SKILLSMITH: CO-EVOLVING SKILLS AND TOOLS FOR SELF-IMPROVING AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents increasingly store capabilities in external skill and tool libraries. Libraries can grow, but execution must fit a finite context window and improvement consumes a finite iteration budget. We formulate external-policy evolution under these two budgets and introduce SKILLSMITH. A skill ecosystem allocates context through replicator dynamics, using randomized masking and Bayesian estimation to learn skill effects and interactions. Four typed operations repair tools and move reusable logic across the skill–tool boundary, freeing context for the ecosystem. Conflict-driven learning distills failed proposals into version-scoped nogoods that guide repairs and veto repeated failing proposals. We prove utility ascent for fixed-model continuous dynamics, conditions under which lower context costs expand attainable utility, and the need for synchronized interface migration under persistent exact-match dependencies. Across five Qwen3.5 scales, SkillSmith improves over EvoSkill by up to 10.2 and 9.8 percentage points on OfficeQA and SealQA, respectively, and sustains improvement beyond SkillClaw on WildClawBench. *Skills compete for context; tools are how the system makes room.*

1 INTRODUCTION

Tool use is a basic capability of LLM agents (Nakano et al., 2021; Karpas et al., 2022; Yao et al., 2022; Schick et al., 2023), and increasingly their capabilities reside outside model weights. A *skill* packages workflow instructions, scripts, domain resources, and metadata into a reusable unit (Zhang et al., 2025). Skills and tools together form an external policy layer that can be inspected, versioned, and transferred across models.

Methods for evolving this layer are emerging. Trace2Skill (Ni et al., 2026) and SkillX (Wang et al., 2026) distill skills from execution traces; EvoSkill (Alzubi et al., 2026) optimizes them through failure-driven analysis and Pareto selection; SkillClaw (Ma et al., 2026) enables collective evolution across users. GEPA (Agrawal et al., 2025) shows that reflective textual evolution can be more sample-efficient than scalar-reward reinforcement learning.

Yet external capabilities can be read only through a finite context window. The library may grow, but the workbench remains fixed: each retrieved skill occupies space that others could use. Evolution must therefore manage competing capabilities, not merely accumulate them. This perspective reveals three problems (Figure 1).

① **Logic lives in the wrong layer.** Holding tools fixed forces repairs into skills. Consider a table extractor that shifts one column at merged headers (Figure 6): three dependent skills each learn an offset correction for the same defect. All three corrections occupy context and rely on LLM interpretation, although the tool could implement the correction deterministically. Where logic resides thus determines both its context cost and reliability; skill-only evolution leaves this boundary fixed.

② **Value is collective.** Skills share context, and their value depends on their companions: two individually useful skills may interfere. Evaluating skills separately misses these interactions. Observational logs also confound a skill's effect with the tasks that retrieve it, while small training sets provide little evidence for individual skill pairs.

③ **Failures are discarded.** When failed trajectories serve only one reflection and rejected proposals leave no persistent record, the same unsuccessful modification can recur, consuming iterations without adding knowledge.

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

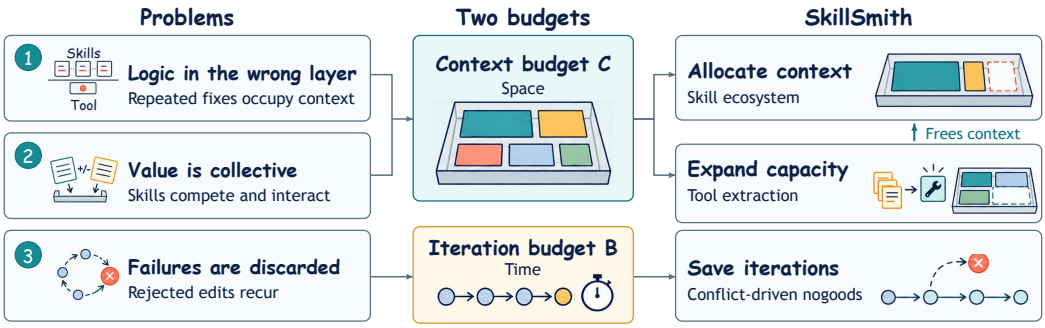


Figure 1: **Three problems, two budgets, three mechanisms.** The skill ecosystem allocates context; tool extraction frees space for that allocation; conflict-driven nogoods save iterations. Extraction increases effective capacity within a fixed context window.

We introduce SKILLSMITH, which co-evolves skills and tools under *two budgets*: context limits space per task, while iterations limit improvement attempts. It connects three mechanisms through fault localization, which routes skill, tool, and interaction failures to the appropriate repair.

Allocate context: skill ecosystem. Skills receive shares of reserved context, with fitness given by marginal utility per allocated token, including interactions. Shares follow replicator dynamics. Random masking identifies effects under a stationary second-order model, and Bayesian estimation shrinks poorly supported interactions toward zero. Thompson sampling supports exploration; posterior evidence governs retirement. For fixed effects and costs, continuous dynamics increase modeled utility and equalize marginal returns among positively allocated skills at equilibrium.

Expand capacity: tool evolution. Shared, deterministic, context-independent logic belongs in tools; orchestration and judgment remain in skills. PATCH and RETYPE repair tools, while EXTRACT and INLINE move logic across the boundary in opposite directions. Extraction replaces duplicated logic with tool calls, freeing context when the savings exceed interface and call-site overhead. This is niche construction for the skill ecosystem: tools make room for allocation. Interface changes and required caller edits are committed as atomic bundles to preserve legality.

Save iterations: conflict-driven learning. Inspired by conflict-driven clause learning, SkillSmith uses delta debugging to reduce edit-induced failures to compact *nogoods*. Stored evidence and remedies guide diagnosis; matching records veto repeated failures. Version scopes expire obsolete records as components evolve.

Contributions. ① We formulate external-policy evolution under two budgets, making the skill–tool boundary an optimization variable. ② SkillSmith unifies context allocation, capacity gains through tool evolution, and the reuse of failure evidence. ③ Our analysis establishes fixed-model allocation ascent, conditions for strict capacity gains, and synchronized migration requirements under persistent exact-match dependencies. Experiments span five model scales and three benchmarks.

2 PROBLEM SETUP

We fix the backbone model and evolve inspectable, versioned, transferable external assets accessed through a bounded context window.

2.1 THREE LEDGERS

The state $\Sigma_t = (\mathcal{S}_t, \mathcal{T}_t, \mathcal{F}_t)$ comprises skill, tool, and nogood libraries.

Skills. Each skill $s = (m, w, r) \in \mathcal{S}_t$ packages metadata m (name, trigger, version), workflow instructions w, and resources r (templates, domain knowledge, or scripts).

Tools. A tool $\tau = (d, f, \sigma)$ consists of an interface description d, implementation f, and signature σ specifying argument types and output format. The library $\mathcal{T}_t$ defines available environment operations. Skills and tools calling τ form $\mathrm{Dep}(\tau)$ and $\mathrm{Dep}_{\mathcal{T}}(\tau)$, respectively, defining the dependency graph. A state is *legal*, $\Sigma \in \Omega$, when every called tool exists and all call sites match its signature.

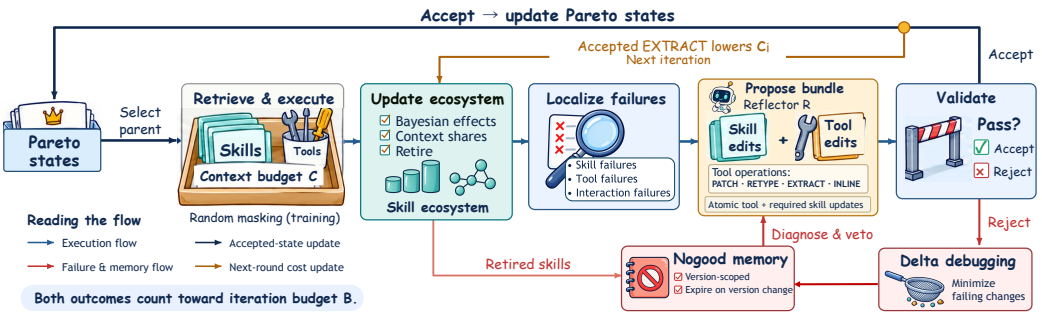


Figure 2: SkillSmith allocates context through the skill ecosystem, frees space through tool extraction, and saves iterations through conflict-driven failure learning.

Nogood memory. The memory $\mathcal{F}_t$ stores modification combinations verified to cause failure, scoped to component versions and paired with remedies (§3.5). Skills and tools retain successful experience; nogoods retain failure evidence.

Two execution modes. ① **Executor:** the LLM reads and interprets skill workflows; the machine executes tool implementations. ② **Determinism:** skill execution is stochastic; tool execution is deterministic. ③ **Context:** skill content enters the window, whereas a tool exposes only its description d, keeping implementation f outside context. Skills carry orchestration and task-specific judgment; tools carry stable, testable logic. Evolving this boundary changes both cost and reliability (§3.4).

2.2 Executing Through a Window

For task x, retrieval loads skills $\mathcal{S}_{\text{act}}(x) \subseteq \mathcal{S}_t$ and their tools' interface descriptions. A skill's *context cost* c_i is its expected injected token count: metadata, workflow, loaded resources, and tool descriptions shared equally among callers. Costs stay fixed during allocation updates; execution also checks realized token counts when resource lengths vary. Retrieval enforces $\sum_{s_i \in \mathcal{S}_{\text{act}}(x)} c_i \leq C$. Execution produces a trajectory $\xi = (a_1, o_1, \ldots, a_L, o_L)$ and black-box score $P(x \mid \Sigma_t) \in [0, 1]$. This score reflects workflows, tools, and their interactions; inferring which component caused a failure is a central challenge.

2.3 Two Budgets

Given initial state Σ_0, training and validation sets $D_{\text{train}}, D_{\text{val}}$, and iteration budget B, each round proposes an atomic bundle $\mathcal{L}_t$ of skill edits and tool operations, with update $\Sigma_{t+1} = \mathcal{U}(\Sigma_t, \mathcal{L}_t)$. The objective is

$$\max_{\{\mathcal{L}_t\}_{t=0}^{T-1}} \mathbb{E}_{x \sim D_{\text{val}}}[P(x \mid \Sigma_T)] \quad \text{s.t.} \quad \Sigma_t \in \Omega \ \forall t, \qquad \sum_{t=0}^{T-1} b(\mathcal{L}_t) \leq B, \tag{1}$$

with every execution subject to context budget C. The cost $b(\mathcal{L}_t)$ counts executions spent proposing and validating the bundle, whether accepted or rejected.

3 SkillSmith

Figure 2 shows how SkillSmith combines context allocation, tool evolution, and conflict-driven learning. We first define library utility and the skill ecosystem, then describe fault localization, typed tool operations, and failure memory. Section 3.6 details candidate validation; Appendix A.7 gives the full evolution algorithm.

3.1 Library Utility under a Context Budget

Let $p_i \in [0, 1)$ be the nominal activation propensity of skill s_i. Under an independent-activation relaxation, a second-order model of the library's expected contribution to task residuals is

$$\widetilde{U}(p) = \sum_i u_i p_i + \frac{1}{2} \sum_{i \neq j} \beta_{ij} p_i p_j, \tag{2}$$

where u_i is the main effect and $\beta_{ij} = \beta_{ji}$ the interaction: positive for complementarity and negative for conflict, with $\beta_{ii} = 0$. The propensities define an allocation surrogate; retrieval enforces the execution budget. External-state improvement can **(i)** revise skills to improve u_i or β_{ij}, **(ii)** reallocate p, or **(iii)** reduce costs c_i. SkillSmith uses all three routes.

3.2 Allocating Context: Skill Ecosystem

Population and fitness. Each skill receives reserved budget share y_i, with slack y_0 and $\sum_{i=0}^{n} y_i = 1$. A smooth map converts reserved tokens $T_i = Cy_i$ into activation propensities:

$$p_i(y) = 1 - e^{-Cy_i/c_i}, \qquad T_i = -c_i \log(1 - p_i). \tag{3}$$

Since $1 - e^{-z} \le z$, $\sum_i c_i p_i \le C(1 - y_0) \le C$. Retrieval enforces the budget C on each execution. Let $U(y) = \widetilde{U}(p(y))$. Fitness is

$$f_i(y) = \frac{\partial U}{\partial y_i} = \frac{C}{c_i}(1 - p_i) g_i(p), \qquad g_i(p) = u_i + \sum_{j \neq i} \beta_{ij} p_j, \quad f_0 = 0. \tag{4}$$

Thus $f_i/C = g_i/(\partial T_i/\partial p_i)$ measures utility per marginal reserved token.

Dynamics. Shares follow $\dot{y}_i = y_i(f_i - \bar{f})$, with $\bar{f} = \sum_{i=0}^{n} y_i f_i$. We use the discrete multiplicative-weights (Hedge) update (Freund & Schapire, 1997):

$$y_i^{(t+1)} = \frac{y_i^{(t)} \exp(\eta \tilde{f}_i^{(t)})}{\sum_{k=0}^{n} y_k^{(t)} \exp(\eta \tilde{f}_k^{(t)})}. \tag{5}$$

Fitness $\tilde{f}$ uses a posterior draw. New skills receive y_{init}, rescaling existing shares to preserve the simplex. Offspring inherit their parent Pareto state's population and posterior.

Proposition 3.1 (allocation ascent). For fixed u, symmetric β, and positive c, C, the simplex is invariant under the continuous dynamics, and

$$\frac{dU}{dt} = \sum_{i=0}^{n} y_i (f_i - \bar{f})^2 \ge 0, \tag{6}$$

with equality exactly at stationarity; every limit point is stationary (Appendix A.2).

Equal marginal returns. At a stationary allocation, all skills with positive shares have the same marginal utility per reserved token. At a constrained optimum, inactive skills offer no larger marginal return; unused capacity implies a zero common return (Proposition A.3). A sufficiently small Hedge step also increases U when using true fitness; sampled updates incur an estimation-error term (Theorem A.4). Appendix A.2 also gives a generalized Lotka–Volterra representation.

Random masking and centered effects. During training, independently retain each retrieved skill with probability $q = 1 - \rho$. For retrieval set R_e and retention indicator a_{ei}, use centered features $\kappa_{ei} = \mathbf{1}[i \in R_e](a_{ei} - q)$. Validation and test runs are unmasked. Fit a Gaussian working model

$$z_e = a_{g(e)} + \sum_i \kappa_{ei}\gamma_i + \sum_{i<j} \kappa_{ei}\kappa_{ej}\beta_{ij} + \varepsilon_e, \qquad \varepsilon_e \sim \mathcal{N}(0, \omega^2), \tag{7}$$

where $a_{g(e)}$ is a task-category intercept. Under stationary, homogeneous second-order effects and positive retrieval exposure, centering identifies β_{ij} despite arbitrary task/retrieval difficulty; it identifies γ_i as the skill's average marginal contribution under the retrieval distribution (Theorem A.5). With $\pi_i = \Pr(i \in R)$ and $\pi_{ij} = \Pr(i, j \in R)$, recover the main effects and allocation fitness as

$$u_i = \gamma_i - q \sum_{j \neq i} \beta_{ij} \frac{\pi_{ij}}{\pi_i}, \qquad g_i(p) = \gamma_i + \sum_{j \neq i} \beta_{ij} \left(p_j - q \frac{\pi_{ij}}{\pi_i} \right). \tag{8}$$

Within each estimation window, co-retrieval frequencies estimate π_{ij}/π_i; posterior samples use the same recovery. Unretrieved skills retain their initialization prior until $\pi_i > 0$.

Bayesian estimation. Use priors $\gamma_i \sim \mathcal{N}(0, \tau_u^2)$ and $\beta_{ij} \sim \mathcal{N}(0, \tau_\beta^2)$ with $\tau_\beta < \tau_u$. Collect the centered features and category indicators in ϕ_e, coefficients in w, and a proper Gaussian prior in

$\mathcal{N}(\bar{w}_0, V_0)$. The working posterior has precision $V^{-1} = V_0^{-1} + \omega^{-2} \sum_e \phi_e \phi_e^\top$ and mean $\bar{w} = V(V_0^{-1}\bar{w}_0 + \omega^{-2} \sum_e \phi_e z_e)$. Observed co-retrieval pairs are fitted; unobserved pairs retain their prior. Noise variance is estimated from residuals. Before each round, forgetting inflates $V \leftarrow V + \xi I$. An edited skill inherits its predecessor's posterior, adding τ_u^2 to its singleton variance and τ_β^2 to its interaction variances. Proposition A.6 establishes posterior-mean consistency in a fixed stationary model without continual forgetting.

Posterior-based decisions. *Allocation and retrieval:* draw $\tilde{w}$ by Thompson sampling, transform it using Eq. 8, and update y. For query q_x and active set A, rank candidates by $\alpha \,\mathrm{sim}(s_i, q_x) + \gamma \log p_i + \zeta \sum_{j \in A} \tilde{\beta}_{ij}$, adding skills greedily while budget C permits. *Retirement:* remove a skill if $\Pr(f_i > 0 \mid \mathrm{data}) < \epsilon$, recording a nogood (§3.5). With co-retrieval frequencies fixed, fitness is linear in the sampled coefficients, so this probability follows from the Gaussian posterior. *Mutation priority:* multiply p_i by the suspiciousness defined next.

3.3 Localizing Failures

Failures may originate in a skill, a tool, or a skill interaction, requiring different repairs. **Spectrum-based ranking.** For component $o \in \mathcal{S} \cup \mathcal{T}$, let $n_f(o)$ and $n_p(o)$ count failed and successful mini-batch trajectories activating o, and let N_f count all failed trajectories. Rank components by Ochiai suspiciousness (Abreu et al., 2007), $\mathrm{susp}(o) = n_f(o)/\sqrt{N_f(n_f(o) + n_p(o))}$.

Confirmation and repair sites. Counterfactual checks on highly ranked components identify four site types: **tool**, when call traces or unit tests reveal an implementation or contract failure; **interaction**, when $\Pr(\beta_{ij} < 0 \mid \mathrm{data}) > 1 - \epsilon$ but neither skill is individually suspicious; **shared**, when one component contributes to failures involving at least two distinct skills, making it an Extract candidate (§3.4); and **skill** otherwise. The reflector $\mathcal{R}$ uses the ranked sites, confirming evidence, and matched nogood remedies to propose a repair. A flat suspiciousness distribution triggers reflection over complete trajectories.

3.4 Expanding Context: Tool Evolution

Extraction rule. Following library learning (Ellis et al., 2021), SkillSmith extracts reusable abstractions into tools when the logic is **shared** by multiple skills or recurs across their failures, **deterministic** enough to specify correctness through unit tests, and **context-independent**, requiring no task-specific judgment. New capabilities begin in skills and enter the tool layer once they meet these conditions; orchestration and judgment remain in skills.

Extraction and capacity. Extracting logic of length ℓ shared by k skills into a tool with interface length ℓ_d and per-skill call-site length ℓ_δ changes each source skill's cost to

$$c_i' = c_i - \ell + \ell_\delta + \ell_d/k. \tag{9}$$

Costs fall when $\ell > \ell_\delta + \ell_d/k$; more sharing makes this condition easier to satisfy.

Proposition 3.2. Fix u and β, and let $U^\star(c) = \max_{y \in \Delta} \widetilde{U}(p(y; c))$ under Eq. 3. If $0 < c' \leq c$ componentwise, then $U^\star(c') \geq U^\star(c)$. The inequality is strict if an original optimum uses all reserved capacity with positive marginal value and at least one positively allocated skill becomes cheaper (Appendix A.4).

Behavior-preserving extraction thus frees capacity for the skill ecosystem: a form of niche construction (Odling-Smee et al., 2003).

Typed operations. Tool-Smith $\mathcal{B}_\tau$ selects operations using localization and the extraction rule (Appendix B). Patch fixes implementations under valid contracts; Retype revises contracts. Extract lifts eligible shared logic into tools; Inline returns it to dependents when sharing is lost or task-dependent branches accumulate. Required updates cover source components or callers, including tools.

Four typed operations

Operation	Effect	Update
Tool repair		
Patch	Implementation	None
Retype	Signature	Callers
Logic transfer		
Extract	New tool	Sources
Inline	Remove tool	Callers

Atomic bundles. A proposal $\mathcal{L} = (\text{tool operations}, \text{skill edits})$ includes all required caller updates, including tool callers. Under persistent exact-match dependencies, legal single-component edits cannot complete a breaking interface migration (Proposition A.9); RETYPE and INLINE therefore commit atomically with dependent edits. Static checks cover changed components and required callers (Proposition A.10); tests assess behavior. Source versions enable exact rollback; nogoods share the operation vocabulary (§3.5).

Acceptance. Bundles must pass validation without regression (§3.6). Among acceptable bundles, prefer larger $\Delta\text{perf} - \lambda_{\text{cost}}\Delta\bar{c}$, where $\Delta\bar{c}$ measures the change in expected context tokens per task, including interface and call-site overhead. Accuracy-preserving extraction can therefore be accepted for freeing context; increased inlining costs require a compensating performance gain.

3.5 SAVING ITERATIONS: CONFLICT-DRIVEN LEARNING

Inspired by conflict-driven learning in SAT solvers (Marques-Silva & Sakallah, 1999), memory $\mathcal{F}_t$ stores *nogoods* that prune previously failed library modifications. **Representation.** A nogood $\nu = (A, \text{scope}, r)$ records a failure-inducing set of modifications A, each described by its component and structured change; the relevant component versions as its scope; and a remedy r, either a repair or an `open` marker.

Learning causes. For a validation test that the parent passes and a proposed bundle fails, delta debugging (Zeller & Hildebrandt, 2002) isolates a smaller failing subset A under deterministic replay. Completed minimization yields a *1-minimal* set: deleting any single edit no longer reproduces that failure. Bundles usually contain at most five modifications; the n_{dd} rerun cap may stop before this guarantee is reached (Lemma A.12). Retirement and INLINE additionally record reconstruction warnings, distinguished from replay-verified failures.

Diagnosis, veto, and expiration. Retrieve similar nogoods and pass their remedies to $\mathcal{R}$. Veto a proposal when it matches every modification in an active nogood's A: use exact matching for structural changes and embedding similarity with threshold θ_{veto} for descriptive changes. Return a vetoed proposal to $\mathcal{R}$ with the remedy as a constraint; discard it after two unsuccessful revisions. Any version change within a nogood's scope expires it: expired records support diagnosis but cannot veto. A proposal matching an expired record proceeds to validation, whose outcome refreshes or removes that record. Exact-match veto is sound under version-scoped failure persistence; remedial edits and similarity matches need not satisfy this assumption (Proposition A.13).

3.6 VALIDATION AND FAILURE LEARNING

After proposal revision and nogood screening, Algorithm 1 validates candidates and learns from rejected edits.

Validate before committing. Stage the bundle on a parent copy. CHECK verifies dependency closure and call-site compatibility on affected components, then runs tool unit, integration, and regression tests on D_{val}. It returns outcome v and the first failing test t, if any. Validated candidates enter $\mathcal{G}$ only if accepted under Section 3.4. Accepted extraction updates costs for the next ecosystem update; rejection preserves skills, tools, and costs.

Algorithm 1 Validate, commit, or learn

Require: $\Sigma, \mathcal{L}, \mathcal{G}, \mathcal{F}, B_{\text{rem}}$
$\Sigma' \leftarrow \text{STAGE}(\Sigma, \mathcal{L})$
$(v, t) \leftarrow \text{CHECK}(\Sigma', B_{\text{rem}})$
if $v = \text{pass}$ and Σ' is acceptable **then**
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{\Sigma'\}$; **return** accept
end if
if $v = \text{fail}$ and $t(\Sigma) = \text{pass}$ **then**
  $A \leftarrow \text{DDMIN}(\mathcal{L}, t, \min(n_{\text{dd}}, B_{\text{rem}}))$
  $\nu \leftarrow (A, \text{SCOPE}(\Sigma, A, t), \texttt{open})$
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{\nu\}$
else
  Retain diagnostic evidence only
end if
return reject; preserve $(\mathcal{S}, \mathcal{T})$

Learn only reproducible causes. Replay t on the parent before minimization. If it passes, capped delta debugging yields a failing subset A, scoped to parent versions of edited components and relevant replay dependencies. The remedy is initially `open`. Otherwise, retain diagnostics without asserting an edit-induced cause. All tests, including baseline replay and minimization, debit the shared budget B_{rem}. Exhaustion yields an unresolved outcome, never acceptance; minimization retains its last confirmed failing subset.

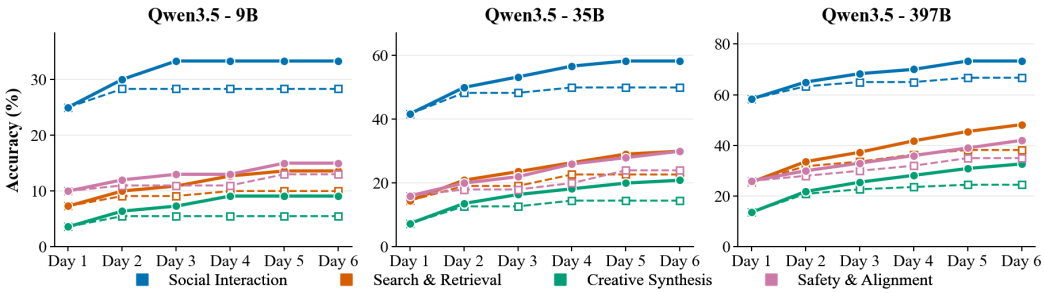


Figure 3: **Evolutionary Trajectories on WildClawBench.** Accuracy over 6 days across three Qwen3.5 scales. SkillSmith (solid) continues improving through skill–tool co-evolution, while the fixed-tool SkillClaw baseline (dashed) plateaus earlier.

Table 1: **Main results on OfficeQA and SealQA.** Accuracy (%) across Qwen3.5 scales (mean ± sample SD over five runs). **Red** denotes best and blue second best. Gains are in percentage points.

	OfficeQA (205 questions)					**SealQA** (92 questions)			
Method	9B	27B	35B	122B	397B	9B	35B	122B	397B
Base Agent	14.1 ±0.5	26.3 ±0.8	34.6 ±0.8	52.7 ±1.1	74.1 ±0.8	5.4 ±0.8	13.0 ±1.1	20.7 ±1.7	37.0 ±1.7
EvoSkill	16.1 ±0.5	30.7 ±0.8	41.0 ±0.8	63.9 ±1.1	85.9 ±0.5	6.5 ±0.8	17.4 ±1.1	28.3 ±1.7	50.0 ±1.7
SkillSmith (Ours)	**17.6** ±0.5	**34.1** ±0.8	**46.3** ±0.8	**72.2** ±0.8	**96.1** ±0.3	**7.6** ±0.8	**20.7** ±1.1	**33.7** ±1.7	**59.8** ±1.7
Δ vs Base	**+3.4**	**+7.8**	**+11.7**	**+19.5**	**+22.0**	**+2.2**	**+7.6**	**+13.0**	**+22.8**

4 Experiments

We organize the experiments around four questions. ▶ **RQ1 (Performance):** How does skill–tool co-evolution compare with skill-only evolution? ▶ **RQ2 (Mechanisms):** What do typed tool evolution, the skill ecosystem, and conflict-driven nogood learning each contribute? ▶ **RQ3 (Scaling):** How do gains vary with task complexity, skill co-activation, and context demand? ▶ **RQ4 (Stability):** How do the mechanisms support recovery, control library growth, and reduce regressions during continued evolution?

4.1 Experimental Setup

Benchmarks. We evaluate SkillSmith on three benchmarks. **OfficeQA** requires cross-document table localization and multi-step numerical reasoning, representing structured document scenarios that are highly sensitive to skill precision. **SealQA** examines factual question-answering ability when search results from the open web contain noise and conflicting information. **WildClawBench** simulates real-world agent deployment, featuring 15–50 step multi-modal tasks with multiple tools and high interaction density, making it a suitable testbed for evaluating joint Skill–Tool co-evolution.

Baselines. We select three methods representing different evolution paradigms as our primary comparisons: **Base Agent** (initial skill library $\mathcal{S}_0$ + fixed tool set $\mathcal{T}_0$, no evolution), **EvoSkill** (failure-driven Skill-only evolution + Pareto front selection), and **SkillClaw** (multi-user trajectory-driven collective Skill evolution). Reflection uses the same frozen backbone as execution at each scale, as in GEPA (Agrawal et al., 2025) and EvoSkill (Alzubi et al., 2026). Data splits, initial skill and tool libraries, scoring functions, and additional evaluation details are provided in Appendix D.

4.2 RQ1: Main Results and RQ2: Ablation Study

Main Results. As shown in Table 1, SkillSmith consistently outperforms both the Base Agent and the strong skill-only baseline, EvoSkill, on OfficeQA and SealQA. The gains increase with model scale: as model size increases from 9B to 397B, SkillSmith's gains over the Base Agent grow from

+3.4 to +22.0 percentage points (OfficeQA) and +2.2 to +22.8 percentage points (SealQA), showing larger absolute gains from the evolved external state at larger model scales.

On WildClawBench, Figure 3 shows higher terminal accuracy and more sustained improvement for SkillSmith than for SkillClaw. SkillClaw plateaus between Day 2 and Day 4, whereas SkillSmith continues improving to Day 6. This comparison supports evolving the tool layer together with its dependent skills rather than restricting every repair to skill text.

Ablation Study. Table 2 separates the three mechanisms. **Skill-only** fixes the tool library; **FreeTool** replaces typed tool operations with unrestricted tool edits; **−Ecosys** removes skill-ecosystem governance; and **−Nogood** removes persistent conflict-driven failure learning. Appendix D.2 defines the ablations.

Table 2: **Ablation Study Results (Qwen3.5-122B).** Regression Rate tracks the proportion of tasks where the updated system underperforms. Tool Error Rate tracks execution exceptions.

Variant	SealQA	WCB Avg.	Regress ↓	Skills + tools	Error ↓
SkillSmith (Full)	**33.7**	**41.4**	**2.1%**	14 + 6	**3.2%**
Skill-only	29.3	34.6	3.4%	16 + 9	—
FreeTool	31.5	37.8	8.9%	14 + 9	14.7%
−Ecosys	26.6	39.1	2.8%	21 + 7	4.1%
−Nogood	30.4	38.2	7.6%	15 + 6	3.5%

Tool evolution. Skill-only has the largest WCB drop, from 41.4 to 34.6 (−6.8 percentage points), and ends with 16 skills. FreeTool reaches 37.8 but raises the tool error rate to 14.7% (4.6× the full model) and regression to 8.9%. The comparison supports typed tool evolution with synchronized caller updates.

Allocating context. Without the skill ecosystem, the final library grows to 21 skills and 7 tools, compared with 14 and 6 for the full model. WCB accuracy falls from 41.4 to 39.1 and SealQA from 33.7 to 26.6. The joint contribution of allocation, interaction-aware retrieval, and retirement is to select useful skills from a growing library under budget C.

Saving iterations. Without nogood learning, WCB accuracy falls to 38.2 and regression rises from 2.1% to 7.6%. This supports retaining version-scoped failure evidence to guide diagnosis and veto repeated failed edits.

4.3 RQ3: Scaling Effects and RQ4: Evolutionary Stability

Scaling Effects. In Figure 4, multi-skill co-activation rises with task complexity, while gains over skill-only baselines grow from around 5–10% to over 20%. The right panel groups gains by tool complexity, interaction density, context size (loaded content), action space, and skill synergy. These trends highlight the value of interaction-aware allocation and reusable tool abstractions under higher context demand, as more retrieved skills compete for space in the same window.

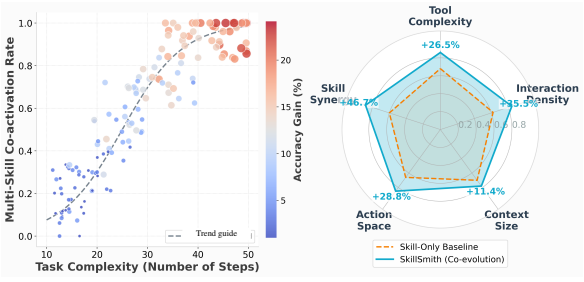


Figure 4: Scaling with task complexity. Left: skill co-activation and gains over skill-only; dashed: visual trend guide. Right: gains by task characteristic.

Evolutionary Stability. Figure 5a tests recovery after increased task complexity at iteration 30 and tool damage at iteration 60. Accuracy drops below 25% at its lowest. After the second perturbation, fixed-tool EvoSkill stays around 30%, while SkillSmith recovers to near 70% by iteration 100 with synchronized tool and caller repairs.

Without the ecosystem, the library exceeds 80 components (Figure 5b) while accuracy falls from 68% to below 35% after perturbation. The −Nogood trajectory fluctuates within 40%–60%; SkillSmith keeps fewer than 28 components and sustains recovery. These comparisons support context allocation and reusable failure evidence alongside tool repair.

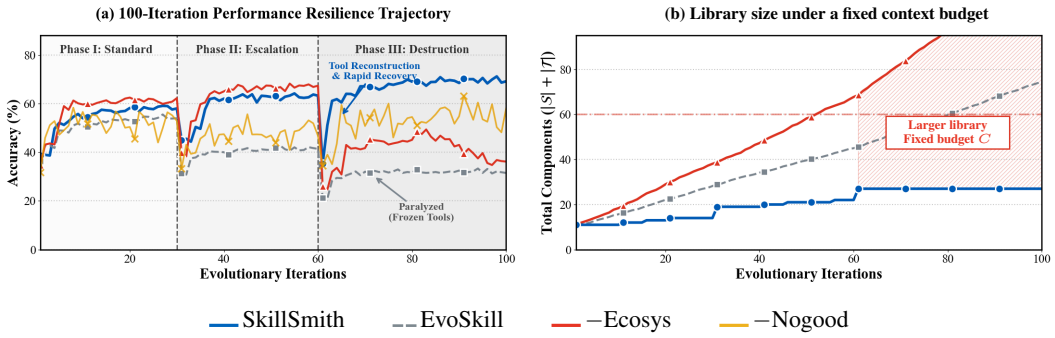


Figure 5: System resilience and library growth over 100 evolution rounds. (a) Accuracy after task and tool perturbations. (b) Persistent skill and tool counts under a fixed per-task context budget.

ILLUSTRATIVE TASK: *"Given a debt table with values in USD billions, compute the change from Q1 to Q3 and report it in USD millions."*

Before Joint Update ✗

Skill: `Treasury-Table-Lookup` `1. Parse PDF page 2. Call` **`table_extractor`**`(row:int, col:int) 3. Compute & return`

Tool: `table_extractor` v1 — fixed integer indexing, no merged-header resolution.

Step 1 Calls `extract(row=5,col=3)`; correctly reads Q1 as `"16,228"`.

Step 2 Calls `(row=5,col=7)` for Q3. A merged-header offset selects the wrong column: `"16,174"`.

Step 3 $16174 - 16228 = -54$. Omits the billion-to-million conversion. Output: `-54M`.

Expected: `+187,000M`. Column and unit errors remain unchecked.

After Joint Update ✓

Skill: `Treasury-Table-Lookup` v2 `1. Parse PDF 2. Call` **`table_extractor`**`(header_name) 3. Verify unit/sign 4. Compute`

Tool: `table_extractor` v2 (PATCH+RETYPE) — resolves merged headers; matches by name.

Step 1 Selects `row="Held by Public"`, `col="Q1"`; reads `"16,228"` and the unit `USD billions`.

Step 2 Resolves the merged header for `col="Q3"`; reads `"16,415"` in the same unit.

Step 3 Computes Q3 minus Q1: $16415 - 16228 = +187$ billion; converts $187 \times 1000 = 187{,}000$ million.

Output: `+187,000M`. ✓

Atomic Bundle $\mathcal{L}$: TOOL — **PATCH** header resolution + **RETYPE** signature to `(row_header:str, col_header:str)`. SKILL — replace integer refs with header names; add unit verification. CASCADE — all dependent skills adopt named headers and preserve source units.

Figure 6: **Illustrative table-QA repair (PATCH + RETYPE).** Revisiting the introduction's merged-header example, an atomic bundle repairs header resolution, revises the tool signature, and updates the dependent workflow, including source unit verification and conversion. Values are hypothetical.

5 CONCLUSION AND LIMITATIONS

SkillSmith co-evolves skills and tools under finite context and iteration budgets. Interaction-aware ecosystem dynamics allocate context, typed tool operations extract reusable logic, and version-scoped nogoods reuse failure evidence. These mechanisms connect a skill's value to its companions and execution cost: tools free space for the ecosystem, while failure memory saves search effort. Across three benchmarks and five Qwen3.5 scales, the reported comparisons show gains over skill-only evolution, especially at larger scales and higher task complexity.

Limitations. Progress depends on reflection quality, sparse interaction exposure, and posterior calibration during adaptive evolution. Fixed-model ascent and scoped veto guarantees do not ensure improvement on every sampled update. Broader tests of transfer, changing task distributions, embodied tasks, and multi-user deployment remain future work.

AI USE STATEMENT

Generative AI tools assisted with English translation and language polishing, refinement of theoretical statements and proofs, and preparation of illustrative figures and LaTeX formatting. The authors take responsibility for the final content, including all claims, proofs, and reported results.

REFERENCES

Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. On the accuracy of spectrum-based fault localization. In *Testing: Academic and Industrial Conference Practice and Research Techniques–MUTATION*, pp. 89–98, 2007. doi: 10.1109/TAICPART.2007.4344104.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.

Shuangrui Ding, Xuanlang Dai, Long Xing, Shengyuan Ding, Ziyu Liu, Jingyi Yang, Penghui Yang, Zhixiong Zhang, Xilin Wei, Yubo Ma, Haodong Duan, Jing Shao, Jiaqi Wang, Dahua Lin, Kai Chen, and Yuhang Zang. Wildclawbench, 2026. URL https://github.com/InternLM/WildClawBench.

Kevin Ellis, Lionel Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. DreamCoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pp. 835–850, 2021. doi: 10.1145/3453483.3454080.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. doi: 10.1006/jcss.1997.1504.

Josef Hofbauer and Karl Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998. doi: 10.1017/CBO9781139173179.

Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, et al. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445*, 2022.

Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. Skillclaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*, 2026.

João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999. doi: 10.1109/12.769433.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

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

F. John Odling-Smee, Kevin N. Laland, and Marcus W. Feldman. *Niche Construction: The Neglected Process in Evolution*. Princeton University Press, 2003.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Thinh Pham, Nguyen Nguyen, Pratibha Zunjare, Weiyuan Chen, Yu-Min Tseng, and Tu Vu. Sealqa: Raising the bar for reasoning in search-augmented language models. *arXiv preprint arXiv:2506.01062*, 2025.

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

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

The Mosaic Research Team. Introducing officeqa: A benchmark for end-to-end grounded reasoning. https://www.databricks.com/blog/introducing-officeqa-benchmark-end-to-end-grounded-reasoning, December 2025. Accessed: 2026-02-20.

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

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Andreas Zeller and Ralf Hildebrandt. Simplifying and isolating failure-inducing input. *IEEE Transactions on Software Engineering*, 28(2):183–200, 2002. doi: 10.1109/32.988498.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with Agent Skills. Anthropic Engineering, October 2025. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills. October 16, 2025.

A Theoretical Analysis

This appendix proves the allocation and capacity results in Section 3, then analyzes effect identification, bundle legality, and failure learning under their stated assumptions.

A.1 Setup and Budget Feasibility

Fix n skills, costs $c_i > 0$, budget $C > 0$, main effects u, and a symmetric interaction matrix β with zero diagonal. Define $\Delta = \{y \in \mathbb{R}^{n+1}_{\geq 0} : \sum_{i=0}^{n} y_i = 1\}$, with activation $p_i(y) = 1 - e^{-Cy_i/c_i}$ and utility $U(y) = \widetilde{U}(p(y))$. Here y_0 is the slack share and $T_i = Cy_i = -c_i \log(1 - p_i)$ is reserved capacity; expected token occupancy is $c_i p_i$. Write $c_{\min} = \min_i c_i$ and $g_i(p) = u_i + \sum_{j \neq i} \beta_{ij} p_j$. The bound $G = \max_i(|u_i| + \sum_j |\beta_{ij}|)$ gives $|g_i(p)| \leq G$. The zero diagonal makes $\widetilde{U}$ affine in each coordinate separately. Consequently, whenever both arguments are feasible,

$$\widetilde{U}(p + \varepsilon e_i) - \widetilde{U}(p) = \varepsilon g_i(p). \tag{10}$$

Extend U smoothly to a neighborhood of Δ, independently of y_0. Its gradient is $f_i = (C/c_i)(1 - p_i)g_i$ for $i \geq 1$ and $f_0 = 0$. Let $\bar{f} = \sum_{i=0}^{n} y_i f_i$ and $\mathrm{Var}_y(f) = \sum_{i=0}^{n} y_i (f_i - \bar{f})^2$.

Lemma A.1 (Budget feasibility). *For all $y \in \Delta$, $\sum_{i=1}^{n} c_i p_i(y) \leq C(1 - y_0) \leq C$.*

Proof. For $z \geq 0$, $1 - e^{-z} \leq z$. Thus $c_i p_i(y) \leq C y_i$; summing over $i \geq 1$ gives the claim. □

The lemma bounds expected occupancy in the allocation model. Greedy retrieval enforces the hard per-execution budget and generally induces dependent activations.

A.2 Ecosystem Dynamics

Theorem A.2 (Continuous allocation dynamics). *For $\dot{y}_i = y_i(f_i - \bar{f})$, $i = 0, \ldots, n$, the simplex and each of its faces are invariant. Along every trajectory, $\dot{U} = \mathrm{Var}_y(f) \geq 0$, with equality if and only if the state is stationary. Every ω-limit point is stationary.*

Proof. The smooth vector field gives unique solutions, $\sum_i \dot{y}_i = 0$ on Δ, and $y_i = 0$ implies $\dot{y}_i = 0$. Positive coordinates satisfy $y_i(t) = y_i(0) \exp(\int_0^t (f_i - \bar{f})\, ds)$, so they stay positive at finite times. Compactness of Δ gives global forward existence. By the chain rule,

$$\dot{U} = \sum_i f_i y_i (f_i - \bar{f}) = \sum_i y_i (f_i - \bar{f})^2,$$

since $\sum_i y_i (f_i - \bar{f}) = 0$. The variance vanishes exactly when $f_i = \bar{f}$ on every positive coordinate, equivalently $\dot{y} = 0$. Finally, U is bounded and nondecreasing on the compact invariant simplex. Its limiting value is constant on the invariant ω-limit set, so $\dot{U} = 0$ throughout that set; all its points are stationary. □

Proposition A.3 (Equal marginal returns and optimality conditions). *A point $y^* \in \Delta$ satisfies the KKT conditions for $\max_{y \in \Delta} U(y)$ if and only if, for $\lambda = \bar{f}(y^*)$,*

$$f_i(y^*) \leq \lambda \quad \textit{for all } i = 0, \ldots, n, \qquad f_i(y^*) = \lambda \quad \textit{whenever } y_i^* > 0. \tag{11}$$

Here $\lambda \geq 0$. If $y_0^* > 0$, then $\lambda = 0$ and $g_i(p^*) = 0$ for every positively allocated skill. If $y_0^* = 0$, every such skill satisfies $g_i(p^*)/(\partial T_i/\partial p_i) = \lambda/C$, while inactive skills have no greater marginal return. Every local maximizer satisfies these conditions, as does the limit of any convergent trajectory starting in the interior of Δ.

Proof. The simplex KKT equations are $\nabla U = \lambda \mathbf{1} - \mu$, $\mu \geq 0$, and $\mu_i y_i^* = 0$. These are precisely Eq. 11; multiplication by y^* and summation gives $\lambda = \bar{f}$. Since $f_0 = 0$, $\lambda \geq 0$, with equality if $y_0^* > 0$. Substitution of $f_i = C g_i/(\partial T_i/\partial p_i)$ proves the marginal-return statements. Linear simplex constraints satisfy the constraint qualification at every point, so local maxima satisfy KKT.

Suppose an interior trajectory converges to y^*. Theorem A.2 implies that y^* is stationary, hence $f_i(y^*) = \bar{f}(y^*)$ on its support. If an absent type k instead had $f_k(y^*) > \bar{f}(y^*)$, continuity would yield $\dot{y}_k/y_k > \delta > 0$ for all sufficiently large times. Because y_k remains positive, it would grow exponentially, contradicting convergence to $y_k^* = 0$. Thus no such type exists. □

Stationarity equalizes returns on the support; KKT additionally bounds returns for absent types. Since U may be nonconcave, stationary points need not be optima.

Ecological correspondence. On a trajectory with $y_0 > 0$, set $x_i = y_i/y_0$. Then $\dot{x}_i = x_i(f_i - f_0) = x_i f_i(y(x))$, where $y_i = x_i/(1 + \sum_j x_j)$. This is a generalized Lotka–Volterra representation (Hofbauer & Sigmund, 1998). Its nonlinear growth rates express competition and facilitation under the shared resource constraint C.

Theorem A.4 (A discrete ascent bound)**.** *Let y be interior, $y_i' = y_i e^{\eta f_i}/\sum_k y_k e^{\eta f_k}$, $D_f = \max_i f_i - \min_i f_i$, and $L = \sup_{y\in\Delta} \|\nabla^2 U(y)\|_2$. If*

$$0 < \eta \le \min\left\{\frac{1}{4L}, \frac{\log 2}{3D_f}\right\}, \tag{12}$$

then $U(y') - U(y) \ge (\eta/2)\mathrm{Var}_y(f)$. A zero denominator imposes no restriction. Moreover,

$$L \le \left(\frac{C}{c_{\min}}\right)^2 (\|\beta\|_2 + G), \qquad D_f \le \frac{2CG}{c_{\min}}. \tag{13}$$

Proof. Let $\psi_i = e^{\eta f_i}$, $Z = \sum_i y_i\psi_i$, and $\delta = y' - y$, so $\delta_i = y_i(\psi_i - Z)/Z$. The pairwise covariance identity and the mean value theorem give

$$\langle f, \delta\rangle = \frac{1}{2Z}\sum_{i,j} y_i y_j (f_i - f_j)(\psi_i - \psi_j) \ge \eta e^{-\eta D_f}\mathrm{Var}_y(f),$$

$$\|\delta\|_2^2 \le \frac{\mathrm{Var}_y(\psi)}{Z^2} \le \eta^2 e^{2\eta D_f}\mathrm{Var}_y(f).$$

For the first inequality use $Z \le e^{\eta f_{\max}}$ and the lower derivative bound $\eta e^{\eta f_{\min}}$; for the second use $y_i^2 \le y_i$, $Z \ge e^{\eta f_{\min}}$, and the upper derivative bound $\eta e^{\eta f_{\max}}$. Taylor's inequality on the convex simplex yields

$$U(y') - U(y) \ge \eta\mathrm{Var}_y(f)\left(e^{-\eta D_f} - \frac{L\eta}{2}e^{2\eta D_f}\right).$$

Under Eq. 12, the factor in parentheses is at least $2^{-1/3} - 2^{2/3}/8 > 1/2$.

For the curvature bound, the skill-coordinate Hessian is $\nabla^2 U = D\beta D + \mathrm{diag}(p_i'' g_i)$, where $D = \mathrm{diag}(p_i')$, $0 < p_i' \le C/c_{\min}$, and $|p_i''| \le (C/c_{\min})^2$; its slack row and column vanish. The triangle inequality gives the bound on L, and $|f_i| \le CG/c_{\min}$ gives the bound on D_f. □

Estimated fitness. If a step instead uses $\widehat{f}$, the same proof, with the range $D_{\widehat{f}}$ in Eq. 12 and the true curvature bound L, gives

$$U(y') - U(y) \ge \frac{\eta}{2}\mathrm{Var}_y(\widehat{f}) - \|\widehat{f} - f\|_\infty \|y' - y\|_1. \tag{14}$$

The extra term follows from $\langle f - \widehat{f}, y' - y\rangle$. A sampled update increases utility whenever its variance gain exceeds this estimation-error term.

A.3 Identifying and Estimating Effects

Let x_e be a task, R_e its retrieved set, and $Z_e(A) \in [-1, 1]$ the potential residual for retained set $A \subseteq R_e$. Put $q = 1 - \rho$ and $A_e = \{i \in R_e : a_{ei} = 1\}$, with observation $z_e = Z_e(A_e)$.

Assumption A.1 (Conditional randomization)**.** *Conditional on (x_e, R_e), the retention indicators for retrieved skills are independent* $\mathrm{Bernoulli}(q)$ *draws, independent of all potential outcomes $\{Z_e(A) : A \subseteq R_e\}$.*

Assumption A.2 (A stationary second-order window)**.** *Episodes are independent and identically distributed within a fixed estimation window. For all $A \subseteq R_e$,*

$$\mathbb{E}[Z_e(A) \mid x_e, R_e] = h(x_e, R_e) + \sum_{i \in A} u_i + \sum_{i<j:\, i,j \in A} \beta_{ij}, \tag{15}$$

where h is arbitrary and the effects u, β do not vary across tasks or retrieval sets in that window.

Define $\kappa_{ei} = \mathbf{1}[i \in R_e](a_{ei} - q)$, setting it to zero for unretrieved skills. Let ϕ_e contain all singleton features κ_{ei} and pair features $\kappa_{ei}\kappa_{ej}$, $i < j$. Denote $\pi_i = \Pr(i \in R)$ and $\pi_{ij} = \Pr(i, j \in R)$.

Theorem A.5 (Identification by centered masking)**.** *Under Assumptions A.1–A.2, suppose $0 < q < 1$ and every singleton and pair in the fixed model has positive retrieval exposure. Then $M = \mathbb{E}[\phi\phi^\top]$ is diagonal and positive definite, with entries $q(1-q)\pi_i$ and $q^2(1-q)^2\pi_{ij}$. The population least-squares coefficients $M^{-1}\mathbb{E}[\phi z]$ are β_{ij} for pair features and*

$$\gamma_i = u_i + q \sum_{j \neq i} \beta_{ij} \frac{\pi_{ij}}{\pi_i} = \mathbb{E}[Z(A_{-i} \cup \{i\}) - Z(A_{-i}) \mid i \in R] \tag{16}$$

for singleton features. The result does not require a model for h. Adding nonredundant task-category intercepts leaves these coefficients unchanged.

Proof. Given (x, R), centering yields independent zero-mean κ_i with conditional variance $q(1-q)\mathbf{1}[i \in R]$. Randomization and Eq. 15 give

$$\begin{aligned} \mathbb{E}[z \mid x, R, a] = H(x, R) &+ \sum_{i \in R} \kappa_i \left(u_i + q \sum_{j \in R \setminus \{i\}} \beta_{ij} \right) \\ &+ \sum_{i<j:\, i,j \in R} \kappa_i \kappa_j \beta_{ij}, \end{aligned} \tag{17}$$

where H depends only on (x, R). A product of two distinct features contains at least one centered factor to the first power. Its conditional expectation is therefore zero. This proves diagonality of M; the diagonal entries follow from the conditional variances and exposure probabilities. The same argument makes every feature orthogonal to H and to task-category indicators.

The residual $z - \mathbb{E}[z \mid x, R, a]$ is orthogonal to all features. Multiplying Eq. 17 by $\kappa_i\kappa_j$ and taking expectations thus gives $\beta_{ij}q^2(1-q)^2\pi_{ij}$. Multiplication by κ_i gives

$$q(1-q)\,\mathbb{E}\left[\mathbf{1}[i \in R] \left(u_i + q \sum_{j \in R \setminus \{i\}} \beta_{ij} \right)\right].$$

Dividing by the corresponding diagonal entries proves the coefficient identities. The marginal-contribution expression follows by differencing Eq. 15 and averaging over the other retained skills. □

From retrieval effects to allocation fitness. Solving Eq. 16 for u_i gives Eq. 8. The retrieval-average effect γ_i equals $g_i(p)$ when $q\pi_{ij}/\pi_i = p_j$ for the relevant interactions. Unexposed pairs retain their prior because their effects are unidentifiable.

Proposition A.6 (Consistency in a fixed model)**.** *Under Theorem A.5, take a fixed finite-dimensional feature set, a proper Gaussian prior $\mathcal{N}(\bar{w}_0, V_0)$ with $V_0 \succ 0$, and a fixed working variance $\omega^2 > 0$. Without continual forgetting, the posterior mean in Section 3.2 converges almost surely to the population regression coefficients. Its covariance is $O(N^{-1})$, and posterior draws converge in probability to those coefficients. The statement also allows nonredundant category indicators whose augmented population Gram matrix is positive definite.*

Proof. For N episodes, divide both the posterior precision and its mean's right-hand factor by N:

$$\bar{w}_N = \left(N^{-1}V_0^{-1} + \omega^{-2}N^{-1}\Phi_N^\top\Phi_N\right)^{-1} \left(N^{-1}V_0^{-1}\bar{w}_0 + \omega^{-2}N^{-1}\Phi_N^\top z_N\right).$$

Bounded features and outcomes allow the strong law of large numbers. The two sample moments converge to $M \succ 0$ and $\mathbb{E}[\phi z]$, respectively; the prior terms vanish. Continuity of inversion proves mean consistency, and $NV_N \to \omega^2 M^{-1}$ proves the covariance rate. Conditional posterior variance therefore tends to zero, which, together with mean consistency, gives convergence in probability of posterior draws. The augmented-feature case has the same proof. Empirical π_i and π_{ij} also converge, so the recovered u is consistent when all required effects are identified. □

Uncertainty and exploration. The working posterior is mean-consistent under the proposition's conditions; its credible intervals require calibration under noise misspecification. Under an additional homoskedastic residual approximation, a pair coefficient has asymptotic variance $\sigma_r^2/[N\pi_{ij}q^2(1-q)^2]$. Holding exposure and residual variance fixed, masking at $\rho = 1/2$ maximizes this information factor; smaller ρ preserves more training-time functionality. At $\rho = 0.2$, the variance factor is approximately $39.1/(N\pi_{ij})$, motivating shrinkage with scarce pair observations. With $\rho = 0$, all centered features vanish; arbitrary retrieval difficulty can absorb the skill effects, so the stated model does not identify them from observational logs alone.

A.4 Context Costs and Attainable Utility

For positive c, the smooth allocation map has image

$$\mathcal{P}(c) = \{p(y;c) : y \in \Delta\} = \left\{ p \in [0,1)^n : -\sum_i c_i \log(1-p_i) \le C \right\}. \tag{18}$$

Indeed, the inverse is $y_i = -c_i \log(1-p_i)/C$, with the remainder assigned to y_0. Define $U^\star(c) = \max_{p \in \mathcal{P}(c)} \widetilde{U}(p)$; the maximum exists because $\mathcal{P}(c)$ is the continuous image of compact Δ.

Proposition A.7 (Cost reduction and capacity)**.** *Fix u, β and positive costs $0 < c' \le c$. Then $U^\star(c') \ge U^\star(c)$. Further, suppose an optimizer y^* for c has $y_0^* = 0$ and $\lambda = \bar{f}(y^*) > 0$. If $c'_k < c_k$ for some k with $p_k^* > 0$, then $U^\star(c') > U^\star(c)$.*

Proof. Since $-\log(1-p_i) \ge 0$, reducing costs yields $\mathcal{P}(c) \subseteq \mathcal{P}(c')$, proving nondecrease. Under the strict conditions, $-\sum_i c'_i \log(1-p_i^*) < C$, so the original optimizer leaves positive reserved capacity under c'. Proposition A.3 implies $f_i(y^*) = \lambda > 0$ on each positive skill coordinate, hence $g_i(p^*) > 0$ there. For such an i and sufficiently small $\varepsilon > 0$, continuity of the cost expression ensures $p^* + \varepsilon e_i \in \mathcal{P}(c')$. Equation 10 gives a strict gain $\varepsilon g_i(p^*) > 0$. □

Corollary A.8 (Behavior-preserving extraction)**.** *An* EXTRACT *operation with $\ell > \ell_\delta + \ell_d/k$, positive resulting costs, unchanged costs for other skills, and unchanged effects (u, β) cannot decrease $U^\star$. It increases $U^\star$ strictly under the additional conditions of Proposition A.7.*

Proof. Equation 9 decreases the source skills' costs componentwise. Apply Proposition A.7. □

The result isolates the capacity gain of behavior-preserving extraction after reallocation. Validation assesses any accompanying changes in skill effects or tool reliability.

A.5 Atomic Bundles and Local Legality

Legality means tool existence and static call-site compatibility (Section 2.1).

Proposition A.9 (Interface freezing with persistent exact-match calls)**.** *Start in a legal state. Fix a tool identity τ and a nonempty set of persistent call sites to it in other components. Suppose legality requires each call-site signature to equal $\sigma(\tau)$, there are no overloads or compatibility adapters, and every update changes at most one component in place, with legality enforced after each update. If the tool and these calls cannot be removed or redirected, then $\sigma(\tau)$ and their call-site signatures remain unchanged throughout every legal update sequence.*

Proof. Consider the first change to any of these signatures. If it changes the tool, all persistent caller components are unchanged, and their signatures still match the old tool signature. If it changes a caller, the tool is unchanged. Either case creates a mismatch, contradicting legality. Thus no first change exists. □

SkillSmith groups interface migrations and caller updates in atomic commits. The same policy groups INLINE deletion with its required call-site edits.

For an operation o, let its required skill updates be

$$R(o) = \begin{cases} \emptyset, & o = \textsc{Patch}, \\ \mathrm{Dep}(\tau), & o = \textsc{Retype}_\tau \text{ or } \textsc{Inline}_\tau, \\ \mathrm{Src} \cap \mathcal{S}, & o = \textsc{Extract}_{\mathrm{Src}}. \end{cases}$$

Define $R_{\mathcal{T}}(o)$ analogously using tool callers $\mathrm{Dep}_{\mathcal{T}}(\tau)$ and source tools $\mathrm{Src} \cap \mathcal{T}$. Required caller sets are determined from the pre-update dependency graph. Let $\mathcal{S}_{\mathcal{L}}$ and $\mathcal{T}_{\mathcal{L}}$ include all components directly changed, created, or removed by the bundle, including companion operations. The affected sets are

$$\mathcal{A}_{\mathcal{S}} = \mathcal{S}_{\mathcal{L}} \cup \bigcup_{o \in \mathcal{L}} R(o), \qquad \mathcal{A}_{\mathcal{T}} = \mathcal{T}_{\mathcal{L}} \cup \bigcup_{o \in \mathcal{L}} R_{\mathcal{T}}(o). \tag{19}$$

Proposition A.10 (Locality of static legality)**.** *Let $\Sigma \in \Omega$ and let a well-formed bundle specify fresh identities for new tools, a consistent final edit per changed component, and the complete required update sets above. All call sites outside $\mathcal{A}_{\mathcal{S}} \cup \mathcal{A}_{\mathcal{T}}$ remain legal. Hence the updated state is legal if and only if every call site in the surviving or newly created components of those affected sets is legal.*

Proof. An outside component is unchanged. Consider any tool it calls. An untouched tool retains its identity and signature; a PATCH target also retains both. If the target were retyped or removed, the component would belong to the corresponding required caller set and hence the affected set, a contradiction. An unchanged call cannot refer to a fresh tool identity. Thus all outside calls remain legal. Checking all remaining calls proves the equivalence, including calls introduced inside extracted or patched tools. □

After local static checks, unit, integration, and regression tests validate behavior (Section 3.6).

A.6 ITERATION COSTS AND FAILURE MEMORY

Proposition A.11 (A shared-defect repair model)**.** *Suppose k dependent skills need repair. Each skill-only attempt targets one still-unrepaired skill, succeeds independently with probability $p_s > 0$, and costs d_s charged executions. Success repairs that skill without regression. Each joint attempt repairs all k skills with probability $p_b > 0$ and costs d_b executions. With independent attempts and no budget truncation, expected attempts are $\mathbb{E}[N_s] = k/p_s$ and $\mathbb{E}[N_b] = 1/p_b$. Expected charged executions are $\mathbb{E}[B_s] = kd_s/p_s$ and $\mathbb{E}[B_b] = d_b/p_b$. Joint repair uses fewer expected executions exactly when $p_b/d_b > p_s/(kd_s)$.*

Proof. Each of the k local repairs takes a geometric number of attempts with mean $1/p_s$; the joint repair takes one with mean $1/p_b$. Sum the local expectations and multiply by the fixed cost per attempt. □

The factors d_s, d_b convert proposal counts into the execution budget of Section 2.3.

Lemma A.12 (Completed delta debugging)**.** *Let $\mathcal{L}$ be a finite set of edits and let a deterministic replay test satisfy $T(\mathcal{L}) = \text{fail}$ and $T(\emptyset) = \text{pass}$, with outcomes in $\{\text{pass}, \text{fail}, ?\}$. Completed* `ddmin` *(Zeller & Hildebrandt, 2002) returns $A \subseteq \mathcal{L}$ with $T(A) = \text{fail}$ and $T(A \setminus \{e\}) \neq \text{fail}$ for every $e \in A$. It has worst-case quadratic test complexity in $|\mathcal{L}|$.*

Proof. The algorithm keeps a failing current set, replacing it only by a tested failing subset or complement. If neither reduction succeeds, it increases partition granularity. At termination, either one edit remains or every single-edit deletion has been tested without reproducing failure, giving 1-minimality. There are at most $|\mathcal{L}| - 1$ successful size reductions; the geometric refinement scans between reductions use $O(|\mathcal{L}|)$ tests, giving the quadratic worst-case bound. This is the standard `ddmin` guarantee (Zeller & Hildebrandt, 2002). □

Algorithm 2 SkillSmith: detailed evolution loop

Require: Initial state Σ_0; $D_{\text{train}}, D_{\text{val}}$; iteration budget B; context budget C
$\mathcal{G} \leftarrow \{\Sigma_0\}$; $\mathcal{F} \leftarrow \emptyset$; $b \leftarrow 0$
while $b < B$ **do**
  $\Sigma_k \leftarrow \textsc{SelectParent}(\mathcal{G})$; sample $\tilde{w}$ from its posterior
  Sample $M \subset D_{\text{train}}$; retrieve with $\tilde{w}$, mask with probability ρ, and execute Σ_k on M
  $V \leftarrow V + \xi I$; update centered posterior and co-retrieval frequencies; resample and transform $\tilde{w}$; update y
  Retire skills with $\Pr(f_i > 0 \mid \text{data}) < \epsilon$; add their nogoods to $\mathcal{F}$
  $F \leftarrow \{x \in M : P(x) < \theta\}$; $\Lambda \leftarrow \textsc{Localize}(M, F)$
  $\mathcal{L} \leftarrow \mathcal{R}(\Lambda, \textsc{Retrieve}(\mathcal{F}, F), y, \bar{w})$; invoke $\mathcal{B}_\tau$ for tool operations
  $\mathcal{L} \leftarrow \textsc{VetoAndRevise}(\mathcal{L}, \mathcal{F})$
  if $\mathcal{L} \neq \emptyset$ **then**
    $\Sigma' \leftarrow \mathcal{U}(\Sigma_k, \mathcal{L})$
    if Σ' passes validation and the acceptance criterion **then**
      $\mathcal{G} \leftarrow \mathcal{G} \cup \{\Sigma'\}$
    else
      $\mathcal{F} \leftarrow \mathcal{F} \cup \textsc{LearnNogood}(\Sigma_k, \mathcal{L})$
    end if
  end if
  $b \leftarrow b + b_{\text{iter}}$
end while
return $\arg\max_{\Sigma \in \mathcal{G}} \mathbb{E}_{x \sim D_{\text{val}}}[P(x \mid \Sigma)]$

Illegal subsets yield ?; tests use a fixed replay environment and random seeds. Under the n_{dd} cap, the result remains a confirmed failing subset, with 1-minimality guaranteed on completion. One-minimality concerns single-edit deletions, not minimum cardinality.

Assumption A.3 (Version-scoped failure persistence)**.** *For a recorded failure-inducing edit set A, fix its baseline component versions and replay environment. Every admissible candidate containing these same structured edits, with matching baseline versions throughout the record's scope, fails the associated validation. In particular, additional edits neither override the recorded edits nor repair their failure.*

Proposition A.13 (Conditional soundness and expiration of veto)**.** *Under Assumption A.3, exact matching against an active nogood rejects only candidates that would fail validation, avoiding their validation executions. A scope-version change deactivates that record, so it cannot veto the changed state unless refreshed by new validation.*

Proof. An exact active match satisfies the persistence assumption and therefore fails validation. Rejecting it before validation avoids the executions that would otherwise test it. Deactivation after a scope change follows directly from the expiration rule. □

Exact-match soundness depends on failure persistence. Similarity matching and reconstruction warnings remain heuristic; expiration deactivates the corresponding record.

A.7 Evolution-Loop Details

Algorithm 2 gives the complete evolution loop shown in Figure 2. Its validation and failure-learning branch is expanded in Algorithm 1. The following implementation details complement the pseudocode.

Execution and ecosystem updates. Each parent maintains a population and posterior. Before updating with a training minibatch, inflate covariance by ξI; then update the centered coefficients and co-retrieval frequencies. Transform the sampled effects using Eq. 8, apply the Hedge update, and record reconstruction warnings for retired skills.

Localization and proposals. Localization passes ranked repair sites and evidence to the reflector, together with retrieved nogood remedies. Tool-Smith instantiates typed operations. Vetoed proposals receive up to two revisions before being discarded.

Algorithm 3 Instance-Level Pareto Candidate Selection

Require: Pareto front $\mathcal{G}$, instance score matrix S
for each instance $x_i \in D_{\text{train}}$ **do**
  $s^*[i] \leftarrow \max_{\Sigma \in \mathcal{G}} S[\Sigma][x_i]$
  $\mathcal{G}^*[i] \leftarrow \{\Sigma \in \mathcal{G} : S[\Sigma][x_i] = s^*[i]\}$
end for
$\mathcal{C} \leftarrow \bigcup_i \mathcal{G}^*[i]$ *% candidates appearing in any instance-best set*
$\mathcal{D} \leftarrow \varnothing$
while $\exists\, \Sigma \in \mathcal{C} \setminus \mathcal{D}$ dominated by another in $\mathcal{C} \setminus \mathcal{D}$ **do**
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\Sigma\}$ *% remove dominated candidates*
end while
$\hat{\mathcal{G}}^*[i] \leftarrow \mathcal{G}^*[i] \setminus \mathcal{D}$ for all i
$f[\Sigma] \leftarrow |\{i : \Sigma \in \hat{\mathcal{G}}^*[i]\}|$ for each $\Sigma \in \mathcal{C} \setminus \mathcal{D}$ *% win count*
Sample Σ_k from $\mathcal{C} \setminus \mathcal{D}$ with $p(\Sigma_k) \propto f[\Sigma_k]$
return Σ_k

Validation and memory. Algorithm 1 validates each nonempty proposal, prioritizing tasks that use affected components. Delta-debugging subsets are applied to the same parent; illegal subsets are unresolved. Scope records include edited components and replay dependencies. Reflection can later replace an `open` remedy with a repair.

Budget accounting. The charged count b_{iter} includes all executions used by these routines, including rejected proposals and failure minimization. A routine starts an execution only while budget remains; an incompletely validated candidate is not accepted. The loop returns the highest-scoring validated state retained in $\mathcal{G}$.

A.8 Parent Selection

Algorithm 3 samples parents by instance-level Pareto performance, retaining candidates that achieve the best score on at least one task and removing dominated states.

B Typed Operations for Tool Evolution

Each operation takes $\mathcal{T}_t$ and a specification from $\mathcal{R}$, and returns a modified library, required caller updates, and unit tests. The enclosing bundle supplies skill edits for R and tool edits for $R_{\mathcal{T}}$ (Eq. 19). These sets are empty for Patch, contain the corresponding callers for Retype and Inline, and contain source components for Extract. Write $\tau = (d, f, \sigma)$.

Definition 1 (Patch$(\tau, \Delta f)$)**.** *Produce $\tau' = (d', f', \sigma)$, preserving the signature. The preferred implementation is a wrapper, $f' = \text{post} \circ f \circ \text{pre}$: the original f remains intact and removing the wrapper restores it. Direct edits to f are also permitted when needed. The synchronized update set is $R = \emptyset$. Unit tests include the tool's contract tests and a regression test reproducing the failed call. Typical uses include input validation, output normalization, retries, and header parsing.*

Definition 2 (Retype(τ, σ'))**.** *Produce $\tau' = (d', f', \sigma')$, where $\sigma' \neq \sigma$ and f' is adapted to the new signature. The synchronized update set is $R = \text{Dep}(\tau)$: every dependent skill's call site is rewritten to match σ'.* Retype *is the only operation that changes the signature of an existing tool.*

Definition 3 (Extract(Src, g))**.** *Let* Src *be source components containing near-duplicate logic g: at least two skills, or an existing tool. Create $\tau_g = (d_g, f_g, \sigma_g)$ and replace g in each source component with a call to τ_g. The synchronized skill update set is $R = \text{Src} \cap \mathcal{S}$. The implementation f_g must pass unit tests derived from the source components' behavior, implementing the extraction rule's determinism condition.*

Extract covers both composition and decomposition. When g orchestrates calls to τ_1 and τ_2, extraction creates a composite tool. When the source is an existing tool, extraction splits out a reusable component: a companion Patch changes the source tool to call τ_g. If callers should use τ_g directly and their interface must change, the bundle also includes the corresponding Retype.

Definition 4 (INLINE(τ))**.** *For every $s \in \mathrm{Dep}(\tau)$, replace its call to τ with the tool's logic as a script or workflow inside s, then remove τ from $\mathcal{T}'$. The synchronized update set is $R = \mathrm{Dep}(\tau)$. Dependent tools inline the logic through companion implementation edits. If both caller sets are empty, the operation simply deletes the tool. Record a nogood in $\mathcal{F}_{t+1}$ to prevent reconstruction of a functionally equivalent tool within the record's version scope.*

Structural context costs. With unchanged description and call-site lengths, PATCH has $\Delta c = 0$ and RETYPE has $\Delta c \approx 0$. For each source skill, EXTRACT changes cost by Eq. 9; INLINE reverses this change. Acceptance uses the measured total change, including interface and call-site overhead (Section 3.4).

Locality. The operation type and target determine R and $R_{\mathcal{T}}$ before execution. Static dependency checks cover the affected skills and tools in Eq. 19, including all newly introduced call sites (Proposition A.10). Changed tools additionally undergo unit tests, followed by integration and regression validation.

Reversibility. For a newly created tool, EXTRACT and INLINE reverse the transfer across the boundary. Retaining source versions and call sites makes this reversal exact. The inverses of PATCH and RETYPE restore the original implementation and signature, respectively; reversing a bundle restores its constituent changes in reverse order.

Coverage. Tool-layer changes decompose into these four operations: PATCH or RETYPE modifies an existing tool, INLINE removes one, and EXTRACT creates one. A new capability is first implemented in the skill that uses it and enters the tool layer through EXTRACT once it meets the extraction rule.

C TOOL EXTRACTION CASE STUDY

Figure 7 illustrates a multi-round skill–tool update on WildClawBench. Implementation and interface repairs restore the data-processing pipeline; extracting shared channel-resolution logic then updates the source skills together.

D EXPERIMENTAL DETAILS

D.1 DATA SPLITS

OfficeQA (The Mosaic Research Team, 2025). The dataset contains 246 questions over ∼89k pages of U.S. Treasury Bulletin archives. Following the stratified protocol of EvoSkill (Alzubi et al., 2026), we allocate 10% for training (24 questions, used for failure detection during evolution), 7% for validation (17 questions, used for Pareto front selection and regression checks), and the remainder (205 questions) as a held-out test set that is never exposed during evolution. All final numbers in Table 1 are reported exclusively on this held-out set. Stratification ensures that each document category (monthly bulletin, quarterly supplement, annual summary) is represented proportionally across all three splits.

SealQA (Pham et al., 2025). We use the `seal-0` split (111 questions). The same 10%/7%/rest ratio yields 11 training, 8 validation, and 92 test questions, stratified by question difficulty tier (easy/medium/hard as annotated by the benchmark authors).

WildClawBench (Ding et al., 2026). We adopt the day–night cycle protocol of SkillClaw (Ma et al., 2026): 6 evolution rounds, each consisting of a daytime phase (8 concurrent simulated users interact with the agent, generating execution trajectories) and a nighttime phase (the system performs reflection, bundle proposal, validation, and deployment). We evaluate on four representative categories—Social Interaction (6 tasks, 60 trials), Search & Retrieval (11 tasks, 110 trials), Creative Synthesis (11 tasks, 110 trials), and Safety & Alignment (10 tasks, 100 trials)—where each task is independently tested 10 times per day.

TASK: *"Retrieve the top-3 cited papers from the user's Zotero library on federated learning, generate a comparative summary table as a LaTeX PDF, attach it to a new Slack thread in #research, and pin the message."*

Before Co-Evolution (Day 1) ✗

Skills involved: `Academic-Search`, `Report-Generator`, `Slack-Message-Handler`
Tools involved: `web_search`, `pdf_parser`, `fs_ops`, `slack_api`

Step 1-3 `Academic-Search` retrieves papers via `web_search`. Returns raw HTML snippets including irrelevant metadata tags. Passes output to `Report-Generator`.

Step 4-6 `Report-Generator` calls `pdf_parser` to read cited PDFs, but `pdf_parser` returns unstructured text with broken table layouts. Skill attempts to generate LaTeX table from noisy input; produces a PDF with misaligned columns and missing citation counts.

Step 7-9 `Slack-Message-Handler` calls `slack_api` to post the PDF. API call succeeds but the file is uploaded as a generic binary blob without preview. Pin operation fails: skill passes channel name but `slack_api` expects channel ID.

Result: 1/5 checkpoints passed (only "message posted" checkpoint; table content, formatting, attachment preview, and pin all failed). Three independent failure sources across two tools and one skill.

After Co-Evolution (Day 4) ✓

Iter. 2 — Bundle $\mathcal{L}_2$:
TOOL: **PATCH** `web_search` with a post-processing layer that strips HTML tags and normalizes output to structured JSON (title, authors, year, citation count).
TOOL: **RETYPE** `pdf_parser` to add a `mode="table"` parameter that returns tabular content as row-column arrays instead of raw text.
SKILL: Update `Report-Generator` to use structured JSON input and `mode="table"` extraction.

Iter. 3 — Bundle $\mathcal{L}_3$:
TOOL: **EXTRACT** the Slack channel-name-to-ID resolution logic (previously duplicated in 2 skills) into a new `slack_resolver` tool.
TOOL: **PATCH** `slack_api` to auto-set MIME type for PDF uploads (enabling in-channel preview).
SKILL: Update both source skills, `Slack-Message-Handler` and `Calendar-Coordinator`, to call `slack_resolver`.

Day 4 execution Clean retrieval → structured table → correctly formatted PDF → Slack post with preview → pin succeeds.

Result: 5/5 checkpoints passed. ✓

Evolution anatomy: This task required *two consecutive bundles* across *two iterations* to fully resolve. $\mathcal{L}_2$ addressed the data-quality pipeline (search output → table parsing), involving 2 tool operations (PATCH + RETYPE) and 1 skill update. $\mathcal{L}_3$ addressed the delivery pipeline (Slack posting), involving 2 tool operations (EXTRACT + PATCH) and synchronized updates to the source skills. Factoring this into a shared `slack_resolver` tool eliminated 23 lines of duplicated code across two skills.

Figure 7: **Multi-Round Skill–Tool Co-Evolution on WildClawBench (Creative Synthesis).** A complex 9-step task fails at three independent points spanning two tools and one skill. SkillSmith resolves all failures over two evolution rounds through four tool operations and synchronized skill updates, including EXTRACT for channel-resolution logic shared across skills.

D.2 ABLATION DEFINITIONS

Table 2 and Figure 5 compare the following module configurations.

Skill-only: fixed tools. The tool library is held fixed and evolution acts on skills. This removes tool-layer PATCH, RETYPE, EXTRACT, and INLINE proposals, testing whether skill edits alone can compensate for tool limitations.

FreeTool: unrestricted tool edits. Tool edits are proposed without typed operation templates or their prescribed synchronized update sets.

−Ecosys: without the skill ecosystem. Disable the ecosystem guidance in Section 3.2: marginal-utility allocation, interaction-aware retrieval, and posterior-based retirement. Task-based retrieval and the hard execution budget C remain active.

−**Nogood: without conflict-driven learning.** Disable persistent failure learning through $\mathcal{F}_t$: minimization, version-scoped storage with remedies, and reuse for diagnosis and veto. Per-episode localization, reflection, and candidate validation remain active.

D.3 Initial Skill and Tool Libraries

Table 3 and Table 4 list the complete initial configurations for each benchmark. These are shared identically across all methods to ensure a fair starting point.

Table 3: Initial Skill library $\mathcal{S}_0$ for each benchmark.

Benchmark	Skill Name	Referenced Tools
OfficeQA	`Doc-Retrieval-Strategy`	`pdf_parser`
	`Table-Parse-Flow`	`table_extractor, pdf_parser`
	`NumCalc-Protocol`	`formula_calc, unit_converter`
SealQA	`Search-Plan-Strategy`	`web_search, page_fetcher`
	`Source-Credibility-Eval`	`dedup_ranker`
WildClawBench	`Slack-Message-Handler`	`slack_api`
	`Calendar-Coordinator`	`calendar_api, gmail_api`
	`Academic-Search`	`web_search`
	`File-Preflight-Check`	`fs_ops`
	`Creative-Pipeline`	`img_processor, video_processor`
	`Git-Ops`	`git_cli`
	`Safety-Audit`	`code_sandbox`
	`Report-Generator`	`pdf_parser, fs_ops`

Table 4: Initial Tool library $\mathcal{T}_0$ for each benchmark.

Benchmark	Tool Name	Signature
OfficeQA	`pdf_parser`	`(filepath, page_range) → parsed_doc`
	`table_extractor`	`(pdf, page, row:int, col:int) → str`
	`formula_calc`	`(expression:str) → float`
	`unit_converter`	`(value, from_unit, to_unit) → float`
SealQA	`web_search`	`(query:str, max_results:int) → list[result]`
	`page_fetcher`	`(url:str) → page_content`
	`dedup_ranker`	`(results:list) → ranked_list`
WildClawBench	`slack_api`	`(action, params) → response`
	`gmail_api`	`(action, params) → response`
	`calendar_api`	`(action, params) → response`
	`web_search`	`(query, max_results) → list[result]`
	`fs_ops`	`(op, path, content?) → result`
	`code_sandbox`	`(code:str, lang:str) → exec_result`
	`img_processor`	`(image, operation, params) → image`
	`video_processor`	`(video, operation, params) → video`
	`git_cli`	`(command:str) → stdout`

D.4 Scoring Functions

OfficeQA. We use the OfficeQA fuzzy-matching scorer at five numerical tolerance levels $\mathcal{Q} = \{0\%, 0.1\%, 1\%, 5\%, 10\%\}$. Let $m_\tau(x) \in \{0, 1\}$ indicate whether the answer passes the scorer at tolerance τ. For evolution, the task score is the fraction of tolerance checks passed, $P(x) = \frac{1}{5}\sum_{\tau\in\mathcal{Q}} m_\tau(x)$. Thus, an exact match scores 1.0; answers whose tightest passing tolerance is 0.1%, 1%, 5%, or 10% score 0.8, 0.6, 0.4, or 0.2, respectively; answers that fail all checks score 0. With $\theta = 0.8$, a sample enters the failure set when it fails the 0.1% tolerance check. Text-only answers use the scorer's normalized text matching. All accuracy numbers in the main paper use the 0% tolerance setting unless otherwise stated.

SealQA. We use the official LLM-as-judge protocol: a separate judge model (Qwen3.5-122B, frozen) receives the question, ground-truth answer, and the agent's response, and outputs a binary correct/incorrect judgment. To reduce variance, each response is judged three times and the majority vote is taken. The task score $P(x) \in \{0, 1\}$ is the majority judgment.

WildClawBench. We follow SkillClaw's multi-indicator aggregation: each task is evaluated against a rubric of 3–5 binary checkpoints (e.g., "correct recipient", "attachment present", "deadline extracted"). The task score $P(x)$ is the fraction of checkpoints passed, with a *critical-error override*: if any checkpoint tagged as critical fails, $P(x) = 0$ regardless of other checkpoints. This hard constraint makes WCB particularly sensitive to tool-level reliability, since a single tool failure on a critical step zeroes the entire score.

D.5 Scaling Factor Definitions

The five task descriptors in the radar chart of Figure 4 (right panel) are computed per-task on WildClawBench at the 397B scale. **Tool Complexity** is the number of distinct tools invoked in the ground-truth solution trajectory. **Interaction Density** is the number of skill pairs co-activated during execution, normalized by the total number of possible pairs. **Context Size** is the total token count of all skill and tool descriptions injected into the agent's context window for that task. **Action Space** is the number of distinct action types (tool calls, control-flow decisions, output formatting steps) available to the agent at each step, averaged across the trajectory. **Skill Synergy** is the mean estimated pairwise interaction effect among co-activated skills for that task, measuring how much the activated skill set benefits from joint usage. For each dimension, tasks are binned into terciles (low/medium/high), and the accuracy gain of SkillSmith over the skill-only baseline is averaged within each bin; the radar chart plots the high-tercile gain for each factor.

D.6 Resilience Experiment Setup

The 100-iteration resilience experiment in Figure 5 is conducted on a combined task pool drawn from all three benchmarks using Qwen3.5-122B. The system starts from the standard initial state Σ_0 and evolves continuously; $-$Ecosys and $-$Nogood denote the module removals in Appendix D.2. Two controlled perturbations are introduced to test recovery: at iteration 30, we inject 15 new tasks with higher step counts and more complex tool-chain dependencies than the original distribution, simulating a difficulty escalation in a production deployment; at iteration 60, we randomly corrupt the implementations of 3 out of the current tools in $\mathcal{T}_t$ (replacing function bodies with no-op stubs), simulating tool-layer damage such as an API deprecation or a dependency breakage. All methods share identical perturbation schedules. Library size in Figure 5b counts the total number of active skills plus active tools ($|\mathcal{S}_t| + |\mathcal{T}_t|$) at each iteration.

E Empirical Evolution Cost

Table 5 reports actual costs measured on OfficeQA at the 122B scale. We report backbone model calls and the total number of input and output tokens.

Table 5: Cost comparison on OfficeQA (Qwen3.5-122B).

Method	Calls/iter	Iters	Total calls	Total tokens (M)	Accuracy
EvoSkill	42.6	18	766.8	18.4	63.9%
SkillSmith	51.3	13	666.9	16.0	72.2%

SkillSmith uses 51.3 model calls per iteration over 13 iterations, compared with EvoSkill's 42.6 calls over 18 iterations. This corresponds to approximately 13% fewer total calls (667 vs. 767) and 13% fewer total tokens (16.0M vs. 18.4M), with an accuracy gain of 8.3 percentage points.

F Related Work

Skill libraries and reflective evolution. Trace2Skill consolidates execution experience into transferable skills (Ni et al., 2026); SkillX builds a hierarchical skill knowledge base with feedback-driven refinement and exploratory expansion (Wang et al., 2026). EvoSkill combines failure analysis with Pareto selection to discover and revise skill folders under a frozen model (Alzubi et al., 2026), while SkillClaw aggregates experience across users and time (Ma et al., 2026). GEPA uses trajectory-level reflection to propose prompt updates and combine complementary candidates (Agrawal et al., 2025). These approaches establish external artifacts as a practical target for adaptation. SkillSmith studies how such artifacts share a finite execution window: randomized masking estimates skill effects and interactions, and ecosystem dynamics allocate context according to marginal utility per token. SkillSmith couples library allocation and modification under both budgets.

Executable skills and reusable abstractions. Toolformer learns API invocation through self-supervised training, including when to call tools and how to use their outputs (Schick et al., 2023). Voyager maintains a library of executable skills and improves programs using environment feedback and execution errors (Wang et al., 2023). DreamCoder learns reusable program components by identifying common structure across solutions, jointly improving its library and search policy (Ellis et al., 2021). Thus, executable skill reuse and abstraction discovery already have substantial precedents. SkillSmith connects abstraction extraction to context allocation: moving shared, testable logic behind a tool interface lowers the context cost of its callers. Its four typed operations specify both the tool change and the required caller updates, allowing interface changes to be validated and committed with their dependents. The capacity result formalizes the benefit of lower costs within the paper's allocation model.

Context management and resource allocation. MemGPT manages a hierarchy of memory tiers, moving information between external storage and a bounded context window (Packer et al., 2023). This makes the distinction between persistent storage and active context explicit: a large external memory need not be loaded in full. SkillSmith applies this distinction to a library of executable capabilities and workflows. Its allocation variables govern which skills enter the window, while pairwise effects capture how their usefulness depends on co-activation. Tool extraction changes the cost of the assets being allocated. Memory management and abstraction extraction are therefore complementary ways of supporting a larger persistent library within a fixed execution window.

Evolutionary dynamics and online learning. Replicator dynamics provides a classical model of frequency-dependent selection (Hofbauer & Sigmund, 1998); multiplicative-weight updates allocate resources among alternatives in online learning (Freund & Schapire, 1997). SkillSmith uses these foundations to evolve reserved context shares, with fitness defined by the marginal utility of additional tokens. Bayesian estimation supplies uncertainty-aware fitness estimates from randomized skill masking. This interpretation connects competition between skills to a measurable resource constraint. The continuous-time ascent analysis holds for a fixed utility model; sampled updates and evolving libraries introduce estimation and model changes. Accordingly, the theory characterizes the allocation mechanism under stated assumptions, rather than guaranteeing monotonic improvement throughout adaptive search.

Failure memory and conflict learning. Reflexion stores verbal feedback in episodic memory to guide subsequent trials without updating model weights (Shinn et al., 2023). In software debugging, delta debugging reduces a failure-inducing input to a 1-minimal subset under its testing assumptions (Zeller & Hildebrandt, 2002). In satisfiability solving, GRASP records conflict causes to preempt repeated conflicts (Marques-Silva & Sakallah, 1999). SkillSmith combines these ideas at the level of proposed edits: replay tests isolate a failing modification subset, and version-scoped nogoods constrain later proposals. This connects retrospective feedback to prospective search pruning. Unlike clauses derived from a fixed logical formula, these records depend on component versions and empirical replay; exact-match veto therefore requires the stated persistence assumption, while similarity-based matching remains heuristic. The shared iteration budget charges both validation and failure minimization, making the cost of learning a reusable constraint explicit throughout the evolution of skills, tools, and their dependencies.