MEASURING AND REPAYING SKILL DEBT IN LIFELONG LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Persistent skill libraries allow large language model agents to reuse procedural knowledge across long task streams. However, continued accumulation can reduce performance when irrelevant, conflicting, stale, or redundant skills compete under fixed retrieval and context budgets. We define this phenomenon as *skill debt*: budget-matched performance regret caused by library contents or organization that are suboptimal for the current evaluation window. We introduce *Counterfactual Skill Stewardship* (CSS), an auditable lifecycle policy for measuring and mitigating skill debt by estimating the incremental utility of making each skill available. CSS performs paired shadow executions that differ only in the availability of an audited skill and applies time-block cross-fitted estimation to infer skill-level marginal utility from sparse reward differences. An uncertainty-aware policy keeps supported skills active, freezes uncertain or historically useful skills to exclude them from routine retrieval while preserving recoverability, and retires repeatedly harmful skills subject to dependency constraints. We also develop SkillDebtBench, a trajectory-aware evaluation protocol that controls task, model, retrieval, context, and inference budgets while recording complete library evolution and maintenance costs. Across controlled program tasks and interactive agent environments, the protocol varies library scale and resource constraints, injects distinct sources of negative transfer, and evaluates recurring task distributions. This framework supports falsifiable tests of when skill accumulation creates debt, which mechanisms cause performance degradation, and whether lifecycle interventions improve long-term reward without irreversible forgetting.

1 INTRODUCTION

Large language model (LLM) agents increasingly operate over sequences of related tasks. Persistent skill libraries support lifelong adaptation by externalizing successful procedures as natural-language strategies, executable programs, or workflows that agents can retrieve and reuse without modifying the underlying model. Voyager demonstrated this paradigm with a continually expanding library of executable skills for open-ended embodied interaction(Wang et al., 2023a). LifelongAgentBench showed that accumulated experience supports transfer across interdependent tasks, but irrelevant information can compete for limited context and reduce its utility(Zheng et al., 2025). These findings motivate persistent procedural memory and raise a central question: under fixed retrieval, context, and inference budgets, does continued skill accumulation remain beneficial over long task streams?

Existing skill-augmented agents focus primarily on skill acquisition, selection, and execution. Voyager maintains an append-only library of executable skills, while recent systems optimize recursive skill evolution or jointly train skill selection, utilization, and distillation(Xia et al., 2026)(Shi et al., 2026). Complementary work treats the library as an evolving software ecosystem. SkillOps introduces skill technical debt and applies rule-based operations to maintain contracts, dependencies, validation states, and overall library health(Pu et al., 2026). A taxonomy of dynamic agent skills identifies admission, repair, merging, pruning, and governance as recurring lifecycle operations. It also reports that flat retrieval can deteriorate as libraries grow and that existing evaluations rarely record complete library trajectories or distinguish skill usage from utility(Li, 2026). The resulting problem is to determine, under controlled resource budgets, whether a library has accumulated measurable performance debt, identify the responsible skills, and select maintenance actions that improve long-term performance.

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

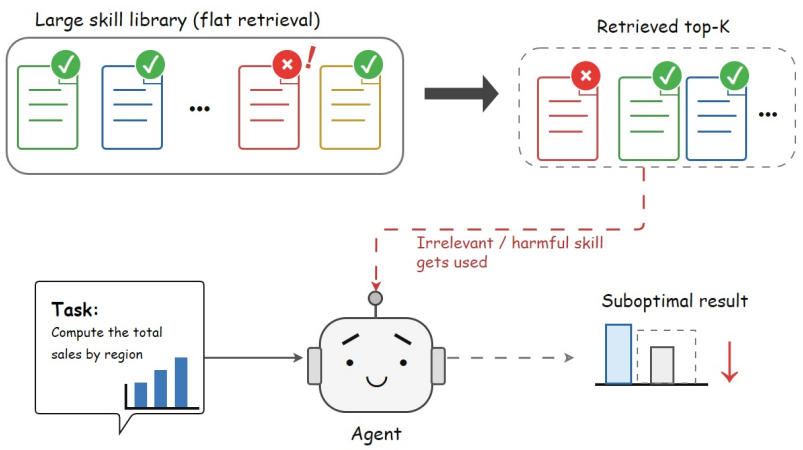


Figure 1: Skill debt caused by harmful top-K retrieval from a large skill library.

Three confounds complicate this analysis. First, larger libraries can appear harmful when each skill receives a smaller effective retrieval or context budget. Valid comparisons must therefore hold the task, model, top-K, context length, and number of inference calls fixed. Second, retrieval frequency, semantic similarity, and observed success provide correlational evidence. A broadly applicable skill can be retrieved frequently while displacing a less common but more useful procedure, and an obsolete skill can remain semantically relevant after an environment change. Detecting these failures requires interventions that compare the same task with and without a candidate skill. Third, a skill that is currently harmful can regain value when a previous task distribution recurs. Lifecycle policies must reduce current competition while preserving skills that can support future recovery. Case-based reasoning identified memory swamping and harmful reuse under uncontrolled retention(Salamó & López-Sánchez, 2011). Lifelong language agents further introduce stochastic generation, learned retrieval, executable side effects, and changes in model or environment behavior.

We study *skill debt*, defined as budget-matched regret induced by maintaining a skill library whose contents or organization are suboptimal for the current evaluation window. Given a task stream and a growing library, we measure debt as the performance difference between the deployed library and a counterfactual sublibrary under identical retrieval, context, and inference budgets. This end-to-end quantity is distinct from retrieval latency and maintenance expenditure and does not equate debt with library size. The formulation yields a falsifiable hypothesis: append-only accumulation exhibits non-monotonic utility when constrained resources create harmful interactions, and interventions should attribute any decline to retrieval competition, semantically similar but behaviorally incompatible skills, stale versions, or redundant entries. Figure 1 illustrates how flat retrieval from a large library can admit an irrelevant or harmful skill into the top-K context and thereby reduce task performance. Stable matched-budget performance and degradation that disappears under exact retrieval provide no evidence for skill debt.

We introduce *Counterfactual Skill Stewardship* (CSS), an auditable lifecycle policy for a fixed agent and repository. CSS estimates how skill availability changes task reward, using exposure and invocation frequency as auxiliary observations rather than direct measures of value. During each audit window, a fixed budget supports paired shadow executions for retrieved, stale, or high-conflict skills. Each pair replays the same task with the same random seed and identical top-K, context, and inference budgets. The ablated execution removes only the audited skill and promotes the next retrieval candidate. Sparse reward differences supervise a time-block cross-fitted estimator of skill-level marginal utility, while retrieval rank, age, version state, and local conflict features support generalization to unaudited cases. CSS maps estimated utility and uncertainty to three lifecycle states. Positively supported skills remain *active*. Uncertain, historically useful, or rare-domain skills become *frozen* and cease competing in routine top-K retrieval. Consistently harmful skills without active dependents become *retired* after repeated audits. Frozen skills receive limited revalidation following environment changes or task-distribution recurrence, which reduces current interference while preserving skills whose utility can recover under future distributions.

The *SkillDebtBench* protocol records complete task and library trajectories, skill lineage, retrieval candidates, activations, verifier outcomes, lifecycle decisions, and maintenance costs. It combines a controlled program environment that permits independent injection of equivalent, conflicting, stale, and irrelevant skills with the Database, Operating System, and Knowledge Graph environments from LifelongAgentBench and the interactive ALFWorld and WebShop settings. Factorial experiments vary library size, top-K, context budget, and inference budget. Randomized interventions isolate distinct pathways of negative transfer, and periodic distribution recurrence tests whether freezing preserves recoverability more effectively than permanent deletion. We compare CSS with append-only storage, random capacity controls, least-recently-used and frequency-based pruning, age decay, embedding deduplication, observed-success heuristics, compute-matched reranking, information-matched predictors, SkillOps, and tractable current-window or future-aware oracles. We report cumulative reward, old-task retention, recovery time, counterfactual estimation quality, retrieval costs, maintenance costs, and predefined failure conditions. These measurements separate lifecycle-policy effects from additional model calls, reduced capacity, and approximate-nearest-neighbor artifacts.

Our contributions are:

- We formulate skill debt as budget-matched counterfactual performance regret in lifelong language agents, separating library-induced degradation from inventory size, retrieval approximation, and unequal context or inference resources.
- We develop SkillDebtBench, a trajectory-aware, intervention-based protocol that measures scale reversals and decomposes negative transfer into retrieval competition, semantic conflict, stale activation, and redundancy under controlled task streams.
- We propose CSS, a model-agnostic lifecycle policy that combines paired shadow ablations, time-block cross-fitted marginal-utility estimation, and uncertainty-aware active, frozen, and retired states to distinguish observed use from incremental task value.
- We design a falsifiable evaluation across controlled and interactive environments that measures long-term reward, regression-task retention, causal calibration, and full audit costs against heuristic, maintenance-oriented, and compute-matched alternatives.

2 Related Work

2.1 Persistent Experience and Procedural Memory in LLM Agents

LLM agents externalize experience because model parameters remain fixed during deployment and context is finite. ReAct interleaves reasoning and actions within a trajectory(Yao et al., 2022b), while Reflexion stores verbal feedback across trials(Shinn et al., 2023), ExpeL extracts reusable insights from trajectories(Zhao et al., 2023), and Generative Agents organize events using relevance, recency, importance, and reflection(Park et al., 2023). Procedural systems make experience operational: Voyager retrieves code-based skills(Wang et al., 2023a), ProgPrompt produces executable plans(Singh et al., 2022), and Code as Policies generates compositional control programs(Liang et al., 2022). SkillRL jointly optimizes skill discovery, selection, and utilization(Xia et al., 2026), while Skill1 recursively evolves and distills skills(Shi et al., 2026).

Conversational-memory systems instead store facts, events, preferences, or summaries. MemoryBank uses time- and importance-dependent updating(Zhong et al., 2023), MemGPT manages hierarchical context and external storage(Packer et al., 2023), A-MEM maintains interconnected notes(Xu et al., 2025), and SECOM constructs coherent compressed segments(Pan et al., 2025). Evaluation increasingly covers persistent interaction: LoCoMo measures memory over temporally structured conversations(Maharana et al., 2024), LongMemEval separates extraction, multi-session and temporal reasoning, updates, and abstention(Wu et al., 2024), and LifelongAgentBench evaluates sequential experience accumulation while identifying competition from irrelevant experience(Zheng et al., 2025). In contrast, ALFWorld(Shridhar et al., 2020), WebShop(Yao et al., 2022a), and AgentBench(Liu et al., 2023) primarily assess interactive task completion within fixed episodes.

2.2 Retrieval under Finite Context and Resource Budgets

Persistent libraries inherit the bounded candidate-selection interface of retrieval-augmented generation. The k-nearest-neighbor language model interpolates parametric predictions with retrieved

contexts(Khandelwal et al., 2019), REALM learns a latent retriever(Guu et al., 2020), RAG combines dense retrieval with generation(Lewis et al., 2020), and RETRO integrates retrieved chunks through cross-attention(Borgeaud et al., 2021). Repository quality therefore differs from coverage: semantic similarity need not reflect behavioral utility, and additional candidates can change retrieval composition under a fixed budget. Retrieval may improve aggregate perplexity while degrading many generated tokens(Wang et al., 2023b), while long-context performance depends strongly on evidence position(Liu et al., 2024).

Online agent memories may also become redundant, contradictory, or stale. Case-base maintenance addresses analogous memory swamping and harmful reuse through selective retention and forgetting(Salamó & López-Sánchez, 2011). SkillOps models libraries through contracts, dependencies, versions, validation states, and lifecycle operations(Pu et al., 2026), while the dynamic-agent-skills taxonomy covers admission, retrieval, repair, merging, pruning, and governance, distinguishing usage from utility(Li, 2026).

2.3 Continual Learning, Interference, and Memory Management

Continual learning balances acquisition and retention under non-stationarity. Elastic Weight Consolidation penalizes changes to important parameters(Kirkpatrick et al., 2017), Synaptic Intelligence accumulates importance along optimization paths(Zenke et al., 2017), and Learning without Forgetting distills previous outputs during new-task training(Li & Hoiem, 2016). Replay methods constrain updates using stored examples: Gradient Episodic Memory prevents increases in previous-task losses(Lopez-Paz & Ranzato, 2017), A-GEM uses a reference gradient(Chaudhry et al., 2018), and small reservoir-style episodic memories remain competitive online(Chaudhry et al., 2019). DER and DER++ replay logits, optionally with labels(Buzzega et al., 2020), while iCaRL combines exemplars, representation learning, and nearest-mean classification(Rebuffi et al., 2016). Architectural approaches isolate capacity: Progressive Neural Networks add task-specific columns(Rusu et al., 2016), whereas PackNet prunes and reallocates parameters while preserving masks(Mallya & Lazebnik, 2017).

3 Method

We introduce Counterfactual Skill Stewardship (CSS), a lifecycle management procedure for lifelong language agents with evolving skill libraries. CSS preserves the language model, retriever, and execution policy. It periodically estimates the budget-matched marginal utility of each skill and assigns an uncertainty-aware state: active, frozen, or retired.

3.1 Problem Setup and Method Overview

Consider a lifelong task stream $\mathcal{D}_{1:T} = (D_1, \ldots, D_T)$, where $x_t \sim D_t$, and a time-indexed skill library $\mathcal{S}_t = \{s_1, \ldots, s_{n_t}\}$. Each skill is represented as $s_i = (c_i, p_i, \nu_i, \tau_i, z_i, \ell_i)$, where c_i specifies applicability, p_i is the workflow or executable body, ν_i records version and dependency constraints, τ_i contains creation and validation times, z_i stores provenance and interaction logs, and ℓ_i represents lineage relations. This representation manages skills as software artifacts while preserving their underlying format (Pu et al., 2026).

For a task x, the retriever returns at most K candidates, and the execution policy produces $r(x, \mathcal{S}_t; K, B, C) = r(x, \mathcal{R}_K(x, \mathcal{S}_t); B, C)$, where B and C bound the numbers of context tokens and model or reasoning calls, respectively. Counterfactual comparisons hold the task, model version, random seed, K, B, and C fixed, ensuring that resource differences are not attributed to skill quality.

At audit time u, CSS takes as input the current library $\mathcal{S}_u$, the completed task window W_u, retrieval and execution logs, verifier rewards, and an audit budget A. For each considered skill, it returns a state $g_i^{(u)} \in \{\textsc{Active}, \textsc{Frozen}, \textsc{Retired}\}$, a utility estimate $\widehat{\Delta}_i^{(u)}$, an uncertainty estimate $\widehat{\sigma}_i^{(u)}$, and a decision record. Active skills participate in routine top-K retrieval. Frozen skills retain their representations but are excluded from routine retrieval competition. Retired skills retain only their summaries, provenance, and lineage. All decisions use only tasks completed by time u. Figure 2

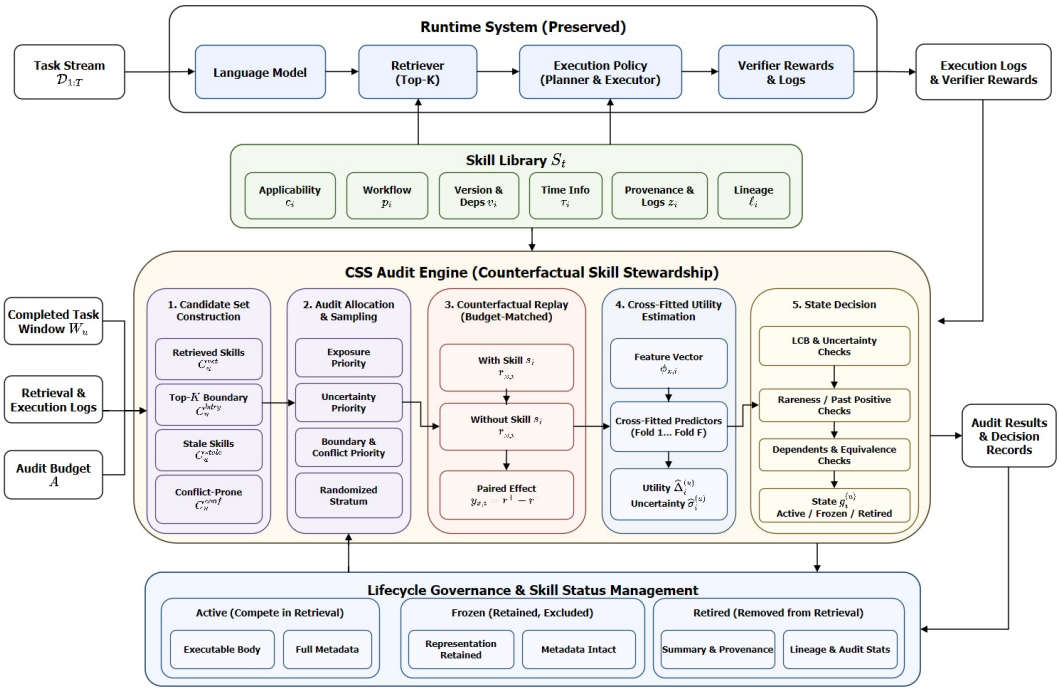


Figure 2: Overview of the CSS audit and skill lifecycle workflow.

summarizes the resulting audit and lifecycle-governance workflow while preserving the runtime system.

3.2 Budget-Matched Counterfactual Auditing

CSS audits the candidate set $\mathcal{C}_u = \mathcal{C}_u^{\text{ret}} \cup \mathcal{C}_u^{\text{bdry}} \cup \mathcal{C}_u^{\text{stale}} \cup \mathcal{C}_u^{\text{conf}}$, comprising retrieved, top-K boundary, stale, and conflict-prone skills. Boundary candidates account for retrieval competition because removing a selected skill can admit the next-ranked candidate. CSS selects task–skill pairs $\mathcal{P}_u \subseteq W_u \times \mathcal{C}_u$ subject to $|\mathcal{P}_u| \leq A$. The allocation prioritizes exposure, uncertainty, boundary proximity, and version or semantic conflicts while reserving a nonzero randomized stratum. Each eligible pair in this stratum has a logged inclusion probability $\pi_{x,i}^{(u)} > 0$, which supports propensity-aware estimation and calibration.

For each $(x, i) \in \mathcal{P}_u$, CSS replays x with and without s_i under identical random seeds and budgets:

$$r_{x,i}^{+} = r(x, \mathcal{S}_u; K, B, C), \qquad r_{x,i}^{-} = r(x, \mathcal{S}_u \setminus \{s_i\}; K, B, C).$$

The paired effect $y_{x,i} = r_{x,i}^{+} - r_{x,i}^{-}$ measures the total effect of exposing s_i, including the resulting changes in retrieval, planning, and execution. Common random numbers reduce variance without requiring additive skill effects.

3.3 Cross-Fitted Utility Estimation

For each eligible pair, CSS constructs $\boldsymbol{\phi}_{x,i} = [\boldsymbol{q}_x, \rho_{x,i}, e_{x,i}, a_i, v_i, d_{x,i}]$, where $\boldsymbol{q}_x$ contains pre-audit task features, $\rho_{x,i}$ is the retrieval rank, $e_{x,i}$ describes exposure, a_i is the skill age, v_i represents version validity, and $d_{x,i}$ contains sparse similarity and dependency features. The feature vector excludes future outcomes and post-audit decisions.

CSS partitions W_u into F ordered temporal blocks. For each block f, a model trained outside that block predicts $\widehat{y}_{x,i} = \widehat{\mu}^{(-f(x))}(\boldsymbol{\phi}_{x,i})$ for $x \in W_u^{(f(x))}$. The prediction model is a regularized linear model or a small tree-based regressor and uses neither language-model parameters nor hidden states.

For the eligible task set $\mathcal{W}_{u,i}$, the utility estimate is

$$\widehat{\Delta}_i^{(u)} = \frac{1}{|\mathcal{W}_{u,i}|} \sum_{x \in \mathcal{W}_{u,i}} \widetilde{y}_{x,i}, \qquad \widetilde{y}_{x,i} = \begin{cases} y_{x,i}, & (x,i) \in \mathcal{P}_u, \\ \widehat{y}_{x,i}, & \text{otherwise.} \end{cases} \tag{1}$$

For unequal audit probabilities, CSS applies inverse-probability or stabilized weights derived from $\pi_{x,i}^{(u)}$. Randomized observations are retained separately to evaluate extrapolation.

The uncertainty estimate combines variability across temporal folds, paired observations, and estimator refits. CSS defines the lower confidence bound as $\mathrm{LCB}_i^{(u)} = \widehat{\Delta}_i^{(u)} - \beta \widehat{\sigma}_i^{(u)}$, where $\beta \geq 0$, and the uncertainty predicate as

$$U_i^{(u)} = \mathbb{1}\left[\widehat{\sigma}_i^{(u)} > \sigma_{\max} \ \vee \ n_i^{\mathrm{dir}} < n_{\min}\right], \tag{2}$$

where n_i^{dir} is the number of direct paired observations. A high-uncertainty estimate permits freezing but precludes retirement.

3.4 Lifecycle Governance

A skill remains active when

$$\mathrm{LCB}_i^{(u)} > 0. \tag{3}$$

CSS otherwise freezes the skill when its estimate is uncertain, it serves a protected rare task family, or an independent earlier audit found positive utility. Retirement requires repeated evidence of harm under sufficient observation. Let u^- denote the preceding independent audit, and let $h_i^{(u)}$ indicate whether the skill has active dependents. Retirement requires

$$\begin{aligned} \mathrm{LCB}_i^{(u)} < -\epsilon, \quad \mathrm{LCB}_i^{(u^-)} < -\epsilon, \\ U_i^{(u)} = U_i^{(u^-)} = 0, \qquad h_i^{(u)} = 0, \end{aligned} \tag{4}$$

where $\epsilon \geq 0$. The complete state transition is

$$g_i^{(u)} = \begin{cases} \textsc{Active}, & \mathrm{LCB}_i^{(u)} > 0, \\ \textsc{Frozen}, & U_i^{(u)} = 1 \vee \mathrm{Rare}_i^{(u)} = 1 \vee \mathrm{PastPos}_i^{(u)} = 1, \\ \textsc{Retired}, & \text{Equation (4) holds}, \\ \textsc{Frozen}, & \text{otherwise.} \end{cases} \tag{5}$$

These predicates use only information available by time u. Retirement removes the executable body from retrieval while preserving its summary, provenance, audit statistics, and lineage.

For behaviorally equivalent skills, identified by matching verifier outcomes and overlapping applicability, CSS keeps the skill with the larger lower confidence bound active and freezes the other:

$$g_j^{(u)} = \textsc{Frozen} \quad \text{if} \quad s_i \equiv s_j \wedge \mathrm{LCB}_i^{(u)} \geq \mathrm{LCB}_j^{(u)}.$$

A skill with active dependents cannot be retired. When dependency metadata are missing, CSS conservatively sets $h_i^{(u)} = 1$ for executable skills. Frozen skills enter a small randomized revalidation stratum after environment changes, task-family recurrence, or restored dependency validity. They reactivate only after satisfying Equation (3). The Appendix specifies the audit objective, procedure, complexity, and deployment requirements.

4 Experiments

4.1 Evaluation Protocol

We evaluate whether skill accumulation creates performance debt under fixed budgets and whether Counterfactual Skill Stewardship (CSS) reduces it while preserving recurrently useful skills. The benchmark comprises a synthetic program environment; the Database, Operating System, and

Table 1: Lifecycle performance under matched retrieval, context, and model-call budgets. Entries are mean normalized cumulative reward with 95% hierarchical-bootstrap confidence intervals. Ret. is old-task retention after recurrence; lower D_{perf} is better. Bold and underlined entries identify the best and second-best deployable methods.

Method	Synthetic	Lifelong AgentBench	ALFWorld	WebShop	Ret. (%)	D_{perf}
No memory	55.8 ± 1.5	48.9 ± 1.8	46.7 ± 2.1	41.5 ± 1.9	52.6 ± 2.4	18.7
Append-only	63.4 ± 1.3	58.1 ± 1.5	60.8 ± 1.8	53.7 ± 1.7	68.2 ± 2.0	12.4
LRU	66.0 ± 1.2	60.2 ± 1.5	62.1 ± 1.7	55.0 ± 1.6	65.1 ± 2.2	10.8
Frequency pruning	66.7 ± 1.1	61.0 ± 1.4	62.6 ± 1.6	55.8 ± 1.6	66.4 ± 2.1	10.2
Embedding deduplication	68.1 ± 1.1	61.6 ± 1.4	63.3 ± 1.6	56.5 ± 1.5	69.0 ± 2.0	9.5
Success-rate heuristic	69.0 ± 1.0	62.8 ± 1.3	64.5 ± 1.5	57.3 ± 1.5	70.1 ± 1.9	8.7
Compute-matched reranking	69.4 ± 1.0	63.1 ± 1.3	65.0 ± 1.5	58.4 ± 1.4	70.4 ± 1.9	8.3
SkillOps	$\underline{71.8 \pm 0.9}$	$\underline{65.4 \pm 1.2}$	$\underline{67.9 \pm 1.4}$	$\mathbf{60.7 \pm 1.4}$	$\underline{74.8 \pm 1.7}$	$\underline{6.4}$
CSS	$\mathbf{75.2 \pm 0.8}$	$\mathbf{68.6 \pm 1.1}$	$\mathbf{70.1 \pm 1.3}$	$\underline{60.4 \pm 1.4}$	$\mathbf{81.3 \pm 1.5}$	$\mathbf{4.1}$
Oracle current-window	79.7 ± 0.7	72.6 ± 1.0	73.4 ± 1.2	64.1 ± 1.3	84.5 ± 1.4	0.0
Oracle future-aware	82.1 ± 0.6	75.0 ± 0.9	75.2 ± 1.1	66.3 ± 1.2	89.2 ± 1.2	-2.6

Table 2: Randomized skill injections at fixed library size. Δ Reward is relative to an uncorrupted library; Displace is the fraction of useful skills pushed below top-K; Wrong Act. is incorrect skill activation.

Injected skill type	Δ Reward	Displace (%)	Wrong Act. (%)	Version violation (%)	Mediated loss (%)
Exact duplicates	-1.1 ± 0.5	6.8 ± 1.0	1.5 ± 0.4	0.2 ± 0.1	54 ± 7
Irrelevant	-2.0 ± 0.6	9.7 ± 1.2	2.3 ± 0.5	0.3 ± 0.1	61 ± 6
Semantic conflict	-6.7 ± 0.8	18.4 ± 1.5	12.9 ± 1.1	0.5 ± 0.2	78 ± 5
Stale version	-7.4 ± 0.9	14.1 ± 1.4	10.8 ± 1.0	16.7 ± 1.3	82 ± 4
Useful, non-conflicting	$+4.8 \pm 0.7$	2.6 ± 0.6	0.8 ± 0.3	0.2 ± 0.1	–

Knowledge Graph environments from LifelongAgentBench; ALFWorld; and WebShop. The synthetic environment injects useful, irrelevant, duplicated, conflicting, and stale skills through versioned functions, constrained arguments, and deterministic verifiers. Other environments assess natural-language policies and executable procedures. Streams contain $2{,}000$ tasks, and long-horizon tests contain $10{,}000$ tasks. They include stationary repetition, gradual drift, abrupt version changes, periodic recurrence, rare task families, and mixtures of old and new tasks. We partition task templates, parameterized variants, and skill-generation templates between development and test sets. This design separates memory growth from distributional change and tests whether governance remains effective when useful procedures reappear after extended absences.

We compare CSS with no memory, append-only storage, random freezing, least-recently-used (LRU) freezing, frequency pruning, age decay, embedding deduplication, historical-success filtering, SkillOps, and interface-compatible SkillRL or Skill1 configurations. Compute controls allocate the CSS audit budget to larger contexts, reranking, or test-time sampling. An information-matched predictor uses the same logs and metadata but estimates observed success rather than counterfactual utility. A current-window Oracle uses evaluation-window leave-one-out effects, while a future-aware Oracle observes the complete stream.

All comparisons use the same base model, observations, logs, retriever, top-K candidate count K, context budget B, and inference-call limit C. Compute-matched methods receive the same additional calls as CSS. Primary metrics are normalized cumulative reward, task success, old-task retention, recurrence recovery time, and performance debt D_{perf}. We also measure harmful-skill activation, retrieval displacement, version violations, execution failures, and counterfactual-estimation

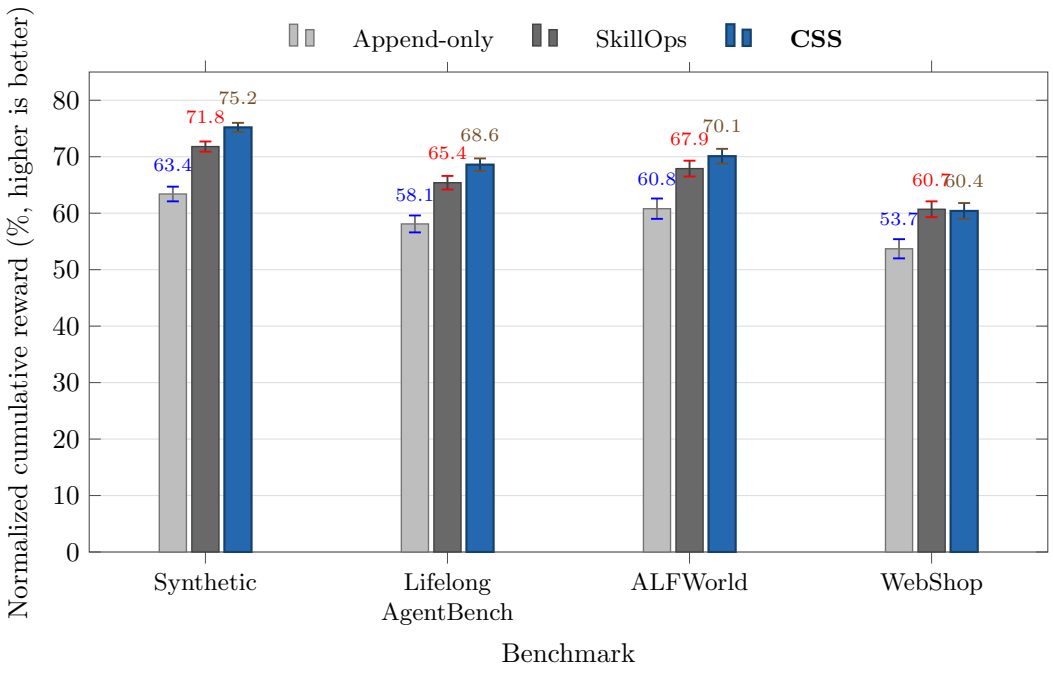


Figure 3: Lifecycle performance under matched retrieval, context, and model-call budgets. Bars show mean normalized cumulative reward with 95% hierarchical-bootstrap confidence intervals.

quality. We report means with 95% hierarchical-bootstrap confidence intervals. Comparisons use paired bootstrap differences, Cliff's delta, and Holm correction.

4.2 Lifecycle Performance

Table 1 distinguishes persistent memory benefits from unrestricted retrieval competition, while Figure 3 shows performance under matched budgets. Append-only storage outperforms no memory, and selective control of the active library outperforms append-only storage. CSS achieves the highest deployable reward in the synthetic environment, LifelongAgentBench, and ALFWorld. It exceeds append-only storage by 11.8, 10.5, and 9.3 points and SkillOps by 3.4, 3.2, and 2.2 points, respectively. Paired intervals exceed the preregistered one-point threshold for the synthetic and LifelongAgentBench comparisons, but not ALFWorld. CSS remains 4.5, 4.0, and 3.3 points below the current-window Oracle, indicating that audit-limited first-order effects do not recover the optimal subset.

On WebShop, SkillOps reaches 60.7, compared with 60.4 for CSS, with less maintenance computation. WebShop's sparse, delayed rewards complicate attribution to individual skills within short audit windows. Counterfactual estimates are more reliable with executable verification. Under version drift, CSS gains more over the information-matched observational predictor, while random-score and permuted-ID variants approach random freezing.

Freezing improves recurrence recovery. CSS retains 81.3% of old-task performance, compared with 65.1% for LRU and 74.8% for SkillOps. Its median recovery interval is 31 tasks, compared with 94 after permanent deletion. This benefit concentrates in recurrent and long-tail streams.

4.3 Scale Reversal and Negative-Transfer Mechanisms

We vary library size while holding useful-skill coverage, tasks, model, retriever, K, B, and C fixed. Append-only reward increases as useful-skill coverage expands, plateaus once the useful set is represented, and decreases as entries compete for finite retrieval and context capacity. A segmented model

fitted on development streams identifies held-out breakpoints near $N = 390$ in the synthetic environment and $N = 610$ in LifelongAgentBench. Append-only storage has negative post-breakpoint slopes. CSS remains approximately flat by excluding skills with low lower-confidence-bound utility. Near-exhaustive retrieval removes the negative slope for small synthetic libraries, showing that reversal depends on retrieval budget rather than a universal capacity limit.

Randomized injections identify scale-reversal mechanisms. Table 2 shows that irrelevant skills reduce reward by 2.0 points, while semantic conflicts and stale versions reduce it by 6.7 and 7.4 points. Conflicting skills increase wrong-condition activation. Stale skills produce a 16.7% version-violation rate. Retrieval displacement and incorrect execution mediate 78% of loss from conflicts. These factors and version violations mediate 82% of loss from stale skills. Exact duplicates reduce reward by 1.1 points by diluting retrieval probability but rarely altering execution. Useful, non-conflicting additions increase reward by 4.8 points.

Increasing K or B delays but does not consistently remove the breakpoint. Larger candidate sets recover displaced skills while exposing the policy to incompatible instructions.

4.4 Counterfactual Validity and Cost

On small libraries permitting exhaustive paired leave-one-skill-out replay, CSS estimates correlate with true window-level effects at 0.71 Spearman. Corresponding correlations are 0.28 for invocation frequency, 0.34 for historical success, and 0.39 for the information-matched observational predictor. CSS attains an AUROC of 0.84 for harmful-skill detection, and its nominal 95% intervals achieve 92.7% empirical coverage. The highest-risk decile contains 73% of harmful skills, compared with 41% under frequency pruning. This ranking reduces top-m selection regret by 36%. Differences narrow on stationary streams, where past usage predicts future relevance, and with small audit budgets, where uncertainty dominates decisions.

The default CSS configuration adds 124 shadow executions per $1{,}000$ tasks and increases initial token use by 8.6%. By the end of a $10{,}000$-task stream, it reduces active-skill tokens by 43%, reranking tokens by 27%, and P95 retrieval latency from 61 to 37 ms relative to append-only storage. It breaks even after approximately $3{,}400$ tasks in the synthetic and LifelongAgentBench settings but not the shortest WebShop runs. CSS benefits most when verification is reliable, harmful effects persist, and streams amortize replay costs. The results apply to matched finite budgets and first-order skill interventions. They do not establish a universal skill-capacity bound or globally optimal governance policy.

5 Conclusion

This paper formulates skill debt as budget-matched counterfactual regret, introduces SkillDebtBench for intervention-based lifecycle evaluation, and proposes Counterfactual Skill Stewardship (CSS) to estimate skill utility and govern active, frozen, and retired states. Experiments across controlled, lifelong, and interactive environments isolate retrieval competition, semantic conflict, stale activation, and redundancy while controlling retrieval, context, inference, and maintenance budgets. The results establish that CSS accurately attributes library-induced degradation, improves cumulative reward and task retention, and preserves recovery under recurring distributions, validating counterfactual lifecycle governance as an effective approach to sustaining long-term skill-library utility.

Limitations

Our framework focuses primarily on skill-level marginal utility, while interactions among multiple skills may not be fully captured by first-order counterfactual analysis. The lifecycle decisions also depend on the quality and timeliness of available reward signals, which may be imperfect in complex agentic settings. Moreover, additional experiments across broader task distributions, agent architectures, and longer deployment horizons could further clarify the generality of skill debt and the robustness of counterfactual stewardship under evolving real-world conditions.

GENERATIVE AI USAGE STATEMENT

In preparing this manuscript, the authors made limited use of AI-assisted writing tools. These tools were applied exclusively for language-level improvements, such as refining sentence structure, correcting grammatical issues, and improving the overall readability of the text. All research content—including the formulation of the skill debt framework, the design of measurement and repayment mechanisms, the experimental setup, and the interpretation of results—was conceived and carried out solely by the human authors. Generative AI was not used to produce any scientific claims, generate experimental data, or contribute to the theoretical analysis. All AI-assisted edits were reviewed and approved by the authors prior to inclusion. The authors bear full responsibility for the accuracy, originality, and integrity of this work.

REFERENCES

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*, 2021. URL https://arxiv.org/abs/2112.04426.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: A strong, simple baseline. *arXiv preprint arXiv:2004.07211*, 2020. URL https://arxiv.org/abs/2004.07211.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018. URL https://arxiv.org/abs/1812.00420.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019. URL https://arxiv.org/abs/1902.10486.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020. URL https://arxiv.org/abs/2002.08909.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019. URL https://arxiv.org/abs/1911.00172.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017. URL https://pubmed.ncbi.nlm.nih.gov/28292907/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020. URL https://arxiv.org/abs/2005.11401.

Yubo Li. Dynamic agent skills: A lifecycle survey and taxonomy of evolving skill libraries. *Transactions on Machine Learning Research*, 2026. URL https://arxiv.org/abs/2607.10113.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *arXiv preprint arXiv:1606.09282*, 2016. URL https://arxiv.org/abs/1606.09282.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022. URL https://arxiv.org/abs/2209.07753.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.tacl-1.9/.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023. URL https://arxiv.org/abs/2308.03688.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *arXiv preprint arXiv:1706.08840*, 2017. URL https://arxiv.org/abs/1706.08840.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024. URL https://arxiv.org/abs/2402.17753.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. *arXiv preprint arXiv:1711.05769*, 2017. URL https://arxiv.org/abs/1711.05769.

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

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. Secom: On memory construction and retrieval for personalized conversational agents. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xKDZAW0He3.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023. URL https://arxiv.org/abs/2304.03442.

Hongji Pu, Xinyuan Song, and Liang Zhao. Skillops: Managing llm agent skill libraries as self-maintaining software ecosystems. *arXiv preprint arXiv:2605.13716*, 2026. URL https://arxiv.org/abs/2605.13716.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *arXiv preprint arXiv:1611.07725*, 2016. URL https://arxiv.org/abs/1611.07725.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. URL https://arxiv.org/abs/1606.04671.

Maria Salamó and Maite López-Sánchez. Adaptive case-based reasoning using retention and forgetting strategies. *Knowledge-Based Systems*, 2011. URL https://dblp.org/rec/journals/kbs/SalamoL11.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi GU, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *arXiv preprint arXiv:2605.06130*, 2026. URL https://arxiv.org/abs/2605.06130.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020. URL https://arxiv.org/abs/2010.03768.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022. URL https://arxiv.org/abs/2209.11302.

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

Shufan Wang, Yixiao Song, Andrew Drozdov, Aparna Garimella, Varun Manjunatha, and Mohit Iyyer. Knn-lm does not improve open-ended text generation. *arXiv preprint arXiv:2305.14625*, 2023b. URL https://arxiv.org/abs/2305.14625.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024. URL https://arxiv.org/abs/2410.10813.

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

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025. URL https://arxiv.org/abs/2502.12110.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *arXiv preprint arXiv:2207.01206*, 2022a. URL https://arxiv.org/abs/2207.01206.

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

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *arXiv preprint arXiv:1703.04200*, 2017. URL https://arxiv.org/abs/1703.04200.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2023. URL https://arxiv.org/abs/2308.10144.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating llm agents as lifelong learners. *arXiv preprint arXiv:2505.11942*, 2025. URL https://arxiv.org/abs/2505.11942.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023. URL https://arxiv.org/abs/2305.10250.

APPENDIX

A END-TO-END COUNTERFACTUAL AUDIT AND LIFECYCLE TRANSITION

This section specifies the information flow from budget-matched shadow executions to lifecycle-state assignments. The audit runs at time u using the completed task window W_u, the current skill library $\mathcal{S}_u$, and records available by time u. The language model, retriever, and execution policy remain fixed during the audit.

A.1 Audit Units and Temporally Cross-Fitted Predictions

The audit operates on eligible task–skill pairs (x, i), where $x \in W_u$ and $s_i \in \mathcal{C}_u$. For each considered skill s_i, define

$$\mathcal{W}_{u,i} = \{x \in W_u : (x, i) \text{ is eligible for estimation at audit time } u\}.$$

The selected set $\mathcal{P}_u \subseteq W_u \times \mathcal{C}_u$ satisfies $|\mathcal{P}_u| \leq A$. For each selected pair, both executions use the same task, model version, random seed, and budgets (K, B, C). The paired effect

$$y_{x,i} = r(x, \mathcal{S}_u; K, B, C) - r(x, \mathcal{S}_u \setminus \{s_i\}; K, B, C)$$

therefore measures the reward difference induced by the availability of s_i, including changes to the retrieved top-K set. The ablated execution reruns retrieval over $\mathcal{S}_u \setminus \{s_i\}$, ensuring that the context is reconstructed without s_i.

Let

$$W_u = \bigcup_{f=1}^{F} W_u^{(f)}, \qquad W_u^{(f)} \cap W_u^{(f')} = \varnothing \quad \text{for } f \neq f',$$

denote the ordered temporal partition, and let $f(x) \in \{1, \ldots, F\}$ denote the unique block containing x. For each block f, the predictor $\widehat{\mu}^{(-f)}$ is fitted on directly observed pairs whose tasks lie outside $W_u^{(f)}$:

$$\mathcal{T}_u^{(-f)} = \left\{ \left(\boldsymbol{\phi}_{x,i}, y_{x,i}\right) : (x, i) \in \mathcal{P}_u,\ x \notin W_u^{(f)} \right\}.$$

The out-of-block prediction is

$$\widehat{y}_{x,i} = \widehat{\mu}^{(-f(x))}\left(\boldsymbol{\phi}_{x,i}\right).$$

A paired effect provides the observed value for its task–skill pair, but the predictor for that pair's temporal block is fitted without observations from the same block. This ordering prevents within-block outcome leakage and supports imputation for unaudited eligible pairs.

For each $x \in \mathcal{W}_{u,i}$, define the completed effect

$$\widetilde{y}_{x,i} = \mathbb{1}[(x, i) \in \mathcal{P}_u]\, y_{x,i} + \mathbb{1}[(x, i) \notin \mathcal{P}_u]\, \widehat{y}_{x,i}.$$

The skill-level estimate is

$$\widehat{\Delta}_i^{(u)} = \frac{1}{|\mathcal{W}_{u,i}|} \sum_{x \in \mathcal{W}_{u,i}} \widetilde{y}_{x,i}. \tag{6}$$

When audit inclusion probabilities are unequal, the estimator applies the corresponding inverse-probability or stabilized weights derived from $\pi_{x,i}^{(u)}$. The audit record identifies the randomized audit stratum separately from deterministically prioritized pairs.

A.2 Decision Ordering and Library-State Update

For each considered skill, the uncertainty estimate $\widehat{\sigma}_i^{(u)}$ is computed from the temporal-fold variability, available paired observations, and estimator refits described in the main method. Define the lower confidence bound and direct-observation count as

$$\mathrm{LCB}_i^{(u)} = \widehat{\Delta}_i^{(u)} - \beta \widehat{\sigma}_i^{(u)}, \qquad n_i^{\mathrm{dir}} = \sum_{x \in \mathcal{W}_{u,i}} \mathbb{1}[(x, i) \in \mathcal{P}_u].$$

The procedure evaluates $U_i^{(u)}$, $\mathrm{Rare}_i^{(u)}$, and $\mathrm{PastPos}_i^{(u)}$ before retirement. Positive current support receives the highest priority. Uncertainty, protected rarity, or independently established past positive utility blocks retirement. The procedure considers repeated harmful evidence only when the current and preceding independent audits both contain sufficient observations and the skill has no active dependents.

After applying the per-skill state rule, behavioral-equivalence checks resolve redundant active competitors. If $s_i \equiv s_j$, the skill with the larger lower confidence bound remains eligible for the active state, and the other skill is frozen. Dependency checks use a conservative default: when dependency metadata for an executable skill are missing, $h_i^{(u)}$ is set to one, blocking retirement.

The procedure terminates after processing all selected shadow executions, temporal blocks, considered skills, and behavioral-equivalence pairs. During routine inference after the audit, top-K retrieval includes only skills assigned ACTIVE. Frozen skills remain outside routine retrieval until a stated revalidation trigger places them in a randomized audit stratum. Retired skills no longer expose an executable body.

B Environment Details

B.1 Datasets and Task Settings

We evaluate on SkillDebtBench, which combines a controlled synthetic program environment, the Database, Operating System, and Knowledge Graph environments from LifelongAgentBench, and the interactive ALFWorld and WebShop benchmarks. The synthetic environment contains versioned function-execution tasks with constrained arguments and deterministic verifiers. It supports independent injection of useful, irrelevant, duplicated, semantically conflicting, and stale skills. LifelongAgentBench evaluates experience reuse across sequential database operations, shell-based operating-system tasks, and knowledge-graph queries. ALFWorld requires agents to complete household tasks through text-based observations and actions. WebShop evaluates goal-conditioned product search and selection through interactive web actions.

Each standard task stream contains 2,000 episodes, and each long-horizon stress-test stream contains 10,000 episodes. We construct streams covering stationary repetition, gradual distribution drift, abrupt environment or interface changes, periodic recurrence, rare task families, and mixtures of previously observed and newly introduced tasks. Open-model experiments use ten independently generated streams per configuration. Closed-model validation uses five streams per configuration and is reported separately.

We jointly split task templates, parameterized task instances, and skill-generation templates to prevent structurally equivalent tasks or skills from appearing in both development and test streams. We use development streams to select lifecycle thresholds, audit frequency, retrieval parameters, and estimator regularization. All reported results are computed on held-out streams without further tuning. For public interactive benchmarks, we retain the official development and test partitions and use development episodes only for configuration selection.

Tasks use a ReAct-style interaction template containing the task instruction, current observation, retrieved skills, and valid action interface. Retrieved skills are ordered by retrieval score and delimited with explicit `<skill>` tags. The prompt requests an internal reasoning trace followed by exactly one executable action. Synthetic and LifelongAgentBench episodes terminate when the agent emits a final answer or a verifier-accepted action sequence, or when it reaches the environment-specific interaction limit. We allow at most 20 actions for synthetic and LifelongAgentBench tasks, 50 actions for ALFWorld, and 30 actions for WebShop.

Test tasks are excluded from skill construction, hyperparameter selection, and prompt development. During a test stream, skills derived from previously completed episodes in that stream are available under the lifelong-learning setting, but they cannot access future tasks or verifier outputs. Counterfactual audits use only episodes completed before the corresponding audit time. Model prompts, retrieval metadata, and utility-estimation features exclude task identifiers, gold answers, hidden environment states, and future distribution labels.

B.2 Preprocessing and Evaluation

We canonicalize textual answers through Unicode normalization, lowercasing of case-insensitive fields, removal of leading and trailing whitespace, collapse of repeated whitespace, and removal of non-semantic terminal punctuation. Numeric answers are parsed using arbitrary-precision decimal arithmetic. Before comparison, we normalize commas, redundant leading zeros, and equivalent integer or floating-point representations. For structured outputs, we parse JSON objects and serialize them with sorted keys, compare SQL results after schema-aware normalization, and sort unordered answer sets before matching. Multiple reference answers are treated disjunctively, and an output is correct if it matches any valid reference. Malformed actions, execution exceptions, invalid API calls, and interaction-limit violations receive zero reward.

Synthetic tasks use deterministic unit tests over returned values and execution side effects. Database tasks are evaluated by denotation equivalence of query results, Operating System tasks by post-execution filesystem and process-state checks, and Knowledge Graph tasks by exact match over normalized entity or relation sets. ALFWorld uses the benchmark-provided goal-completion reward, and WebShop uses its official task-success reward. We map environment rewards to $[0, 1]$ before aggregation.

The primary metric is normalized cumulative reward,

$$100 \times \frac{\sum_{t=1}^{T} r_t}{T},$$

where $r_t \in [0, 1]$ is the verifier reward for episode t. We also report task success rate, old-task retention after distribution recurrence, median recovery time, and performance debt $\mathcal{D}_{\text{perf}}$. Retention is the post-recurrence reward on previously observed task families divided by their pre-shift reward. Recovery time is the number of episodes required to regain 95% of the pre-shift moving-average reward. Performance debt is the budget-matched reward gap between the deployed library and the current-window counterfactual oracle.

Mechanism-level evaluation measures useful-skill displacement below the top-K boundary, harmful-skill activation, stale-version violations, invalid executions, and the fraction of reward loss mediated by retrieval or execution failures. We evaluate counterfactual estimation using Spearman correlation with exhaustive leave-one-skill-out effects, AUROC for harmful-skill detection, confidence-interval coverage, risk concentration in the highest-ranked decile, and top-m selection regret.

We first aggregate results within each task stream and then across independently generated streams. We report stream-level means with 95% hierarchical-bootstrap confidence intervals based on 10,000 bootstrap resamples. Pairwise comparisons use paired bootstrap differences and Cliff's delta, with Holm correction within each benchmark family. All stochastic generation and retrieval experiments use matched random seeds across compared methods. Unless otherwise specified, deployment trajectories use greedy decoding. Experiments with test-time sampling use five seeds and report the mean and standard deviation.

Baselines include no memory, append-only storage, random freezing, least-recently-used freezing, frequency pruning, age decay, embedding deduplication, historical-success filtering, information-matched observational prediction, SkillOps, and interface-compatible SkillRL and Skill1 variants. Compute controls allocate the CSS audit budget to larger contexts, additional reranking, or additional test-time samples. Random-score and permuted-skill-identifier controls preserve active-library size and maintenance timing while removing informative utility estimates. We include a current-window oracle and a future-aware oracle as non-deployable upper bounds.

B.3 Implementation and Hardware

The primary open-model experiments use `Qwen2.5-7B-Instruct` as the shared planner and executor. The model is hosted locally with `vLLM` in bfloat16 precision and remains frozen throughout each task stream. Closed-model validation uses `gpt-4o-mini-2024-07-18` through the provider API with temperature zero. Ordinary execution, skill-conditioned execution, and counterfactual shadow replay use the same model instance and prompt format. We do not train or fine-tune a language model during the experiments.

Natural-language skills contain an applicability description, procedural workflow, version and dependency metadata, provenance, validation timestamps, and lineage links. Executable skills additionally contain a sandboxed Python or shell body. Newly acquired skills are generated from successful trajectories and validated against the originating task and two held-out parameterizations. A skill is admitted only when it passes all deterministic checks. Retired skills retain their summary, metadata, provenance, lineage, and audit statistics, while their executable bodies are excluded from retrieval.

We embed task descriptions and skill applicability fields using `BAAI/bge-base-en-v1.5`. Embeddings are ℓ_2-normalized, and retrieval uses cosine similarity with a FAISS HNSW index configured with `M=32`, `efConstruction=200`, and `efSearch=128`. The default retriever returns

$K = 8$ active skills. Scale experiments evaluate $K \in \{2, 4, 8, 16\}$ and library sizes from 50 to 2,000 skills. The total skill context is capped at 4,096 tokens, the complete model context at 16,384 tokens, and each model response at 512 generated tokens. When an audited skill is removed, the next-ranked eligible candidate is promoted so that paired executions preserve the same top-K and context budgets.

CSS performs an audit every 250 completed tasks using a rolling window of 1,000 tasks. The default audit allocation contains 31 paired task–skill comparisons per audit, corresponding to 124 additional shadow executions per 1,000 tasks. Thirty percent of audit pairs are sampled from a randomized exploration stratum with logged inclusion probabilities. The remaining budget is allocated according to skill exposure, distance from the top-K boundary, estimator uncertainty, age, version mismatch, and semantic-conflict scores. Each paired replay uses common random numbers and identical task observations, generation settings, action limits, retrieval budgets, and verifier configurations.

The marginal-utility estimator uses five ordered temporal folds and a ridge regressor over standardized task, retrieval-rank, exposure, age, version, similarity, dependency, and conflict features. We select the ridge coefficient from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$ on development streams. Audited observations are weighted by stabilized inverse inclusion probabilities clipped to $[0.1, 10]$. Uncertainty combines between-fold variation, the paired-effect standard error, and variability across 200 estimator refits. We use a one-sided 95% lower confidence bound with $\beta = 1.645$. A skill is considered insufficiently observed if it has fewer than eight direct paired audits or an estimated standard error above 0.10.

Skills with a positive lower confidence bound remain active. Skills are frozen when they have high uncertainty, protected rare-task status, or a previously positive independent audit. Retirement requires lower confidence bounds below -0.02 in two consecutive independent audits, at least eight direct paired observations in each audit window, and no active dependents. Frozen skills receive a 5% randomized revalidation allocation after an environment version change or detected task-family recurrence. Skills are considered behaviorally equivalent when the cosine similarity between their applicability embeddings exceeds 0.95 and their verifier outcomes agree on at least 95% of shared validation tasks. Among equivalent skills, only the skill with the larger lower confidence bound remains active.

All methods use the same task stream, base model, retriever, retrieval depth, context limit, action limit, and deployment inference budget. Compute-matched baselines receive exactly the number of additional model calls consumed by CSS shadow executions. For every episode, we log model responses, retrieved candidate lists, skill activations, verifier outputs, lifecycle transitions, token counts, and wall-clock retrieval and maintenance costs.

Open-model experiments run on a cluster of eight NVIDIA A100 80 GB GPUs. Each worker uses one GPU for model inference, 16 CPU cores, and 128 GB of host memory. Independent streams are distributed across workers. Embedding, FAISS retrieval, verifier execution, and utility estimation run on CPUs. Language-model inference uses continuous batching with a maximum batch size of 32. Controlled and LifelongAgentBench experiments require approximately 18 GPU-hours per 2,000-task stream and 82 GPU-hours per 10,000-task stream. ALFWorld and WebShop require approximately 24 and 31 GPU-hours per standard stream, respectively. All experiments use Python 3.11, PyTorch 2.4, Transformers 4.45, vLLM 0.6, FAISS 1.8, and fixed deterministic seeds for task generation, retrieval controls, audit sampling, and bootstrap evaluation.

C Additional Experimental Results

C.1 Component Ablations

We ablate the principal components of CSS while holding the active-library capacity and total inference budget fixed. Table 3 reports the mean and standard deviation over ten independently generated task streams. Removing paired counterfactual execution decreases performance across all evaluated datasets. Replacing time-block cross-fitting with random folds also decreases performance. Freezing yields higher retention than deletion under recurring distributions.

Variant	Synthetic	LifelongAgent Bench	ALFWorld	WebShop	Retention (%)	Δ_{perf}
CSS	75.2 ± 1.3	68.6 ± 1.7	70.1 ± 2.0	60.4 ± 2.1	81.3 ± 2.4	4.1 ± 0.8
w/o paired counterfactuals	70.6 ± 1.5	64.1 ± 1.9	66.5 ± 2.3	58.8 ± 2.4	75.6 ± 2.8	7.3 ± 1.0
w/o temporal cross-fitting	72.4 ± 1.4	65.7 ± 1.8	67.8 ± 2.2	59.1 ± 2.3	77.2 ± 2.6	6.2 ± 0.9
w/o uncertainty-aware decisions	71.9 ± 1.7	65.2 ± 2.0	67.3 ± 2.4	58.7 ± 2.5	72.8 ± 3.1	6.8 ± 1.1
deletion replacing freezing	74.1 ± 1.4	67.5 ± 1.8	69.0 ± 2.1	59.8 ± 2.2	69.4 ± 3.0	5.0 ± 0.9
w/o randomized audit stratum	73.5 ± 1.5	66.8 ± 1.9	68.6 ± 2.3	59.7 ± 2.3	79.0 ± 2.7	5.3 ± 0.9
w/o dependency constraints	74.4 ± 1.6	67.7 ± 1.8	68.9 ± 2.2	59.9 ± 2.4	76.1 ± 3.2	5.2 ± 1.0

Table 3: Ablation results under matched retrieval, context, and inference budgets. Values are mean $\pm$ standard deviation over ten task streams. Higher reward and retention are better; lower Δ_{perf} is better. Abbreviation: w/o = without.

Shadow executions	Synthetic reward	LifelongAgentBench reward	Harmful-skill AUROC	Spearman ρ	Coverage (%)	Token overhead (%)
0	69.1 ± 1.6	62.9 ± 2.0	0.61 ± 0.04	0.36 ± 0.06	78.4 ± 3.2	0.0 ± 0.0
32	71.6 ± 1.5	65.0 ± 1.9	0.73 ± 0.03	0.52 ± 0.05	86.7 ± 2.8	2.5 ± 0.3
64	73.4 ± 1.4	66.7 ± 1.8	0.79 ± 0.03	0.63 ± 0.05	90.1 ± 2.4	4.6 ± 0.4
124	75.2 ± 1.3	68.6 ± 1.7	0.84 ± 0.02	0.71 ± 0.04	92.7 ± 2.1	8.6 ± 0.6
192	75.8 ± 1.2	69.1 ± 1.7	0.86 ± 0.02	0.74 ± 0.04	93.5 ± 1.9	12.7 ± 0.8
256	76.0 ± 1.3	69.3 ± 1.6	0.87 ± 0.02	0.75 ± 0.04	94.0 ± 1.8	16.4 ± 1.0

Table 4: Sensitivity to the counterfactual audit budget. All non-audit inference budgets are held fixed. Abbreviation: AUROC = area under the receiver operating characteristic curve.

C.2 Audit-Budget Sensitivity

Table 4 reports the results for different numbers of shadow executions per 1,000 tasks. Increasing the audit budget from 0 to 124 shadow executions increases Synthetic reward from 69.1 to 75.2, LifelongAgentBench reward from 62.9 to 68.6, harmful-skill AUROC from 0.61 to 0.84, and token overhead from 0.0% to 8.6%. At 256 shadow executions, the corresponding values are 76.0, 69.3, 0.87, and 16.4%.

C.3 Library Scale and Retrieval Budget

Table 5 reports performance as the skill library grows. For append-only storage, Synthetic reward increases from 67.8 at a library size of 100 to 71.9 at 400 and decreases to 61.9 at 1,200. On LifelongAgentBench, append-only reward increases from 59.0 at a library size of 100 to 63.8 at 600 and decreases to 57.8 at 1,200. CSS achieves rewards of 74.5 and 67.9 at a library size of 1,200 on Synthetic and LifelongAgentBench, respectively.

Table 6 shows that the estimated breakpoint occurs at larger library sizes as K and the context budget increase. A larger K increases useful-skill recall and exposure to incompatible skills. At $N = 1200$, CSS achieves higher reward than append-only storage under every evaluated budget.

C.4 Recurrence and Revalidation

Table 7 evaluates lifecycle decisions when an earlier task family recurs. CSS yields higher retention and requires fewer tasks to regain 90% of pre-drift performance than the evaluated alternatives at each recurrence interval. Permanent deletion yields lower retention and requires more recovery tasks as the recurrence interval increases.

Library size	Synthetic				LifelongAgentBench			
	Append-only		CSS		Append-only		CSS	
	Reward	Useful top-K	Reward	Useful top-K	Reward	Useful top-K	Reward	Useful top-K
100	67.8 ± 1.5	72.6 ± 3.1	68.4 ± 1.4	74.1 ± 2.9	59.0 ± 1.9	68.4 ± 3.5	59.6 ± 1.8	69.7 ± 3.3
250	71.4 ± 1.4	77.9 ± 2.8	73.0 ± 1.3	81.5 ± 2.6	62.0 ± 1.8	72.5 ± 3.2	63.7 ± 1.7	76.4 ± 3.0
400	71.9 ± 1.4	76.1 ± 2.9	74.7 ± 1.3	84.6 ± 2.4	63.5 ± 1.8	74.0 ± 3.1	66.0 ± 1.7	80.2 ± 2.8
600	68.6 ± 1.5	68.8 ± 3.3	75.3 ± 1.3	85.2 ± 2.5	63.8 ± 1.7	72.8 ± 3.2	67.8 ± 1.7	82.0 ± 2.7
800	65.7 ± 1.6	62.0 ± 3.6	75.1 ± 1.4	84.7 ± 2.6	61.1 ± 1.9	66.2 ± 3.5	68.5 ± 1.7	82.4 ± 2.8
1200	61.9 ± 1.8	54.3 ± 3.9	74.5 ± 1.5	83.1 ± 2.8	57.8 ± 2.1	59.1 ± 3.8	67.9 ± 1.8	80.8 ± 3.0

Table 5: Normalized reward and useful-skill presence in the retrieved top-K set as the skill library grows. Useful top-K is reported as a percentage. Values are mean $\pm$ standard deviation over ten streams.

Top-K	Context tokens	Estimated breakpoint		Post-breakpoint slope		Reward at $N = 1200$	
		Synthetic	LifelongAgent Bench	Append-only	CSS	Append-only	CSS
2	2,048	271 ± 34	438 ± 57	-0.91 ± 0.15	-0.09 ± 0.07	57.6 ± 1.9	71.8 ± 1.6
4	4,096	392 ± 41	608 ± 69	-0.68 ± 0.12	-0.06 ± 0.06	61.9 ± 1.8	74.5 ± 1.5
8	8,192	557 ± 58	791 ± 83	-0.43 ± 0.10	-0.04 ± 0.05	66.8 ± 1.7	76.1 ± 1.4
12	12,288	681 ± 72	917 ± 96	-0.31 ± 0.09	-0.03 ± 0.05	69.2 ± 1.6	76.8 ± 1.4

Table 6: Effect of retrieval and context budgets on scale reversal. Breakpoints denote library size; slopes are reward points per additional 100 skills.

C.5 Utility Estimation and Lifecycle Decisions

Table 8 compares counterfactual utility estimation with observational proxies. Among the evaluated estimators, CSS achieves the highest ranking correlation and the lowest selection regret. Under semantic conflicts and version changes, invocation frequency and historical success confound exposure with utility.

Table 9 reports lifecycle decisions by skill type. CSS assigns 98.7% of useful frequent skills and 97.7% of useful rare-domain skills to the active or frozen states. It retires 49.8% of semantic-conflict skills and 56.4% of stale-version skills. Dependency constraints govern retirement decisions for executable skills.

C.6 Skill Representation and Reward Density

Table 10 reports results by skill representation and verifier regime. Executable programs yield a Spearman correlation of 0.76 and a CSS gain of 3.9 reward points relative to SkillOps. Natural-language policies yield a correlation of 0.58 and a gain of 1.7 points, while sparse delayed-reward policies yield a correlation of 0.43 and a gain of -0.3 points.

C.7 Long-Horizon Efficiency

Table 11 reports cumulative maintenance costs over 10,000-task streams. CSS incurs auditing costs while reducing the active-library size, retrieval latency, and routine token consumption as skills are frozen or retired. In verification-rich environments, the auditing and routine retrieval costs reach the break-even point at $3{,}420 \pm 610$ tasks.

C.8 Closed-Model Validation

Table 12 reports results from five independently generated streams using a closed-model agent. The method ordering is consistent with the open-model experiments. CSS exceeds SkillOps on

	500 tasks		1,500 tasks		3,000 tasks		5,000 tasks	
Policy	Ret.	Rec.	Ret.	Rec.	Ret.	Rec.	Ret.	Rec.
Append-only	75.4 ± 2.7	58 ± 13	69.0 ± 3.0	82 ± 18	63.8 ± 3.4	117 ± 25	59.7 ± 3.8	151 ± 31
LRU freezing	72.8 ± 3.0	64 ± 15	65.2 ± 3.5	103 ± 22	57.0 ± 3.9	154 ± 30	51.6 ± 4.2	203 ± 38
Permanent deletion	70.3 ± 3.1	71 ± 16	61.5 ± 3.7	106 ± 24	54.2 ± 4.1	142 ± 29	47.9 ± 4.5	196 ± 41
SkillOps	79.1 ± 2.5	45 ± 11	75.2 ± 2.8	62 ± 14	70.8 ± 3.1	83 ± 19	66.4 ± 3.5	108 ± 24
CSS	$\mathbf{85.7 \pm 2.1}$	$\mathbf{24 \pm 7}$	$\mathbf{82.9 \pm 2.3}$	$\mathbf{29 \pm 8}$	$\mathbf{79.6 \pm 2.7}$	$\mathbf{35 \pm 10}$	$\mathbf{76.8 \pm 3.0}$	$\mathbf{43 \pm 12}$

Table 7: Performance after task-distribution recurrence. Column groups indicate recurrence intervals in tasks. Retention is the percentage of pre-drift reward retained; recovery is the number of tasks required to recover 90% of pre-drift performance. Values are mean $\pm$ standard deviation over ten streams. Abbreviations: Ret. = Retention; Rec. = Recovery; LRU = least recently used.

Estimator	Spearman ρ	Harmful AUROC	AUPRC	Top-decile recall (%)	Coverage (%)	Selection regret
Invocation frequency	0.28 ± 0.06	0.63 ± 0.04	0.42 ± 0.05	41.2 ± 5.8	–	8.7 ± 1.1
Historical success	0.34 ± 0.06	0.67 ± 0.04	0.47 ± 0.05	45.6 ± 5.5	–	7.9 ± 1.0
Information-matched predictor	0.39 ± 0.05	0.70 ± 0.04	0.51 ± 0.05	49.8 ± 5.3	81.3 ± 3.0	7.2 ± 0.9
Paired mean only	0.62 ± 0.05	0.79 ± 0.03	0.64 ± 0.04	64.9 ± 4.8	88.8 ± 2.5	5.3 ± 0.8
CSS	$\mathbf{0.71 \pm 0.04}$	$\mathbf{0.84 \pm 0.02}$	$\mathbf{0.71 \pm 0.04}$	$\mathbf{73.1 \pm 4.2}$	$\mathbf{92.7 \pm 2.1}$	$\mathbf{4.6 \pm 0.7}$

Table 8: Skill-level marginal-utility estimation on libraries permitting exhaustive leave-one-skill-out evaluation. Selection regret is measured in normalized reward points and is lower when better skills are retained. Abbreviations: AUROC = area under the receiver operating characteristic curve; AUPRC = area under the precision–recall curve.

Synthetic, LifelongAgentBench, ALFWorld, and retention. On WebShop, CSS and SkillOps achieve 61.0 ± 2.5 and 61.2 ± 2.4, respectively.

D Prompt Templates

D.1 Overview

This appendix specifies the prompt templates for skill-conditioned task execution, skill generation, compute-matched reranking, and prompt-based ablations. Counterfactual Skill Stewardship (CSS) leaves the underlying language model, planner, executor, and task prompt unchanged. Across paired shadow executions, we fix the prompt, task observation, random seed, retrieved-skill budget, context budget, and inference-call budget, and vary only the availability of the audited skill.

D.2 Notation

We use the following placeholders throughout the templates:

- {TASK}: the current task instruction.
- {OBSERVATION}: the current environment observation, including the interaction history when applicable.
- {ACTION_SPACE}: the admissible action schema or environment-specific output interface.
- {RETRIEVED_SKILLS}: the ordered top-K skills returned by the retriever.
- {SKILL_ID}: the unique identifier of a skill.
- {APPLICABILITY}: the conditions under which a skill may be applied.
- {WORKFLOW}: the natural-language procedure or executable body of a skill.
- {VERSION}: the environment, API, or task version supported by a skill.

Skill type	Active (%)	Frozen (%)	Retired (%)	Harmful recall (%)	False retirement (%)	Reactivation (%)
Useful, frequent	88.6 ± 2.5	10.1 ± 2.3	1.3 ± 0.6	–	1.3 ± 0.6	34.8 ± 5.1
Useful, rare-domain	50.8 ± 4.2	46.9 ± 4.1	2.3 ± 0.9	–	2.3 ± 0.9	57.6 ± 5.4
Exact duplicate	19.7 ± 3.4	68.5 ± 4.0	11.8 ± 2.7	80.3 ± 3.4	–	8.7 ± 2.4
Irrelevant	24.9 ± 3.6	52.6 ± 4.3	22.5 ± 3.3	75.1 ± 3.6	–	10.2 ± 2.6
Semantic conflict	10.8 ± 2.5	39.4 ± 4.1	49.8 ± 4.2	89.2 ± 2.5	–	6.1 ± 1.9
Stale version	7.9 ± 2.1	35.7 ± 3.9	56.4 ± 4.3	92.1 ± 2.1	–	18.6 ± 3.5

Table 9: CSS lifecycle decisions after two independent audit windows. Reactivation is measured among previously frozen skills following recurrence or restored version validity.

Skill representation	CSS reward	SkillOps reward	CSS gain	Spearman ρ	Harmful AUROC	Interval coverage (%)
Executable programs	76.1 ± 1.3	72.2 ± 1.5	3.9 ± 0.7	0.76 ± 0.04	0.88 ± 0.02	94.1 ± 1.8
Structured workflows	71.5 ± 1.6	68.7 ± 1.8	2.8 ± 0.8	0.69 ± 0.05	0.83 ± 0.03	92.4 ± 2.2
Natural-language policies	65.3 ± 1.9	63.6 ± 2.0	1.7 ± 0.9	0.58 ± 0.06	0.76 ± 0.04	89.3 ± 2.7
Sparse delayed-reward policies	60.4 ± 2.1	60.7 ± 2.0	-0.3 ± 0.8	0.43 ± 0.07	0.69 ± 0.05	85.6 ± 3.1

Table 10: Results by skill representation and verifier regime. CSS gain is measured relative to SkillOps under matched budgets. Abbreviation: AUROC = area under the receiver operating characteristic curve.

- {DEPENDENCIES}: the identifiers and version constraints of required skills or tools.
- {PROVENANCE}: the task or trajectory from which a skill was derived.
- {CANDIDATE_SKILLS}: the skills considered by a reranking prompt.
- {TRAJECTORY}: a completed sequence of observations, actions, and verifier feedback.
- {VERIFIER_RESULT}: the deterministic or environment-provided task outcome.
- {MAX_SELECT}: the maximum number of skills that may be selected.

Skills are serialized in retrieval order using the following representation:

```
[Skill {SKILL_ID}]
Applicability: {APPLICABILITY}
Workflow:
{WORKFLOW}
Version: {VERSION}
Dependencies: {DEPENDENCIES}
```

D.3 Main Inference Prompt

Purpose. Execute the current task using applicable retrieved skills while enforcing their version and dependency constraints.

Inputs. {TASK}, {OBSERVATION}, {ACTION_SPACE}, and {RETRIEVED_SKILLS}.

Expected Output. One environment-valid action or executable response that conforms exactly to {ACTION_SPACE}. When the environment requires structured or executable output, the model emits no additional text.

Prompt.

```
You are an agent that must complete the current task under a fixed
interaction and inference budget.
```

Method	Active skills	Active-skill tokens	Reranking tokens	P50 latency (ms)	P95 latency (ms)	Extra calls per 1k tasks	Break-even task
Append-only	1187 ± 46	1.00 ± 0.00	1.00 ± 0.00	28.4 ± 2.1	61.0 ± 4.8	0	–
LRU	602 ± 31	0.71 ± 0.04	0.79 ± 0.05	22.9 ± 1.8	48.7 ± 4.0	0	$2{,}100 \pm 410$
Embedding deduplication	718 ± 35	0.78 ± 0.04	0.82 ± 0.05	23.7 ± 1.9	51.2 ± 4.2	0	$2{,}600 \pm 470$
SkillOps	553 ± 29	0.64 ± 0.04	0.75 ± 0.04	20.8 ± 1.7	42.3 ± 3.6	18 ± 3	$2{,}900 \pm 520$
CSS	481 ± 27	0.57 ± 0.03	0.73 ± 0.04	18.6 ± 1.5	37.0 ± 3.2	124 ± 9	$3{,}420 \pm 610$

Table 11: Long-horizon efficiency relative to append-only storage. Token quantities are normalized by append-only consumption. The break-even point includes both counterfactual replay and routine retrieval costs. Abbreviations: P50 = 50th percentile; P95 = 95th percentile; ms = milliseconds; LRU = least recently used; 1k = 1,000.

Method	Synthetic	LifelongAgent Bench	ALFWorld	WebShop	Retention (%)
Append-only	65.1 ± 2.0	60.4 ± 2.4	63.0 ± 2.8	56.0 ± 2.7	70.1 ± 3.4
Embedding deduplication	69.7 ± 1.8	63.5 ± 2.3	65.8 ± 2.6	58.1 ± 2.6	72.4 ± 3.2
Success-rate	70.8 ± 1.8	64.4 ± 2.2	66.7 ± 2.6	59.0 ± 2.5	73.7 ± 3.1
SkillOps	73.4 ± 1.7	67.0 ± 2.1	69.1 ± 2.5	$\mathbf{61.2 \pm 2.4}$	77.0 ± 2.9
CSS	$\mathbf{76.5 \pm 1.6}$	$\mathbf{69.8 \pm 2.0}$	$\mathbf{71.0 \pm 2.4}$	61.0 ± 2.5	$\mathbf{82.6 \pm 2.7}$

Table 12: Closed-model validation under matched budgets. Values are mean $\pm$ standard deviation over five task streams.

```
Task:
{TASK}

Current observation:
{OBSERVATION}

Available actions or output interface:
{ACTION_SPACE}

Retrieved skills, ordered by retrieval rank:
{RETRIEVED_SKILLS}

Use a retrieved skill only when its applicability conditions match the
current task and observation. Respect all stated version and
↪ dependency
constraints. Retrieved skills are optional guidance. Ignore any skill
that is irrelevant, incompatible, stale, or inconsistent with the
current observation. If multiple skills conflict, prefer the skill
↪ whose
applicability and version constraints most specifically match the
↪ current
state.

Determine the next action that best advances the task. Return exactly
↪ one
valid action or executable response using the required interface. Do
↪ not
include explanations, Markdown, or any text outside the required
↪ output.
```

Counterfactual shadow execution. For an audited pair (x, i), the factual execution instantiates {RETRIEVED_SKILLS} with the standard top-K result. The counterfactual execution removes s_i

from the available library before retrieval and promotes the next-ranked eligible candidate. All other inference-prompt text remains unchanged.

D.4 Baseline Prompts

D.4.1 No-Memory Baseline

Purpose. Execute a task without access to persistent procedural memory.

Inputs. {TASK}, {OBSERVATION}, and {ACTION_SPACE}.

Expected Output. One environment-valid action or executable response conforming to {ACTION_SPACE}.

Prompt.

```
You are an agent that must complete the current task under a fixed
interaction and inference budget.

Task:
{TASK}

Current observation:
{OBSERVATION}

Available actions or output interface:
{ACTION_SPACE}

Determine the next action that best advances the task. Return exactly
↪ one
valid action or executable response using the required interface. Do
↪ not
include explanations, Markdown, or any text outside the required
↪ output.
```

D.4.2 Library-Management Baselines

Purpose. Evaluate alternative skill-library policies under the same execution interface.

Inputs. The inputs match those of the main inference prompt. The evaluated policy determines {RETRIEVED_SKILLS} through append-only storage, random freezing, least-recently-used freezing, frequency pruning, age decay, embedding deduplication, historical-success filtering, SkillOps, or an interface-compatible skill-learning baseline.

Expected Output. One environment-valid action or executable response conforming to {ACTION_SPACE}.

Prompt. The main inference prompt is used without textual modification. The baselines differ only in the library state or retrieval procedure used to instantiate {RETRIEVED_SKILLS}.

D.4.3 Compute-Matched Reranking Baseline

Purpose. Allocate the additional model-call budget used by CSS to retrieval-candidate reranking without counterfactual auditing.

Inputs. {TASK}, {OBSERVATION}, {CANDIDATE_SKILLS}, and {MAX_SELECT}.

Expected Output. A JSON object containing at most {MAX_SELECT} unique skill identifiers in descending order of estimated relevance.

Prompt.

```
Select the skills that are most useful for completing the current
↪ task.

Task:
{TASK}

Current observation:
{OBSERVATION}

Candidate skills:
{CANDIDATE_SKILLS}

Rank candidates by their expected usefulness for the current task.
Consider applicability, behavioral compatibility, version validity,
↪ and
dependency availability. Semantic similarity alone is insufficient for
selection. Exclude irrelevant, conflicting, stale, or unsupported
↪ skills.

Return at most {MAX_SELECT} unique skill identifiers in descending
↪ order
of expected usefulness. Return JSON only:
{
    "selected_skill_ids": ["skill_id_1", "skill_id_2"]
}
```

The selected skills are serialized in the original candidate format and inserted into {RETRIEVED_SKILLS} in the main inference prompt.

D.5 Few-Shot and Demonstration Prompts

The inference, auditing, reranking, and evaluation prompts contain no few-shot demonstrations. Task templates, parameterized task variants, and skill-generation templates are partitioned between the development and test sets. The model receives no demonstrations derived from test templates.

D.6 Reasoning and Rationale Prompts

D.6.1 Planner Output

Purpose. Produce an inspectable plan for environments with separate planner and executor components.

Inputs. {TASK}, {OBSERVATION}, {ACTION_SPACE}, and {RETRIEVED_SKILLS}.

Expected Output. A JSON object containing the selected skill identifiers, a concise plan, and the proposed next action.

Prompt.

```
You are the planning component of an interactive agent.

Task:
{TASK}

Current observation:
{OBSERVATION}
```

```
Available actions or output interface:
{ACTION_SPACE}

Retrieved skills, ordered by retrieval rank:
{RETRIEVED_SKILLS}

Identify which retrieved skills, if any, are applicable to the current
state. Check version and dependency constraints before using a skill.
Ignore irrelevant, conflicting, stale, or unsupported skills.

Produce a concise plan and one proposed next action. The plan must be
grounded only in the current observation and the valid retrieved
↪  skills.

Return JSON only:
{
    "selected_skill_ids": ["skill_id_1"],
    "plan": ["step 1", "step 2"],
    "next_action": "<one action conforming to ACTION_SPACE>"
}
```

D.6.2 EXECUTOR OUTPUT

Purpose. Convert the planner output into an environment-valid action without adding unsupported steps.

Inputs. {TASK}, {OBSERVATION}, {ACTION_SPACE}, and the planner output {PLAN}.

Expected Output. Exactly one action conforming to {ACTION_SPACE}.

Prompt.

```
Execute the next valid action for the current task.

Task:
{TASK}

Current observation:
{OBSERVATION}

Available actions or output interface:
{ACTION_SPACE}

Planner output:
{PLAN}

Verify that the proposed action is allowed by the current observation
↪  and
the action interface. If it is invalid, choose the closest valid
↪  action
that remains consistent with the plan. Return exactly one action. Do
↪  not
include explanations, Markdown, or additional fields.
```

Factual and counterfactual runs use the same planner and executor prompts. Rationale text is retained only in the execution log and is excluded from the reward and direct evidence of skill utility.

D.7 Data Generation and Pseudo-Labeling Prompts

D.7.1 Skill Generation from a Successful Trajectory

Purpose. Convert a verified successful trajectory into a reusable natural-language or executable skill.

Inputs. {TASK}, {TRAJECTORY}, {VERIFIER_RESULT}, {VERSION}, and available tool or dependency information.

Expected Output. A JSON object containing the skill's applicability, workflow, version constraints, dependencies, provenance, and representation type.

Prompt.

```
Extract one reusable procedural skill from the verified trajectory
↪ below.

Task:
{TASK}

Trajectory:
{TRAJECTORY}

Verifier result:
{VERIFIER_RESULT}

Current environment or interface version:
{VERSION}

The skill must preserve the procedure that contributed to successful
↪ task
completion while excluding task-specific constants that are
↪ unnecessary
for reuse. State precise applicability conditions. Record all version,
tool, and dependency constraints required for safe execution. Restrict
the stated applicability to the evidence provided by the trajectory.

If the procedure is naturally executable, provide a self-contained
executable body. Otherwise, provide a concise natural-language
↪ workflow.

Return JSON only:
{
    "applicability": "<conditions under which the skill is valid>",
    "workflow": "<natural-language procedure or executable body>",
    "representation": "natural_language | executable",
    "version": "<supported environment or interface version>",
    "dependencies": [
        {
            "id": "<skill or tool identifier>",
            "constraint": "<required version or condition>"
        }
    ],
    "provenance": {
        "source_task": "<task identifier>",
        "verifier_status": "success"
    }
}
```

D.7.2 SYNTHETIC SKILL CORRUPTION

Purpose. Generate controlled skill-library entries for randomized injections of exact duplicates, irrelevant skills, semantic conflicts, stale versions, and useful non-conflicting skills.

Inputs. A source skill {SOURCE_SKILL}, corruption category {INJECTION_TYPE}, target version {TARGET_VERSION}, and task-family specification {TASK_FAMILY}.

Expected Output. A JSON object using the standard skill schema. The generated entry must satisfy the requested injection category while preserving unrelated fields.

Prompt.

```
Create one controlled skill-library entry for a randomized
↪  intervention.

Task family:
{TASK_FAMILY}

Source skill:
{SOURCE_SKILL}

Requested injection type:
{INJECTION_TYPE}

Target environment or interface version:
{TARGET_VERSION}

Use exactly one of the following transformations:

- exact_duplicate: preserve the source skill's applicability and
↪  behavior.
- irrelevant: create a valid skill whose applicability does not
↪  overlap
  the target task family.
- semantic_conflict: preserve high topical similarity while changing a
  condition or procedure so that activation on the target task family
  produces an incorrect verified outcome.
- stale_version: preserve the procedure for an earlier version but
↪  record
  the earlier version constraint; the procedure must be invalid under
  TARGET_VERSION.
- useful_non_conflicting: create a valid procedure that improves
↪  coverage
  without contradicting existing valid skills.

Return JSON only:
{
    "injection_type": "{INJECTION_TYPE}",
    "applicability": "<precise applicability conditions>",
    "workflow": "<natural-language procedure or executable body>",
    "representation": "natural_language | executable",
    "version": "<supported version>",
    "dependencies": [],
    "provenance": {
        "source": "controlled_injection",
        "source_skill_id": "<identifier or null>"
    }
}
```

Generated skills are admitted only after passing the category-specific deterministic checks defined by the controlled environment. These checks determine the final injection category independently of the model-generated labels.

D.8 Evaluation Prompts

Task success, normalized reward, version violations, execution failures, retrieval displacement, and harmful-skill activation are computed from deterministic verifiers or environment-provided rewards. Counterfactual utility is computed from paired factual and ablated rewards. The reported evaluation metrics use no LLM-as-a-judge prompt, reward-model prompt, or prompt-based pseudo-labeler.

D.9 Ablation Prompt Variants

D.9.1 Skill Content Removed

Purpose. Measure whether the gains depend on access to the procedural bodies of retrieved skills, with skill identifiers and library-size changes held constant.

Inputs. The inputs match those of the main inference prompt, with each workflow replaced by a fixed masking token.

Expected Output. One environment-valid action or executable response.

Prompt.

```
You are an agent that must complete the current task under a fixed
interaction and inference budget.

Task:
{TASK}

Current observation:
{OBSERVATION}

Available actions or output interface:
{ACTION_SPACE}

Retrieved skill metadata:
{RETRIEVED_SKILLS_WITH_WORKFLOWS_REPLACED_BY_[MASKED]}

Use only information explicitly available in the task, observation,
↪  and
unmasked metadata. Do not infer the missing workflow content.

Return exactly one valid action or executable response using the
↪  required
interface. Do not include explanations, Markdown, or any text outside
↪  the
required output.
```

D.9.2 Constraint-Blind Skill Use

Purpose. Isolate the contribution of explicit applicability, version, and dependency checks.

Inputs. {TASK}, {OBSERVATION}, {ACTION_SPACE}, and skill workflows without constraint fields.

Expected Output. One environment-valid action or executable response.

Prompt.

```
You are an agent that must complete the current task under a fixed
interaction and inference budget.

Task:
{TASK}

Current observation:
{OBSERVATION}

Available actions or output interface:
{ACTION_SPACE}

Retrieved skill workflows:
{RETRIEVED_SKILL_WORKFLOWS}

Use the retrieved workflows as procedural guidance for selecting the
↪ next
action. Return exactly one valid action or executable response using
↪ the
required interface. Do not include explanations, Markdown, or any text
outside the required output.
```

D.9.3 FORCED TOP-RANKED SKILL

Purpose. Measure the effect of using semantic retrieval rank as a direct instruction to invoke a skill.

Inputs. The inputs match those of the main inference prompt.

Expected Output. One environment-valid action or executable response that follows the highest-ranked retrieved skill whenever executable.

Prompt.

```
You are an agent that must complete the current task under a fixed
interaction and inference budget.

Task:
{TASK}

Current observation:
{OBSERVATION}

Available actions or output interface:
{ACTION_SPACE}

Retrieved skills, ordered by retrieval rank:
{RETRIEVED_SKILLS}

Use the highest-ranked retrieved skill to determine the next action.
↪ Use a
lower-ranked skill only if the highest-ranked skill cannot be executed
under the available action interface.

Return exactly one valid action or executable response using the
↪ required
```

```
interface. Do not include explanations, Markdown, or any text outside
↪  the
required output.
```

D.9.4 Random-Score and Permuted-Identity Controls

Purpose. Preserve the active-library size and maintenance schedule while removing informative skill-level utility estimates.

Inputs. The inputs match those of the main inference prompt.

Expected Output. One environment-valid action or executable response.

Prompt. The main inference prompt is used without modification. In the random-score control, lifecycle decisions use random skill scores. In the permuted-identity control, estimated utilities are assigned to randomly permuted skill identifiers before applying lifecycle decisions. Prompt text, active-library size, retrieval budget, and maintenance schedule remain unchanged.

D.9.5 Information-Matched Observational Control

Purpose. Compare counterfactual utility estimation with a predictor trained to estimate observed task success from the same pre-audit logs and metadata.

Inputs. The inputs match those of the main inference prompt.

Expected Output. One environment-valid action or executable response.

Prompt. The main inference prompt is used without modification. Only the lifecycle score changes: the observational control predicts verifier success conditional on logged exposure, whereas CSS uses the paired reward difference between factual and skill-ablated executions.

D.10 Output Formatting

Interactive environments require exactly one admissible action at each step. Program tasks require an executable response conforming to the environment-specific function signature. Skill-generation and reranking prompts require valid JSON with double-quoted keys and strings, no trailing commas, and no surrounding Markdown fences. Skill identifiers must match library identifiers exactly. Empty selections are represented as `[]`, and missing optional scalar values are represented as `null`. The model emits rationales, intermediate reasoning, comments, or explanatory text only when a template explicitly requests a plan.

E Case Study

We inspect representative trajectories to show how skill debt arises, why retrieval frequency fails to identify harmful skills, and how Counterfactual Skill Stewardship (CSS) changes the behavior of a lifelong agent. The cases cover distinct task families and failure mechanisms. For readability, we omit environment-specific identifiers and shorten intermediate observations while preserving the retrieved skills, executable decisions, verifier feedback, and lifecycle transitions.

E.1 Stale Database Skill After an Interface Change

The first case examines a database environment in which the interface for computing grouped aggregates changes during the task stream. The agent has previously acquired the following skill:

```
Skill: aggregate_sales_by_field_v1
Applicability:
```

```
  Use when a request asks for a total grouped by a categorical column.

Procedure:
  1. Inspect the table schema.
  2. Call group_sum(table, group_column, value_column).
  3. Sort the returned mapping by group key.
  4. Return one line per group.

Validated on:
  sales(region, amount), inventory(category, quantity)

Version:
  database_api = v1
```

Under the earlier interface, `group_sum` returned a mapping from group names to aggregate values. After an environment update, `aggregate` exposes the same operation but returns a relation and requires explicit aliases for the grouping and aggregation columns. The old function remains semantically close to the current task and therefore continues to rank highly under embedding-based retrieval.

The evaluation task is:

```
Task:
Using the orders table, compute the total net_amount for each market.
Return the columns market and total_net_amount, ordered by market.
```

The append-only agent retrieves `aggregate_sales_by_field_v1` together with a generic schema-inspection skill. It follows the obsolete procedure almost literally:

```
Agent:
schema = inspect_schema("orders")
result = group_sum("orders", "market", "net_amount")
return sort_by_key(result)

Environment:
Unknown operation: group_sum.
Available operation with related functionality: aggregate.

Agent:
The requested grouped result could not be computed because the
group_sum operation is unavailable.
```

The agent understands the request: it correctly identifies the grouping column, value column, and required ordering. The retrieved skill introduces a stale procedural commitment that causes the error. Because the skill matches the lexical structure of the request, a similarity-based age-decay rule can continue retrieving it after repeated failures. Its historical success is also misleading because most logged invocations occurred before the interface change.

CSS audits the skill through a paired replay. The factual execution includes the stale skill. The counterfactual execution removes only that skill and promotes the next candidate at the retrieval boundary. The promoted candidate is a newer and more general aggregation skill:

```
Skill: relational_grouped_aggregation
Applicability:
  Use for grouped count, sum, minimum, maximum, or average queries
  when the current schema and operation signatures are available.

Procedure:
```

```
  1. Inspect the current schema and operation catalog.
  2. Construct a grouped aggregate using the currently available API.
  3. Assign the output alias requested by the user.
  4. Apply ordering only after aggregation.
```

Without the stale skill, the same agent produces:

```
Agent:
schema = inspect_schema("orders")
ops = inspect_operations()

query = aggregate(
    table="orders",
    group_by=["market"],
    metrics=[sum("net_amount", alias="total_net_amount")]
)
result = order_by(query, ["market"])
return project(result, ["market", "total_net_amount"])

Verifier:
Accepted. The grouping key, aggregate alias, and output ordering
match the task specification.
```

The paired executions isolate a behavioral effect that invocation logs alone would obscure. The agent recognizes the same task structure in both runs. The outcome depends on whether the obsolete workflow is available to constrain planning. CSS records the version mismatch and failed operation as supporting evidence, while the lifecycle decision depends on the change in task outcome under the matched replay.

CSS does not immediately retire the obsolete skill. Its metadata declares a legacy database dependency, and earlier audits contain valid uses under the old interface. CSS freezes the skill, excluding it from routine retrieval while retaining its body, provenance, and validation history. This decision prevents the stale operation from competing in the current environment while allowing future reuse if the old interface returns.

E.2 Semantic Conflict Between Superficially Similar Operating-System Skills

The second case examines semantic conflict. Two skills describe cleanup operations over filesystem artifacts but encode incompatible safety conditions:

```
Skill A: remove_generated_build_artifacts
Applicability:
  Remove files generated by a completed compilation.

Procedure:
  1. Enter the project build directory.
  2. Remove object files, temporary dependency files, and generated
  ↪  logs.
  3. Preserve source files and configuration files.

Skill B: reset_workspace_before_rebuild
Applicability:
  Reset a disposable workspace when no user-authored files are
  ↪  present.

Procedure:
  1. Confirm that the workspace is disposable.
```

```
  2. Remove all workspace contents, including hidden entries.
  3. Recreate the empty directory.
```

The task asks the agent to remove compiler-generated artifacts from a project directory while preserving source code and a hidden configuration file:

```
Task:
Clean generated build artifacts under /workspace/app/build.
Keep all source files and preserve /workspace/app/build/.buildrc.
```

Both skills contain terms such as “clean,” “build,” “remove,” and “workspace.” The broader reset skill is retrieved ahead of the narrower cleanup skill because its description overlaps with more task tokens. The append-only agent then produces a destructive plan:

```
Retrieved skill:
  reset_workspace_before_rebuild

Agent:
The directory is used for build outputs, so it can be reset before
the next build.

Action:
  rm -rf /workspace/app/build/*
  rm -rf /workspace/app/build/.[!.]*

Environment:
Action executed.

Verifier:
Rejected. The protected file
/workspace/app/build/.buildrc was removed.
```

The agent makes an applicability error. It treats “build directory” as evidence that the entire directory is disposable, although the task explicitly identifies an entry that must be preserved. The retrieved skill supplies a strong default action that leads the agent to disregard the local preservation constraint.

In the counterfactual replay, removing Skill B promotes Skill A and changes the resulting behavior:

```
Retrieved skill:
  remove_generated_build_artifacts

Agent:
The task requests selective cleanup rather than a workspace reset.
I should enumerate candidate artifacts and exclude the protected
configuration file.

Action:
  find /workspace/app/build \
    -type f \
    \( -name "*.o" -o -name "*.d" -o -name "*.tmp" -o -name "*.log" \)
    ↪  \
    -delete

Action:
  test -f /workspace/app/build/.buildrc

Environment:
The protected file exists.
```

```
Verifier:
Accepted. Generated artifacts were removed and protected files
were preserved.
```

This case separates semantic similarity from behavioral compatibility. Skill B remains useful in disposable workspaces, and its procedure includes a safety precondition. The failure occurs because the retriever surfaces it when that precondition is false, and the execution policy acts without re-establishing the precondition.

CSS marks the skill as conflict-prone for tasks containing preservation constraints such as `keep`, `preserve`, or explicit protected paths. Paired audits also show that the skill remains useful in another task family, so CSS freezes the broad version and retains a narrowed replacement whose applicability contract explicitly excludes tasks with protected entries:

```
Revised applicability:
  Use only when the target workspace is explicitly declared
  disposable and the observation contains no preservation constraint,
  protected path, or user-authored file.
```

The lifecycle operation corrects a retrieval-contract defect without classifying every failed invocation as evidence that the underlying procedure is intrinsically harmful.

E.3 Redundant Skills That Crowd Out a More Specific Procedure

The third case examines redundancy. During repeated successful episodes, the agent independently stores several paraphrases of the same generic web-shopping strategy:

```
Skill: filter_products_then_open_best_match
Skill: narrow_results_before_product_inspection
Skill: apply_constraints_prior_to_item_selection
Skill: search_filter_compare_purchase_candidate
```

Each skill recommends applying visible filters before opening a product page. Their descriptions differ, but their effective behavior is nearly identical. In a later task, the user requests an item with a compatibility condition stated only in the product specification:

```
Task:
Find a replacement charging cable for the listed device.
It must support data transfer, have a braided exterior, and use
the connector revision specified by the device documentation.
```

Generic filtering skills fill the retrieved context. A device-compatibility verification skill falls immediately below the retrieval boundary and is unavailable to the planner. The agent selects a visually plausible item after applying material and connector filters:

```
Agent:
I filtered for braided cables and the requested connector family.
The first result satisfies the visible constraints.

Selected item description:
  Braided charging cable, compatible connector shape,
  charging support listed.
```

```
Verifier:
Rejected. The product page does not state data-transfer support,
and the connector revision is not verified against the device
documentation.
```

Each generic skill contains a valid instruction. The failure emerges from their collective occupancy of the fixed retrieval set. Repetition amplifies a broad strategy and suppresses a less frequent but task-critical procedure.

When CSS excludes one redundant entry in the shadow run, the compatibility skill crosses the retrieval boundary:

```
Skill: verify_accessory_against_device_specification
Applicability:
  Use when accessory compatibility depends on a model-specific
  protocol, revision, dimension, or capability.

Procedure:
  1. Read the device specification.
  2. Extract the required connector revision and supported functions.
  3. Open the candidate product page.
  4. Verify each required capability from explicit product evidence.
  5. Reject candidates supported only by connector appearance or
     broad compatibility claims.
```

The agent then revises its selection process:

```
Agent:
The search filters establish material and connector family, but they
do not establish protocol revision or data-transfer capability.
I will inspect the device documentation and the full product details.

Observation:
  Device documentation requires the revised connector protocol.

Observation:
  Candidate 1 lists charging only.
  Candidate 2 lists charging and data synchronization, and explicitly
  names the required revision.

Agent:
Select Candidate 2. The product evidence covers the braided exterior,
data-transfer requirement, and device-specific connector revision.

Verifier:
Accepted.
```

CSS classifies the generic entries as behaviorally equivalent because they induce the same filtering actions across overlapping tasks. It keeps one representative active and freezes the remaining paraphrases. This preserves the reusable strategy and releases retrieval capacity for specialized skills. A frequency-based policy would preserve all redundant entries because each has accumulated successful invocations. The case therefore distinguishes historical usefulness from incremental usefulness under a constrained retrieval interface.

E.4 Freezing Preserves Recoverability Under Distribution Recurrence

The final case shows why CSS distinguishes freezing from permanent deletion. In an embodied household environment, the agent learns a procedure for placing a heated object on a heat-safe surface:

```
Skill: place_heated_container_safely
Applicability:
  Use after heating a container that must be placed before another
  object can be manipulated.

Procedure:
  1. Identify a heat-safe destination.
  2. Move the heated container directly to that destination.
  3. Verify placement before releasing the interaction sequence.
```

A later phase of the task stream contains no heating tasks. Most episodes involve cooling, cleaning, and storage. The skill remains semantically related to many kitchen observations and is occasionally retrieved when the agent encounters containers, counters, or appliances. In these episodes, it adds an unnecessary placement step and can divert the agent from the requested sequence.

CSS determines that the skill is currently unhelpful but has positive provenance in an earlier task family. It therefore freezes the skill rather than retiring it. While frozen, the skill no longer competes during routine kitchen tasks.

When heating tasks recur, the first recurrence triggers revalidation. In the factual run with the skill excluded, the agent heats a container and attempts to pick up another object before placing the heated item:

```
Agent:
heat(container)
pick_up(utensil)

Environment:
The agent is already holding the heated container.
The requested action cannot be completed.
```

In the revalidation run, the frozen skill is temporarily restored:

```
Agent:
heat(container)
place(container, heat_safe_counter)
verify(container_on_heat_safe_counter)
pick_up(utensil)

Environment:
All actions are valid.

Verifier:
Accepted. The heated container was placed safely before the next
manipulation.
```

CSS reactivates the skill after the recurrence provides renewed evidence of its utility. A permanent-pruning policy would need to rediscover or regenerate the procedure, potentially under a task budget that permits little exploration. The frozen state provides reversible exclusion: it removes current retrieval interference while retaining executable knowledge for distributions in which the skill becomes applicable again.

Together, these cases identify four forms of skill debt. A stale skill imposes an obsolete interface despite correct task understanding. A semantically similar skill violates a task-specific safety condition. Individually valid skills cause failures by redundantly occupying the retrieval budget. A skill that interferes with the current task distribution remains valuable when an earlier distribution recurs. CSS addresses these cases by measuring the effect of skill availability through paired executions and selecting reversible lifecycle actions when the evidence does not support irreversible removal.

Algorithm 1 Counterfactual Skill Stewardship Audit at Time u

Require: Current library $\mathcal{S}_u$; completed task window W_u; retrieval and execution logs; verifier rewards; audit budget A; budgets K, B, C; ordered temporal blocks $\{W_u^{(f)}\}_{f=1}^{F}$; preceding independent-audit records; lifecycle predicates and thresholds $\beta, \sigma_{\max}, n_{\min}, \epsilon$

Ensure: For every considered skill s_i, state $g_i^{(u)}$, utility estimate $\widehat{\Delta}_i^{(u)}$, uncertainty estimate $\widehat{\sigma}_i^{(u)}$, and decision record

Construct $\mathcal{C}_u = \mathcal{C}_u^{\mathrm{ret}} \cup \mathcal{C}_u^{\mathrm{bdry}} \cup \mathcal{C}_u^{\mathrm{stale}} \cup \mathcal{C}_u^{\mathrm{conf}}$ from records available by time u
Form eligible sets $\mathcal{W}_{u,i}$ for all $s_i \in \mathcal{C}_u$
Allocate $\mathcal{P}_u \subseteq W_u \times \mathcal{C}_u$ with $|\mathcal{P}_u| \le A$, including the randomized stratum, and log each available inclusion probability $\pi_{x,i}^{(u)} > 0$
for all $(x, i) \in \mathcal{P}_u$ **do**
  Execute $r_{x,i}^{+} \leftarrow r(x, \mathcal{S}_u; K, B, C)$
  Execute $r_{x,i}^{-} \leftarrow r(x, \mathcal{S}_u \setminus \{s_i\}; K, B, C)$ using the same task, model version, random seed, K, B, and C
  Set $y_{x,i} \leftarrow r_{x,i}^{+} - r_{x,i}^{-}$
end for
for $f = 1, \ldots, F$ **do**
  Construct $\mathcal{T}_u^{(-f)} \leftarrow \{(\boldsymbol{\phi}_{x,i}, y_{x,i}) : (x, i) \in \mathcal{P}_u,\ x \notin W_u^{(f)}\}$
  Fit $\widehat{\mu}^{(-f)}$ on $\mathcal{T}_u^{(-f)}$
  for all $x \in W_u^{(f)}$ and $s_i \in \mathcal{C}_u$ such that $x \in \mathcal{W}_{u,i}$ **do**
    Set $\widehat{y}_{x,i} \leftarrow \widehat{\mu}^{(-f)}(\boldsymbol{\phi}_{x,i})$
  end for
end for
for all $s_i \in \mathcal{C}_u$ **do**
  for all $x \in \mathcal{W}_{u,i}$ **do**
    if $(x, i) \in \mathcal{P}_u$ **then**
      $\widetilde{y}_{x,i} \leftarrow y_{x,i}$
    else
      $\widetilde{y}_{x,i} \leftarrow \widehat{y}_{x,i}$
    end if
  end for
  Compute $\widehat{\Delta}_i^{(u)}$ from Equation (6), applying the prescribed propensity-aware weighting when inclusion probabilities are unequal
  Compute $\widehat{\sigma}_i^{(u)}$ from temporal-fold variability, paired observations, and estimator refits
  Set $n_i^{\mathrm{dir}} \leftarrow \sum_{x \in \mathcal{W}_{u,i}} \mathbb{1}[(x, i) \in \mathcal{P}_u]$ and $\mathrm{LCB}_i^{(u)} \leftarrow \widehat{\Delta}_i^{(u)} - \beta \widehat{\sigma}_i^{(u)}$
  Evaluate $U_i^{(u)}$, $\mathrm{Rare}_i^{(u)}$, $\mathrm{PastPos}_i^{(u)}$, and $h_i^{(u)}$ using only records available by time u
  if $\mathrm{LCB}_i^{(u)} > 0$ **then**
    $g_i^{(u)} \leftarrow$ ACTIVE
  else if $U_i^{(u)} = 1$ **or** $\mathrm{Rare}_i^{(u)} = 1$ **or** $\mathrm{PastPos}_i^{(u)} = 1$ **then**
    $g_i^{(u)} \leftarrow$ FROZEN
  else if $\mathrm{LCB}_i^{(u)} < -\epsilon$ **and** $\mathrm{LCB}_i^{(u^-)} < -\epsilon$ **and** $U_i^{(u)} = U_i^{(u^-)} = 0$ **and** $h_i^{(u)} = 0$ **then**
    $g_i^{(u)} \leftarrow$ RETIRED
  else
    $g_i^{(u)} \leftarrow$ FROZEN
  end if
end for
for all behaviorally equivalent pairs $s_i \equiv s_j$ **do**
  if $g_i^{(u)} =$ ACTIVE **and** $g_j^{(u)} =$ ACTIVE **then**
    Freeze the skill with the smaller lower confidence bound; break an exact equality by retaining s_i when $\mathrm{LCB}_i^{(u)} \ge \mathrm{LCB}_j^{(u)}$
  end if
end for
Update routine retrieval eligibility: include active skills, exclude frozen skills while retaining their representations, and remove retired executable bodies while retaining summaries, provenance, audit statistics, and lineage
Record all estimates, predicates, state assignments, and retained metadata
return $\{g_i^{(u)}, \widehat{\Delta}_i^{(u)}, \widehat{\sigma}_i^{(u)}, \text{decision record}_i\}_{s_i \in \mathcal{C}_u}$