), or retrospective Zhang et al. (2025) feedback, but primarily evaluate reasoning or behavior rather than the selected skill. Thus, skill-level credit assignment remains largely unexplored Xia et al. (2026); Tu et al. (2026); Ni et al. (2026). SCPO addresses this gap by explicitly declaring the selected skill at each step, assigning it retrospective credit, and using these signals to jointly optimize the policy and evolve the skill bank.

3 Method

We first introduce the long-horizon decision-making task and then present SCPO's key components.

3.1 Problem Formulation

We consider a goal-oriented decision-making task, where an agent interacts with an environment through sequential observations and actions. The task is defined by an observation space $\mathcal{O}$, an action space $\mathcal{A}$, and a goal space $\mathcal{G}$. Given a goal $g \in \mathcal{G}$, the agent generates a trajectory $\tau = \{(o_1, a_1), \ldots, (o_T, a_T)\}$ through environment interaction, where $o_t \in \mathcal{O}$ and $a_t \in \mathcal{A}$ denote the observation and action at step t.

To leverage historical experience, the agent is equipped with a non-parametric skill bank $\mathcal{M}$, storing distilled strategic principles (high-level experience) extracted from previous interactions. Given a goal g, a skill set $\mathcal{S}_t \subseteq \mathcal{M}$ is first retrieved from the skill bank. At each step t, the agent selects a skill m_t from $\mathcal{S}_t$ based on the current context, and makes decisions using the augmented context $x_t = (o_{\leq t}, g, m_t)$. The agent follows a policy $\pi(a_t|x_t)$ to generate actions and complete the task.

However, merely conditioning the policy on retrieved skills does not explicitly identify which skill is being followed at each step, leaving the relationship between skills and decisions unobservable. When an incorrect action occurs, it is difficult to determine whether the failure stems from select-

ing an inappropriate skill or incorrectly executing an appropriate one, resulting in ambiguous credit assignment for policy optimization. To address this challenge, we propose SCPO, which explicitly models skill usage through skill declaration and retrospective attribution, and incorporates the resulting skill-aware credit signals into policy optimization.

3.2 Framework Overview

Figure 2 illustrates the overall pipeline of SCPO, which consists of four components: (1) skill bank initialization, (2) explicit skill declaration, (3) step-level retrospective attribution, and (4) dual-track credit-guided evolution. Given the goal of a task (e.g., *Clean an apple and put it on the dining table*), SCPO first retrieves a set of candidate skills relevant to the task from the skill bank. At each time step, the policy explicitly declares the intended skill before generating the action, making its decision rationale observable. After execution, SCPO retrospectively attributes each skill usage to distinguish between incorrect skill selection, incorrect execution of the declared skill, and appropriate skill usage. Then, these attribution signals are integrated with trajectory-level rewards to optimize the policy under the GRPO framework. Finally, a dual-track credit-guided evolution is proposed to improve the skill bank. Together, these components establish a closed feedback loop in which the policy produces interpretable skill decisions, retrospective attribution evaluates their effectiveness, and the resulting credit signals continuously improve the skill bank for future interactions. Algorithm 1 summarizes the overall procedure of SCPO.

Algorithm 1 Skill-Conditioned Policy Optimization (SCPO)

Require: Policy π_θ, reference policy π_{ref}, skill bank $\mathcal{M}$, $G, K, \alpha, \beta, \epsilon, \lambda, \eta, N_{\text{soft}}, N_{\text{hard}}, \tau_{\text{rev}}, \tau_{\text{gen}}$
▷ *Skill Bank Structure and Initialization (Sec. 3.3)*
Initialize $\mathcal{M}$ with task-specific/general skills and fallback skill m_{fb}; set $w_m \leftarrow 0$
for training iteration $u = 1, 2, \ldots$ **do**
  Sample a batch of tasks $\mathcal{B} \sim \mathcal{G}$
  for each task $g \in \mathcal{B}$ and rollout $i = 1, \ldots, G$ **do**
    Initialize trajectory $\tau_i \leftarrow \emptyset$
    Retrieve skill set $\mathcal{S}_{i,t}$ with top-K skills
    for step $t = 1, \ldots, T_i$ **do**
    ▷ *Explicit Skill Declaration (Sec. 3.4)*
      Generate $y_{i,t} = [m_{i,t}; \text{think}_{i,t}; a_{i,t}]$, where $m_{i,t} \in \mathcal{S}_{i,t}$; execute $a_{i,t}$
    end for
    Obtain terminal reward $R(\tau_i)$
  end for
  ▷ *Step-level Retrospective Attribution (Sec. 3.5)*
  Evaluate $A_{\text{raw},i,t}$; compute $A_{\text{step},i,t}$ and $A_{\text{SCPO},i,t}$
  Update θ by maximizing $J_{\text{SCPO}}(\theta)$ in Eq. equation 5
  ▷ *Dual-track Credit-guided Evolution (Sec. 3.6)*
  if $u \bmod N_{\text{soft}} = 0$ **then**
    Update w_m by Eq. equation 7
  end if
  if $u \bmod N_{\text{hard}} = 0$ **then**
    Perform in-place revision and success-driven addition
  end if
end for
Output: Optimized policy π_θ, evolved skill bank $\mathcal{M}$

3.3 Skill Bank

We follow prior works Xia et al. (2026); Lu et al. (2026) to initialize the skill bank $\mathcal{M}$, which consists of three types of skills. Task-specific skills provide specialized knowledge for particular task categories (e.g., *Sink First for Cleaning* for cleaning tasks), general skills provide broadly applicable experience (e.g., *Systematic Exploration*), and a fallback skill serves as a default option when none of the retrieved skills is suitable for the current state, preventing the policy from being forced to select an irrelevant skill [2].

3.4 Explicit Skill Declaration

When retrieved skills are provided only as contextual information, the agent's chosen execution strategy remains implicit. When an incorrect action occurs, it becomes impossible to determine whether the agent selected a wrong skill or failed to execute the intended one. This ambiguity weakens the learning signal available for policy optimization.

To resolve this issue, SCPO introduces Explicit Skill Declaration, requiring the policy to explicitly specify the skill it intends to execute before predicting the next action. At each time step t, the agent selects a specific skill m_t from the candidate skill set $\mathcal{S}_t$ as an explicit decision and generates the following response sequence [3]:

$$y_t = \{m_t; \text{think}_t; a_t\} \tag{1}$$

[2] See Appendix I for the skill bank taxonomy and definitions.

[3] See Appendix I for details on skill retrieval.

Consequently, every executed action is associated with an explicit behavioral intention, making the decision process interpretable and enabling subsequent skill-level reasoning [4].

3.5 STEP-LEVEL RETROSPECTIVE ATTRIBUTION

Explicit skill declaration exposes the policy's intended strategy but does not indicate whether the declared skill is actually appropriate or correctly executed. Therefore, after each action is performed, SCPO retrospectively analyzes the decision outcome to determine how the declared skill contributes to the resulting behavior.

Given the resulting trajectory (including the retrieved skill set, the declared skill sequence, the executed action sequence, etc.), SCPO assigns each declared skill to one of three attribution categories: (1) Wrong Skill Selection, where the declared skill is inappropriate for the current context; (2) Correct Skill but Wrong Execution, where the selected skill is appropriate but the generated action fails to execute it correctly; (3) Appropriate Skill Use, where both skill selection and execution are consistent with the desired behavior.

Unlike previous trajectory-level rewards, these attribution labels provide dense supervision at every decision step, explicitly identifying the source of policy failures and providing targeted feedback for subsequent optimization.

GRPO Backbone. SCPO is implemented on top of GRPO Shao et al. (2024). SCPO redesigns the advantage estimation by injecting skill-aware attribution signals into the policy optimization process.

During policy optimization, GRPO samples a group of G trajectories $\{\tau_i\}_{i=1}^{G}$ for a given task goal g, and derives a normalized group-relative advantage from the corresponding terminal rewards:

$$\widehat{A}_i = \frac{R(\tau_i) - \frac{1}{G}\sum_{j=1}^{G} R(\tau_j)}{\mathrm{std}\left(\{R(\tau_j)\}_{j=1}^{G}\right) + \epsilon_{\mathrm{adv}}}. \tag{2}$$

This conventional trajectory reward measures whether the overall task is successfully completed. This trajectory-level supervision captures global task performance but provides limited information about which intermediate decisions contribute to success or failure.

To provide denser supervision, SCPO converts retrospective attribution into step-level skill credit. Decisions involving incorrect skill selection or incorrect execution receive negative credit, while appropriate skill usage receives positive credit. These skill-aware signals directly supervise both which skill to choose and how to execute the chosen skill.

Specifically, upon completion of the interaction trajectory τ_i, an LLM-based skill judge (Gemini-3-Flash) retrospectively evaluates the skill $m_{i,t}$ declared at each time step t, providing a raw step-level attribution advantage $A_{\mathrm{raw},i,t} \in \{+1, -1\}$, mapping the first two categories to -1 and the third to $+1$ [5]. This signal is then processed into a normalized step-level attribution advantage $A_{\mathrm{step},i,t}$:

$$A_{\mathrm{step},i,t} = \frac{A_{\mathrm{raw},i,t} - \bar{A}_{\mathrm{raw},i}}{\sigma_{\mathcal{B}}} \tag{3}$$

where $\bar{A}_{\mathrm{raw},i}$ is the intra-trajectory mean of $A_{\mathrm{raw},i,t}$ and $\sigma_{\mathcal{B}}$ denotes the standard deviation of the centered attribution signals computed over the training batch $\mathcal{B}$.

Then, the composite SCPO advantage $A_{\mathrm{SCPO},i,t}$ is:

$$A_{\mathrm{SCPO},i,t} = \hat{A}_i + \alpha \cdot A_{\mathrm{step},i,t} \tag{4}$$

where $\hat{A}_i$ is the trajectory-level advantage computed from the terminal reward as defined in Eq. equation 2, and $\alpha > 0$ is a hyperparameter modulating the attribution weight.

[4] See Appendix A for a theoretical analysis of how Explicit Skill Declaration decomposes the policy gradient and mitigates the credit assignment bottleneck of implicit skill declaration.

[5] See Appendix J for details on the skill judge design and Appendix C for validation of its agreement with human judgments.

The final optimization objective of SCPO is:

$$J_{\text{SCPO}}(\theta) = \mathbb{E}\Big[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i}\min\left(r_{i,t}A_{\text{SCPO},i,t}, \text{clip}(r_{i,t}, 1-\epsilon, 1+\epsilon)A_{\text{SCPO},i,t}\right) - \beta D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}})\Big] \tag{5}$$

where $r_{i,t} = \frac{\pi_\theta(y_{i,t}|x_{i,t})}{\pi_{\text{old}}(y_{i,t}|x_{i,t})}$ denotes the step-level importance ratio, and $A_{\text{SCPO},i,t}$ is the final advantage combining trajectory-level and step-level attribution advantages.

3.6 Dual-track Credit-guided Skill Evolution

However, optimizing the policy alone is insufficient. The skill bank remains external to the model parameters, and its retrieval distribution and skill contents do not automatically improve as the policy evolves. Consequently, retrieval may continue to favor semantically relevant but executionally ineffective skills, while outdated or incomplete skills persist in the bank. Thus, SCPO further introduces a *dual-track credit-guided evolution* mechanism that continuously refines the skill bank, which operates on two complementary tracks.

Track 1: Skill Selection Refinement (Soft Update). This track dynamically adjusts the retrieval weights of individual skills according to their historical attribution advantages, allowing the retriever to gradually prioritize skills that consistently lead to successful execution.

For a candidate skill m, the retrieval score $S_{ret}(m)$ is defined as a weighted fusion of semantic relevance and historical performance:

$$S_{ret}(m) = \lambda \cdot \text{sim}(e_g, e_m) + (1-\lambda)\cdot\sigma(w_m) \tag{6}$$

where e_g and e_m represent the embedding vectors of the task goal g and the skill m respectively, while $\text{sim}(\cdot,\cdot)$ denotes the cosine similarity. w_m represents the cumulative credit weight of skill m, and λ is a hyperparameter.

Every N_{soft} training iterations, the system updates w_m based on the average attribution advantage of the skill during that period:

$$w_m \leftarrow w_m + \eta \cdot \bar{A}_{\text{step}}(m) \tag{7}$$

where η denotes the update step size, and $\bar{A}_{\text{step}}(m)$ represents the mean step-level attribution advantage across all invocations of skill m within the current period.

Track 2: Skill Content Refinement (Hard Update). Low-credit skills are revised through LLM-based rewriting and new skills are distilled from successful trajectories to continuously enrich the skill bank. If the window-averaged attribution advantage of a skill falls below the threshold τ_{rev} at each hard-update round N_{hard}, an LLM rewrites and updates the skill while resetting its weight w_m to ensure an unbiased re-evaluation.

Additionally, we incorporate a success-driven addition mechanism to expand the skill bank. If the average success rate for a particular task category falls below τ_{gen}, the system performs a trace-based analysis to distill new strategic principles, which are subsequently formalized as new skill entries and stored in $\mathcal{M}$ [6].

Together, the GRPO-based policy optimization and the dual-track skill bank evolution form a closed feedback loop. While the former improves the policy parameters through dense attribution supervision, the latter continuously adapts the external memory by refining both retrieval quality and skill knowledge, enabling the policy and the skill bank to co-evolve in long-horizon decision making.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate SCPO on ALFWorld Shridhar et al. (2020) and WebShop Yao et al. (2022a), two representative long-horizon benchmarks covering household and online shopping interactions, respectively.

[6] See Appendix J for details on skill revision and expansion.

Table 1: Performance on ALFWorld and WebShop. ALFWorld reports average success rates (%) by task category and overall, while WebShop reports average score and overall success rate (%). Best and second-best results are highlighted in red and blue.

Method	**ALFWorld**							**WebShop**	
	Pick	Clean	Cool	Look	Heat	Pick2	**All**	Score	Success
Closed-source LLMs									
GPT-4o	75.3	31.2	21.6	60.8	56.7	49.8	48.0	31.8	23.7
Gemini-3-Flash	96.4	57.1	96.2	85.7	72.2	95.3	85.2	14.1	16.5
Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3	42.5	35.9
Base Model: Qwen2.5-7B-Instruct									
Origin	28.6	38.5	7.1	33.3	9.1	0.0	18.8	26.4	7.8
GRPO	87.5±2.1	70.0±3.5	65.5±4.2	65.6±6.1	77.1±3.1	79.2±4.5	76.6±1.7	86.2±1.8	72.7±0.6
Mem0+GRPO	88.8±3.3	90.2±4.1	67.5±3.0	61.1±7.9	93.9±4.3	46.8±8.7	78.1±2.2	83.5±2.0	73.9±3.5
SimpleMem+GRPO	94.0±2.8	91.7±5.9	70.6±3.8	74.5±5.4	76.6±3.2	55.6±4.5	79.2±3.8	86.1±1.0	72.4±1.0
D2Skill	91.1±2.3	94.6±3.8	72.6±3.3	75.8±5.8	92.8±5.3	72.8±6.8	84.4±1.9	85.3±1.1	74.5±0.7
SkillRL	92.3±1.6	90.6±4.1	80.0±4.7	88.9±7.9	71.3±8.0	86.3±5.8	87.0±1.9	85.4±2.8	75.5±2.0
SCPO (Ours)	93.0±1.2	95.8±5.9	87.9±6.6	93.3±9.4	97.0±4.3	89.8±1.3	92.7±1.7	90.4±0.8	79.7±1.3
Base Model: Qwen2.5-3B-Instruct									
Origin	21.4	15.4	0.0	33.3	9.1	0.0	10.9	9.7	1.6
GRPO	88.7±4.5	87.4±2.2	67.0±4.5	49.5±2.6	77.1±8.0	45.8±1.5	74.0±2.6	79.1±2.4	57.1±2.2
Mem0+GRPO	90.5±3.4	84.0±3.5	60.5±4.4	73.0±4.5	78.4±4.4	48.1±2.6	74.5±1.0	75.3±1.1	54.7±1.7
SimpleMem+GRPO	90.9±3.6	83.8±4.4	67.8±3.5	70.9±3.9	35.0±2.4	47.0±2.4	70.3±2.2	74.2±0.9	54.2±1.2
D2Skill	90.3±4.9	76.6±5.0	67.5±3.0	53.3±4.7	59.4±3.7	50.3±9.8	69.8±2.7	70.4±2.0	56.0±1.0
SkillRL	95.2±3.4	83.2±4.0	64.7±5.2	71.1±3.1	67.7±1.9	60.1±3.2	75.5±1.3	80.0±1.2	59.6±2.1
SCPO (Ours)	92.2±3.0	91.7±5.9	68.3±6.6	82.1±6.4	81.8±6.4	62.7±4.6	80.7±1.9	81.3±1.4	61.1±0.7

Baselines We compare SCPO with four categories of baselines: (1) closed-source LLMs (GPT-4o Hurst et al. (2024), Gemini-3-Flash, Gemini-2.5-Pro Team et al. (2023)); (2) base and RL models (Qwen2.5-Instruct Bai et al. (2023) and GRPO Shao et al. (2024)); (3) memory-augmented methods (Mem0 Chhikara et al. (2025) and SimpleMem Liu et al. (2026)), which leverage raw historical experiences; and (4) skill-augmented methods (D2Skill Tu et al. (2026) and SkillRL Xia et al. (2026)), which extract reusable strategic skills from experience.

Implementation Details We apply SCPO to Qwen2.5-7B-Instruct and Qwen2.5-3B-Instruct. The learning rate is 1×10^{-6}, the training batch size is 16, the rollout group size is 8, and the attribution weight α is 0.1 [7].

4.2 Main Results

Table 1 presents the main results. **SCPO consistently achieves the best performance across tasks and model scales.** Under Qwen2.5-7B-Instruct, it reaches 92.7% on ALFWorld and 79.7% on WebShop, exceeding the strongest baseline by 5.7% and 4.2%. **Memory and skill augmentation provide limited improvements across task categories.** While they generally outperform vanilla GRPO, their gains remain inconsistent. For instance, SkillRL improves several categories but still exhibits a clear performance drop on *Heat*, indicating that implicit skill augmentation alone cannot ensure robust skill utilization. **Explicit skill declaration and step-level attribution improve robustness across task categories.** SCPO consistently strengthens the weakest categories without sacrificing performance on stronger ones, indicating that explicit skill reasoning reduces skill misapplication and stabilizes policy learning.

4.3 Ablations

We evaluate four SCPO variants: (1) *Random Judge*, which replaces the LLM judge scores used for step-level retrospective attribution with random scores; (2) *w/o Soft Update*, which disables Track 1 skill updates; (3) *w/o Hard Update*, which disables Track 2 skill updates; and (4) *w/o Dual Track*, which disables both tracks.

[7] See Appendix B for implementation details and Appendix G for sensitivity to the attribution weight α.

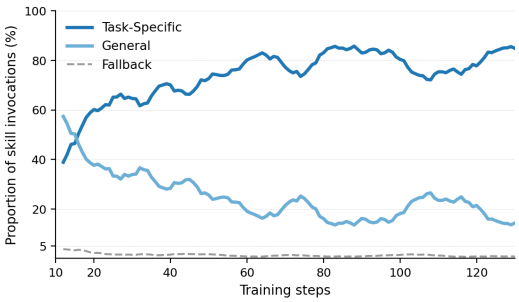


Figure 3: Skill category dynamics during training.

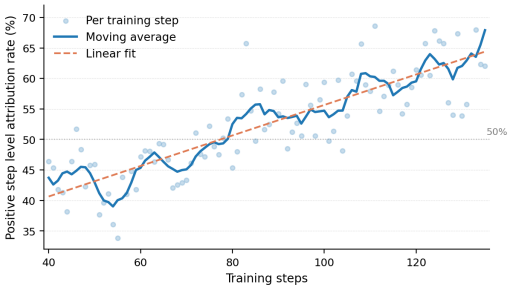


Figure 4: Skill quality during training.

As shown in Table 2, full SCPO consistently outperforms all variants. *Random Judge* performs worse, confirming that reliable step-level attribution is essential for providing informative optimization signals. Disabling either soft or hard updates also degrades performance, while *w/o Dual Track* yields the largest drop, highlighting the complementary roles of retrieval-weight adaptation and skill evolution. Overall, the results validate our design: declaration enables observability, attribution enables optimization, and dual-track evolution enables continuous adaptation.

Table 2: Ablation and evaluator robustness results on Qwen2.5-7B-Instruct (overall success rate, %).

Method	ALFWorld	WebShop
SCPO (Gemini-3-Flash)	92.7	79.7
Ablations		
w/ Random Judge	83.7	73.7
w/o Soft Update	90.6	78.1
w/o Hard Update	90.1	77.8
w/o Dual Track	88.0	77.3
w/ phase-level	85.9	75.8
Alternative Skill Judges		
w/ GPT-4o	92.2	80.5
w/ QwQ-32B	90.1	78.1

Next, we isolate the contribution of explicit skill declaration and examine the impact of different declaration types. A direct removal of skill declaration from SCPO does not provide a fair comparison, as the declared skill m_t is also involved in step-level attribution and skill-bank updates. Therefore, we construct controlled variants based on *w/o Dual Track*. Specifically, we introduce a *Phase-Level Declaration* variant, where the agent is required to declare one of four predefined task phases (`EXPLORE`, `PREPARE`, `EXECUTE`, and `VERIFY`) before reasoning and acting, instead of declaring skills. The four task phases are defined as follows: `EXPLORE` for information gathering, `PREPARE` for satisfying execution prerequisites, `EXECUTE` for performing the main operation, and `VERIFY` for checking the execution outcome.

w/o Dual Track (skill-level declaration) outperforms *Phase-Level Declaration* by 2.1% on ALFWorld and 1.5% on WebShop. This shows the gain comes from explicitly identifying reusable skills rather than merely adding an extra structured field. Besides, skill-grounded declarations provide more informative guidance than coarse task-phase labels.

4.4 More Quantitative and Qualitative Analysis

Skill Declaration Evolution. We examine how skill declaration evolves during training to analyze SCPO shifts from implicit skill injection to context-aware skill selection.

Figure 3 illustrates how the distribution of skill declaration categories evolves during training. Initially, task-specific skills account for only 38.5% of declarations, while general skills dominate at 57.7%. As training progresses, task-specific skill declarations increase to 81.3%, whereas general and fallback skills decrease to 17.6% and 1.1%, respectively. This shift suggests that the policy gradually moves away from broadly applicable skills and learns to selectively activate more targeted skills when appropriate, enabling more effective use of specialized knowledge.

Figure 4 further reports the positive step-level attribution rate throughout training, defined as the proportion of explicit skill declarations considered appropriate by the retrospective attribution mechanism. In the early stages, many declarations receive negative attribution, indicating that explicit declaration alone does not ensure effective skill utilization. As training progresses, the positive at-

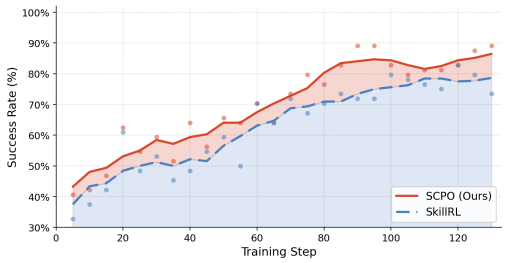


Figure 5: Overall success rate during training with Qwen2.5-7B-Instruct on ALFWorld.

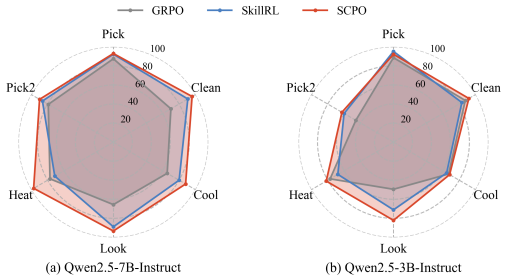


Figure 6: Task-level performance across ALFWorld categories. The radial axis denotes success rate (%).

tribution rate steadily increases, suggesting that SCPO learns to make increasingly appropriate skill choices rather than simply declaring skills more frequently.

These results show that the policy increasingly favors and appropriately selects task-specific skills.

Overall Convergence against SkillRL. Figure 5 compares the validation success rate of SCPO and SkillRL across training steps. It shows that SCPO consistently achieves higher validation success rates throughout training, with faster early-stage improvement and a higher final plateau. This demonstrates that explicit skill declaration and step-level attribution provide more effective learning signals than implicit skill injection, enabling more direct and reliable skill optimization.

Task-level Performance across Categories. Figure 6 visualizes ALFWorld performance across task categories. GRPO exhibits uneven performance with category-specific drops, while SkillRL improves several categories but remains inconsistent. In contrast, SCPO achieves a more balanced and uniformly improved performance across both model scales [8].

Robustness to the Skill Judge. As shown in Table 2, replacing the default Gemini-3-Flash skill judge with GPT-4o Hurst et al. (2024) or QwQ-32B Yang et al. (2024) yields stable overall performance, indicating that SCPO's gains are robust across different Skill judges [9].

Training-Time Computational Cost. We further evaluate the training-time cost of SCPO on ALFWorld. As shown in Table 3, SCPO maintains a wall-clock training time comparable to SkillRL across checkpoints, indicating that skill attribution and skill-bank evolution introduce only modest computational overhead.

Table 3: ALFWorld training cost (wall-clock seconds per step at three checkpoints).

Method	Time (s) / Step		
	Step 50	Step 100	Step 150
GRPO	634.1	576.0	592.5
SkillRL	691.0	654.4	685.6
SCPO (Ours)	**706.3**	**673.1**	**688.5**
⌞ Common RL	671.5	638.2	654.7
⌞ Skill Judge	+28.6	+29.1	+27.5
⌞ Skill Bank	+6.2	+5.8	+6.3

Additional Analysis. Appendix C validates the skill judge against human judgments and analyzes disagreements. Specifically, on ALFWorld, human-human agreement reaches 94.3% accuracy with a Cohen's κ of 0.886. Our skill judge achieves 89.7% accuracy and 89.5% macro-F1 scores, indicating strong consistency with human judgments. Appendix H presents case studies illustrating SCPO's execution and how explicit skill declarations guide task completion.

5 Conclusion

We introduced SCPO, a closed-loop framework that unifies explicit skill reasoning, retrospective attribution, and continuous skill evolution for long-horizon agentic decision making. Extensive experiments demonstrate that jointly optimizing the parametric policy and non-parametric skill bank consistently improves long-horizon task performance.

[8] See Appendix E for detailed dynamics for each task category.

[9] See Appendix F for ablation and robustness result details

REFERENCES

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi Gu, Chengcheng Han, Dengchang Zhao, Hui Su, Kefeng Zhang, et al. Vitabench: Benchmarking llm agents with versatile interactive tasks in real-world applications. *arXiv preprint arXiv:2509.26490*, 2025.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R Fung. Xskill: Continual learning from experience and skills in multimodal agents. *arXiv preprint arXiv:2603.12056*, 2026a.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. Sok: Agentic skills–beyond tool use in llm agents. *arXiv preprint arXiv:2602.20867*, 2026b.

Hao Li, Chunjiang Mu, Jianhao Chen, Siyue Ren, Zhiyao Cui, Yiqun Zhang, Lei Bai, and Shuyue Hu. Organizing, orchestrating, and benchmarking agent skills at ecosystem scale. *arXiv preprint arXiv:2603.02176*, 2026a.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026b.

Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, et al. Skillnet: Create, evaluate, and connect ai skills. *arXiv preprint arXiv:2603.04448*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*, 2026.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*, 2026.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using llm agents as research assistants. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, 2025.

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

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *arXiv preprint arXiv:2605.06130*, 2026.

Zhengliang Shi, Shen Gao, Lingyong Yan, Yue Feng, Xiuyi Chen, Zhumin Chen, Dawei Yin, Suzan Verberne, and Zhaochun Ren. Tool learning in the wild: Empowering language models as automatic tool agents. In *Proceedings of the ACM on Web Conference 2025*, pp. 2222–2237, 2025.

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

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

Songjun Tu, Chengdong Xu, Qichao Zhang, Yaocheng Zhang, Xiangyuan Lan, Linjing Li, and Dongbin Zhao. Dynamic dual-granularity skill bank for agentic rl. *arXiv preprint arXiv:2603.28716*, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Jiayu Wang, Yifei Ming, Zixuan Ke, Shafiq Joty, Aws Albarghouthi, and Frederic Sala. Skillorchestra: Learning to route agents via skill transfer. *arXiv preprint arXiv:2602.19672*, 2026.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026.

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, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, et al. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*, 2026.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022b.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Yuanheng Zhu, Qichao Zhang, Zijie Zhao, and Dongbin Zhao. Criticsearch: Fine-grained credit assignment for search agents via a retrospective critic. *arXiv preprint arXiv:2511.12159*, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, et al. Skillrouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 2026.

A THEORETICAL ANALYSIS OF EXPLICIT SKILL DECLARATION

We analyze why Explicit Skill Declaration provides a direct optimization path for skill invocation. At step t, let $x_t = (o_{\le t}, g, \mathcal{S}_t)$, where $\mathcal{S}_t \subseteq \mathcal{M}$ is the retrieved candidate skill set. Following Eq. equation 1, define $\bar{y}_t = [\text{think}_t; a_t]$ as the remaining generated tokens after the declared skill identifier m_t. For notation, m_t is treated as an atomic declaration unit; in implementation, $\log \pi_\theta(m_t \mid x_t)$ is the sum of token log-probabilities.

By autoregressive factorization,

$$\pi_\theta(y_t \mid x_t) = \pi_\theta(m_t \mid x_t) \cdot \pi_\theta(\bar{y}_t \mid x_t, m_t). \tag{8}$$

This factorization makes the declared skill identifier part of the response sequence, so that skill declarations are directly exposed to policy gradients.

Policy-gradient decomposition. During each policy update, $\mathcal{S}_t$, the evaluator output, and retrieval weights are fixed. Let

$$C_{i,t} = \mathrm{sg}(A_{\mathrm{SCPO},i,t}), \tag{9}$$

where $A_{\mathrm{SCPO},i,t}$ is the composite SCPO advantage defined in Eq. equation 4, and $\mathrm{sg}(\cdot)$ denotes the stop-gradient operation, ensuring the advantage acts as a fixed scalar weight during policy optimization. Ignoring clipping and KL regularization, the first-order term of Eq. equation 5 is

$$\widetilde{J}_{\mathrm{SCPO}}(\theta) = \mathbb{E}_{\tau_i \sim \pi_{\mathrm{old}}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i} r_{i,t} C_{i,t}\right], \tag{10}$$

where $r_{i,t}$ is the step-level importance sampling ratio defined in Eq. equation 5.

Theorem 1. *Under Explicit Skill Declaration, the unclipped SCPO policy gradient decomposes into the skill-level term for the declared skill identifier and the remaining response-level term conditioned on the declared skill.*

Proof. Applying the likelihood-ratio identity to Eq. equation 10 gives

$$\nabla_\theta \widetilde{J}_{\mathrm{SCPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i} r_{i,t} C_{i,t} \nabla_\theta \log \pi_\theta(y_{i,t} \mid x_{i,t})\right]. \tag{11}$$

From Eq. equation 8,

$$\nabla_\theta \log \pi_\theta(y_{i,t} \mid x_{i,t}) = \nabla_\theta \log \pi_\theta(m_{i,t} \mid x_{i,t}) + \nabla_\theta \log \pi_\theta(\bar{y}_{i,t} \mid x_{i,t}, m_{i,t}). \tag{12}$$

Substituting it into Eq. equation 11 yields

$$\nabla_\theta \widetilde{J}_{\mathrm{SCPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i} r_{i,t} C_{i,t} \cdot \Big(\nabla_\theta \log \pi_\theta(m_{i,t} \mid x_{i,t}) + \nabla_\theta \log \pi_\theta(\bar{y}_{i,t} \mid x_{i,t}, m_{i,t})\Big)\right]. \tag{13}$$

This proves the theorem.

The first component in Eq. equation 13 is the skill-level term:

$$g^m_{i,t} = r_{i,t} C_{i,t} \nabla_\theta \log \pi_\theta(m_{i,t} \mid x_{i,t}). \tag{14}$$

Hence, the declared skill identifier is directly exposed to policy-gradient optimization. In the unclipped local update, a positive composite SCPO advantage tends to increase the probability of the corresponding skill declaration, while a negative one tends to suppress it. Clipping and KL regularization may attenuate or truncate this term, but they do not remove the factorization in Eq. equation 8.

Why skill grounding matters. The decomposition above also applies to other generated fields, so the existence of a direct policy-gradient term alone does not guarantee a performance improvement. Its relevance to skill selection arises because m_t identifies a concrete entry in the retrieved candidate set $\mathcal{S}_t$. A generic phase label may also receive policy-gradient updates, but because it is not grounded in the skill bank, these updates do not directly change the policy's relative preference among candidate skills. Theorem 1 therefore establishes the optimization path rather than guaranteeing a performance improvement. Whether this path is useful depends on grounding the generated declaration in a specific skill-bank entry and assigning credit that reflects the downstream value of that choice.

Credit assignment bottleneck in implicit skill injection. For implicit skill injection, let $z_t = (o_{\leq t}, g)$ and let $q(m_t \mid z_t)$ denote an external retriever. The joint distribution under implicit skill injection is

$$p_\theta^{\text{implicit}}(\bar{y}_t, m_t \mid z_t) = q(m_t \mid z_t) \cdot \pi_\theta(\bar{y}_t \mid z_t, m_t). \tag{15}$$

Since q is decoupled from the agent policy parameters θ, $\nabla_\theta \log q(m_t \mid z_t) = 0$. Therefore,

$$\nabla_\theta \log p_\theta^{\text{implicit}}(\bar{y}_t, m_t \mid z_t) = \nabla_\theta \log \pi_\theta(\bar{y}_t \mid z_t, m_t). \tag{16}$$

Eq. equation 16 shows that implicit skill injection optimizes the response conditioned on the retrieved skill, but does not directly optimize skill selection through the policy gradient of the agent policy. In other words, the policy-gradient update of θ is blind to the retrieval decision over $\mathcal{M}$. If an inappropriate skill is retrieved, the negative learning signal can only affect the response generated under that skill, while the retrieval decision itself remains outside this policy-gradient path. This creates a credit assignment bottleneck between skill selection and policy optimization. This statement does not preclude external mechanisms from updating retrieval weights or skill content; it only characterizes the first-order update of the agent policy parameters θ.

In contrast, SCPO makes m_t part of the policy output. The same credit signal $C_{i,t}$ directly modulates the probability of the declared skill identifier through Eq. equation 14, thereby establishing a direct optimization channel for declared skill invocations.

Effect of step-level retrospective attribution. The above decomposition holds for any scalar credit signal $C_{i,t}$. SCPO further uses the composite SCPO advantage

$$A_{\text{SCPO},i,t} = \hat{A}_i + \alpha \cdot A_{\text{step},i,t}, \tag{17}$$

where $\hat{A}_i$ is the trajectory-level advantage and $A_{\text{step},i,t}$ is the normalized step-level attribution advantage. Substituting Eq. equation 17 into Eq. equation 14 gives

$$g_{i,t}^m = r_{i,t}(\hat{A}_i + \alpha \cdot A_{\text{step},i,t}) \nabla_\theta \log \pi_\theta(m_{i,t} \mid x_{i,t}). \tag{18}$$

Therefore, the skill-level term is not only driven by the terminal reward through $\hat{A}_i$, but also adjusted by the step-level retrospective attribution mechanism. This allows SCPO to assign different credit signals to different declared skill identifiers within the same interaction trajectory.

For a fixed context x, let

$$\mu(m; x) = \mathbb{E}[A_{\text{SCPO},t} \mid x_t = x, m_t = m] \tag{19}$$

denote the expected composite SCPO advantage of declaring skill m. Under a local softmax parameterization and an infinitesimal unclipped gradient step, with fixed context x and fixed candidate skill set $\mathcal{S}_t$, the expected update of the skill declaration probability is proportional to

$$\pi_\theta(m \mid x) \Big(\mu(m; x) - \sum_{m' \in \mathcal{S}_t} \pi_\theta(m' \mid x) \mu(m'; x) \Big). \tag{20}$$

Here, the dynamically weighted average

$$\sum_{m' \in \mathcal{S}_t} \pi_\theta(m' \mid x) \mu(m'; x) \tag{21}$$

acts as a natural baseline. Thus, skills whose expected composite SCPO advantage is above the current average tend to be encouraged, while skills whose expected composite SCPO advantage is below the current average tend to be suppressed. This provides a fine-grained mechanism to evaluate whether each explicit skill choice truly suits the current state.

Conclusion. Explicit Skill Declaration transforms a retrieved skill from an implicit contextual input into a skill-grounded policy output. Unlike an arbitrary generated label, m_t identifies a concrete entry in the retrieved candidate set and introduces a direct policy-gradient term over candidate skill choices. Compared with implicit skill injection, SCPO preserves the optimization of the response conditioned on the retrieved skill while adding a direct skill-level term for the declared skill identifier. By combining the trajectory-level advantage with the normalized step-level attribution advantage, SCPO injects fine-grained advantage signals into the policy-gradient loop and directly connects skill selection with policy optimization. This establishes a direct optimization channel for declared skill invocations and supports more appropriate skill selection during interactions.

Table 4: Main hyperparameters used in SCPO.

Hyperparameter	Value
Model	
Backbone models	Qwen2.5-7B-Instruct / Qwen2.5-3B-Instruct
Evaluator / teacher model	Gemini-3-Flash
RL Training	
Learning rate	1×10^{-6}
Training batch size	16
Rollout group size	8
KL coefficient	0.01
Max prompt length	6,000
Max response length	1,024
Training iterations	180 (ALFWorld) / 150 (WebShop)
SCPO	
Attribution weight α	0.1
Retrieved skills per step	6 task-specific + 6 general + 1 fallback
Retrieval balance λ	0.7
Soft update interval	10 iterations
Hard update interval	20 iterations
Revision threshold	-0.2
Generation threshold	0.4

B Experimental Setup Details

To ensure a rigorous and fair comparison, we keep the core optimization and evaluation configurations identical across trainable baselines. For both Qwen2.5-7B-Instruct and Qwen2.5-3B-Instruct backbones, we use consistent hyperparameters, including learning rate, batch size, rollout group size, KL coefficient, and sequence lengths. SCPO is implemented as an incremental extension over GRPO, introducing explicit skill declaration, step-level retrospective attribution, and credit-driven evolution. Beyond these specific modules, the training protocol remains aligned with GRPO and other reinforcement learning baselines. For closed-source LLM baselines, the GPT-4o results and Gemini-2.5-pro results are taken from SkillRL Xia et al. (2026), while the Gemini-3-Flash results are taken from D2Skill Tu et al. (2026). All main experiments were conducted on 8 NVIDIA A100 GPUs, with each full SCPO training run taking approximately 50 hours. Most experiments are run three times.

The initial skill bank in SCPO is the same as Xia et al. (2026); Lu et al. (2026). We do not use validation or test trajectories to construct or modify the initial skill bank in our experiments. Throughout training, the skill bank evolves solely through the credit-driven update mechanism described in the methodology. During retrieval, each step incorporates 6 task-specific skills, 6 general skills, and 1 mandatory fallback skill. Gemini-3-Flash serves as the step-level evaluator and the teacher model for skill updates. For fair comparison, we use the same Gemini-3-Flash as the teacher model for the trainable baselines. The outputs of Gemini are used strictly as attribution and evolution signals and do not participate in the gradient computation of the policy model. The primary hyperparameters are summarized in Table 4.

We use ALFWorld and WebShop in accordance with their MIT licenses. All artifacts are used only for research and benchmark evaluation, not for real-world deployment. We do not release any new dataset or repackaged benchmark artifact.

C Validation of Skill Attribution by the Skill Judge

Human Validation of Skill Attribution To evaluate the consistency between the skill judge's step-level attribution and human judgments, we sampled 300 decision steps each from ALFWorld and WebShop. Each dataset maintains a balanced distribution (150 from successful and 150 from failed trajectories). In ALFWorld, Skill Judge identified 152 samples as GOOD and 148 as BAD, while in WebShop, the distribution was exactly 150/150. Two independent annotators, who are full-

Table 5: Error analysis of disagreements between the Skill Judge and human reference labels.

Disagreement Type	**Count**	**Ratio**
Stale skill declarations after task-state transitions	31	43.1%
Ambiguity between overlapping or related skills	22	30.6%
Insufficient evidence for binary attribution	19	26.4%

Table 6: Detailed component ablation and evaluator robustness results on Qwen2.5-7B-Instruct. We report ALFWorld success rates (%) across different task types and overall performance, together with WebShop score and success rate (%).

Method	**ALFWorld**							**WebShop**	
	Pick	Clean	Cool	Look	Heat	Pick2	All	Score	Success
SCPO (Gemini-3-Flash)	93.0	95.8	87.9	93.3	97.0	89.8	92.7	90.4	79.7
Component Ablations									
Random Judge	86.3	93.5	65.0	82.0	90.5	79.7	83.7	84.9	73.7
w/o Soft Update	100.0	87.5	92.3	80.0	100.0	75.0	90.6	89.3	78.1
w/o Hard Update	95.8	98.1	76.8	72.2	88.9	86.4	90.1	88.7	77.8
w/o Dual Track	97.8	95.8	78.4	60.0	93.9	75.0	88.0	87.1	77.3
Phase-level Declaration	94.4	90.0	70.0	87.5	81.8	85.7	85.9	86.2	75.8
Alternative Skill Judges									
w/ GPT-4o	94.4	96.1	84.6	95.2	96.7	88.6	92.2	92.4	80.5
w/ QwQ-32B	91.5	93.9	83.1	93.3	93.9	88.1	90.1	93.1	78.1

time internal NLP researchers at our institution, labeled each sample following the exact outcome-conditioned evaluation rules provided to the skill judge. Specifically, they assessed whether each declared skill was appropriate for the current state and whether the subsequent reasoning and action followed its principle, without judging action correctness in isolation. Both annotators provided informed consent for this data evaluation task, and their participation was compensated as part of their standard employment salary, ensuring compliance with fair wage guidelines. We measured inter-annotator agreement (Accuracy and Cohen's κ Cohen (1960)) and evaluated Skill Judge against consensus human labels, with a focus on BAD-class metrics to reflect its ability to detect skill misuse.

- **ALFWorld:** Human-human Accuracy was 94.3% with a Cohen's κ of 0.886. Skill Judge achieved an Accuracy of 89.7%, a Macro-F1 of 0.895, and a BAD-class F1 of 0.882.
- **WebShop:** Human-human Accuracy was 91.0% with a Cohen's κ of 0.821. Skill Judge yielded an Accuracy of 86.3%, a Macro-F1 of 0.861, and a BAD-class F1 of 0.848.

Disagreement analysis. We further examine all 72 disagreements between the Skill Judge and the human reference labels, including 31 from ALFWorld and 41 from WebShop. Three recurring patterns emerge, as summarized in Table 5.

The largest group consists of stale skill declarations after task-state transitions. In these cases, the agent has already moved to a new subgoal, while the declaration still refers to a skill associated with the preceding phase. The second group reflects ambiguity between overlapping or related skills, where both a general skill and a more task-specific skill appear plausible. The remaining cases provide insufficient evidence to support a reliable binary attribution. These results show that most disagreements occur near attribution boundaries rather than indicating a broad failure of the evaluation criterion.

More fine-grained rules could further improve attribution accuracy. Once a task-phase transition has been completed, the declaration should be evaluated against the new state, and continued invocation of a previous-phase skill should be marked BAD. The rubric could also specify that the most specific applicable skill takes precedence when several related skills overlap. Cases with insufficient evidence could be excluded from attribution supervision or reviewed separately instead of forcing

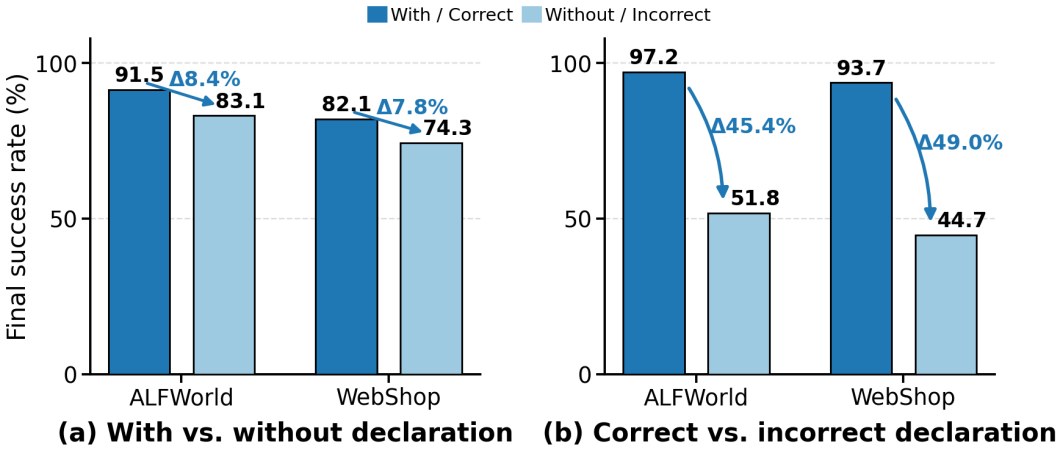


Figure 7: Performance of SkillRL (7B) Xia et al. (2026) on ALFWorld and WebShop. (a) Final success rates of interaction trajectories with and without explicit skill declarations. (b) Final success rates of explicitly declared trajectories, further categorized by whether the agent selected correct or incorrect skills.

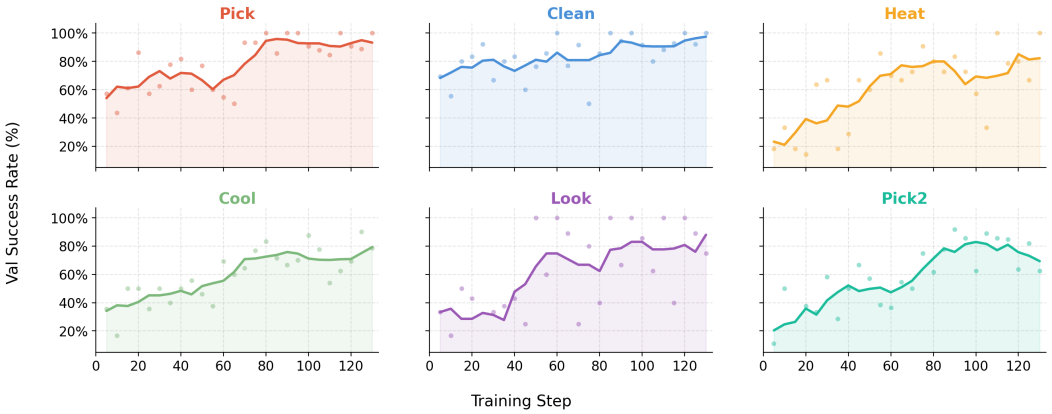


Figure 8: Subtask-level validation success rates of SCPO (7B) during training on ALFWorld.

an uncertain binary label. Accordingly, we report confidence intervals, analyze the disagreement patterns, and identify possible refinements to the attribution rules.

D An Empirical Motivating Example

The motivating example in Figure 1 illustrates why implicit skill use is difficult to diagnose. When the agent holding an uncleaned apple directly moves to the dining table, the action is clearly inappropriate, but the underlying skill choice remains hidden. The agent may have ignored the cleaning skill, selected the direct delivery skill, or failed to use any retrieved skill. Explicit skill declaration makes this previously hidden choice observable. However, as the same example shows, an observable declaration may still be inappropriate for the current state.

We provide an additional experimental analysis of this phenomenon using test trajectories generated by trained SkillRL (7B) checkpoints. For each benchmark, the compared trajectories are generated by the same fixed checkpoint. The analysis is therefore a within-checkpoint observational diagnostic rather than a causal intervention.

Making Skill Selection Observable. Figure 7(a) separates trajectories according to whether the checkpoint produces an explicit skill declaration during interaction. On ALFWorld, trajectories with explicit declarations achieve a final success rate of 91.5%, compared with 83.1% for trajectories without declarations. On WebShop, the corresponding success rates are 82.1% and 74.3%. Averaged across the two benchmarks, the success rate is 86.8% with explicit declarations and 78.7% without them.

This comparison reflects the first issue illustrated by the apple example. When no declaration is produced, the retrieved skills are visible as contextual information, but the skill selected by the agent remains implicit. When the agent explicitly declares a skill, skill selection becomes an observable policy output that can be evaluated separately.

Observable Does Not Mean Appropriate. Figure 7(b) further examines the subset of trajectories containing explicit skill declarations. The declarations are categorized according to whether the selected skill is appropriate for the corresponding task state, denoted as correct or incorrect in the figure. On ALFWorld, trajectories with correct skill selections achieve 97.2% success, whereas those with incorrect selections achieve 51.8%. On WebShop, the corresponding success rates are 93.7% and 44.7%. Averaged across both benchmarks, correct declarations achieve 95.5% success, compared with 48.3% for incorrect declarations.

This result corresponds to the second part of the apple example. Declaring *go to the sink first* is appropriate while the apple remains uncleaned, whereas declaring *move directly to the destination* exposes an inappropriate skill choice. Thus, explicit skill declaration resolves the ambiguity of hidden skill selection, but does not itself ensure that the declared skill is appropriate for the current state.

Interpretation. The observed differences in Figure 7 should not be interpreted as the causal effect of adding a declaration field. Trajectories are grouped according to outputs already generated by the same trained checkpoint, and the groups may also differ in task difficulty or model confidence. Instead, this diagnostic shows what becomes measurable once skill selection is made explicit. It exposes a large difference between appropriate and inappropriate declarations and therefore motivates declaration-specific feedback.

The qualitative example in Figure 1 further shows that an appropriate declaration may still be followed by reasoning or an action that violates the declared skill. SCPO therefore introduces step-level retrospective attribution to evaluate both whether each declared skill is appropriate for the current state and whether the subsequent behavior is consistent with it. Controlled evidence for the effect of explicit skill declaration is provided by the matched SkillRL variants in Section 4.3, which keep the skill bank, retrieval mechanism, training configuration, and evaluation protocol unchanged.

E Detailed Category-level Training Dynamics

Figure 8 reports SCPO's validation success rate across six ALFWorld task categories during training. Despite stochastic fluctuations, all categories exhibit overall upward trends. Pick and Clean quickly stabilize at high success rates, while more challenging categories such as Heat, Cool, Look, and Pick2 improve steadily from lower initial performance. These results suggest that explicit skill declaration and step-level retrospective attribution help SCPO learn across heterogeneous task types rather than overfitting to a single easy category.

F Detailed Ablation and Robustness Results

Table 6 presents the complete results for the component ablations and alternative Skill Judges, including subtask-level success rates on ALFWorld and both score and success rate on WebShop. For evaluator robustness, we replace the default Gemini-3-Flash judge with GPT-4o Hurst et al. (2024) and QwQ-32B Yang et al. (2024) while keeping the remaining training configuration unchanged.

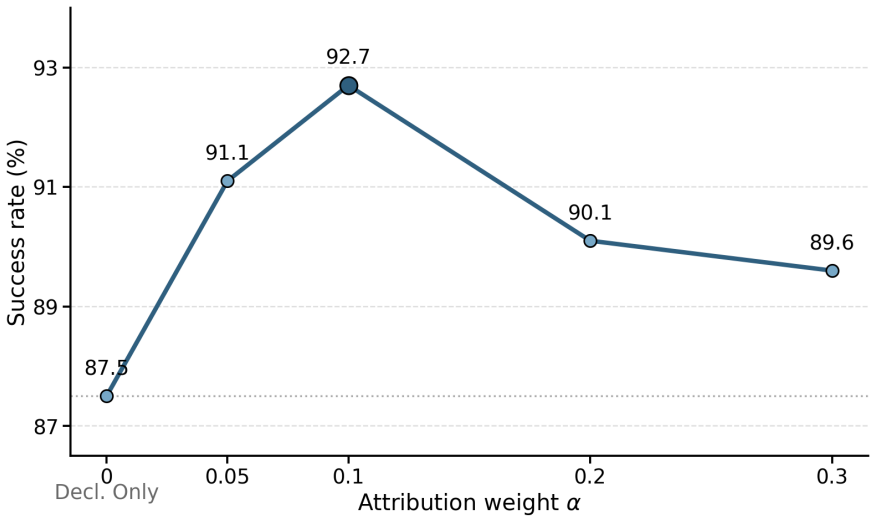


Figure 9: Sensitivity to attribution weight α. Results are reported on ALFWorld using Qwen2.5-7B-Instruct.

G Sensitivity to Attribution Weight α

We further study the sensitivity of SCPO to the attribution weight α, which controls how strongly the step-level retrospective attribution signal affects policy optimization. When $\alpha = 0$, the model reduces to the Decl. Only setting: the agent is still required to explicitly declare a skill, but these declarations do not receive step-level corrective feedback during training.

As shown in Figure 9, performance first improves as α increases and then declines when α becomes too large. With $\alpha = 0$, Decl. Only achieves a success rate of 87.5%, showing that explicit skill declaration alone is helpful but insufficient. Increasing α to 0.05 improves the success rate to 91.1%, and the best result is obtained at $\alpha = 0.1$, reaching 92.7%. This indicates that moderate step-level attribution provides useful corrective feedback for declared skills. However, further increasing α to 0.2 and 0.3 reduces performance to 90.1% and 89.6%, respectively. This suggests that overly large attribution weights may place too much emphasis on local declaration feedback, making optimization less balanced with the final task outcome.

Overall, the results show that step-level retrospective attribution is important for improving explicit skill declaration, but its weight should be properly balanced. We therefore set $\alpha = 0.1$ as the default value in our main experiments.

H Case Studies

We present two case studies to illustrate the practical execution and internal mechanisms of SCPO.

Table 7 shows a successful trajectory on a Heat task, highlighting how appropriate explicit skill declarations provide precise, step-by-step strategic guidance that effectively drives task completion.

Table 8 further emphasizes the critical role of our step-level evaluator on a Clean task. It showcases how the evaluator detects subtle phase-mismatch errors, such as the agent inappropriately re-invoking a cleaning skill after the target object is already clean. This demonstrates that step-level attribution provides fine-grained, corrective credit signals that are entirely invisible to sparse trajectory-level rewards.

I Skill Taxonomy and Definitions

A skill in SCPO is a reusable natural-language strategic principle distilled from historical interactions. It provides high-level guidance for decision making rather than specifying an executable

Table 7: Case study of SCPO on a heat task. The agent explicitly declares a skill identifier at each step and all declarations are judged as appropriate by the step-level evaluator.

Task: heat some cup and put it in cabinet.

Declared Skills
`[hea_002]` **Systematic One-Pass Search**; `[hea_001]` **Secure Exact Target First**
`[hea_003]` **Open Then Heat**: Upon reaching the microwave with the target in hand, **always open the door, place the object inside, and execute the heat action before leaving.**
`[hea_005]` **Direct Post-Heat Placement**: After heating, navigate straight to the specified destination and place the object once, avoiding extra exploration or detours.

Step	Declared Skill	Think	Action	Judge
1–2	`hea_002` `hea_001`	*[Search & Acquire Phase: The agent searches for the target and picks up the exact cup before using the microwave.]*	`go to countertop 1` → `take cup 1 from countertop 1`	**GOOD** × 2
3–5	`hea_003`	**(3)** I should go to the microwave and prepare to heat the cup. This aligns with **[hea_003]**. **(4)** I should open the microwave before heating the object, following the required heating procedure. **(5)** The cup is ready to be heated, so I should **execute the heat action** with the microwave.	**(3)** `go to microwave 1` **(4)** `open microwave 1` **(5)** `heat cup 1 with microwave 1`	**GOOD** × 3
6–7	`hea_005`	**(6)** After heating, I should avoid extra exploration and **move directly to the specified destination**. **(7)** I should **place the heated cup directly in the cabinet** without any additional steps.	**(6)** `go to cabinet 1` **(7)** `move cup 1 to cabinet 1`	**GOOD** × 2
Result: Task completed successfully in 7 steps.				**Success**

action or a fixed action sequence. Each task-specific or general skill contains a unique identifier, a short title, a strategic principle, and a description of when the skill should be applied. The identifier is used by the policy during explicit skill declaration, while the remaining fields describe the decision strategy represented by the skill and its applicable conditions.

The skill bank $\mathcal{M}$ consists of three types of skills: task-specific skills, general skills, and a fixed fallback skill.

Task-Specific Skills. Task-specific skills provide specialized guidance for particular task categories or recurring subtasks. They capture task-dependent prerequisites, operation order, and intermediate objectives. For example, a cleaning skill in ALFWorld may specify that the agent should first acquire the target object, navigate to a water source, complete the cleaning operation, and only then move to the destination. Similarly, a WebShop task-specific skill may describe how to verify product specifications or price constraints before purchase. These skills provide precise guidance when the current state corresponds to the associated task category or subtask.

General Skills. General skills encode broadly applicable experience that can be reused across different task categories. They describe decision strategies such as systematic exploration, goal decomposition, prerequisite checking, and constraint verification. Unlike task-specific skills, general skills are not tied to a particular operation or product category. They provide high-level guidance when a broadly applicable strategy is more suitable than a specialized task-specific skill.

Fallback Skill. Both benchmarks use the same fixed universal fallback skill:

```
[gen_000] Fallback Reasoning: When no other skill
applies to the current state, select this skill.
```

Table 8: Case study of SCPO on a clean task. The trajectory succeeds overall, while the step-level evaluator precisely detects a phase-mismatched `cle_003` declaration at Step 5.

Task: clean some apple and put it in diningtable.

Declared Skills
`[gen_006]` **Establish Spatial Relations** ; `[cle_002]` **Pick Before You Wander**; `[cle_001]` **Phase-Ordered Plan**
`[cle_003]` **Sink First for Cleaning**: Upon holding the target object, go straight to the nearest sink, basin, or faucet and issue the clean command **before any placement attempts**.

Step	Declared Skill	Think	Action	Judge
1–4	`gen_006` `cle_002` `cle_003`(2)	*[Setup & Cleaning Phase: The agent locates the apple, picks it up, moves to the sinkbasin, and cleans it.]*	*[Navigation, Pick & Clean actions]*	**GOOD** $\times 4$
5	`cle_003`	Since the apple is now cleaned. I need to **place it on the dining table as per the task requirements**.	`go to diningtable 1`	**BAD**
6	`cle_001`	Since the apple is already clean, I should **proceed with placing it on the dining table as per the fixed sequence**.	`move apple 2 to diningtable 1`	**GOOD**

Attribution Note: Step 5 is judged as **BAD** because `cle_003` is only appropriate before cleaning. After the apple is already clean, re-declaring `cle_003` becomes phase-mismatched, even though the action `go to diningtable` is still reasonable. This shows that step-level attribution can catch skill-declaration errors hidden by the successful trajectory reward.

Result: Task completed successfully in 6 steps. **Success**

The fallback skill is always included in the retrieved skill set. It provides a valid declaration when none of the retrieved task-specific or general skills is suitable for the current state, preventing the policy from being forced to select an irrelevant skill. Unlike task-specific and general skills, `gen_000` does not encode a benchmark-specific strategic principle and does not contain a separate "when to apply" field.

Initial Skill-Bank Composition. For the main experiments, we use the same initial skill banks as the skill-augmented baselines to avoid confounding the effect of SCPO with differences in skill-bank construction. The initial task-specific and general skills are adopted from the released SkillRL Xia et al. (2026) resources. The ALFWorld skill bank contains 44 task-specific and general skills, including 32 task-specific skills and 12 general skills, while the WebShop skill bank contains 54 skills, including 39 task-specific skills and 15 general skills. Each benchmark additionally includes the fixed fallback skill `gen_000`, resulting in 45 and 55 entries in the complete initialized skill banks, respectively. Table 9 presents representative examples from the initialized skill banks. No validation or test trajectories are used to construct or modify the initial skill banks. During SCPO training, subsequent retrieval-weight adjustments, skill revisions, and skill additions are based only on training interactions.

Retrieval and Explicit Declaration. Following prior skill-retrieval pipelines Xia et al. (2026); Tu et al. (2026), we use `Qwen3-Embedding-0.6B` as the embedding model. For each task-specific or general skill m, we concatenate its title, strategic principle, and when-to-apply description as its textual representation and encode it as e_m. The current task goal g is encoded as e_g in the same embedding space, and cosine similarity between e_g and e_m measures their semantic relevance. Following Eq. equation 6, this similarity is combined with the historical credit weight to obtain the final retrieval score. SCPO ranks task-specific and general skills separately and retrieves the top $K = 6$ skills from each pool. The mandatory fallback skill `gen_000` is then added, yielding thirteen candidate skills at each step. Before generating its reasoning and action, the policy must explicitly declare exactly one skill from this set. This makes the selected skill an observable policy output while preserving a valid fallback when none of the retrieved task-specific or general skills is appropriate.

Table 9: Representative skill examples from the initialized ALFWorld and WebShop skill banks.

Dataset	ID	Skill Title	Principle	When to Apply
ALFWorld	gen_010	Progressive Goal Decomposition	Decompose the task into locating, acquiring or transforming, locating the destination, and completing placement.	At the start of an episode or when the current sub-goal becomes invalid.
ALFWorld	cle_001	Phase-Ordered Plan	Acquire the target, clean it at a water source, navigate to the destination, and place it.	When the goal requires a clean object.
WebShop	gen_010	Respect Constraint Hierarchy	Treat product type, category, price, size, and color as hard constraints that secondary preferences cannot override.	When evaluating whether to inspect or purchase a product.
WebShop	ele_003	Hard-Gate Price & Specs	Verify the title, specifications, and price, and reject products that exceed the budget or miss required features.	After configuring variants and before purchase.

J PROMPTS

This section presents the prompt templates used in SCPO. Unless otherwise specified, the same prompt structure is used for ALFWorld and WebShop, with only environment-specific observations, action spaces, and task descriptions adapted.

Agent Execution Prompts At each step, the agent receives the task goal, interaction history, current observation, and retrieved candidate skills. The prompt requires the agent to output a structured response consisting of an explicit skill declaration, reasoning, and an environment action:

$$y_t = \{m_t; \text{think}_t; a_t\}.$$

This format is used for both ALFWorld and WebShop.

Step-level Retrospective Attribution Prompts We use Gemini-3-Flash as the Skill Judge, with the complete prompt shown in Table 12. After each trajectory, the judge evaluates every declared skill invocation using the task goal, final outcome, available skill set, and complete interaction trajectory. It checks two conditions: (1) whether the declared skill is appropriate for the current state, and (2) whether the subsequent reasoning and action follow the declared skill principle.

The raw attribution label is binary by design. A skill invocation is assigned $+1$ only when both conditions are satisfied, corresponding to Appropriate Skill Use. It is assigned -1 when either condition fails, covering both Wrong Skill Selection and Correct Skill but Wrong Execution. We do not use an intermediate raw value because the judge is asked to determine whether a skill invocation is valid rather than to calibrate its degree of correctness. Relative, graded credit is introduced subsequently through intra-trajectory centering and scaling by the batch-level standard deviation in Eq. 3.

Skill Bank Evolution Prompts For skill bank evolution, we use teacher-model prompts to revise low-credit skills and generate new skills from underperforming task categories. The revision prompt refines an existing skill, while the generation prompt distills new reusable skills from representative trajectories.

Table 10: Prompt Template for Agent Execution with Explicit Skill Declaration (ALFWorld)

System Prompt:
You are an expert agent operating in the ALFRED Embodied Environment. Your task is to:
{task_description}

Retrieved Relevant Experience
General Principles
{retrieved_general_skills}
Task-Special Skills
{retrieved_task_special_skills}
Universal Fallback
[gen_000] Fallback Reasoning: When no other skill applies to the current state, select this skill.

Current Progress
Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].

Now it's your turn to take an action. You should select the SINGLE most relevant skill that best guides your action at this step. Declare it using <skill_invoked>skill_id</skill_invoked> (e.g., <skill_invoked>{example_skill_id}
</skill_invoked>). Do not list multiple IDs; only the single most decisive ID is allowed. Then reason within <think > </think > tags, and choose an action within <action> </action> tags.

Table 11: Prompt Template for Agent Execution with Explicit Skill Declaration (WebShop)

System Prompt:
You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to:
{task_description}.

Retrieved Relevant Experience
General Principles
{retrieved_general_skills}
Task-Special Skills
{retrieved_task_special_skills}
Universal Fallback
[gen_000] Fallback Reasoning: When no other skill applies to the current state, select this skill.

Current Progress
Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].

Now it's your turn to take an action. You should select the SINGLE most relevant skill that best guides your action at this step. Declare it using <skill_invoked>skill_id</skill_invoked> (e.g., <skill_invoked>{example_skill_id}
</skill_invoked>). Do not list multiple IDs; only the single most decisive ID is allowed. Then reason within <think > </think > tags, and choose an action within <action> </action> tags.

Table 12: Prompt Template for Declared Skill Invocation Evaluation

System Prompt:
You are a professional skill invocation evaluator specializing in retrospective attribution analysis. Given an agent's task trajectory and final score, evaluate each declared skill invocation.

Evaluation rules:
If final score > 0 (task succeeded): GOOD means the declared skill is appropriate for the current state and the subsequent reasoning and action follow its principle; BAD means either the skill selection is inappropriate or the subsequent reasoning or action fails to follow the declared skill.
If final score ≤ 0 (task failed): GOOD means the declared skill is appropriate and correctly followed at the current step, with the failure attributable to later steps or the environment; BAD means either the skill selection is inappropriate or the subsequent reasoning or action fails to follow the declared skill and contributes to the failure.

Focus: evaluate whether the declared skill is appropriate and followed by the subsequent reasoning and action, rather than judging action correctness in isolation. For each step, first write Analysis (one sentence), then write Judgment (GOOD or BAD).

Task: {goal}
Final score: {score} ({result})
Skill bank (available skills): {skill_list}
Trajectory: {trajectory}
For each step output: Step N - Analysis: <brief analysis> - Judgment: GOOD/BAD

Table 13: Prompt Template for Trace-Based Skill Revision

System Prompt:
You are analyzing why an AI agent skill is performing poorly.

SKILL BEING ANALYZED: skill_id: {skill_id}, title: {title}, principle: {principle}, when_to_apply: {when_to_apply}

FAILED TRAJECTORIES (where this skill was invoked but the task failed): {fail_sections}

SUCCESS TRAJECTORIES (same task type, for contrast): {success_section}

TASK: First, identify FIRST_ERROR_STEP: N (1-based turn number where agent first went wrong, 0 if unclear). Then, revise the skill to fix the root cause. Only update principle and/or when_to_apply if needed.

Output Format: Return ONLY a JSON object (no other text): {"first_error_step": N, "principle": "revised principle", "when_to_apply": "revised when_to_apply"}
If the skill is fundamentally correct and the failure is due to environment or other skills, return: {"first_error_step": N, "no_change": true}

Table 14: Prompt Template for Trajectory-Based Skill Generation

System Prompt:
Analyze these failed agent trajectories and suggest NEW skills to add to the skill bank.

FAILED TRAJECTORIES: {failure_examples}
EXISTING SKILL TITLES (avoid duplicating these): {existing_titles}

Generation Requirements: Generate 1-{max_new_skills} NEW actionable skills that would help avoid these failures. Each skill must have: skill_id, title (3-5 words), principle (1-2 sentences), when_to_apply. Use skill_ids: {example_ids}

Output Format: Return ONLY a JSON array of skills, no other text.

Example format: `[{"skill_id":"dyn_001","title": "...","principle": "...", "when_to_apply":"..."}]`

Table 15: Prompt Template for Under-Represented Task Skill Generation

System Prompt:
An AI agent is consistently failing at {task_type} tasks. Analyze the failures and generate NEW skills.

FAILED TRAJECTORIES: {fail_sections}
SUCCESS TRAJECTORIES (for contrast): {success_section}
EXISTING SKILL TITLES (avoid duplicating): {existing_titles}

Generation Requirements: Generate 1-{max_new_skills} NEW actionable skills that would help avoid these failures. Use skill_ids: {example_ids}

Output Format: Return ONLY a JSON array: `[{"skill_id": "dyn_001", "title": "...", "principle": "...", "when_to_apply": "..."}]`