REVISE, VERIFY, DISTILL: SELF-EVOLVING SKILLS FOR LANGUAGE AGENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Textual skills let language agents reuse past experience, but a skill that looks reasonable may still fail when executed. We study how agents can turn these failures into experience that transfers to new tasks and improves the model itself. To achieve this, our framework operates in three stages. First, SKILL EVOLUTION repairs a skill on the task that produced it: after a failed rollout it localizes where progress stopped, changes one rule using evidence from the trajectory, and tests the result in the environment; the first successful version is certified, and its failed parent is kept to record what changed. Second, SKILL TRANSFER asks whether repaired skills also help on tasks they were never written for: certified skills are generalized into a frozen bank for held-out tasks, while later repairs stay local. Third, SKILL INTERNALIZATION moves the corrections into the model itself. We introduce EVOOPD: a frozen copy of the base model scores the skill-free student's own actions under both skill versions, the certified one supplying the training target and their score difference weighting the actions affected by the repair. Together, these stages repair a skill where it failed, reuse it where it was never tested, and fold the verified edits into the parameters; we call the overall process EVOLUTION LEARNING. Revision raises source-task success from 13.62% to 46.27% on ALFWorld and from 24.54% to 28.96% on WebShop. On held-out ALFWorld tasks, the frozen bank raises success from 6.43% to 37.86% (seen) and from 5.22% to 35.07% (unseen). With all skills removed at test time, EVOOPD reaches 54.52% and 64.18% over three seeds, compared with 7.86% and 4.48% for the base model. The same verified edits therefore support both skill reuse and model learning.

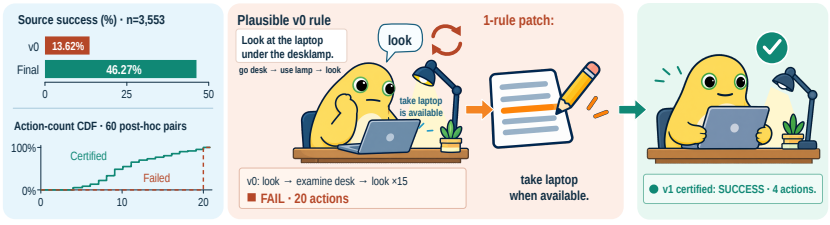

Figure 1: ALFWorld evidence for local skill repair, showing aggregate success, action efficiency, and a one-rule change that turns a failed trajectory into a four-step success.

1 INTRODUCTION

Language agents often repeat the same errors in long-horizon tasks. In household manipulation and web shopping, an agent may skip a required subgoal or repeat an action that leaves the environment unchanged (Yao et al., 2023). These errors only become clear during interaction. Textual skills offer a simple way to reuse this experience: the agent records an instruction from a rollout and reads it on later tasks. The text can be updated without training model parameters and inspected directly.

Prior work stores experience either as text or in model weights. Text-based methods use verbal reflections, distilled workflows, reasoning memories, and libraries of reusable skills (Shinn et al.,

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.

2023; Zhao et al., 2024; Wang et al., 2025; Ouyang et al., 2026; Mi et al., 2026; Lu et al., 2026b). They support cheap, inspectable updates at test time. Distillation methods train a skill-free student from a teacher that receives a skill (Yang et al., 2026a; Wu et al., 2026). The student then runs without skill text at inference time.

Both approaches aim to reuse experience, but storing a skill or learning from it does not by itself show that the experience improves task execution. A study of repository skills for coding agents also finds that useful-looking skill text need not lead to clear performance gains (Kozyrev et al., 2026). Our central question is:

> How can an agent gain useful experience from execution failures and use it to improve performance on later tasks?

This requires answering three questions: (1) Does the experience help solve the task where it was obtained? (2) Does it also help on new tasks? (3) Can the model learn from the corrections and improve without receiving skill text? As shown in Figure 1, an instruction to look at a laptop under a desk lamp leaves the agent repeatedly looking around, even though it can already pick up the laptop. Changing one rule to take the laptop when available lets it finish in four actions. This case shows why the first question requires execution tests. Whether the repaired skill transfers to other tasks, and whether the model can learn from the repair, remain separate questions.

To address the first question, we introduce SKILL EVOLUTION, which uses feedback from a failed rollout to locate where progress stopped, change one rule, and test the revised skill in a fresh rollout. It certifies the first successful version and retains its failed parent. The skill-evolution loop is *trajectory* $\rightarrow$ *failure localization* $\rightarrow$ *skill patch* $\rightarrow$ *environment validation* $\rightarrow$ *promote* or *revise*. For the second, SKILL TRANSFER generalizes the certified skills into a frozen bank and evaluates their use on held-out tasks. It also tests whether further repair helps when a retrieved skill fails; those changes stay local to the new task. For the third, we study SKILL INTERNALIZATION and introduce EVOOPD as its training method. It trains a skill-free student using the successful and failed versions. The student first produces its own actions, and a frozen copy of the base model scores them under each skill version. The successful version supplies the training target, while disagreement between the two versions gives more weight to actions whose scores change after the repair. Evaluation then tests what the student has learned with all skill text removed. We call this overall process EVOLUTION LEARNING.

Skill revision raises source-task success from 13.62% to 46.27% on ALFWorld and from 24.54% to 28.96% on WebShop. On held-out ALFWorld tasks, the frozen bank raises success from 6.43% to 37.86% on the seen split and from 5.22% to 35.07% on the unseen split. After distillation, the skill-free student reaches 54.52% and 64.18% over three seeds, compared with 7.86% and 4.48% for the same base checkpoint.

Our contributions are: (1) **SKILL EVOLUTION**, a skill repair process that tests each revision in the environment and retains successful versions together with their failed predecessors as records of tested corrections; (2) **SKILL TRANSFER**, an evaluation of whether certified skills help on held-out tasks, separating frozen-bank retrieval from further task-local repair; and (3) **SKILL INTERNALIZATION**, which moves the verified corrections into the model itself: we introduce EVOOPD, a training method that uses the difference between successful and failed skill versions to weight distillation from the successful version, and we evaluate the resulting student without skills.

2 RELATED WORK

2.1 EXPERIENCE REUSE AND VERIFIED SKILL EVOLUTION

ReAct provides the basic reasoning–action interface used by many language agents (Yao et al., 2023). Reflexion, ExpeL, Agent Workflow Memory, and ReasoningBank extend this loop across episodes by storing reflections, reusable experience, workflows, or outcome-conditioned reasoning memories (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2025; Ouyang et al., 2026). ACE maintains an evolving playbook through generation, reflection, and incremental curation (Zhang et al., 2026). Trace2Skill turns execution traces into transferable skills, while WikiSkill accumulates

knowledge that guides later skill updates (Ni et al., 2026; Tang et al., 2026). In computer-use tasks, Hu et al. (2026) study a versioned skill library whose updates become available in later iterations.

A parallel line treats skills as executable or optimizable objects: Voyager grows a code library, Skill-Pro performs non-parametric skill optimization, and ContractSkill makes skills explicitly repairable (Wang et al., 2024; Mi et al., 2026; Lu et al., 2026b). TextGrad uses language-model feedback to optimize prompts and other system components, and GEPA reflects on execution traces to propose and test prompt updates (Yuksekgonul et al., 2025; Agrawal et al., 2026). SkillOpt accepts bounded skill edits when they improve a held-out validation score; COBRA-Skills uses contextual bandits to allocate the evaluation budget among skill candidates (Yang et al., 2026b; Lu et al., 2026a). Recent studies further show that retrieved skills can be harmful, which motivates selection and repair checks (Li et al., 2026; Wang et al., 2026c). Our focus is a source-task repair and what can be learned from it: an environment rollout on the task that produced the skill certifies the successful version, rather than a held-out validation score as in SkillOpt, its failed predecessor is retained for distillation, and transfer-time edits use a task-local copy.

2.2 On-policy distillation and skill internalization

Knowledge distillation transfers a teacher distribution into a student (Hinton et al., 2015); on-policy variants rescore responses sampled from the current student, reducing mismatch between training data and the student's own behavior (Agarwal et al., 2024; Zhao et al., 2026). Skill-SD, OPID, and SEED show that privileged skill context can guide multi-turn or agentic learning while being removed at inference time (Wang et al., 2026a; Yang et al., 2026a; Wu et al., 2026). We build on this teacher–student principle. EvoOPD uses the same frozen base checkpoint to score the same student response under a certified skill and its failed predecessor. The certified view supplies the target, and disagreement between the two views sets the action weights. The student generates responses without a skill and is the only model updated during training.

3 Problem Formulation

Let $\tau = (g, \mathcal{E}, c)$ denote an interactive task with instruction g, environment $\mathcal{E}$, and its official success oracle c. Under a fixed execution specification ξ—the prompt and history renderer, action interface and parser, decoding rule, reset procedure, and horizon—an agent π_θ receives observation o_t and acts from the history

$$h_t = (g, o_0, a_0, o_1, \ldots, a_{t-1}, o_t), \qquad a_t \sim \pi_\theta(\cdot \mid h_t, s), \tag{1}$$

where s is an optional textual skill and $s = \emptyset$ denotes skill-free execution. A completed episode produces trajectory $\zeta_\tau^\xi(s; \pi_\theta) = (o_0, a_0, o_1, \ldots, a_{T-1}, o_T)$ and realized outcome

$$Y_\tau^\xi(s; \pi_\theta) = c\big(\zeta_\tau^\xi(s; \pi_\theta)\big) \in \{0, 1\}. \tag{2}$$

We omit π_θ when the actor is fixed. We call a skill version *certified for* τ when one completed validation rollout under ξ yields $Y_\tau^\xi(s) = 1$. This certificate applies to the task, actor checkpoint, and execution specification used for that rollout. Transfer to other tasks is evaluated separately.

A skill s^k is a structured document at version k, identified by its content hash and immutable parent. A failed rollout yields evidence e^k and a localized branch b^k. An editor proposes a bounded semantic patch Δ^k, while a deterministic Harness checks that the patch names the exact parent and admissible evidence before applying it:

$$\Delta^k = E_\psi(\tau, s^k, \zeta^k, e^k, b^k), \qquad s^{k+1} = \mathcal{H}(s^k, \Delta^k). \tag{3}$$

The Harness rejects malformed, identity, duplicate, or out-of-scope edits. For text edits, the editor supplies replacement content. For action-grounded edits, it selects a listed admissible action and target rule; the Harness constructs the conditional rule text. With a revision budget K_{rev}, source-task self-evolution returns the first successful version

$$k^* = \min\{k \in \{0, \ldots, K_{\text{rev}}\} : Y_\tau^\xi(s^k) = 1\}, \tag{4}$$

or marks the lineage unresolved if the set is empty.

When $k^* > 0$, certification also identifies the last failed predecessor $s_\tau^- = s^{k^*-1}$, the accepted patch $\Delta_\tau^- = \Delta^{k^*-1}$, and the certified successor $s_\tau^+ = s^{k^*}$. Let γ_τ^+ denote the environment certificate

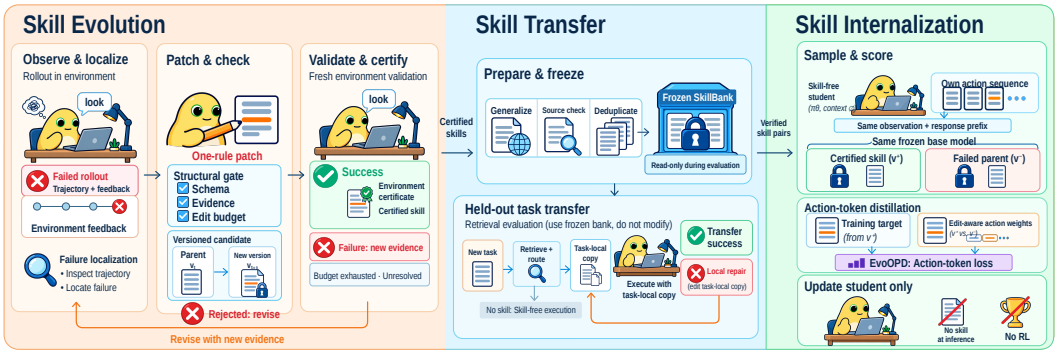

Figure 2: Overview of the three stages in EVOLUTION LEARNING. SKILL EVOLUTION uses execution feedback to patch and certify skills, and SKILL TRANSFER freezes the certified bank before retrieval and task-local repair. The SKILL INTERNALIZATION stage applies EVOOPD: certified skills provide training targets, failed parents set action-token weights, and the student is then deployed without skills or RL.

binding τ, ξ, the actor checkpoint, the content hash of s_τ^+, and its successful validation trajectory. We define the resulting *certified evolution transition* as

$$\chi_\tau = \left(s_\tau^-, \Delta_\tau^-, s_\tau^+, \gamma_\tau^+\right), \quad Y_\tau^\xi(s_\tau^-) = 0, \quad s_\tau^+ = \mathcal{H}(s_\tau^-, \Delta_\tau^-), \quad Y_\tau^\xi(s_\tau^+) = 1. \tag{5}$$

The learning object is an environment-verified change from a failed version to its successful child. If $k^* = 0$, the run yields a positive-only certified record $\chi_\tau^+ = (s_\tau^+, \gamma_\tau^+)$; this record has no failed-to-success contrast, so downstream learning uses its positive signal only.

These definitions give three research questions. **RQ1: Skill Evolution.** Can the versioned update in Equation 3 convert a skill-free failure into a certified skill within K_{rev} revisions and, when a revision is needed, produce the transition in Equation 5? **RQ2: Skill Transfer.** Given a frozen bank $\mathcal{B}$ of source-certified skills, can a selector $R(q_\tau, \mathcal{B}) \in \mathcal{B} \cup \{\emptyset\}$ improve direct success on an unseen task, and can a task-local lineage recover when the selected skill is imperfect while $\mathcal{B}$ stays frozen? **RQ3: Skill Internalization.** Can certified transitions χ_τ, together with positive-only records χ_τ^+, improve the skill-free objective

$$J_0(\theta) = \mathbb{E}_{\tau \sim \mathcal{D}}\left[Y_\tau^\xi(\emptyset; \pi_\theta)\right], \tag{6}$$

when skills are available during training but removed at evaluation? Here $\mathcal{D}$ is the target task distribution; the expectation also includes any randomness in execution under ξ. The actor parameters θ are separate from ξ and are updated only during internalization.

4 EVOLUTION LEARNING

4.1 FRAMEWORK OVERVIEW

Figure 2 organizes the three questions above into one lifecycle: SKILL EVOLUTION produces verified skill versions on the source task, SKILL TRANSFER reuses the certified versions on new tasks, and SKILL INTERNALIZATION folds them into the student. The editor, Harness, environment, and learner have distinct responsibilities: propose, enforce, verify, and internalize.

4.2 SKILL EVOLUTION: ENVIRONMENT-VERIFIED SKILL REPAIR

We seek a skill that helps the agent solve its source task, with execution evidence to support each accepted revision. For a task τ, we first run π_θ without a skill under specification ξ and record its trajectory and outcome. A language-model editor E_ψ uses this record to write the initial skill s^0, which we then test on the same task. This step is applied after both successful and failed baseline rollouts. If the skill fails, the next question is where to change it: a long trajectory may contain many legal actions that do not help complete the task. We therefore compare observed transitions with goal milestones, such as acquiring an object or completing a required transformation. The locator ranks

missed progress and unrecovered errors, including invalid or repeated actions and missed admissible goal actions, to select one repair branch b^k.

Using this evidence, the editor proposes a small patch Δ^k to named rules in the current skill. The deterministic Harness applies the update in Equation 3 only after checking the parent version, cited evidence, and edit budget; malformed, unchanged, duplicate, or previously visited versions are rejected and returned to the editor for another proposal. Free-text edits supply replacement text, whereas action-grounded edits select a rule and an action from the environment's admissible list, from which the Harness constructs a conditional rule. These constraints keep the change tied to the observed failure, but execution is still needed to determine whether it helps. Each accepted candidate therefore receives one completed fresh rollout on the same task under ξ, with infrastructure failures retried separately. If $Y_\tau^\xi(s^k) = 1$, we certify the version and add it to the SkillBank; otherwise, its new trajectory provides evidence for the next revision. We stop at the first successful version as in Equation 4, or retain the unresolved record if the revision or patch budget is exhausted. Earlier versions remain unchanged. When a revision succeeds, we retain both the successful skill and its failed predecessor, together with the patch and execution certificate, as the transition in Equation 5. The successful skill can then be tested for transfer, and the paired versions can guide model learning. Appendix B gives the document schema and editor protocol.

4.3 Skill Transfer: Certified bank construction and reuse

A source certificate shows that a skill repairs one instance, but does not establish whether it helps on other tasks. To prepare these skills for reuse, we first canonicalize certified documents and collapse exact duplicates. For a near-duplicate group within one task family, a representative may absorb another member after it completes one real rollout on that member's source task; otherwise both are retained. The raw bank, trajectories, and certificates remain immutable, and merging similar skills therefore requires evidence that the representative works on the other source task. A generalizer G_ϕ then rewrites each retained document from instance constants to semantic roles while preserving operation order, object multiplicity, and destination constraints. It also emits a short catalog card containing the task family, capabilities, applicable conditions, and rejection conditions. Since this rewrite may change how the skill behaves, we retain the generalized proposal $\tilde{s}_i = G_\phi(s_i)$ only after it passes a fresh source-regression rollout,

$$Y_{\tau_i}^\xi(\tilde{s}_i) = 1, \tag{7}$$

after which the same validation-gated deduplication rule is applied again. The result is a compact, certified bank $\mathcal{B}$ whose snapshot hash is frozen before transfer evaluation.

With this bank fixed, we next test whether its skills help on unseen tasks. For a task τ, we reset the environment and first collect the skill-free outcome in Equation 2. A coarse retriever then uses the instruction, task family, and initial context to form a small eligible set, from which a language-model router uses the catalog cards to select at most one candidate or abstains:

$$\hat{s}_\tau = R_\rho(q_\tau, \mathcal{B}) \in \mathcal{B} \cup \{\emptyset\}. \tag{8}$$

Here q_τ contains the instruction, task family, and initial observation; ρ specifies the fixed router model and routing configuration. The router matches transformation order, multiplicity, and destination semantics, and the full document is loaded after selection. If the router abstains, the task is executed without a skill and keeps that skill-free outcome in both the retrieval and the local-repair setting. Otherwise, the selected skill is cloned as a task-local $\bar{s}_\tau^0$, validated once, and, if necessary, revised by the same counterexample-guided loop. To distinguish reuse from further adaptation, we report direct-transfer success and success after local repair separately. During this process,

$$\mathcal{B}_{\text{after}} = \mathcal{B}_{\text{before}}, \tag{9}$$

because retrieval-time edits remain local and the shared bank stays unchanged.

Table 1: Source-task Skill Evolution on ALFWorld and WebShop. Success rates (%) are shown without skills, with the initial skill, and after environment-verified revision.

	ALFWorld							WebShop					
Skill stage	All	Pick	Pick2	Clean	Heat	Cool	Look	All	Fashion	Beauty	Elec.	Garden	Grocery
No skill	9.82	16.20	8.61	9.38	3.92	2.44	19.16	21.18	28.81	21.57	23.41	13.11	18.80
Initial skill v_0	13.62	17.59	16.73	11.38	10.02	5.07	20.13	24.54	31.76	25.82	25.08	16.73	22.85
Ours	**46.27**	**61.90**	**43.17**	**49.23**	**39.87**	**24.20**	**55.84**	**28.96**	**36.44**	**30.74**	**29.44**	**20.68**	**26.94**

Table 2: Skill Transfer on held-out ALFWorld and WebShop tasks. Success rates (%) compare skill-free execution, retrieval from a frozen SkillBank, and task-local repair after retrieval fails. Local repair uses additional rollouts and environment feedback beyond the single-attempt settings above it.

	ALFWorld Seen							ALFWorld Unseen							WebShop
Setting	All	Pick	Pick2	Clean	Heat	Cool	Look	All	Pick	Pick2	Clean	Heat	Cool	Look	All
No skill	6.43	5.71	8.33	3.70	0.00	4.00	23.08	5.22	0.00	0.00	6.45	0.00	0.00	27.78	22.60
Frozen bank + router	37.86	54.29	25.00	48.15	37.50	8.00	53.85	35.07	45.83	23.53	32.26	26.09	38.10	44.44	22.80
+ Local repair	**48.57**	**62.86**	**41.67**	**62.96**	**56.25**	**8.00**	**61.54**	**55.22**	**58.33**	**47.06**	**51.61**	**43.48**	**52.38**	**83.33**	**30.20**

4.4 Skill Internalization with EvoOPD

External skills increase test-time context and require a reliable selector. We therefore use the verified lineages to train a model that can act without these skills, following the objective in Equation 6. Each training record is a certified transition χ_τ from Equation 5, or a positive-only record χ_τ^+ when s_0 succeeds immediately. The student and both teacher views start from the same base checkpoint, but only the student is updated. The student receives no skill and controls the environment, while the frozen teacher checkpoint is reused with s^+ as the positive context and, when available, s^- as the counter context. The objective uses no reward model, reference policy, task-reward loss, or policy-gradient term.

Following on-policy distillation (Agarwal et al., 2024), at update n the skill-free student π_{θ_n} first collects a batch of episodes with the prescribed decoding rule (greedy in our runs), before any parameter update. Index their responses by j and response tokens by i. For response y_j generated from history h_j, let $x_{ji} = (h_j, y_{j,<i})$. The two frozen skill views then rescore the same response prefixes, so their scores can be compared at each position:

$$p^+_{ji}(v) = \pi_0(v \mid x_{ji}, s^+_j), \qquad p^-_{ji}(v) = \pi_0(v \mid x_{ji}, s^-_j), \tag{10}$$

where $s^\pm_j$ are the skill versions for that response's task and p^-_{ji} is defined only when a failed predecessor exists. All signals are mapped back to the student's response coordinates and applied to parsed environment-action tokens $\mathcal{M}_{a,j}$. The counter view uses the same prompt template as the positive view and carries no failure label.

We use disagreement between these two views to give more training weight to actions whose scores change with the skill version. To measure this difference, let $\tilde{p}^+_{ji}$ and $\tilde{p}^-_{ji}$ be the teacher distributions, each normalized on its own top-K_{tok} support, then aligned on their union with zeros for missing tokens. For a record with a failed predecessor, we measure their normalized Jensen–Shannon distance

$$d_{ji} = \sqrt{\frac{\mathrm{JS}(\tilde{p}^+_{ji} \| \tilde{p}^-_{ji})}{\log 2}} \in [0, 1]. \tag{11}$$

To assign a common weight to tokens within an action, we average d_{ji} within each contiguous environment-action span and broadcast the mean $\bar{d}_{j,\mathrm{span}(i)}$ back to its action tokens. The evolution weight is

$$w_{ji} = \begin{cases} \mathrm{clip}\big(1 + \alpha \bar{d}_{j,\mathrm{span}(i)}, 1, w_{\max}\big), & s^-_j \text{ exists and } i \in \mathcal{M}_{a,j}, \\ 1, & \text{otherwise.} \end{cases} \tag{12}$$

Here $\alpha \geq 0$ controls the weight increase and $w_{\max} \geq 1$ caps it. The positive view supplies the target, while the failed view only affects the weight. If no failed predecessor is available, the positive-only record has $w_{ji} = 1$, giving the same token loss as positive-skill AgentOPD on the same responses.

Table 3: Source-task success rates (%) for skill-repair methods using the same actor and action interface. Dark and light shading mark the best and second-best values in each column.

	ALFWorld							WebShop					
Method	All	Pick	Pick2	Clean	Heat	Cool	Look	All	Fashion	Beauty	Elec.	Garden	Grocery
Single ReAct rollout	9.82	15.82	8.86	9.38	4.14	2.44	19.16	21.18	28.63	21.61	23.52	13.21	18.75
Repeated ReAct	10.22	16.33	9.59	9.38	4.58	2.63	19.48	22.14	30.93	22.37	23.95	13.89	19.32
One-shot skill v_0	13.62	17.59	16.73	11.38	10.02	5.07	20.13	24.54	31.76	25.82	25.08	16.73	22.85
Reflexion (Shinn et al., 2023)	**43.63**	53.42	42.07	**53.69**	31.81	**24.20**	52.60	**28.02**	**37.82**	**28.43**	**29.17**	18.18	**26.08**
ReasoningBank (Ouyang et al., 2026)	39.91	52.03	35.30	39.38	**36.17**	**24.02**	**55.19**	26.61	36.63	26.28	27.13	17.50	25.13
Skill-Pro (Mi et al., 2026)	37.18	**54.81**	**42.19**	33.85	22.88	15.76	44.16	26.24	34.88	26.62	27.88	16.88	24.65
ContractSkill (Lu et al., 2026b)	23.44	30.00	24.23	25.38	16.34	14.45	26.62	26.81	**37.36**	26.83	28.26	**18.32**	22.94
SKILL EVOLUTION (ours)	**46.27**	**61.90**	**43.17**	**49.23**	**39.87**	**24.20**	**55.84**	**28.96**	36.44	**30.74**	**29.44**	**20.68**	**26.94**

We then combine these weights with distillation from the positive view. For V_{ji}^{+}, the positive teacher's top-K_{tok} token set, we define the nonnegative truncated forward-KL contribution

$$D_{ji}^{+}(\theta) = \left[\sum_{v \in V_{ji}^{+}} p_{ji}^{+}(v) \log \frac{p_{ji}^{+}(v)}{\pi_\theta(v \mid x_{ji})} \right]_{+}. \tag{13}$$

Let $\mathcal{C}_n$ contain all retained responses from this batch and $N_n = \sum_{j \in \mathcal{C}_n} |\mathcal{M}_{a,j}|$ count their action tokens. Weighting each contribution and averaging over these action tokens gives, for $N_n > 0$, the batch loss

$$\mathcal{L}_{\text{EvoOPD}}^{(n)}(\theta) = \frac{\beta}{N_n} \sum_{j \in \mathcal{C}_n} \sum_{i \in \mathcal{M}_{a,j}} w_{ji} D_{ji}^{+}(\theta). \tag{14}$$

The denominator counts tokens across the whole batch, not per response or by total weight. Histories, generated tokens, teacher scores, and weights are held fixed during differentiation; only the student's response log probabilities receive gradients. One optimizer step uses this accumulated batch gradient, without differentiating through rollout generation. The coefficient β scales distillation; our runs use $\beta = \alpha = 1$, $w_{\max} = 2$, and $K_{\text{tok}} = 32$. Responses lacking action masks or teacher coverage are discarded. After training, evaluation removes all skills and teachers to test the student's skill-free performance. Algorithm 1 gives the batch procedure, and Appendix E.2 analyzes this objective.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate on ALFWorld for grounded household manipulation (Shridhar et al., 2021) and WebShop for search and purchase decisions (Yao et al., 2022). Evolution covers all 3,553 ALFWorld and all 10,587 WebShop training tasks. Transfer uses the held-out ALFWorld splits of 140 seen and 134 unseen tasks—seen instances reuse training scenes with new goals, unseen instances come from scenes absent from training—and the 500-task WebShop test set; skill-free internalization uses the two ALFWorld splits only. Every held-out task is disjoint from every source task. We report the six native ALFWorld task families and the official WebShop product category of the target item.

Baselines. All methods use the same frozen Qwen3-8B checkpoint (Yang et al., 2025) under one execution protocol: the source-task repair baselines of Table 3, skill-free execution for transfer, and the internalization methods of Table 4, none of which use RL. Each external method is a component adaptation under this shared protocol rather than its original implementation; Appendix G states every deviation. Every row uses a zero-shot ReAct prompt without the few-shot demonstrations of the original ALFWorld setup, so absolute rates are not directly comparable to published few-shot numbers. Appendix F gives the full protocol.

Metrics. The primary metric is task success rate (%). For WebShop we also report the official task score, which credits partial attribute and option matches and therefore separates buying more often from buying better. Internalization is reported as the mean over training seeds 42–44 with its sample standard deviation; SKILL EVOLUTION and SKILL TRANSFER are single runs, so we attach no interval to them. Every reported number is bound to an immutable run manifest.

Table 4: Skill-free internalization on ALFWorld. Trained methods report mean success rates (%) over three seeds, with standard deviations below. All rows use the internalization protocol of Appendix F.2, whose step cap and renderer differ from the transfer setting of Table 2.

	ALFWorld Seen							ALFWorld Unseen						
Training method	All	Pick	Pick2	Clean	Heat	Cool	Look	All	Pick	Pick2	Clean	Heat	Cool	Look
Frozen base model	7.86	14.29	4.17	0.00	6.25	0.00	30.77	4.48	0.00	0.00	0.00	0.00	0.00	33.33
	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Vanilla AgentOPD	51.90	76.19	**65.28**	41.98	**41.67**	**17.33**	**61.54**	**60.95**	**75.00**	62.75	**58.06**	63.77	31.75	**75.93**
	±2.18	±1.65	±2.41	±5.66	±3.61	±6.11	±0.00	±3.68	±7.22	±3.40	±3.23	±5.02	±11.98	±3.21
SOD (Zhong et al., 2026)	49.76	80.95	63.89	32.10	33.33	**18.67**	**56.41**	57.46	66.67	54.90	50.54	**68.12**	**34.92**	72.22
	±1.80	±1.65	±2.41	±2.14	±13.01	±4.62	±4.44	±2.69	±4.17	±3.40	±7.45	±2.51	±2.75	±0.00
TCOD-F2B (Wang et al., 2026b)	50.00	76.19	61.11	40.74	**43.75**	16.00	51.28	57.21	69.44	**66.67**	46.24	66.67	30.16	70.37
	±3.27	±7.19	±2.41	±3.70	±0.00	±4.00	±4.44	±4.37	±6.36	±3.40	±6.72	±10.04	±5.50	±3.21
TCOD-B2F (Wang et al., 2026b)	50.48	72.38	62.50	**46.91**	37.50	**17.33**	**56.41**	55.97	70.83	56.86	54.84	52.17	30.16	72.22
	±6.44	±10.03	±0.00	±11.32	±12.50	±2.31	±4.44	±2.99	±4.17	±12.25	±8.53	±4.35	±7.27	±0.00
FutureBridge-OPD (Chen et al., 2026)	**54.29**	**82.86**	**65.28**	**46.91**	**43.75**	14.67	**61.54**	58.21	**73.61**	62.75	**58.06**	52.17	28.57	**75.93**
	±4.34	±8.57	±2.41	±10.69	±6.25	±2.31	±0.00	±2.69	±2.41	±3.40	±8.53	±4.35	±4.76	±3.21
EvoOPD (ours)	**54.52**	**84.76**	**66.67**	**45.68**	37.50	**17.33**	**61.54**	**64.18**	**75.00**	**64.71**	**62.37**	**71.01**	**38.10**	**74.07**
	±4.19	±1.65	±4.17	±7.71	±10.83	±9.24	±0.00	±3.25	±4.17	±5.88	±4.93	±2.51	±12.60	±3.21

5.2 Main Results

Skill Evolution. The final evolved skills reach 46.27% success on ALFWorld and 28.96% on WebShop (Table 1). On WebShop, the initial skill raises success from 21.18% to 24.54%, and further revisions add 4.42 percentage points. All five product categories improve over the initial skill, with gains of 3.95–4.92 points. The gains therefore span categories rather than coming from a single subset. More tasks are fully completed, but the mean WebShop score is 51.25 after evolution, compared with 53.10 without skills.

Figure 3 shows that ALFWorld continues to gain certificates across later revisions, while WebShop saturates earlier and more often ends without an admissible patch.

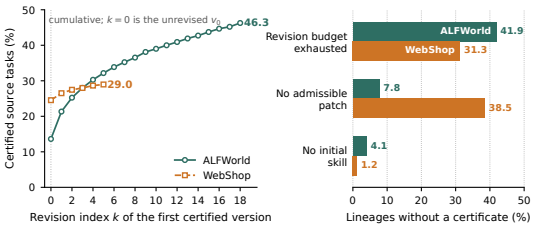

Figure 3: Certification across revisions and unresolved lineage outcomes.

The repair baselines in Table 3 help separate skill updates from repeated attempts. Repeated ReAct changes success only from 9.82% to 10.22% on ALFWorld and from 21.18% to 22.14% on WebShop. With a 20-rollout cap on ALFWorld, Skill Evolution reaches 46.27%, compared with 43.63% for Reflexion and 39.91% for ReasoningBank. It uses 12.61 completed rollouts per task, compared with 12.68 and 14.31, respectively. On WebShop, Skill Evolution reaches 28.96%, compared with 28.02% for Reflexion and 26.81% for ContractSkill. It also uses fewer completed rollouts per task: 4.81, compared with 5.47 and 5.50, respectively (Table 19).

Skill Transfer. Frozen skills transfer well to held-out ALFWorld tasks (Table 2). Retrieval raises success from 6.43% to 37.86% on Seen and from 5.22% to 35.07% on Unseen, without changing the bank. Local repair then reaches 48.57% and 55.22%, adding 10.71 and 20.15 points beyond retrieval, with additional test-time rollouts. Thus the source skills help on new tasks, while further repair addresses failures that retrieval alone does not resolve. WebShop shows a different pattern: the frozen bank changes success only from 22.60% to 22.80% while lowering the mean task score from 53.83 to 46.04, whereas local repair reaches 30.20% and recovers the score to 52.04. The additional fully completed WebShop tasks therefore come from task-local adaptation rather than direct reuse.

Skill Internalization. With skills removed at evaluation, our model reaches the highest mean success in Table 4: 54.52 ± 4.19 on Seen and 64.18 ± 3.25 on Unseen. The margin is largest on the unseen split, where it improves over Vanilla AgentOPD, the strongest baseline there, by 3.23 percentage points, and over every other method by 5.97 to 8.21 points. On Seen it is 0.24 points above FutureBridge-OPD and 2.62 above Vanilla AgentOPD. Appendix F.5 reports paired intervals for these comparisons across the three seeds.

The gains are not uniform across task families. On Unseen, the largest improvements over Vanilla AgentOPD are on Heat and Cool, at 7.25 and 6.35 points, while Look does not improve. Together,

Table 5: Component ablations across six ALFWorld task families. Higher success rates (%) within each full–ablation pair are shaded and bold; EvoOPD reports means and standard deviations over three seeds. Setting every action-token weight to one gives the Vanilla AgentOPD training objective, so the w/o Contrast column is that run of Table 4.

	Skill Evolution (Source)		Skill Transfer (Seen)		Skill Transfer (Unseen)		EvoOPD (Seen)		EvoOPD (Unseen)	
Family	Full	w/o Loc.+Edit	Full	w/o Router	Full	w/o Router	Full	w/o Contrast	Full	w/o Contrast
All	**46.27**	18.27	**37.86**	35.71	35.07	**38.06**	**54.52**±4.19	51.90±2.18	**64.18**±3.25	60.95±3.68
Pick	**61.90**	23.04	**54.29**	51.43	**45.83**	33.33	**84.76**±1.65	76.19±1.65	**75.00**±4.17	**75.00**±7.22
Pick2	**43.17**	20.05	25.00	**37.50**	23.53	**35.29**	**66.67**±4.17	65.28±2.41	**64.71**±5.88	62.75±3.40
Clean	**49.23**	16.92	**48.15**	40.74	32.26	**54.84**	**45.68**±7.71	41.98±5.66	**62.37**±4.93	58.06±3.23
Heat	**39.87**	15.47	**37.50**	25.00	**26.09**	13.04	37.50±10.83	**41.67**±3.61	**71.01**±2.51	63.77±5.02
Cool	**24.20**	6.19	8.00	**12.00**	**38.10**	**38.10**	**17.33**±9.24	**17.33**±6.11	**38.10**±12.60	31.75±11.98
Look	**55.84**	29.22	**53.85**	38.46	44.44	**50.00**	**61.54**±0.00	**61.54**±0.00	74.07±3.21	**75.93**±3.21

these results show that certified skills can support model learning that remains useful when skill context is removed at test time.

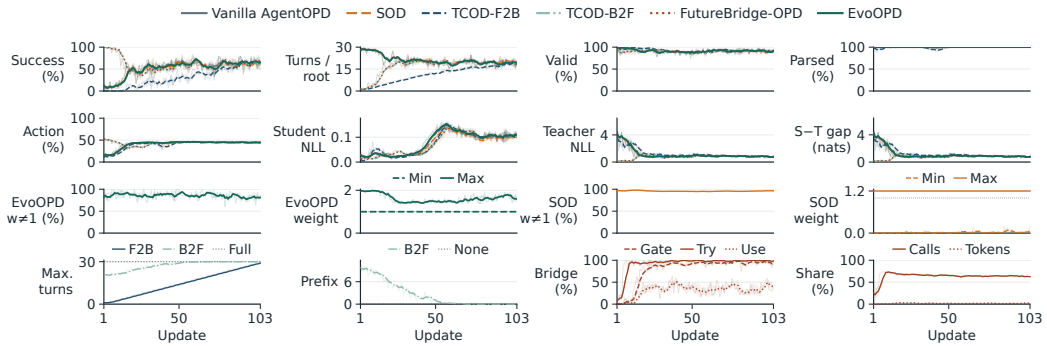

Figure 4: ALFWorld training dynamics: behavior and action-level fit in the upper two rows; weighting, curricula, and bridge validation in the lower two. Lines show five-update means across three seeds and shading shows one sample standard deviation; Appendix H gives metric definitions and additional diagnostics.

Figure 4 shows that training changes more than response format: parsing is already nearly perfect at the start, yet training success rises from 10.8% to 65.0% over the first and last five updates and the mean student–teacher action gap falls from 3.52 to 0.77 nats. The non-unit weight share still averages 81.1% late in training, so the contrast stays active as the student approaches the teacher. Skill-free generalization and the contrast's contribution are measured in Tables 4 and 5.

5.3 Analysis and Ablations

Table 5 examines guided editing, skill routing, and failed-version weighting on ALFWorld, with each component evaluated at the stage where it is used. On source tasks, removing failure localization and its action-grounded editing tools, while retaining the task set, revision budget, and validation, lowers success from 46.27% to 18.27%. This supports their joint use in skill revision.

For transfer, we replace the LLM router with the top-ranked candidate from the same coarse retriever, keep the bank frozen, and allow no local repair. The router changes Seen success from 35.71% to 37.86% but lowers Unseen success from 38.06% to 35.07%, so reranking is not consistently useful.

For internalization, setting all action-token weights to one gives the Vanilla AgentOPD training objective, so the w/o Contrast column is that run of Table 4. Mean success drops from 54.52% to 51.90% on Seen and from 64.18% to 60.95% on Unseen, indicating that the failed version helps locate where the student should learn.

AI USE STATEMENT

Generative AI tools were used to polish the paper and assist with experimental code development.

ETHICS STATEMENT

All experiments use simulated ALFWorld and WebShop tasks; they involve no human participants or real purchases. A successful benchmark rollout certifies task completion under its evaluation protocol, but does not establish that a skill is safe or appropriate for real-world use. Deploying reusable skills in consequential settings would require additional safeguards and human oversight.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, pp. 21246–21263, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5be69a584901a26c521c2b51e40a4c20-Paper-Conference.pdf.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, pp. 8479–8565, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0e9e708b6f48e14fd0ac29e167413f76-Paper-Conference.pdf.

Chishui Chen, Yaoyou Fan, Te Sun, Yi Yang, Chenghao Sun, Delin Mao, Hongbo Qiao, Zuowei Zhang, Junxi Wang, Chenxing Sun, Yangen Hu, Lu Pan, Xuyang Liu, and Linfeng Zhang. Look ahead before you distill: Future trajectory validation of teacher guidance for agentic on-policy distillation, 2026. URL https://arxiv.org/abs/2608.01953.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Longtao Hu, Xiao Liang, and Linchao Zhu. From interaction traces to persistent skills: Online evolution for computer-use agents, 2026. URL https://arxiv.org/abs/2609.04869.

Mykhailo Kozyrev, Andrei Kozyrev, and Anton Podkopaev. Skill issue: Lessons from optimizing repository SKILLs for coding agents, 2026. URL https://arxiv.org/abs/2609.12742.

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, 2026. URL https://arxiv.org/abs/2602.12670.

Pingchen Lu, Xiangyi Wang, Xiang Li, Jie Mao, Zikun Qu, Junfeng Luo, Yao Shu, Bryan Kian Hsiang Low, and Zhongxiang Dai. COBRA-Skills: Contextual bandit-guided evolution for agent skill optimization, 2026a. URL https://arxiv.org/abs/2609.11682.

Zijian Lu, Yiping Zuo, Yupeng Nie, Xin He, Weibei Fan, Lianyong Qi, and Shi Jin. ContractSkill: Repairable contract-based skills for multimodal web agents, 2026b. URL https://arxiv.org/abs/2603.20340.

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. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.01869.

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, 2026. URL https://arxiv.org/abs/2603.25158.

Frank Nielsen and Richard Nock. On the chi square and higher-order chi distances for approximating f-divergences. *IEEE Signal Processing Letters*, 21(1):10–13, 2014. doi: 10.1109/LSP.2013.2288355. URL https://doi.org/10.1109/LSP.2013.2288355.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, pp. 94327–94354, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/980ea04d23d1f6908964eba2a74afe45-Paper-Conference.pdf.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Liyan Tang, Cyrus Rashtchian, Chun-Sung Ferng, Andrew Tomkins, Da-Cheng Juan, and Tu Vu. WikiSkill: Compiling agent experience into persistent knowledge for skill evolution, 2026. URL https://arxiv.org/abs/2608.27454.

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. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and Honggang Qi. Skill-SD: Skill-conditioned self-distillation for multi-turn LLM agents, 2026a. URL https://arxiv.org/abs/2604.10674.

Jiaqi Wang, Wenhao Zhang, Weijie Shi, Yaliang Li, and James Cheng. TCOD: Exploring temporal curriculum in on-policy distillation for multi-turn autonomous agents, 2026b. URL https://arxiv.org/abs/2604.24005.

Yixuan Wang, Yiyang Zhou, Yiming Liang, Congyu Zhang, Fuxiao Liu, Jiawei Zhou, and Huaxiu Yao. Not all skills help: Measuring and repairing agent knowledge, 2026c. URL https://arxiv.org/abs/2606.15390.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SEED: Self-evolving on-policy distillation for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2607.14777.

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. URL https://arxiv.org/abs/2505.09388.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, and Jianhua Tao. OPID: On-policy skill distillation for agentic reinforcement learning, 2026a. URL https://arxiv.org/abs/2606.26790.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. SkillOpt: Executive strategy for self-evolving agent skills, 2026b. URL https://arxiv.org/abs/2605.23904.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022. doi: 10.52202/068431-1508. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025. doi: 10.1038/s41586-025-08661-4. URL https://www.nature.com/articles/s41586-025-08661-4.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, pp. 86069–86100, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/8a94ff6f922d995d7d3f4ebf4143e442-Paper-Conference.pdf.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38 (17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2601.18734.

Qiyong Zhong, Mao Zheng, Mingyang Song, Xin Lin, Jie Sun, Houcheng Jiang, Xiang Wang, and Junfeng Fang. SOD: Step-wise on-policy distillation for small language model agents, 2026. URL https://arxiv.org/abs/2605.07725.

APPENDIX CONTENTS

A Conclusion and Limitations **15**

A.1 Conclusion 15

A.2 Limitations 15

B Skill schema and editor protocol **15**

B.1 Data and sampling for Figure 1 16

B.2 Example of a single-rule repair 16

B.3 Patch rejection and termination 17

C Environment adapters and execution prompts **17**

D Generalization, deduplication, and routing details **18**

D.1 Negative transfer on ALFWorld 19

D.2 WebShop transfer: success and partial credit 19

E EvoOPD derivation and implementation details **20**

E.1 One training step 21

E.2 What the evolution weight changes 21

E.3 Why the square root responds to small teacher differences 22

E.4 The gradient error from keeping only top tokens 23

F Complete experimental protocol and additional results **23**

F.1 Stage-specific evaluation 23

F.2 Skill-free internalization settings 23

F.3 Shared source-task execution settings 24

F.4 Revision depth and interaction cost 24

F.5 Paired performance comparisons 25

G Baseline implementations and reproducibility **26**

G.1 Shared source-task settings 26

G.2 Source-task baseline implementations 26

G.3 Internalization data and shared training procedure 28

G.4 Internalization baseline implementations 29

G.5 Training cost and implementation details 30

H Training dynamics and computational cost **30**

H.1 Optimization stability and interpretation 30

H.2 Computational cost and action supervision 31

I Case study: learning beyond a certified trajectory **32**

I.1 Source certification and student evaluation 32
I.2 Different students, identical positive supervision 32
I.3 Shorter successful routes without copying the reference 33

A Conclusion and Limitations

A.1 Conclusion

We presented Evolution Learning, which treats a skill as a revisable hypothesis rather than a finished summary of experience. Environment feedback guides edits, and successful execution certifies each accepted version. Certified skills then support transfer to new tasks and skill-free model learning. Skill Evolution reaches 46.27% source-task success on ALFWorld and 28.96% on WebShop. Frozen skills improve held-out ALFWorld performance, while most of the WebShop transfer gain comes from task-local repair.

EvoOPD uses certified skills as supervision and failed skill versions to weight the actions being learned. With no skills at evaluation, it reaches mean success rates of 54.52% on ALFWorld Seen and 64.18% on Unseen over three training seeds, with its largest margin on the unseen split, 3.23 points above the strongest baseline there. These results support environment-verified skill changes as a link between agent experience and model learning. Limitations are discussed in Appendix A.2.

A.2 Limitations

Our study covers two benchmarks, ALFWorld and WebShop, with Qwen3-8B as the base model, and evaluates skill-free internalization on ALFWorld only. Whether the same lifecycle carries over to other model scales, to open-ended environments, or to WebShop internalization is left open.

Skill Evolution and Skill Transfer are reported as single runs, and EvoOPD over three training seeds.

Verification also has a price. Every proposed revision consumes another episode, the editor and the router each require model calls, and EvoOPD rescores student actions under two skill-conditioned teacher views. Making verification cheaper without weakening it is the natural next step.

B Skill schema and editor protocol

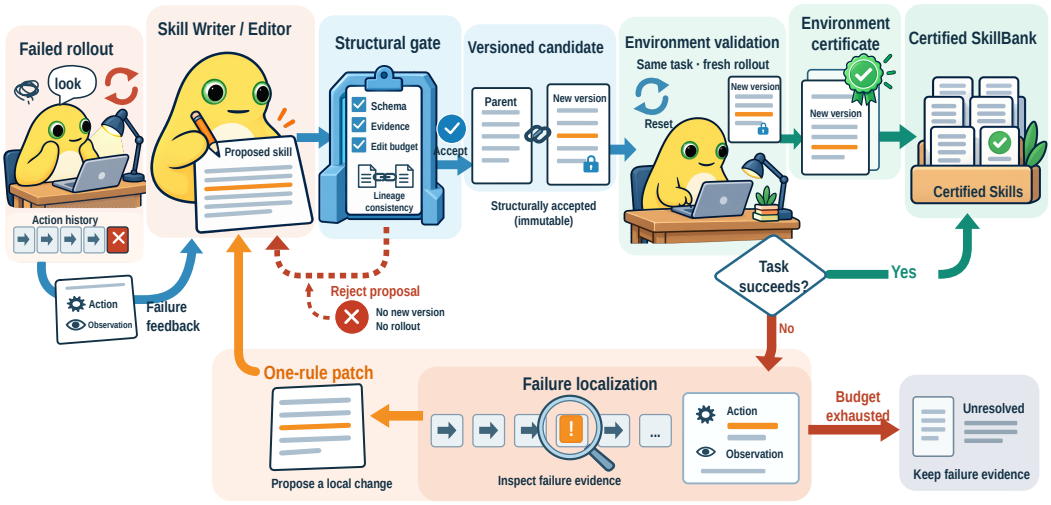

Figure 5: Skill evolution under structural and environmental validation. The Harness checks the schema, evidence, edit budget, and lineage consistency before creating an immutable candidate version. A fresh same-task rollout either certifies the candidate, returns localized failure evidence for another one-rule patch, or keeps the lineage unresolved when the revision budget ends.

A skill document carries six typed sections—preconditions, procedure, actions to avoid, corner cases, recovery, and completion checks—and every rule inside a section holds a stable identifier. The identifier is what makes a revision local: an accepted patch names one rule, and the Harness rewrites that rule alone, leaving all other rules unchanged.

B.1 Data and sampling for Figure 1

Figure 1 combines full-source success counts with a paired analysis of existing ALFWorld trajectories. The success bars use all 3,553 source tasks from the source-task run in Table 1. The initial skill succeeds on 484 tasks (13.62%), and a certified skill is obtained on 1,644 tasks (46.27%). Each task contributes once to each count. These bars summarize the full run; they are not estimated from the 60 trajectory pairs used in the action-count panel.

For the paired analysis, we consider tasks whose initial skill failed and a later revision was certified. Within each of the six task families, we sort task identifiers by the SHA-256 hash of the sampling seed and task identifier, using seed 20260907, and select the first ten. This gives 60 distinct tasks. For each task, we compare the first certified version with its immediately preceding failed parent, not necessarily with the initial skill. Each pair contains the observed rollouts of both versions and a single-rule revision.

The action-count panel plots the empirical cumulative distribution of the number of environment actions in each recorded rollout: at action count a, the curve gives the fraction of the 60 rollouts with at most a actions. All failed parents use the full 20-step cap; their certified children average 10.88 actions. In 54 pairs, the failed parent repeats at least one action. The illustrated laptop case is one of these pairs: replacing one rule changes a 20-action failure into a four-action success. Equal sampling across task families does not reflect their frequencies in the full dataset, and the pairs are selected because repair eventually succeeded. The curves therefore describe the repairs observed in this sample.

B.2 Example of a single-rule repair

Table 6 shows an ALFWorld task solved after one skill revision. We select this example from the stratified sample because its first revision succeeds ($k^* = 1$). The initial skill repeats an action that leaves the environment unchanged until the step budget is exhausted. Using the twenty transitions in this failed rollout, the editor replaces one procedure rule. The revised skill then solves the task in five actions.

Table 6: A certified evolution transition on one ALFWorld source task. Actions are reproduced verbatim from the recorded rollouts. Evidence types are the labels the locator assigned to the twenty transitions of the failed trajectory.

Task Pick family	
Instruction	`put a candle in toilet.`
Skill-free outcome	Failure
Version v_0 failed execution	
Rule `procedure-0001`	Look around to confirm the location of the drawer containing the candle.
Actions (20 of 20)	`look` × 20
Locator evidence	1 no-op, 18 repeated-action, 1 terminal-failure-with-progress (20 records)
Failure mode	Repeated, state-preserving action loop; step budget exhausted
Patch one `replace` operation on section `procedure`, rule `procedure-0001`	
New rule text	When the task is incomplete and `go to drawer 1` is present in the current admissible actions, execute the admissible action `go to drawer 1` immediately; do not repeat `look`.
Version v_1 certified execution	
Actions (5)	`go to drawer 1` → `open drawer 1` → `take candle 2 from drawer 1` → `go to toilet 1` → `move candle 2 to toilet 1`
Repeated actions	0

B.3 PATCH REJECTION AND TERMINATION

The Harness rejects malformed edits, no-op paraphrases, duplicate rules, and returns to an earlier skill version. It gives the editor the reason for each rejection so that another patch can be proposed. Table 7 reports rejection frequency and the reasons skill evolution ends. WebShop has 2.27 rejected patches per task, compared with 0.53 on ALFWorld. Failure to produce an admissible patch is its most common termination condition.

Table 7: Harness refusals and lineage termination on the source-task runs. Counts are over all completed tasks in each run; percentages share those denominators. A lineage that terminates on refusal inserts nothing into the bank.

	ALFWorld	**WebShop**
Completed source tasks	3,553	10,587
Refused patches per task	0.53	2.27
Tasks with ≥ 1 refusal	843 (23.7%)	7,429 (70.2%)
Largest number of refusals on one task	8	8
Lineage certified	1,644 (46.3%)	3,066 (29.0%)
Terminated: revision budget exhausted	1,489 (41.9%)	3,314 (31.3%)
Terminated: no admissible patch	276 (7.8%)	4,079 (38.5%)
Terminated: no initial skill produced	144 (4.1%)	128 (1.2%)

C ENVIRONMENT ADAPTERS AND EXECUTION PROMPTS

The source-task baseline adapters share a renderer and parser on each benchmark. The internalization runs use a separate, shared ALFWorld renderer for training and evaluation. Table 8 states the interface settings, and the displays below distinguish these stage-specific prompts. Section G specifies the method adaptations, reference rows, and differences in interaction budgets.

Table 8: Environment adapter settings. Source-task execution and trained internalization use the stage-specific step caps below. Curriculum methods can use a shorter student horizon within the cap.

	ALFWorld	**WebShop**
Response protocol	`react_xml`: one `<think>...</think>` block followed by exactly one `<action>...</action>` block	Identical
Environment step cap	20 for source-task evolution and transfer; 30 for internalization	15
Visible history window	20 executed action–feedback pairs	2 executed action–feedback pairs
Action presentation	The full ordered admissible-command list is appended to the current observation	The ordered clickable list is appended to the current observation
Observation handling	Score and reward metadata are omitted from actor prompts; the per-turn layout depends on the stage	Score/reward fields and repeated page instruction headers are omitted
Parser	`legacy_tolerant`: strict lower-case XML first, then a bracketed fallback	Identical
Unparsable response	An empty action that consumes one environment interaction	Identical
Skill injection	Appended to the system message; removed entirely for skill-free evaluation	Identical

Table 9: ALFWorld system message, verbatim.

```
You are an expert agent operating in the ALFRED Embodied Environment.
At each step, first reason step-by-step about the current situation.
The reasoning process MUST be enclosed within literal angle-bracket tags:  <think>
</think>.
After reasoning, choose exactly one admissible action for the current step.
The action MUST be enclosed within literal angle-bracket tags:  <action> </action>.
For example, your output should be like this:
<think>I need to inspect the current location first.</think><action>look</action>
Do not output anything before <think> or after </action>.
```

Table 10: WebShop system message, verbatim.

```
You are an expert autonomous agent operating in the WebShop e-commerce environment.
You should first reason step-by-step about the current situation, then think carefully
which admissible action best advances the shopping goal.
This reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current
step and present it within <action> </action> tags.
For example:
<think>I should search for a product using the user's required
attributes.</think><action>search[wireless mouse]</action>
Do not output anything before <think> or after </action>.
```

Table 11: ALFWorld source/transfer current-turn user template. The initial message uses the reset observation and step 1. Later turns retain the initial message, recent assistant responses and observation messages, and this current message. The external Table 3 adapters use this renderer.

```
Your task is to: {task description}
You are now at step {step number} and your current observation is: {observation}
Your admissible actions of the current situation are: [{ordered commands,
comma-separated}].
Now it's your turn to take an action.
```

Table 12: WebShop source/transfer current-turn user template. The history layout follows the same initial/recent/current message structure, with two recent action–feedback pairs.

```
Your task is to: {task description}.
You are now at step {step number} and your current observation is: {observation}.
Your admissible actions of the current situation are:
[
{ordered actions, one per line}
].
Now it's your turn to take one action for the current step.
```

Table 13: ALFWorld internalization user templates, used by all six trained methods in both training and evaluation. The reset message is retained; assistant/feedback pairs are truncated to the latest 20, followed by the current observation and full ordered command list.

```
Initial environment state:
{initial observation}
Observation:
{observation after each executed action}
Current observation:
{current observation}
Available commands: {ordered admissible commands, semicolon-separated}
Choose one action using <think>...</think><action>...</action>.
```

D GENERALIZATION, DEDUPLICATION, AND ROUTING DETAILS

On ALFWorld, Skill Evolution was run on the 3,553 source tasks under two revision budgets. The five-revision run certifies 1,160 skills, and the eighteen-revision run certifies 1,644, which is the 46.27% reported in Table 1. The transfer bank is curated from the five-revision certificates: 1,160 skills enter curation and 653 are retained after the gates of Table 14. Internalization instead draws from the eighteen-revision certificates, selecting 822 of the 1,311 lineages that also have a failed skill-free baseline (Section G.3). The transfer bank and the training set are therefore different subsets of the two certified pools, and the transfer numbers are obtained with the smaller certified pool.

We construct the frozen bank by filtering and merging source-certified skills. Table 14 reports the number retained at each stage for the 1,160 ALFWorld skills certified under a five-revision budget. The WebShop bank follows the same procedure, curated from the 3,066 skills certified on its 10,587 source tasks and reduced to 420 skills after generalization and validation-gated deduplication. Two stages account for most of the reduction. Source-regression certification removes 313 of 1,086 generalized proposals, that is, in 28.8% of cases rewriting instance constants into semantic roles

destroyed the utility the original skill had on its own source task. Validation-gated deduplication then removes 120 near-duplicate groups, but only after a representative completed a real rollout on another member's source task; 54 of the 174 attempted absorptions failed that test and both members were retained. Textual similarity alone never merges two skills.

Table 14: Construction of the frozen ALFWorld skill bank. Each row reports entries entering the stage, entries the stage removes, and entries retained. Removals are decided by environment rollouts wherever the Gate column names one.

Stage	Operation	Gate	Removed	Retained
1	Source-certified skills entering curation	—	—	1,160
2	Goal-consistency screen	Deterministic	47	1,113
3	Exact-duplicate collapse after role normalization	Deterministic	1	1,112
4	Near-duplicate absorption by a representative	**Environment**	26	1,086
Active bank				**1,086**
5	Rewrite instance constants into semantic roles	—	—	1,086
6	Source-regression certification of the rewrite	**Environment**	313	773
7	Post-generalization near-duplicate absorption	**Environment**	120	653
Frozen bank used for transfer				**653**

Cross-task validation rollouts: 34 at stage 4 (27 passed, 7 failed);
174 at stage 7 (120 passed, 54 failed). A failed absorption retains both members.

D.1 Negative transfer on ALFWorld

Table 15: Paired skill-free and retrieved-skill outcomes on the held-out ALFWorld splits, using the frozen bank of 653 skills. Rows are tasks, not task–seed pairs; the four cells partition each split exactly. *Broken* counts tasks the skill-free agent solved and the retrieved skill did not, which is the negative-transfer event.

	Seen (n=140)	**Unseen** (n=134)
Router abstentions	0	0
Tasks receiving a candidate	140	134
Both succeed	8	5
Repaired: skill only	**45**	**42**
Broken: skill-free only	1	2
Neither succeeds	86	85
Negative transfer, % of candidates	0.7	1.5
Recovered later by task-local repair	15	27

Because the skill-free outcome and the retrieved-skill outcome are collected on the same held-out task under the same specification, transfer can be scored as a paired contingency rather than as a difference of two rates. Table 15 gives that contingency. The router abstained on no task, so every held-out task received exactly one candidate and the retrieval column is complete. Retrieval converted 45 seen and 42 unseen failures into successes while breaking 1 seen and 2 unseen tasks that the skill-free agent had already solved, so negative transfer is 0.7% and 1.5% of retrieved tasks.

D.2 WebShop transfer: success and partial credit

Table 16 reports exact success and the mean official task score on the same 500 held-out WebShop tasks. Frozen retrieval changes success from 22.60% to 22.80%, but lowers the mean score from 53.83 to 46.04. Local repair raises success to 30.20% and recovers part of the score loss, reaching 52.04. Thus, more tasks meet every requirement after repair, but average partial-credit performance remains below skill-free execution.

The small net change under frozen retrieval hides substantial changes in which tasks succeed. Retrieval solves 38 tasks that the skill-free agent fails, but fails on 37 tasks that it previously solved. Both settings succeed on 76 tasks and fail on 349; these counts include skill-free fallback on the 50 tasks without a retrieved candidate. After local repair, 19 tasks that the skill-free agent solved still fail. Checking both gains and regressions is therefore important: the higher final success rate alone does not capture the change in partial-credit performance.

Table 16: WebShop transfer on 500 held-out tasks. Success requires a complete match; the mean official score gives partial credit and is scaled to 0–100.

Setting	Success (%)	Mean score
No skill	22.60	53.83
Frozen bank + router	22.80	46.04
+ Local repair	30.20	52.04

E EvoOPD derivation and implementation details

The weight in Equation 12 uses the Jensen–Shannon *distance* rather than the divergence. For two teacher views that differ only slightly the divergence is second order in that difference, so $1 + \alpha\,\mathrm{JS}$ rounds back to exactly 1 in single precision on a large fraction of action tokens, leaving their weights unchanged. The square root is bounded, symmetric, and a metric, and it recovers a usable dynamic range without introducing a reward or a policy-gradient term.

Whether that matters is an empirical question, and Table 17 and Figure 6 answer it for the run reported in Table 4. Pooling over all optimized action tokens of a run, the weight is active on roughly 86% of them, whereas the corresponding last-five-update mean plotted in Figure 4 is 81.1%. The per-update maximum sits near 1.5 rather than at the clipping ceiling: the ceiling binds in 6–11% of updates depending on the seed. The signal is therefore driven by measured teacher disagreement, with clipping affecting a minority of updates.

Table 17: Evolution weights over the three internalization seeds. *Non-unit* counts optimized action tokens with $w_t > 1$, that is, tokens where a failed predecessor existed and the two teacher views disagreed. Medians are taken over the 103 updates of a seed.

	Seed 42	**Seed 43**	**Seed 44**
Optimization updates	103	103	103
Optimized action tokens	90,522	90,826	91,956
Non-unit weight tokens, % of all	85.6	86.8	85.6
Median per-update non-unit share, %	89.4	89.7	89.9
Median per-update largest w_t	1.47	1.47	1.53
Updates reaching the ceiling $w_{\max}$=2, %	5.8	10.7	7.8

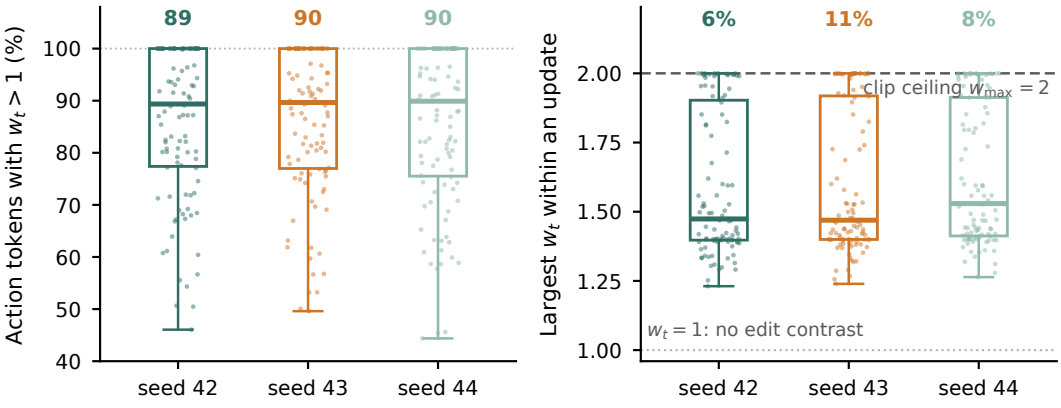

Figure 6: Distribution of evolution weights during training. **Left**, share of optimized action tokens carrying $w_t > 1$, one point per optimization update (n=103 per seed); boxes give the median and interquartile range, whiskers span the full observed range, and the number above each box is the median. **Right**, largest w_t within an update, on the same points; the dashed line is the clipping ceiling $w_{\max} = 2$, the dotted line is $w_t = 1$ at which the counter view contributes nothing, and the percentage above each box is the share of updates that reach the ceiling. Horizontal offsets are jitter for visibility and carry no information.

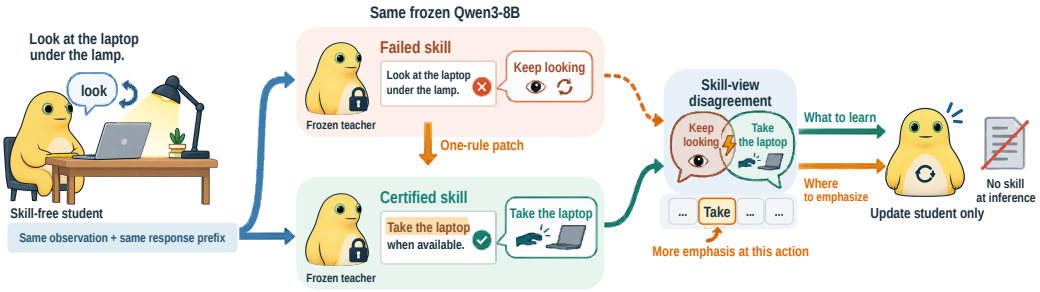

Figure 7: Illustration of skill-conditioned supervision in EvoOPD. The certified view supplies the distillation target, while disagreement between skill views weights the student's action-token loss.

E.1 One training step

Figure 7 illustrates the supervision in one training step. Algorithm 1 states the procedure of Section 4.4 for one update batch. The student first collects eight root episodes without changing its parameters. All retained responses share one action-token denominator and contribute to one optimizer step. The frozen checkpoint scores each response under its task's certified and failed skill views. Only the certified view supplies targets; a record without a failed predecessor uses unit weights.

Algorithm 1 One batch update of EvoOPD.

Require: current student π_{θ_n}; frozen teacher π_0; batch of tasks with certified records; $K_{\text{tok}}, \alpha, \beta, w_{\max}$

Verify certificates and extract each task's s^+ and optional s^-
Collect skill-free episodes using π_{θ_n} and the fixed decoding rule
Retain valid responses $\mathcal{C}_n$ with action masks and teacher coverage
Parse $\mathcal{M}_{a,j}$ for each response; set $N_n \leftarrow \sum_j |\mathcal{M}_{a,j}|$
if $N_n = 0$ **then**
    return without an optimizer update
end if
for each response $j \in \mathcal{C}_n$ **do**
    Set $x_{ji} \leftarrow (h_j, y_{j,<i})$ for each token position i
    Score $p^+_{ji}(v) \leftarrow \pi_0(v \mid x_{ji}, s^+_j)$; retain top-$K_{\text{tok}}$ set V^+_{ji}
    if s^-_j exists **then**
        Score the same prefixes with $p^-_{ji}(v) \leftarrow \pi_0(v \mid x_{ji}, s^-_j)$
        Normalize each teacher's top-K_{tok} support; align on the union with zeros
        $d_{ji} \leftarrow \sqrt{\mathrm{JS}(\tilde{p}^+_{ji} \| \tilde{p}^-_{ji}) / \log 2}$
        Average d_{ji} within each action span to obtain $\bar{d}_{j,\mathrm{span}(i)}$
        $w_{ji} \leftarrow \mathrm{clip}(1 + \alpha \bar{d}_{j,\mathrm{span}(i)}, 1, w_{\max})$
    else
        $w_{ji} \leftarrow 1$
    end if
    Detach prefixes, teacher scores, and weights from the student gradient
    $D^+_{ji}(\theta) \leftarrow \left[\sum_{v \in V^+_{ji}} p^+_{ji}(v) \log \frac{p^+_{ji}(v)}{\pi_\theta(v|x_{ji})}\right]_+$
end for
$\mathcal{L}^{(n)} \leftarrow \frac{\beta}{N_n} \sum_{j \in \mathcal{C}_n} \sum_{i \in \mathcal{M}_{a,j}} w_{ji} D^+_{ji}(\theta)$
Take one optimizer step at $\theta = \theta_n$; keep π_0 frozen

Ensure: updated skill-free student $\pi_{\theta_{n+1}}$

E.2 What the evolution weight changes

We analyze one fixed update batch, with histories, response tokens, teacher scores, top-token sets, and weights held constant. These are local properties of the training loss, not guarantees of task

success. In this section, u indexes an included action token across the batch, p_u is its positive teacher distribution, and $q_u = \mathrm{softmax}(z_u)$ is the student distribution over the full vocabulary $\mathcal{V}$. Define

$$\widehat{D}_u = \sum_{v \in S_u} p_u(v) \log \frac{p_u(v)}{q_u(v)}, \qquad m_u = \sum_{v \in S_u} p_u(v), \qquad \bar{p}_u(v) = \frac{p_u(v)\mathbf{1}\{v \in S_u\}}{m_u}, \tag{15}$$

where S_u is the positive teacher's top-K_{tok} set and $m_u > 0$. Thus $\widehat{D}_u$ is the partial sum inside the brackets of Equation 13, and the batch loss of Equation 14 is $\mathcal{L}_w = (\beta/N) \sum_{u=1}^{N} w_u [\widehat{D}_u]_+$ for $N > 0$. Unlike the distributions used to compute JS, the teacher probabilities in $\widehat{D}_u$ are not renormalized on S_u.

Proposition 1 (weighted distillation gradient). With detached weights and teacher scores, the logit gradient at a token with $\widehat{D}_u > 0$ is

$$\nabla_{z_u} \mathcal{L}_w = \frac{\beta w_u m_u}{N} (q_u - \bar{p}_u), \tag{16}$$

and it is zero at a token with $\widehat{D}_u < 0$, with a subgradient choice at zero. Thus, $w_u > 0$ scales the gradient for a fixed token without changing its direction. It can change the direction of the shared parameter gradient after contributions from different tokens are combined.

Proof. Where $\widehat{D}_u > 0$ we have $[\widehat{D}_u]_+ = \widehat{D}_u$. The teacher-dependent term $C_u = \sum_{v \in S_u} p_u(v) \log p_u(v)$ is constant, so $\widehat{D}_u = C_u + m_u \, \mathrm{CE}(\bar{p}_u, q_u)$. Using $\partial \log q_u(v)/\partial z_u(a) = \mathbf{1}\{a = v\} - q_u(a)$ gives $\partial \widehat{D}_u / \partial z_u(a) = m_u q_u(a) - p_u(a)\mathbf{1}\{a \in S_u\}$. Multiplying by $\beta w_u / N$ proves the result.

The failed skill therefore changes how strongly a token contributes, not which teacher distribution it is trained to match. Setting all weights to one recovers the positive-skill OPD loss and gradient on the same fixed batch. It does not imply that separately trained policies will later collect the same trajectories.

E.3 Why the square root responds to small teacher differences

We use a local Taylor expansion to describe the weight near agreement. Such expansions relate smooth f-divergences to a chi-square term (Nielsen & Nock, 2014). The result below applies to fixed, common support; it does not assume that top-token sets remain fixed globally.

Proposition 2 (local JS expansion). On a finite support, let $p(v) \geq \eta > 0$ and $p_\epsilon(v) = p(v) + \epsilon\delta(v)$, where $\sum_v \delta(v) = 0$ and ϵ is small enough that $p_\epsilon(v) > 0$. With natural logarithms,

$$\mathrm{JS}(p \| p_\epsilon) = \frac{\epsilon^2}{8} \sum_v \frac{\delta(v)^2}{p(v)} + O(|\epsilon|^3). \tag{17}$$

For a nonzero perturbation δ, its normalized square root satisfies

$$\sqrt{\frac{\mathrm{JS}(p \| p_\epsilon)}{\log 2}} = \frac{|\epsilon|}{\sqrt{8 \log 2}} \left(\sum_v \frac{\delta(v)^2}{p(v)} \right)^{1/2} + O(\epsilon^2). \tag{18}$$

Proof. For entropy $H(p) = -\sum_v p(v) \log p(v)$,

$$H(p + a\epsilon\delta) = H(p) + a\epsilon \nabla H(p)^\top \delta - \frac{a^2 \epsilon^2}{2} \sum_v \frac{\delta(v)^2}{p(v)} + O(|\epsilon|^3).$$

Substitute $a = 1/2$ and $a = 1$ into $\mathrm{JS}(p \| p_\epsilon) = H(p + \epsilon\delta/2) - \frac{1}{2}H(p) - \frac{1}{2}H(p + \epsilon\delta)$. The constant and linear terms cancel, leaving the coefficient $1/8$. Taking the square root gives Equation 18.

Near agreement, adding JS to a unit weight changes it at second order, whereas adding its square root changes it at first order in the size of the perturbation. This explains the greater sensitivity of

the distance-based weight to small differences. Since $0 \leq \mathrm{JS} \leq \log 2$, span averaging and clipping preserve $1 \leq w_u \leq w_{\max}$ for $\alpha \geq 0$ and $w_{\max} \geq 1$.

The actual teacher views may have different top-token sets and zero-filled entries. At support changes, the positive common-support assumption fails, so this local expansion need not apply. The weight measures teacher-context disagreement; the expansion does not prove that every weighted action caused the original failure or that the positive view is correct on every new prefix.

E.4 The gradient error from keeping only top tokens

The size of a top-token set alone does not determine how closely its gradient matches full forward KL. The relevant quantity is the omitted teacher mass. For one token with an active contribution, omit the index u and let $\epsilon_{\mathrm{tail}} = 1 - m$, with m and $\widehat{D}$ as in Equation 15.

Proposition 3 (tail-mass bound). For $q = \mathrm{softmax}(z)$ over the full vocabulary, let $g = \nabla_z \mathrm{KL}(p\|q)$ and $g_K = \nabla_z \widehat{D}$. Then

$$\|g_K - g\|_1 \leq 2\epsilon_{\mathrm{tail}}. \tag{19}$$

With the same detached weight $w \leq w_{\max}$ applied to both losses,

$$\|wg_K - wg\|_1 \leq 2w\epsilon_{\mathrm{tail}} \leq 2w_{\max}\epsilon_{\mathrm{tail}}. \tag{20}$$

Proof. Full KL gives $g = q - p$, while Proposition 1 gives $g_K = mq - p\mathbf{1}_S$. If $\epsilon_{\mathrm{tail}} > 0$, define the normalized tail distribution $r(v) = p(v)\mathbf{1}\{v \notin S\}/\epsilon_{\mathrm{tail}}$. Subtracting the gradients gives

$$g_K - g = \epsilon_{\mathrm{tail}}(r - q).$$

Both r and q are probability distributions, so $\|r - q\|_1 \leq 2$. If $\epsilon_{\mathrm{tail}} = 0$, the gradients agree exactly. Multiplying by the fixed nonnegative weight proves the weighted bound.

This bounds logit gradients for the unrenormalized partial sum where its contribution is active, with the weight held fixed. It does not bound the separate error in computing the weight from truncated teacher views. A parameter-gradient bound would additionally require control of the model's logit Jacobian. In particular, $K_{\mathrm{tok}} = 32$ does not by itself guarantee a small error; that depends on the retained teacher mass and gives no direct bound on downstream success.

F Complete experimental protocol and additional results

F.1 Stage-specific evaluation

Skill Evolution is scored on the source task that produced the skill, Skill Transfer is scored on held-out tasks with a bank fixed before evaluation, and Skill Internalization is scored on the same held-out tasks with skill injection disabled. We implement this stage with EvoOPD. The external source-task baselines share a frozen executor, action interface, prompt renderer, visible history, and per-episode step cap; they decode greedily, except for Repeated ReAct, which samples its attempts. The trained internalization methods share initialization within each seed and their final evaluation protocol. Their repair budgets, curriculum rollouts, teacher calls, and auxiliary context differ as specified in Section G; the one-shot and frozen-base references use the configurations specified there.

All WebShop methods use the same tolerant action parser. It accepts environment-valid search actions even when they are not in the displayed candidate list.

F.2 Skill-free internalization settings

The six trained rows of Table 4 start from the same Qwen3-8B checkpoint and identical LoRA parameters within a seed. Each seed uses 824 slots from 822 unique source tasks and completes 103 updates of eight root episodes. Their skill-free evaluation uses a 30-step environment cap, 20-turn history, greedy decoding, and a 512-token generation limit. The frozen base row and the six trained rows all use the zero-shot templates of Section C, with no few-shot demonstrations at training or

evaluation. Training uses the same interface with method-specific curricula or bridge validation, and an action-only positive-teacher distillation loss without a policy-gradient term. Section G.3 gives the data selection, repetition, optimizer, precision, masking, and training loss; Section G.4 describes each method and the frozen-base reference.

F.3 SHARED SOURCE-TASK EXECUTION SETTINGS

To compare repair strategies, the six external source-task baselines use the same task set, actor, prompts, action interface, and visible history within each benchmark (Table 18). They share greedy decoding, except for Repeated ReAct, which samples its attempts. Skill Evolution uses the same ALFWorld task set and execution settings. The method-specific update rules and rollout budgets are given in Section G.

Table 18: Shared execution settings for the source-task baselines in Table 3. Settings are identical within each benchmark, except that Repeated ReAct samples at temperature 1.0; method-specific rollout budgets are listed separately in Table 22.

	ALFWorld	**WebShop**
Executor	Qwen3-8B	Qwen3-8B
Response protocol	`react_xml`	`react_xml`
Parser policy	`legacy_tolerant`	`legacy_tolerant`
Environment step cap	20	15
Visible history window	20	2
Decoding temperature	0.0 (1.0 for Repeated ReAct)	0.0 (1.0 for Repeated ReAct)
Nucleus parameter	1.0	1.0
Maximum new tokens per turn	512	512
Thinking mode	Disabled	Disabled
Actor seed	42	42
Frozen source tasks	3,553	10,587

F.4 REVISION DEPTH AND INTERACTION COST

Table 19: Average completed environment rollouts per source task for the methods in Table 3. Single ReAct and the one-shot skill perform no revision, so their cost reflects less work rather than higher efficiency; shading marks the lowest cost among the revision methods. These counts exclude editor and action-scoring calls.

	Rollouts per task ↓	
Method	**ALFWorld**	**WebShop**
Single ReAct rollout (no revision)	1.00	1.00
Repeated ReAct	18.08	5.70
One-shot skill v_0 (no revision)	1.96	1.99
Reflexion	**12.68**	5.47
ReasoningBank	14.31	5.53
Skill-Pro	14.15	5.55
ContractSkill	16.00	5.50
SKILL EVOLUTION (ours)	**12.61**	**4.81**

We measure how many revisions are needed for a skill to succeed and how many rollouts this requires (Table 20 and Figure 3). On ALFWorld the unrevised v_0 certifies 13.62% of tasks and revision continues to add certificates out to $k = 18$, with 32.65 points of the final 46.27% arriving only after v_0 was rejected. On WebShop the increments are already small by $k = 2$ and the lineage budget itself ends at $k = 5$, so revision adds 4.42 points over v_0 in total. The two benchmarks were configured with different revision budgets, 18 revisions on ALFWorld and 5 on WebShop, and each revision may carry at most one patch operation. Read together with Table 7, the binding WebShop constraint is the editor's ability to get any further patch admitted: refusal terminates 38.5% of its lineages, against 7.8% on ALFWorld.

Table 20: Revision depth and realized interaction cost on the source-task runs. Rollouts per task counts one skill-free baseline rollout plus one validation rollout for each accepted version, which is the accounting used in Table 19. Certified and unresolved lineages are reported separately because their costs differ by a factor of nearly three on ALFWorld.

	ALFWorld	**WebShop**
Completed source tasks	3,553	10,587
Skill-free baseline, %	9.82	21.18
Unrevised v_0, %	13.62	24.54
First certified version, %	**46.27**	**28.96**
Certified only after a revision, points	32.65	4.42
Configured revision budget	18	5
Largest revision index that still certifies	18	5
Rollouts per task, all lineages	12.61	4.81
Rollouts per task, certified lineages	6.33	2.33
Rollouts per task, unresolved lineages	18.03	5.81

F.5 Paired performance comparisons

We compare the results in Tables 3 and 4 using paired observations. For source tasks, we match each task's outcome across methods and compute 95% percentile intervals from 100,000 paired bootstrap samples. These intervals measure variation across tasks within the recorded runs. For internalization, we first compute the success-rate difference within each training seed, then form a paired t-interval across the three seeds. We use two-sided exact McNemar tests for source tasks and paired t-tests for internalization, with Holm correction applied separately to each group of ten comparisons. All differences are computed before rounding.

Table 21: Paired success-rate differences relative to each baseline, in percentage points. Positive differences favor our method; source-task intervals use paired task resampling, and internalization intervals use variation across three training seeds.

Skill Evolution	**ALFWorld**		**WebShop**	
Baseline	Δ	95% CI	Δ	95% CI
Repeated ReAct	**+36.05**	$[+34.45, +37.66]$	**+6.82**	$[+6.18, +7.46]$
Reflexion	**+2.65**	$[+0.84, +4.45]$	**+0.94**	$[+0.26, +1.62]$
ReasoningBank	**+6.36**	$[+4.56, +8.16]$	**+2.35**	$[+1.68, +3.02]$
Skill-Pro	**+9.09**	$[+7.32, +10.86]$	**+2.72**	$[+2.06, +3.39]$
ContractSkill	**+22.83**	$[+21.11, +24.54]$	**+2.15**	$[+1.50, +2.81]$
Skill Internalization (EvoOPD)	**Seen**		**Unseen**	
Baseline	Δ	95% CI	Δ	95% CI
Vanilla AgentOPD	**+2.62**	$[-2.50, +7.74]$	**+3.23**	$[-6.98, +13.44]$
SOD	**+4.76**	$[-3.24, +12.76]$	**+6.72**	$[-7.76, +21.20]$
TCOD-F2B	**+4.52**	$[-0.60, +9.65]$	**+6.97**	$[-4.95, +18.88]$
TCOD-B2F	**+4.05**	$[-5.72, +13.82]$	**+8.21**	$[+3.30, +13.11]$
FutureBridge-OPD	**+0.24**	$[-2.47, +2.95]$	**+5.97**	$[-3.84, +15.78]$

On the aggregate success rate, Skill Evolution improves over all five source-task baselines on both benchmarks, with Holm-adjusted $p < 0.01$ in every comparison. Against Reflexion, the gain is 2.65 percentage points on ALFWorld and 0.94 points on WebShop; both paired intervals remain above zero.

EvoOPD has a higher mean success rate than every internalization baseline, and the margins are consistently larger on Unseen: 3.23 points over Vanilla AgentOPD, the strongest baseline on that split, and 5.97 to 8.21 points over the remaining methods, with the TCOD-B2F interval excluding zero. The gain over FutureBridge-OPD is positive in all three seeds. Seen differences are smaller, at 0.24 points against FutureBridge-OPD.

G BASELINE IMPLEMENTATIONS AND REPRODUCIBILITY

G.1 SHARED SOURCE-TASK SETTINGS

We evaluate all source-task baselines on the task sets in Table 18, using a frozen Qwen3-8B actor and, when needed, the same model as editor. Their task-local state starts empty and is discarded between tasks. The shared actor interface fixes the observations, ordered action candidates, parser, and history window. Decoding is greedy for every baseline except Repeated ReAct, which samples its attempts. Each external editor request permits 1,024 output tokens, uses temperature zero and seed 42, and has an 80,000-character input ceiling; the injected guidance body is limited to 6,000 characters. These character limits are not token limits. Editors receive the failed trajectory, including actions and feedback; their compact view can shorten long observation text and display at most 80 action candidates per recorded step. This editor view is distinct from the actor's full current action list.

Each source-task rollout has an environment cap of 20 steps on ALFWorld and 15 steps on WebShop, and uses the same actor, prompt renderer, history window, action interface, and parser. Rollout caps are listed in Table 22. An external baseline stops at its first successful rollout or when its rollout budget is exhausted. An editor update permits at most three total format attempts; exhausting these attempts retains the previous artifact, and the next rollout still consumes the task budget. The reported rollout cost counts completed environment episodes. It excludes editor requests and Skill-Pro's action-scoring requests, and is not a measure of total model computation or failed infrastructure attempts. Infrastructure retries are handled separately from semantic repair. Each task contributes one final result to the success rate.

Table 22: Source-task budgets and editor use. Caps include the initial skill-free rollout; editor calls are averaged over completed tasks. ALF and WS denote ALFWorld and WebShop. Skill-Pro scoring requests are additional to these editor calls. The one-shot cap counts only baseline collection and first-version evaluation, even when extracted from a longer evolution run.

	Rollout cap		**Editor calls/task**	
Method	ALF	WS	ALF	WS
Single ReAct rollout	1	1	0	0
Repeated ReAct	20	7	0	0
One-shot skill v_0	2	2	Initial-skill generation only	
Reflexion	20	7	11.68	4.47
ReasoningBank	20	7	13.32	4.54
Skill-Pro	20	7	55.91	19.35
ContractSkill	20	7	27.98	5.96
SKILL EVOLUTION	20	7	12.14	6.07

G.2 SOURCE-TASK BASELINE IMPLEMENTATIONS

Single ReAct rollout. The actor executes one skill-free episode from the task's reset state. Its system instruction requests reasoning followed by one XML action, and each user turn presents the task, current step, observation, and admissible actions. The message list retains the initial task message and the configured recent action–observation history. The model weights remain frozen, and no editor, memory induction, skill retrieval, or retry policy is invoked. The result is the environment's task-success flag for that episode, with a 20-step cap on ALFWorld and a 15-step cap on WebShop. We use a zero-shot ReAct prompt without the original few-shot examples. Thinking mode is disabled in the Qwen chat template; the XML instruction and tolerant action parser still apply to the generated text.

Repeated ReAct. This control uses the same actor, reset state, and prompt construction as Single ReAct. After an unsuccessful episode, it starts another episode without additional guidance, carrying over neither a reflection nor the previous trajectory. The total allowance is 20 completed rollouts per task on ALFWorld and seven on WebShop, and execution stops at the first success. Attempts are sampled at temperature 1.0, and each attempt uses a different actor seed, starting at 42 and incremented by one per attempt, so each one is an independent plan rather than a repetition of the same greedy execution. A task succeeds if any attempt succeeds, and its cost is the number of

attempts performed. The method uses no editor or parameter updates and receives no new guidance between attempts.

One-shot skill v_0. We take the first skill document produced by the Skill Evolution editor from the task's baseline trajectory and evaluate that document once with the frozen actor. The document uses the same typed skill schema as the evolution method, but this row uses no subsequent patch, localization-guided revision, or retrieval from another task. If initial generation produces no admissible document, the task remains in the denominator and has no successful v_0 execution. The accounting includes baseline collection and the first-version rollout when a document exists. The ALFWorld reference uses the first-version results from the same source-task run as the final evolved skills in Section F.4. WebShop uses the first-version results of its reported evolution run. Both the one-shot and final evolved-skill columns measure success with the stated skill version; they do not automatically retain an earlier skill-free success as the external retry adapters do.

Reflexion. Our adapter retains Reflexion's verbal feedback between attempts (Shinn et al., 2023). After a failed rollout, the editor receives the task, its failed trajectory, and up to three previous plans. It is asked to diagnose the strategy and propose one concrete new plan for the same task. A nonempty JSON reflection is appended to the memory, the oldest entry is dropped when necessary, and the most recent three reflections are rendered into the actor's system context for the next reset. An exhausted format-correction sequence leaves that memory unchanged. This memory is local to the current task, has no trainable parameters, and is never retrieved for a different task. We retain the reflection-and-retry mechanism while adapting the output format, actor, action parser, and trial budget to the shared text-action evaluator. Environment success, rather than an additional LLM judge, determines when the trial loop stops.

ReasoningBank. The adapter induces one to three memory items after a failed episode, each containing a title, applicability description, and actionable content (Ouyang et al., 2026). The editor is instructed to identify the failure and avoid copying instance-specific object numbers, products, or query strings. Valid items are appended to a task-local bank. The renderer selects the newest complete items that fit the 6,000-character guidance budget and preserves their order in the actor context; it does not truncate a selected item midway. Failed induction leaves the bank unchanged. This preserves structured memory induction and append-only accumulation, but the evaluated row is a same-task repair adaptation. It does not run the original cross-query embedding retrieval, success-and-failure memory pool, LLM success classifier, or memory-aware test-time scaling procedure. The loop allows up to 20 rollouts on ALFWorld and seven on WebShop, stopping at the first environment success. This comparison measures the use of accumulated failure guidance within a task.

Skill-Pro. We implement a non-parametric semantic update and probability gate (Mi et al., 2026). Each failed episode first produces a diagnosis and separate proposed changes to skill initiation, policy, and termination. The editor then proposes three candidate skills with two to eight policy steps each. For the old skill and each candidate, the frozen model rescores the failed actions under the corresponding system context. The local scorer reconstructs an `<action>` command span and averages its token log probabilities; it does not score the complete recorded reasoning trace. The gate exponentiates candidate-minus-old action scores, uses uniform negative advantages for the failed trajectory, and applies a PPO-style ratio clip of 0.2. We keep the best candidate only when its gate improvement exceeds 0.001; otherwise the old skill remains active. This gate changes text, not model parameters. The implementation evaluates one task-local skill under the shared actor and a budget of 20 rollouts on ALFWorld or seven on WebShop, without the original persistent cross-task skill-library maintenance. Candidate generation and rescoring explain why its model-call cost exceeds its environment-rollout cost.

ContractSkill. The adapter represents a skill by applicability, preconditions, indexed conditional steps, a success contract, recovery instructions, and an append-only repair history (Lu et al., 2026b). The first failed episode produces the initial contract. On later failures, a deterministic locator first checks for an environment-rejected action, then a repeated action with no state change, and otherwise selects the terminal failure. The editor receives the nearby transitions and terminal evidence, chooses one listed leaf identifier, and returns replacement text plus a repair note. The adapter applies that replacement mechanically, requires exactly one contract field to change, and appends one history entry.

Invalid proposals retain the previous contract. Each valid repair is then tested by the shared actor. This preserves explicit contracts, local diagnosis, and minimal text repair, while replacing browser-specific selectors and intermediate DOM verifiers with text-action evidence. The local loop uses first environment success as its stopping rule; it does not implement the original cost-and-edit-regularized acceptance objective.

G.3 Internalization data and shared training procedure

The six trained rows of Table 4 use the same frozen ALFWorld source selection and held-out task identities. From the 1,644 certified source lineages, 1,311 also have a failed skill-free baseline. A fixed task-ID hash order selects 822 of these positive-utility records; selection does not rank held-out performance, trajectory length, or revision count. Their certified, task-specific skills provide teacher context, rather than the generalized bank used for Skill Transfer. Each selected skill is paired with its successful source-task trajectory. Before training, we replay each reference action sequence in the environment to verify that it completes the task. There are 707 unique records with a failed immediate predecessor and 115 without one. The first two selected records are each repeated once at unit weight, producing 824 slots and 709 counter-bearing slots. Each seed fixes a permutation of those slots. The Seen and Unseen evaluation sets contain 140 and 134 tasks, respectively, and their task identities are disjoint from training.

Table 23: Shared settings for the six trained internalization methods. A root episode is one scheduled task instance; curriculum prefixes and FutureBridge validation add the method-specific work described below. The frozen-base row is discussed separately.

Setting	Value
Student and teacher	Qwen3-8B; separate frozen base teacher with certified skill context
Trainable parameters	LoRA on all linear projections; rank 16, scaling $\alpha = 32$, dropout 0
Seeds and initialization	42, 43, 44; identical LoRA initialization across methods within each seed
Data and updates	822 unique tasks, 824 slots; eight root episodes per update, 103 updates
Optimizer	AdamW; constant learning rate 10^{-5}, betas $(0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01
Accumulation and clipping	One response per backward call; accumulate all turns in the eight-episode batch; gradient-norm cap 1
Precision	BF16 model loading on GPU; FP32 logits and log-softmax for scoring
Context limits	Student 32,768; teacher 34,816 tokens, including generation allowance; no input truncation
Prompt allowances	Initial student prompt at most 2,048 tokens; added skill context at most 2,048 tokens
Actor decoding	Greedy, repetition penalty 1, at most 512 new tokens per turn; Qwen thinking mode disabled
Environment and history	ALFWorld deterministic fixed-game reset; 30-step root cap; last 20 action–feedback pairs
Distillation support	Teacher top 32 tokens at temperature 1; original probability mass retained, no tail bucket
Loss scope	Parsed action-content tokens; unweighted global action-token denominator
Checkpoint and evaluation	Final update 103; one episode per held-out task; no teacher, skill, curriculum, or bridge at evaluation
Aggregation	Equal-weight mean of the three per-seed success rates; sample standard deviation ($n - 1$ denominator)

The policy is fixed while collecting the eight root episodes and any permitted bridge branches. Teacher scores are computed on the exact generated response tokens under the corresponding history. Each response is then rescored with student gradients before a single optimizer step; trajectories are not reused for additional optimization epochs. A seed changes adapter initialization and slot order as well as recorded episode seeds. Greedy generation does not make the episodes independent stochastic samples. Final checkpoints are selected by update count, without selecting the best held-out score.

The training loss sums $p^{+}_{ji}(v)\big(\log p^{+}_{ji}(v) - \log \pi_\theta(v \mid x_{ji})\big)$ over the positive teacher's top-32 support, clips a negative sum to zero, multiplies by the method weight, and divides by the number of included action-content tokens across the batch. This is the positive-part form of Equation 14 and Algorithm 1. The denominator is not the sum of weights. Reasoning text, XML delimiters, observations, and replayed reference-prefix actions have zero loss mask. Unparsable responses execute an empty action and consume an environment step, but contribute no action-token loss. No trained row adds a task-reward, policy-gradient, PPO, or GRPO objective.

G.4 INTERNALIZATION BASELINE IMPLEMENTATIONS

Frozen base model. This row is an untrained, skill-free Qwen3-8B reference. It has no optimizer updates, teacher scoring, or learned adapter. The frozen base model and all six trained methods are evaluated with the same 30-step environment cap, prompt renderer, history window, action interface, parser, and decoding settings. The same held-out task lists and aggregation protocol apply to every row.

Vanilla AgentOPD. The student starts each training episode without a skill and acts for up to 30 environment steps. A separate, frozen copy of its Qwen3-8B base checkpoint receives the certified skill for that training task and rescores the student's actual responses. Every valid action token has weight one in the shared top-32 partial forward-KL loss. No failed skill, curriculum prefix, teacher intervention, or task reward enters this baseline. Its teacher differs from the student only through its certified context. This is the uniform positive-skill distillation control for EvoOPD: it matches the source records, within-seed initialization, update schedule, and skill-free final evaluation. Its student trajectories are collected from its own evolving policy, so matching the task slots does not imply that different methods produce identical actions or visit identical states.

SOD. We retain the step-wise disagreement weighting from SOD (Zhong et al., 2026) and attach it to the common direct-KL trainer. For each valid environment action k, the adapter computes d_k, the mean absolute difference between the frozen rollout student's and positive teacher's log probabilities of the actually generated action tokens. The first valid action has weight one. For subsequent actions, the weight is $\min\left\{\prod_{j=1}^{k-1} \frac{d_j+10^{-6}}{d_{j+1}+10^{-6}},\ 1.2\right\}$. The cumulative product is not itself clipped before the next step, and the result is detached from gradients and broadcast to that action's tokens. An unparsable step has no defined action disagreement and is skipped in this sequence of valid actions. Episodes retain the full 30-step allowance. This is the SOD weighting adaptation: it uses action spans instead of complete assistant responses and omits the original reward-advantage and PPO training components. It shares Vanilla AgentOPD's teacher, data, loss denominator, and optimizer.

TCOD-F2B. The front-to-back adaptation changes the extent of student interaction while keeping uniform action-token weights (Wang et al., 2026b). At zero-based optimizer update $u \in \{0, \ldots, 102\}$, its horizon is $q(u) = 1 + \lfloor 29u/102 \rfloor$. The student always starts from the initial environment state and generates at most $q(u)$ actions, stopping earlier on a terminal state. Thus early batches distill short prefixes and the final update permits the complete 30-step interaction. The unexecuted suffix incurs no teacher or environment calls and contributes no loss. The schedule advances once per optimizer update, not per response or gradient-accumulation call. This retains TCOD's front-to-back temporal curriculum, mapped to the common 103-update schedule, and uses our certified-skill teacher and partial forward-KL objective in place of the original sampled distillation surrogate. Its eight root episodes per update therefore entail fewer environment steps than full-horizon Vanilla AgentOPD.

TCOD-B2F. The back-to-front adaptation uses the same $q(u)$ but starts the student after a verified prefix of the task's certified reference trajectory (Wang et al., 2026b). For a reference of length L, it replays $p = \max(0, L - q(u))$ actions in the real environment and then permits up to $30 - p$ student actions. The latter allowance is the remaining environment budget, not necessarily $q(u)$. Prefix actions and their observed feedback enter the visible history, rendered as executed action messages; the unused reference suffix is never exposed to the student. Prefix actions receive no loss. Reference-prefix replay must reproduce the required task state before the student continues; a mismatched replay is rejected. Student-generated actions receive uniform positive-teacher KL supervision. The controlled changes from the original TCOD execution are the certified source trajectories, common actor interface, 103-update schedule, and direct-KL objective. Actual prefix steps remain part of the environment-call accounting.

FutureBridge-OPD. We implement FutureBridge-OPD from its paper (Chen et al., 2026) and adapt it to the shared trainer. It starts with the TCOD-B2F reference-prefix curriculum and attempts at most one bridge per root episode. Among nonfinal student turns with valid action tokens, it chooses the largest mean sampled student-minus-teacher log probability, breaking ties by the earliest turn. The environment is reset and replayed to that state, and the certified-skill teacher proposes one replacement action. Two fresh three-turn continuations are generated by the same fixed student policy

after the original and replacement actions. A bridge is accepted only if its continuation has a strictly larger fraction of action tokens that the teacher scores above the student. Missing tokens, insufficient remaining steps, or a continuation ending before the complete horizon cause rejection. Only an accepted bridge action is added to the original root records for training; both future continuations remain validation-only with zero training masks. The original root state is restored afterward. This preserves localization and paired future validation while adapting their score scope to action tokens and their training objective to the shared forward KL. The gate does not use reward or future success as an optimization target.

EvoOPD as the shared-protocol reference. EvoOPD uses Vanilla AgentOPD's full-horizon student rollout and positive certified-skill target. For a record with a failed immediate predecessor, the same frozen teacher also rescores the same student response under that failed skill. The action-span mean of the normalized Jensen–Shannon distance between the two teacher views sets $w = \mathrm{clip}(1 + \bar{d}, 1, 2)$; gradients do not pass through the weight. Each view is normalized on its own top-32 support and aligned on their union for this distance calculation only. A record without a failed predecessor keeps unit weight and requires no counter-view scoring. The failed view changes where the positive supervision is emphasized, but never becomes a target to imitate. There is no reference-prefix curriculum or future-trajectory gate. The additional teacher scoring is accounted for separately, and the final student is evaluated under the same skill-free protocol as the other five trained methods. Section E gives the weight construction and its measured activity; the training loss is specified above.

G.5 Training cost and implementation details

Equal root-task and optimizer budgets do not equalize total environment work or teacher computation. Table 24 reports environment steps and teacher computation as a range across the three seeds. Replayed prefix and restoration steps are included in environment steps; future-validation branches are also charged even though they have no loss. Teacher scoring counts all generated response tokens passed through scoring forwards, including tokens subsequently masked out of the loss. The positive teacher scores responses during collection and again when assembling supervision; EvoOPD adds the failed view, and FutureBridge adds bridge generation and paired-branch scoring. The accounting covers internalization training, not the earlier construction and certification of the shared skill source.

Table 24: Training cost per seed, minimum–maximum over seeds 42–44. All rows complete 824 scheduled root slots. Scored tokens are token positions processed by the teacher, not the number of optimized action tokens. Counts cover completed updates and exclude unmeasured computation from interrupted attempts.

Method	Environment steps	Teacher-scored tokens	Teacher-generated tokens
Vanilla AgentOPD	17,119–17,523	432,634–442,302	0
SOD	17,364–17,673	435,668–445,832	0
TCOD-F2B	9,455–9,746	245,440–255,822	0
TCOD-B2F	15,834–16,534	357,916–371,800	0
FutureBridge-OPD	46,119–47,948	431,412–444,344	10,782–10,907
EvoOPD	17,101–17,580	617,781–635,809	0

The three FutureBridge runs and SOD seed 44 resumed after interruptions. Their reported costs omit computation from the interrupted attempts and therefore do not measure the full cost of these runs.

Reproduction requires the shared trainer, the method-specific implementations described above, and their experiment configurations.

H Training dynamics and computational cost

H.1 Optimization stability and interpretation

We analyze the completed training runs behind Table 4: six methods, three seeds, and 103 optimizer updates per seed. Each update consumes eight scheduled root episodes. For noncumulative curves, we

first take a trailing five-update mean within each seed; fewer points are used at the start. Figure 4 then shows the mean and sample standard deviation across seeds, with faint lines retaining the unsmoothed means. Figure 8 instead retains each seed as a separate curve.

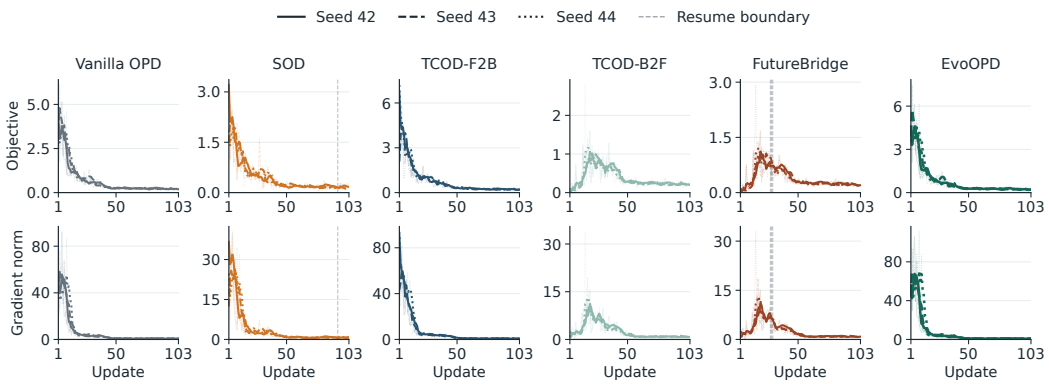

Figure 8: Training objective and pre-clipping gradient norm for each method and seed. Faint lines show individual updates, heavier lines show five-update means, and vertical dashed lines mark resumed runs; each method has its own vertical scale.

The full-horizon methods show a sharp early reduction in the training objective and gradient norm, followed by smaller fluctuations. For EvoOPD, the mean objective falls from 4.13 over the first five updates to 0.22 over the last five, while the corresponding pre-clipping gradient norm falls from 57.71 to 0.89. The large early norms are measured before the shared clipping cap of one. SOD reweights the loss, so its smaller objective reflects that reweighting rather than task performance.

TCOD-B2F and FutureBridge follow a different early path: their initially small objectives rise as the certified prefixes are withdrawn, then decline. Their high initial root success in Figure 4 is measured after reference-prefix replay, whereas TCOD-F2B initially permits only short student interactions from the task's initial state. This explains why early success and loss should be read together with the curriculum panels, not as a ranking of the initial models. The identical planned prefixes and horizons of B2F and FutureBridge share the B2F curve; Full and None denote the no-curriculum horizon and zero-prefix reference.

In the behavior panels, Valid and Parsed count root turns with an admissible and a nonempty parsed action, respectively. Action is the fraction of generated response tokens included in the root action mask. Student NLL, Teacher NLL, and S–T gap are taken-action negative log probabilities and their absolute log-probability difference, in nats; they are measured on each method's own generated actions, not a fixed evaluation corpus. The weight panels show the non-unit token share and the per-update minimum and maximum. In the bridge panel, Try, Gate, and Use count attempted bridges, completed paired comparisons, and accepted bridges as fractions of all root episodes. Calls is the validation share of environment calls; Tokens is the accepted bridge share of loss-bearing action tokens. These last two shares have different denominators.

H.2 Computational cost and action supervision

Figure 9 separates generation, teacher scoring, prompt processing, environment calls, and action supervision. Cumulative curves use the unsmoothed seed means and sample standard deviations. Budget-axis panels retain each seed and connect means at the same optimizer update, without interpolating to a common budget. Costs include replay and future validation for completed updates, but exclude unmeasured computation from interrupted attempts. The axes measure token processing and environment calls, not wall time or FLOPs.

EvoOPD and Vanilla AgentOPD both use about 17.3 thousand environment calls per seed and produce similar amounts of loss-bearing action tokens. Their teacher cost differs: EvoOPD processes 625.9 thousand scored tokens, compared with 437.4 thousand for Vanilla AgentOPD, an increase of about 43%. Teacher prefill shows the same pattern. The added work comes mainly from scoring

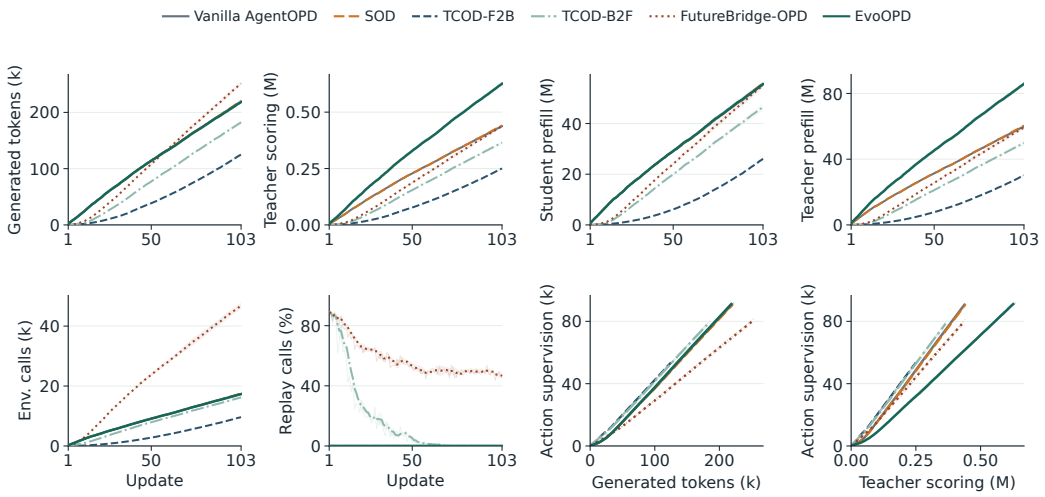

Figure 9: Training cost and resulting action supervision. Curves retain all three seeds; cumulative panels are unsmoothed, while replay shares use five-update means. The final two panels relate loss-bearing action tokens to generation and teacher-scoring budgets.

the failed skill view, rather than generating more student trajectories. The shared root-task budget therefore does not imply an equal-teacher-compute comparison.

FutureBridge instead uses 46.8 thousand environment calls per seed, about 2.70 times Vanilla AgentOPD. Across its runs, future validation accounts for 65.4% of recorded environment calls, while accepted bridge actions supply only 1.7% of loss-bearing action tokens. The gate can therefore consume much more interaction than the amount of supervision it ultimately retains. Only the bridge action enters training; the paired futures are used solely to decide whether to keep it.

TCOD-F2B has the lowest measured cost, with 9.6 thousand environment calls and 53.6 thousand supervised action tokens per seed, compared with 17.3 thousand calls and 90.7 thousand tokens for Vanilla AgentOPD. Its short early horizons reduce both interaction and learning exposure. Together, these curves distinguish three costs that an update count alone hides: obtaining student experience, scoring extra teacher views, and validating candidate corrections.

I Case study: learning beyond a certified trajectory

I.1 Source certification and student evaluation

The 46.27% source rate and the 54.52%/64.18% held-out rates are measured in different settings: the former is source-task certification success (1,644/3,553), whereas the latter evaluate the skill-free student on held-out Seen and Unseen tasks. We therefore compare students under a shared teacher rather than ranking student against teacher: whether different students benefit differently from the same teacher, and whether a student can execute a more efficient route than the recorded certified trajectory.

I.2 Different students, identical positive supervision

We pair the final EvoOPD and Vanilla AgentOPD outcomes by task and training seed, with the same initial model, positive teacher, certified skills, training tasks, and evaluation protocol. Table 25 covers 420 Seen and 402 Unseen task–seed pairs. EvoOPD gains 20 successes and loses 9 on Seen, and gains 28 and loses 15 on Unseen, giving net improvements of 2.62 and 3.23 percentage points. The gains are not uniform across task families: on Seen Heat, EvoOPD gains one success but loses three. Under shared positive supervision, the net gains are attributable to the evolution-weighted training rule.

Table 25: Paired final-checkpoint outcomes.

Split	Both succeed	EvoOPD only	Vanilla only	Both fail
Seen	209	20	9	182
Unseen	230	28	15	129

I.3 Shorter successful routes without copying the reference

For a post-hoc comparison of action sequences, we examine all eight episodes collected immediately before the final update of each EvoOPD seed: 24 source-task episodes at policy version 102, all collected with the training prompt rather than the evaluation protocol. All 24 reference trajectories complete their source tasks. Replaying them starts from the same initial observation and admissible-action set as the student on each task. The skill-free student succeeds in 14 episodes: six use fewer actions than the reference, two use the same number, and six use more. It fails in the other ten. Figures 10–12 visualize the two largest successful action savings and one heating failure. Because the reference and student rollouts use different interaction settings, the comparison covers action sequences only.

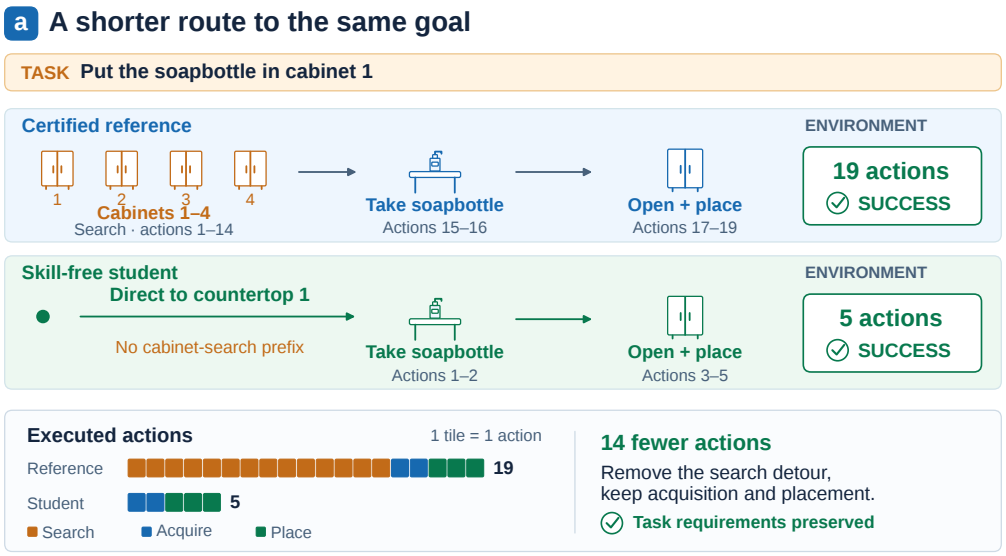

Figure 10: Student execution skips a certified search detour.

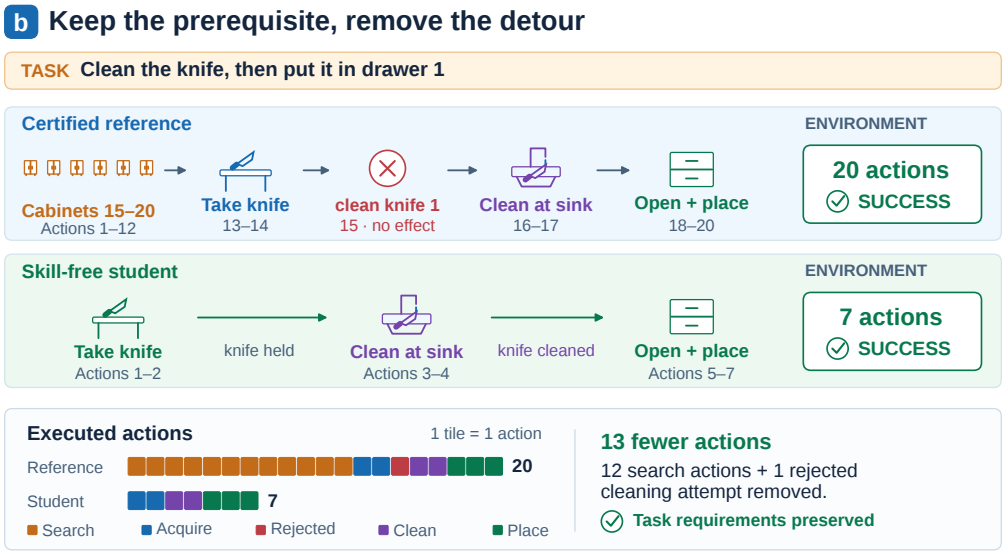

Figure 11: Student execution retains cleaning before placement.

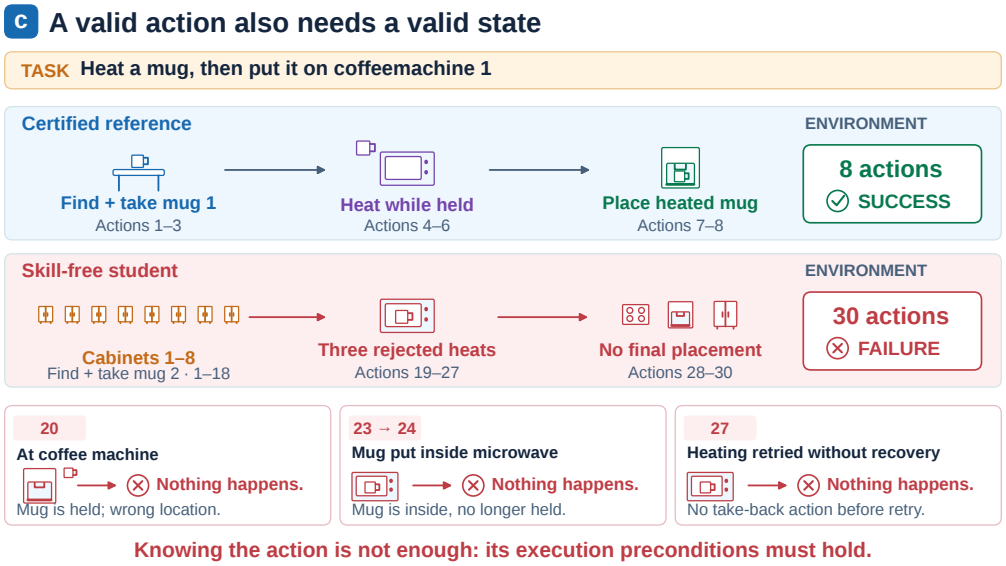

Figure 12: Heating fails when execution preconditions are not met.