SHAPING THE GROUP VIA SKILL-PROBED POLICY OPTIMIZATION FOR LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Improving language agents on long-horizon tasks hinges on exploration: an agent can only reinforce the behaviors it manages to discover through interaction. In complex environments with delayed rewards, however, breakthrough behaviors stay buried in the low-probability tail of a combinatorial action space, so passive sampling keeps rediscovering the same failures. This bottleneck is especially acute for group relative reinforcement learning such as Group Relative Policy Optimization (GRPO), which forgoes memory-intensive value networks and instead computes advantages from performance differences within a group of rollouts. When the rollouts fail identically, the advantage vanishes and training suffers gradient starvation. To surface the buried behaviors, we propose SPPO, which turns exploration from passive sampling into active group shaping. For each task, an auxiliary generator produces a current skill that approaches the reachable performance ceiling and a lagged skill from a historical snapshot that provides an intermediate coaching floor. Paired with unguided baseline rollouts, these skills prune the search space and shape a trajectory group with a wide reward spread, restoring non-zero, discriminative advantages. The generator itself is trained from environment feedback, with the current and lagged rollouts placed in one group. Crucially, the actor is optimized via a skill-stripped objective with asymmetric credit attribution: positive-advantage trajectories are double-scored to internalize breakthroughs into the autonomous policy, while negative-advantage trajectories are attributed to the skill to protect the standalone policy from probe-induced errors. On a challenging long-horizon agent benchmark, group shaping lowers the fraction of gradient-starved all-fail groups by about 5 points on average over training relative to unguided sampling, and the resulting agent operates completely autonomously at inference time. Our code is available at `https://anonymous.4open.science/r/SPPO-shaping-the-group`.

1 INTRODUCTION

Autonomous language agents are increasingly expected to solve open-ended, long-horizon tasks across complex software environments, web navigation, and heterogeneous tool orchestration. Standard supervised fine-tuning (SFT) (Ouyang et al., 2022) is fundamentally inadequate for such dynamic domains: cloning static human demonstrations cannot teach an agent how to explore alternative reasoning branches, recover from cascading execution errors, or self-correct upon encountering unexpected environment feedback. Reinforcement learning (RL) (Mnih et al., 2013; Schulman et al., 2017) is therefore indispensable for agent self-evolution, enabling policies to discover robust decision sequences through trial and error. Within agent RL, relative policy optimization—epitomized by Group Relative Policy Optimization (GRPO) (Shao et al., 2024)—has rapidly superseded traditional Actor-Critic architectures like PPO (Schulman et al., 2017). By eliminating memory-heavy critic networks and computing policy advantages purely through intra-group comparisons, GRPO offers superior computational scalability and bypasses the severe value estimation drift common in multi-turn interactions.

However, GRPO's critic-free formulation harbors a critical vulnerability: its optimization engine strictly depends on intra-group performance variance. Without a discriminative contrast between successful and suboptimal attempts, empirical advantages uniformly collapse to zero, causing gradient starvation. While stochastic sampling provides sufficient variance for short-horizon reasoning

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.

(e.g., mathematics), it notoriously fails in long-horizon agent domains. Confronted with vast combinatorial tool spaces and delayed feedback, passive sampling often triggers homogeneous collapse: all rollouts succumb to identical failure loops, rendering scaling the group size N statistically futile.

Crucially, this collapse does not imply an agent lacks task-solving capability. Rather, its latent competence is buried in the low-probability tail as unguided multi-step likelihoods decay exponentially. Can natural-language skills serve as effective probes to elicit these latent behaviors? We argue that skills operate as strategic priors and action-space pruners. Rather than prescribing low-level action tokens, a skill injects an abstract strategic prior—such as an environment decomposition pattern or a fault-recovery heuristic—that redistributes the model's internal attention. By pruning away vast combinatorial branches of irrelevant trial-and-error, the skill confines exploration to a localized, high-value solution corridor. Consequently, the conditional probability of sampling a breakthrough trajectory surges dramatically:

$$P(\tau \mid \text{task}, \text{skill}) \gg P(\tau \mid \text{task}) \tag{1}$$

This probability shift pulls latent, high-reward behaviors out of the tail and into empirical realization, transforming unsearchable spaces into tractable trajectories.

Building on this insight, we propose SPPO (**S**kill-**P**robed **P**olicy **O**ptimization), a framework that deliberately shapes the rollout group into an advantage rich training spectrum. Instead of waiting for unguided search to stumble upon diverse outcomes, an auxiliary generator yields two levels of probes for each task: an up-to-date *current skill* from its latest policy to unlock the actor's reachable performance ceiling, and a *lagged skill* (sampled from a K-step-lagged snapshot) to establish an intermediate coaching floor. By executing rollouts under current guidance, lagged guidance, and the unguided task context, SPPO constructs a structured trajectory triad spanning breakthrough successes, intermediate attempts, and the autonomous baseline. This establishes a *temporal self-contrast* rather than an adversarial dynamic: it proactively encourages a steep return spread within every group, directly revitalizing the advantage signals required for stable policy optimization.

To ensure the agent achieves true autonomy without relying on external skills during test-time deployment, SPPO couples this exploration with a skill-stripped actor optimization scheme. Following rollout collection, the guiding skills are discarded, and trajectories are evaluated under the agent's original, unguided context. To resolve the resulting distribution shift without corrupting the autonomous policy, we introduce an asymmetric credit attribution mechanism: (1) Positive-advantage trajectories are double-scored under both the skill-free and skill-conditioned contexts, internalizing elicited breakthrough successes into the standalone policy while preserving the actor's responsiveness to future probes. (2) Negative-advantage trajectories are scored exclusively under the skill-conditioned context, correctly attributing execution failures to the suboptimal probe that provoked them rather than penalizing the standalone policy for actions it would rarely take on its own. In parallel, the skill generator co-evolves via environment feedback by learning to propose skills that outperform its own historical snapshot.

The primary contributions of this work are three-fold:

- We recast exploration in group-relative agent RL as active group shaping. SPPO rolls out each task under three conditions, and co-evolves the skill generator. Every group retains the return spread required for discriminative advantages.
- We design a skill-stripped actor objective with asymmetric credit assignment. For skill guided trajectories, positive advantage ones are double-scored to internalize elicited successes while keeping the actor responsive to future probes, and negative advantage ones are attributed to the skill, yielding an agent that runs with no skill at inference time.
- On a long-horizon agent benchmark, SPPO eliminates gradient starvation and both ramps faster and settles higher than baselines, and our analysis attributes the gain to the group-shaping mechanism rather than to skills serving as better test-time inputs.

2 Related Work

Reinforcement learning for language agents. Policy optimization is a central mechanism for improving language agents from multi-step interaction and environment feedback. GRPO (Shao et al.,

2024) provides a practical group-relative objective, and recent work applies related optimization to web navigation and knowledge-intensive tasks (Qi et al., 2025; Wei et al., 2025; Sun et al., 2026), as well as coupled improvements of complementary agent roles (Wang et al., 2026). A parallel line enriches the feedback or collection process: rubric scaffolding (Zhou et al., 2025), hindsight experience (Zhang et al., 2026b), and generated experience (Zhai et al., 2025) steer agents toward more useful interactions, while MACPO (Lyu et al., 2025) constructs contrastive positive and negative behavior for preference optimization. CriPO (Xia et al., 2026a) diagnoses unexplored and suppressed rubric criteria and uses on-policy self-distillation to address them without retaining privileged guidance in the policy's rollout distribution. These methods improve the task distribution, supervision, or available guidance. In contrast, SPPO targets the intra-group variance that group-relative optimization requires, actively shaping the rollout group so that rare high-value trajectories enter it rather than improving the reward or supervision.

Skill-based agents and skill-guided behavior. Skills provide a complementary level of agent organization by packaging procedural knowledge into reusable artifacts. Voyager (Wang et al., 2023) demonstrates the value of a reusable skill library, and recent work studies agent-skill architectures (Xu & Yan, 2026), concise language instructions for tool use (Yuan et al., 2025), agent-generated reusable tools (Wölflein et al., 2025), and modular skillpacks for LLM improvement (Li et al., 2026). Concurrent skill-centric RL studies explore unified skill evolution (Shi et al., 2026), in-context skill internalization (Lu et al., 2026), recursive skill augmentation (Xia et al., 2026b), and joint skill internalization and utilization (Zhu et al., 2026). MemRL (Zhang et al., 2026a) instead evolves episodic memory through runtime RL. In these systems, skills, memories, or derived tools are primarily persistent components of execution, retrieval, or capability accumulation. However, SPPO uses skills drawn from a temporal self-contrast, a current skill from the up-to-date generator and a lagged skill from an earlier snapshot of the same generator, as task-specific rollout probes and strips them before actor optimization. The generator improves the probes by learning to outperform its own lagged snapshot from environment rollout outcomes, allowing skills to enrich the training data without becoming a required actor input at inference time.

3 Framework of SPPO

We present SPPO, a skill-probed policy optimization framework for language agents (Figure 1). Group relative optimization needs a spread of returns within each task group to produce non-zero advantages, but normal rollouts in long-horizon tasks often collapse to identical failures. SPPO addresses this by actively shaping the rollout group. A skill generator proposes skills for a task under two conditions: current skills from its up-to-date weights and lagged skills from an earlier snapshot of the same model. The actor is guided by these skills during rollout collection. Once the interaction data have been collected, the skill is stripped. The resulting action responses are then used for a group relative policy update, refined per trajectory so that skill-induced successes are internalized into the skill-free policy while skill-induced failures are attributed to the skill. In parallel, the generator is optimized from the outcomes of the skills it produced, with the current and lagged rollouts grouped together. The actor and generator thus co-evolve: the generator shapes an advantage rich group for the actor and gets rewards, while the actor's improving behavior redefines which skills are useful.

3.1 Problem Setup and Training Roles

Each task x specifies an instruction and an initial environment observation. For a trajectory, let o_t denote the environment observation at interaction step t, and let $\mathcal{H}_t$ denote the interaction history available immediately before the t-th action. The history is written as:

$$\mathcal{H}_t = (o_0, a_1, o_1, \ldots, a_{t-1}, o_t) \tag{2}$$

Let s denote optional skill guidance, which is empty for a normal rollout, and let $\pi_{\theta_{\text{old}}}$ denote the actor policy held fixed during rollout collection. The actor samples an action according to:

$$a_t \sim \pi_{\theta_{\text{old}}}(\cdot \mid x, \mathcal{H}_t, s) \tag{3}$$

A complete interaction trajectory is denoted by $\tau = (a_1, \ldots, a_T)$. The trainable actor policy is π_θ and the skill generator q_ϕ, each has a frozen reference (π_{ref}, q_{ref}) fixed at initialization for the

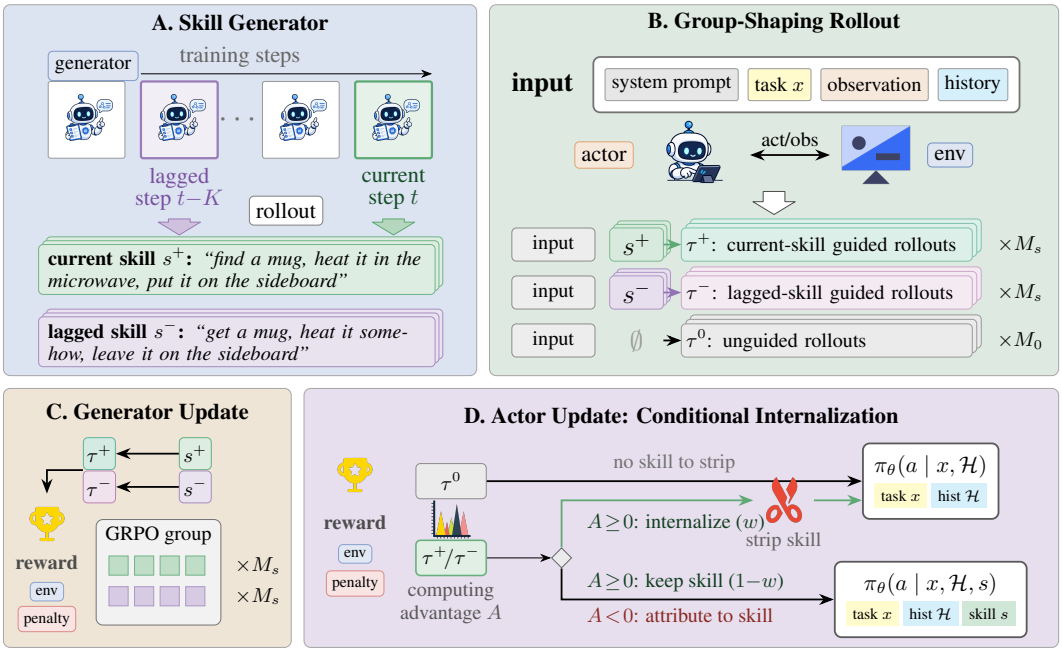


Figure 1: Overview of SPPO. **Top (rollout):** a generator emits a current skill s^+ and a lagged skill s^-. The actor rolls out under three conditions with a wider return spread, restoring non-zero advantages. **Bottom (optimization):** the actor is trained by skill-stripped GRPO with conditional internalization, while the generator learns from the same rewards.

KL regularizers. To form a temporal self-contrast during rollout, we additionally maintain a lagged snapshot $q_{\phi^-} \equiv q_{\phi_{t-K}}$ of the generator, i.e. its weights from K steps earlier, which tracks the generator at a fixed delay rather than staying fixed at initialization.

3.2 Skill-Probed Rollout Construction

For every task, the generator samples two types of skills that differ only in the age of the weights with an identical generation prompt:

$$s^+ \sim q_\phi(\cdot \mid x), \qquad s^- \sim q_{\phi^-}(\cdot \mid x) \tag{4}$$

Here, s^+ is the current skill from the up-to-date generator and s^- the lagged skill from the K-step snapshot q_{ϕ^-}, an earlier, decorrelated counterpart of s^+ from the same generator. Each generated skill is paired with exactly one trajectory: for every task we draw M_s rollouts under each guided condition and M_0 unguided rollouts. Let $\mathbb{P}(\cdot \mid x, s; \pi)$ denote the distribution over trajectories induced by policy π and the environment under task x with optional skill guidance s:

$$\tau^+ \sim \mathbb{P}(\cdot \mid x, s^+; \pi_{\theta_{\text{old}}}), \qquad \tau^- \sim \mathbb{P}(\cdot \mid x, s^-; \pi_{\theta_{\text{old}}}), \qquad \tau^0 \sim \mathbb{P}(\cdot \mid x; \pi_{\theta_{\text{old}}}) \tag{5}$$

Each task therefore contributes a group of $2M_s + M_0$ interaction sequences. Unguided samples are drawn directly from the actor policy, whereas skill-conditioned samples deliberately expose behaviors that receive low probability under that policy and are consequently unlikely to appear without external skill.

Because each skill is one-shot and conditioned only on the task x, not on the actor's in-episode state, and the task schedule is fixed by the dataloader order, we obtain the lagged skills by generating ahead rather than by storing past weights. At step $t-K$, the current generator q_{ϕ_t} pre-generates and caches the skill for the task that will be trained at step t. When that step arrives, the cached skill is K steps stale, realizing $s^- \sim q_{\phi_{t-K}}(\cdot \mid x)$ without a second generator or any weight swap. For the first K steps the cache is empty and the lagged condition falls back to a current sample, which is harmless since the two are nearly identical early in training.

3.3 SKILL-STRIPPED ACTOR OPTIMIZATION

The actor update treats skill-conditioned and unguided interaction data uniformly after skill removal, and is applied at the granularity of individual action step rather than whole trajectories. A trajectory τ_i consists of action responses $a_{i,t}$ for $t = 1, \ldots, T_i$, and over a task group we index responses by the pair (i, t). Each response receives a scalar outcome-based score:

$$R_{i,t} = R_i^{\text{env}} - P_{i,t}^{\text{inv}} \tag{6}$$

Here, $R_i^{\text{env}} \in \{0, R^{\text{win}}\}$ is the trajectory's environment outcome, where R^{win} is the reward for a successful trajectory, broadcast onto each of its actions, and $P_{i,t}^{\text{inv}} = \lambda_{\text{inv}}\, \mathbb{I}[a_{i,t} \text{ is invalid}]$ penalizes the single turn whose action fails to parse into a valid environment action, so the penalty stays local to the turn.

For each action, let $\rho_{i,t}(\theta)$ denote its importance ratio under the original task context:

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(a_{i,t} \mid x, \mathcal{H}_{i,t})}{\pi_{\theta_{\text{old}}}(a_{i,t} \mid x, \mathcal{H}_{i,t})} \tag{7}$$

We form group relative, normalized advantages from the action rewards:

$$A_{i,t} = \frac{R_{i,t} - \mu_x}{\sigma_x + \eta} \tag{8}$$

where μ_x and σ_x are the mean and standard deviation of the action rewards in the group of task x, and η is a small numerical stabilizer. To avoid a length bias, these statistics weight each trajectory equally regardless of its number of responses, so that a long trajectory does not dominate a short one. The exact per-trajectory weighting is given in Appendix C.

Writing $\ell(\rho, A) = \min\big(\rho A, \text{clip}(\rho, 1-\epsilon, 1+\epsilon)A\big)$ for the clipped surrogate, the natural objective sums $\ell\big(\rho_{i,t}(\theta), A_{i,t}\big)$ over all skill-conditioned and unguided responses, as is standard in multi-turn group-based RL, and it also carries the usual GRPO reference-model KL penalty $\beta_{\text{ref}}\, D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big)$ toward the initial actor policy. What is deliberate here is that every response is scored under the skill-free ratio of equation 7, which strips the skill from the policy-gradient context so that this term trains exactly the policy $\pi_\theta(a \mid x, \mathcal{H})$ deployed at inference. Current and lagged skills serve only to shape which action sequences enter the group and how often informative outcomes appear.

Scoring every response under the skill-free context is exactly the internalization we want for unguided responses and for skill-guided successes, folding elicited breakthroughs into the deployable policy. For a losing skill-guided sample, however, it treats the trajectory as if the skill-free policy had produced it, pushing that policy away from a sequence it never generates on its own. We therefore refine the objective per action according to the sign of its advantage and whether it carries a skill. Let $\mathcal{B}$ denote the actor-update batch and $\mathcal{B}_s \subseteq \mathcal{B}$ its skill-conditioned subset. For a response of a skill-conditioned trajectory $\tau_i \in \mathcal{B}_s$, whose skill s_i is retained, define the skill-conditioned importance ratio:

$$\rho_{i,t}^{s}(\theta) = \frac{\pi_\theta(a_{i,t} \mid x, \mathcal{H}_{i,t}, s_i)}{\pi_{\theta_{\text{old}}}(a_{i,t} \mid x, \mathcal{H}_{i,t}, s_i)} \tag{9}$$

The refined per-action objective is:

$$\mathcal{L}_{\text{PG}}^{\text{int}}(\theta) = -\mathbb{E}\left[\frac{1}{\sum_i T_i} \sum_i \sum_{t=1}^{T_i} g_{i,t}(\theta)\right], \tag{10}$$

where the per-action term $g_{i,t}(\theta)$ is

$$g_{i,t}(\theta) = \begin{cases} w\, \ell\big(\rho_{i,t}(\theta), A_{i,t}\big) + (1-w)\, \ell\big(\rho_{i,t}^{s}(\theta), A_{i,t}\big), & \tau_i \in \mathcal{B}_s,\ A_{i,t} \geq 0, \\ \ell\big(\rho_{i,t}^{s}(\theta), A_{i,t}\big), & \tau_i \in \mathcal{B}_s,\ A_{i,t} < 0, \\ \ell\big(\rho_{i,t}(\theta), A_{i,t}\big), & \tau_i \notin \mathcal{B}_s. \end{cases} \tag{11}$$

Three cases are distinguished, mirroring the actor-update branches in Figure 1. A skill-guided action with negative advantage is scored only under the skill-conditioned context, so the resulting suppression is attributed to the skill that steered the actor into a low-return action, rather than penalizing the

skill-free policy for behavior it did not generate. A skill-guided action with non-negative advantage is *double-scored*: it contributes a term under the skill-free context, which internalizes the success into the deployable policy, weighted by w, and a term under the skill-conditioned context, which keeps the actor following the skill that discovered it, weighted by $1 - w$. Responses of unguided trajectories keep the single skill-free term $\ell\big(\rho_{i,t}(\theta), A_{i,t}\big)$ as normal GRPO does.

3.4 Skill Generator Optimization

The generator is trained from a trajectory-level signal per skill-guided rollout in $\mathcal{B}_s$. A trajectory τ_j^+/τ_j^- is scored by its environment outcome with an invalid action penalty, $\widehat{R}_j = R_j^{\text{env}} - \lambda_{\text{inv}}^q \, r_j^{\text{inv}}$, where r^{inv} is the fraction of invalid actions in the trajectory. A skill that fails to parse is instead assigned a fixed format penalty $\widehat{R}_j = -\lambda_{\text{fmt}}^q$.

The two conditions are combined into a single self-contrast at the advantage stage. For each task, the M_s current-skill and M_s lagged-skill rollouts are pooled into one group, so the group baseline is the mean return of both conditions and the normalized advantage of a current-skill rollout measures how much it improves on the actor:

$$\widetilde{A}_j = \frac{\widehat{R}_j - \widehat{\mu}_x}{\widehat{\sigma}_x + \eta} \tag{12}$$

where $\widehat{\mu}_x$ and $\widehat{\sigma}_x$ are the mean and standard deviation of the generator's rewards for task x. The lagged-skill rollouts are produced by K-step-stale weights and are off-policy for the generator being updated. We therefore keep them only as part of this baseline: after computing equation 12, we set the advantage of every lagged-skill rollout to zero, so no policy-gradient signal flows through the stale samples and only the M_s current-skill rollouts drive the generator update.

For a generated skill response s_j with token-level ratio $\widetilde{r}_{j,k}(\phi)$ and normalized generator advantage $\widetilde{A}_j$ from equation 12, the generator uses the same GRPO-style clipped objective:

$$\mathcal{L}_{\text{gen}}(\phi) = -\mathbb{E}_j \left[\frac{1}{|s_j|} \sum_{k=1}^{|s_j|} \min \left(\widetilde{r}_{j,k}(\phi)\widetilde{A}_j, \text{clip}\left(\widetilde{r}_{j,k}(\phi), 1-\widetilde{\epsilon}, 1+\widetilde{\epsilon}\right)\widetilde{A}_j \right) \right] \tag{13}$$

The generator is also regularized toward its frozen SFT reference through a conventional token-level KL term:

$$\mathcal{L}_{\text{gen,total}}(\phi) = \mathcal{L}_{\text{gen}}(\phi) + \beta_{\text{ref}}^q \mathcal{L}_{\text{ref}}^q(\phi) \tag{14}$$

Here, $\mathcal{L}_{\text{ref}}^q$ measures the KL divergence between q_ϕ and q_{ref} on generated skills.

Together, the actor and generator form a coupled optimization loop. The actor consolidates informative behavior under the original task context, while the generator continually improves, relative to its own lagged snapshot, the skill used to discover such behavior.

4 Experiments

We evaluate SPPO on ALFWorld (Shridhar et al., 2021), a long-horizon embodied agent benchmark, with no skill at inference time.

4.1 Experimental Setup

Dataset. ALFWorld is a text-based embodied environment in which an agent completes household goals through multi-turn interaction, spanning six task categories: Pick & Place (Pick), Examine in Light (Look), Clean & Place (Clean), Heat & Place (Heat), Cool & Place (Cool), and Pick Two & Place (Pick2). We train on the ALFWorld training split and hold out a disjoint set of games for validation. The exact partition into RL-training, generator-SFT, and validation games is given in Appendix A. All settings follow GiGPO (Feng et al., 2025). Both the generator's skill sampling and the actor's rollouts use temperature 1.0 during training, and the actor is evaluated at temperature 0.4.

Type	Method	ALFWorld						
		Pick	Look	Clean	Heat	Cool	Pick2	All
Closed-Source Model								
Prompting	GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0
Prompting	Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3
Qwen2.5-1.5B-Instruct								
Prompting	Qwen2.5	5.9	5.5	3.3	9.7	4.2	0.0	4.1
Prompting	ReAct	17.4	20.5	15.7	6.2	7.7	2.0	12.8
Prompting	Reflexion	35.3	22.2	21.7	13.6	19.4	3.7	21.8
RL Training	PPO (with critic)	64.8	40.5	57.1	60.6	46.4	47.4	54.4
RL Training	RLOO	88.3	52.8	71.0	62.8	66.4	56.9	69.7
RL Training	GRPO	85.3	53.7	84.5	78.2	59.7	53.5	72.8
RL Training	$\text{GiGPO}_{\text{w/ std}}$	94.4	67.5	**94.8**	94.4	79.8	76.4	86.7
RL Training	$\text{GiGPO}_{\text{w/o std}}$	96.0	76.5	91.8	91.3	71.7	**79.5**	86.1
RL Training	**SPPO (ours)**	**97.3**	**91.7**	92.0	**100.0**	**87.5**	76.2	**90.6**
Qwen2.5-7B-Instruct								
Prompting	Qwen2.5	33.4	21.6	19.3	6.9	2.8	3.2	14.8
Prompting	ReAct	48.5	35.4	34.3	13.2	18.2	17.6	31.2
Prompting	Reflexion	62.0	41.6	44.9	30.9	36.3	23.8	42.7
RL Training	PPO (with critic)	92.3	64.0	92.5	89.5	80.3	68.8	80.4
RL Training	RLOO	87.6	78.2	87.3	81.3	71.9	48.9	75.5
RL Training	GRPO	90.8	66.1	89.3	74.7	72.5	64.7	77.6
RL Training	$\text{GiGPO}_{\text{w/ std}}$	97.7	82.7	98.8	83.7	89.3	79.2	90.8
RL Training	$\text{GiGPO}_{\text{w/o std}}$	91.8	88.6	95.9	90.2	86.5	**85.2**	90.2
RL Training	**SPPO (ours)**	**100.0**	**90.0**	**100.0**	**100.0**	**96.0**	82.1	**94.5**

Table 1: Success rate (%) on ALFWorld across six task categories and the overall average. Baseline results are taken from Feng et al. (2025). The actor of SPPO is evaluated alone under the original prompt, with no skill at inference time. **Bold** marks the best result in each column.

Base models and baselines. We use Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct (Team, 2024) as base actors. Following the setup of prior agent-RL work, we compare against three families of baselines: (1) closed-source models (GPT-4o (Hurst et al., 2024), Gemini-2.5-Pro (Comanici et al., 2025)) prompted directly; (2) prompting agents that update no parameters (the base Qwen2.5 model, ReAct (Yao et al., 2023), and Reflexion (Shinn et al., 2023)); and (3) RL post-training methods, including the actor–critic PPO (Schulman et al., 2017) and the critic-free group-based methods RLOO (Ahmadian et al., 2024), GRPO (Shao et al., 2024), and GiGPO (Feng et al., 2025). Baseline numbers are taken from Feng et al. (2025) under matched settings.

Training and rollout configuration. SPPO builds on a GRPO-style update. For each task we collect $M_0 = 8$ unguided rollouts together with $M_s = 4$ current-skill and $M_s = 4$ lagged-skill rollouts, where the lagged skills come from a $K = 10$-step-stale snapshot of the generator. Positive-advantage skill-guided responses are double-scored with weight $w = 0.5$ on the skill-free objective. The generator is initialized by SFT and co-optimized with the actor, and sees only the task and the initial observation, never the walkthrough. All remaining hyperparameters and the generator SFT data are given in Appendix A and Appendix B.

4.2 Main Results

Table 1 reports success rates on ALFWorld for both model scales. Among the baselines, RL post-training substantially outperforms prompting agents and closed-source models, and the group-based reinforcement learning methods (RLOO, GRPO, GiGPO) are the strongest, with GiGPO leading the overall average at both scales. We compare SPPO against these methods under the same training protocol, evaluating the actor with no skill. SPPO reaches 90.6% average success rate at the 1.5B scale and 94.5% at the 7B scale, outperforming all baselines in both settings.

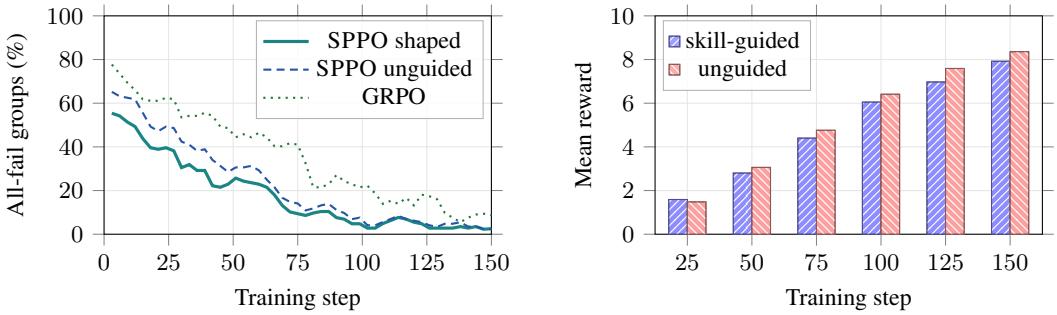


Figure 2: Group shaping over training (Qwen2.5-1.5B). **(a)** Fraction of *all-fail* groups (9-step moving average). **(b)** Mean episode reward of the skill-guided (current + lagged) versus the unguided rollouts, over 25-step blocks.

Faster ramp and higher ceiling. Figure 3 tracks validation success rate over training for SPPO, GiGPO, and GRPO on the 1.5B actor. SPPO ramps up faster and settles higher throughout. The faster ramp follows directly from group shaping: when passive sampling would leave a task with no successful rollout, the injected skills break the outcome homogeneity that would otherwise zero out the advantage, so the actor receives gradient from the outset rather than waiting for a lucky success as GiGPO and GRPO do. Meanwhile, double-scoring folds each positive-advantage skill trajectory back onto the skill-free objective (w=0.5). The deployable actor therefore internalizes successful behaviors it would almost never sample on its own, and continues improving past the point where unguided exploration stalls.

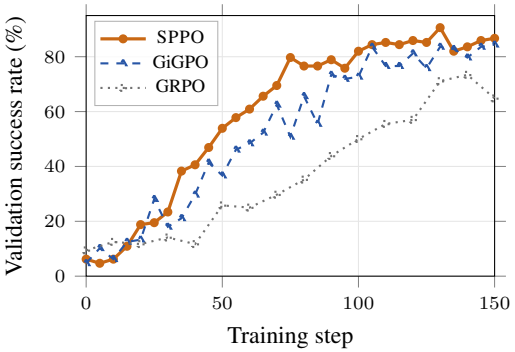


Figure 3: Validation success rate (%) over training on ALFWorld (Qwen2.5-1.5B-Instruct).

Group shaping restores non-zero advantages. The engine of any GRPO-style update is the within-group reward spread: the per-task standard deviation σ_x is the denominator that turns raw scores into advantages, so a group whose rollouts are all-fail or all-pass yields a degenerate σ_x and no usable gradient. Figure 2(a) reports the fraction of *all-fail* groups over training. Unguided sampling alone collapses a large share of groups to all-fail, especially early: near the start of training $\sim$65% of tasks have every unguided rollout failing. The full shaped group, which adds current- and lagged-skill rollouts, brings this down to $\sim$55% and keeps the all-fail share consistently lower through the early ramp (about a 5-point average reduction over training). Both remain well below the external GRPO baseline, which starts near $\sim$80%. Notably, this reduction is not explained by the skill-guided rollouts earning more reward. Figure 2(b) shows their mean episode reward tracking—and often sitting slightly below—the unguided rollouts throughout training. The skills help instead by diversifying outcomes within otherwise-homogeneous groups: a hard task on which every unguided rollout fails occasionally receives a breakthrough skill-guided trajectory, converting a collapsed group into a discriminative one. This mechanism coincides with SPPO's faster early ramp in Figure 3. The slightly-lower skill reward also helps at the opposite extreme. On easy tasks, it pulls the group's reward baseline down and sharpens the positive advantage.

4.3 Current and Lagged Skills: A Difficulty-Dependent Separation

A natural concern is whether the current and lagged skills are redundant, or whether the K-step lag makes one systematically weaker. Figure 4 reveals that the answer depends on task difficulty. Each panel plots the *any-win rate*—the fraction of tasks in a tier on which a skill condition lands at least one winning trajectory, count-weighted over 25-step blocks. We split tasks per step by the unguided pass rate p over the $M_0 = 8$ no-skill rollouts—easy ($p > 0.5$), medium ($0 < p \leq 0.5$),

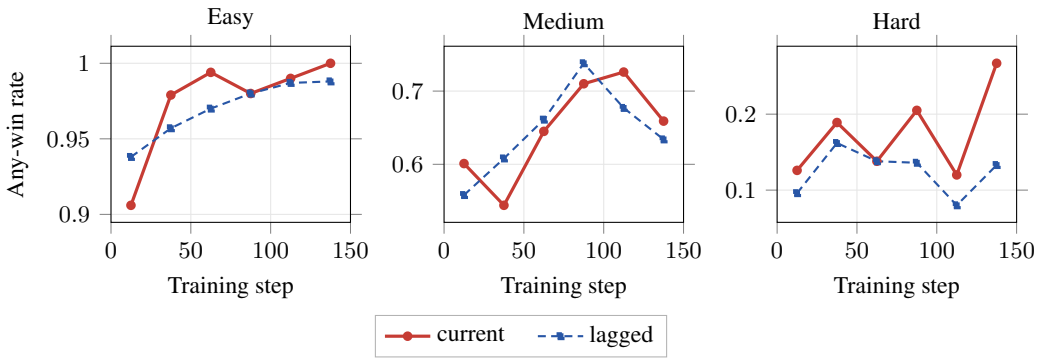


Figure 4: Any-win rate by difficulty tier over training (SPPO, ALFWorld, Qwen2.5-1.5B). Current and lagged skills coincide on easy and medium tasks but separate on hard tasks.

and hard ($p = 0$). The current and lagged skills are interchangeable on easy and medium tasks, but they separate clearly on hard tasks: the current skill rescues a winning trajectory on 15.5% of hard tasks versus 12.4% for the lagged skill (a $\sim 25\%$ higher rescue rate), and the gap widens over training. Because the separation appears only in the hard tier—exactly the tasks that gate learning—it reflects genuine skill content rather than a uniform prompt-perturbation effect: the current skill is the stronger probe where it matters, while the lagged skill remains an equally strong but decorrelated fallback everywhere else. This is what sustains a real current-over-lagged signal for the generator's temporal self-contrast.

4.4 Ablation Studies

We ablate the two components that make SPPO more than a skill-augmented GRPO: the lagged skill snapshot of Section 3.2 and the skill-conditioned credit assignment of Section 3.3. Table 2 reports the test success rate on the 1.5B actor. Every variant is trained under the identical protocol and evaluated as the skill-free actor.

Variant	Test SR (%)	Δ
SPPO (full)	**90.6**	—
w/o lagged skills	85.9	−4.7
w/o double-scoring	63.3	−27.3

Table 2: Component ablation on ALFWorld.

Lagged skills. The *w/o lagged skills* variant replaces the four lagged skills with four additional current skills, holding the group size at 16 so the only change is the removal of the K-step-stale snapshot. It reaches 85.9%, 4.7 points below the full method, so the lagged condition contributes beyond simply drawing more current-skill rollouts: the temporal self-contrast it provides is what the generator's advantage signal is built on.

Double-scoring. Turning off double-scoring is the most damaging ablation. A skill trajectory with positive advantage is then scored only under the skill-free context ($w{=}1$), so the actor is still credited for the outcome but is no longer reinforced along the skill-following path that produced it. Without that term the actor gradually stops acting on the skills, the skill-guided rollouts stop surfacing breakthroughs, and the group shaping that supplies gradient decays with them. This matches the role of the $(1{-}w)$ term in Section 3.3, which preserves the actor's responsiveness to future probes.

5 Conclusion

We presented SPPO, a group-relative agent RL framework that uses generated skills as rollout-time probes, not inference-time crutches. Passive sampling can collapse a group's rollouts to one outcome, eliminating GRPO's within-group variance and advantage signal. SPPO mixes current-, lagged-skill, and unguided rollouts, and uses a skill-stripped objective to fold elicited successes into a skill-free inference policy under the original prompt. On ALFWorld, it ramps faster and settles higher, outperforming all baselines, with gains tied to the mechanism rather than test-time skills. Future work includes broader environments and larger actors.

AI USE STATEMENT

We used large language models as general-purpose assistants at several stages of this work: to aid and polish the writing, to help retrieve and discover related work, to draft portions of the paper, and to generate synthetic data (the skill-generator SFT targets described in Section 4.1). All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final paper.

ETHICS STATEMENT

This paper studies reinforcement learning for language agents. Our experiments involve no human subjects, no personally identifiable information, and no sensitive or private data, and we release no new dataset. The method improves how an agent explores and internalizes successful behaviors within a simulated benchmark, and we are not aware of direct harmful applications beyond the general dual-use considerations that accompany any advance in autonomous agents.

REPRODUCIBILITY STATEMENT

We provide our full source code, all hyperparameters (Section 4.1 and Appendix A), the random seed, and the runtime environment (Appendix A). The ALFWorld dataset is publicly available for download. Together these ensure that our results are reproducible.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 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 (ACL)*, 2024.

Gheorghe Comanici et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

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

Zhuo Li, DU Guodong, Zesheng Shi, Weiyang Guo, Weijun Yao, Yuan Zhou, Jiabo Zhang, and Jing Li. Skill weaving: Efficient llm improvement via modular skillpacks. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 40000–40023, 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.

Yougang Lyu, Lingyong Yan, Zihan Wang, Dawei Yin, Pengjie Ren, Maarten de Rijke, and Zhaochun Ren. Macpo: weak-to-strong alignment via multi-agent contrastive preference optimization. In *International Conference on Learning Representations*, volume 2025, pp. 63022–63048, 2025.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

OpenAI. Introducing GPT-5.5, 2026. URL https://openai.com/index/introducing-gpt-5-5/.

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

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 79791–79821, 2025.

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

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.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03768.

Xueqiao Sun, Xiao Liu, Bowen Lv, Hanchen Zhang, Bohao Jing, Zehan Qi, Yifan Xu, Yuxiao Dong, and Jie Tang. Karl: Reinforcement learning for llm agents on multi-turn knowledge-intensive agentic tasks. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 47539–47558, 2026.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

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.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Co-evolving llm coder and unit tester via reinforcement learning. *Advances in Neural Information Processing Systems*, 38:143630–143664, 2026.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7920–7939, 2025.

Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelovic, and Jakob Nikolas Kather. Llm agents making agent tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26092–26130, 2025.

Mingxuan Xia, Yuhang Yang, Chao Ye, Shuai Zhu, Shenzhi Yang, Guangcheng Zhu, Yuhang Zhang, Cheng Peng, Haobo Wang, and Siqing Wang. Enhancing rubric-based rl via self-distillation. *arXiv preprint arXiv:2607.18082*, 2026a.

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*, 2026b.

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

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

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. Easytool: Enhancing llm-based agents with concise tool instruction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 951–972, 2025.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, et al. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026a.

Wenjian Zhang, Kongcheng Zhang, Jiaxin Qi, Baisheng Lai, and Jianqiang Huang. Experience is the best teacher: Motivating effective exploration in reinforcement learning for llms. *arXiv preprint arXiv:2603.20046*, 2026b.

Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Kongcheng Zhang, Jiale Zhao, Jingwen Yang, Yihe Zhou, Jianwei Lv, Tongya Zheng, et al. Breaking the exploration bottleneck: Rubric-scaffolded reinforcement learning for general llm reasoning. *arXiv preprint arXiv:2508.16949*, 2025.

Jiapeng Zhu, Jianxiang Yu, Yibo Zhao, Chengcheng Han, Qi Gu, Xunliang Cai, Xiang Li, and Weining Qian. Skill0. 5: Joint skill internalization and utilization for out-of-distribution generalization in agentic reinforcement learning. *arXiv preprint arXiv:2605.28424*, 2026.

A Experimental Details

Compute environment. All experiments run on two nodes, each equipped with eight NVIDIA A800-SXM4-80GB GPUs and 1 TB of system memory (NVIDIA driver 535.161.07, CUDA 12.2), under Linux 5.10.0-1.0.0.35.

Random seed. We use a fixed random seed of 0 for both training and evaluation. The 128 evaluation workers draw their games from this base seed, with worker i using seed $1000 + i$, matching GiGPO's evaluation draw.

Dataset split and training schedule. The $3{,}553$-game ALFWorld training split is partitioned into three mutually disjoint subsets: $2{,}560$ games for actor RL training, 708 games for skill-generator SFT, and 285 games held out for validation. All reported results use the standard in-distribution test split (`valid_seen`), which is disjoint from the entire training split. Following Feng et al. (2025), we sample 128 test games from its 140 games under GiGPO's random draw with matched seeds, so that both methods are scored on the same games, and report per-category and overall success rate (%). Validation on the held-out 285-game set is run every 5 steps, and we take the checkpoint with the best validation success rate reached during training. Matching GiGPO, each episode runs for up to 50 environment steps, the interaction history length is 2, and the actor and generator are trained for 150 optimization steps with a training batch of 16 games per step. Following GiGPO, the actor learning rate is $1\mathrm{e}{-6}$ and the KL coefficient toward the frozen reference actor is 0.01.

Reward and penalty coefficients. The environment returns $R^{\mathrm{win}} = 10$ for a successful trajectory and 0 otherwise. The actor's invalid-action penalty coefficient is $\lambda_{\mathrm{inv}} = 0.1$. For the generator, the invalid-action coefficient is $\lambda^{q}_{\mathrm{inv}} = 5$ and a skill that fails to parse receives a format penalty $\lambda^{q}_{\mathrm{fmt}} = 10$.

B SKILL-GENERATOR SFT PROMPT AND EXAMPLES

The current-skill generator is initialized by supervised fine-tuning on skills written by GPT-5.5 (OpenAI, 2026). For each ALFWorld training game, GPT-5.5 is shown the task and its reference walkthrough and asked to distill a single high-level strategy under the prompt in Table 3. The generated skill is used as the SFT target. The prompt deliberately forbids restating the walkthrough as fact and instead asks for a search order plus a processing plan, so the target reflects only what an actor could act on before exploring the room. At RL time and at inference the generator conditions on the task and the initial observation alone and never sees the walkthrough. Two representative targets are shown in Table 4.

GPT-5.5 skill-generation prompt (ALFWorld).

```
You are generating one strategy skill for an ALFWorld task.
The skill is handed to an actor before it has explored the room, so it must be useful
without asserting anything the actor could not have worked out for itself.

The reference walkthrough is privileged information for you only. Do not restate it as
fact. Convert it into a search order plus a processing plan.

Rules:
– Generate exactly one good skill, one sentence, at most 35 words.
– Shape it as: where to look first, then any required processing step, then the destination.
– Give two or three candidate receptacles to search, in order, with the one the
  walkthrough actually uses first. Never claim the object IS in a particular place.
– Name the receptacle that performs a required transformation (cooling, heating, cleaning)
  whenever the task needs one.
– Use receptacle names without their numeric index: write "dresser", not "dresser 1".
  Never replace a receptacle name with a placeholder such as "its current surface",
  "its current location", or "storage".
– Do not mention the walkthrough, step numbers, or that you were shown a solution.
– Return JSON only in this exact format:
{"skill": "your skill"}

Task:
{question}

Reference walkthrough:
{reference_answer}
```

Table 3: Prompt given to GPT-5.5 to produce the skill-generator SFT targets on ALFWorld. `{question}` and `{reference_answer}` are filled with each training game's task and reference walkthrough.

C LENGTH-UNBIASED GROUP NORMALIZATION

Section 3.3 normalizes each action-response score $R_{i,t}$ by group statistics μ_x and σ_x that weight every trajectory equally, regardless of how many responses (turns) it contains. Weighting all response rows equally instead would let a long trajectory contribute proportionally more rows to the statistics, so that, for example, a 50-turn failure could outweigh a 12-turn success.

Consider the group of trajectories collected for a task x, indexed by i, where trajectory i has T_i responses with scores $\{R_{i,t}\}_{t=1}^{T_i}$, and let n be the number of trajectories in the group. We assign each response the weight $\omega_{i,t} = 1/T_i$, so that every trajectory carries total weight 1 and $\sum_{i,t} \omega_{i,t} = n$. The group mean and bias-corrected variance are

$$\mu_x = \frac{1}{n} \sum_i \frac{1}{T_i} \sum_{t=1}^{T_i} R_{i,t}, \qquad \sigma_x^2 = \frac{n}{n-1} \cdot \frac{\sum_{i,t} \omega_{i,t} \left(R_{i,t} - \mu_x\right)^2}{\sum_{i,t} \omega_{i,t}}, \tag{15}$$

Representative GPT-5.5 skill targets (ALFWorld).

Task: put some remotecontrol on coffeetable.
Reference walkthrough:
  go to dresser 1
  take remotecontrol 2 from dresser 1
  go to coffeetable 1
  move remotecontrol 2 to coffeetable 1
Skill: Search the dresser, then the sofa or TV stand for a remotecontrol; once found, take it directly to the coffeetable.

Task: put a hot plate in cabinet.
Reference walkthrough:
  go to diningtable 1
  take plate 3 from diningtable 1
  go to microwave 1
  heat plate 3 with microwave 1
  go to cabinet 5
  ...
Skill: Check the diningtable first, then the countertop or side table for a plate; heat it in the microwave, then open the cabinet and place it inside.

Table 4: Two skill-generator SFT targets produced by the prompt in Table 3: a simple pick-and-place task and one requiring a heating step. Each skill gives an ordered search over receptacle types, names the transformation receptacle when needed, and never asserts the object's true location.

where the factor $n/(n-1)$ applies the standard correction over the n independent trajectories rather than the response rows. The advantage of Eq. equation 8 is then $A_{i,t} = (R_{i,t} - \mu_x)/(\sigma_x + \eta)$.

Two boundary cases are handled explicitly. If the group contains a single trajectory ($n = 1$), we set $\mu_x = 0$ and $\sigma_x = 1$, leaving the score unnormalized. If every trajectory in the group shares the same outcome, i.e. $\max_t R_{i,t}$ is identical across all i (e.g. an all-success or all-failure group), there is no comparative signal and the only variance comes from the sparse invalid-action penalties on individual responses. We then set $\sigma_x = 1$ so that these penalties stay in their raw reward units instead of being amplified by a near-zero standard deviation. We read the trajectory outcome as $\max_t R_{i,t}$ because the penalty only ever subtracts, so the maximum recovers the unpenalized outcome. All other (non-degenerate) groups use σ_x as defined in Eq. equation 15.