FOCUSED BUT FRAGILE: DIAGNOSING TAIL-RULE LOSS UNDER RECURSIVE AGENT SKILL COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Compression can preserve an agent skill's routine while omitting the rule needed for an exception. We study this failure mode in an artifact-only maintenance regime, where each rewrite sees only its predecessor and cannot revisit the source evidence used to create the skill. We introduce TAILSKILLS, a controlled benchmark of 208 task variants derived from 54 base tasks and 14 exception types. Each variant pairs an ordinary environment with an exception-triggering environment and is admitted only after reference-oracle execution and deterministic verification.

The benchmark reveals a selective durability gap. Among the 56 variants with detectable registered markers, unit-weighted S4 retention is 48.7% for regular workflow units but only 15.7% for tail recovery units. A separate high-precision analysis over all 208 variants reproduces the artifact-level separation across four pipelines: at S4, regular retention ranges from 31.2% to 43.9%, whereas tail retention ranges from 2.1% to 15.2%. On a fixed set of 52 tail variants, S4 success is lower than S1 in three of four tested pipelines, with endpoint drops of 3.8–7.7 percentage points; the remaining pipeline follows a non-monotone trajectory and ends at its S1 level. Paired execution audits further show that artifact loss can become task failure in some executor–task settings, but does not produce a uniform downstream drop. These findings indicate that ordinary-task success is insufficient to assess the durability of compressed procedural skills: recovery rules require direct evaluation.

1 INTRODUCTION

An LLM agent can still complete an ordinary task after its skill has lost the rare recovery rule needed to handle an exception. A data-analysis workflow may work on a clean CSV but fail when its header contains a byte-order mark; an output routine may work until its destination becomes read-only. The main procedure survives in both cases, while a short recovery rule determines whether it remains reliable. As agents increasingly rely on reusable skill files as external procedural memory for tools, checks, and recovery (Li et al., 2026; Ni et al., 2026; Wang et al., 2026; Jiang et al., 2026), this creates a gap between the usefulness of a skill on ordinary tasks and the durability of the knowledge it stores.

We study that durability under repeated compression. A complete skill is rewritten into a shorter document, which becomes the input to the next rewrite. Each step can consult its predecessor but cannot revisit the source evidence used to create the complete skill. In this artifact-only setting, an omitted rule can remain absent in later versions. A rewriter may reconstruct it from prior knowledge, but it can no longer check that reconstruction against the original source. This setting isolates the maintenance case in which the evidence used to construct a skill is no longer available during later rewriting.

Existing evaluations address parts of this question. Skill benchmarks measure the utility of a supplied artifact; robustness tests expose failures under perturbed inputs or environments; and prompt-compression methods assess utility under shorter contexts (Li et al., 2026; Ribeiro et al., 2020; Goel et al., 2021; Jiang et al., 2023; Pan et al., 2024). Measuring skill durability requires asking two related questions: what survives in the maintained artifact, and whether missing content changes execution. A rule may disappear lexically while remaining operationally recoverable, or remain present while an executor fails to use it. We therefore evaluate artifact retention, operational recoverability, and downstream behavior separately.

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.

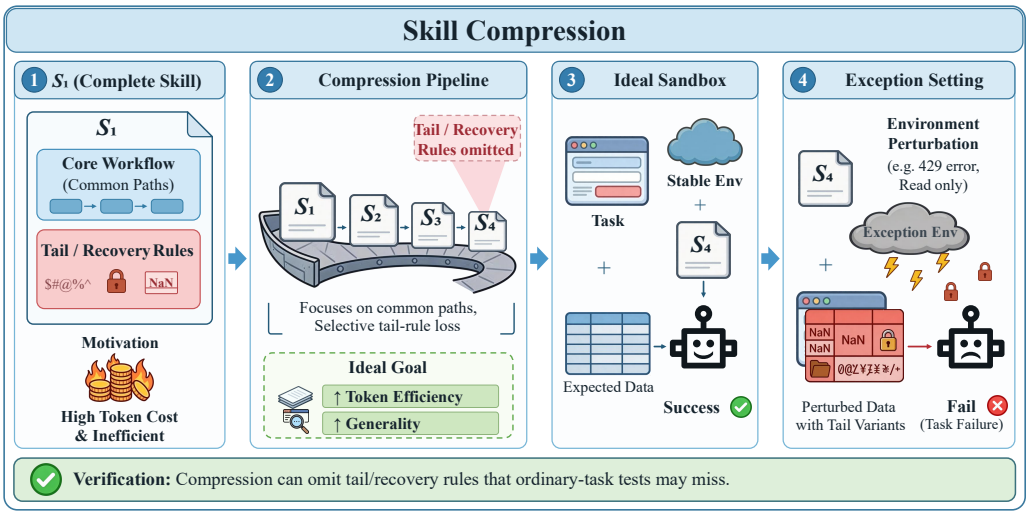


Figure 1: Our proposed TAILSKILLS benchmark evaluates whether recovery rules survive repeated artifact-only rewriting. A complete skill is rewritten through S1–S4 using only its predecessor; paired common and exception environments separate routine execution from recovery behavior.

We introduce TAILSKILLS, a controlled benchmark built around this distinction. It pairs ordinary workflows with controlled exception environments, allowing the same maintained artifact to be evaluated for what it preserves and what it fails to recover. The benchmark comprises 208 variants derived from 54 base tasks across 14 exception types; expert-defined overlays are admitted only after reference-oracle execution and deterministic verification. The suite measures responses to controlled failures, and its category frequencies do not estimate the prevalence of production incidents.

Using this benchmark, we find a selective durability gap. In the marker audit, S4 retains 48.7% of regular workflow units but only 15.7% of tail recovery units among the 56 variants with detectable registered markers. A separate high-precision analysis over all 208 variants shows the same regular–tail separation for each of the four pipelines, broadening the result beyond the marker-audited subset. On the fixed Target52 set, three of four tested pipelines finish below their S1 success rate, while GLM returns to its S1 level after an intermediate rebound. Matched one-shot and source-grounded controls test whether endpoint compression and access to source evidence change this pattern; paired execution then examines when missing recovery content becomes task failure. Together, these experiments connect what a compressed skill contains with what an executor can do with it.

Contributions. We make three contributions. First, we introduce TAILSKILLS, a controlled benchmark for testing whether maintained skills preserve recovery procedures under exceptional conditions. Second, we show that artifact-only rewriting can preferentially omit tail rules relative to regular workflow units, both in a strict marker audit and in a separate all-208-variant cross-pipeline analysis. Third, we connect artifact audits to semantic checks, matched controls, and paired execution to identify when missing recovery content becomes operationally consequential.

2 RELATED WORK

Agent skills as external procedural memory. Agent skills externalize procedural knowledge beyond model weights. SkillsBench evaluates supplied skill files across diverse tasks (Li et al., 2026); WildSkills and SWE-Skills-Bench extend evaluation to realistic and software-engineering settings (Liu et al., 2026; Han et al., 2026). Trace2Skill learns transferable artifacts from trajectories (Ni et al., 2026), while co-evolutionary and reinforcement-learning methods refine skills through feedback (Zhang et al., 2026; Xia et al., 2026). These activities form a lifecycle of discovery, distillation, storage, composition, evaluation, and update (Jiang et al., 2026). We study the durability of knowledge already stored in a skill during later maintenance.

Compression, faithfulness, and recursive information loss. Prompt-compression methods shorten context while preserving task utility (Jiang et al., 2023; 2024; Li et al., 2023; Pan et al., 2024). Summarization research shows that fluent rewrites can still be unfaithful (Maynez et al., 2020; Pagnoni et al., 2021). Recursive synthetic-data training offers a related analogy: repeated generation can reduce coverage of distributional tails (Shumailov et al., 2024). We instead study an external procedural artifact and ask which recovery instructions survive each rewrite.

Robustness evaluation under controlled perturbations. CheckList and Robustness Gym support behavioral testing through targeted perturbations and evaluation slices (Ribeiro et al., 2020; Goel et al., 2021); WILDS studies distribution shifts in real-world settings (Koh et al., 2021). We cross controlled environment changes with successive versions of a skill, holding task intent fixed and assessing execution with deterministic verifiers.

These lines of work evaluate supplied skills, compressed contexts, or robustness to perturbations. TAILSKILLS addresses the remaining maintenance question: whether recovery procedures survive repeated rewriting of the artifact that stores them. It combines paired exception environments with retention audits, matched compression controls, and execution tests.

3 PROBLEM FORMULATION

Artifact-only maintenance. For tail variant i, let x_i^c and x_i^t denote its paired common and tail environments, and let $s_{i,1}$ be a complete skill containing the ordinary workflow and the required recovery procedure. We take artifact durability as the central analysis level: the primary contrast asks whether regular workflow units and tail recovery units survive at different rates under matched compression. A rewrite operator $\mathcal{R}_\theta$ takes an artifact and a target length budget b and produces a shorter artifact:

$$s_{i,k+1}^{\mathrm{rec}} = \mathcal{R}_\theta(s_{i,k}^{\mathrm{rec}}; b_{k+1}), \qquad s_{i,1}^{\mathrm{rec}} = s_{i,1}, \qquad k = 1, \dots, K-1, \tag{1}$$

The rewrite is artifact-only: after the initial skill is constructed, later steps cannot access the original trajectory or immutable source evidence. We use $K = 4$ in the reported protocol and index artifacts rather than rewrite operations, so $s_{i,4}^{\mathrm{rec}}$ is the endpoint after three rewrites.

Matched rewrite conditions. Let a_i denote immutable source evidence that contains the original recovery constraints. To separate repeated inheritance from endpoint compression, we generate a matched one-shot endpoint directly from $s_{i,1}$:

$$s_{i,K}^{\mathrm{one}} = \mathcal{R}_\theta(s_{i,1}; b_K). \tag{2}$$

To test source access as a mitigation condition, source-grounded rewriting retains access to a_i at every step:

$$s_{i,k+1}^{\mathrm{src}} = \mathcal{R}_\theta^{\mathrm{src}}(s_{i,k}^{\mathrm{src}}, a_i; b_{k+1}), \qquad s_{i,1}^{\mathrm{src}} = s_{i,1}, \qquad k = 1, \dots, K-1. \tag{3}$$

Let $\ell(s)$ be the frozen length measure and define the normalized length for variant i as $\rho_i(s) = \ell(s)/\ell(s_{i,1})$. With frozen endpoint target $\rho_K^\star$ and tolerance ϵ_{len}, the primary endpoint comparison retains conditions satisfying

$$\left|\rho_i(s_{i,K}^{\mathrm{rec}}) - \rho_K^\star\right| \le \epsilon_{\mathrm{len}}, \qquad \left|\rho_i(s_{i,K}^{\mathrm{one}}) - \rho_K^\star\right| \le \epsilon_{\mathrm{len}}. \tag{4}$$

The same endpoint rule is used for SRC. All three conditions use the generic prompt family; the source-grounded prompt may preserve source-supported constraints but may not reveal the benchmark category or add a task-specific rule absent from a_i. Thus, ONE provides the endpoint control for repeated inheritance, whereas SRC changes access to evidence and is interpreted as a mitigation condition rather than as a second one-shot baseline.

Artifact and semantic retention. Let $\mathcal{U}^{z,\mathrm{obs}}$ and $\mathcal{U}^{z,\mathrm{sem}}$ denote, respectively, the registered units in the observable audit and the units sampled for the blinded semantic audit, with $z \in \{\mathrm{reg}, \mathrm{tail}\}$ for regular-workflow and tail-handling units. These universes need not contain the same units or variants because the semantic audit is a separately sampled adjudication package. Let $m(u, s) \in \{0, 1\}$ indicate whether unit u passes the frozen registered-unit criterion; this is a strict observable audit,

rather than a generic keyword match. Let $L(u, s) \in \{\text{retained}, \text{partial}, \text{absent}, \text{not_adjudicable}\}$ be the blinded semantic label, and let $\mathcal{U}^{z,\text{sem}}_{\text{adj}} \subseteq \mathcal{U}^{z,\text{sem}}$ contain the units with adjudicable labels. The strict semantic score counts only fully retained units; partial and absent contribute zero, while not-adjudicable units are excluded from the semantic denominator. Thus,

$$M_z(k) = \frac{1}{|\mathcal{U}^{z,\text{obs}}|} \sum_{u \in \mathcal{U}^{z,\text{obs}}} m(u, s_{i(u),k}),$$
$$S_z(k) = \frac{1}{|\mathcal{U}^{z,\text{sem}}_{\text{adj}}|} \sum_{u \in \mathcal{U}^{z,\text{sem}}_{\text{adj}}} \mathbb{1}\{L(u, s_{i(u),k}) = \text{retained}\}, \tag{5}$$
$$z \in \{\text{reg}, \text{tail}\}.$$

Here $i(u)$ identifies the artifact instance containing unit u, and $s_{i(u),k}$ denotes its depth-k artifact. The central observable contrast is

$$\Delta_M(k) = M_{\text{reg}}(k) - M_{\text{tail}}(k). \tag{6}$$

When the semantic audit provides aligned regular and tail units, the corresponding semantic contrast is $\Delta_S(k) = S_{\text{reg}}(k) - S_{\text{tail}}(k)$. In the frozen audit package, we report the adjudicated label composition and the aligned-tail estimate separately rather than treating them as a full-population semantic curve. For a condition-specific comparison at an available depth k, the same definitions apply with $s_{i(u),k}$ replaced by the corresponding artifact $s^{\gamma}_{i(u),k}$ for $\gamma \in \{\text{REC}, \text{SRC}\}$; for ONE, the corresponding quantities are evaluated only at endpoint K using $s^{\text{ONE}}_{i(u),K}$. Both $M_z(k)$ and $S_z(k)$ are unit-weighted point estimates. Uncertainty is clustered by base task, so units from the same task are not treated as independent observations; denominators and any boundary-excluded units are reported with the corresponding estimate.

Execution outcomes and discovery contrast. Let $Y_{e,a}(s, x) \in \{0, 1\}$ be the outcome of attempt a by executor e, where the outcome is determined by a deterministic verifier applied to that execution. A single-attempt result uses $Y_{e,1}$; for an A-attempt audit, the corresponding pass indicator is $\mathbb{1}\{\max_{a \le A} Y_{e,a}(s, x) = 1\}$. For a cohort of valid instances, pass@A is the mean of this indicator over the cohort. Infrastructure-invalid attempts follow the prespecified retry, held, and void rules rather than being silently counted as verifier failures.

For the original discovery protocol, let $s^c_{j,k}$ denote the depth-k recursive artifact for common task j, and let x^c_j denote its ordinary environment. We compare common-task and tail-variant success at each depth:

$$A_c(k) = \frac{1}{|\mathcal{B}|} \sum_{j \in \mathcal{B}} Y_{e,1}(s^c_{j,k}, x^c_j), \qquad A_t(k) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} Y_{e,1}(s^{\text{rec}}_{i,k}, x^t_i), \tag{7}$$

where $\mathcal{B}$ contains 54 original base tasks and $\mathcal{V}$ contains 208 admitted tail variants. The tail durability gap is

$$G_k = [A_t(1) - A_t(k)] - [A_c(1) - A_c(k)]. \tag{8}$$

A positive G_k means that tail performance deteriorates more than common performance under the same rewrite schedule. Because $\mathcal{B}$ and $\mathcal{V}$ are not identical samples, G_k is a benchmark-level descriptive contrast. The paired audit instead evaluates the same variant skill in both x^c_i and x^t_i and clusters uncertainty by base task.

4 THE TAILSKILLS BENCHMARK

TAILSKILLS separates ordinary execution from recovery under controlled exceptions while holding task intent fixed. Each item pairs a common instance with a tail instance derived from the same base task and user instruction; only the latter receives an exception-triggering overlay. We construct 208 admitted variants from 54 base tasks and 14 exception types spanning encoding, filesystem, data-boundary, network, dependency, and security faults. The common/tail pair is the unit for within-variant audits, while the discovery lane also reports separate common-task and tail-variant cohorts.

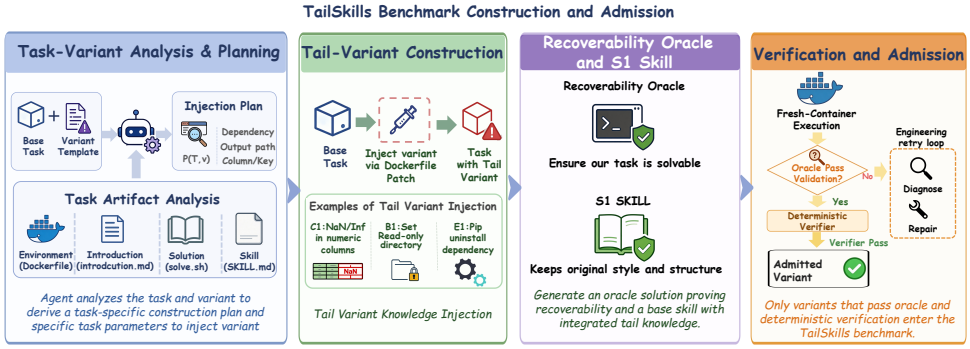


Figure 2: TAILSKILLS construction and admission. An expert exception template is instantiated as a task-specific overlay, producing paired common and tail instances. A fresh-container oracle checks that the prescribed recovery path exists, and an independent deterministic verifier checks the resulting output before the item is admitted.

Construction and admission. Experts define each exception template, its applicability conditions, recovery procedure, and verifier checks. A construction agent instantiates it as an overlay on a copied environment. A variant is admitted only after the oracle and verifier checks below, while the common counterpart remains the paired control. Admission establishes solvability under the prescribed procedure, not executor discovery or use. Evaluated executors cannot access construction utilities or oracle solutions.

What is held fixed. Within a pair, the user instruction, task intent, interface, upstream environment, and complete skill are fixed; only the exception overlay and staged fixture differ. The benchmark therefore tests recovery of an unannounced exception rather than response to an explicitly named anomaly.

Recoverability and independent verification. A fresh-container reference oracle equipped with the prescribed recovery logic must obtain full verifier credit before admission. An independent deterministic verifier then checks task-specific output invariants without replaying the oracle's action trace; neither step guarantees that an evaluated executor will discover or apply the procedure.

Provenance and scope. Each admission record links the task pair, source skill, overlay, oracle and verifier results, and relevant hashes; the field-level schema is in the appendix. Exception counts describe this constructed benchmark rather than production prevalence, and primary uncertainty is clustered by base task.

5 RECURSIVE SKILL COMPRESSION

5.1 SETUP

The protocol applies the same category-neutral compression instruction three times, targeting 100%, 70%, 49%, and 34% of the starting artifact length. Each rewrite sees only its predecessor; the prompt does not identify tail categories or inserted recovery rules.

Gemini 2.5 Flash serves as both rewriter and executor. Each condition runs once in an isolated container and is scored by a deterministic verifier; the four depths over 54 common tasks and 208 tail variants yield 1,048 runs. This discovery lane exposes the benchmark-level pattern; Section 6 tests its scope with matched controls and additional executors.

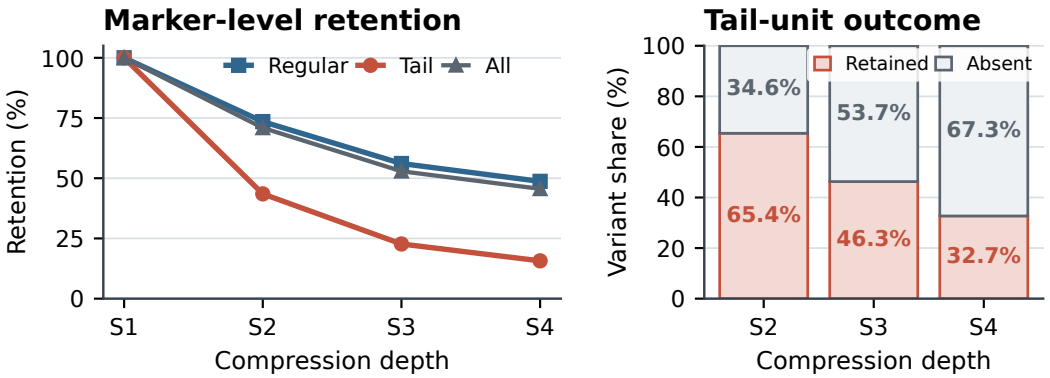


Figure 4: Selective loss of tail recovery units in a marker audit of 56/208 variants with detectable registered markers. Left: unit-weighted retention across compression depths. Right: variant-level shares with at least one versus no registered tail unit detected.

5.2 Discovery Result

In the fixed discovery lane, ordinary-task success remains comparatively stable while exception-task success declines toward the endpoint (Figure 7). This descriptive separation motivates the artifact audit below, which tests whether the divergence corresponds to selective loss of recovery content.

The corresponding success curves and benchmark-level contrast are reported in Appendix C. The matched ONE-S4 control in Section 6 tests whether repeated inheritance adds loss beyond direct compression to the same endpoint.

5.3 Tail-Rule Retention

We decompose each complete tail-aware skill into registered recovery units and the remaining regular workflow. The marker audit counts a unit as retained when its registered marker or structural signature remains detectable. For a BOM variant, for example, a generic mention of "encoding" does not satisfy the registered criterion for BOM-safe decoding or removal.

At s_4, the marker audit retains 48.7% of regular workflow units but only 15.7% of tail recovery units, a 33.0 percentage-point gap (Figure 4). These unit-weighted estimates use the 56 of 208 variants with detectable inserted tail markers. Among those 56 audited variants, 67.3% contain no detectable registered tail unit at s_4. A separate 32-item blinded semantic audit tests whether marker-level omissions correspond to missing operational content when lexical matching is insufficient.

5.4 Cross-Pipeline Artifact Retention

The strict marker audit above is deliberately restricted to the 56 variants with detectable registered markers and uses unit-weighted retention. To assess the breadth of the artifact pattern, we separately score the high-precision exception-retention measure on all 208 admitted variants at every recursive depth for the four evaluation pipelines. The scorer decomposes each complete s_1 artifact into high-precision tail-handling units and remaining regular content; a unit is retained only when its concrete handling step remains recoverable, while keyword-only matches do not count. This is a distinct all-208 analysis with unit-weighted retention and a variant-level tail-absent rate; it is not pooled with the 56-variant marker denominator.

Table 1 reports the endpoint values. At s_4, regular retention exceeds tail retention in every pipeline, by 20.9–41.8 percentage points, and the tail-absent share is high in all four lanes. The same ordering is already visible at s_2 and becomes more pronounced toward s_4 (the complete depth-wise table is in Appendix E). Thus, the strict marker audit supplies the fine-grained registered-unit result, while the

Table 1: All-208 S4 retention. Total, Regular, and Tail are unit-weighted; Absent is the share of variants with no retained tail unit. All entries are percentages.

Pipeline	Total	Regular	Tail	Absent
GLM-5	34.6	36.8	4.0	94.6
DeepSeek V4 Flash	30.0	31.2	10.3	73.2
Qwen3-Max	36.8	38.5	15.2	71.4
GPT-5.4	41.5	43.9	2.1	91.1

Table 2: Illustrative failure modes. The main workflow remains while the local recovery rule is omitted.

Exception	Rule lost	Failure
BOM	BOM-safe decoding	Column mismatch
Read-only	Permission recovery	Missing output
Extreme values	Sentinel filtering	Wrong result

all-208 analysis extends the same artifact-level separation beyond that audited subset and across four pipelines.

6 Controls and Behavioral Boundaries

The discovery result raises four questions: is the recovery package needed, does the contrast persist under matched length and source access, does endpoint loss transfer across pipelines, and when does an artifact difference become execution failure? We address them in that order and close with an exploratory probe.

6.1 Package-Level Necessity

We select cases where the complete skill succeeds and replace the augmented skill with its original SkillsBench counterpart, removing the added recovery rule and helper logic. The complete skills pass 48/48 valid runs; the replacements pass 0/48. This supports the necessity of the complete augmented package on the selected subset. Because selection depends on s_1 success and the replacement removes more than one instruction, the comparison does not isolate the effect of deleting a single rule during compression.

6.2 Matched-Length and Source-Grounded Controls

We add two endpoint controls. ONE-S4 rewrites the full s_1 artifact directly to the same endpoint-length window used for REC-S4, allowing repeated inheritance to be compared with direct compression at a matched endpoint. SRC-S4 follows the recursive schedule while retaining access to immutable source evidence. The matched audit uses the same task manifest, rewrite budget, endpoint-length tolerance, and structural validity rules across conditions; the detailed chain ledger and held-row accounting are reported in Appendix E.

Generation QC leaves 533/612 DeepSeek chains and 577/612 GLM chains available for condition comparisons (Appendix C). These counts establish which outputs are available for comparing rewrite conditions; they do not by themselves establish a content advantage for ONE-S4 or SRC-S4. Appendix E records the model-specific counts, and Table 4 gives the interpretation rules.

6.3 Cross-Pipeline Endpoint Durability

The preceding analysis is artifact-level. We next ask whether the same endpoint pattern is visible in execution on a fixed task set. Target52 contains the same 52 tail variants at every depth for GPT-5.4, DeepSeek V4 Flash, GLM-5, and Qwen3-Max. Strict Pass@1 requires reward equal to one. From S1 to S4, success falls by 3.8 pp for GPT-5.4, 7.7 pp for DeepSeek V4 Flash, and 3.8 pp for Qwen3-Max. GLM-5 returns to its S1 level after improving at intermediate depths. Thus, endpoint durability loss appears in three of four tested pipelines, while the depth trajectory is not uniformly monotone. The complete trajectories and counts are reported in Appendix E. We report the four lanes separately because the evaluated pipeline includes both the executor and its harness.

6.4 Semantic Qualification of Marker Loss

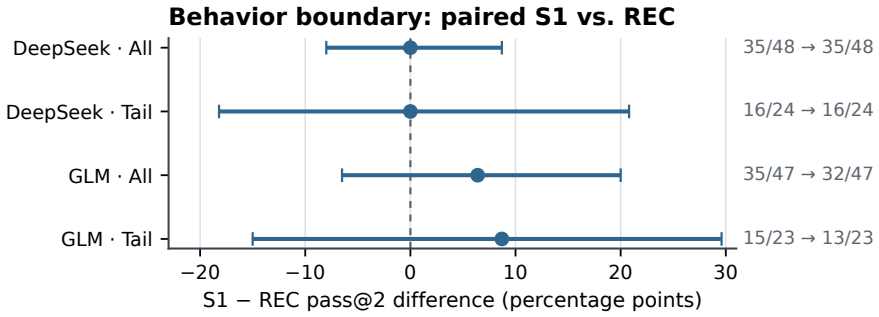


Figure 6: Paired behavior audit. Points show the S1–REC difference in pass@2 percentage points; positive values favor the complete skill. Intervals are clustered by base task, and labels give the exact S1/REC pass counts. GLM shows positive point estimates in both cohorts, whereas DeepSeek is unchanged in this audit.

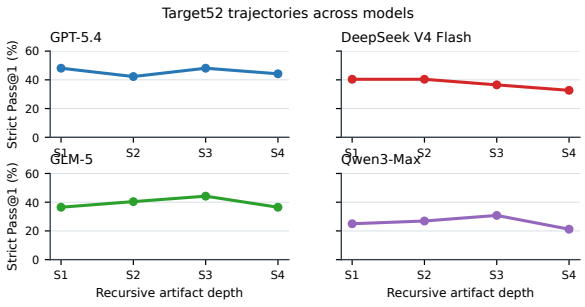


Figure 5: Compact Target52 trajectories across four pipelines. The same tail set is evaluated at each recursive depth; the curves emphasize trajectory shape rather than exact rates.

The strict marker audit is complemented by a frozen 32-item, two-rater blind audit: 103/124 adjudicable units are retained (83.1%), 18 partial, 3 absent, and 4 not adjudicable; the aligned tail subset retains 19/23 (82.6%). This is a semantic qualification on a frozen sample; annotation rules are in the appendix.

6.5 Two-Executor Behavior Audit

The paired audit compares the complete and compressed skills on a frozen core of 24 task variants, each evaluated in common and tail environments. S1 and REC each receive two attempts with DeepSeek and GLM executors. DeepSeek has identical S1 and REC pass@2 counts (35/48); GLM has 35/47 versus 32/47, a +6.4 pp S1–REC difference. We estimate uncertainty with 10,000 base-task cluster-bootstrap replicates using seed 20260901. The GLM 95% interval is [-6.5, +20.0] pp and includes zero. The exact paired counts, clustered intervals, and tail-only comparison are reported in Appendix K.

Four GLM rows and one DeepSeek row switch from S1 pass to REC failure. These cases provide a starting point for inspecting execution, but the outcome change alone does not identify its cause. We examine the available traces next.

6.6 Observable Trace Analysis

In two S1-pass/REC-fail cells (pdf-C1 and xlsx-B3), eight complete trajectories contain no `SKILL.md` reads or skill-script use. The executor was expected to discover the mounted skill through the filesystem, but the S1 and REC system-plus-user contexts were byte-identical because the skill was not injected into the prompt. The failures were executor-side: NaN values were counted as edits in pdf-C1, and a derived-mean calculation was wrong in xlsx-B3. These traces identify executor-side error and skill discoverability as boundary conditions rather than evidence of skill-content mediation.

The broader trace package does not provide the same process coverage for every outcome. We therefore treat this mechanism account as case evidence, not as a population rate; full counts, event schemas, and per-run joins are in Appendix L.

6.7 Exploratory Mitigation Probe

As a separate exploratory intervention, we test whether a generic instruction to preserve recovery procedures improves compressed skills at a matched length. On a 30-variant subset, the tail-aware prompt increases S4 success from 6/29 (20.7%) under the regular prompt to 11/30 (36.7%), a 16.0 percentage-point difference. S2 and S3 do not improve, and the intervention does not form a complete S1–S4 lineage or establish common-task utility. We therefore treat this result as a mitigation signal for further testing; the appendix reports the regular and tail-aware results by depth.

7 Discussion

Regular workflow units survive artifact-only maintenance more often than recovery units. The 56-variant marker audit measures this gap under a strict registered-unit criterion; the separate all-208 analysis finds the same ordering across four pipelines. The semantic audit adds a distinction: content that fails an observable criterion can remain operationally specified. Direct inspection of recovery procedures therefore complements both marker retention and task success.

REC, ONE, and SRC define comparisons of repeated inheritance, endpoint length, and source access. Their generation ledger establishes coverage for these comparisons. Target52 supplies complementary execution evidence: three of four pipelines end below their S1 success rate, although the depth trajectories are not monotone across all pipelines.

Paired execution identifies S1-pass/REC-fail cases, while traces show why outcome changes require closer inspection: the executor never read the mounted skill in the fully traced cases. Recovery durability depends on both preserving an instruction and making it available during execution. The exploratory tail-aware prompt suggests a possible intervention that warrants testing across complete lineages and ordinary tasks.

Limitations. TailSkills uses controlled exception overlays rather than a naturally sampled distribution of production incidents, so its category counts describe the benchmark rather than deployment prevalence. The marker audit uses a strict registered-unit criterion, and the blinded semantic audit covers a frozen subset; operationally equivalent paraphrases can therefore be missed outside that audit. Finally, the trace and mitigation studies are targeted follow-up analyses: they identify behavioral boundaries and promising directions without serving as population-wide mechanism or causal estimates.

8 Conclusion

TailSkills makes recovery durability an explicit evaluation target for maintained agent skills. Under artifact-only recursive maintenance, regular workflow units are preserved more often than recovery rules in both the strict marker audit and the all-208 cross-pipeline analysis, while fixed-tail endpoint tests show decreases in three of four evaluated pipelines. Paired audits identify concrete task settings in which the complete skill succeeds while the compressed skill fails. These results show why ordinary workflow success alone is insufficient: recovery procedures must be evaluated directly when judging whether a maintained skill remains dependable.

Reproducibility Statement

The estimands and interpretation rules are defined in the main text and Appendix A; benchmark construction and admission are specified in Appendix B; prompt, length, artifact-QC, and cross-pipeline accounting are given in Appendices C–E; and the behavior, trace, and semantic-audit schemas are provided in Appendices K–M. The supplementary package contains the frozen 208-item manifest, artifact ancestry and hashes, model and prompt settings, verifier outputs, trace records, and scripts for

the reported tables. Discovery-lane artifacts and matched-control runs remain separate provenance lanes and are never pooled into one denominator.

AI USE STATEMENT

Generative AI tools were used to instantiate synthetic task variants from expert-defined templates; assist with benchmark and experiment design, code, analysis, and result interpretation; rewrite skill artifacts; execute benchmark tasks; prepare figures and tables; and draft or edit prose. Human authors defined the research questions, perturbation taxonomy, admission rules, experimental freezes, statistical analyses, and final claims. Model-generated variants were admitted only through deterministic oracle verification, and all AI-assisted code, analyses, artifacts, figures, and text were checked against the underlying tests, artifacts, and manifests. The authors take responsibility for the final submission and its claims.

REFERENCES

Karan Goel, Nazneen Rajani, Jesse Vig, Samson Tan, Jason Wu, Stephan Zheng, Caiming Xiong, Mohit Bansal, and Christopher Ré. Robustness gym: Unifying the NLP evaluation landscape. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, 2021. URL https://aclanthology.org/2021.naacl-demos.6/.

Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. SWE-Skills-Bench: Do agent skills actually help in real-world software engineering?, 2026. URL https://arxiv.org/abs/2603.15401.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://aclanthology.org/2023.emnlp-main.825/.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.91/.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. SoK: Agentic skills—beyond tool use in LLM agents, 2026. URL https://arxiv.org/abs/2602.20867.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, et al. WILDS: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 5637–5664, 2021. URL https://proceedings.mlr.press/v139/koh21a.html.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks, 2026. URL https://arxiv.org/abs/2602.12670.

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://aclanthology.org/2023.emnlp-main.391/.

Yujian Liu, Jiabao Ji, Li An, Tommi Jaakkola, Yang Zhang, and Shiyu Chang. How well do agentic skills work in the wild: Benchmarking LLM skill usage in realistic settings, 2026. URL https://arxiv.org/abs/2604.04323.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1906–1919, 2020. URL https://aclanthology.org/2020.acl-main.173/.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills, 2026. URL https://arxiv.org/abs/2603.25158.

Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4812–4829, 2021. URL https://aclanthology.org/2021.naacl-main.383/.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, 2024. URL https://aclanthology.org/2024.findings-acl.57/.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912, 2020. URL https://aclanthology.org/2020.acl-main.442/.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. AI models collapse when trained on recursively generated data. *Nature*, 631:755–759, 2024. doi: 10.1038/s41586-024-07566-y. URL https://www.nature.com/articles/s41586-024-07566-y.

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

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, 2026. URL https://arxiv.org/abs/2602.08234.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. Coevoskills: Self-evolving agent skills via co-evolutionary verification, 2026. URL https://arxiv.org/abs/2604.01687.

A Experimental Map and Estimands

Table 3 lists the comparison and measurement used in each experiment. The following sections give the model settings, sample sizes, annotation criteria, and execution records.

Table 3: Experimental comparisons and primary measurements.

Experiment	Generation and evaluation	Research question	Measurement
Discovery	Gemini; recursive S1–S4; one attempt per condition	Does recursive compression separate common and tail performance?	Success curves, common–tail gap, marker retention
REC-S4	DeepSeek/GLM; recursive endpoint	What survives repeated inheritance at the matched endpoint?	Structural QC and semantic retention
ONE-S4	DeepSeek/GLM; direct $s_1 \to s_4$ rewrite	Is endpoint loss explained by final length alone?	Matched endpoint contrast
SRC-S4	DeepSeek/GLM; recursive rewriting with source access	Does immutable source evidence preserve recovery units?	Source-grounded mitigation contrast
Cross-pipeline artifact	Four pipelines; all 208 variants at S1–S4	Does the artifact pattern recur beyond the marker-audited subset?	Unit-weighted retention and tail absence
Behavior	DeepSeek/GLM executors; paired S1/REC; two attempts	When does artifact loss become execution failure?	pass@2 and task-level flips
Target52	Four pipelines; same 52 tail variants at S1–S4	Does the endpoint pattern transfer across pipelines?	Strict Pass@1 trajectories
Tail-aware	Qwen; separate 30-variant, length-matched intervention	Can generic preservation improve a deep tail endpoint?	Regular/intervention contrast
Semantic	32 frozen items; two blinded raters	Does marker loss survive operationally equivalent paraphrase?	Adjudicated semantic retention

Table 4: Prespecified interpretation rules for the robustness experiments.

Observed pattern	Interpretation
Regular loss $>$ tail loss	Do not support selective tail loss; inspect semantic audit and protocol validity.
Tail loss across pipelines	Report the all-208 artifact-level asymmetry separately from the 56-variant marker audit.
REC $<$ matched ONE	Supports a recursion-specific contrast.
REC $\approx$ ONE	Report compression fragility without a recursion-specific claim.
Artifact loss without behavior loss	Report executor/task-dependent behavioral boundary.
Semantic audit disagrees with marker audit	Restrict the claim to marker/structural loss.
SRC improves tail and preserves common performance	Report source-grounded mitigation; otherwise report the result as negative or inconclusive.

B Benchmark Construction and Controlled Overlays

Each tail variant is constructed by copying the base-task environment and applying a registered overlay to the copy. The user-facing `instruction.md`, task identity, complete skill, and verifier contract are held fixed within the paired common/tail evaluation; only the exception-triggering overlay and its associated fixture differ. The same complete skill is therefore evaluated in both members of each pair. The overlay changes an environmental condition that the corresponding recovery procedure is designed to handle, while the recoverability oracle establishes that the task remains solvable under that condition.

The construction record for each admitted variant contains the applicability predicate, overlay or fixture hash, expected recovery procedure, oracle result, and verifier result. Table 5 gives representative examples. These examples describe the construction mechanism, not category prevalence.

Table 5: Representative controlled overlays. The instruction and complete skill are shared by the paired common and tail environments; only the listed overlay/fixture is changed.

Type	Overlay or fixture change	Required recovery	Verifier invariant
A1	Prepend a UTF–8 BOM to a task-relevant text header.	Detect or strip the BOM before column lookup.	The original output and schema checks still apply.
B1	Make the designated output location read-only.	Detect the write failure and use the permitted recovery path before writing.	The output must be produced at the prescribed location and pass the original checks.
C1	Insert NaN or infinite values into task-relevant fields.	Detect non-finite values before computation and apply the task-specific cleanup path.	The original computation and numerical sanity checks must pass.
C4	Insert finite placeholder values or other semantically extreme values into task-relevant fields.	Apply domain-aware range validation before computation, then filter or impute according to the task procedure.	The original computation and numerical sanity checks must pass.
E1	Remove a required dependency from the execution image.	Check availability and restore the dependency before importing or running the task.	The task must complete with the original verifier and output contract.

The principal denominators are fixed by the corresponding analysis level: the marker audit uses the 56 of 208 variants with detectable registered tail markers; the blinded semantic package contains 32 selected items and 124 adjudicable units; Target52 contains the same 52 variants at every depth and pipeline; the paired behavior core contains 24 variants evaluated in common and tail environments with symmetric S1/REC attempts; and the package-level necessity check contains 48 valid runs per skill condition. Matched artifact QC plans three condition chains for each of 204 runnable variants, or 612 chains per rewriter, with four held variants outside that denominator.

The oracle is a variant-aware reference execution with the minimal recovery logic needed for the overlay. It establishes that the exception is solvable under the prescribed procedure; it does not guarantee that an evaluated executor will discover or apply that procedure. Candidate variants are admitted only after a fresh-environment oracle pass and deterministic verifier success. Static callgraph and infrastructure checks remove rows whose apparent success or failure cannot be attributed to the registered task condition.

C Artifact QC and Marker Retention

Marker retention. Table 6 gives the depth-wise values plotted in Figure 4. The unit-weighted columns measure surviving registered content; the last column records how often no detectable tail unit remains within the audited variants.

Table 6: Marker-retention audit on the 56 of 208 variants with a detectable registered tail marker. Total, Regular, and Tail are unit-weighted retention rates; Tail absent (variants) is the variant-weighted fraction with no recoverable registered tail unit within this audited subset.

Depth	Total	Regular	Tail	Tail absent (variants)
s_1	100.0%	100.0%	100.0%	–
s_2	70.9%	73.5%	43.5%	34.6%
s_3	52.9%	56.1%	22.7%	53.7%
s_4	45.6%	48.7%	15.7%	67.3%

Representative failure modes. The qualitative examples summarized in Section 5 are illustrative instances from admitted benchmark categories, not estimates of category prevalence. The compressed artifact can keep the main recipe while dropping the local rule activated by the tail environment.

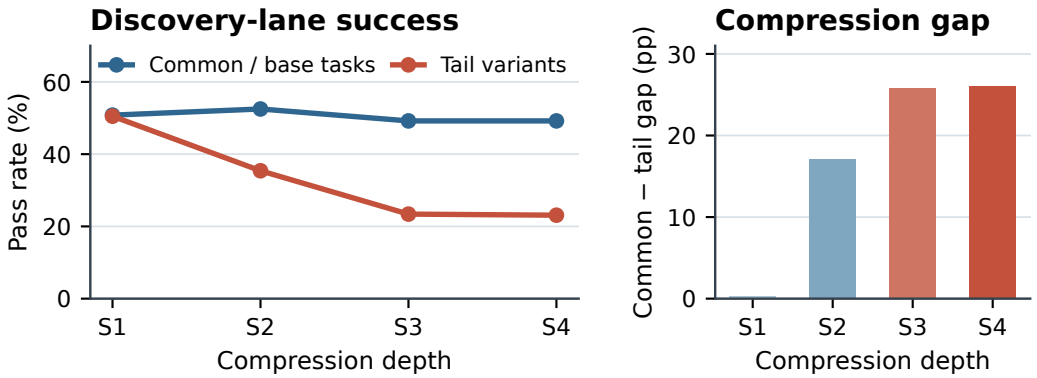


Figure 7: Discovery-lane success under repeated skill rewriting. Common and tail cohorts follow different trajectories across compression depth; the right panel shows their benchmark-level contrast.

Table 7: Qualitative failure modes. The compressed artifact keeps the main recipe but drops the local rule activated by the tail environment.

Variant	Workflow retained	Rule removed	Failure
BOM	Load CSV; aggregate columns	BOM-safe decoding	Column mismatch
Read-only dir	Write required output	Permission repair/fallback	Missing output
Extreme values	Run attribution workflow	Filter sentinel values	Plausible but wrong result

Matched-control generation. For each rewriter, the plan contains three condition chains for each of 204 runnable variants, or 612 chains in total. DeepSeek completes 533/612 under strict QC. Fifty of the 54 fenced outputs pass the prespecified lenient content check, giving 583/612 content-valid chains in the sensitivity analysis. The strict count is the primary one.

GLM completes 577/612 chains under strict QC. The other rows contain 30 empty responses, four structural/name-validity failures, one fence failure, and two length-kept failures. Two infrastructure failures were recovered through recorded retries. GLM uses an additional instruction against code fences within the shared prompt family.

Four canonical variants were held rather than counted as failures: E1_missing_dep and E2_version_drift for `fix-visual-stability`, and D1_dns_jitter and E1_missing_dep for `software-dependency-audit`. They sit outside the planned 204-variant, 612-chain ledger and are reported separately.

Table 8: Matched-control artifact generation. Entries are complete chains from the archived 612-chain ledger per rewriter, covering 204 runnable variants. Four held variants are tracked separately and are not part of this denominator. Completion is structural/length QC, not semantic retention.

Rewriter	Condition	Complete	QC completion	Median endpoint
DeepSeek	REC-S4	191/204	93.6%	49.7% of S1
DeepSeek	ONE-S4	196/204	96.1%	49.4% of S1
DeepSeek	SRC-S4	146/204	71.6%	54.7% of S1
DeepSeek	All	533/612	87.1%	–
GLM	REC-S4	191/204	93.6%	53.32% of S1
GLM	ONE-S4	201/204	98.5%	50.59% of S1
GLM	SRC-S4	185/204	90.7%	52.82% of S1
GLM	All	577/612	94.3%	–

D PACKAGE-LEVEL NECESSITY CHECK

The deletion-style check uses a selected subset of variants for which the complete augmented skill succeeds. The replacement condition uses the corresponding original SkillsBench skill, leaving the task environment, verifier, and ordinary task workflow unchanged while removing the added tail recovery package and its helper logic. All 48 complete-skill runs pass, whereas all 48 valid replacement runs fail. This is package-level necessity evidence: the comparison does not isolate one sentence or one helper script as the unique cause.

Table 9: Package-level necessity check on the selected valid subset. The replacement removes the augmented recovery package; it is not a single-rule deletion ablation.

Skill condition	Valid runs	Pass
Complete augmented skill	48	48/48 (100.0%)
Original counterpart	48	0/48 (0.0%)

E CROSS-MODEL REPLICATION DETAILS

Here we record two related but distinct cross-pipeline analyses. The first is the high-precision exception-retention analysis over all 208 admitted variants; the second is the structural completion ledger for the matched REC/ONE/SRC artifact lanes. Appendix C gives the generation coverage and held rows; Table 4 gives the interpretation rules. Results remain separated by model, denominator, and protocol.

Table 10: Depth-wise all-208-variant cross-pipeline exception retention. The shared s_1 row is the complete-skill baseline. Total, Regular, and Tail are unit-weighted high-precision retention rates; Tail absent is the variant-level share with no retained tail unit.

Pipeline	Depth	Total	Regular	Tail	Tail absent
Shared baseline	s_1	100.0%	100.0%	100.0%	–
GLM-5	s_2	68.5%	70.5%	52.4%	30.4%
	s_3	46.3%	49.1%	7.8%	87.5%
	s_4	34.6%	36.8%	4.0%	94.6%
DeepSeek V4 Flash	s_2	59.5%	61.6%	31.5%	37.5%
	s_3	41.4%	43.1%	15.0%	64.3%
	s_4	30.0%	31.2%	10.3%	73.2%
Qwen3-Max	s_2	63.6%	65.3%	42.5%	30.4%
	s_3	46.0%	47.8%	26.6%	50.0%
	s_4	36.8%	38.5%	15.2%	71.4%
GPT-5.4	s_2	73.2%	75.8%	54.2%	28.6%
	s_3	53.2%	56.1%	14.3%	73.2%
	s_4	41.5%	43.9%	2.1%	91.1%

The all-208 analysis reproduces the regular–tail ordering at every recursive depth for all four pipelines. It is reported as a breadth and replication result, not pooled with the unit-weighted 56-variant marker audit in Section 5.

The confirmatory subset uses a fixed seed over exception-family, depth, and base-task strata, chosen before inspecting outputs. The delivered Qwen package is different: it is a 48-slot expedited executor-only diagnostic. All slots pass lineage, loader, and infrastructure QC, but none belongs to the original protocol or pass@2, and there is no Qwen-generated S1→REC/ONE/SRC artifact lane. B04–B06 are floor/ceiling tasks and B01 has no frozen binary threshold, so we keep these results as exploratory boundary evidence.

E.1 TASK SUCCESS AND TAIL-AWARE COMPRESSION

Table 12 gives the counts behind Figure 5. All four model pipelines use the same 52 tail variants at every depth. Strict Pass@1 counts only reward equal to one; partial rewards count as failures.

Table 11: Matched-control artifact-lane accounting. Structural completion and exploratory roles are reported separately; "no matched-control lane" does not exclude the separate all-208-variant retention analysis above.

Rewriter	Planned units	Structural status	Artifact role	Tail–regular gap
DeepSeek	612 chains	533 strict-complete	confirmatory artifact lane	not pooled here
GPT (fixed snapshot)	executor-only plan	no matched-control lane	all-208 retention only	not estimated
GLM	612 chains	577 strict-complete	confirmatory artifact lane	not pooled here
Qwen	48 slots	0/48 matched-control protocol; executor-only	exploratory; all-208 retention separate	not estimated
Gemini (discovery)	208 variants	discovery lane	historical benchmark contrast	descriptive only

Table 12: Strict Pass@1 on Target52. Each entry gives successful runs, valid runs, and the corresponding percentage. The task set is the same in every cell.

Pipeline	S1	S2	S3	S4
GPT-5.4	25/52 (48.1%)	22/52 (42.3%)	25/52 (48.1%)	23/52 (44.2%)
DeepSeek V4 Flash	21/52 (40.4%)	21/52 (40.4%)	19/52 (36.5%)	17/52 (32.7%)
GLM-5	19/52 (36.5%)	21/52 (40.4%)	23/52 (44.2%)	19/52 (36.5%)
Qwen3-Max	13/52 (25.0%)	14/52 (26.9%)	16/52 (30.8%)	11/52 (21.2%)

The Qwen tail-aware experiment compares two rewrite prompts on 30 variants at the same nominal depth and length budget. Regular versus tail-aware success is 8/29 versus 7/30 at S2, 11/30 versus 10/30 at S3, and 6/29 versus 11/30 at S4. The +16.0 pp S4 difference is an exploratory prompt comparison. Each prompt generates its own sequence of skills.

Supplementary repeats use different protocols and are reported separately; they do not form a uniform two-attempt evaluation of Target52.

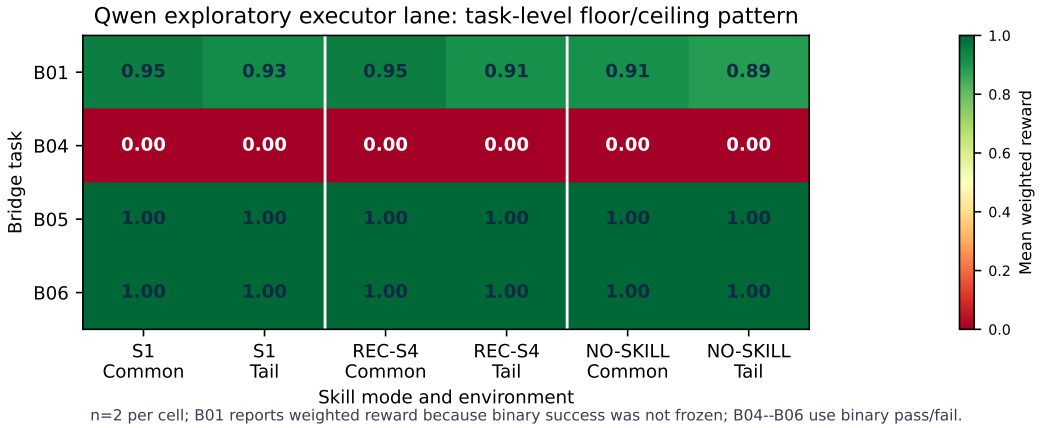


Figure 8: Exploratory expedited Qwen executor lane. Cells show weighted reward for B01 and binary success for B04–B06 across S1, REC-S4, and NO-SKILL in common and tail environments; each cell contains two attempts. The lane is excluded from the confirmatory denominator and pass@2 estimates.

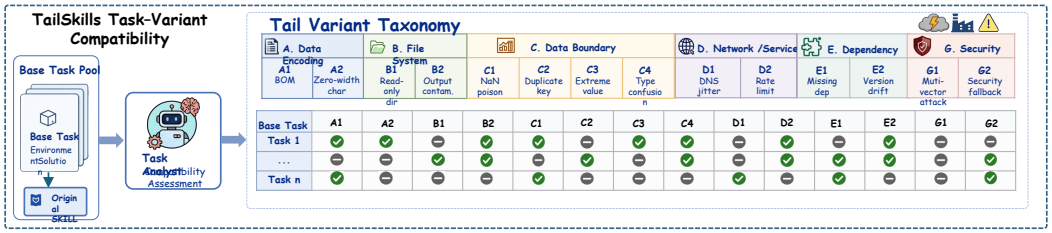


Figure 9: Task–variant compatibility and taxonomy used in benchmark construction. The applicability matrix records which base tasks support each registered exception type; it is a construction aid and does not represent production incidence.

F Benchmark Taxonomy and Applicability

The 14 registered exception types are grouped into encoding and text corruption (BOM and zero-width characters), filesystem and output faults (read-only directories and output contamination), data-boundary violations (NaN/Inf, duplicate keys, extreme values, and type confusion), network and service faults (DNS jitter and throttling), dependency faults (missing packages and version drift), and security or multi-vector failures. Each variant is represented by a base task, an applicability predicate, a staged overlay, an expected recovery procedure, and a deterministic verifier. The public manifest records these fields without exposing coordinator-only semantic mappings.

The benchmark uses controlled templates. It measures whether a maintained artifact preserves registered operational constraints; it does not estimate how often these incidents occur in production. Small category slices are descriptive, and confirmatory uncertainty is clustered by base task.

G Task–Variant Admission and Verifier Audit

An item enters the benchmark only after a fresh-environment oracle pass, a matching instruction/overlay record, and a deterministic verifier outcome. We retain the static callgraph audit as a validity check because some candidate scripts contain unconditional exits, runtime dependency installation, network calls, or indirect tail assertions. Those rows are kept out of the valid-failure count. The frozen manifest records admitted overlays, rows needing review, held rows, and exploratory rows, together with the per-variant callgraph, verifier hash, fixture hash, and admission decision.

H Distillation Prompts and Length Rules

The confirmatory artifact lanes use the provider-neutral prompt profile frozen on 2026-09-03. The recursive schedule is 70%, 70%, and 80% of the preceding normalized body for REC and SRC, giving a theoretical endpoint of 39.2% of the S1 body. ONE starts from S1 and targets 39% directly. All requests use the same ± 15 percentage-point request window and the paired endpoint comparison applies the frozen $\leq 5\%$ endpoint tolerance. The length measure is `body-chars-v1`: UTF–8 is decoded, line endings are normalized, the first YAML front matter block is removed, and Unicode

code points in the remaining body are counted. Front matter is required but is not part of the primary length denominator.

System message

```
You are an expert editor of agent skill documentation. Rewrite the skill for
concision while preserving every operationally necessary instruction,
constraint, edge case, and recovery step. Do not invent facts or tools.
Preserve a valid YAML front matter block with the original name and a concise
description; the front matter is required and is not counted in the body
length. Output only the rewritten Markdown and no commentary.
```

User message template

```
Rewrite the following agent skill to approximately {target_pct}% of its
original body length.

The normalized body (after CRLF-to-LF normalization and removal of the first
YAML front matter block) is {body_chars} Unicode characters. Produce between
{min_body_chars} and {max_body_chars} normalized body characters.
Preserve the YAML front matter with a valid name and description. Preserve task
intent, required file names and paths, tool constraints, validation steps,
failure handling, and rare/exception cases. Remove only redundancy,
repetition, and non-operational prose. Do not add a BOM hint, hidden files, or
unsupported tools.

Original skill:

'''markdown
{skill_content}
'''

Return only the rewritten Markdown.
```

For SRC, the immutable S1 source is appended in a second fenced Markdown block with the instruction to use it only to restore information present there and not to add unrelated facts. REC receives no source evidence beyond its current input. ONE starts from S1 and uses the same endpoint budget as final REC. The output contract rejects BOMs, invalid front matter, commentary prefixes, unsupported paths or tools, and any length/QC violation; retries preserve the original attempt lineage and are not silently merged with evaluation attempts.

The Qwen exploratory lane uses a separate target-relative $\pm 15\%$ window and provider-default reasoning. Its artifacts do not form a complete S1–S4 lineage and are not treated as directly matched to the DeepSeek or GLM artifact lanes.

I Models, Infrastructure, and Provenance

Gemini 2.5 Flash serves as both rewriter and executor in the discovery experiment. The DeepSeek artifact experiment requested `deepseek-v4-flash`; the provider resolved all 1,327 requests to `deepseek-flash`. Temperature was zero and the output budget was 65,536 tokens; the request ledger records no 429, timeout, empty-body, or transport errors. GLM uses the shared prompt family with an additional instruction against code fences. Qwen uses the recorded `qwen3.8-max-0902` snapshot and provider-default reasoning.

For each experiment, model configuration is linked to task and condition identifiers, input and output artifacts, container metadata, and verifier outcomes. Requested and provider-resolved model names are recorded separately. Content hashes identify the exact prompts, fixtures, artifacts, and analysis rules used; their values belong to the reproducibility manifest rather than the narrative description.

J Release Contents and Reproducibility

The release package contains the benchmark fixtures and admission records, compression prompts and settings, artifact ancestry, evaluation outcomes, semantic annotations, and analysis scripts. Stable task, condition, and attempt identifiers link each estimate to its inputs. The manifest records content hashes

and analysis-rule versions; anonymous submission materials omit private coordinator mappings and local paths.

K Behavior Audit Details

We classify behavior outcomes as valid pass, valid fail, infrastructure retryable, held, or void. Qualification attempts and infrastructure retries are not second stochastic attempts. S1 and REC use the same number of attempts in each reported comparison. Bootstrap intervals resample base tasks and then variants within each task, rather than treating generated variants as independent.

For the paired common-to-tail S1 comparison, the point estimates are $+12.5\,\mathrm{pp}$ for DeepSeek and $+17.4\,\mathrm{pp}$ for GLM; both intervals cross zero. We retain task-level collapse cases for trace analysis and do not use them to define a new confirmatory subset.

L Trace Schema and Mechanism Analysis

The trace records show which instructions and tools the executor used. Run and attempt identifiers link each trajectory to its task, environment, skill artifact, model configuration, and verifier result. Events include skill loading, tool calls, file operations, environment changes, verifier execution, retries, and termination. This lets us separate a missing instruction from a failure to discover or consume the supplied skill.

The mechanism categories cover missing recovery knowledge, incorrect parameters or paths, omitted validation, wrong tool choice, format or structural invalidity, verifier failure, and infrastructure failure. A mechanism label requires process evidence; an outcome alone is not enough. The standardized package contains 604 records: 280 passes, 134 infrastructure failures, and 190 valid failures without the fields needed for classification. These counts support outcome accounting, not aggregate mechanism rates.

The targeted trajectory audit covers 20 runs from two flip cells and two control cells. Complete trajectories are available for eight runs in the flip cells pdf-C1 and xlsx-B3. All eight contain no `SKILL.md` reads or skill-script use, and their S1/REC initial contexts are byte-identical. Table 13 lists the observed errors. Control-cell traces are less complete, so the mechanism interpretation applies only to the fully traced cases.

Table 13: Observed errors in the fully traced S1-pass/REC-fail cells. Eight trajectories across these cells show no skill-file or skill-script use; the observed failures therefore do not establish mediation by the compressed skill content.

Cell	Observed execution error	Interpretation
pdf-C1	NaN values counted as edits	Executor-side handling of missing values
xlsx-B3	Incorrect derived-mean calculation	Executor-side numerical error

M Blinded Semantic Audit

The semantic audit covers 32 selected items. Each item contains one tail unit and three regular or condition-control units. Two authors label every unit independently without seeing the model, condition, or behavior outcome. They check whether the artifact states the boundary condition, procedure and ordering, and verification requirement, and record unsupported additions or cases needing adjudication.

Labels follow Section 3: retained, partial, absent, and not adjudicable. The primary score counts only fully retained units; partial and absent receive zero, and not-adjudicable units leave the denominator. The final adjudication contains 103 retained, 18 partial, 3 absent, and 4 not-adjudicable units. Among 124 adjudicable units, strict retention is 83.1%; the aligned tail subset is 19/23 (82.6%). Raw agreement is a calibration quantity, not the retention estimate.

Table 14: Abridged event order for the eight complete trajectories in the two flip cells. The table reports the shared trace pattern; it is mechanism case evidence, not a population mechanism rate.

Stage	Observation	Interpretation
Skill mounted	The skill directory is available in the environment	Availability does not imply that the executor consumed the artifact.
Skill discovery	0 `SKILL.md` reads in 8/8 complete traces	No process evidence that the supplied skill content mediated the outcome.
Skill scripts	0 skill-script invocations in 8/8 traces	The observed path did not use a bundled recovery helper.
Task execution	File and tool events occur under byte-identical S1/REC contexts	The executor proceeded through the task without an artifact-context change.
Verifier	The tail verifier fails in the REC endpoint	The failure is an execution outcome; the trace does not identify a skill-content cause.
Error review	NaN-as-edit counting (pdf-C1) or an incorrect derived mean (xlsx-B3)	Both traced errors are executor-side numerical or data-handling failures.

Table 15 gives three adjudicated examples from the frozen package. The examples show why semantic labels are not reducible to lexical marker matching: a paraphrase can retain an operational rule, while a shortened workflow can preserve its topic but omit a condition or required action.

Table 15: Examples from the blinded semantic audit. Labels are final adjudicated labels; excerpts are shortened for readability.

Unit	Source requirement	Candidate evidence	Label and rationale
Q003/U01	Remove invisible Unicode (Cf/Cc), or run `clean_bibtex.py`, before parsing.	Candidate preserves the category test and the cleaning helper.	Retained: condition and executable action remain recoverable.
Q001/U03	Deduplicate, clean, normalize, depth-filter, remove prefixes, and preserve the processed `source_level`.	Candidate keeps the early preprocessing steps but omits prefix removal and the naming constraint.	Partial: the workflow topic remains, but an essential action and output constraint are missing.
Q028/U01	Generate security integration tests.	Candidate describes security and Spring tooling but contains no corresponding test-generation step.	Absent: the reader cannot recover the required procedure from the candidate.

N Additional Qualitative Example

The C4 lake-warming variant inserts finite placeholder values such as 0, -1, and -999. A generic NaN/type check does not detect them. The compressed artifact can retain the trend-estimation workflow while omitting the domain-range check, leaving a syntactically valid artifact without the recovery step needed for this overlay. The table below records the case construction and the affected recovery unit; it deliberately omits the historical endpoint percentages and is not used as an aggregate mitigation estimate.

Table 16: Illustrative C4 artifact case. The case links one controlled overlay to the recovery unit that it tests; it is not a prevalence estimate or an aggregate mitigation result.

Element	Case description
Base task	Lake-warming attribution with trend estimation and driver attribution.
Overlay	Task-relevant numeric fields contain finite placeholder values (0, -1, or -999).
Routine workflow	Load the data, estimate the trend, classify drivers, run attribution, and write the output file.
Tail recovery unit	Apply domain-aware range validation before attribution; filter or impute invalid records and verify that the resulting contribution is plausible.
Case interpretation	The routine workflow can remain in a compressed artifact after the range check is omitted, while the recovery branch is no longer specified.