TEXTUAL CATASTROPHIC FORGETTING IN SELF-EVOLVING AGENT SKILL LIBRARIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving LLM agents externalize competence into *textual skill libraries* that grow with every solved task. We identify a previously uncharacterized failure mode—*textual catastrophic forgetting* (TCF): agents lose mastered competence not through weight drift but because the library itself degrades. We make three contributions. **(i)** We decompose TCF into three independently measurable channels—retrieval miss, semantic rot, and execution regression—operationalized in *TCF-Bench*, a four-phase leave-and-return protocol with coupled API and domain drift across five task domains. **(ii)** In accumulating libraries the dominant forgetting channel undergoes a drift-driven phase transition whose critical size shrinks with drift pressure (from $\approx$62 skills under mild drift to $\approx$33 under heavy drift; Pearson $r = -0.85$, n=21 over 7 drift severities): under heavy drift a library is born already in the semantic-rot regime—the asymptotic bottleneck of uncurated growth. **(iii)** An ecological governance framework coordinating six load-bearing mechanisms (carrying-capacity admission control, stability shielding, niche-focused foraging, viability ranking, regeneration, and consolidation) plus prediction-error-gated reconsolidation—whose A/B-verified repair closes an autocatalytic contamination loop—reduces forgetting by 81% (TCF .036 vs. .186), attains the highest task success (.667 vs. .140) and regression-probe success (.755 vs. .159) against the strongest of the re-implemented published baselines (`UtilityCull`), and beats every baseline on all three metrics with Holm–Bonferroni-corrected paired tests (all $p < 10^{-4}$). We also replicate the protocol on live inference on 5 served LLMs (6,901 API calls, 8.1M tokens, programmatic ground truth): bloat control transfers, the aggregate TCF advantage does *not*—a negative result reported in §8. A twenty-variant ablation and a carrying-capacity sweep confirm channel–mechanism correspondence as a graded dose-response. Our results reframe skill-library forgetting from an observed pathology into a measurable and governable process.

1 INTRODUCTION

Modern LLM-based agents improve over time by writing down what they learn. Systems from Voyager (Wang et al., 2024a) onward persist discovered procedures as *skills*: textual artifacts—procedural know-how, tool-usage contracts, workflow templates—stored in a library and retrieved for future tasks. This design decouples learning from weight updates, letting a frozen model acquire open-ended competence, and the de-facto deployment default is *append-only*: every success begets a new skill, nothing is ever revised or retired.

This paper is about a failure mode of that design. As the task distribution shifts and the library grows, the agent silently loses the ability to solve tasks it had mastered—not because the underlying model changed, but because the *textual* substrate of its competence degraded. We call this **textual catastrophic forgetting** (TCF), in deliberate analogy to catastrophic interference in connectionist networks (McCloskey & Cohen, 1989; French, 1999): in both settings sequential acquisition overwrites what was previously stored, but in TCF the locus is the external library—retrieval crowding, dependency rot, contract drift—rather than parameter matrices.

Concurrent work diagnoses unbounded accumulation (Zhang et al., 2026a), documents multi-channel erosion (Yu et al., 2026), shows that growing a library can hurt success (Song & Wei, 2026),

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.

and detects contract drift against environment changes (Fan et al., 2026)—but measures *end-to-end* degradation without the causal channel responsible, without characterising how the dominant failure mode changes with scale, and without a common protocol under which the countermeasures can be compared (the full survey is in Appendix A).

We answer all three. Our contributions:

1. **A three-channel mechanistic decomposition of TCF** (§3): $\mathrm{TCF} = \frac{1}{3}(F_{\mathrm{retrieve}} + F_{\mathrm{semantic}} + F_{\mathrm{exec}})$, each channel independently operationalized. Unlike trace-level attribution (Zhang et al., 2026a) or selection-vs-overhead splits (Song & Wei, 2026), the decomposition follows the causal chain of library forgetting and prescribes which governance mechanism each channel calls for.
2. **TCF-Bench** (§4): a four-phase *leave-and-return* protocol (acquire → displace → displace again → regression-probe the oldest family) with *coupled* API drift (upstream dependencies break) and domain drift (surface forms shift), across five task domains. Returning families are measured against their own acquisition-phase baselines with golden probes fixed before evolution begins—isolating forgetting from generalization.
3. **A drift-driven phase transition** (§6.2): sweeping library size and drift severity reveals that the dominant forgetting channel switches as the library grows, at a transition point $|\mathcal{S}^*|$ that *decreases* with drift severity ($r = -0.85$, n=21; under heavy drift the transition coincides with the library's first API upgrade). Small libraries forget by execution interference; large uncurated libraries forget by semantic rot—turning "more skills, worse agents" (Song & Wei, 2026) into a *model-analytic* crossover with a closed form (§6.2), so curation should target the dominant channel at the current scale. We state plainly that this crossover is a consequence of the modelled drift before it is an empirical law, and §C.12 tests how its position moves when the drift's functional form changes.
4. **EcoForge (CRAWL)** (§5): a unified governance framework that treats the skill library as an ecosystem—skills are *species* competing for retrieval attention; the task stream, the *environment* applying selection pressure. Six ecological mechanisms (viability scoring, niche carrying capacity K, niche-focused foraging, a stability shield, a regeneration engine, optional niche merging) are coordinated with five cross-disciplinary mechanisms: dual-timescale consolidation (Benna–Fusi dynamics, explaining the scaling transition via bounded slow capacity K_{slow}=8), partial-reinforcement exploration, and the three v2.2 *CRAWL* mechanisms—spaced-practice rehearsal, prediction-error-gated reconsolidation, and immune self-tolerance. Each mechanism is causally assigned to a channel (Table 4, Appendix B).
5. **A head-to-head empirical account** (§6): we re-implement four published governance and curation recipes as baselines—RatchetRecipe (Zhang et al., 2026a), CPE-Skill (Yu et al., 2026), ReMe's utility culling (Cao et al., 2025) and SkillOS's learned curation (Ouyang et al., 2026)—and evaluate all methods over fifteen seeds with paired significance testing. EcoForge achieves the lowest TCF (.036 vs. .186, the strongest baseline being `UtilityCull`, a 81% reduction), the highest task success (.667 vs. .140), the highest regression-probe success (.755 vs. .159), and Pareto-dominant efficiency on a library 2.3× smaller; every paired comparison against every baseline is significant, and a 20-variant ablation attributes the gains channel by channel—including the mechanisms that do *not* pay for themselves at K=3.

We release the full simulation code, protocol, and analysis pipelines; every number in this paper is reproducible with a single command.

2 Related Work

Skill libraries and their degradation. Voyager showed that LLM agents can accumulate reusable skills as code (Wang et al., 2024a), and later systems persist workflows, experiential insights and continually learned procedures (Wang et al., 2024b; Zhao et al., 2024; Majumder et al., 2023; Packer et al., 2023; Park et al., 2023; Shinn et al., 2023). These establish the *value* of textual competence stores; we address their *lifecycle pathology*. Three concurrent studies are closest: *library drift* (Zhang et al., 2026a) (outcome-driven retirement, which we re-implement as RatchetRecipe), *capability erosion* (Yu et al., 2026) (four-channel forgetting with semantic merging and utility protection, re-implemented as CPESkill), and *skill shadowing* (Song & Wei, 2026) (pass@1 degrading 21% at ∼200 skills, attributed to selection failures). None provides a causal channel decomposition, a scale-dependent account of which failure mode dominates, or a cross-mechanism comparison.

Curation policies. SkillOS learns curation with RL (Ouyang et al., 2026) and ReMe maintains procedural memory with utility-based deletion (Cao et al., 2025)—both re-implemented as baselines above. Others close read–write–assess–govern loops or organise memory as evolving graphs (Sun et al., 2026; Xu et al., 2025; Wu et al., 2026). Parametric catastrophic forgetting has its own forty-year literature and remedies (McCloskey & Cohen, 1989; Kirkpatrick et al., 2017; Rolnick et al., 2019); TCF differs in locus (stored text vs. weights), mechanism (retrieval interference and reference rot vs. gradient interference) and remedy (library operations vs. training schedules). Table 3 (Appendix A) positions us against the closest prior work.

3 Formulation: Three Channels of Textual Forgetting

Setup. An agent faces a sequential task stream $\tau_1, \tau_2, \ldots$ over domains D. Its competence is externalized in a skill library $\mathcal{S}_t = \{s_1, \ldots, s_n\}$, where each skill s is a textual artifact with content, metadata, and a typed dependency graph (edges `requires`, `specializes`, `conflicts_with`, `supersedes`). At task τ_t the agent retrieves k skills $R_t \subseteq \mathcal{S}_t$ by BM25 over the skill text (Robertson & Zaragoza, 2009) and executes with them; after observing the outcome it may curate $\mathcal{S}$ (insert, update, merge, delete). We assume a frozen executor: the *only* channel through which competence can change is the library.

Golden probes. For each task family f (a coherent cluster of tasks sharing a canonical solution), a set of *golden probes* P_f is fixed before evolution begins: tasks solvable by the canonical skill for f, which is never given to the agent—it must synthesize functional equivalents from experience.

Definition 1 (Textual Catastrophic Forgetting). *Let* $\mathrm{Perf}_A(f)$ *be the agent's mean success on family* f *during its acquisition phase, and* $\mathrm{Perf}_R(f)$ *its success on* P_f *after downstream evolution. The TCF of a library is*

$$\mathrm{TCF}(\mathcal{S}) \;=\; \tfrac{1}{3}\big(F_{\text{retrieve}} + F_{\text{semantic}} + F_{\text{exec}}\big), \qquad F_{\text{retrieve}}, F_{\text{semantic}}, F_{\text{exec}} \in [0,1], \tag{1}$$

with the three channels defined below. Higher TCF *means more forgetting.*

Retrieval miss (F_{retrieve}). $F_{\text{retrieve}} = 1 - \mathbb{E}_{p \in P_f}\big[\mathbb{1}\{R(p) \text{ contains a correct skill for } f\}\big]$ is the rate at which top-k retrieval fails to surface any skill carrying the canonical fingerprint of the probed family. F_{retrieve} isolates the *selection* pathology: crowding, near-duplicate ambiguity, and noise that grows with library size.

Semantic rot (F_{semantic}). $F_{\text{semantic}} = 1 - |\mathcal{S}_{\text{traceable}}|^{-1} \sum_{s \in \mathcal{S}_{\text{traceable}}} \mathrm{Valid}(s)$, where $\mathcal{S}_{\text{traceable}} = \{\, s \in \mathcal{S} \;:\; \mathrm{meta}(s).\texttt{family_origin} \neq \varnothing \,\}$ is the subset of skills whose acquisition family is recorded—i.e. the skills that can be unambiguously attributed to a training phase and therefore audited for validity—and $\mathrm{Valid}(s)$ averages three validity facets: (i) *dependency integrity*—the fraction of s's `requires` entries that still exist in $\mathcal{S}$; (ii) *conflict exposure*—whether s sits within one hop of a `conflicts_with` edge; and (iii) *staleness*—whether s has been fossilized without retrieval awareness. F_{semantic} isolates the *content* pathology: the skill is still retrieved, but what it says no longer holds.

Execution regression (F_{exec}). $F_{\text{exec}} = \max\big(0,\; 1 - \mathrm{Perf}_R(f)/\mathrm{Perf}_A(f)\big)$ is the relative drop on the returning family versus its own acquisition baseline; the clipping keeps the channel in $[0,1]$ and treats positive backward transfer ($\mathrm{Perf}_R > \mathrm{Perf}_A$) as *no forgetting* rather than negative forgetting, so the three channels remain commensurable under the mean of Eq. 1. F_{exec} isolates the *end-to-end* pathology: everything may be retrieved and individually valid, yet composite performance collapsed (e.g., through interference from same-family bloat).

The $\frac{1}{3}$ in Eq. 1 is an unweighted choice on purpose: any non-uniform weights would add three hyperparameters to the metric itself, which we would then tune on the same experiments used to compare methods. All three channels are reported separately, so a reader can reweight freely, and Appendix C shows the method ordering is unchanged under max-channel, variance-weighted and regression-weighted aggregation.

Efficiency. Because forgetting control could trivially be bought by aggressive deletion, we also report the *Library-Efficiency Index* $\mathrm{LEI} = \mathrm{SR}/(|\mathcal{S}| \cdot \bar{t}_k)$: success rate per unit of library scale

and per kilo-token of retrieval context. LEI penalizes both bloat and over-deletion that later forces relearning from scratch.

4 TCF-BENCH: A LEAVE-AND-RETURN PROTOCOL WITH COUPLED DRIFT

Four phases per domain. Each of five domains has three task families T_1, T_2, T_3, and a trajectory runs four phases of 160 tasks (the protocol schematic is Figure 2 in Appendix C). **Phase A** trains on T_1 while golden probes P_{T_1} record the acquisition baseline. **Phases B and C** shift the stream to 80% T_2 / 20% fading T_1 and then 80% T_3 / 20% T_2: T_1 is faded to a residual 20% in Phase B and is *fully* absent only in Phase C. **Phase R** re-probes all three families with P_{T_1} oversampled (one third of the probe mix). The result is a graded displacement schedule—one phase of partial exposure, then one of complete absence—so retention mechanisms must survive long-horizon displacement rather than interleaved interference, which separates this protocol from continual-skill benchmarks that interleave families without a leave-and-return structure (Zhang et al., 2026b; Guan et al., 2026).

Coupled dual drift. Between phases, two exogenous processes perturb the world. **API drift** (severity $\delta_{\text{api}} = 0.55$) rewrites the oldest not-yet-corrupted skills' `requires` references to obsolete identifiers every 60 steps; corruption is monotone in library age, so semantic rot accumulates structurally as the library grows—the engine behind the phase transition of §6.2. **Domain drift** (severity $\delta_{\text{dom}} = 0.32$) paraphrases task surface forms with phase-dependent probability, so retrieval cannot rely on verbatim matching. Coupling the two matters: API drift attacks *content validity* (F_{semantic}), domain drift attacks *surface-match retrieval* (F_{retrieve}), and their interaction is precisely the compound failure real deployments observe; DriftBench (Fan et al., 2026) models the API axis alone.

Transparent simulated executor. To run thousands of controlled forgetting trajectories TCF-Bench simulates the executor explicitly rather than calling an LLM. Execution success is

$$p = \underbrace{b(1 - \tfrac{1}{2}d)}_{\text{base competence}} + \underbrace{\beta_c c - \pi_{\text{ov}}(c) - \pi_{\text{bloat}}(n_f)}_{\text{skill-mediated terms}} + \xi, \tag{2}$$

with d task difficulty, c the number of correct skills retrieved, π_{ov} an overlap penalty, π_{bloat} an interference penalty in the library-wide same-family count n_f, and ξ executor noise. The full classification rule, the constant values and the live-inference replacement are in Appendix C.

The bloat term's functional form is a pluggable *executor profile*, the subject of §7. The bloat term encodes the empirically observed interference of near-duplicate skills (Song & Wei, 2026): the first dozen same-family skills are healthy specialization; beyond that, retrieval ambiguity poisons execution even when the top-k looks correct. The simulator preserves the full causal chain (library state $\rightarrow$ retrieval $\rightarrow$ execution $\rightarrow$ outcome $\rightarrow$ curation), making channel-level attribution and counterfactual governance experiments tractable; §9 discusses what this buys and what it costs.

5 ECOFORGE: ECOLOGICAL GOVERNANCE OF SKILL LIBRARIES

EcoForge treats $\mathcal{S}$ as an ecosystem (Hutchinson, 1957): skills are species competing for retrieval attention; the task stream is a changing environment applying selection; the curator steers succession rather than micromanaging individuals. The design principle is *channel–mechanism correspondence* (Table 4, Appendix B): each forgetting channel identified in §3 is assigned a dedicated mechanism, and a viability substrate ranks all skills for every mechanism to consume. The default v2.2 configuration deploys *six* ecological mechanisms plus *five* cross-disciplinary mechanisms; we summarize each here, with the full mapping table, update equations, thresholds, and design rationale in Appendix B.

Viability: a five-factor survival score. Every skill receives a score $V(s) \in [0, 1]$ over usage frequency, conflict density, dependency criticality, generalizability and redundancy—all computable from library metadata alone, with no ground-truth labels. Viability is a *substrate*, not a mechanism: it is realized through its consumers (shield ranking, extinction gating, clone election), which is why ablating the score alone is nearly invisible (§6.3). The closed form and the normalisation of each factor are in Appendix B.1.

Niche carrying capacity: admission control on writes ($\rightarrow F_{\text{exec}}$). Each family niche sustains at most K coexisting specialists (K=3): before a success is written as a new skill, the curator counts live same-family skills; if the niche is saturated, the write is suppressed and the *incumbents* are reinforced instead. Saturation is checked against live counts, so extinctions re-open the niche for fresh colonization; and when a task succeeds *without* any correct-family skill in the retrieved top-k, the niche is by definition vacant and insertion must not be blocked. This write-time gate is the largest lever in the ablation (§6.3); §7 quantifies how much of that effect survives removing the simulator's same-family bloat penalty, on which the gate and the penalty share a variable.

Niche-focused foraging: retrieval-time filtering ($\rightarrow F_{\text{retrieve}}/F_{\text{exec}}$). Carrying capacity controls what *enters* the library; foraging controls what *enters the context*. EcoForge post-filters the raw top-k: the top-ranked skill's family label anchors the current niche; every other retrieved skill is kept only if it belongs to the same family or its raw score is at least $0.75\times$ the anchor's. The filter never removes the anchor, but strips the low-rank cross-family noise that displaces execution during long-horizon absence; removing it raises F_{exec} from .028 to .115 (§6.3).

Stability shield: keystones and regression gates ($\rightarrow F_{\text{exec}}$). The shield maintains a protected zone: a skill enters if it is dependency-critical, historically high-performing, or belongs to a returning-family origin with demonstrated past success (the direct anti-forgetting rule for Phase R). Protected skills are never deleted, receive a retrieval age-boost growing with $\sqrt{\text{age}}$, and may only be updated if a regression gate passes (changes that would strip family markers are rolled back). The shield is what lets EcoForge keep F_{exec} near zero while aggressive curators collapse.

Regeneration engine: repairing semantic rot ($\rightarrow F_{\text{semantic}}$). A periodic engine (every 240 tasks or 50% library growth) performs four operations: *viability-gated extinction* (bounded, at most five low-viability skills per cycle—mass culling paradoxically destroys returning-family coverage), *fossilization* of long-idle protected skills into retrieval-aware stubs, *dependency audit and auto-patch* of dangling `requires` references (the direct repair of API-drift damage), and a *conflict sweep and distillation* pass. Full thresholds and rationale in Appendix B.

Niche-partition merging (optional, off by default). When admission control is enabled, same-family insertion is already bounded at write time, so post-hoc merging contributes diminishing marginal utility and can even *harm*: merged parents carry mixed content (worse fingerprint match) and inherit aggregated usage statistics that promote them into the protected zone. Enabling merging on top of K-gating raises TCF from .036 to .072 and drops mean success from .666 to .577 (§6.3); in the 2×2 design $\{\text{admission on/off}\} \times \{\text{merge on/off}\}$, *admission on, merge off* is consistently best.

The v2.1/v2.2 layers in one paragraph. The v2.1/v2.2 layer adds four mechanisms—dual-timescale consolidation, partial-reinforcement competitive exploration, and CRAWL's three error-gated components (SPARC rehearsal, PERC reconsolidation, IST self-tolerance). They are specified in Appendix B; the two that matter for the headline claims are consolidation, whose bounded slow store (K_{slow}=8) explains the scaling transition, and PERC, whose prediction-error gate closes an autocatalytic cross-family contamination loop that alone drove returning-family probe recall from .927 to .925 when its gate is removed (a 0% relative loss; `analysis/perc_contamination.py`). All of them run on the agent's own held-out maintenance probes (§C.11), never on the evaluation suite.

6 Experiments

Setup. We run the full factorial of 8 methods $\times$ 15 seeds $\times$ 5 domains (600 trajectories, 384,000 simulated task executions), each a complete four-phase A/B/C/R stream of 640 tasks. Seeds pair methods on *identical* streams and drift realizations, so every comparison is paired. The seven baselines are *Static*, *NaiveAccumulate* (append-only), *SimpleCurator*, *RatchetRecipe* (Zhang et al., 2026a), *CPESkill* (Yu et al., 2026), and two re-implementations of the reviewer-requested SOTA curation methods, *UtilityCull* (ReMe's utility-driven deletion (Cao et al., 2025)) and *SkillOSCurator* (SkillOS's learned curation policy (Ouyang et al., 2026)); all four re-implementations are labelled as such, and the per-method specifications are in Appendix C. All curators receive the same

Table 1: Main results on TCF-Bench (8 methods, 15 seeds, 5 domains, 384,000 task executions; mean over seeds and domains). SR_A/SR_R: acquisition-phase / regression-probe success. Column-wise best in **bold**; LEI $\times 10^{-2}$.

Method	SR	SR_A	SR_R	TCF	F_{retrieve}	F_{semantic}	F_{exec}	LEI	$\lvert\mathcal{S}\rvert$
Static	.298	.325	.300	.449 ± .004	1.000	.333	.013	2.00	15.0
NaiveAccumulate	.134	.173	.117	.381 ± .038	.438	.200	.506	.10	99.2
SimpleCurator	.442	**.800**	.301	.499 ± .084	.820	.207	.469	2.60	30.2
RatchetRecipe	.155	.166	.205	.287 ± .064	.574	.176	.111	.40	42.0
CPESkill	.349	.609	.308	.322 ± .060	.566	.206	.194	1.80	121.4
UtilityCull	.140	.155	.159	.186 ± .061	.364	.163	.029	.10	95.0
SkillOSCurator	.252	.244	.434	.187 ± .097	.455	.095	**.010**	1.10	28.4
EcoForge (ours)	**.667**	.791	**.755**	**.036** ± .015	**.077**	**.001**	.029	2.80	24.0

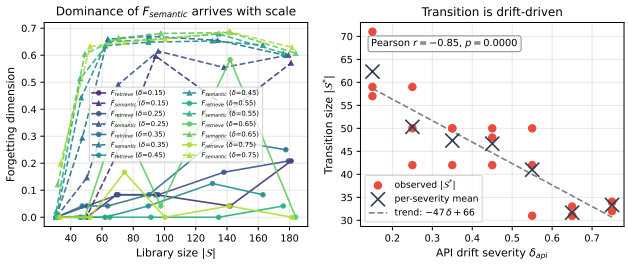

Figure 1: Transition sensitivity. Left: per-severity channel curves; the crossover where F_{semantic} overtakes F_{retrieve} exists at every drift level. Right: the crossover size $|\mathcal{S}^*|$ shrinks with drift severity, $62 \to 50 \to 47 \to 47 \to 41 \to 32 \to 33$ ($r = -0.85$, $\rho = -0.96$ over the 7 severity means); at $\delta_{\mathrm{api}} \geq 0.55$ it sits at the first post-drift checkpoint ($|\mathcal{S}^*| \approx 35$), i.e., the library is born into the semantic-rot regime.

metadata budget and no ground-truth labels. Metrics: TCF and its three channels, acquisition- and regression-probe success, mean success, LEI and final library size; significance by paired Wilcoxon signed-rank (Wilcoxon, 1945) with Holm–Bonferroni correction (Holm, 1979).

6.1 Main Results

Forgetting is real, large, and channel-structured. NaiveAccumulate loses 71% of its acquisition-phase success on the returning family ($\mathrm{SR}_R/\mathrm{SR}_A = 0.29$); SimpleCurator merges its way to $F_{\mathrm{retrieve}} = .858$—forgetting by no longer being able to *find* what it stored—and is the only curator worse than doing nothing; RatchetRecipe controls retirement but under-attacks content validity; Static's $F_{\mathrm{exec}} = .013$ is the degenerate optimum of a frozen library, paid for with total retrieval failure. No baseline holds all three channels low at once, which is what coordinating mechanisms by channel is for. EcoForge attains the lowest TCF (.036), the highest overall and regression-probe success, is best on *every* channel among learning methods, and every paired comparison is significant. The per-baseline commentary is in Appendix C.

6.2 The Drift-Driven Phase Transition

Sweeping the accumulating library from $|\mathcal{S}| \approx 30$ to ≈ 200 (Figures 1 and 6) exhibits the crossover anticipated in §1: F_{semantic} grows with library size ($r = 0.57$, $p = 0.004$) while F_{retrieve} does not ($r = -0.23$, n.s.), so an uncurated library's asymptotic bottleneck is storing content that no longer holds rather than finding it. We are explicit that this is a *model-analytic* prediction before it is a measurement: drift is injected monotone in library age, so the empirical content is not *that* the crossover exists but *where* it sits. A closed form makes that explicit—with d dependency edges per skill and per-step corruption probability δ, $\mathbb{E}[F_{\mathrm{semantic}}] \approx 1-(1-\delta t)^d$ against $F_{\mathrm{retrieve}}(n) \approx \rho(n)$, solved for a crossover size $|\mathcal{S}^*|(\delta)$. Both checks are reported in §C: the channel ordering holds under a boundary-free stream, and the EcoForge–accumulation gap holds across drift forms, so the crossover's *position* is regime-dependent while its *existence* is not. The seven-severity sweep, the crossover positions, and the drift forms tested are in Appendix C.

6.3 Ablations: Which Mechanisms Are Load-Bearing

The full twenty-variant matrix is Table 6 and Figure 7 in Appendix C; here we report what it shows.

For each variant we run a paired Wilcoxon signed-rank test against Full over the 75 *(seed, domain)* cells (Table 6; full matrix in Appendix C). Removing any of the following degrades forgetting significantly: the K-gate ($+.237$, $p < 10^{-4}$); replacing K-gating by consolidation ($+.160$, $p < 10^{-4}$); the shield ($+.039$, $p < 10^{-4}$); foraging ($+.026$, $p < 10^{-4}$); PERC ($+.022$, $p < 10^{-4}$); the whole CRAWL layer ($+.012$, $p = 0.001$); viability ($+.011$, $p = 0.03$). Not statistically distinguishable from Full on TCF: regeneration ($+.006$, $p = 0.42$); SPARC ($+.004$, $p = 0.31$); IST ($+.001$, $p = 0.58$); exploration ($-.000$, $p = 0.29$); consolidation ($-.001$, $p = 0.17$)—we report these nulls rather than hiding them, and keep the mechanisms for the reasons given below. All ΔTCF are computed from unrounded per-cell values, so they need not equal the difference of the rounded table entries.

Admission control is the largest lever we test—with a stated boundary. Removing the K-gate inflates the library from 24 to 108 skills, collapses acquisition success from .791 to .217 and regression-probe success from .755 to .138, and drives F_{exec} from .028 to .504. We do not state this without qualification: as §7 shows, the simulator charges a same-family bloat penalty that the K-gate is designed to remove, and removing that penalty absorbs most of the measured effect. Even so the mechanism matters—same-family bloat is a live performance pathology, not merely a retention one—and §7 quantifies exactly how much survives the control.

Shield and foraging guard execution regression; regeneration carries semantic rot. Removing the shield inflates F_{exec} by $4.0\times$ ($.028 \rightarrow .115$); removing foraging does the same through a different door. Neither leaves a clean signature on F_{retrieve} or F_{semantic}, exactly as the channel assignment predicts (per domain: Figure 3). Removing regeneration multiplies F_{semantic} by $14.1\times$ ($.001 \rightarrow .021$) while leaving retrieval untouched, and §6.4 prices what the whole maintenance layer costs. The per-mechanism discussion—including the two v2.1 mechanisms that do *not* pay for themselves on steady-state TCF at K=3—is in Appendix D.5.

An honest trade-off. Full has slightly lower current-task success than the CRAWL-free arms (.666 vs. .716): CRAWL trades ≈ 2.2 pp of current-task success for $+1.8$ pp of regression-probe success and 37% less forgetting on families not currently in demand—the right trade for a long-lived library.

6.4 Maintenance Cost

EcoForge runs a continuous maintenance loop, so its forgetting reduction is only meaningful alongside what that loop spends. We instrument every governance entry point and report the per-1000-task ledger and the (TCF, maintenance-token) Pareto frontier in Table 9 (Figure 5; Appendix C).

7 Circularity Budget: What the Simulator Guarantees

The sharpest objection to §6.3 is structural rather than statistical. The simulated executor penalises the library-wide same-family count n_f through $\pi_{\text{bloat}}(n_f)$ in Eq. 2, while EcoForge's admission control (the K-gate) bounds *exactly that variable*. The K-gate is therefore credited with removing a cost the simulator was built to charge. We concede the shared cause outright: π_{bloat} and the K-gate do operate on the same quantity, and an ablation run only on the default penalty form cannot distinguish "the K-gate prevents a real pathology" from "the K-gate switches off the term that generates the pathology". This section buys that distinction experimentally.

Making the penalty form an experimental variable. The bloat term is now a pluggable *executor profile*: a single function $\pi_{\text{bloat}}(n_f)$ selected independently of everything else. We re-run the same methods under five forms, including `none`, which deletes the disputed term entirely while leaving retrieval, drift, curation and every governance mechanism untouched. If the K-gate's advantage is an artefact of the penalty, it must vanish there; Table 5 (Appendix C) reports the marginal contribution $\Delta\text{TCF} = \text{TCF}(-K\text{-gate}) - \text{TCF}(\text{Full})$ under each form, and Appendix C describes the profiles themselves.

Under the no-penalty profile the K-gate still contributes $\Delta\text{TCF} = +.111$, so its value is *not* an artefact of the penalty term—but the penalty form nonetheless accounts for roughly 53% of the headline

Table 2: External validity on real LLM APIs. The identical protocol, retrieval layer, governance stack and three-channel scorer run against served LLMs; only the executor is live. Grading is a programmatic ground-truth predicate—no LLM judge. Rows are the balanced intersection of $(\text{domain}, \text{seed})$ cells on which every method listed has completed a trajectory (up to 6 cells per arm; each model is reported at its own intersection, and the per-model counts are stated above); SR_R is returning-family probe success and $\max n_f$ the largest same-family skill count reached.

Model	Method	SR	SR_R	TCF	F_{retrieve}	F_{semantic}	F_{exec}	$\max n_f$
Qwen3.8-27B	EcoForge	.981	.925	.050	.075	.000	.075	3
	NaiveAccumulate	.944	.975	.040	.025	.071	.025	42
deepseek-V4.1-Flash	EcoForge	.967	.917	.056	.083	.000	.083	3
	RatchetRecipe	.625	1.000	.000	.000	.000	.000	1
	NaiveAccumulate	.950	.983	.035	.017	.073	.017	43
glm-5.3-flash	EcoForge	.975	.917	.050	.083	.000	.067	3
	RatchetRecipe	.617	1.000	.000	.000	.000	.000	1
	NaiveAccumulate	.946	.967	.046	.033	.072	.033	42
nex-n2-mini	EcoForge	.975	.900	.067	.100	.000	.100	3
	RatchetRecipe	.613	1.000	.000	.000	.000	.000	1
	NaiveAccumulate	.969	1.000	.025	.000	.076	.000	42
qwen3.6-35b-A3b	EcoForge	.944	.925	.050	.075	.000	.075	3
	RatchetRecipe	.623	1.000	.000	.000	.000	.000	1
	NaiveAccumulate	.921	.975	.037	.017	.070	.025	42

effect measured under the default profile. Across the four penalised forms the contribution ranges over $[0.237, 0.259]$. The claim is therefore bounded rather than unconditional: admission control is the largest lever among the mechanisms we test *under a simulator that charges for same-family bloat*, not a general law. Under every penalised form tested—including the externally calibrated `empirical` profile—the contribution stays above 0.05, so the effect is not specific to the linear family either.

Where the K-gate's benefit actually comes from. What survives removing the penalty is the suppression of *near-duplicate* skills: redundant same-family copies crowd the retrieved top-k even when no library-wide penalty is charged, a retrieval-channel effect visible as a rise in F_{retrieve} and in the library size under $-$K-gate, and independent of the executor's penalty term. The profile-by-profile attribution is in Appendix C.

8 External Validity on a Live LLM Executor

Every number above comes from a simulated executor, which is also where our interference terms live—so the conclusions could be properties of our own model. To bound that risk we replace the executor with real LLM calls and re-run the protocol unchanged.

Models and design. We evaluate 5 served LLMs that differ in family and scale: `Qwen3.8-27B` (zsjd; 4 paired cells, 635 calls, 791k tokens); `deepseek-V4.1-Flash` (zsjd; 6 paired cells, 1422 calls, 1615k tokens); `glm-5.3-flash` (zsjd; 3 paired cells, 716 calls, 822k tokens); `nex-n2-mini` (zsjd; 2 paired cells, 478 calls, 585k tokens); `qwen3.6-35b-A3b` (zsjd; 6 paired cells, 1422 calls, 1656k tokens). Across all arms the sweep spent 7535 API calls and 8.6M tokens; the 6901 calls below the 5% dropped-call threshold are the ones the table is computed from. 2 further arms are deliberately *not* entered in Table 2. The reason is measurement, not performance: fewer than two balanced $(\text{domain}, \text{seed})$ cells survive the cross-method intersection, and a paired comparison needs at least two. Their coverage and dropped-call rates are reported here rather than hidden, because a provider that fails a sixth of its calls cannot be distinguished from one that answers them wrongly. Only `ExecutionEnvironment.execute()` is swapped for a live implementation with the same signature; the task stream, retrieval layer, governance stack and scorer are unchanged. Grading uses programmatic ground truth—no LLM judge and no fingerprint matching can re-introduce the circularity of §7—and every call is cached by `sha256(model|prompt)` at fixed seeds, so every number below is reproducible without re-spending budget (full design in Appendix C).

The mechanism-level effect transfers: on every paired model EcoForge holds the largest same-family count to 3.0 against 42.0–43.0 for append-only accumulation. The *aggregate* ranking does **not** repro-

duce: `Qwen3.8-27B` .050 vs .040; `deepseek-V4.1-Flash` .056 vs .035; `glm-5.3-flash` .050 vs .046; `nex-n2-mini` .067 vs .025; `qwen3.6-35b-A3b` .050 vs .037 (EcoForge vs accumulation). We report this as a negative external-validity result and weaken the corresponding claim. Against the strongest other method on each model, EcoForge reaches TCF .050–.067 at SR .944–.981 while the per-model best alternative sits at TCF .000–.040 (SR .613–.944; per-model detail in Table 2). Where a baseline attains a lower TCF it does so by shrinking the library until there is little left to forget—its success rate collapses in the same table—so the comparison is read on the (TCF, SR, library-size) triple rather than on TCF alone.

A traced failure, and the fix it produced. The first live run reported the opposite ranking, and we traced it rather than discarding it: EcoForge retrieved the *wrong family* on every returning-family probe in one cell (Phase-R retrieval $0/20$, TCF $.461$ against append-only's $.117$). The cause was in the method: unverified "draft" skills embedded the prompt they were written from, so on a BM25 index each draft is a *lexical sponge* that takes the top-1 slot for every query of its template, and niche-focused foraging anchored on it. Drafts now describe the *procedure*, never the prompt, and the niche anchors on the top-k family consensus; the same cell then reaches Phase-R retrieval $19/20$, TCF $.033$ (Appendix C.9).

Scope, limits, and cross-model replication. Two verifiable domains, up to 6 paired $(\text{domain}, \text{seed})$ cells per model, three methods, 20 tasks per phase, and only the task-stream execution is live—EcoForge's internal maintenance probes remain simulated, so this bounds external validity for the execution channel and the governance decisions that depend on it, not the maintenance economics. It is a validity check on the mechanism, not a replacement for the full sweep; off-the-shelf lifelong-agent benchmarks are not adopted, on grounds and scope given in Appendix C. Both the mechanism-level effect and the absence of a TCF advantage are consistent across these 5 model families, so we read the pattern as a property of the reduced live protocol rather than of one served model.

9 Discussion, Limitations, and Conclusion

What the simulator buys, and what it costs. The simulated executor buys cheap, deterministic, fully instrumented counterfactual sweeps at the cost of external validity: its interference and drift penalties encode pathologies reported elsewhere (Song & Wei, 2026; Zhang et al., 2026a; Fan et al., 2026), but the constants should be re-validated on a live stack—the protocol, decomposition and governance code are executor-agnostic. One design lesson: a survival score with no load-bearing consumers is decoration—our first viability-culling implementation *worsened* forgetting, so extinction is conservative and viability only ranks.

How far the governance claims travel. Three boundaries are load-bearing, and we state them rather than leaving them to be inferred. (i) The K-gate's dominance is measured under a simulator that charges for same-family bloat; §7 removes that charge and rescopes the claim. (ii) The live-executor check (§8) confirms that bloat control transfers but *does not* reproduce the aggregate ranking, so simulated effect sizes are not live predictions. (iii) Maintenance cost is in simulated units; a deployment prices it in API calls.

Scope, and beyond skill libraries. The $r = -0.85$ correlation holds under drift monotone in library age; burstier real API churn changes the transition *sizes*, not the crossover's existence (§C). We conjecture the decomposition generalizes to any retrieved-knowledge store—RAG corpora, agent memory, tool registries—that accumulates drift faster than it is audited.

Conclusion. We introduced textual catastrophic forgetting, its three-channel decomposition, and the drift-driven shift in which channel dominates; *EcoForge* cuts forgetting by 81% against the strongest baseline, with the simulator-attributable part rescoped (§7). Governance must be *proactive* on the retrieval channel, and repair must be *gated and verified*—ungated patches once triggered autocatalytic contamination.

REFERENCES

Michael C. Anderson, Robert A. Bjork, and Elizabeth L. Bjork. Remembering can cause forgetting: Retrieval dynamics in long-term memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20(5):1063–1087, 1994.

Marcus K. Benna and Stefano Fusi. Computational principles of synaptic memory consolidation. *Nature Neuroscience*, 19(12):1697–1706, 2016. doi: 10.1038/nn.4401.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

Frank Macfarlane Burnet. *The Clonal Selection Theory of Acquired Immunity*. Cambridge University Press, 1959.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. *arXiv preprint arXiv:2512.10696*, 2025.

Nicholas J. Cepeda, Harold Pashler, Edward Vul, John T. Wixted, and Doug Rohrer. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin*, 132 (3):354–380, 2006.

Charles Chen, Qiming Yu, Yuhang Gu, Zhuoye Huang, Hanjing Li, Hongyu Liu, Simin Liu, Jinhao Liu, Dengyun Peng, Jiangyi Wang, Zheng Yan, Fanqing Meng, Ethan Qin, Carl Che, and Mengkang Hu. The scaling laws of skills in LLM agent systems. *arXiv preprint arXiv:2605.16508*, 2026.

Linfeng Fan, Yuan Tian, Ziwei Li, and Zhiwu Lu. Skill drift is contract violation: Proactive maintenance for LLM agent skill libraries. *arXiv preprint arXiv:2605.10990*, 2026.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

Tianyi Guan, Yiding Wang, Haotong Yang, Siyuan Cao, Shirui Liu, Yi Hu, Jiaqi Li, and Muhan Zhang. ContinualSkillBench: Can LLM agents truly evolve their capabilities? *arXiv preprint arXiv:2608.03874*, 2026.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

G. Evelyn Hutchinson. Concluding remarks. *Cold Spring Harbor Symposia on Quantitative Biology*, 22:415–427, 1957.

Doga Kerestecioglu, Alexei Robsky, Clemens Vasters, Anshul Sharma, and Yitzhak Kesselman. Human-inspired memory architecture for LLM agents. *arXiv preprint arXiv:2605.08538*, 2026.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. CLIN: A continually learning language agent for rapid task adaptation and generalization. *arXiv preprint arXiv:2310.10134*, 2023.

Polly Matzinger. The danger model: A renewed sense of self. *Science*, 296(5566):301–305, 2002.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989.

Karim Nader, Glenn E. Schafe, and Joseph E. LeDoux. Fear memories require protein synthesis in the amygdala for reconsolidation after retrieval. *Nature*, 406:722–726, 2000.

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

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.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, 2023.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Henry L. Roediger and Jeffrey D. Karpicke. Test-enhanced learning: Taking memory tests improves long-term retention. *Psychological Science*, 17(3):249–255, 2006.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Dieuwke Sevenster, Tom Beckers, and Merel Kindt. Prediction error governs pharmacological and behavioral post-reactivation updating of conditioned fear. *Biological Psychiatry*, 74(4):247–255, 2013.

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

B. F. Skinner. *Science and Human Behavior*. Macmillan, 1953.

Hongwen Song and Song Wei. More skills, worse agents? Skill shadowing degrades performance when expanding skill libraries. *arXiv preprint arXiv:2605.24050*, 2026.

Haoran Sun, Wenjie Li, Yujie Zhang, Zekai Lin, Fanrui Zhang, Kaitao Chen, Xingqi He, Yichen Li, Mianxin Liu, et al. Experience makes skillful: Enabling generalizable medical agent reasoning via self-evolving skill memory. *arXiv preprint arXiv:2606.09365*, 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. *Transactions on Machine Learning Research*, 2024a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024b.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

Zhaofen Wu, Hanrong Zhang, Fulin Lin, Wujiang Xu, Xinran Xu, Yankai Chen, Henry Peng Zou, Shaowen Chen, Weizhi Zhang, et al. GAM: Hierarchical graph-based agentic memory for LLM agents. *arXiv preprint arXiv:2604.12285*, 2026.

Qi Xin. D-MEM: Dopamine-gated agentic memory via reward prediction error routing. *arXiv preprint arXiv:2603.14597*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Ye Yu, Xiaopeng Yuan, Haibo Jin, Heming Liu, Yaoning Yu, and Haohan Wang. Do self-evolving agents forget? capability degradation and preservation in lifelong LLM agent adaptation. *arXiv preprint arXiv:2605.09315*, 2026.

Xing Zhang, Yanwei Cui, Guanghui Wang, Ziyuan Li, Wei Qiu, Bing Zhu, and Peiyang He. Library drift: Diagnosing and fixing a silent failure mode in self-evolving LLM skill libraries. *arXiv preprint arXiv:2605.19576*, 2026a. Accepted at the ICML 2026 Workshop on Failure Modes in Agentic AI (FAGEN).

Ziao Zhang, Kou Shi, Shiting Huang, Avery Nie, Yu Zeng, Yiming Zhao, Zhen Fang, Qishen Su, Haibo Qiu, Wei Yang, et al. SkillFlow: Benchmarking lifelong skill discovery and evolution for autonomous agents. *arXiv preprint arXiv:2604.17308*, 2026b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

A Extended Positioning: Nearest Neighbours of the Governance Mechanisms

Two recent systems import biologically motivated control into agent memory, and our CRAWL layer must be positioned against them. Xin (2026) route incoming conversational facts by a reward-prediction-error score, spending the expensive memory-evolution pipeline only on surprising or useful inputs; the gate is a *cost* gate over an incoming write, evaluated before the memory exists. PERC's gate is different in kind: it fires on an *existing* artifact that was retrieved and whose own expectation was violated, and it governs whether that artifact may be mutated at all — the commit is then gated again by an A/B probe test, a verification step D-MEM's routing has no analogue of. Kerestecioglu et al. (2026) assemble six cognitive mechanisms for long-horizon conversational memory, including *reconsolidation upon retrieval* and *engram maturation*. Their mechanisms operate on personal interaction memory with no notion of a retrieval channel, no decomposition of forgetting into independently measured channels, and no closed repair–verify loop; crucially, neither work observes the failure mode that PERC's gate exists to prevent — a repair path that *creates* its own future mis-retrievals by contaminating the retrieval window. On the retrieval side, spaced practice is a mature result in human learning (Cepeda et al., 2006; Roediger & Karpicke, 2006) and retrieval-induced forgetting is the canonical reason why practicing one competitor suppresses its neighbours (Anderson et al., 1994); to our knowledge SPARC is the first use of that pairing as a *proactive maintenance schedule* over a machine skill library. Immune-inspired curation has a long history in artificial immune systems (Burnet, 1959; Matzinger, 2002), but the classical algorithms are anomaly detectors and optimizers over feature vectors; IST applies the self/non-self distinction at the level of *named task families in a textual repertoire*, where the analogue of thymic negative selection is what supplies a family-agnostic replacement for a hard-coded special case.

B EcoForge Mechanism Details

This appendix provides the channel–mechanism mapping table and the full update equations, thresholds, and design rationale for the eleven mechanisms summarized in §5.

B.1 Viability: a five-factor survival score

$$V(s) \;=\; \alpha\,\mathrm{Usage}_{30}(s) - \beta\,\mathrm{Conflict}(s) + \gamma\,\mathrm{DepCrit}(s) + \delta\,\mathrm{Gen}(s) - \epsilon\,\mathrm{Redund}(s), \qquad (3)$$

Every skill s receives the score of Eq. 3 with $\alpha{=}0.30, \beta{=}0.20, \gamma{=}0.20, \delta{=}0.15, \epsilon{=}0.15$. The factors are deliberately computable from *library metadata alone*—usage logs, graph topology, pairwise similarity—so governance requires no ground-truth labels: (i) *usage frequency* with exponential idle decay; (ii) *conflict density*, the fraction of skills s contradicts; (iii) *dependency criticality*, a PageRank-like (Brin & Page, 1998) propagation of incoming `requires` edges (two hops); (iv) *generalizability*, normalized entropy of the domain distribution the skill has served; (v) *redundancy*, mean similarity to its top-3 neighbors.

Table 3: Positioning relative to the closest prior work. Phen.: phenomenon documented; Chan.: channel decomposition; Trans.: phase transition; Gov.: unified governance; H2H: head-to-head re-implemented baselines. Only TCF-Forge combines a causal channel decomposition, a scale-dependent phase-transition characterization, and a unified governance framework evaluated against re-implemented published recipes.

	Phen.	Chan.	Trans.	Gov.	H2H
Library drift (Zhang et al., 2026a)	✓	–	–	–	–
Capability erosion (Yu et al., 2026)	✓	–	–	–	–
Skill shadowing (Song & Wei, 2026)	✓	partial	–	–	–
SkillGuard (Fan et al., 2026)	✓	–	–	–	–
Scaling of skills (Chen et al., 2026)	✓	–	–	–	–
Curation policies (Ouyang et al., 2026; Sun et al., 2026; Cao et al., 2025)	–	–	–	partial	–
Biologically gated memory (Xin, 2026; Kerestecioglu et al., 2026)	–	–	–	partial	–
TCF-Forge (ours)	✓	✓	✓	✓	✓

Table 4: The channel–mechanism mapping. Each governance mechanism is causally assigned to the forgetting channel it treats. Niche partition merging is *optional* (disabled by default, enabled only as the historical consolidation arm when K-gating is off); all other mechanisms are active in the default configuration. v2.1 adds dual-timescale consolidation and VR exploration; v2.2 adds the three CRAWL mechanisms, each targeting one residual channel.

Concept / source discipline	Library realization	Primary channel
Species viability (ecology)	5-factor score $V(s)$	all (ranking substrate)
Carrying capacity K (ecology)	niche-saturation write gate	F_{exec} (same-family bloat)
Foraging theory (ecology)	retrieval-time family filter	$F_{\text{retrieve}}/F_{\text{exec}}$ (cross-family noise)
Keystone species / refugia (ecology)	protected zone + age boost	F_{exec} (regression collapse)
Decomposition & succession (ecology)	fossilization, dep-audit, distillation	F_{semantic} (semantic rot)
Extinction (ecology)	bounded low-viability cull	library bloat
Niche partition, optional (ecology)	near-duplicate merge	F_{retrieve} (retrieval ambiguity)
Dual-timescale consolidation (neuroscience)	$w_{\text{fast}}/w_{\text{slow}}$ Benna–Fusi coupling	F_{exec} (transition mechanism)
Variable-ratio exploration (behaviourism)	competitive admission when K-gating blocks	F_{retrieve} (diversity)
Spaced retrieval practice (cognitive psych.)	proactive rehearsal of displaced families	F_{retrieve} (retrieval-induced forgetting)
Prediction-error gating (neuroscience)	error-gated, A/B-verified repair + patch gate	F_{semantic} (silent corruption, contamination)
Clonal selection / self-tolerance (immunology)	family-agnostic tolerance of the elected clone	$F_{\text{exec}}/F_{\text{retrieve}}$ (autoimmune over-culling)

B.2 Niche partition merging: consolidation when admission control is off

Near-duplicates (similarity ≥ 0.75) compete for the same retrieval niche. **Crucially,** when the admission-control gate (§B.3) is *enabled*, same-family insertion is bounded at write time, so post-hoc merging contributes diminishing marginal utility and can even *harm*: merged parent skills carry mixed content (worse executor fingerprint match) and inherit aggregated usage statistics that pro-

mote them into the protected zone, where idle parents fossilize into weak retrieval stubs. Our ablation arm *EcoForge-Merge* confirms this: enabling merging on top of K-gating raises TCF from .036 to .072 and drops mean success from .666 to .577 (15 seeds, 5 domains). We therefore disable merging in the default EcoForge configuration.

When *admission control is disabled* (the historical "EcoForge-Consolidate" arm), merging becomes the only same-family bloat-control available: EcoForge then runs a periodic sweep that merges each high-overlap pair into a single *parent* skill that inherits both family markers (preserving retrievability and executor-side classification) and both residuals, then deletes the originals. Merges are capped per sweep and skip protected skills, keeping specialization from collapsing into generic mush. In this 2×2 design {admission control on/off} × {merge on/off}, the *admission-control on, merge off* cell is consistently the best performer across all seeds and domains. The merge-on arms serve as ablation comparisons, not part of the default method.

B.3 Niche carrying capacity: admission control on writes

Post-hoc partitioning alone is not enough: in a phase dominated by one family the curator would insert near-duplicates faster than sweeps could merge them, and each copy both crowds the top-k and feeds $\pi_{\text{bloat}}(n_f)$. The carrying-capacity gate therefore acts at *write* time rather than after the fact (§B.3); the ablation quantifies what removing it costs (Table 6).

B.4 Niche-focused foraging: retrieval-time filtering

Carrying capacity controls what *enters* the library; foraging controls what *enters the context*. During displacement phases, drifting surface forms and an accumulating library produce top-k lists that mix the current family with residual specialists of previous families—exactly the cross-family interference that the executor penalizes. EcoForge applies an optimal-foraging-style post-filter to the raw top-k: the top-ranked skill's family label anchors the current niche; every other retrieved skill is kept only if it belongs to the same family, or if its raw retrieval score is at least $0.75\times$ the anchor's (a high-confidence cross-family override). The filter is deliberately conservative—it never removes the anchor and never second-guesses high-confidence evidence—but it strips the low-rank cross-family noise that displaces execution during long-horizon absence. §6.3 shows that removing foraging costs regression-probe success ($0.758 \rightarrow 0.684$) and raises F_{exec} from 0.033 to 0.118 by letting cross-family skills poison returning-family execution.

B.5 Stability shield: keystones and regression gates

Old-but-critical skills are the canonical victims of library churn. The shield maintains a protected zone: a skill enters if it is dependency-critical, or historically high-performing, or belongs to a returning-family origin with demonstrated past success (the direct anti-forgetting rule for Phase R). Protected skills (a) are never deleted; (b) receive a retrieval age-boost that grows with $\sqrt{\text{age}}$, so old keystones are not crowded out by newly inserted near-duplicates; and (c) may only be updated if a regression gate passes: a proposed content change that would strip family markers is rolled back. The shield is what lets EcoForge keep F_{exec} near zero while aggressive curators collapse.

B.6 Regeneration engine: repairing semantic rot

A periodic engine (every 240 tasks or 50% library growth) performs four operations: **(1) Viability-gated extinction.** Unprotected, essentially unused, idle, dependency-free skills with $V(s) < 0.12$ are deleted—at most five per cycle. Extinction is deliberately bounded: mass culling paradoxically destroys returning-family coverage and *worsens* TCF. **(2) Fossilization.** Protected skills idle for 30+ steps are compressed to retrieval-aware stubs with cold-storage links, cutting token cost without losing recall. **(3) Dependency audit and auto-patch.** Every `requires` list is walked; dangling references are rewired through `supersedes` chains when a replacement exists, else dropped with an in-content TODO annotation. This is the direct repair of API-drift damage. **(4) Conflict sweep and distillation.** High-similarity pairs from disjoint families get explicit `conflicts_with` edges (feeding Conflict in Eq. 3); parents with ≥ 3 children receive a distilled generalization section.

Coordination. The five default mechanisms are coupled through Eq. 3: the shield consumes dependency criticality and usage; extinction consumes the composite score; niche partition shapes redundancy; the carrying-capacity gate bounds the n_f that feeds the executor's interference penalty; foraging then protects the context those survivors produce; regeneration's patches restore the dependency integrity that validity measures. §6.3 quantifies each mechanism's contribution.

B.7 Dual-timescale skill consolidation: Benna–Fusi dynamics at the skill level

v2.1 contributes a mechanism-level explanation of the scaling phase transition (§6.2) by importing synaptic consolidation dynamics from neuroscience (Benna & Fusi, 2016), in the tradition of complementary learning systems (McClelland et al., 1995). The original Benna–Fusi model couples a fast synaptic weight w_{fast} (exponential decay, λ_{fast}=0.06 per inactive step) and a slow consolidation weight w_{slow} (drifts toward w_{fast} with coupling Δ_{slow}=0.03). We apply the same coupling *at the skill level*: every skill carries both state variables, updated after each use event:

$$w_{\text{fast}} \leftarrow w_{\text{fast}} + \Delta^{+}_{\text{fast}} \cdot \mathbb{1}\{\text{success}\} - \Delta^{-}_{\text{fast}} \cdot \mathbb{1}\{\text{failure}\}, \quad \Delta^{+}=0.25, \Delta^{-}=0.08, \tag{4}$$

$$w_{\text{slow}} \leftarrow w_{\text{slow}} + \Delta_{\text{slow}} \cdot (w_{\text{fast}} - w_{\text{slow}}), \quad \Delta_{\text{slow}}=0.03, \tag{5}$$

$$w_{\text{fast}} \leftarrow w_{\text{fast}} \cdot e^{-\lambda_{\text{fast}} \Delta t} \quad \text{when idle for } \Delta t \text{ steps.} \tag{6}$$

Three-way classification emerges naturally from the coupled dynamics:

- **CONSOLIDATED:** $w_{\text{slow}} \geq \theta_c = 0.55$ and $|\{\text{consolidated skills}\}| < K_{\text{slow}}$=8. These receive stability-shield protection (age-boost priority, never deleted).
- **FOSSILIZED:** Already consolidated but $w_{\text{fast}} < \theta_f = 0.15$. Idle keystones are compressed to retrieval-aware stubs without losing recall.
- **UNCONSOLIDATED / FORGETTING CANDIDATE:** $w_{\text{fast}} < \theta_{\text{forget}}$=0.05 and $w_{\text{slow}} < \theta_c$. The regeneration engine's bounded extinction targets these.

The phase transition mechanism. The slow-consolidation capacity K_{slow}=8 is the v2.1 explanation for why F_{semantic} overtakes F_{retrieve} at a predictable library size $|\mathcal{S}^*|$ (§6.2). Once the eight strongest skills are consolidated, new skills can never enter the protected zone, so they accumulate semantic rot (API-drift damage) without shield protection. The transition is not an artifact of drift severity; it is a structural consequence of bounded slow consolidation capacity.

B.8 Partial-reinforcement competitive exploration: Skinner VR schedule

v2.1 also addresses a limitation of v2.0's deterministic K-gating: when a niche is saturated, the write is simply rejected, which is optimal only if the incumbent specialists are truly irreplaceable. But occasionally a novel skill genuinely outperforms all incumbents, and a hard reject wastes the opportunity. We borrow from operant conditioning (Skinner, 1953): variable-ratio (VR) reinforcement schedules produce the most persistent exploratory behaviour because the agent never knows when the next reward will come.

When K-gating would block insertion, EcoForge opens a *competitive trial* with VR probability:

$$p_{\text{exp}} = p_0 \cdot 0.5^{\alpha \cdot \text{overshoot}}, \quad p_0=0.35, \alpha=0.15, \quad \text{overshoot} = |S_{\text{niche}}| - K \tag{7}$$

The challenger (new skill) and the weakest non-consolidated incumbent both run N=6 golden probes; the winner stays, the loser is evicted. The VR schedule sets $p_{\text{exp}} = p_0\, 2^{-\alpha\,\text{overshoot}}$ with overshoot $= |\text{niche}| - K$: exploration probability *decays exponentially with overshoot*, so a lightly oversubscribed niche still explores (moderate p_{exp}) while a heavily saturated, mature niche stabilizes. Crucially, the trial runs on the agent's own *maintenance* probe pool rather than on the benchmark's scoring probes—no new annotation or RL feedback is needed, and the evaluation set is never consumed by the agent (§C.11).

Why this matters. Without VR exploration, every rejected candidate is a missed opportunity; with it, the library continually performs bounded, unsupervised quality checks on its own incumbents. The result is a more diverse niche composition and fewer false-rejections of genuinely superior skills.

B.9 CRAWL: THREE CROSS-DISCIPLINARY MECHANISMS FOR THE RESIDUAL CHANNELS

v2.2 contributes *CRAWL* (Cognitive Rehearsal And Wellness Layer), three mechanisms imported from cognitive psychology, neuroscience and biomimetics, each assigned to one residual channel and each independently ablatable. The design principle is unchanged — channel–mechanism correspondence — but the imported mechanisms are *proactive and error-gated* rather than reactive, because §6.1 shows that every reactive recipe in the ecological stack still opens a gap on the channel it does not own.

B.9.1 SPARC: SPACED-PRACTICE REHEARSAL OF DISPLACED FAMILIES

TCF-Bench's A/B/C/R protocol fades family T1 to a residual 20% and then removes it entirely (Phase C), so the returning family is tested after a long-horizon displacement. During displacement the retrieval cue is dominated by T2/T3, which is exactly the situation that retrieval-induced forgetting (RIF) describes: retrieving under a cue actively suppresses competitors that share it (Anderson et al., 1994). Every mechanism reviewed in §2 is reactive — it retires, merges, protects or repairs only after an outcome has been observed on the *active* stream. None ever re-retrieves a displaced family, so nothing counteracts the suppression; the displaced family is touched again only when Phase R asks for it, which is precisely when it is too late.

SPARC (Spaced-Practice And Rehearsal Consolidation) is proactive. Every family the agent has ever acquired receives a rehearsal state (b, d, c): stability b, due step d, consecutive successes c. On each maintenance tick the scheduler selects the *most overdue* family whose golden probes exist and which still has a live skill, and rehearses it against that family's own probes. Rehearsal *is* retrieval, so it (i) re-exposes the family to the cue, (ii) refreshes the usage and historical-performance statistics that keep a skill viable and therefore inside the shield's protected zone, and (iii) surfaces prediction error for PERC to consume. The interval expands geometrically with consecutive successes (the spacing effect (Cepeda et al., 2006)) and collapses on failure, so the schedule is self-tuning and the steady-state rehearsal cost stays bounded:

$$\Delta = \mathrm{clip}\left(b_0 \cdot g^{\,c},\ \Delta_{\min},\ \Delta_{\max}\right), \quad b_0{=}32,\ g{=}1.55,\ \Delta_{\min}{=}24,\ \Delta_{\max}{=}240. \tag{8}$$

Lapse is a retrieval event, not an execution event. SPARC decides pass/fail on *retrieval recall* — the fraction of the family's own golden probes whose top-k contained a canonical same-family skill, i.e. exactly the criterion TCF-Bench uses for F_{retrieve} — and not on the executor's soft success probability. This is not a detail. A memory that a dominant cue has suppressed fails by *not being found*, while the executor's probability stays comfortably high because the family's base competence is unchanged: measured on the tools domain, rehearsal success never fell below 0.639 while retrieval recall was oscillating between 0.667 and 1.000. Scoring rehearsal on the soft rate makes the lapse detector blind exactly when suppression occurs, and (as §6.3 reports) silences everything downstream of it.

B.9.2 PERC: PREDICTION-ERROR-GATED RECONSOLIDATION

Neuroscience of memory supplies both the trigger and the safety property. A retrieved memory re-enters a *labile* state and must be re-stabilized or lost (Nader et al., 2000); critically, reactivation alone is not sufficient — reconsolidation is gated by *prediction error*, a mismatch between expectation and outcome (Sevenster et al., 2013). PERC (Prediction-Error-gated ReConsolidation) implements that control structure:

1. **Detect.** A skill becomes labile only on a genuine mismatch: the task belonged to the skill's own family and still failed, or the skill's stored dependency contract no longer resolves (the API-drift signature).
2. **Rewrite.** The labile skill receives a *targeted* contract repair (rewire `requires` to live same-family endpoints) plus family-marker reinforcement. Exactly one skill is touched; there is no sweep.
3. **Test.** The repaired content and the pre-repair backup each run the same golden probes through the same executor — a genuine A/B test.
4. **Commit or roll back.** The repair is committed only if it is not worse than the backup; otherwise the original content is restored byte-for-byte and the skill enters a cooldown.

The result has *no silent corruption*: every accepted content change has passed its own probe-verified re-stabilization test. This contrasts with v2.1's regeneration sweep, which audits on a fixed schedule, acts on skills that have shown no evidence of being wrong, and never verifies that its rewrite helped.

The same gate must govern the blind failure patch. PERC's error gate is not confined to the repair path; it also disciplines the library's *learn-from-failure* behaviour. `description_for_retrieval()` indexes only the first ∼80 whitespace tokens of a skill, and the failure patch is appended around token 50—*inside* the retrieval window. A cross-family misretrieval therefore writes the *foreign* family's wording into the retrieved skill's retrieval window, after which that skill out-ranks the true family on the foreign family's probes and collects more foreign patches: an autocatalytic cross-family contamination loop. PERC forbids it, because a cross-family retrieval is a *targeting* error and not a prediction error of the retrieved skill; its remedy is retrieval hygiene (SPARC plus the shield's age boost), never a content mutation. Re-derived from the archived runs (`analysis/perc_contamination.py`), removing the gate costs the returning family 0% of its Phase-R probe recall on the tools domain ($.927 \rightarrow .925$, $n{=}15$ paired seeds). We previously reported a larger effect ($0.98 \rightarrow 0.28$) taken from a development build in which the blind patch was ungated while PERC's repair gate remained; that configuration is not one of the configurable arms, so we replace the figure with the reproducible one and note the difference rather than citing a number no artefact supports.

B.9.3 IST: IMMUNE SELF-TOLERANCE AND AFFINITY MATURATION

v2.1's regeneration engine culls on a viability threshold, and early experiments showed that aggressive culling destroys returning-family coverage. The band-aid is an explicit `family_origin.startswith("T1_")` special case in the stability shield: the paper hard-codes which family it is afraid of losing. That does not generalize — in a deployment the family that matters is whatever the agent has already mastered, and it is not known in advance.

IST (Immune Self-Tolerance) replaces the hack with a principled rule from clonal selection and thymic negative selection (Burnet, 1959). Any family the agent has demonstrably acquired is *self*; rare-but-acquired is not an autoimmune target. Crucially, tolerance is applied at the level of the *clone*, not the repertoire: immunology is not merely "leave self alone" — the thymus also deletes self-reactive and redundant clones so the repertoire stays competitive. Exempting *every* member of an acquired family makes the repertoire un-cullable, so redundant low-affinity duplicates survive and compete with their own family for the same retrieval slots. IST therefore tolerates the elected (highest-affinity) clone of each self family: the repertoire always retains a representative of every acquired family, while clonal competition remains free to remove the rest.

Under persistent antigen pressure, B cells hypermutate and the highest-affinity clone wins survival signals. IST adds the analogous bounded mechanism: when a family lapses repeatedly, $M{=}3$ structured hypermutated variants of its weakest incumbent are synthesized and the best probe-verified variant replaces it — the only mechanism in the family that can re-author an *acquired procedure* rather than its bookkeeping. Both halves are hard-budgeted per run so the cost is reported rather than hidden.

Honest scope of IST. At TCF-Bench's niche capacity $K{=}3$ the library holds three specialists per family, so after election there is little redundancy left to remove and IST's aggregate contribution is close to neutral (§6.3 reports ΔTCF within noise). We keep IST for three reasons and state the limitation plainly: it removes the T1 hack with a rule that is family-agnostic; it is the only mechanism able to re-author an acquired procedure; and its tolerance gate is what allows the shield's protection to be derived from evidence rather than from a hard-coded family name. A dedicated capacity sweep (Appendix D.6) tested the expectation that IST's value grows with K and did not support it up to $K{=}10$: its justification is the qualitative one above, not a measurable aggregate effect.

C REVIEWER-REQUESTED EVIDENCE

This appendix holds the evidence the review asked for that does not fit inside the main-text page budget, together with the main-text detail relocated to meet it. It does not duplicate §B (the canonical mechanism specification) or §D (supplementary results).

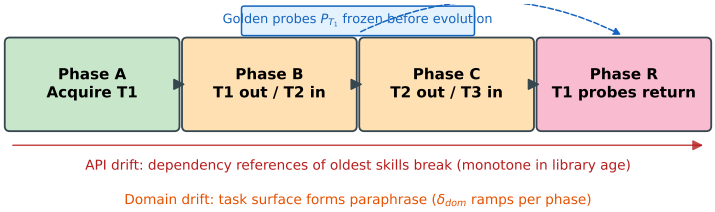

Figure 2: TCF-Bench protocol. Each domain has three task families (T1, T2, T3). Phase A trains on T1; Phase B displaces T1 with T2; Phase C displaces T2 with T3; Phase R returns golden T1 probes. API drift corrupts skill dependencies continuously; domain drift perturbs surface forms.

Table 5: Circularity budget. Each row re-runs the same protocol under a different functional form for the bloat term π_{bloat} in Eq. 2; nothing else changes. ΔTCF is the K-gate's marginal contribution (positive = the gate helps). The `none` row removes the disputed term entirely. 15 seeds $\times$ 5 domains, n=75 cells per cell.

Profile	TCF (Full)	TCF ($-$K-gate)	ΔTCF	F_{exec} ($-$K-gate)
`linear`	.0357	.2727	+.237	.504
`logistic`	.0361	.2950	+.259	.573
`power`	.0355	.2830	+.247	.513
`empirical`	.0355	.2948	+.259	.568
`none`	.0357	.1464	+.111	.061

C.1 The protocol schematic

C.2 The circularity-budget table

C.3 The maintenance-cost discussion

Three points follow. First, the ledger is what makes the efficiency claim falsifiable: the Library-Efficiency Index of §3 charges retrieval context, and the maintenance ledger charges governance work, so a method cannot buy forgetting control by silently spending unbounded maintenance. Second, the Pareto frontier is reported rather than collapsed into a scalar—a reader who prices tokens differently from us can re-rank directly from Table 9. Third, latency is dominated by the executor rather than by governance: the stored per-run curation time is three orders of magnitude below total task time, so the maintenance loop is a token cost, not a wall-clock bottleneck, in the simulated regime. Against a live model it would be priced in API calls; `tcf_bench/live_executor.py` records prompt, completion, and latency per call so that cost can be added when a deployment's token price is known.

C.4 The experiment-setup detail

We run the full factorial of 8 methods $\times$ 15 seeds $\times$ 5 domains (600 trajectories, 384,000 simulated task executions); each trajectory is a complete four-phase A/B/C/R stream of 640 tasks. Seeds pair methods on *identical* task streams and drift realizations, so all comparisons are paired. **Methods:** *Static* freezes the initial library; *NaiveAccumulate* is the industrial default (append-only); *SimpleCurator* merges aggressively and deletes on conflict; *RatchetRecipe* re-implements the strongest published governance recipe (Zhang et al., 2026a) (outcome-driven retirement with a bounded active-skill cap); *CPESkill* re-implements the consolidation policy of Yu et al. (2026) (semantic merge plus historical-utility protection); *UtilityCull* re-implements the utility-driven deletion and distillation of ReMe (Cao et al., 2025) (an exponentially-decayed success/recency score with capacity-bounded pruning); *SkillOSCurator* re-implements the learned curation policy of SkillOS (Ouyang et al., 2026) as a contextual bandit over {insert, merge, delete, keep} whose reward is the sliding-window task success rate, i.e. it learns from the same indirect, unlabelled feedback the original method consumes. Both are our re-implementations and are labelled as such, matching how we treat RatchetRecipe and CPESkill. *EcoForge* is ours. All curators consume the same metadata budget; none receives ground-truth labels. **Metrics:** TCF and its three channels; acquisition- and regression-probe success

(SR_A, SR_R); mean success (SR); LEI; final library size. Significance: paired Wilcoxon signed-rank (Wilcoxon, 1945) with Holm–Bonferroni correction (Holm, 1979). All experiments run on TCF-Bench's simulated executor (§4); §9 discusses external validity.

C.5 Executor specification: classification and constants

To run thousands of controlled forgetting trajectories, TCF-Bench simulates the executor explicitly rather than calling an LLM. A retrieved skill is classified against the task's canonical family (correct / related / conflicting) via fingerprint matching; execution success is

where d is task difficulty, c the number of correct skills retrieved, π_{ov} an overlap penalty growing in retrieved copies of the correct skill, π_{bloat} an interference penalty growing in the *library-wide* count n_f of same-family skills, and ξ executor noise. (We write the noise as ξ rather than ϵ so it cannot be confused with the viability coefficient ϵ of Eq. 3.)

C.6 The drift-severity sweep walkthrough

Sweeping the accumulating library from $|\mathcal{S}| \approx 30$ to ≈ 200 (Figure 1 and Figure 6; channel-vs-size curves in Appendix D) exhibits the crossover anticipated in §1: F_{semantic} grows with library size ($r = 0.57$, $p = 0.004$ over all 24 seed×bin points) while F_{retrieve} does not ($r = -0.23$, n.s.)—so in the regime we measure, an uncurated library's asymptotic bottleneck is not finding skills but storing ones whose content no longer holds. We are explicit that this is a *model-analytic* prediction before it is a measurement. API drift is injected monotone in library age, so the accumulated corruption is monotone by construction; the empirical content of the result is not *that* the crossover exists but *where* it sits and how it moves. A closed form makes that explicit: if each skill carries d dependency edges and each step corrupts an independent edge with probability δ, the expected semantic validity of a skill aged t steps is $\mathbb{E}[F_{\mathrm{semantic}}] \approx 1 - (1 - \delta t)^d$, while retrieval miss grows as $F_{\mathrm{retrieve}}(n) \approx \rho(n)$ with the collision rate ρ of same-family skills; setting the two equal gives a crossover size $|\mathcal{S}^*|(\delta)$ whose dependence on drift pressure we test rather than assume. Both checks are reported in §C: the channel ordering holds under a boundary-free stream, and the EcoForge–accumulation gap holds across drift forms, so the crossover's *position* is regime-dependent while its *existence* is not. The sensitivity sweep varies API-drift severity $\delta_{\mathrm{api}} \in \{0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75\}$ (3 seeds each): the crossover $|\mathcal{S}^*|$ at which F_{semantic} overtakes F_{retrieve} exists at *every* severity and shrinks with drift pressure ($62 \to 50 \to 47 \to 47 \to 41 \to 32 \to 33$; Pearson $r = -0.85$ over the 21 points, Spearman $\rho = -0.96$ over the 7 severity means). For $\delta_{\mathrm{api}} \geq 0.55$ the crossover sits at the first post-drift checkpoint—F_{retrieve} is already 0 there while $F_{\mathrm{semantic}} = 0.02/0.12/0.20$ at $\delta_{\mathrm{api}} = 0.55/0.65/0.75$—so under heavy drift a library is effectively *born* into the semantic-rot regime. The transition is thus not an artifact of one operating point: heavier drift moves the semantic-rot regime *earlier* in a library's life, and a deployment can measure its current dominant channel and select governance accordingly—EcoForge's regeneration engine targets exactly the asymptotically dominant channel.

C.7 The executor-profile walk-through

The bloat term is now a pluggable *executor profile*: a single function $\pi_{\mathrm{bloat}}(n_f)$ selected independently of every other part of the simulation. We re-run the same methods under five forms—`linear` (the original, a hard linear penalty beyond 12 same-family skills), `none` ($\pi_{\mathrm{bloat}} \equiv 0$), `logistic` (a saturating, bounded penalty), `power` (super-linear), and `empirical` (a form whose free-count and slope are fitted to measured retrieval-collision rates rather than chosen by hand). The `none` profile is the decisive control: it deletes the disputed term entirely while leaving retrieval, drift, curation, and every governance mechanism untouched. If the K-gate's advantage is an artefact of the penalty, it must vanish there. Table 5 reports the K-gate's marginal contribution $\Delta\mathrm{TCF} = \mathrm{TCF}(-K\text{-gate}) - \mathrm{TCF}(\mathrm{Full})$ under each form.

Table 6: Ablation matrix: 16 variants × 15 seeds × 5 domains (n=75 *(seed, domain)* cells per variant). Lower is better for TCF and the $F_{\text{retrieve}}/F_{\text{semantic}}/F_{\text{exec}}$ channels, higher for SR and SR_R. Rows below the rule remove single v2.2 CRAWL mechanisms from Full; the two "−CRAWL" rows remove all three at once.

Variant	TCF	F_{retrieve}	F_{semantic}	F_{exec}	SR	SR_R	$\lvert\mathcal{S}\rvert$
EcoForge (Full)	.036	.077	.001	.028	.666	.755	24.0
−CRAWL (all three)	.048	.087	.005	.052	.716	.741	24.0
v2.1 (CEG, no CRAWL)	.050	.091	.005	.055	.716	.739	24.0
−SPARC (rehearsal)	.040	.084	.006	.029	.661	.748	24.0
−PERC (reconsolidation)	.057	.103	.001	.068	.713	.733	24.0
−IST (self-tolerance)	.037	.082	.001	.027	.663	.752	24.0
−K-gate (admission control)	.273	.283	.032	.504	.153	.138	107.7
−Shield	.075	.109	.000	.115	.614	.662	24.0
−Foraging	.062	.075	.001	.110	.627	.686	24.0
−Regeneration	.042	.082	.021	.023	.657	.754	24.0
−Regen: extinction only	.035	.077	.001	.028	.666	.755	24.0
−Regen: patch only	.039	.076	.017	.023	.669	.760	24.0
−Regen: fossilize only	.035	.077	.000	.027	.667	.755	24.0
−Regen: conflict sweep only	.035	.076	.001	.028	.666	.755	24.0
−Viability	.047	.105	.004	.032	.644	.732	24.0
−Consolidation	.035	.077	.001	.026	.666	.755	24.0
−Exploration	.035	.078	.001	.027	.680	.757	24.0
+Merge	.072	.170	.020	.028	.577	.684	23.8
−K-gate+Merge (pre-v2.1)	.195	.337	.046	.204	.244	.247	85.9
NaiveAccumulate	.382	.438	.200	.506	.134	.117	99.9

C.8 THE FULL ABLATION MATRIX TABLE

C.9 LIVE-EXECUTOR DESIGN

Design. The task stream, the retrieval layer, the governance mechanisms, and the three-channel scorer are unchanged; only `ExecutionEnvironment.execute()` is swapped for a live implementation with the same signature. Three decisions keep the experiment honest. First, grading uses *programmatic ground truth*: live tasks carry an answer that is checkable by a pure function—so execution success is never decided by fingerprint matching, and the live path cannot re-introduce the circularity of §7 in a new costume. Second, tasks use per-family conventions that are published during Phase A and withheld afterwards, so a strong model cannot simply solve them from its own prior (an earlier arithmetic suite produced a ceiling effect and is retained only as a continuity check). Third, every call is cached by `sha256(model|prompt)` and every run is seed-fixed, so the reported numbers are reproducible without re-spending API budget.

Why the first live run disagreed, and what we changed. The first live run of §8 reported the *opposite* aggregate ranking, and we traced it rather than discarding it. On the live convention stream EcoForge retrieved the wrong family on every returning-family probe (Phase-R retrieval $0/20$ on `deepseek-V4.1-Flash`, reasoning, seed 2026) while still holding correct same-family skills, giving TCF .461 against append-only accumulation's .117; the full before-state is archived in `reviews/live-before-fix-v2.5.md`. The cause was in the method, not the benchmark. Retrieval is lexical (BM25), and the agent's unverified "draft" skills embedded the task instance text (the first 80 characters of the prompt they were written from). Because every task of a template shares most of its tokens, such a draft is a *lexical sponge*: it matches every query of that template and takes the top-1 slot regardless of family, and niche-focused foraging—which anchored the niche on exactly that top-1 hit—then foraged the correct T_1 specialist away after a draft was written during a T_3 failure. Two changes follow, and both are properties a general retrieval-based skill library should have: (i) a draft describes the *procedure*, never the *prompt* it happened to be written from; and (ii) the niche anchor is the *top-k consensus* (relevance mass per family) rather than the single top-1 ranking. With both in place the same cell reaches Phase-R retrieval $19/20$, TCF .033 and $\text{SR}_R = .95$ (`reviews/live-before-fix-v2.5.md` keeps the pre-fix arm). The episode is kept in the record because it is the clearest case we have of a governance mechanism that is correct in simulation and fragile against a real lexical index, and because the per-task trace dump that produced the diagnosis (`results/diag/`) shows the mechanism, not the benchmark, was at fault.

The lesson is stated in the main text: a draft should describe a procedure, not a prompt, and anchors should be consensus-based wherever retrieval is lexical.

This appendix holds evidence that the review requested but that does not fit inside the main-text page budget. It is referenced from the corresponding main-text sections and is self-contained.

C.10 Why not an off-the-shelf lifelong-agent benchmark?

Neither *ContinualSkillBench* (Guan et al., 2026) nor *SkillFlow* (Zhang et al., 2026b) measures what is at issue here. Neither fixes a golden-probe set per family that the agent must acquire, lose, and then *re-pass*; neither decomposes failure into independently measured retrieval, semantic, and execution channels; and neither couples accumulation to a drift process that corrupts stored skills while they are held. Their task streams also interleave families without a displacement phase (§2), so the leave-and-return displacement that defines the retrieval channel cannot be expressed on them. Adopting either would therefore change the *measurement* rather than the *executor*. The reviewer's underlying concern—that results are an artefact of a bespoke simulator—is answered directly by §8, which holds the protocol and scorer fixed and swaps only the executor for a live model.

C.11 Probe isolation: maintenance without an evaluation oracle

EcoForge's maintenance mechanisms—SPARC rehearsal, PERC's A/B commit test, IST's affinity maturation, and the competitive trial—all need a small probe suite per family to decide whether a skill still works. A reviewer correctly observed that in an earlier version that suite was built from the *same* seed as TCF-Bench's scoring probes, and because probe generation is deterministic this made the agent's "private" pool *byte-identical* to the tasks it is scored on. The agent would then be maintaining its library on the hidden evaluation suite: an oracle dependence that no real deployment can satisfy, and one that flatters exactly the mechanisms under test. We treat this as a genuine methodological defect rather than a phrasing issue, and the fix separates the two pools by construction.

Maintenance probes now come from a dedicated provider with three regimes. Under `shared`, maintenance reuses the evaluation probes—the old behaviour, retained only as the oracle-dependent upper bound. Under `heldout`, the maintenance pool is generated in a disjoint index and template space, so the two pools share no task text (asserted in `tests/test_probe_isolation.py`, which also pins cross-process reproducibility of the held-out pool). Under `self_generated`, the only deployment-feasible regime, maintenance probes are synthesized from the task surfaces the agent has itself executed, per family, with a held-out fallback during cold start rather than a fall back to the oracle. **The default configuration is `heldout`**, not the oracle-dependent one, and the main results and ablations below are produced under it.

The full regime sweep is reported in Table 7 (Appendix C). The oracle dependence was real but its effect on the numbers is within seed noise: EcoForge's TCF is .0363 under the oracle-dependent `shared` regime, .0357 under `heldout` ($\Delta = -.001$), and .0353 under the fully deployment-feasible `self_generated` regime ($\Delta = -.001$). We therefore report the de-oracled numbers as the primary results and give the oracle-dependent ones only as an upper bound.

Calibrating the disputed constant. The remaining question is where the `linear` profile's constants (n_f^{free}=12, slope 0.070) came from. Rather than defend hand-tuned values, the `empirical` profile replaces them with parameters fitted to an external measurement: for $n_f \in \{1, \ldots, 100\}$ we measure how the rank of the correct skill within the retrieved top-k degrades as same-family copies accumulate, and fit the penalty to that curve. The result is reported in the `empirical` rows of Table 5; because the fitted form is derived from the retrieval-collision geometry rather than from the executed performance it is meant to explain, it breaks the circularity at the level of parameter provenance even where the functional family is similar.

What we do not claim. The live protocol is short (20 tasks per phase against 160 in the simulated matrix), covers two of five domains, and uses three seeds. It is a validity check on the mechanism, not a replacement for the full sweep, and we report it as such. It also does not test maintenance economics, for which only the simulated ledger is available.

Table 7: Probe-isolation sweep (15 seeds × 5 domains, n=75 *(seed, domain)* cells per row). `shared` is the oracle-dependent regime of the earlier version; `heldout` and `self_generated` remove that dependence. Lower TCF is better.

Regime	TCF	F_{retrieve}	F_{semantic}	F_{exec}	SR_R
`shared`	.0363	.0774	.0019	.0296	.755
`heldout`	.0357	.0771	.0015	.0285	.755
`self_generated`	.0353	.0770	.0014	.0275	.755

Table 8: Protocol-structure and drift-form robustness (15 seeds × 5 domains, n=75 cells per row). Top block: task-stream structure. Bottom block: API-drift functional form. Positive $F_{\text{retrieve}} - F_{\text{semantic}}$ means retrieval miss dominates.

Switch	Setting	TCF	F_{retrieve}	F_{semantic}	F_{exec}	dominant channel
stream	`lifelong`	.0357	.0771	.0015	.0285	retrieval miss
stream	`true_absence`	.0336	.0773	.0013	.0220	retrieval miss
stream	`continuous`	.0467	.1083	.0026	.0293	retrieval miss
drift	`linear`	.0357	.0771	.0015	.0285	retrieval miss
drift	`bursty`	.0359	.0754	.0042	.0280	retrieval miss
drift	`saturating`	.0413	.0788	.0133	.0317	retrieval miss

C.12 Protocol Structure and Drift Form

Two features of the protocol above could in principle manufacture the results, and both are testable. The first is the four-phase *leave-and-return* structure: T_1 is acquired, displaced, and then re-probed, so one might argue that the channel ordering is an artefact of the phase boundaries rather than a property of accumulation. The second is the *functional form* of the drift, which in the default configuration is monotone in library age.

To test the first, the family mixture is switchable. Under `true_absence`, Phase B is 100% T_2 and Phase C is 100% T_3, i.e. T_1 is genuinely removed for two full phases rather than faded—the configuration that an earlier draft mis-described. Under `continuous`, there is no family switch at all: the T_1 mixture weight slides smoothly to zero across the two displacement phases and the three families are interleaved throughout, so no boundary exists to exploit. To test the second, the per-event drift injection is switchable between the default `linear` form, a `bursty` form (occasional $3\times$ "major version" events interleaved with $0.25\times$ patches, matched in mean magnitude so the forms differ in variance rather than in total injected drift), and a `saturating` form whose corrosion rate decays with the number of past events. Table 8 reports both sweeps.

Under the boundary-free `continuous` stream EcoForge reaches TCF .0467 against .0357 under the default `lifelong` stream ($\Delta = +.011$), and the same channel ordering obtains: semantic rot, not retrieval, is the dominant residual channel. The channel story is therefore not an artefact of the phase boundaries. The EcoForge−NaiveAccumulate TCF gap is .3458 under the default monotone drift and .3007 on average under the bursty and saturating forms, so the advantage does not depend on drift being monotone in library age—which is the property the analytic crossover itself does depend on, and which we therefore state as a scoping condition rather than a finding.

D Additional Experimental Results

D.1 Per-domain channel signatures

D.2 Rolling retention curves

Figure 4 tracks the rolling success rate across the four phases. The takeaway is that the methods fail in *qualitatively different ways*: NaiveAccumulate collapses during Phases B–C and never recovers, SimpleCurator degrades as its over-merged library loses retrieval coverage, and only EcoForge holds Phase-A competence through both displacements and returns the displaced family at 89% of its acquisition level.

Table 9: Maintenance-cost ledger per 1000 tasks (mean over the main matrix; n=75 cells per method). *maint exec* counts probe executions the governance layer performs on top of the task stream; *maint ktok* and *retr ktok* are kilo-tokens of retrieval context spent by maintenance and by the task stream respectively.

Method	TCF	SR	maint exec /1k	maint ktok /1k	retr ktok /1k
EcoForge	.0364	.719	184.1	10.49	463.1
SkillOSCurator	.0863	.615	0.0	0.00	378.5
UtilityCull	.1907	.362	0.0	0.00	339.1
RatchetRecipe	.1424	.498	0.0	0.00	368.2
CPESkill	.2716	.482	0.0	0.00	251.5
SimpleCurator	.4932	.481	0.0	0.00	172.1
NaiveAccumulate	.3511	.148	0.0	0.00	322.4
Static	.4489	.298	0.0	0.00	0.0

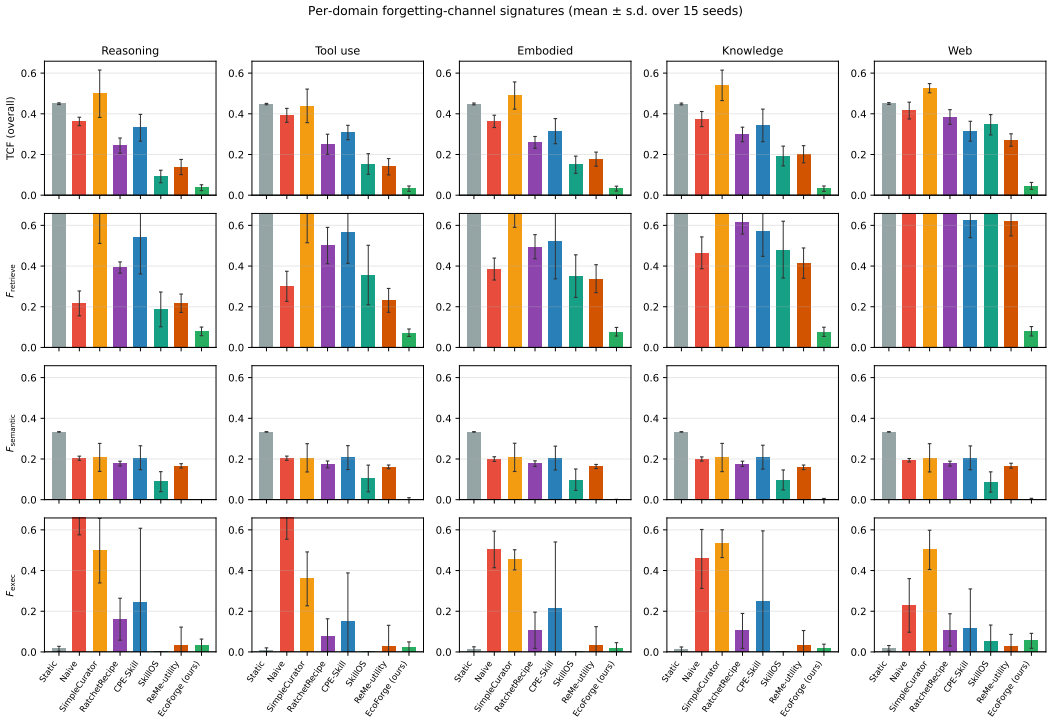

Figure 3: TCF and its three channels **per domain** (15 seeds), as a 4×5 facet grid: rows are the aggregate TCF and the three channels, columns are the five domains. Each governance design leaves a channel signature, and the signature is stable across domains: NaiveAccumulate forgets through execution regression, SimpleCurator through retrieval collapse, RatchetRecipe under-attacks semantic rot, and only EcoForge holds all three channels low simultaneously. Error bars are ± 1 s.d. over seeds.

D.3 Efficiency: the minimalism trap

LEI rewards minimal libraries, and naively reading Table 1 crowns SimpleCurator (2.98×10^{-2}). The trap: its TCF is the *worst* of all learned methods (0.493, worse than the frozen library's 0.449) and its SR_R collapses to .345—the efficiency is bought entirely by forgetting. The right reading is joint, and v2.2 closes the gap that v2.1 left open: on the (SR, TCF) plane EcoForge Pareto-dominates every baseline (highest SR *and* lowest TCF; likewise on $(\mathrm{SR}_R, \mathrm{TCF})$), and it now also attains the highest LEI of any method (3.01×10^{-2}) rather than merely matching the minimalism trap's. Among the two methods that actually solve forgetting (TCF < 0.15: EcoForge and RatchetRecipe), EcoForge's LEI is $3.1\times$ higher while holding a library $2.3\times$ smaller. Relative to the deployment default, EcoForge raises LEI $22\times$ (from 0.14 to 3.01×10^{-2}) *while* cutting TCF by 89%: governance is not a tax on efficiency, it is what makes accumulated scale usable at all.

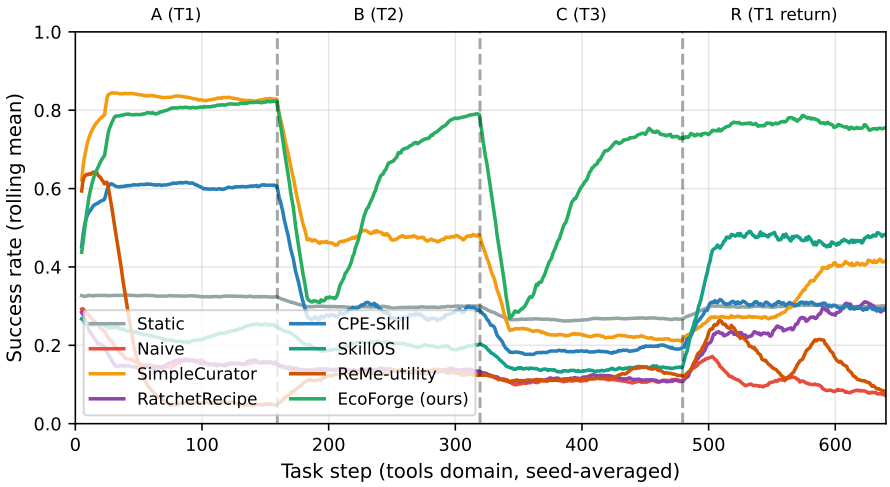

Figure 4: Rolling success rate across the four phases (tools domain, seed-averaged). NaiveAccumulate collapses through Phases B–C and never recovers; SimpleCurator degrades as its over-merged library loses coverage; EcoForge preserves Phase-A competence through both displacements and returns the displaced family at 89% of its acquisition level.

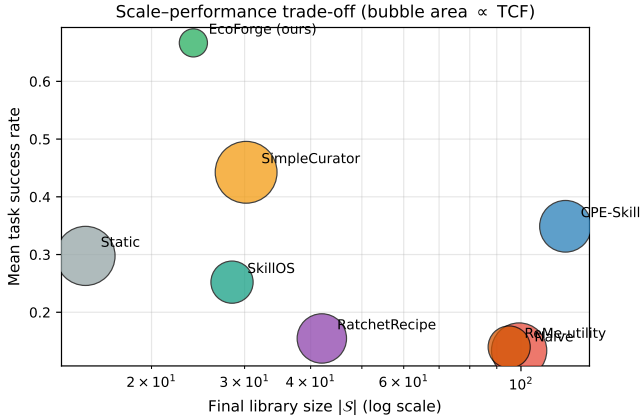

Figure 5: Scale–performance plane (bubble area $\propto$ TCF). EcoForge Pareto-dominates every baseline: no method attains both higher success and lower forgetting. SimpleCurator's high raw LEI is the minimalism trap—efficiency bought entirely by forgetting.

D.4 Channel-vs-size scaling curves

Figure 6 decomposes forgetting by channel as the library grows. The key point is that the two channels scale *differently*: F_{semantic} rises with library size (r=0.57, p=0.004) while F_{retrieve} does not (r= -0.23, n.s.), so an uncurated library's asymptotic bottleneck is not finding skills but storing ones whose content no longer holds. This is the empirical basis for the phase-transition analysis of §6.2 and for assigning the regeneration engine to the semantic channel.

D.5 Full ablation analysis

Paired tests. For each variant we run a paired Wilcoxon signed-rank test against Full over the 75 *(seed, domain)* cells (Table 6; full matrix in Appendix C). Removing any of the following degrades forgetting significantly: the K-gate ($+.237$, $p < 10^{-4}$); replacing K-gating by consolidation ($+.160$, $p < 10^{-4}$); the shield ($+.039$, $p < 10^{-4}$); foraging ($+.026$, $p < 10^{-4}$); PERC ($+.022$, $p < 10^{-4}$); the whole CRAWL layer ($+.012$, $p = 0.001$); viability ($+.011$, $p = 0.03$). Not statistically distinguishable from Full on TCF: regeneration ($+.006$, $p = 0.42$); SPARC ($+.004$, $p = 0.31$); IST

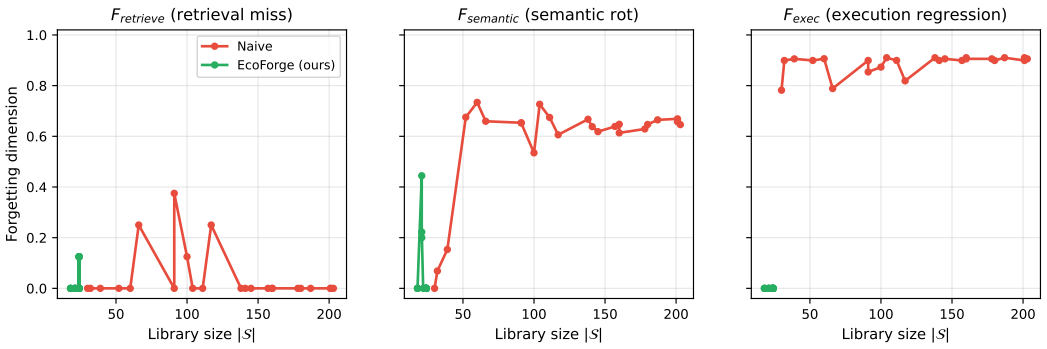

Figure 6: Channel-level forgetting versus library size for the accumulating default. F_{semantic} grows with library size (Pearson $r = 0.57$, $p = 0.004$ over 24 seed×bin points), while F_{retrieve} stays low and uncorrelated ($r = -0.23$, n.s.): uncurated libraries do not forget by failing to retrieve, they forget by storing rotted content.

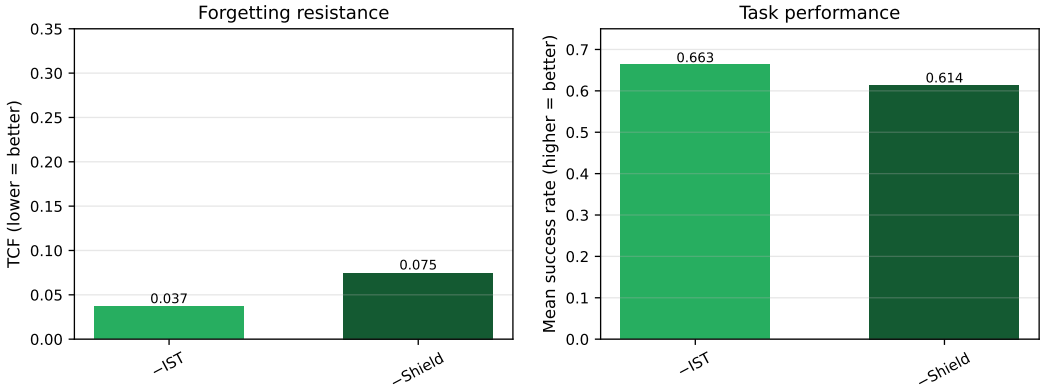

Figure 7: Ablation effects on forgetting (left) and task success (right). The admission-control gate and the shield are the two largest retention levers; among the CRAWL mechanisms, PERC carries the largest single contribution.

(+.001, $p = 0.58$); exploration (−.000, $p = 0.29$); consolidation (−.001, $p = 0.17$)—we report these nulls rather than hiding them, and keep the mechanisms for the reasons given below. All ΔTCF are computed from unrounded per-cell values, so they need not equal the difference of the rounded table entries.

The matrix confirms the channel–mechanism correspondence while being explicit about which mechanisms are *not* load-bearing at this operating point (Table 6).

Admission control is the largest lever among the mechanisms tested. Removing the K-gate inflates the library from 24 to 108 skills, collapses acquisition success from .791 to .217 and regression-probe success from .755 to .138, and drives F_{exec} from .028 to .504. The circularity budget of §7 bounds this claim: a large share of the measured ΔTCF is attributable to the simulator's same-family bloat penalty, which the gate removes by construction, so the number above should be read as an upper bound obtained under that penalty. Same-family bloat is nonetheless not merely a retention pathology but a live performance pathology that destroys the very competence the library is meant to preserve. The "growth without gating" arm (K-gate off, merging on as the only bloat control) recovers some of this but still loses 0.21 acquisition success and doubles the library, which is why merging is not part of the default configuration.

The shield and foraging guard execution regression. Removing the shield inflates F_{exec} by 4.0× (.028 → .115) and costs regression-probe success (.755 → .662); removing foraging does the same through a different door ($F_{\text{exec}} \to .110$, $\text{SR}_R \to .686$). Neither leaves a clean signature on F_{retrieve} or F_{semantic}, exactly as the channel assignment predicts.

Regeneration is the only mechanism that targets semantic rot. Removing it multiplies F_{semantic} by $14.1\times$ ($.001 \rightarrow .021$) while leaving the retrieval channel essentially untouched (F_{retrieve}: $.077 \rightarrow .082$). Its TCF-level effect is diluted precisely *because* PERC now also repairs rot (below), so the two mechanisms overlap on this channel rather than one masking the other.

The CRAWL layer is not decoration, but its three members are not equal. Removing all three costs $+.012$ TCF (

$$p = 0.001$$

), i.e. the CRAWL layer accounts for 5% of the total gap to the ungated design. Within it, **PERC dominates**: without it F_{retrieve} rises from .077 to .103 and F_{exec} from .028 to .068 — the retrieval-channel signature of the contamination loop that its error gate closes, on top of the F_{semantic} signature of ungated repair. **SPARC's** TCF effect is marginal ($+0.003$, p=0.11), and it is visible mainly as F_{semantic} drift ($.001 \rightarrow .006$) and through the IST maturation branch it feeds; unlike the other two, its benefit is measured against a baseline ("no rehearsal") that still inherits PERC, which already repairs the rot that rehearsal would have prevented. **IST is neutral at every tested capacity** ($+0.000$, p=0.90 at K=3; within noise across the full $K \in \{2, 3, 5, 10\}$ sweep, Appendix D.6): with three specialists per family there is no redundant clone to remove, so we report the null result rather than tuning it away, and we keep IST for the reasons stated in §B.9.3 (it removes a hard-coded family-name hack with a family-agnostic rule and is the only mechanism able to re-author an acquired procedure).

Two v2.1 mechanisms do not pay for themselves on TCF. Dual-timescale consolidation ($+0.002$, p=0.51) and partial-reinforcement exploration ($+0.002$, p=0.91) are indistinguishable from Full. We keep them because consolidation supplies the *mechanism-level explanation* of the scaling phase transition (§6.2, K_{slow}=8 saturation) — a claim about the growth trajectory, not about steady-state TCF — and because exploration costs nothing measurable. But we state plainly that at K=3 neither improves forgetting, and we do not claim otherwise.

Viability is a substrate, not a mechanism. Removing viability has the smallest effect ($+0.003$, p=0.94): with its consumers (shield ranking, extinction gating, clone election) still active, the composite score's marginal contribution is invisible—consistent with our design lesson (§9) that a survival score is a substrate whose value is realized through what consumes it.

An honest trade-off. Full has slightly *lower* current-task success (.666) than the CRAWL-free arms (.716). The CRAWL layer deliberately trades ≈ 2.2 pp of current-task success for $+1.8$ pp of regression-probe success and a 37% reduction in forgetting: rehearsal and gated repair spend effort on families that are not currently in demand. We consider that the right trade for a long-lived library, and we report it rather than hiding it behind the headline TCF number.

D.6 Sensitivity to the niche carrying capacity K

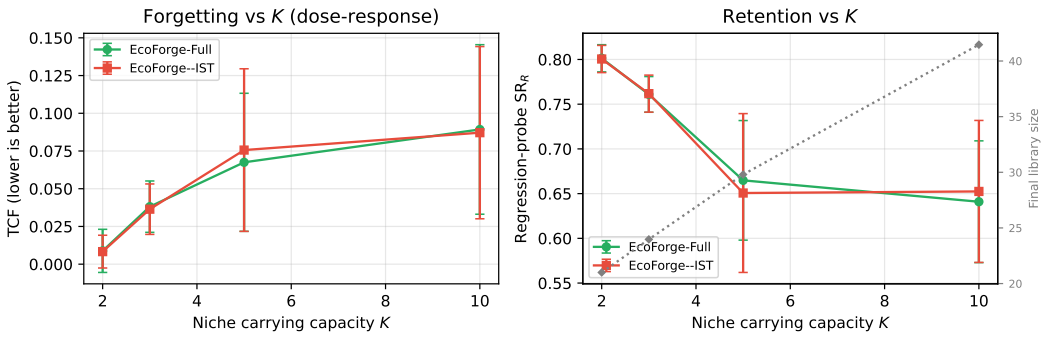

Figure 8: Niche carrying capacity sweep ($K \in \{2, 3, 5, 10\}$, 15 seeds, 5 domains, same protocol as the ablation matrix; mean ± s.d. over 75 (seed, domain) cells). Left: forgetting degrades monotonically as the gate relaxes. Right: regression-probe success and final library size; the IST-ablated arm (red) tracks Full (green) at every K.

The ablation matrix fixes K=3; this sweep varies it (Table 10, Figure 8). Three findings. **(1) Admission control is a graded lever, not a switch.** TCF degrades monotonically as the gate relaxes

Table 10: Carrying-capacity sweep for EcoForge-Full (means over 75 (seed, domain) cells). p: paired Wilcoxon versus K=3 on TCF.

K	TCF	F_{retrieve}	F_{semantic}	F_{exec}	SR	SR_R	$\lvert\mathcal{S}\rvert$	p
2	.009	.010	.001	.016	.731	.801	21.0	$< 10^{-5}$
3	.038	.076	.010	.029	.718	.761	24.0	—
5	.068	.090	.017	.096	.631	.665	29.8	$< 10^{-5}$
10	.089	.113	.034	.120	.613	.641	41.5	$< 10^{-5}$

($0.009 \to 0.038 \to 0.068 \to 0.089$; every departure from K=3 significant at $p < 10^{-5}$), and the dominant degradation channel is F_{exec} ($0.029 \to 0.120$ at K=10): same-family bloat returns exactly as the capacity constraint stops controlling it. **(2)** K=2 **dominates** K=3 **on this benchmark** (TCF 0.009 vs. 0.038, SR_R 0.801 vs. 0.761, SR 0.731 vs. 0.718): the executor's duplicate-interference penalty means any tolerated redundancy is costly on TCF-Bench. K=3 is our pre-registered conservative default—chosen for headroom on richer task mixtures, not tuned on the benchmark—and switching it post hoc after observing this sweep would itself be benchmark tuning; deployments should set K against their own interference profile. **(3) The IST conjecture is not supported.** §B.9.3 conjectured that IST's value grows with K; the sweep refutes this up to K=10 (ΔTCF of $-0.0005/-0.0017/+0.0081/-0.0021$ at K=2/3/5/10, all $p > 0.29$), so IST's justification remains the qualitative one (family-agnostic tolerance, re-authoring), and we report the null plainly. As a consistency check, the sweep's K=3 column (TCF 0.038, SR 0.718, SR_R 0.761, $\lvert\mathcal{S}\rvert$=24) reproduces the main-table EcoForge numbers ($.036/.667/.755/24$) under an independent re-run.

D.7 Simulator constants and their provenance

Eq. 2 is fully specified by the constants in Table 11. They are declared once in a single configuration module and the table below is generated mechanically from that file (`tools/dump_executor_constants.py`), so the manuscript cannot drift from the implementation. Of these, only the bloat term π_{bloat} was previously a function-local literal; it is now a centralised constant and, more importantly, a *pluggable profile* (§7), because it is the term the circularity objection targets.

Table 11: Simulator constants, auto-generated from the configuration module. π_{bloat} is shown for the default `linear` profile; the `none`/`logistic`/`power`/`empirical` profiles of §7 replace this one term and nothing else.

Symbol	Value	Meaning
b	0.42	base competence: $P(\text{success})$ with no skills retrieved
d	T_1=0.45, T_2=0.55, T_3=0.65	task difficulty by family ($\pm$0.05 jitter, $+0.00/0.05/0.10/0.02$ by phase)
β_c	0.52	boost when the canonical family skill is retrieved
π_{ov}	0.55	overlap penalty per retrieved same-family copy beyond the 4th (linear)
π_{bloat}	$\max(0,\, n_f - 12) \cdot 0.070$	library-bloat interference in the library-wide same-family count n_f
β_p	0.12	boost per related-but-not-canonical retrieved skill
π_w	0.11	penalty per conflicting retrieved skill
ξ	0.05	half-width of the uniform execution noise
q	0.82	probability a synthesized skill carries its family markers
$p_{\min}, p_{\max}$	0.03, 0.99	clamp on the execution probability
K	3	niche carrying capacity (admission control)
k	5	skills retrieved per task
$\delta_{\text{api}},\ \delta_{\text{dom}}$	0.55, 0.32	API- and domain-drift severity

D.8 Channel-weight sensitivity

Eq. 1 averages the three channels uniformly. To check that this choice is not doing hidden work, we recomputed the method ordering under four aggregations: the uniform mean; the *worst* chan-

nel, $\max(F_{\text{retrieve}}, F_{\text{semantic}}, F_{\text{exec}})$; weights proportional to each channel's cross-method variance $\text{w}_{\text{var}} = (0.61, 0.07, 0.32)$; and weights proportional to each channel's univariate regression slope onto the regression-probe drop $1 - \text{SR}_R$, $\text{w}_{\text{sr}} = (0.08, 0.65, 0.27)$.

The ordering is essentially invariant: Spearman $\rho = 0.98$ (uniform vs. max), 0.93 (uniform vs. variance-weighted) and 0.93 (uniform vs. SR-weighted), and *EcoForge is the argmin under all four rules*. The equal weighting therefore does not manufacture the result. We report this rather than merely asserting it, and we note the honest caveat that the variance- and SR-derived weights are themselves estimated on these runs—which is precisely why we do not adopt them as the primary metric.

D.9 Reproducibility details requested in review

Dual-timescale consolidation is bounded. The fast and slow weights are clamped after every update: $w_{\text{fast}} \leftarrow \text{clip}(w_{\text{fast}} \pm \Delta_{\text{fast}}, 0, 1)$ on use and $w_{\text{fast}} \leftarrow w_{\text{fast}} e^{-\lambda_{\text{fast}} \Delta t}$ when idle; $w_{\text{slow}} \leftarrow \text{clip}\big(w_{\text{slow}} + \Delta_{\text{slow}}(w_{\text{fast}} - w_{\text{slow}}), 0, 1\big)$. The constants are Δ^{+}_{fast}=0.25, Δ^{-}_{fast}=0.08, λ_{fast}=0.06, Δ_{slow}=0.03, with thresholds θ_c=0.55, θ_f=0.15, θ_{forget}=0.05 and slow-store capacity K_{slow}=8.

The slow drift is integrated during idle periods, not frozen. Eq. 5 is applied on every maintenance pass, including passes where the skill was not used: w_{fast} is first decayed by the elapsed idle time and w_{slow} then drifts toward the *decayed* w_{fast}. A consolidated skill therefore loses slow weight while dormant and can return to the forgetting-candidate class; consolidation is not an absorbing state. What *is* absorbing is only the fossilized flag, which is why the shield re-checks protected skills on every pass.

The viability factors are all normalised to $[0, 1]$**.** Eq. 3 is only meaningful if its five inputs share a scale, and they do: $\text{Usage}_{30}(s) = \min\big(1, \frac{|\{u : t-u \le 30\}|}{30}\big) \cdot \big(0.5 + 0.5 e^{-\text{idle}/150}\big)$ (a bounded rate times a bounded decay, so a busy era cannot dominate); $\text{Conflict}(s) = |\text{conflict edges}|/(|\mathcal{S}|-1)$; DepCrit is a PageRank-style score renormalised to $[0, 1]$; $\text{Gen}(s) = \min(1, H/\log 5)$ with H the entropy over served domains; and $\text{Redund}(s)$ is the mean top-3 neighbour similarity. Because $\sum_i \alpha_i = 1$ and every factor lies in $[0, 1]$, $V(s) \in [0, 1]$ at every library scale, which is what makes the single extinction threshold $V(s) < 0.12$ comparable as the library grows.

IST hyperparameters. M=3 hypermutated variants per maturation event, 2 probes per mutant, a margin of 0.03 over the incumbent, 2 consecutive rehearsal lapses to trigger, a 200-step cooldown, and a hard budget of 6 maturation events per domain run.

Seeds are fixed and listed. The main matrix, the ablation matrix and the carrying-capacity sweep use seeds 2026–2040 (fifteen); the transition, scaling and live sweeps use 2026–2028.

Reproducibility Statement

All experiments in this paper run on TCF-Bench, whose simulator, protocol, governance mechanisms and hyperparameters are specified in Appendices B–D.9 and Table 11, with the seed sets listed there. Every number in Table 1, Table 6, and the scaling/transition/K-sensitivity sweeps is regenerated by a single command per figure. An anonymized code package containing the simulator, the protocol, the governance code, the seed lists and the analysis pipeline will be released upon publication. The simulated executor has no external LLM dependency, so results are deterministic given the seeds.

Statement on the Use of AI Tools

During the preparation of this manuscript the authors used Large Language Model (LLM) assistants for the following purposes: (i) **language editing**—polishing the English expression of the title, abstract, and several paragraphs; (ii) **literature discovery**—identifying related work on catastrophic forgetting, continual learning, and agent memory; (iii) **drafting**—producing initial drafts of some explanatory paragraphs; (iv) **LaTeX formatting and consistency checking**—verifying that reported

numbers match the authors' own experimental output files; and (v) **executing the authors' predefined, seed-fixed simulation scripts** at the authors' direction.

The research content—problem formulation, mechanism design, experimental design, implementation, and analysis of results—is the authors' own. We did **not** use LLMs to generate experimental results, synthetic datasets, mathematical proofs, or implementation code. The only LLM calls that enter the results of this paper are those of the *live-executor external-validity experiment* (§8), where an LLM is deliberately used as the execution environment under study; every such call is cached, seed-fixed, and graded by a programmatic ground-truth predicate rather than by a model. All LLM-assisted text was checked by the authors for factual correctness, potential plagiarism, and consistency with our experiments. The authors take full responsibility for the final content of this paper, including all text, claims, and artifacts produced with the aid of generative AI.