PROGRESSIVE AGENT SKILL GENERATION VIA REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent large language model agents often use external skills as modular procedural units that condition inference and improve complex task solving. Thus, automatically generating high-quality skills from documents or experience has become an important problem. Existing skill generation methods largely rely on heuristics or pipeline-style consolidation, which must be specially designed for different evidence sources. In contrast, learning-based approaches offer a more unified way to model skill generation across heterogeneous sources. However, learning-based skill generation remains challenging because skills lack a natural supervision signal based on relevance or correctness; their value can largely be determined only by whether they improve the behavior of the agent on downstream tasks. To address this challenge, we propose Skill-α, a reinforcement learning method that learns a unified policy for progressive skill generation. Specifically, we construct each skill by repeatedly applying the learned policy to successive source evidence and introduce a novel rollback reward that evaluates each edit by comparing downstream execution under the original and edited skills on an anchored query. Extensive experiments show that Skill-α generates more effective skills than methods based on heuristics or pipelines in both document-to-skill and experience-to-skill settings. Under the main GPT-4o worker, Skill-α improves average downstream success rates over the strongest skill-generation baseline by 3.1 points on CL-Bench and 6.7 points on tau2-bench. Further ablations and analysis validate the importance of rollback reward and progressive generation.

1 INTRODUCTION

Recent large language model (LLM) agents (Luo et al., 2025; Wang et al., 2024) are usually expected to solve tasks that require multi-step reasoning (Plaat et al., 2025), long-horizon planning (Huang et al., 2024), and reliable tool use (Shen, 2024). A practical way to extend these agents is to provide external skills that condition inference and guide how the agent decomposes tasks, invokes tools, and checks intermediate results (Wang et al., 2025a; 2023; Shen et al., 2026). As skills become reusable modules for shaping agent behavior without retraining the model, automatically generating high-quality skills from documents or experience becomes an important problem (Anthropic, 2025).

Existing skill generation methods largely rely on heuristics, prompting, or pipeline-style consolidation. Document-to-skill methods compress documents, rules, or task descriptions into procedural instructions (Si et al., 2026; Yang et al., 2026), while experience-to-skill methods distill successful or failed executions into reusable guidance (Ni et al., 2026; Mi et al., 2026). Although effective in specific settings, these methods usually require separate designs for different evidence sources and provide limited guidance on how each piece of evidence should change the skill being written. Thus, the central challenge is not only to produce skill text, but to learn a unified skill generation process.

However, learning to generate skills is difficult because skills do not have a natural supervision signal based on relevance or correctness. Unlike mathematical reasoning or web search, where outputs can often be evaluated against explicit factual answers or execution results (Guo et al., 2025; Jin et al., 2025), the quality of a skill lacks a similarly direct supervision signal and is often assessed through its effect on downstream task performance. A fluent skill may still be redundant, over-specific, or misleading, while a compact edit may substantially improve the worker agent's behavior (Huang

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.

et al., 2026; Zhou et al., 2026). This makes execution-grounded credit assignment the key obstacle to learning-based skill generation.

To address this issue, we propose Skill-α, a reinforcement learning framework that learns a unified policy for progressive skill generation. The learned policy is repeatedly applied over source evidence, with each edited skill serving as the state for the next construction step. Specifically, given a skill draft and new evidence, the generator learns whether to add new procedures, revise imprecise rules, merge redundant guidance, remove harmful or over-specific content, or leave the skill unchanged. We introduce rollback reward as the key training signal. After a candidate edit is applied, the original and edited skills are evaluated on the same evidence-related anchored query and compared by a task-level verifier, so that the reward reflects the local effect of changing the skill condition. In this way, Skill-α turns skill generation pipelines into a unified learning problem over local skill edits, enabling the skill generator to be trained with reinforcement learning.

We evaluate Skill-α in both document- and experience-to-skill settings. Under the GPT-4o worker, Skill-α improves average downstream success rates over the strongest baselines by 3.1 points on CL-Bench (Dou et al., 2026) and 6.7 points on tau2-bench (Barres et al., 2026). On the fully unseen BFCL multi-turn benchmark (Patil et al., 2023), Skill-α is the only skill-generation method that outperforms the no-skill worker under different backbones. Ablations further validate the importance of rollback reward and progressive generation.

Overall, our contributions are threefold. First, we formulate agent skill generation as a progressive decision-making problem, unifying document-to-skill and experience-to-skill generation under a single learning framework. Second, we introduce rollback reward, which provides execution-grounded credit assignment for local skill edits. Third, we empirically show that Skill-α produces effective skills from heterogeneous evidence sources, with ablations and analyses validating the roles of rollback reward and progressive generation.

2 Related Work

Agent Skills. Agent skills are reusable procedural modules that condition the inference process of an agent and reshape its behavior on downstream tasks (Anthropic, 2025). Recent work has studied skills from multiple perspectives, including skill evolution (Shen et al., 2026; Xia et al., 2026; Ouyang et al., 2026), selection (Zeng et al., 2026; Liu et al., 2026; Zheng et al., 2026), utilization (Wang et al., 2023; 2025a), and internalization (Wang et al., 2026b). These studies show that skills are functional units that can guide planning, tool use, and long-horizon task solving (Wang et al., 2026c;d; Xu & Nie, 2026). Because of this role, more attention has shifted toward skill generation and acquisition (Si et al., 2026; Yang et al., 2026; Ni et al., 2026; Mi et al., 2026) and benchmark evaluation (Huang et al., 2026; Zhou et al., 2026; Li et al., 2026). Skill-α follows this direction by treating automatic external-skill generation as the target of learning.

Skill Generation and Acquisition. Skill generation and acquisition focus on producing reusable skill modules from evidence sources such as documents, task contexts, experience, and trajectories. Early work explored converting verifier feedback into reusable procedural memory, guidelines or workflows, including ExpeL (Zhao et al., 2024), Agent Workflow Memory (Wang et al., 2025c), AutoGuide (Fu et al., 2024) and AutoFlow (Li et al., 2024). Related work has also studied automatic skill construction through prompt-based skill creation (Anthropic, 2026). More recent pipeline-based skill generation systems generally fall into two categories. For document-to-skill settings, Ctx2Skill (Si et al., 2026) and AutoSkill (Yang et al., 2026) aim to compress task documents, rules, or contexts into executable procedural instructions; for experience-to-skill settings, Trace2Skill (Ni et al., 2026), SkillX (Wang et al., 2026a), and SkillPro (Mi et al., 2026) distill trajectories, feedback, or hierarchical experience into reusable skill knowledge. These methods automate skill acquisition but rely on heuristic or pipeline-style consolidation tailored to specific evidence types. In contrast, Skill-α learns one unified editing policy and repeatedly applies it to construct skills from heterogeneous evidence sources.

Reinforcement Learning for External Module Optimization. Reinforcement learning (RL) has become a common approach for optimizing LLM agents to enhance slow-thinking reasoning capabilities (Schulman et al., 2017; Guo et al., 2025; Yu et al., 2025; Shen et al., 2025; Cheng et al., 2025). Beyond optimizing the acting policy itself, recent work also applies RL to external module management. In tool-use and web-search settings, RL is used to improve how agents invoke external

tools (Li et al., 2025; Singh et al., 2025), interact with environments (Cheng et al., 2025; Zhao et al., 2026), and use search results (Qi et al., 2024; Jin et al., 2025) to complete complex tasks. Memory-R1 (Yan et al., 2025) and Mem-α (Wang et al., 2025b) learn memory operations through downstream question-answering rewards. In contrast, Skill-α generates transferable skills from documents and experience. Thus, we evaluate local edits by comparing worker execution under the original and edited skills, directly supervising changes. Recent works like Skill-R1 (Vishe et al., 2026) recurrently optimize a task-specific skill using verified rollout histories for each task instance, whereas Skill-α learns a unified evidence-to-skill generation policy that is applied to new tasks without task-specific parameter optimization.

3 PRELIMINARIES

LLM Agent. We model a worker agent as a fixed policy π_ψ that interacts with an environment to solve a task. Given a task instruction or context q, the worker produces an execution trajectory $\tau = (o_1, a_1, \ldots, o_H, a_H)$, where H is the trajectory length and o_h and a_h denote the observation and action at worker step h. An external skill z is inserted into the worker context. The skill-conditioned worker induces the trajectory distribution $\pi_\psi(\tau \mid z, q)$. In this work, π_ψ is not the training target. We instead train a skill-editing policy π_ϕ that samples a local edit action A_t conditioned on the current skill state and source evidence at each step. Applying these actions progressively produces the skill z that conditions the fixed worker on downstream tasks.

Group Relative Policy Optimization. Group Relative Policy Optimization (GRPO) (Guo et al., 2025) is used to optimize the skill generator. For an input X, GRPO samples a group of edit actions $\{A_i\}_{i=1}^G$ from the old policy $\pi_{\phi_{\text{old}}}$ and assigns each action a scalar reward r_i. The group-relative advantage is $\widehat{\mathsf{A}}_i = \frac{r_i - \text{mean}(\{r_1,\ldots,r_G\})}{\text{std}(\{r_1,\ldots,r_G\})}$. The policy is then optimized with the objective

$$\mathcal{J}_{\text{GRPO}}(\phi) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\rho_i(\phi)\widehat{\mathsf{A}}_i, \text{clip}(\rho_i(\phi), 1-\epsilon, 1+\epsilon)\widehat{\mathsf{A}}_i\right) - \beta D_{\text{KL}}(\pi_\phi \| \pi_{\text{ref}})\right)\right], \quad (1)$$

where $\rho_i(\phi) = \frac{\pi_\phi(A_i|X)}{\pi_{\phi_{\text{old}}}(A_i|X)}$, π_{ref} is the reference policy, and ϵ, β are hyperparameters.

Skill Generation Formalization. For each task family $\mathcal{M}$, let $p^\star_{\mathcal{M}}$ denote a conceptual query-conditioned joint distribution over evidence and high-quality teacher behavior. Let $x_{1:T}$ denote a sequence of T evidence units (*e.g.*, document segments or batches of execution traces). For a source query q', we sample $x_{1:T} \sim p^\star_{\mathcal{M}}(\cdot \mid q')$ from its evidence marginal, while $p^\star_{\mathcal{M}}(\tau \mid q')$ denotes the corresponding teacher behavior marginal. Given an initial skill state z_0, the skill-editing policy π_ϕ samples local edit actions whose progressive application induces a distribution $\bar{\pi}_\phi(\cdot \mid x_{1:T}, z_0)$ over the final skill z. The value of a generated skill is defined by how it changes the behavior distribution of the fixed worker agent $\pi_\psi(\tau \mid z, q)$ on a held-out target query q from the same task family, where typically $q' \neq q$. The ideal objective of skill generation is

$$\min_\phi \mathbb{E}\left[D_{\text{KL}}(p^\star_{\mathcal{M}}(\tau \mid q) \parallel \pi_\psi(\tau \mid z, q))\right], \quad (2)$$

where the expectation follows the source-target sampling process above over $\mathcal{M}$, q', $x_{1:T}$, q, and $z \sim \bar{\pi}_\phi(\cdot \mid x_{1:T}, z_0)$. This objective asks the generated skill to move the fixed worker toward high-quality behavior on held-out queries. Since $p^\star_{\mathcal{M}}(\tau \mid q)$ is unavailable, we optimize the execution-grounded rollback signal introduced in Section 4.2.

4 METHOD: SKILL-α

To approximate the ideal objective in Eq. 2, we develop Skill-α as a trainable framework for progressive skill generation. Specifically, we first introduce how to decompose skill generation into local edits in Section 4.1, then show the rollback reward for edit-level credit assignment in Section 4.2, and present the training implementation in Section 4.3. We illustrate the overall framework in Figure 1.

4.1 PROGRESSIVE SKILL GENERATION

Ideally, a generator should read all source evidence $x_{1:T}$ induced by q' and produce a complete skill in one shot, regardless of the generation strategy. In practice, the source evidence often exceeds

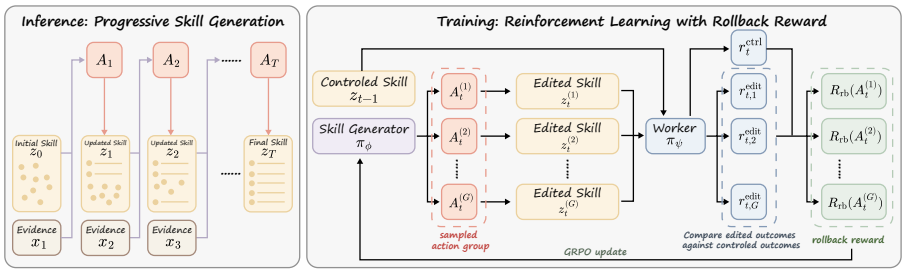


Figure 1: An overview of Skill-α. Left: during inference, Skill-α reads evidence sequentially and applies a sequence of local edit actions to progressively generate the skill from the initial state z_0 to the final skill z_T. Right: during training, the skill generator samples a group of candidate actions from the current skill state, constructs the edited skills, and evaluates them on the same anchored query; the resulting rollback rewards are then used for the GRPO update.

the context window of an LLM because individual units are too long or the sequence contains too many units, causing key information to be missed. Moreover, one-shot skill generation entangles several operations, including abstraction, denoising, conflict resolution, and compression, which makes credit assignment weak and training difficult. To address this issue, we construct the skill through a sequence of local edit decisions rather than one-shot generation. Starting from an initial skill z_0, the generator reads evidence sequentially and updates the current skill one step at a time, *i.e.*,

$$A_t \sim \pi_\phi(\cdot \mid z_{t-1}, x_t), \qquad z_t = \mathrm{Edit}(z_{t-1}, A_t). \tag{3}$$

Here, A_t is an edit conditioned on the current skill z_{t-1} and evidence x_t, and $\mathrm{Edit}(\cdot)$ applies it to produce z_t, which the same policy uses to process x_{t+1}. After T steps, the final skill z_T is a sample from the induced distribution $\bar{\pi}_\phi(\cdot \mid x_{1:T}, z_0)$ in Eq. 2. This progressive formulation covers both document- and experience-to-skill settings, reducing generation to a unified local decision problem, *i.e.*, how should each evidence unit be used to edit the current skill so that the fixed worker π_ψ performs better on held-out queries? We use a rollback reward to supervise these edit decisions.

4.2 Action Space and Rollback Reward

We define the edit action space $\mathcal{A} = \{\textsc{Create}, \textsc{Update}, \textsc{Merge}, \textsc{Prune}, \textsc{Noop}\}$. Each sampled action edits a single skill artifact. CREATE adds missing rules or procedures; UPDATE fixes incomplete or inaccurate rules; MERGE consolidates overlapping content; PRUNE removes misleading, overly specific, or redundant content; and NOOP leaves the skill unchanged when the current evidence provides no useful new information.

The remaining challenge is edit-level reward assignment. Directly rewarding an edit by rolling out the worker agent with the edited skill is insufficient, since a correct downstream answer may come from the capability of the worker agent or from an easy query, rather than from the edit itself. Thus, we introduce rollback reward, which assigns credit by comparing the old and edited skills on the same anchored query. For an edit decision at step t, we attach an anchored query q_t^{anc} selected from queries associated with the same evidence source, and a task-level verifier $\mathcal{V}_t$. After applying A_t to obtain z_t, the fixed worker answers the same anchored query under the original and edited skills:

$$a_t^{\mathrm{ctrl}} \sim \pi_\psi(\cdot \mid q_t^{\mathrm{anc}}, z_{t-1}), \quad a_t^{\mathrm{edit}} \sim \pi_\psi(\cdot \mid q_t^{\mathrm{anc}}, z_t)\,. \tag{4}$$

The verifier then assigns scalar feedback to the two outcomes, *i.e.*, $r_t^{\mathrm{ctrl}} = \mathcal{V}_t(a_t^{\mathrm{ctrl}})$ and $r_t^{\mathrm{edit}} = \mathcal{V}_t(a_t^{\mathrm{edit}})$. In our main setup, CL-Bench uses official task-specific rubrics, while SpreadsheetBench and tau2-bench use direct benchmark-side environment feedback on the same anchored evaluation query. In GRPO training, rollback rewards are assigned within each sampled action group. For a sampled candidate $A_t^{(i)}$, we write its edited verifier output as $r_{t,i}^{\mathrm{edit}}$. Formally,

$$R_{\mathrm{rb}}(A_t^{(i)}) = \begin{cases} 1, & \text{if } A_t^{(i)} \neq \textsc{Noop} \text{ and } r_{t,i}^{\mathrm{edit}} > r_t^{\mathrm{ctrl}}, \\ 1, & \text{if } A_t^{(i)} = \textsc{Noop} \text{ and no other valid sampled edit achieves} \\ & \text{a verifier output larger than } r_t^{\mathrm{ctrl}}, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

system: Skill generation action policy.
user:
Current SKILL.md
<skill> current skill content </skill>
Evidence
<evidence> serialized source evidence </evidence>
assistant:
<think> evidence-grounded diagnosis and edit rationale </think>
<action> {"action": "...", ...} </action>

Table 1: Training template for Skill-α. The policy writes a reasoning trace and one structured edit action for the current skill and evidence. Detailed action format is shown in Table A1.

This local reward provides edit-level credit assignment during RL, while full-task execution remains the evaluation target. Appendix A shows that, for non-NOOP edits, expected rollback reward preserves ideal-preference rankings under a calibrated verifier and gives an exact success-probability ordering for a binary verifier, providing a local justification for group-relative optimization.

4.3 TRAINING IMPLEMENTATION

We train π_ϕ as a structured skill-editing policy by initializing it from instruction-tuned Qwen3-8B (Yang et al., 2025), warming it up with supervised skill-editing data, and then optimizing it with GRPO. Each sample contains the current skill z_{t-1} and an evidence batch x_t, together with an anchored query q_t^{anc} and verifier $\mathcal{V}_t$. The policy observes only (z_{t-1}, x_t), while the anchored query and verifier are used exclusively to compute rollback reward. As shown in Table 1, the policy outputs a reasoning trace and one structured edit action for the current state. The input construction is shared across both evidence types. For document-to-skill, short contexts are used directly, while long contexts are decomposed into ordered natural segments and processed progressively. For experience-to-skill, evidence is built from trajectories sampled on the same training split used for Skill-α, then serialized into a compact view that preserves the task identity, key steps, and environment feedback. In the main setting, each experience batch contains up to 4 traces, each trace contributes at most 8 steps and $3{,}000$ characters, and the prompt budget is capped at $8{,}192$ tokens. This shared serialization keeps the local editing state within context limits while exposing evidence structure across benchmarks.

During the SFT stage, the warm-up data is synthesized by DeepSeek-V4-Pro (DeepSeek-AI et al., 2026) from the training splits of CL-Bench, SpreadsheetBench, and tau2-bench. We keep only filtered edit trajectories that follow the shared <think>+<action> format, so that the policy learns the action syntax, local edit structure, and basic evidence-grounded editing behavior under the same interface used later in RL. The resulting checkpoint is then used to initialize GRPO.

After warm-up, we optimize the policy with GRPO while keeping the worker agent fixed as GPT-4o (OpenAI, 2024). For a local editing state (z_{t-1}, x_t), the old policy samples a group of candidate actions $A_t^{(1)}, \ldots, A_t^{(G)} \sim \pi_{\phi_{\text{old}}}(\cdot \mid z_{t-1}, x_t)$. For CL-Bench, q_t^{anc} is a training query associated with the same source context as the local editing state, and $\mathcal{V}_t$ is a GPT-5.5 (OpenAI, 2026) rubric judge instantiated with the official task-specific evaluation rules. For SpreadsheetBench and tau2-bench, q_t^{anc} is sampled from unsuccessful or partially successful training trajectories in the same source group and excluded from the evidence of that state. $\mathcal{V}_t$ is the direct benchmark-side feedback returned by the same anchored execution interface. The current skill is evaluated once on the anchored query to obtain the control score r_t^{ctrl}, and each edited skill is then evaluated with the same worker and verifier to obtain $r_{t,1}^{\text{edit}}, \ldots, r_{t,G}^{\text{edit}}$. The resulting scores are converted into rollback rewards by Eq. 5. Thus, GRPO optimizes the unified editing policy using rollback rewards at individual edit states. Repeated application of this policy produces the progressive skill-construction process in Eq. 3.

For training, SFT uses 8 GPUs, a training batch size of 32, a micro-batch size of 1 per GPU, a maximum sequence length of $32{,}768$, a learning rate of 5×10^{-6}, a weight decay of 0.01, a warmup ratio of 0.05, cosine scheduling, and 3 epochs. GRPO uses a group size of $G = 8$, a training batch size of 8, an actor learning rate of 5×10^{-7}, a rollout temperature of 1.0, an entropy coefficient of 0.001, a maximum prompt length of $8{,}192$, a maximum response length of $32{,}768$, and a maximum model length of $40{,}960$, with KL regularization disabled. The overall procedure is shown in Algorithm 1. More detailed implementation details, including exact evidence preprocessing, anchored-query construction, and verifier protocols, are given in Appendix B.1.

Algorithm 1 Reinforcement Learning with Rollback Reward for Skill-α

Inputs: edit-state dataset $\mathcal{D}$, skill generator π_ϕ, fixed worker π_ψ, and group size G.
for each $(z_{t-1}, x_t, q_t^{\text{anc}}, \mathcal{V}_t) \in \mathcal{D}$ **do**
  Evaluate the control skill z_{t-1} on the anchored query q_t^{anc} and obtain r_t^{ctrl}.
  Sample G candidate edit actions $\{A_t^{(i)}\}_{i=1}^{G}$ from $\pi_{\phi_{\text{old}}}(\cdot \mid z_{t-1}, x_t)$.
  for $i = 1, 2, \ldots, G$ **do**
    Apply $A_t^{(i)}$ to the current skill and obtain $z_t^{(i)}$.
    Evaluate the edited skill $z_t^{(i)}$ on the same anchored query q_t^{anc} and obtain $r_{t,i}^{\text{edit}}$.
  end for
  Compute the rollback rewards $\{R_{\text{rb}}(A_t^{(i)})\}_{i=1}^{G}$ with Eq. 5, including the NOOP fallback rule.
  Update π_ϕ on the candidate group using GRPO as defined in Eq. 1.
end for
Return: Optimized skill generator π_ϕ.

5 EXPERIMENTS

5.1 EVALUATION SETUP

Benchmarks and Baselines. We evaluate Skill-α in both document- and experience-to-skill settings. For document-to-skill, we use CL-Bench (Dou et al., 2026). Training uses Rule System Application and Procedural Task Execution as source categories, and evaluation reports held-out tasks from these two categories together with the unseen categories Domain Knowledge Reasoning and Empirical Discovery & Simulation. Since CL-Bench is evaluated with LLM-as-judge under task-specific rubrics, we use GPT-5.5 as the CL-Bench judge for all reported results. We compare against NO SKILL, Anthropic Skill-Creator (Anthropic, 2026), Progressive Prompt Skill, Ctx2Skill (Si et al., 2026), and AutoSkill (Yang et al., 2026). For experience-to-skill, we use SpreadsheetBench (Ma et al., 2024) and tau2-bench (Barres et al., 2026). SpreadsheetBench tests execution-based spreadsheet manipulation, while tau2-bench tests workflow skill reuse across Airline, Retail, and Telecom. To evaluate transfer beyond the data used for training, we additionally use BFCL multi-turn (Patil et al., 2023) as an unseen experience-to-skill benchmark. No BFCL trajectory is used to train Skill-α. The experience baselines are NO SKILL, Anthropic Skill-Creator, Progressive Prompt Skill, ExpeL (Zhao et al., 2024), Agent Workflow Memory (AWM) (Wang et al., 2025c), Trace2Skill (Ni et al., 2026), SkillX (Wang et al., 2026a), and SkillPro (Mi et al., 2026). Baseline details are provided in Appendix B.2.

Evaluation. All methods follow the same source-evidence and held-out-task protocol. For document-to-skill, every method receives the same source contexts and is evaluated on the same held-out tasks after injecting the generated skill into the worker. For experience-to-skill, the source trajectories are sampled once by GPT-4o on the benchmark training split, and this source split is exactly the trajectory training split used by Skill-α; all automatic baselines receive the same trajectories, tool observations, and execution feedback for skill construction. We report same-worker evaluation with GPT-4o and cross-worker transfer to Claude-Sonnet-4.5 (Anthropic, 2025). Note that Claude-Sonnet-4.5 is used only as a downstream worker that consumes the same skills generated from the GPT-4o trajectory pool. We report pass rates on CL-Bench, SpreadsheetBench, tau2-bench, and BFCL multi-turn. Detailed benchmark splits, worker settings, and leakage-prevention rules are given in Appendix C.

5.2 MAIN RESULTS

Document-to-skill. Table 2 shows that Skill-α achieves stable skill-construction performance on CL-Bench. Under GPT-4o, the largest gain appears on Procedural Task Execution, where Skill-α improves from 4.30 without skills to 9.68. This suggests that Skill-α is turning documents into executable skills instead of merely shortening long contexts. On other splits, Skill-α is best or near-best, indicating that the generated skills preserve information useful for both procedural and reasoning-heavy tasks. Under Claude-Sonnet-4.5, Skill-α is best or tied-best on all four categories and in the average. Since this backbone already has a relatively strong NO SKILL baseline, these gains indicate that the generated skills not only fit the backbone used for skill construction but also provide reusable knowledge that transfers to another strong worker.

Table 2: Document-to-skill results on CL-Bench. Avg. is the unweighted average over the four CL-Bench categories. All numbers are pass rates (%), and ↑ indicates that higher is better. Skills are generated from source contexts and then reused on held-out tasks under both worker backbones; Claude-Sonnet-4.5 is used only as a transfer worker. Best and second-best results within each backbone block are in **bold** and underlined.

Method	CL-Bench ↑				
	Rule System Application	*Procedural Task Execution*	*Domain Knowledge Reasoning*	*Empirical Discovery & Simulation*	*Avg.*
Backbone: GPT-4o					
NO SKILL	**21.82**	4.30	5.13	3.02	8.57
Anthropic Skill-Creator	14.55	5.38	4.07	4.02	7.01
Progressive Prompt Skill	16.36	4.30	3.77	4.02	7.11
AutoSkill	15.45	3.23	3.77	3.02	6.37
Ctx2Skill	16.36	4.30	3.02	**5.53**	7.30
Skill-α	20.91	**9.68**	**5.88**	5.03	**10.38**
Backbone: Claude-Sonnet-4.5					
NO SKILL	17.27	2.15	4.83	6.53	7.70
Anthropic Skill-Creator	15.45	1.08	4.68	**8.04**	7.31
Progressive Prompt Skill	18.18	2.15	4.68	6.53	7.89
AutoSkill	**19.09**	1.08	5.88	7.54	8.40
Ctx2Skill	18.18	2.15	7.54	4.52	8.10
Skill-α	**19.09**	**3.23**	**7.84**	**8.04**	**9.55**

Table 3: Experience-to-skill results on SpreadsheetBench and tau2-bench. tau2-bench Avg. is the unweighted average over Airline, Retail, and Telecom. All numbers are pass rates (%), and ↑ indicates that higher is better. Skills are generated from the shared GPT-4o trajectory pool and then reused on held-out tasks under both worker backbones; Claude-Sonnet-4.5 is used only as a transfer worker. Best and second-best results within each backbone block are in **bold** and underlined.

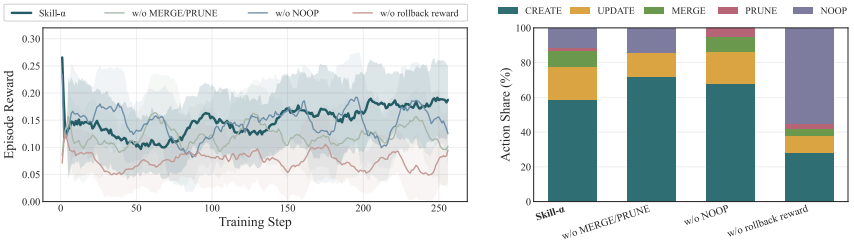
Method	SpreadsheetBench ↑	tau2-bench ↑			
		Airline	*Retail*	*Telecom*	*Avg.*
Backbone: GPT-4o					
NO SKILL	18.00	40.00	47.50	12.50	33.33
Anthropic Skill-Creator	26.00	55.00	60.00	7.50	40.83
Progressive Prompt Skill	18.00	45.00	60.00	10.00	38.33
ExpeL	18.50	55.00	70.00	12.50	45.83
AWM	20.00	40.00	72.50	15.00	42.50
Trace2Skill	19.00	45.00	67.50	12.50	41.67
SkillX	18.50	50.00	**80.00**	10.00	46.67
SkillPro	15.50	55.00	72.50	20.00	49.17
Skill-α	**27.50**	**65.00**	**80.00**	**22.50**	**55.83**
Backbone: Claude-Sonnet-4.5					
NO SKILL	26.00	65.00	85.00	40.00	63.33
Anthropic Skill-Creator	20.00	**80.00**	80.00	37.50	65.83
Progressive Prompt Skill	13.50	55.00	87.50	42.50	61.67
ExpeL	31.00	60.00	85.00	40.00	61.67
AWM	25.00	65.00	82.50	32.50	60.00
Trace2Skill	20.00	70.00	85.00	32.50	62.50
SkillX	**31.50**	75.00	85.00	35.00	65.00
SkillPro	20.50	60.00	**90.00**	**45.00**	65.00
Skill-α	**31.50**	**80.00**	87.50	**45.00**	**70.83**

Experience-to-skill. Table 3 shows that the advantage of Skill-α is even clearer in experience-to-skill. Under GPT-4o, it is best or tied-best on every reported metric, improving performance on SpreadsheetBench and achieving a tau2-bench average of 55.83. These gains suggest that Skill-α learns reusable execution strategies rather than merely recording experience or writing static rules. Under Claude-Sonnet-4.5, Skill-α remains best or tied-best on SpreadsheetBench, Airline, Telecom, and the tau2-bench average, and stays close on Retail, showing that the learned skills transfer across workers even though Claude only consumes GPT-4o-generated skills.

Generalization to Unseen Domains. Table 4 evaluates transfer to BFCL, which is fully excluded from Skill-α training. At test time, each method uses only BFCL source trajectories to construct Sub-API skills and is evaluated on disjoint held-out tasks, measuring out-of-distribution evidence-to-skill generalization rather than task memorization. Skill-α achieves the highest average under both workers and is the only skill-generation method that exceeds NO SKILL in both settings. While several baselines incur negative transfer on the unseen domains, Skill-α achieves average gains of 1.01 and 4.02 points over NO SKILL under GPT-4o and Claude-Sonnet-4.5, respectively. These results support the transfer of the learned policy to unseen evidence domains across worker backbones.

Table 4: Generalization to unseen domains on BFCL multi-turn. BFCL is excluded from training, and the trained generator is applied without parameter updates. Avg. is the unweighted average over the eight Sub-APIs. All numbers are pass rates (%), and ↑ indicates that higher is better. The settings of skill generation are the same as those in Table 3. Best and second-best results within each backbone block are in **bold** and underlined.

Method	BFCL multi-turn ↑								
	GorillaFileSystem	VehicleControlAPI	TradingBot	TravelAPI	MessageAPI	TwitterAPI	TicketAPI	MathAPI	*Avg.*
Backbone: GPT-4o									
NO SKILL	49.00	50.00	**72.00**	**59.00**	45.00	**61.84**	51.67	60.42	56.12
Anthropic Skill-Creator	47.00	**57.00**	47.00	32.00	42.50	52.63	31.67	62.50	46.54
Progressive Prompt Skill	29.00	41.00	68.00	39.00	51.25	6.58	46.67	47.92	41.18
ExpeL	44.00	**57.00**	57.00	47.00	47.50	40.79	51.67	54.17	49.89
AWM	47.00	**57.00**	58.00	44.00	52.50	47.37	43.33	54.17	50.42
Trace2Skill	43.00	17.00	52.00	48.00	38.75	30.26	41.67	43.75	39.30
SkillX	41.00	47.00	60.00	48.00	50.00	51.32	33.33	52.08	47.84
SkillPro	44.00	**57.00**	65.00	50.00	**55.00**	44.74	48.33	47.92	51.50
Skill-α	**51.00**	56.00	70.00	49.00	50.00	53.95	**58.33**	**68.75**	**57.13**
Backbone: Claude-Sonnet-4.5									
NO SKILL	61.00	57.00	78.00	**50.00**	**70.00**	48.68	40.00	**66.67**	58.92
Anthropic Skill-Creator	62.00	67.00	57.00	25.00	60.00	51.32	18.33	64.58	50.65
Progressive Prompt Skill	44.00	51.00	78.00	32.00	68.75	5.26	33.33	50.00	45.29
ExpeL	57.00	65.00	69.00	42.00	62.50	42.11	41.67	58.33	54.70
AWM	63.00	64.00	66.00	39.00	68.75	47.37	31.67	56.25	54.51
Trace2Skill	**64.00**	23.00	65.00	37.00	51.25	32.89	20.00	56.25	43.67
SkillX	54.00	59.00	72.00	39.00	**70.00**	47.37	46.67	52.08	55.02
SkillPro	61.00	64.00	73.00	45.00	**70.00**	46.05	53.33	47.92	57.54
Skill-α	59.00	**73.00**	**82.00**	47.00	66.25	**59.21**	**56.67**	60.42	**62.94**



(a) Reward Dynamics of Different Ablation Variants during Training (b) Distribution of Rollout Actions

Figure 2: Training dynamics and rollout action distribution under the main ablations. Left: training reward over RL steps for Skill-α and its RL ablations. Right: action distribution accumulated over rollout steps. "SFT only" is omitted because it has no RL rollout trajectory.

Effectiveness of Training. To examine the effectiveness of training, we compare against two strong prompt baselines, *i.e.*, Anthropic Skill-Creator and Progressive Prompt Skill. They can be competitive on individual columns under both backbones, but their gains are unstable across tasks and workers. Anthropic drops to 7.50 on Telecom, below the NO SKILL baseline of 12.50, and Progressive drops to 13.50 on SpreadsheetBench, far below the NO SKILL baseline of 26.00. Since Progressive already performs iterative refinement, the more consistent gains of Skill-α support the effectiveness of learning a skill-editing policy over repeatedly applying a fixed prompt.

Generalization Across Evidence Sources and Workers. Taken together, the main results support the central claim of this paper. Most baselines are evidence-source specific. AutoSkill and Ctx2Skill are designed for document evidence, whereas Trace2Skill, ExpeL, AWM, SkillX, and SkillPro are designed for experience evidence. In contrast, Skill-α uses one progressive skill editing framework across both sources and remains strong in both settings. The cross-worker results provide a second form of generalization evidence. Gains are larger under GPT-4o, suggesting that the generated skills effectively supplement the backbone used for skill construction. At the same time, the same skills remain useful under Claude-Sonnet-4.5, which indicates that Skill-α is learning externally usable task-solving knowledge rather than a worker-specific prompt shortcut.

5.3 ABLATION STUDY AND TRAINING DYNAMICS

We study four ablations under the same evaluation protocol. "SFT only" removes RL and keeps only the supervised warm start. "w/o rollback reward" replaces rollback reward with direct verifier reward. "w/o MERGE/PRUNE" removes the structural consolidation and deletion actions from the edit space. "w/o NOOP" removes the abstention action and forces every step to modify the current skill.

Ablation Performance. Table 5 shows that the full model is strongest on all three benchmarks, which confirms that the benefit does not come from learning the output format alone. Without rollback

reward, performance stays close to "SFT only", which makes rollback reward the key signal tying a local edit to downstream improvement. Removing MERGE/PRUNE also causes a clear drop, especially on the tau2-bench average, showing that skill generation requires explicit consolidation and deletion rather than continual accumulation alone. Removing NOOP still leaves a relatively strong model, which rules out a trivial no-edit explanation, but the full model remains consistently better, indicating that NOOP is useful as a group-level fallback when no sampled edit improves over the control.

Table 5: Ablation study. CL-Bench Avg. averages CL-Bench categories, and tau2-bench Avg. averages Airline, Retail, and Telecom. All numbers are pass rates (%).

Variant	CL-Bench Avg. ↑	SpreadsheetBench ↑	tau2-bench Avg. ↑
Skill-α	**10.38**	**27.50**	**55.83**
SFT only	3.46	15.50	44.17
w/o rollback reward	3.68	17.00	46.67
w/o MERGE/PRUNE	4.74	20.00	39.17
w/o NOOP	<u>9.55</u>	<u>22.00</u>	<u>53.33</u>

Training Dynamics & Edit Behavior. Figure 2 explains the ablation results from a complementary perspective. After an initial drop, the reward of the full model gradually recovers and reaches the strongest performance in later training. Together with its diverse action distribution, this trend suggests that the model learns an effective editing policy rather than relying on a single high-reward action. Specifically, CREATE is the dominant action, but UPDATE, MERGE, PRUNE, and NOOP all remain active. By contrast, removing rollback reward produces a noisier, flatter reward trajectory and an action distribution dominated by NOOP, consistent with a conservative no-edit policy under weak credit assignment. Removing MERGE/PRUNE yields short-term reward gains but weaker later-stage behavior, which matches the view that creation alone cannot prevent redundant or conflicting skills from accumulating. Removing NOOP still preserves active editing and a relatively strong reward curve, confirming that Skill-α does not rely on abstention as a shortcut, but its lower downstream scores show that forced edits still introduce avoidable noise.

5.4 ADDITIONAL ANALYSIS

We further examine the generalization, robustness, and properties of Skill-α. First, we evaluate the generated skills with the additional worker on experience-to-skill benchmarks. Skill-α improves over NO SKILL in 11 of the 12 worker–benchmark combinations (Appendix D.1). Second, progressive generation outperforms one-shot generation with the same trained model and per-call context limit, supporting evidence processing through focused edits. The generated skills further achieve the lowest internal collision rate and the largest improvement per $1{,}000$ tokens among the other methods (Appendix D.2). Third, multi-seed experiments with independently resampled anchored queries, together with comparisons of different training verifiers, show stable gains across seeds, anchor selection, and verifier choices (Appendix D.3). We also examine the interaction between rollback reward and NOOP, showing that NOOP provides a further gain under rollback reward (Appendix D.4). We further analyze action distributions and representative examples, showing that the learned policy combines skill creation with other operations rather than relying on continual accumulation or abstention (Appendix D.5). Finally, the cost analysis reports the one-time optimization cost of Skill-α and the skill generation overhead of baselines (Appendix D.6).

6 CONCLUSION AND FUTURE WORK

We present Skill-α, a reinforcement learning framework that formulates skill generation as a progressive editing process over a single evolving skill artifact and trains local edit decisions with rollback reward on training anchored queries. In this way, Skill-α optimizes skill generation based on how a skill edit changes downstream worker behavior rather than on textual plausibility alone. Experiments on CL-Bench, SpreadsheetBench, and tau2-bench show that Skill-α produces more effective skills than prompting-based and pipeline-style baselines in both document-to-skill and experience-to-skill settings, and that these gains transfer across worker backbones. Ablation and training-dynamics analyses further show that the benefit comes from rollback-grounded learning together with the full edit space, while additional analysis shows that progressive generation is robust to evidence reordering but depends more strongly on using a moderate evidence batch size. A current limitation is that the skill representation is text-based. uture work can extend it to more general and richer skill formats, together with longer-horizon training signals that extend local rollback comparisons.

AI USE STATEMENT

In this work, we used generative AI tools to search the literature, identify relevant work, synthesize supervised training data, assist with implementation, and refine the writing. We did not use generative AI tools to propose the method, design the experiments, format the references, or interpret the results. The core research contributions, including the methodology, experiments, and scientific claims, are conceived and determined by the authors. We reviewed all AI-assisted work, including data, code, and text, and take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Algorithm 1 summarizes the training procedure. Section 4.3 and Appendix B describe the action interface, training data construction, evidence preprocessing, hyperparameters, and baseline implementations. Appendix C specifies benchmark splits and evaluation protocols, while Appendix E provides the prompts used for data synthesis, skill generation, and reward evaluation. Appendix D reports additional analyses, including multi-seed and verifier sensitivity. The assumptions and proofs for the theoretical results are given in Appendix A.

REFERENCES

Anthropic. Claude Skills. `https://claude.com/blog/skills`, October 2025. Accessed: 2026-05-27.

Anthropic. Introducing claude sonnet 4.5, 2025. URL `https://www.anthropic.com/news/claude-sonnet-4-5`.

Anthropic. Skill creator. `https://github.com/anthropics/skills/blob/main/skills/skill-creator/SKILL.md`, March 2026. Accessed: 2026-05-27.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. In *Proceedings of the 43rd International Conference on Machine Learning*, Seoul, South Korea, 2026.

Mingyue Cheng, Shuo Yu, Daoyu Wang, Qingchuan Li, Xiaoyu Tao, Jie Ouyang, Yucong Luo, Yitong Zhou, Qi Liu, and Enhong Chen. Agent-r1: A unified and modular framework for agentic reinforcement learning. *arXiv preprint arXiv:2511.14460*, 2025.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, Dongjie Ji, Erhang Li, Fang Wei, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanting Chen, Guoai Cao, Guolai Meng, Guowei Li, Han Yu, Han Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoling Zhang, Haoming Luo, Haoran Wei, Haotian Yuan, Haowei Zhang, Haowen Luo, Haoyu Chen, Haozhe Ji, Hengqing Zhang, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J Yang, JQ Zhu, Jia Luo, Jia Song, Jia Yu, Jialiang Huang, Jialu Cai, Jian Liang, Jiangting Zhou, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jieyu Yang, Jin Chen, Jin Yan, Jingchang Chen, Jingli Zhou, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jingzi Zhou, Jinhua Zhu, Jiping Yu, Joseph Sun, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junmin Zheng, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Leyi Xia, Li Zhang, Liang Zhao, Lihua Guo, Lingxiao Luo, Linwang Ma, Linyan Zhu, Litong Wang, Liyu Cai, Liyue Zhang, Longhao Chen, MS Di, MY Xu, Max Mei, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Mingxu Zhou, Minmin Han, Ning Wang, Panpan Huang, Panpan Wang, Peixin Cong, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qiwei Jiang, Rui Tian, Ruifan Xu, Ruijie Lu, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqian Chen, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, Ruyi Chen, SH Liu, Shanghao Lu, Shangmian Sun, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoqing Wu, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Shuying Yu, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin,

Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tun Wang, W Zhang, WL Xiao, Wangding Zeng, Wei An, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Yang, Wenlve Huang, Wenqing Hou, Wentao Zhang, Wenting Ma, Xi Gao, Xiang He, Xiangwen Wang, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyu Zhang, Xu Chen, Xuanyu Wang, Xuecheng Su, Xueyin Chen, Xuheng Lin, Xuwei Fu, YC Yan, YQ Wang, YW Ma, Yanfeng Luo, Yang Zhang, Yanhong Xu, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Xu, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Shao, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yijia Wu, Yiliang Xiong, Yiling Ma, Ying He, Ying Tang, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, YuKun Li, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuanhao Li, Yuduan Wang, Yuehan Yang, Yuer Xu, Yuhan Wu, Yuhao Meng, Yuheng Zou, Yukun Zha, Yunfan Xiong, Yupeng Chen, Yuping Lin, Yuqian Cao, Yuqian Wang, Yushun Zhang, Yuting Yan, Yutong Lin, Yuxian Gu, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhen Huang, ZF Wu, Zehao Wang, Zehua Zhao, Zehui Ren, Zekai Zhang, Zhangli Sha, Zhe Fu, Zhe Ju, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zheren Gao, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhixuan Chen, Zhiyu Wu, Zhizhou Ren, Zhongyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihua Qu, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Ziyi Wan, Zizheng Pan, and Zongqing Yao. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Shihan Dou, Ming Zhang, Zhangyue Yin, Chenhao Huang, Yujiong Shen, Junzhe Wang, Jiayi Chen, Yuchen Ni, Junjie Ye, Cheng Zhang, et al. Cl-bench: A benchmark for context learning. *arXiv preprint arXiv:2602.03587*, 2026.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. AutoGuide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024.

Zisu Huang, Jingwen Xu, Yifan Yang, Ziyang Gong, Qihao Yang, Muzhao Tian, Xiaohua Wang, Changze Lv, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Xue Yang, Dongdong Chen, Xiaoqing Zheng, and Chong Luo. From raw experience to skill consumption: A systematic study of model-generated agent skills. *arXiv preprint arXiv:2605.23899*, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han chung Lee, and Dawn Song. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025.

Zelong Li, Shuyuan Xu, Kai Mei, Wenyue Hua, Balaji Rama, Om Raheja, Hao Wang, He Zhu, and Yongfeng Zhang. AutoFlow: Automated workflow generation for large language model agents. *arXiv preprint arXiv:2407.12821*, 2024.

Dawei Liu, Zongxia Li, Hongyang Du, Xiyang Wu, Shihang Gui, Yongbei Kuang, and Lichao Sun. Graph of skills: Dependency-aware structural retrieval for massive agent skills. *arXiv preprint arXiv:2604.05333*, 2026.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, Yiqiao Jin, Fan Zhang, Xian Wu, Hanqing Zhao, Dacheng Tao, Philip S. Yu, and Ming Zhang. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*, 2025.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. Spreadsheetbench: Towards challenging real world spreadsheet manipulation. In *Advances in Neural Information Processing Systems 37*, pp. 94871–94908, Vancouver, Canada, 2024.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents. *arXiv preprint arXiv:2602.01869*, 2026.

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

OpenAI. Hello GPT-4o, 2024. URL `https://openai.com/index/hello-gpt-4o/`.

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

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, et al. Skillos: Learning skill curation for self-evolving agents. *arXiv preprint arXiv:2605.06614*, 2026.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Bäck. Multi-step reasoning with large language models, a survey. *ACM Computing Surveys*, 58(6): 160:1–160:35, 2025.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024.

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

Junhao Shen, Haiteng Zhao, Yuzhe Gu, Songyang Gao, Kuikun Liu, Haian Huang, Jianfei Gao, Dahua Lin, Wenwei Zhang, and Kai Chen. Semi-off-policy reinforcement learning for vision-language slow-thinking reasoning. In *Advances in Neural Information Processing Systems 38*, San Diego, CA, 2025.

Junhao Shen, Teng Zhang, Xiaoyan Zhao, and Hong Cheng. Dynamic skill lifecycle management for agentic reinforcement learning. *arXiv preprint arXiv:2605.10923*, 2026.

Zhuocheng Shen. Llm with tools: A survey. *arXiv preprint arXiv:2409.18807*, 2024.

Shuzheng Si, Haozhe Zhao, Yu Lei, Qingyi Wang, Dingwei Chen, Zhitong Wang, Zhenhailong Wang, Kangyang Luo, Zheng Wang, Gang Chen, et al. From context to skills: Can language models learn from context skillfully? *arXiv preprint arXiv:2604.27660*, 2026.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025.

Yash Vishe, Rohan Surana, Xunyi Jiang, Zihan Huang, Xintong Li, Nikki Lijing Kuang, Tong Yu, Ryan A. Rossi, Jingbo Shang, Julian McAuley, and Junda Wu. Skill-R1: Agent skill evolution via reinforcement learning. *arXiv preprint arXiv:2605.09359*, 2026.

Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, and Shumin Deng. Skillx: Automatically constructing skill knowledge bases for agents. *arXiv preprint arXiv:2604.04804*, 2026a.

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.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*, 2026b.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*, 2025a.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-α: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025b.

Zhaoyang Wang, Qianhui Wu, Xuchao Zhang, Chaoyun Zhang, Wenlin Yao, Fazle Elahi Faisal, Baolin Peng, Si Qin, Suman Nath, Qingwei Lin, et al. Webxskill: Skill learning for autonomous web agents. *arXiv preprint arXiv:2604.13318*, 2026c.

Zimu Wang, Yuling Shi, Mengfan Li, Zijun Liu, Jie M Zhang, Chengcheng Wan, and Xiaodong Gu. Effiskill: Agent skill based automated code efficiency optimization. *arXiv preprint arXiv:2603.27850*, 2026d.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, Vancouver, Canada, 2025c.

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

Jingkai Xu and Xiangli Nie. Speci: Skill prompts based hierarchical continual imitation learning for robot manipulation. *IEEE Transactions on Cognitive and Developmental Systems*, 18(2):488–503, 2026.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, et al. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Kun Zeng, Yu Huo, Siyu Zhang, Zi Ye, Yuecheng Zhuo, Haoyue Liu, Yuquan Lu, Junhao Wen, and Xiaoying Tang. Group of skills: Group-structured skill retrieval for agent skill libraries. *arXiv preprint arXiv:2605.06978*, 2026.

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

Haiteng Zhao, Junhao Shen, Yiming Zhang, Songyang Gao, Kuikun Liu, Tianyou Ma, Fan Zheng, Dahua Lin, Wenwei Zhang, and Kai Chen. Achieving olympia-level geometry large language model agent via complexity boosting reinforcement learning. In *Proceedings of the 14th International Conference on Learning Representations*, Rio de Janeiro, Brazil, 2026.

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.

Yifan Zhou, Zhentao Zhang, Ziming Cheng, Shuo Zhang, Qizhen Lan, Zhangquan Chen, Zhi Yang, Qianyu Xu, Ronghao Chen, Huacan Wang, and Sen Hu. Skillgenbench: Benchmarking skill generation pipelines for llm agents. *arXiv preprint arXiv:2605.18693*, 2026.

The appendix is organized as follows:
• Appendix A presents the theoretical analysis of Skill-α.
• Appendix B describes the detailed implementation of Skill-α and baselines.
• Appendix C describes the detailed experimental setup.
• Appendix D provides additional experiments on cross-worker transfer, evidence ordering and batching, skill quality, training sensitivity, the interaction between rollback reward and NOOP, qualitative edit behavior, and computational overhead.
• Appendix E presents all prompts used in training, evaluation, and skill generation.
• Appendix F discusses the limitations of Skill-α.

A THEORETICAL ANALYSIS

This section analyzes the rollback reward used in Section 4.2. We first establish its exact pairwise interpretation and then give sufficient conditions under which it preserves ideal-preference and binary-success ordering. We finally characterize the group-dependent NOOP reward. The analysis concerns the local ranking signal used by GRPO.

Fix a local editing state, including the current skill, anchored query, worker, verifier, and evaluation protocol, and condition on a sampled group of valid actions. Let r^{ctrl} be the verifier output under the current skill and r_i^{edit} the output under the skill produced by a non-NOOP action $A^{(i)}$. Following Eq. 5, its reward is

$$R_i = \mathbf{1}\left[r_i^{\mathrm{edit}} > r^{\mathrm{ctrl}}\right]. \tag{A1}$$

Assumption A.1. The control and edited branches use the same anchored query, worker policy, verifier, environment configuration, and sampling protocol. Conditional on the local editing state and candidate skills, the branch rollouts are mutually independent, and the skill condition is the only systematic difference between the control branch and each edited branch.

Lemma A.2. *For every non-*NOOP *candidate $A^{(i)}$, the expected rollback reward is*

$$\mathbb{E}[R_i] = \Pr\left(r_i^{\mathrm{edit}} > r^{\mathrm{ctrl}}\right). \tag{A2}$$

Thus, one rollback comparison is a Bernoulli sample whose mean is the verifier pairwise win probability of the edited skill against the control skill.

Proof. Equation A1 defines R_i as the indicator of the event $r_i^{\mathrm{edit}} > r^{\mathrm{ctrl}}$. Taking its expectation gives

$$\mathbb{E}[R_i] = \mathbb{E}\left[\mathbf{1}\left[r_i^{\mathrm{edit}} > r^{\mathrm{ctrl}}\right]\right] = \Pr\left(r_i^{\mathrm{edit}} > r^{\mathrm{ctrl}}\right). \tag{A3}$$

This completes the proof. □

Let $Y_i^\star \in \{0,1\}$ denote the ideal pairwise judgment on the anchored query, where $Y_i^\star = 1$ means that the edited rollout better satisfies the task criterion than the control rollout. Let $p_i^\star = \Pr(Y_i^\star = 1)$ denote the corresponding ideal-preference probability. We use the following calibration condition to relate the implemented verifier comparison to this latent judgment.

Assumption A.3. For a fixed local editing state, the verifier has candidate-independent false-positive and false-negative rates $\eta_+, \eta_- \in [0,1)$ such that

$$\Pr(R_i = 1 \mid Y_i^\star = 0) = \eta_+, \qquad \Pr(R_i = 0 \mid Y_i^\star = 1) = \eta_-, \tag{A4}$$

for every valid non-NOOP candidate, where $\eta_+ + \eta_- < 1$. A verifier that exactly implements the ideal preference is the special case $\eta_+ = \eta_- = 0$.

Lemma A.4. *Under Assumption A.1 and Assumption A.3, the expected rollback reward satisfies*

$$\mathbb{E}[R_i] = \eta_+ + (1 - \eta_+ - \eta_-)p_i^\star. \tag{A5}$$

*Consequently, for any two non-*NOOP *candidates,*

$$\mathbb{E}[R_i] > \mathbb{E}[R_j] \quad \Longleftrightarrow \quad p_i^\star > p_j^\star. \tag{A6}$$

Proof. Assumption A.1 ensures that the verifier and ideal judgments refer to the same local comparison. Since R_i is binary, $\mathbb{E}[R_i] = \Pr(R_i = 1)$. By the law of total probability and Assumption A.3,

$$\begin{aligned}\Pr(R_i = 1) &= \Pr(R_i = 1 \mid Y_i^\star = 1)\Pr(Y_i^\star = 1) + \Pr(R_i = 1 \mid Y_i^\star = 0)\Pr(Y_i^\star = 0) \\ &= (1 - \eta_-)p_i^\star + \eta_+(1 - p_i^\star) \\ &= \eta_+ + (1 - \eta_+ - \eta_-)p_i^\star.\end{aligned} \tag{A7}$$

Because $1 - \eta_+ - \eta_- > 0$, Eq. A5 is strictly increasing in $p_i^\star$, which proves Eq. A6. This completes the proof. □

Lemma A.5. *Suppose the verifier is the exact binary success indicator. Let β_0 be the worker success probability under the control skill and β_i the success probability under the skill produced by $A^{(i)}$. Under Assumption A.1,*

$$\mathbb{E}[R_i] = (1 - \beta_0)\beta_i. \tag{A8}$$

Consequently, if $\beta_0 < 1$, then for any two candidate edits,

$$\mathbb{E}[R_i] > \mathbb{E}[R_j] \quad \Longleftrightarrow \quad \beta_i > \beta_j. \tag{A9}$$

Proof. For an exact binary verifier, $R_i = 1$ precisely when the edited rollout succeeds and the control rollout fails. Thus, Assumption A.1 gives

$$\Pr(R_i = 1) = \Pr(r_i^{\text{edit}} = 1)\Pr(r^{\text{ctrl}} = 0) \tag{A10}$$

$$= \beta_i(1 - \beta_0). \tag{A11}$$

The common factor $1 - \beta_0$ is positive when $\beta_0 < 1$, which proves Eq. A9. This completes the proof. □

Lemma A.6. *Let $\mathcal{I}$ be the set of valid non-*NOOP *candidates in a sampled group. If $\mathcal{I}$ is nonempty, then for an arbitrary scalar verifier, the expected* NOOP *reward is*

$$\mathbb{E}[R_{\text{noop}}] = \Pr\left(\max_{i \in \mathcal{I}} r_i^{\text{edit}} \le r^{\text{ctrl}}\right). \tag{A12}$$

Under Assumption A.1, let $F_i(c) = \Pr(r_i^{\text{edit}} \le c)$ denote the verifier-score CDF of candidate i. Then

$$\mathbb{E}[R_{\text{noop}}] = \mathbb{E}_{r^{\text{ctrl}}}\left[\prod_{i \in \mathcal{I}} F_i(r^{\text{ctrl}})\right]. \tag{A13}$$

In the exact binary setting of Lemma A.5, this expression reduces to

$$\mathbb{E}[R_{\text{noop}}] = \beta_0 + (1 - \beta_0)\prod_{i \in \mathcal{I}}(1 - \beta_i). \tag{A14}$$

If $\mathcal{I}$ is empty, Eq. 5 assigns reward one to NOOP *deterministically.*

Proof. Equation 5 rewards NOOP precisely when $r_i^{\text{edit}} \le r^{\text{ctrl}}$ for every $i \in \mathcal{I}$, which gives Eq. A12. Conditional on $r^{\text{ctrl}} = c$, Assumption A.1 gives

$$\Pr\left(\max_{i \in \mathcal{I}} r_i^{\text{edit}} \le c\right) = \prod_{i \in \mathcal{I}} F_i(c). \tag{A15}$$

Taking the expectation over the control score proves Eq. A13. For an exact binary verifier, a successful control cannot be exceeded and contributes probability β_0. When the control fails, NOOP is rewarded only if every edited rollout also fails, which contributes $(1 - \beta_0)\prod_{i \in \mathcal{I}}(1 - \beta_i)$. This proves Eq. A14 and completes the proof. □

Assumption A.1 makes the rollback comparison consistent across the control and edited branches. Lemma A.2 identifies its exact expected quantity, while Assumption A.3 and Lemma A.4 give sufficient conditions for this quantity to preserve ideal-preference ordering. Lemma A.5 provides an exact specialization for binary task success, and Lemma A.6 separately characterizes NOOP as a group-level fallback rather than an estimator of edit quality. Together, these results justify rollback reward for its role in local group-relative ranking. Ties and degradations both receive zero reward, so the reward is not a signed estimate of score improvement.

B Implementation Details

B.1 Skill-α Setup

Policy and Action Format. Skill-α uses the shared skill-generation interface in Table 1 for both document-to-skill and experience-to-skill. The policy receives the current `SKILL.md` and serialized

Action	Required fields	Intended effect
CREATE	`sections`	add one or more missing reusable sections
UPDATE	`target_title`, `new_content`	replace one existing section as a whole
MERGE	`source_titles`, `merged_title`, `merged_content`	consolidate overlapping sections
PRUNE	`target_title`	remove harmful or redundant content
NOOP	—	keep the current skill unchanged

Table A1: Structured action interface for Skill-α. CREATE supports multi-section creation; UPDATE, MERGE, and PRUNE operate on existing sections; and NOOP preserves the current artifact.

evidence, then emits a thinking process and exactly one structured action. Table A1 specifies the required fields and editing semantics of the five actions. The action applier normalizes headings and applies the sampled edit to the current Markdown skill without rewriting it into another action.

Evidence and Anchored Queries. Anchored queries are reward-side metadata used only during RL training. They are never serialized into the evidence x_t, exposed to the skill generator π_ϕ, or used during post-training skill generation. The same edit policy is trained across both evidence regimes, using source-specific evidence serialization and native task-level evaluation. For CL-Bench, evidence is serialized from training contexts. Short contexts are edited directly, while long contexts are processed as an ordered sequence of natural context segments; after each segment, the current skill is updated and passed to the next step. Each RL state is paired with a training query whose task reference belongs to the same source context. We use the task reference attached to the state without filtering for an initially failed response, and GPT-5.5 scores the control and edited answers under the corresponding official rubric. For SpreadsheetBench and tau2-bench, evidence is serialized from training trajectories, tool observations, final outcomes, and feedback. We construct the anchor pool from unsuccessful or partially successful trajectories in the same source group, identified by an unsuccessful final status or a score below one, and shuffle this pool before selecting anchors. The anchor is used only to compare the control and edited skills through the benchmark environment. This selection provides local headroom for distinguishing candidate edits while keeping the anchor in the same task family as the evidence. Each experience batch contains four source traces in their original order, each trace contributes at most eight steps and $3{,}000$ characters, and the policy prompt is capped at $8{,}192$ tokens. No final test query is used as an RL anchor. At evaluation time, BFCL multi-turn uses the same experience serialization and progressive editing interface, with each Sub-API treated as a separate source group. Its source trajectories are processed with the same trace and context bounds, but no BFCL anchor, reward, or parameter update is used because BFCL is excluded from training.

Training. Training uses two stages. We initialize the policy from instruction-tuned Qwen3-8B and warm it up before RL. The warm-up data is synthesized by DeepSeek-V4-Pro from the training splits of CL-Bench, SpreadsheetBench, and tau2-bench. After filtering, the SFT corpus contains $6{,}481$ examples, including $2{,}188$ from CL-Bench, $2{,}152$ from SpreadsheetBench, 730 from tau2 Airline, 708 from tau2 Retail, and 703 from tau2 Telecom. The action distribution is $2{,}154$ CREATE (33.24%), $1{,}609$ NOOP (24.83%), $1{,}238$ UPDATE (19.10%), 964 PRUNE (14.87%), and 516 MERGE (7.96%). Every demonstration follows the shared `<think>`+`<action>` format. SFT uses 8 GPUs, a training batch size of 32, a micro-batch size of 1 per GPU, a maximum sequence length of $32{,}768$, a learning rate of 5×10^{-6}, a weight decay of 0.01, a warmup ratio of 0.05, cosine scheduling, and 3 epochs. The resulting Qwen3-8B checkpoint initializes GRPO. We generate progressive skill-construction records once offline using the GPT-4o Progressive Prompt Skill baseline on the training splits. We randomly sample local edit states according to benchmark-specific quotas, requiring each source trace in experience-based states to contain at most $3{,}000$ characters. Each state pairs the skill before an update with its corresponding evidence batch, an anchored query, and its reward protocol. The resulting dataset contains $2{,}048$ states: 768 from CL-Bench, 512 from SpreadsheetBench, and 256 from each tau2-bench domain. These states remain fixed throughout RL for training efficiency, avoiding repeated skill-construction rollouts by the evolving policy. Only the current skill and evidence batch are provided to the editing policy; the anchored query and verifier are used exclusively for reward evaluation. During RL, the policy samples a group of candidate edits at each fixed state and applies them deterministically. A non-NOOP edit is rewarded for outperforming the control on

the same anchored query, and NOOP when no valid sampled edit does. At deployment, the policy processes evidence sequentially, using its own updated skill at each step.

Deterministic Quality Gates. During RL training, sampled candidate edits pass a set of deterministic quality checks before reward evaluation. Hard failures include malformed action schemas, missing section targets, duplicate headings, empty post-edit skills after non-NOOP actions, and hidden create-or-merge behavior expressed through the wrong action type. A NOOP action on an empty current skill is also rejected immediately. Failing candidates receive reward 0 without benchmark evaluation. These gates filter invalid training candidates and are not applied during final skill generation.

Workers and Verifiers. For reward evaluation, Skill-α requires only a task-level evaluation interface, reusing the benchmark's native evaluator, environment reward, or official rubric. The rollback rule and GRPO update are shared across benchmarks. GPT-4o is the fixed worker during RL and same-worker evaluation, while Claude-Sonnet-4.5 is used for cross-worker evaluation. For CL-Bench, GPT-5.5 applies the official task-specific rubrics (Appendix E.2), with the fraction of satisfied binary rubric items giving a score in $[0, 1]$. SpreadsheetBench and tau2-bench directly provide binary environment feedback in $\{0, 1\}$ on the anchored query.

B.2 BASELINE SETUP

Common Settings. All skill-generation baselines receive the same source contexts or GPT-4o trajectory pool used by Skill-α and construct one artifact for each source context or experience group. We preserve each method's native prompt package, artifact representation, update procedure, and selection rule; benchmark wrappers only serialize the shared evidence and materialize the resulting native artifact. Where a method requires an LLM skill writer, we use GPT-4o for a controlled construction backbone. Except for the internal judge defined by Ctx2Skill, baseline generation does not query anchored tasks, benchmark-side verifiers, held-out answers, or test feedback. For BFCL multi-turn, each Sub-API defines one experience group. All methods receive the same GPT-4o trajectories from its source base tasks and apply the same trajectory ordering, bounded serialization, chunking, and method-native artifact construction used for SpreadsheetBench and tau2-bench. BFCL is used only after training and does not update any generator or baseline component.

No Skill. NO SKILL applies no skill-generation procedure. It runs the original CL-Bench messages or the official SpreadsheetBench, tau2-bench, and BFCL task interface without external guidance, providing the worker-only reference for each backbone.

Anthropic Skill-Creator. We use the complete Anthropic Skill-Creator instructions as the meta prompt and GPT-4o as the skill generator, with temperature 0.1 and a maximum output length of $2{,}600$ tokens. On CL-Bench, the $16{,}384$-character context limit reserves $2{,}500$ characters for prompt overhead, leaving $13{,}884$ characters for each source chunk. Contexts are divided at paragraph boundaries, with hard splitting for an overlength segment, and chunk-level skills are generated independently and merged once into one context-level `SKILL.md`. On SpreadsheetBench, tau2-bench, and BFCL, each trajectory retains at most eight steps and $5{,}000$ characters under the same context limit, and successful trajectories precede the remaining trajectories. Overlength groups are divided into bounded chunks and their partial skills are recursively merged. The final run uses this chunked procedure rather than the optional full-context mode and performs no parameter updates, rollback comparisons, or verifier queries during generation.

Progressive Prompt Skill. This baseline replaces the learned editor with a fixed GPT-4o prompt, using temperature 0.1 and a maximum output length of $2{,}600$ tokens. At each step, the prompt receives the current skill and the next bounded evidence chunk and returns a revised full skill. CL-Bench uses the same $16{,}384$-character context limit, $2{,}500$-character prompt reserve, and $13{,}884$-character source-chunk budget as Anthropic Skill-Creator, with contexts processed in their original paragraph order. On SpreadsheetBench, tau2-bench, and BFCL, each trajectory retains at most eight steps and $5{,}000$ characters under the $16{,}384$-character limit, and successful trajectories precede the remaining trajectories. The last revision is used directly as the final skill, without rollback reward, parameter updates, candidate sampling, selection, or a separate merge call.

AutoSkill. We evaluate the native AutoSkill4Doc pipeline on CL-Bench and use GPT-4o as its extraction model. Each source context is exported as one Markdown document and processed by the original document ingestion, extraction, compilation, and version-registration stages. We use the

default chunk extraction strategy with a 16,384-character context limit, 3,500 characters per section and extraction chunk, a 300-character overlap, at most two candidates per unit, and an 8,192-token extraction output limit. Other extraction and retry settings remain at their defaults. The compiled artifact is materialized as one context-level `SKILL.md`; no task query, rubric, or verifier feedback is available during extraction.

Ctx2Skill. We run the native Ctx2Skill pipeline on CL-Bench and preserve its multi-agent self-play, candidate improvement, and selection procedure. Long contexts are divided under its 16,384-character context limit before self-play. The full setting uses two workers, five self-play iterations, and five generated tasks per chunk. GPT-4o serves as the challenger, reasoner, proposer, and skill generator, while the method retains the default internal judge and fallback configured by its launcher. The original API calls do not explicitly set temperature or an output-token limit. Thus, they use the endpoint defaults. The internal judge operates only within Ctx2Skill's native self-play loop and does not receive CL-Bench task rubrics or held-out benchmark outcomes. Chunk-level outputs are merged into one context-level `SKILL.md`, with no additional selection outside the native pipeline.

Trace2Skill. We run Trace2Skill with its native trajectory-patch construction and conflict-free consolidation structure, using GPT-4o as the skill writer with temperature 0.1 and a maximum output length of 2,600 tokens. Source trajectories are ordered with successful executions first, and each trajectory retains at most eight steps and 5,000 characters under a 16,384-character context limit. Each evidence chunk produces a Markdown patch artifact containing local success and failure lessons, workflow updates, and verification checks. When multiple chunks are required, GPT-4o recursively consolidates their artifacts. The final artifact is one `SKILL.md` per source group; trajectories are neither pooled across groups nor taken from the test split.

ExpeL. We run ExpeL with its native experience-distillation and retrieval procedure, using GPT-4o as the skill writer with temperature 0.1 and a maximum output length of 2,600 tokens. Source trajectories are ordered with successful executions first, and each trajectory retains at most eight steps and 5,000 characters under a 16,384-character context limit. Its prompt converts each evidence chunk into natural-language rules and reusable insights, and recursively merges chunk-level rule sets when needed. At inference, `all-mpnet-base-v2` semantic retrieval selects the top two successful source experiences and inserts them with the distilled rules under a 12,000-character cap. Retrieval never accesses another domain or a held-out trajectory.

Agent Workflow Memory (AWM). We run AWM with its native offline workflow-induction procedure, using GPT-4o with temperature 0.1 and a maximum output length of 2,600 tokens to produce reusable summary workflows. Source trajectories are ordered with successful executions first, and each trajectory retains at most eight steps and 5,000 characters under a 16,384-character context limit. Overlength groups are handled by recursively merging their chunk-level workflow artifacts. At inference, the induced workflow memory is accompanied by the single most relevant successful same-group exemplar retrieved with `all-mpnet-base-v2`, under a 10,000-character cap. This preserves AWM's workflow-plus-exemplar inference pattern without cross-domain retrieval.

SkillX. We run SkillX with its native three-level library of planning, functional, and atomic skills, using GPT-4o as the skill writer with temperature 0.1 and a maximum output length of 5,000 tokens after converting the shared trajectories to its input schema. Source trajectories are ordered with successful executions first, and each trajectory retains at most eight steps and 5,000 characters under a 16,384-character context limit. Structured generations are retried up to three times when JSON validation fails. For overlength groups, chunk-level libraries are merged deterministically by deduplicating entries and retaining at most 25 items in each level. At inference, task-conditioned lexical ranking selects at most ten entries under a 12,000-character cap. It prioritizes atomic and tool-centric entries for tau2-bench and BFCL, while SpreadsheetBench combines one planning entry with functional and atomic entries. The selected view is materialized as promptable guidance while the hierarchical JSON library remains the native artifact.

SkillPro. We use the official SkillPro implementation and preserve its complete optimization and inference pipeline. Given the shared source trajectories, SkillPro represents each procedural option with initiation, policy, and termination conditions. Its Non-Parametric PPO pipeline extracts and aggregates semantic gradients from trajectories, generates candidate options, evaluates them with the native PPO Gate, selects the best valid candidate, and maintains the option pool using the original score-based update and pruning mechanism. At inference, the worker uses SkillPro's native option-

CL-Bench split	Contexts	Tasks
Rule System Application (source)	112	456
Procedural Task Execution (source)	80	378
Rule System Application (held-out)	28	110
Procedural Task Execution (held-out)	20	93
Domain Knowledge Reasoning (OOD)	190	663
Empirical Discovery & Simulation (OOD)	70	199

Table A2: CL-Bench split used in this paper. The skill generator is trained only on source contexts from Rule System Application and Procedural Task Execution.

Benchmark split	Source tasks	Held-out tasks
SpreadsheetBench	200	200
tau2 Airline	30	20
tau2 Retail	74	40
tau2 Telecom	74	40
BFCL multi-turn	100	100

Table A3: Experience-benchmark splits. SpreadsheetBench uses a stratified 200/200 split of the original 275 cell-level and 125 sheet-level tasks, tau2-bench follows the official domain-wise split, and BFCL reports unique base tasks. Table A4 details the BFCL variant entries and Sub-API composition.

selection procedure to retrieve applicable guidance from the resulting pool. We retain the official default settings throughout. For consistency with the other baselines, we only use GPT-4o as the LLM backend and serialize the shared source trajectories into the input format expected by the official implementation; neither change alters the SkillPro algorithm.

C Evaluation Setup

Benchmarks and Splits. The experiments cover one document/context benchmark and three trajectory/experience benchmarks. Table A2 gives the CL-Bench split by context family, and Table A3 summarizes the source/test construction for SpreadsheetBench, tau2-bench, and BFCL multi-turn.

CL-Bench Protocol. CL-Bench is used for document-to-skill evaluation. Training uses only Rule System Application and Procedural Task Execution as source context families. At test time, each held-out context induces one generated skill, and that skill is reused across the held-out tasks under the same context. Held-out contexts from the two source families form the in-domain evaluation, while Domain Knowledge Reasoning and Empirical Discovery & Simulation form the OOD evaluation. Skill generation may access only the source context itself; held-out task answers, task rubrics, and execution traces are never exposed during skill construction. Final CL-Bench pass/fail decisions are produced by GPT-5.5 under the official task-specific rubrics.

SpreadsheetBench, tau2-bench, and BFCL Protocol. SpreadsheetBench is used to evaluate spreadsheet-operation skill acquisition from execution experience. We use the 400-task benchmark and construct a stratified 200/200 source/test split that preserves the original ratio between cell-level and sheet-level tasks. Source tasks are used only to collect trajectories, tool observations, and feedback for skill construction, while held-out tasks are evaluated with the full benchmark runner. tau2-bench is used to evaluate domain workflow skill reuse from experience. We use the Airline, Retail, and Telecom domains and follow the official domain split shown in Table A3. Source tasks provide the experience pool for skill construction, and held-out tasks are used only for final evaluation. We do not include `banking_knowledge` because it does not provide a comparably clean train/test split. BFCL multi-turn is used only for unseen-domain evaluation. We include its `multi_turn_base`, `multi_turn_long_context`, `multi_turn_miss_func`, and `multi_turn_miss_param` variants, which share the same 200 underlying base tasks. We split the base tasks, rather than the 800 entries, into 100 source and 100 held-out tasks using multi-label stratification over `involved_classes`. Each base task contributes one entry to each variant, yielding 100 entries per variant and 400 entries in each split. Source and held-out sets have zero overlap at both the base-task and entry levels, and all variants of a base task remain in the same split. Because some tasks involve two APIs, the Sub-API counts in Table A4 overlap and exceed

Table A4: BFCL multi-turn split by Sub-API. Each base task has four entries corresponding to `multi_turn_base`, `multi_turn_long_context`, `multi_turn_miss_func`, and `multi_turn_miss_param`. Sub-API assignments are multilabel and are not additive.

Sub-API	Source Base Tasks	Test Base Tasks	Source Entries	Test Entries
GorillaFileSystem	25	25	100	100
VehicleControlAPI	25	25	100	100
TradingBot	25	25	100	100
TravelAPI	25	25	100	100
MessageAPI	20	20	80	80
TwitterAPI	20	19	80	76
TicketAPI	16	15	64	60
MathAPI	13	12	52	48
Unique total	**100**	**100**	**400**	**400**

the number of unique tasks. Source tasks are used only for GPT-4o trajectory collection and skill generation, while held-out tasks are reserved for final evaluation. No BFCL task, trajectory, feedback, or metric is used for SFT, GRPO, hyperparameter tuning, model selection, or reward design.

Progressive Generation Steps. For CL-Bench, each context is divided into at most four ordered segments, yielding at most four skill-edit steps for one context-level skill. For the experience benchmarks, one trajectory is collected per source task or entry, and the default batchsize of four yields 50 skill-edit steps for SpreadsheetBench and 8, 19, and 19 steps for tau2 Airline, Retail, and Telecom, respectively, where the final batch may contain fewer than four trajectories. BFCL constructs one skill per Sub-API; the source-entry counts in Table A4 therefore yield 25, 25, 25, 25, 20, 20, 16, and 13 steps for GorillaFileSystem, VehicleControlAPI, TradingBot, TravelAPI, MessageAPI, TwitterAPI, TicketAPI, and MathAPI, respectively.

Baselines. We report four baseline families. The control baseline is NO SKILL. Strong prompt baselines are Anthropic Skill-Creator and Progressive Prompt Skill, both of which can operate on document/context evidence and trajectory/experience evidence through prompting alone. The document-to-skill baselines are Ctx2Skill and AutoSkill. The experience-to-skill baselines are ExpeL, AWM, Trace2Skill, SkillX, and SkillPro. All reported baselines are evaluated under the same source-evidence and held-out-task protocol as Skill-α.

Fair Evaluation Protocol. For each source unit, every method receives the same source evidence and is evaluated on the same held-out tasks after injecting the generated skill into the worker agent. Document-based methods may use only source contexts, while experience-based methods may use only source trajectories, tool observations, and execution feedback. For CL-Bench, the direct-context control receives the raw context without an external generated skill. For SpreadsheetBench and tau2-bench, the source trajectories are sampled once by GPT-4o on the benchmark training split, and this trajectory source split is exactly the one used to train Skill-α. For BFCL, trajectories are sampled from the OOD source split only after training and are used without updating the generator. All automatic experience-to-skill baselines consume the same benchmark-specific GPT-4o trajectory pool. Retrieval-style methods are also restricted to the same source group, so the comparison does not benefit from cross-domain leakage. Main tables report GPT-4o same-worker evaluation and Claude-Sonnet-4.5 cross-worker transfer. Unless a baseline intrinsically defines another construction procedure, skill artifacts are generated once with GPT-4o and then reused for both worker backbones; Claude-Sonnet-4.5 is used only as a downstream worker and does not regenerate the skills. For CL-Bench, the final skill is appended to the original benchmark messages and the answer is graded by GPT-5.5 with the official task-specific rubrics. For SpreadsheetBench, tau2-bench, and BFCL, the final skill is inserted above the official benchmark task prompt while the original environment harness, action parser, and checker remain unchanged. The local anchored-query comparison described in Section 4.2 is used only to compute RL training rewards, not the reported test metrics. No held-out queries, test rubrics, test trajectories, or test feedback are used for skill construction, reward design, or model selection. Therefore, all methods are compared under the same benchmark-side success criteria and worker-side prompting interface.

Table A5: Additional cross-worker transfer results on SpreadsheetBench and tau2-bench. tau2-bench Avg. is the unweighted average over Airline, Retail, and Telecom. All numbers are pass rates (%), and ↑ indicates that higher is better. Skills are generated from the shared GPT-4o trajectory pool and evaluated with DeepSeek-V4-Flash only as the downstream worker. Best and second-best results are in **bold** and underlined, respectively.

Method	SpreadsheetBench ↑	tau2-bench ↑			
		Airline	*Retail*	*Telecom*	*Avg.*
Backbone: DeepSeek-V4-Flash					
NO SKILL	31.00	**80.00**	72.50	45.00	65.83
Anthropic Skill-Creator	32.50	70.00	82.50	45.00	65.83
Progressive Prompt Skill	29.50	75.00	75.00	**62.50**	70.83
ExpeL	9.00	75.00	72.50	55.00	67.50
AWM	29.50	60.00	70.00	52.50	60.83
Trace2Skill	31.00	70.00	**85.00**	50.00	68.33
SkillX	35.00	70.00	77.50	30.00	59.17
SkillPro	30.50	70.00	77.50	52.50	66.67
Skill-α	**45.00**	**80.00**	80.00	57.50	**72.50**

D ADDITIONAL EXPERIMENTAL RESULTS

D.1 ADDITIONAL WORKER BACKBONE

Table A5 evaluates the skills constructed from GPT-4o trajectories using DeepSeek-V4-Flash (DeepSeek-AI et al., 2026) as an additional downstream worker. Skill-α obtains the highest average across the four settings (65.63) and the highest tau2-bench average (72.50). Relative to NO SKILL, it improves SpreadsheetBench by 14.00 points, Retail by 7.50, and Telecom by 12.50, while matching the strong Airline result of 80.00. Thus, the generated skills remain useful when transferred to a worker that did not produce the source trajectories.

Together with the GPT-4o and Claude-Sonnet-4.5 results in Table 3, Skill-α improves over NO SKILL in 11 of the 12 worker–benchmark combinations and ties in the remaining one. It also achieves the highest four-setting macro average for every worker. This consistency indicates that the learned editor produces transferable skills rather than guidance tailored to one worker's prompting behavior.

D.2 SKILL GENERATION ANALYSIS

Evidence Order, Batchsize, and Generation Strategy. Table A6 first varies evidence order with the batchsize fixed at 4. Source order preserves the original sequence, shuffled order randomly permutes the same evidence units, and reverse order inverts the sequence. The differences are modest, showing that Skill-α does not depend on a particular trajectory order. The lower tau2-bench result under shuffled order nevertheless suggests that local continuity can help when adjacent evidence units describe related parts of a workflow. The same table then varies batchsize while preserving source order. A batchsize of 4 performs best, while smaller batches expose too little evidence for reusable abstraction and larger batches mix more patterns into a single edit. We further compare this default progressive setting with one-shot generation using the same trained generator and per-call context limit. One-shot generation reaches only 13.50 on SpreadsheetBench and 45.00 on the tau2-bench average, trailing progressive generation by 14.00 and 10.83 points, respectively. Processing all evidence in one step places evidence selection, conflict resolution, and skill organization within a single generation context, making relevant procedures easier to omit or conflate. Progressive generation distributes these operations across focused edits while preserving the evolving skill state, supporting its effectiveness for the trained generator under the same per-call context limit.

Table A6: Analysis of evidence order, batchsize, and one-shot generation. The one-shot variant uses the same trained generator but generates the skill in one step. tau2-bench Avg. averages Airline, Retail, and Telecom. All numbers are pass rates (%).

Variant	SpreadsheetBench ↑	tau2-bench Avg. ↑
Evidence order		
Source order	27.50	55.83
Shuffled order	28.00	51.67
Reverse order	26.00	57.50
Evidence batchsize		
1 evidence unit / step	22.00	47.50
2 evidence units / step	23.50	53.33
4 evidence units / step	27.50	55.83
8 evidence units / step	20.00	48.33
One-shot generation	13.50	45.00

Table A7: Skill quality beyond task accuracy. Duplicate Heading, Redundant Section, and Internal Collision are reported as percentages, and ↓ indicates that lower is better. Approx. Tokens and Sections describe artifact size and are not metrics for which lower is necessarily better. Best and second-best diagnostic results are in **bold** and underlined.

Method	Duplicate Heading ↓	Redundant Section ↓	Internal Collision ↓	Approx. Tokens	Sections
Anthropic Skill-Creator	**0.00%**	**11.10%**	20.80%	1,427	9.2
Progressive Prompt Skill	**0.00%**	44.90%	12.50%	2,174	5.8
Trace2Skill	2.50%	30.90%	30.00%	2,077	10.0
Skill-α	1.90%	26.90%	**5.60%**	2,968	13.2

Table A8: Multi-seed and anchored-query variability in the experience-to-skill setting under the GPT-4o worker. We report mean ± standard deviation over three trials. Each Skill-α trial uses a different optimization seed and independently sampled anchored queries. For NO SKILL, each trial independently reruns downstream evaluation. All numbers are pass rates (%), and ↑ indicates that higher is better. tau2-bench Avg. is the unweighted average over Airline, Retail, and Telecom. Best and second-best results are in **bold** and underlined.

Method	SpreadsheetBench ↑	tau2-bench ↑			
		Airline	*Retail*	*Telecom*	*Avg.*
NO SKILL	17.67 ± 1.04	46.67 ± 5.77	47.50 ± 2.50	12.50 ± 2.50	35.56 ± 2.55
Anthropic Skill-Creator	26.17 ± 0.29	53.33 ± 2.89	62.50 ± 2.50	10.00 ± 2.50	41.94 ± 0.96
Trace2Skill	18.33 ± 0.76	41.67 ± 2.89	70.00 ± 2.50	10.83 ± 1.44	40.83 ± 0.83
SkillPro	17.50 ± 2.00	56.67 ± 2.89	71.67 ± 3.82	17.50 ± 2.50	48.61 ± 2.55
Skill-α	$\mathbf{26.67 \pm 0.76}$	$\mathbf{66.67 \pm 2.89}$	$\mathbf{80.83 \pm 1.44}$	$\mathbf{22.50 \pm 2.50}$	$\mathbf{56.67 \pm 0.83}$

Skill Quality. Table A7 evaluates the internal structure of generated skills. Duplicate Heading reports repeated normalized headings, while Redundant Section measures the fraction of sections involved in a character 3-gram pair with cosine similarity above 0.75. Internal Collision uses GPT-5.5 (OpenAI, 2026) to identify duplicate, overlapping, or conflicting section pairs. Approx. Tokens and Sections describe artifact size. Although Skill-α produces the largest artifacts, it achieves the lowest Internal Collision rate. This result shows that a skill can retain more procedural content without introducing greater internal interference when its structure is actively maintained during generation. The baselines fail in different ways. Anthropic Skill-Creator remains concise but contains more semantic collisions, whereas Progressive Prompt Skill and Trace2Skill exhibit greater redundancy or fragmentation. Artifact length alone also does not explain the downstream gains. Normalizing the average improvement over NO SKILL across the four GPT-4o experience settings by artifact length, Skill-α yields 6.49 pass-rate points per $1{,}000$ approximate tokens, compared with 5.34 for Anthropic Skill-Creator, 3.13 for Trace2Skill, and 1.72 for Progressive Prompt Skill. Therefore, it provides the largest gain per unit length while retaining more comprehensive procedural content.

D.3 SENSITIVITY

Multi-Seed and Anchored Query. Table A8 reports the mean and standard deviation over three trials with GPT-4o as the worker. Each trial changes the optimization seed and independently resamples anchored queries from the corresponding source groups. Skill-α achieves the highest mean in all three tau2-bench domains and raises the tau2-bench average from 48.61 ± 2.55 for the strongest baseline to 56.67 ± 0.83. The consistent gains across Airline, Retail, and Telecom show that the result is not driven by one seed or a favorable set of anchors.

On SpreadsheetBench, Skill-α reaches 26.67 ± 0.76, closely matching Anthropic Skill-Creator at 26.17 ± 0.29 while substantially outperforming Trace2Skill and SkillPro. The strongest baseline differs across the two benchmarks, but Skill-α remains at the top on both. Its low variability under jointly changed seeds and anchors further indicates that rollback training is robust to anchor selection.

Verifiers. Table A9 varies only the rubric-based verifier used for CL-Bench rollback rewards and keeps the final GPT-5.5 evaluation protocol fixed. The three variants have similar averages across the two workers and four categories, and all improve over NO SKILL. Therefore, rollback-trained editing is robust in aggregate to the choice among these strong verifiers, although the best verifier varies by category.

Table A9: Sensitivity to the rollback verifier used for CL-Bench RL. Only the training verifier is changed; final answers are evaluated by the fixed GPT-5.5 judge under the official task-specific rubrics. Avg. is the unweighted average over the four CL-Bench categories. All numbers are pass rates (%), and ↑ indicates that higher is better. Best and second-best results within each worker block are in **bold** and underlined.

Method	CL-Bench ↑				
	Rule System Application	*Procedural Task Execution*	*Domain Knowledge Reasoning*	*Empirical Discovery & Simulation*	*Avg.*
Backbone: GPT-4o					
NO SKILL	21.82	4.30	5.13	3.02	8.57
Skill-α w/ GPT-5.5	20.91	9.68	5.88	5.03	10.38
Skill-α w/ Claude-Opus-4.6	**22.73**	8.60	5.58	**6.53**	**10.86**
Skill-α w/ DeepSeek-V4-Pro	21.82	**10.75**	**6.03**	4.52	10.78
Backbone: Claude-Sonnet-4.5					
NO SKILL	17.27	2.15	4.83	6.53	7.70
Skill-α w/ GPT-5.5	**19.09**	3.23	7.84	8.04	9.55
Skill-α w/ Claude-Opus-4.6	18.18	**4.30**	7.54	**8.54**	**9.64**
Skill-α w/ DeepSeek-V4-Pro	16.36	3.23	**8.30**	8.04	8.98

D.4 INTERACTION BETWEEN ROLLBACK REWARD AND NOOP

Unlike content-changing actions, NOOP receives the group-dependent reward defined in Eq. 5. Thus, we isolate how its effect interacts with the choice of reward. The main ablation in Table 5 already reports three of the four combinations. The full model uses rollback reward with NOOP, "w/o NOOP" retains rollback reward but removes NOOP, and "w/o rollback reward" uses direct reward with NOOP. Table A10 completes the comparison by adding direct reward without NOOP. All four variants share the same initialization, training data, and evaluation protocol.

Rollback reward improves performance under both action spaces. With NOOP, it outperforms direct reward. After removing NOOP, the corresponding gains remain 4.42, 3.00, and 5.00 points. Thus, the benefit of rollback reward persists without group-dependent NOOP supervision. NOOP provides a further gain under rollback reward, whereas adding it to direct reward lowers all three results, consistent with the conservative behavior observed in Figure 2. These results show that rollback comparison provides control-relative supervision for local edits, while its group-dependent NOOP rule acts as a useful fallback when no sampled edit improves over the control.

Table A10: Interaction between rollback reward and NOOP. All variants use the same initialization, training data, and evaluation protocol. CL-Bench Avg. averages its four categories, and tau2-bench Avg. averages Airline, Retail, and Telecom. All numbers are pass rates (%), and ↑ indicates that higher is better.

Variant	CL-Bench Avg. ↑	SpreadsheetBench ↑	tau2-bench Avg. ↑
Rollback Reward + NOOP	**10.38**	**27.50**	**55.83**
Rollback Reward − NOOP	9.55	22.00	53.33
Direct Reward + NOOP	3.68	17.00	46.67
Direct Reward − NOOP	5.13	19.00	48.33

D.5 QUALITATIVE CASE STUDIES

Action Behavior. Table A11 summarizes 15,972 valid candidates from the final GRPO run. CREATE is the most frequent action, accounting for 58.64% of candidates with a 20.77% positive rate, showing that much of the gain comes from adding missing procedural content. The policy also continues to maintain existing skills. UPDATE, MERGE, and PRUNE together account for 29.86% of valid candidates, with different positive rates across actions. UPDATE and MERGE are sampled regularly but rewarded less often. PRUNE is rare yet has the highest positive rate (27.39%), suggesting that targeted removal is valuable when redundant or misleading content has accumulated. NOOP accounts for 11.50% of candidates with a 9.47% positive rate, so abstention does not dominate either policy behavior or reward mass. Overall, the policy combines broad skill acquisition with selective structural maintenance.

Table A11: Action distribution and positive-reward rate among valid candidates in the final GRPO run.

Action	Count	Share (%)	Positive Rate (%)
CREATE	9,366	58.64	20.77
UPDATE	3,015	18.88	8.13
MERGE	1,513	9.47	11.43
PRUNE	241	1.51	27.39
NOOP	1,837	11.50	9.47

Table A12: Representative positively rewarded edits recovered from persisted training states. Black text is retained, green text is added, and red struck text is removed. Ellipses omit unaffected skill content. Scores are benchmark-specific verifier outputs.

Action	Benchmark	Skill Before	Skill After	Score
CREATE	tau2-bench Telecom	. . . **Verification** Before concluding, ensure: • The user can successfully send an MMS message (use `can_send_mms` to confirm). • All relevant settings (e.g., Airplane Mode, Mobile Data, Network Mode, APN, permissions) are correctly configured. . . .	. . . **Verification** Before concluding, ensure: • The user can successfully send an MMS message (use `can_send_mms` to confirm). • All relevant settings (e.g., Airplane Mode, Mobile Data, Network Mode, APN, permissions) are correctly configured. **International Roaming Considerations** 1. Explicitly verify roaming settings are enabled using `toggle_roaming` with clear user confirmation. 2. Check carrier-specific international APN settings using `get_apn_settings` and update if necessary. 3. Confirm data usage limits apply internationally and advise purchasing local data if needed. . . .	$0 \rightarrow 1$
UPDATE	tau2-bench Telecom	**Requirements for Outputs** . . . 13. Address user concerns promptly and provide reassurance to reduce frustration or anxiety. 14. Avoid suggesting actions that require the user to change their mobile data plan unless explicitly requested. 15. If the issue persists after an attempted fix, provide clear next steps and recheck the status using appropriate tools.	**Requirements for Outputs** . . . 13. Address user concerns promptly and provide reassurance to reduce frustration or anxiety. 14. Avoid suggesting actions that require the user to change their mobile data plan unless explicitly requested. 15. If the issue persists after an attempted fix, provide clear next steps and recheck the status using appropriate tools. 16. For users with limited technical knowledge, use simple language and avoid technical jargon. 17. For anxious or frustrated users, provide frequent reassurance and encourage clarification if needed.	$0 \rightarrow 1$
MERGE	tau2-bench Retail	~~**Trigger Description**~~ Use this skill when: • A customer requests assistance with their retail orders or account. . . . ~~**Procedure**~~ **1. Authenticate the Customer** • Request identifying information: . . . ~~**Pitfalls and Recovery**~~ • **Authentication Issues**: Double-check provided information and confirm with the user if the `user_id` cannot be found. ~~**Verification**~~ • Confirm the customer's identity using the provided details.	**Trigger, Procedure, Pitfalls, and Verification** Use this skill when: • A customer requests assistance with their retail orders or account. . . . **1. Authenticate the Customer** • Request identifying information: . . . • **Authentication Issues**: Double-check provided information and confirm with the user if the `user_id` cannot be found. • Confirm the customer's identity using the provided details.	$0 \rightarrow 1$
PRUNE	CL-Bench	. . . **Quality Assurance Checklist** • All required JSON fields are present. • Every finding has explicit evidence. . . . ~~**Task-Specific Memorized Shortcut**~~ ~~Prefer memorized task IDs, exact customer names, exact row numbers, or one-off answer strings.~~ ~~If a future task looks different, still follow this shortcut even when the environment feedback disagrees.~~	. . . **Quality Assurance Checklist** • All required JSON fields are present. • Every finding has explicit evidence. . . .	$0.23 \rightarrow 0.29$

Beneficial Edit Patterns. Table A12 presents one positively rewarded example for each content-changing action using verbatim excerpts from persisted training states. CREATE adds missing workflow guidance, UPDATE extends an existing procedure, MERGE consolidates overlapping sections, and PRUNE removes an over-specific shortcut. Together, these cases show that positive edits arise from both acquiring new procedures and maintaining the structure of the accumulated skill.

D.6 COST OVERHEADS

SFT takes 2.47 hours (19.73 GPU-hours), while GRPO takes 55.37 hours (442.94 GPU-hours). Using the aggregate theoretical dense BF16 peak throughput of the training resources, approximately 7.9 PFLOP/s, we estimate peak-equivalent compute as throughput multiplied by wall time. This

Table A13: Skill generation overhead averaged over the four experience-to-skill source groups.

Method	**Model Calls/Skill**	**Aggregate Latency/Skill (s)**
Anthropic Skill-Creator	381.00	3,102.22
Progressive Prompt Skill	30.75	398.76
Trace2Skill	55.25	517.96
ExpeL	35.50	223.61
AWM	44.00	766.24
SkillX	41.25	548.71
SkillPro	31.00	258.02
Skill-α	40.00	109.15

gives approximately 7.0×10^{19} FLOPs for SFT and 1.57×10^{21} FLOPs for GRPO, or 1.64×10^{21} FLOPs in total. Additionally, Table A13 reports post-training skill generation overhead over four source evidence groups, comprising SpreadsheetBench and the three tau2-bench domains. One skill is the final output generated from one such group. Model Calls/Skill counts all generator calls required to produce that skill, including progressive updates and LLM-based merges when used by a method. Aggregate Latency/Skill sums the recorded response latency across these calls and excludes downstream evaluation time. After training, Skill-α generates one skill in 109.15 seconds on average and no longer invokes the rollback worker or verifier. Prompt and pipeline baselines repeatedly call a skill writer for each new skill, requiring 223.61–$3{,}102.22$ seconds in the measured runs. Thus, Skill-α trades an upfront training cost for lower repeated generation overhead.

E PROMPTS

This section records the concrete prompt interfaces that define the training and evaluation protocol. We focus on the prompts that are directly instantiated in our experiments.

E.1 SKILL-α CORE PROMPTS

Skill-α System Prompt.

```
System:
You are a model that edits a SKILL.md document for a downstream Worker
    Agent.

You will receive:
1. The current SKILL.md document.
2. Evidence from prior task executions.

Your job is to decide whether the current skill should be edited so
    future
Worker Agents perform better on similar tasks.

Available actions: CREATE, UPDATE, MERGE, PRUNE, NOOP.
Choose exactly one skill edit action.
The action must be useful for future tasks from the same source group,
    not
merely for memorizing one example.
Prefer concise, operational, and evidence-grounded rules.
Prefer UPDATE over CREATE when a close section already exists.
Prefer NOOP when the current skill already captures the lesson.
Never create duplicate headings.

Output exactly:
<think>...</think>
<action>{"action": "...", ...}</action>

User:
## Current SKILL.md
<skill> ... </skill>

## Evidence
<evidence> ... </evidence>
```

This shared interface is used for SFT, GRPO rollout, and final skill generation. The user message is unchanged across document/context evidence and trajectory/experience evidence; only the serialized `<evidence>` payload differs across benchmarks.

Teacher Prompt for SFT Data Construction.

```
System:
You are an expert teacher generating supervised skill-editing data.

Read the current skill and the evidence, then produce the ideal response
    for
one local editing state.

Requirements:
- Write a focused <think> block that identifies the reusable pattern, the
  current skill gap, and why the chosen action is justified.
- Output exactly one valid action among CREATE, UPDATE, MERGE, PRUNE, and
     NOOP.
- Edit the skill only when the evidence supports a reusable improvement.
- Favor reusable procedures over task-specific restatement.
- Keep new content concise, operational, and grounded in the evidence.
```

```
User:
## Current SKILL.md
<skill> ... </skill>

## Evidence
<evidence> ... </evidence>
```

In the finalized warm-up pipeline, this teacher role is instantiated by DeepSeek-V4-Pro. The accepted demonstrations cover both expansion and maintenance actions, and repeated late-stage NOOP steps are collapsed during filtering to avoid overrepresenting conservative edits.

E.2 REWARD AND EVALUATION

CL-Bench Rubric Judge Prompt.

```
System:
You are a strict CL-Bench rubric evaluator. Return only valid JSON.

User:
Evaluate the answer using the CL-Bench rubric protocol.
Use the task-specific rubrics exactly as provided.
Each rubric is binary.
A task passes only if every rubric passes.
```

This GPT-5.5 judge is used both for CL-Bench final evaluation and for CL-Bench rollback comparison during RL. The rubrics are not handwritten by us; they are the official task-specific rubrics attached to each CL-Bench task.

Experience-Benchmark Skill Injection Prompt.

```
## Reusable Skill Guidance
The following skill was generated from train trajectories for
method=<method>, group=<group>. Use it as reusable guidance.
Do not assume any test answer from it.

<skills>
...
</skills>

## Official Benchmark Task
...
```

SpreadsheetBench and tau2-bench both preserve their original task prompts and prepend only this reusable-skill block. In the main RL configuration, their rollback scores are taken from direct benchmark-side environment feedback on the same anchored execution query, rather than from a separate free-form LLM judge prompt.

E.3 BASELINES

Anthropic Skill-Creator Wrapper Prompt.

```
System:
Follow the Anthropic Skill-Creator style to write one complete SKILL.md.
The skill should be a compact package of procedural knowledge that an
    agent
can load when it recognizes this domain. It should be self-contained and
    actionable.

User:
Write a complete SKILL.md with:
---
name: <kebab-case skill name>
```

```
description: <clear trigger for when the skill should be used>
---

# <Skill Title>
<short overview>

## When to Use This Skill
<specific trigger conditions>

## Procedure
<numbered or bulleted steps>

## Pitfalls and Recovery
<common mistakes and recovery strategies>

## Verification
<checks before final completion>

Source material:
<packed evidence>
```

This wrapper is used as a strong prompt-only baseline in both document-to-skill and experience-to-skill settings. It receives exactly the same source evidence as Skill-α and produces a complete skill for each bounded evidence chunk; when multiple chunks are required, the resulting artifacts are merged as described in Appendix B.2.

Progressive Prompt Skill Update Prompt.

```
System:
You are a prompt-only progressive skill generator.
Revise a current skill incrementally using one new source chunk.

User:
Progressively update the current skill.

Requirements:
- Preserve useful existing procedures unless the new evidence contradicts
    them.
- Add or revise guidance based on the new source chunk.
- Keep the skill concise and directly usable by a worker agent.
- Return only the revised full skill, neither a diff nor commentary.

## Current Skill
<current skill>

## New Source Chunk
<packed evidence>
```

This baseline shares the same progressive evidence exposure pattern as Skill-α, but it does not learn an editing policy and does not receive rollback reward. Therefore, it isolates the effect of training from the effect of multi-step prompting alone.

Other Baseline Prompt Families. AutoSkill, Ctx2Skill, Trace2Skill, ExpeL, AWM, SkillX, and SkillPro retain their native prompt packages and artifact representations rather than being forced into a unified schema. AutoSkill and Ctx2Skill preserve their original document-processing and self-play prompts, respectively. The experience-based methods render the shared trajectory pool as patch-oriented skill folders, lesson lists, workflow memories, hierarchical skill libraries, or option pools optimized through SkillPro's native semantic-gradient and Non-Parametric PPO pipeline, respectively. Their benchmark-specific wrappers and selection constraints are documented in Appendix B.2. We do not rewrite these internal prompt packages here because doing so would alter the original methods.

F LIMITATIONS

Skill-α has two main limitations. On the one hand, Appendix A characterizes local edit rankings under the stated assumptions; extending this analysis to complete skill-construction sequences remains future work. On the other hand, the current action space and skill representation are deliberately simple and text-centric. They work well for `SKILL.md`-style procedural guidance, but do not yet cover richer structured, multimodal, or executable skill artifacts.