INHERITED, ADDED, OR ENDED UNRESOLVED: ATTRIBUTION ACCOUNTING FOR WARM-STARTED CODING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

A warm start lets a coding agent continue from a repository that another session has modified. Testing the final repository measures system success, but does not separate solutions already present at handoff from those added by the successor. It also leaves unresolved whether the successor lost an inherited solution. We measure these transitions with a *prefix-only* arm: replay the recorded prefix and submit the inherited tree without starting a successor. This requires no additional model calls. Pairing its score Y^{prefix} with the resumed session's Y^{final} gives a four-cell accounting and the identity $\mathbb{E}[Y^{\text{final}}] - \mathbb{E}[Y^{\text{prefix}}] = \Pr(\text{added}) - \Pr(\text{ended unresolved})$. The same accounting expresses a published takeover benchmark's 77.0% result for Qwen→Devstral on already-solved handoffs as both a final solved rate and a preservation rate, rather than a rate of newly added solutions. In our historical 260-task SWE-bench Pro analysis with SWE-agent at a 50% action cut, 92.1% of gpt-oss-120b's resumed-arm solves and 70.1% of deepseek-v4-flash's were already resolved at handoff. These shares condition on branch pairs where replay landed edits in both warm-started arms; under frozen intent-to-treat scoring, their net score changes are $+1.2$ and $+14.0$ points. On a separate 100-task cut sweep with a fixed 400-slot denominator, the inherited share rises from 7.9% to 78.7% as handoff moves later, while final score point estimates span 3.25 points. This non-random subset supports a within-sample trend, not a claim that its levels or curve shape transfer to other pools. We recommend reporting the prefix-only baseline alongside the final score so that system success and successor contribution remain distinct.

1 INTRODUCTION

Coding agents may outlast a single conversation or pass work to a fresh session. A successor can inherit a modified repository even when the predecessor's conversation is unavailable. This motivates a warm start, but also creates an attribution question: how much of the final result was already present at handoff? SWE-bench-family evaluation tests the submitted repository state; our SWE-agent setup extracts it as the working tree's `git diff` against the base commit. That patch includes inherited edits as well as the successor's own work. We study how to distinguish their contributions to the score.

The end-state score is a sum of two terms, and hides a third. Let Y^{prefix} be the outcome of the repository state the successor *inherits*, scored by the same harness, and Y^{final} the outcome of the state it *submits*. Then

$$\mathbb{E}[Y^{\text{final}}] = \underbrace{\Pr(Y^{\text{prefix}}{=}1, Y^{\text{final}}{=}1)}_{\text{inherited and preserved}} + \underbrace{\Pr(Y^{\text{prefix}}{=}0, Y^{\text{final}}{=}1)}_{\text{added by the successor}}, \tag{1}$$

$$\mathbb{E}[Y^{\text{final}}] - \mathbb{E}[Y^{\text{prefix}}] = \underbrace{\Pr(Y^{\text{prefix}}{=}0, Y^{\text{final}}{=}1)}_{\text{added}} - \underbrace{\Pr(Y^{\text{prefix}}{=}1, Y^{\text{final}}{=}0)}_{\text{ended unresolved}}. \tag{2}$$

The $1 \to 0$ cell is absent from Eq. 1: final failure does not distinguish a checkpoint that never resolved the task from one that did and then ended unresolved. Scoring the checkpoint makes that distinction

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.

observable (Table 2). Thus $\mathbb{E}[Y^{\text{final}}]$ measures system success, while $\mathbb{E}[Y^{\text{final}} - Y^{\text{prefix}}]$ measures the successor's net score change. An end-state score alone, or a comparison with a reverted tree, does not identify both (§3.3). This corrects a conflation in an earlier version of our analysis.

Published takeover evidence. KC & Budathoki (2026) already separate handoffs that arrive resolved from those that arrive unresolved, recognising that one final *solved* label can describe different takeover outcomes. We express that published distinction in our transition-accounting notation: in the stratum where the checkpointed repository is already resolved, no takeover can *add* a solution, so the reported repository-state-only figure of 77.0% ($47/61$, their Qwen→Devstral successor) is both a final solved rate and a *preservation* rate, rather than a rate of newly added solutions. In that same Qwen→Devstral stratum 14 of the 61 takeovers left a task unresolved after it had been resolved at handoff—23.0%—despite a prompt instructing the successor to preserve changes that already satisfy the task. These integer counts are reconstructed from the published one-decimal percentages under binary, equally weighted outcomes (§2, Appendix R).

Measure the inherited baseline. Replay the logged prefix and submit the resulting tree without starting a successor. This *prefix-only* arm, D, adds no model calls and uses the same patch extraction and judge. Its score supplies Y^{prefix}; separate replay runs require an empirical agreement check (§3.2). A reverted-tree control answers a different question: what retaining code state changes when a successor runs.

Our formal evaluation pool contains 519 SWE-bench Pro tasks. We report its system-level `pass@8A` and `pass@4A+4spread` results first, then use the historical 260-task subsample for the detailed attribution analysis. In that historical analysis, 92.1% of gpt-oss-120b's resumed-arm solves and 70.1% of deepseek-v4-flash's were already resolved at handoff, on pairs where replay landed edits in both warm-started arms. The corresponding net score changes are $+1.2$ and $+14.0$ points (Table 2). Byte-identical submitted patches provide a separate artifact-level check (§5.2). These observations concern score attribution, not how much effort a successor made.

Contributions and scope. We give an accounting identity, a prefix-only measurement protocol, and evidence from two models using SWE-agent on one benchmark family. A fixed 100-task, 400-slot sweep further separates final scores from net score changes as the handoff moves later; its non-random sample does not establish transfer to other pools. The identity applies wherever a checkpoint can be reconstructed and scored by the same evaluator. It lets studies report system success and successor contribution as distinct quantities, without treating either as a substitute for the other.

2 Related Work

Our result concerns attribution under state inheritance. When a final score is attributed to a successor that began from a modified repository, that score can include solutions already supplied by the predecessor. Our diff-extraction convention makes this visible, but the attribution question also arises for other evaluators that score only the final state. Appendix A gives the full comparison.

State reuse spans takeover, restart, training, and search. Handoff Debt resumes coding tasks from checkpointed repositories (KC & Budathoki, 2026), while ContinuityBench studies interrupted state across several agent environments (Gulati, 2026). SWE-Replay branches from archived coding trajectories (Ding & Zhang, 2026); RestartSmart instead starts with a pristine source tree and offers prior edits as an optional overlay (Wang et al., 2026a). Prefix-GRPO replays teacher prefixes in TextCraft, BabyAI, and ALFWorld (Wang et al., 2026c), extending state reuse beyond code repositories. A related control is Fu et al. (2026)'s warm repair followed by clean replay, with a delta-repair sanity check: it establishes cold-start reproducibility of environment setup, whereas our controls measure successor attribution (Appendix A). In the benchmark we reanalyse, the authors already distinguish preserving resolved work from completing unresolved work. We express their published stratification in our transition-accounting notation; this preserves their efficiency claim and makes explicit the inherited baseline behind the Qwen→Devstral 77.0% (Appendix R).

Adjacent boundaries, and sampling as the alternative. Context-management research studies memory and information retention (Packer et al., 2023; Mei et al., 2025; Zerhoudi et al., 2026; Chen et al., 2026b), while multi-agent failure analysis also examines coordination and task verification (Cemri et al., 2025). Recovery methods retain linguistic feedback (Shinn et al., 2023; Madaan et al., 2023; Chen et al., 2024b) or branch a tree search that carries the reasoning (Zhou et al., 2024; Antoniades et al., 2024). We study a specific boundary: **the conversation is destroyed and only the filesystem survives**. Arm A implements independent attempts from scratch (Brown et al., 2024), one approach within the broader test-time compute literature (Snell et al., 2024). Our use of oracle `pass@8` makes verifier limitations directly relevant (Stroebl et al., 2024) (Appendix A).

Where attribution contamination sits among the validity critiques. SWE-bench (Jimenez et al., 2024) and SWE-bench Pro (Deng et al., 2025) evaluate submitted patches with tests; our SWE-agent setup (Yang et al., 2024a) extracts that patch from the working tree. Studies of this benchmark family identify weak tests and solution leakage (Aleithan et al., 2024; Wang et al., 2025c; Yu et al., 2026), and evidence consistent with contamination (Liang et al., 2025; Prathifkumar et al., 2025). Other coding benchmarks study exploitation of the scoring channel (Zhong et al., 2026; Lodkaew et al., 2026; Zhao et al., 2026). These concerns motivate scrutiny of evaluation validity (Kapoor et al., 2024; Bean et al., 2025; Zhu et al., 2025). Shao et al. (2026) give the schema we adopt: identify an *exposure*, establish *how the agent used it*, quantify the *mislead gap*. Ours is the donor's diff left in the tree; the use is *null*—the agent need not act at all; and the gap is $(\mathsf{B}-\mathsf{C})-(\mathsf{B}-\mathsf{D}) = \mathsf{D}-\mathsf{C}$, 15.8 points on gpt-oss's treated slots—the checkpoint against the reverted tree, *not* the $1\to1$ share, which Table 2 reports directly. Repository-history reports describe agents actively accessing an answer source, for example when `git log --all` exposes a fixing commit (Kahn, 2025; Adams, 2026). In our null-use case, no such access is needed: the harness credits inherited bytes to the successor's submission.

3 Design: four arms and an accounting identity

3.1 Arms

A *donor* is an agent session from a pristine checkout. We cut its executed action list at fraction f and mechanically replay that prefix in a fresh container to obtain a checkpoint. No donor conversation is carried over. A comprises eight full-budget fresh sessions, $A_{s_0}, \ldots, A_{s_7}$. Each other arm has four branches h_i, using the prefix of A_{s_i}: B (`resume`) runs a successor on the replayed tree; C (`revert-in-place`) first runs `git checkout -- .` and `git clean -fd`, retaining the environment while restoring source edits; D (*prefix-only*) submits the checkpoint without a successor or model call. Residual submodule pointers after restoration are documented in Appendix N. All four arms use SWE-agent's patch convention, `git add -A && git diff --cached`, and the same frozen scoring rule. Figure 1 summarises the design.

3.2 Checkpoint agreement

Arms B, C and D replay the same prefix in separate, identically provisioned containers. Arm D targets B's starting checkpoint and C's checkpoint *before* restoration, not the restored tree its successor receives. Its recorded tracked-file fingerprint matches both arms in $975/975$ gpt-oss and $1039/1040$ deepseek-v4-flash slots in the historical analysis. These truncated summaries are not tree hashes: agreement does not prove byte identity. The one deepseek-v4-flash mismatch is a real `go.mod` change whose harmlessness is unestablished (Appendix S.1). Checks on replay failure and untracked files staged by patch extraction are in Appendices N and T.2.

3.3 Scores, populations, and contrasts

Cross the checkpoint score Y^{prefix} from D with the same slot's final score Y^{final} from B to obtain Table 2. The $1\to0$ cell is called *ended unresolved*: the scoring rule includes harness and infrastructure failures, so this label does not identify the cause (Appendix I). The four-cell analysis uses *treated* slots, where replay landed edits in both warm-started arms. When replay lands nothing, D submits an empty patch and scores zero; these slots remain in full-pool comparisons and form the `dirty=0` placebo for B versus C.

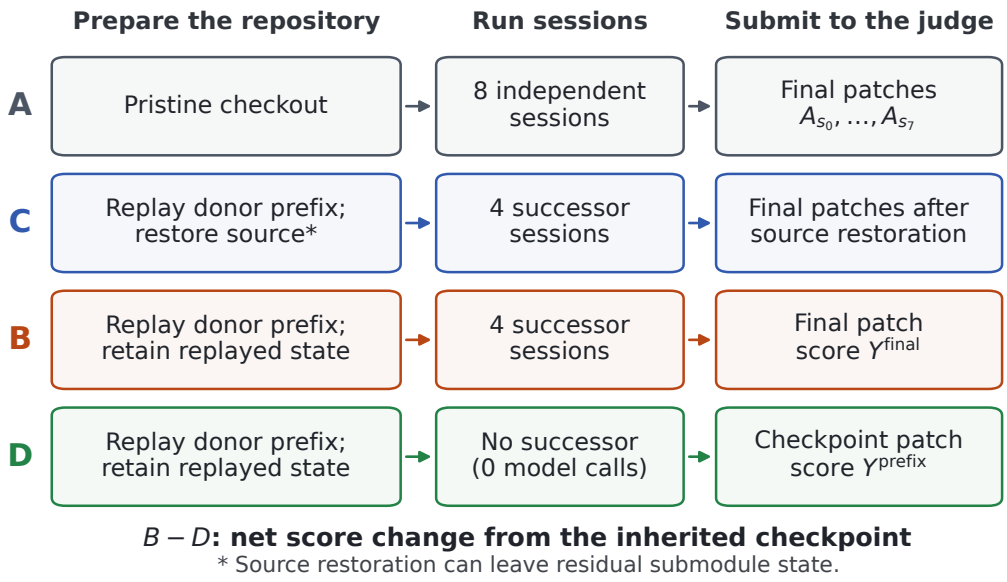

Figure 1: The four arms. B, C and D separately replay the same recorded donor prefix in equivalently provisioned containers. At action50%, the cut halves the action list, not the amount of source editing. C restores source before its successor starts, subject to residual submodule state. D submits the checkpoint without a successor; Y^{prefix} and Y^{final} are the judged patch scores. Replay agreement is measured, not guaranteed by the shared recipe (§3.2). B − D is net score change, not effort or general capability. Every successor begins without the donor conversation.

The four contrasts answer different questions. B − D is the successor's net score change from its inherited checkpoint; B − C is the effect of retaining code state when a successor runs. C − A compares the replayed environment with a fresh start, and D − A compares the checkpoint alone with a fresh full-budget session. They are not components of one reported sum: our treated and full-pool readings use different populations. Arithmetic identities require the same model, candidate set, pairing and scale. Pairing is by task and candidate index, using h_i across replay arms and the held-out $A_{s_{4+i}}$ for comparisons with A.

B − D measures score change, not the successor's effort or general capability. Rewriting a correct fix into an equivalent correct fix contributes zero to this score difference. **D − A is not cost-matched**: D uses no model calls, whereas A has a full budget. On the historical deepseek-v4-flash pool it is -23.65 points per branch; D is a measured baseline, not a competitive policy (Appendix T.4).

3.4 Judging coverage

The frozen intent-to-treat (ITT) rule scores a candidate unresolved if it is `G1Infeasible`, submits an empty or oversized patch ($>$ 5 MB), remains unjudgeable after three attempts, or was never collected. No task is dropped. Every expected historical candidate has a terminal ITT outcome, including all 411 gpt-oss and 648 deepseek-v4-flash treated slots in D. **A terminal outcome need not have readable tests.** In those D slots, readable tests/exhausted attempts/empty patches number $254/112/45$ for gpt-oss and $540/60/48$ for deepseek-v4-flash. The observed $1 \to 1$ cells have readable tests on both sides ($70/70$ and $276/276$), but missing tests in other cells may conceal further inherited solutions. Thus the table is complete for frozen ITT scores, not necessarily for true correctness (Appendix T.3). Setting a missing prefix score to zero favours the successor's recorded net change; when final scores are also missing, the error direction is undetermined. Completed collection does not remove this measurement limitation (Appendix J).

3.5 CUT, BUDGET, STATISTICS, AND BEHAVIOURAL PROXIES

The main cut is action50%: half the donor's executed actions, whose median length is 110 for deepseek-v4-flash and 17 for gpt-oss. The budget rule charges a successor for prefix edit actions, $B' = 250 - |\{\text{prefix edit actions}\}|$, leaving medians of 248 and 250 model calls. This rule favours warm starts; prefix-only D makes no model calls. The sweep adds 25% and 75% cuts (§5.4).

Primary intervals are 10,000-draw task-clustered paired bootstraps (seed 12345). Two-sided exact sign tests use discordant task pairs for `pass@8` and slots for candidate-level contrasts; their uncorrected slot-level p-values do not adjust for within-task dependence. Equivalence uses the predeclared ± 3-point SESOI and the 90% interval criterion (Appendix T.1). We also report zero-edit rates and an own-edit credit gate as behavioural proxies. A lack of detected edits does not mean no model calls or inspection. Editing does not remove all inherited work, and editing-based strata are post-treatment descriptions, not causal subgroups (Appendices C and T.10).

4 EXPERIMENTAL SETUP

Benchmark and populations. The formal pool is 519 usable tasks from SWE-bench Pro's public split (Deng et al., 2025); Appendix B records the $731 \to 519$ exclusions. Table 1 reports system results on this pool. Detailed checkpoint attribution, replay agreement and coverage below retain their historical fixed-seed random 260-task subsample; they are not relabelled as full-pool measurements. Appendix U gives the available full-pool attribution results. The separate cut sweep fixes 100 tasks and 400 slots. We use SWE-agent (Yang et al., 2024a); model settings and configuration appear in Appendix T.5.

The two models show different branch behaviour: deepseek-v4-flash completes most sessions normally, whereas gpt-oss-120b terminates roughly half on a malformed tool call within its first few turns. Section T.7 relates this behaviour to the observed zero-edit rates. **The gpt-oss arms do not share an inference protocol**—A was collected through the Responses API, the branch arms through chat completions—so its C − A and B − A describe the collected systems but do not isolate the effect of the warm-start strategy. Its B − C and B − D do not have this protocol mismatch (Appendix F). All deepseek-v4-flash arms use one protocol.

Candidates, and the two scales a contrast can be read on. Per task, A contributes eight independent samples (4,152 on 519 tasks; 2,080 on 260); B, C and D contribute four each, h_i descending from A_{s_i} (2,076 or 1,040 per arm per model, respectively). Because the arm-level comparison shares its first four samples across arms, it dilutes the arm-level `avg@8` Δ by exactly two (Appendix D); `pass@8` is a per-task indicator and admits no such factorisation, so the identity below is stated for `avg@8` only:

$$\Delta_{\text{arm}} \;=\; \tfrac{1}{8}\Big(\textstyle\sum_i h_i - \sum_i A_{s_{4+i}}\Big) \;=\; \tfrac{1}{2}\,\Delta_{\text{branch}}. \tag{3}$$

We report both, with Δ_{branch}—one warm-started branch against one fresh sample—as **the primary scale**: it is the unit an engineer choosing "resume" over "sample again" acts on, and the unit the ± 3-point SESOI was declared in.

Collection and missing tests. The historical 50% cells are collected at 1,040/1,040 per branch arm per model, and sweep cells at 400/400. Complete collection and terminal ITT outcomes do not imply that every candidate has readable tests. Per-arm counts, sensitivity to missing tests and replay checks are in Appendices I and N.

5 RESULTS

5.1 SYSTEM RESULTS ON THE FORMAL 519-TASK POOL

At the same eight-candidate allowance, deepseek-v4-flash's nominal difference favours `pass@4A+4spread` by 0.19 percentage points, but the paired test detects no advantage ($p = 1.0000$). This does not establish equivalence, nor does equal candidate count imply equal realised model calls or wall-clock cost. The gpt-oss row cannot isolate an allocation effect because its fresh and branch sessions use different inference protocols.

Table 1: System success on all 519 tasks, action50%, frozen ITT. Each column uses eight candidates: eight fresh trajectories, or four shared fresh trajectories plus four resume branches. For deepseek-v4-flash, the difference is $+0.19$ percentage points (13 tasks solved only by `pass@8A` and 14 only by `pass@4A+4spread`; two-sided exact paired sign test, $p = 1.0000$). This does not establish equivalence. gpt-oss uses different APIs for fresh and branch sessions, so its difference does not isolate a warm-start strategy effect.

Model	`pass@8A`	`pass@4A+4spread`
deepseek-v4-flash	68.98%	69.17%
gpt-oss-120b	47.78%	42.77%

This does not conflict with the historical treated-slot $\mathsf{B} - \mathsf{D}$ result of $+14.04$ points: that contrast compares continuing from a checkpoint with submitting it without continuation, whereas Table 1 compares two uses of an eight-candidate allowance. The pools, conditioning and aggregation also differ, so their magnitudes are not interchangeable. The final system score does not distinguish inherited solutions from those added after handoff; we examine that distinction next on the historical 260-task subsample, with its own treated-slot denominators and measured replay coverage.

5.2 The accounting: what the successor added

Table 2: Prefix and final scores on historical 260-task treated slots, where replay landed edits in both warm-started arms. Rows cross the prefix-only outcome (Y^{prefix}, arm D) with the resumed outcome (Y^{final}, arm B) under frozen ITT. All 411 gpt-oss and 648 deepseek-v4-flash slots have terminal outcomes, but some lack readable tests. The observed $1 \to 1$ cells are test-backed on both sides[†]; missing tests elsewhere still limit the interpretation in terms of true correctness. The $1 \to 0$ label is "ended unresolved" because the rule also scores harness and infrastructure failures as unresolved (§3.4).

		gpt-oss-120b ($n = 411$)		deepseek-v4-flash ($n = 648$)	
Cell	Reading	n	%	n	%
$0 \to 0$	neither	334	81.3	227	35.0
$0 \to 1$	**added by the successor**	6	1.5	118	18.2
$1 \to 0$	ended unresolved	1	0.2	27	4.2
$1 \to 1$	inherited and preserved	70	17.0	276	42.6
$\mathbb{E}[Y^{\text{final}}]$ (the reported score)		18.5		60.8	
$\mathbb{E}[Y^{\text{prefix}}]$ (the inherited tree alone)		17.3		46.8	
$\mathsf{B} - \mathsf{D}$ (**incremental contribution**)		$+\mathbf{1.2}$		$+\mathbf{14.0}$	
inherited share of the resumed arm's solves		$70/76 = 92.1\%$		$276/394 = 70.1\%$	
$1 \to 1$ test-backed on both sides[†]		$70/70 = 100\%$		$276/276 = 100\%$	

On these historical treated slots, gpt-oss's $\mathsf{B} - \mathsf{D}$ is $+1.2$ points (95% CI $[0.00, +2.54]$; six positive and one negative change, sign-test $p = 0.125$). The remaining 404 of 411 treated slots retain the same frozen score; the seven changes occur on seven distinct tasks. For gpt-oss at action50% on the historical 260-task pool, the candidate-level $\mathsf{B} - \mathsf{C}$ contrast is $+17.0$ points ($[+12.4, +22.1]$) on the 411 slots with `dirty>0` after replay in both warm-started arms: retaining code state and adding score after handoff are different comparisons. deepseek-v4-flash's $\mathsf{B} - \mathsf{D}$ is $+14.0$ ($[+9.5, +18.7]$), with 118 positive and 27 negative changes on 648 slots. The model-specific treated sets differ; these numbers do not identify a cross-model effect ratio.

The byte-level audit, which does not go through the judge. The prefix-only patch is non-empty and *byte-identical* to what the resumed session submitted in 324 of 975 gpt-oss and 165 of 1040 deepseek-v4-flash slots. These counts use all available artifact-comparison slots in the historical pool; the narrower census below also conditions on outcomes and successor activity. Censusing the

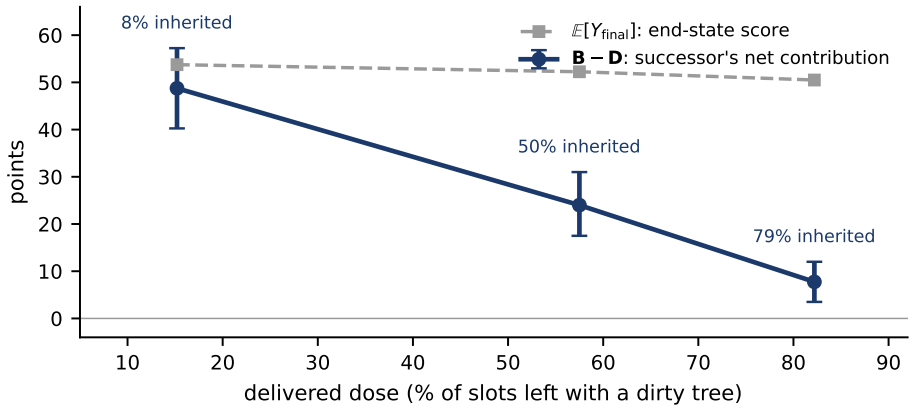

Figure 2: **Later handoffs have smaller net score gains in this deepseek-v4-flash subset.** All cuts use the same 100 tasks and 400 candidate slots. The x-axis is delivered dose: the fraction with dirty trees in both warm-started arms (61, 230, 329 of 400), distinct from the nominal $25/50/75\%$ action cut. Error bars show 95% intervals for $\mathsf{B}-\mathsf{D}$; the gray final-score line shows point estimates. The inherited share is the fraction of final solves already resolved at handoff. This non-random subset supports a within-sample comparison; transfer of its levels or curve shape is unestablished (Appendix O).

cleanest case—resumed arm resolved, branch issued zero repository-modifying actions, $Y^{\text{prefix}} = 1$, revert control unresolved, patch SHA-256-identical to the non-empty prefix-only patch—gives **58** slots over 42 tasks on gpt-oss and 8 over 7 on deepseek-v4-flash, every one a $1 \to 1$ cell the end-state contrast records as a successor win over the control. Byte equality bypasses both the fingerprint and the judge; what it establishes is that the *extracted artifact* is the same, not that the successor never touched the tree—**a session that edited and then reverted its own edit would look identical**. Dropping only the requirement that the revert control is unresolved gives 62 gpt-oss slots over 43 tasks; this is the only leave-one-out variant computed. Five inspected examples with hashes are in Appendix T.7.

A concrete inherited solve. Consider the audited gpt-oss candidate h_0 for `ansible__ansible-3b823d908e` on the historical pool. The prefix-only checkpoint already scores as resolved; the resumed branch also resolves the task without a recorded repository-modifying action, while the revert control does not resolve it. The two non-empty submitted patches have matching recorded SHA-256 hashes (reported prefix `edda3c85cb`), placing this candidate in the inherited-and-preserved $1 \to 1$ cell. These checks concern the submitted patches, not identity of the entire repository state; the full five-condition audit and the other examples are in Appendix T.7.

5.3 Retaining code state on the historical deepseek-v4-flash pool

On the historical 260-task deepseek-v4-flash pool, we test equivalence of mean per-branch scores against the pre-declared ± 3-point margin using two one-sided tests. System-level results are reported separately in Appendix T.9.

On the per-branch scale, none of the three unconditional contrasts establishes equivalence at ± 3. The bounds exclude an environment-advancement benefit above $+4.04$ points and a whole-warm-start benefit above $+3.65$. On the arm-level `avg@8` scale, all three contrasts satisfy equivalence within ± 3: Eq. 3 halves each contrast, so this is a different reading of the same data, not independent evidence.

Conditional on delivered dose, the 90% upper bound for $\mathsf{B}-\mathsf{C}$ is $+0.92$ points on the treated stratum and $+0.00$ where both branches really edited. These bounds rule out gains above about a point, but both intervals **include zero** (Appendix T.6).

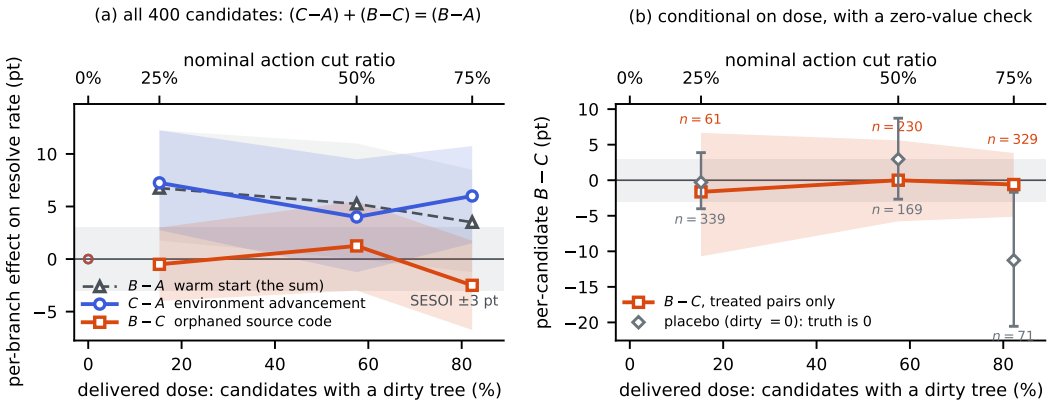

Figure 3: **The total effect of retaining code state, across its dose range** (deepseek-v4-flash), as the action cut is swept over $0/25/50/75\%$ (top axis) against the dose *delivered*—the fraction of branch candidates whose tree was dirty after replay (bottom axis), $15.2/57.5/82.2\%$. **(a)** One estimand for all three curves: per-branch effects over all 400 candidates per arm per cut, so the decomposition closes at every dose and one ± 3-point SESOI strip applies to all three. Its point at dose 0 is zero *by construction*, not a measurement—no candidate is treated at a 0% cut—and is drawn as an open marker detached from the fitted line so it is not read as an observed dose. **(b)** A *different* estimand, plotted separately rather than on the same axis: candidate-level $\mathsf{B} - \mathsf{C}$ on *treated* pairs only, beside the same-cut placebo, whose true value is zero by construction and which is shown as a descriptive check rather than as a threshold; panel (b) has no dose-0 point, since no pair is treated there. Bands are 95% paired bootstrap intervals, task-clustered as in the main analysis (§3); $2{,}400$ branch candidates (§5.4).

5.4 Dose–response across action cuts

We vary only the action cut, using the first 100 historical tasks in canonical order (400 slots per arm per cut). This non-random subset covers 5 of 11 repositories and has 39% of tasks in one repository. At $25/50/75\%$ cuts, replay delivers edits to $15.2/57.5/82.2\%$ of slots: nominal cut and delivered dose differ because early actions are often reads.

On the fixed 400 slots (Figure 2), $\mathsf{B} - \mathsf{D}$ falls from $+48.75$ (95% CI $[+40.25, +57.25]$) to $+24.00$ ($[+17.50, +31.00]$) to $+7.75$ ($[+3.50, +12.00]$). The intervals do not overlap. Final score point estimates span 3.25 points ($53.75 \to 52.25 \to 50.50$), while the inherited share of final solves rises $7.9\% \to 49.8\% \to 78.7\%$. Thus later handoffs in this sample have smaller net gains and larger inherited shares; equivalence of final scores across cuts is not established. The treated subsets change with the cut, and their $\mathsf{B} - \mathsf{C}$ intervals establish neither a nonzero effect nor equivalence at ± 3 points. Figure 3 separates the full-slot contrasts from the conditional $\mathsf{B} - \mathsf{C}$ comparisons and their placebo checks; Appendix O gives the full numerical results and editing-based exposure measures.

As a separate control, $\mathsf{B} - \mathsf{C}$ on the `dirty=0` stratum has a pooled 90% equivalence bound of ± 2.96 points (579 candidate pairs across cuts). The 75% cut alone gives -11.27 points (95% CI $[-20.55, -1.69]$; $n = 71$), so the pooled result does not establish equivalence at every cut. This check concerns $\mathsf{B} - \mathsf{C}$ on selected zero-dose pairs; it does not calibrate uncertainty in the fixed-400-slot $\mathsf{B} - \mathsf{D}$ curve.

5.5 Process costs alongside uncertain outcome contrasts

On the historical deepseek-v4-flash treated slots, retaining donor edits lengthens protocol-phase wall clock by 587 s and lowers the clean-submission rate by 9.9 points ($n = 648$; discordant pairs $58{:}122$, $p = 2 \times 10^{-6}$), with no measurable difference in model calls (Table 16). Placebo intervals include zero on all three measures. The corresponding outcome intervals do not establish a benefit. The resume arm's zero-edit rate is 13.1% versus 3.8%: editing itself changes across arms, which is why strata conditioned on successor editing remain descriptive.

6 DISCUSSION AND CONCLUSION

Report the inherited baseline. A final score combines preserved and newly added solutions. A reverted-tree comparison measures the effect of retaining code state for a successor; a checkpoint comparison measures net score change after handoff. Neither replaces the other. Where reconstruction is possible, replay and submit the prefix without a successor, using the same evaluator. This adds no model calls and exposes the $1 \to 0$ outcomes hidden by a final score.

A reporting protocol. We recommend three items whenever a checkpoint can be reconstructed:

1. Report the final score together with the prefix-only baseline on the same paired candidates and evaluator.
2. Report the added $(0 \to 1)$ and *ended unresolved* $(1 \to 0)$ fractions separately; their difference is the net score change (Eq. 2).
3. State the task pool, candidate-pair denominator and selection rule for each conditional result, keeping treated-subset and full-pool results distinct.

What the evidence supports. High inherited shares occur in both models despite different observed activity: deepseek-v4-flash makes median 138 model calls and 7 own edits, versus 4 and 0 for gpt-oss, pooled over B and C on the historical 260-task pool (Table 11). This does not establish a general relationship between capability and inheritance. For historical deepseek-v4-flash treated slots, the 90% interval for $\mathsf{B} - \mathsf{C}$ is $[-5.96, +0.92]$ points: it limits a benefit while allowing zero and several points of harm. Longer wall clock and fewer clean finishes accompany this uncertain outcome contrast.

Scope and limitations. We study one scaffold, one benchmark family and two models, with donor conversation removed and a budget rule favourable to warm starts. The formal system comparison uses 519 tasks; detailed attribution, identity and coverage evidence comes from the historical 260-task pool. Available full-pool attribution results appear separately in Appendix U. The non-random 100-task sweep and its changing treated subsets support only descriptive comparisons within that sample.

The `dirty=0` placebo checks B against C, not either against A. gpt-oss's fresh samples use Responses while its successors use Chat Completions; its comparisons against A therefore do not isolate the warm-start strategy. The two models' environment components are not comparable (Appendix J). Its both-edited stratum is also too small to resolve the conditional code-state effect (Appendix F).

Replay agreement uses truncated fingerprints and includes one real deepseek-v4-flash mismatch of unknown effect. Terminal ITT coverage is complete, but some candidates lack readable tests. The observed $1 \to 1$ cases are test-backed on both sides; missing tests elsewhere still limit claims about true correctness and inherited share (Appendices S.1 and T.3).

AI USE STATEMENT

Large language models had two roles in this work. deepseek-v4-flash and gpt-oss-120b were the experimental subjects: they produced the agent trajectories and repository edits evaluated under the SWE-agent protocol. The scoring and replay procedures are described in the methods and appendix.

Claude and Codex assistants also helped draft and revise the manuscript, orchestration code, diagnose infrastructure failures, search and check references.

REPRODUCIBILITY STATEMENT

The arm construction, budgets and scoring rule are in §3; task-pool provenance, configuration, edit detection, statistics and replay checks are in Appendices B, T.5, C, T.1 and S.1. Appendix V lists reproduction records and separates deterministic analysis settings from variable rollouts and replay outcomes. Project records include task lists, run artifacts and analysis code; anonymous artifact access is pending, so full external reproducibility is not yet established.

REFERENCES

Connor B. Adams. Git reward hacking in SWEBench pro OSS. scaleapi/SWE-bench_Pro-os issue #93, https://github.com/scaleapi/SWE-bench_Pro-os/issues/93, 2026. The same future-history exposure is reachable in the public SWE-bench Pro images.

Aradhye Agarwal, Ayan Sengupta, and Tanmoy Chakraborty. The art of scaling test-time compute for large language models. *arXiv preprint arXiv:2512.02008*, 2025.

Reem Aleithan, Haoran Xue, Mohammad Mahdi Mohajer, Elijah Nnorom, Gias Uddin, and Song Wang. SWE-Bench+: Enhanced coding benchmark for LLMs. *arXiv preprint arXiv:2410.06992*, 2024.

Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Wang. SWE-Search: Enhancing software agents with monte carlo tree search and iterative refinement. *arXiv preprint arXiv:2410.20285*, 2024.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. SWE-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. arXiv:2505.20411.

Paul Barbaste, Tristan Darrigol, Germain Vu, and Tom Wiltberger. Harness engineering: Anatomy, architecture, and evolution of coding agents – a source-code study of eleven systems. *arXiv preprint arXiv:2609.00006*, 2026.

Andrew M. Bean, Ryan Othniel Kearns, Angelika Romanou, Franziska Sofia Hafner, Harry Mayne, Jan Batzner, Negar Foroutan, Chris Schmitz, Karolina Korgul, Hunar Batra, Oishi Deb, Emma Beharry, Cornelius Emde, Thomas Foster, Anna Gausen, María Grandury, Simeng Han, Valentin Hofmann, Lujain Ibrahim, Hazel Kim, Hannah Rose Kirk, Fangru Lin, Gabrielle Kaili-May Liu, Lennart Luettgau, Jabez Magomere, Jonathan Rystrøm, Anna Sotnikova, Yushi Yang, Yilun Zhao, Adel Bibi, Antoine Bosselut, Ronald Clark, Arman Cohan, Jakob Foerster, Yarin Gal, Scott A. Hale, Inioluwa Deborah Raji, Christopher Summerfield, Philip H. S. Torr, Cozmin Ududec, Luc Rocher, and Adam Mahdi. Measuring what matters: Construct validity in large language model benchmarks. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. arXiv:2511.04703.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. arXiv:2503.13657.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2207.10397.

Dong Chen, Shaoxin Lin, Muhan Zeng, Daoguang Zan, Jian-Gang Wang, Anton Cheshkov, Jun Sun, Hao Yu, Guoliang Dong, Artem Aliev, Jie Wang, Xiao Cheng, Guangtai Liang, Yuchi Ma, Pan Bian, Tao Xie, and Qianxiang Wang. CodeR: Issue resolving with multi-agent and task graphs. *arXiv preprint arXiv:2406.01304*, 2024a.

Kang Chen, Junjie Nian, Yixin Cao, and Yugang Jiang. Disagree to explore, agree to commit: Routing-guided test-time scaling for software agents. *arXiv preprint arXiv:2608.22191*, 2026a.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to Self-Debug. In *International Conference on Learning Representations (ICLR)*, 2024b. arXiv:2304.05128.

Zhuofu Chen, Rui Pan, Yinwei Dai, and Ravi Netravali. Slipstream: Trajectory-grounded compaction validation for long-horizon agents. *arXiv preprint arXiv:2605.08580*, 2026b.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-Bench pro: Can AI agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.

Yifeng Ding and Lingming Zhang. SWE-Replay: Efficient test-time scaling for software engineering agents. *arXiv preprint arXiv:2601.22129*, 2026.

Ryan Ehrlich, Bradley Brown, Jordan Juravsky, Ronald Clark, Christopher Ré, and Azalia Mirhoseini. CodeMonkeys: Scaling test-time compute for software engineering. *arXiv preprint arXiv:2501.14723*, 2025.

Abul Ehtesham, Aditi Singh, Gaurav Kumar Gupta, and Saket Kumar. A survey of agent interoperability protocols: Model context protocol (MCP), agent communication protocol (ACP), agent-to-agent protocol (A2A), and agent network protocol (ANP). *arXiv preprint arXiv:2505.02279*, 2025.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang (Eric) Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. Magentic-One: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

Sihan Fu, Oucheng Liu, Shiyuan Wang, Jin Shi, and Chengkun Wei. BootstrapAgent: Distilling repository setup into reusable agent knowledge. *arXiv preprint arXiv:2605.15815*, 2026.

Matthias Galster, Seyedmoein Mohsenimofidi, Jai Lal Lulla, Muhammad Auwal Abubakar, Christoph Treude, and Sebastian Baltes. Harness engineering for agentic AI coding tools: An exploratory study. *arXiv preprint arXiv:2602.14690*, 2026.

Aryan Gulati. ContinuityBench: A framework and taxonomy for evaluating agent recovery from interrupted state. In *ICML 2026 Workshop on Combining Theory and Benchmarks: Towards A Virtuous Cycle to Understand and Guarantee Foundation Model Performance*, 2026. OpenReview `https://openreview.net/forum?id=3N3BzvoLbG`.

Tooraj Helmi. Modeling response consistency in multi-agent LLM systems: A comparative analysis of shared and separate context approaches. *arXiv preprint arXiv:2504.07303*, 2025.

Qishuo Hua, Lyumanshan Ye, Dayuan Fu, Yang Xiao, Xiaojie Cai, Yunze Wu, Jifan Lin, Junfei Wang, and Pengfei Liu. Context engineering 2.0: The context of context engineering. *arXiv preprint arXiv:2510.26493*, 2025.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.06770.

Jacob Kahn. Repo state loopholes during agentic evaluation. SWE-bench issue #465, `https://github.com/SWE-bench/SWE-bench/issues/465`, 2025. Agents reach future repository state via `git log --all` and related commands.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: Optimizing context compression for long-horizon LLM agents. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2510.00615.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *arXiv preprint arXiv:2407.01502*, 2024.

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, Franck Ndzomga, Dheeraj Oruganty, Sophie Luskin, Kangheng Liu, Botao Yu, Amit Arora, Dongyoon Hahm, Harsh Trivedi, Huan Sun, Juyong Lee, Tengjun Jin, Yifan Mai, Yifei Zhou, Yuxuan Zhu, Rishi Bommasani, Daniel Kang, Dawn Song, Peter Henderson, Yu Su, Percy Liang, and Arvind Narayanan. Holistic agent leaderboard: The missing infrastructure for AI agent evaluation. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.11977.

Dipesh KC and Anjila Budathoki. Handoff debt: The rediscovery cost when coding agents take over interrupted tasks. *arXiv preprint arXiv:2606.02875*, 2026. URL `https://arxiv.org/abs/2606.02875v2`. Version 2.

Joongwon Kim, Wannan Yang, Kelvin Niu, Hongming Zhang, Yun Zhu, Eryk Helenowski, Ruan Silva, Zhengxing Chen, Srinivasan Iyer, Manzil Zaheer, Daniel Fried, Hannaneh Hajishirzi, Sanjeev Arora, Gabriel Synnaeve, Ruslan Salakhutdinov, and Anirudh Goyal. Scaling test-time compute for agentic coding. *arXiv preprint arXiv:2604.16529*, 2026.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022. arXiv:2203.07814.

Yujiang Li, Zhenyu Hou, Yi Jing, Jie Tang, and Yuxiao Dong. CompactionRL: Reinforcement learning with context compaction for long-horizon agents. *arXiv preprint arXiv:2607.05378*, 2026.

Shanchao Liang, Spandan Garg, and Roshanak Zilouchian Moghaddam. The SWE-Bench illusion: When state-of-the-art LLMs remember instead of reason. *arXiv preprint arXiv:2506.12286*, 2025.

Jiacheng Liu, Xiaohan Zhao, Xinyi Shang, and Zhiqiang Shen. Dive into Claude Code: The design space of today's and future AI agent systems. *arXiv preprint arXiv:2604.14228*, 2026.

Thanawat Lodkaew, Johannes Ackermann, Soichiro Nishimori, Nontawat Charoenphakdee, Masashi Sugiyama, and Takashi Ishida. Do coding agents deceive us? detecting and preventing cheating via capped evaluation with randomized tests. *arXiv preprint arXiv:2606.07379*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2303.17651.

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, Chenlin Zhou, Jiayi Mao, Tianze Xia, Jiafeng Guo, and Shenghua Liu. A survey of context engineering for large language models. *arXiv preprint arXiv:2507.13334*, 2025.

OpenAI. Introducing SWE-bench verified. `https://openai.com/index/introducing-swe-bench-verified/`, 2024. Human-validated 500-instance subset of SWE-bench.

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.

Thanosan Prathifkumar, Noble Saji Mathews, and Meiyappan Nagappan. Does SWE-Bench-Verified test agent ability or model memory? *arXiv preprint arXiv:2512.10218*, 2025.

Muhammad Shihab Rashid, Christian Bock, Yuan Zhuang, Alexander Buchholz, Tim Esler, Simon Valentin, Luca Franceschi, Martin Wistuba, Prabhu Teja Sivaprasad, Woo Jung Kim, Anoop Deoras, Giovanni Zappella, and Laurent Callot. SWE-PolyBench: A multi-language benchmark for repository level evaluation of coding agents. *arXiv preprint arXiv:2504.08703*, 2025.

Benjamin Rombaut. Inside the scaffold: A source-code taxonomy of coding agent architectures. *arXiv preprint arXiv:2604.03515*, 2026.

Jon Saad-Falcon, Estefany Kelly Buchanan, Mayee Chen, Tzu-Heng (Brian) Huang, Brendan McLaughlin, Tanvir Bhathal, Shang Zhu, Ben Athiwaratkun, Frederic Sala, Scott Linderman, Azalia Mirhoseini, and Christopher Ré. Weaver: Shrinking the generation-verification gap by scaling compute for verification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. doi: 10.52202/085713-2639. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/715f48023c26ad210321a4c0da67f27c-Abstract-Conference.html.

Jiaqi Shao, Hanck Chen, Wei Zhang, Maxm Pan, and Bing Luo. Do agent benchmarks measure capability? protocol validity in the age of agentic AI. *arXiv preprint arXiv:2607.22368*, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Shivalika Singh, Yiyang Nan, Alex Wang, Daniel D'Souza, Sayash Kapoor, Ahmet Üstün, Sanmi Koyejo, Yuntian Deng, Shayne Longpre, Noah A. Smith, Beyza Ermis, Marzieh Fadaee, and Sara Hooker. The leaderboard illusion. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. arXiv:2504.20879.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Benedikt Stroebl, Sayash Kapoor, and Arvind Narayanan. The limits of inference scaling through resampling. *arXiv preprint arXiv:2411.17501*, 2024.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon LLM agent via context-folding. *arXiv preprint arXiv:2510.11967*, 2025.

SWE-agent Team. mini-SWE-agent: The minimal AI software engineering agent. https://github.com/SWE-agent/mini-swe-agent, 2025.

Chenyu Wang, Yunbo Lyu, Junda He, Zhou Yang, Chenxing Zhong, Yaniv Harel, and David Lo. Fail-fast, restart-smart: Early failure prediction and restart for SWE agentic tasks. *arXiv preprint arXiv:2608.03222*, 2026a.

Fali Wang, Hui Liu, Zhenwei Dai, Jingying Zeng, Zhiwei Zhang, Zongyu Wu, Chen Luo, Zhen Li, Xianfeng Tang, Qi He, and Suhang Wang. AgentTTS: Large language model agent for test-time compute-optimal scaling strategy in complex tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a. arXiv:2508.00890.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations (ICLR)*, 2025b. arXiv:2407.16741.

Xingyao Wang, Simon Rosenberg, Juan Michelini, Calvin Smith, Hoang Tran, Engel Nyst, Rohit Malhotra, Xuhui Zhou, Valerie Chen, Robert Brennan, and Graham Neubig. The OpenHands software agent SDK: A composable and extensible foundation for production agents. In *Proceedings of Machine Learning and Systems (MLSys)*, 2026b. arXiv:2511.03690.

Yihan Wang, Zhong Guan, Haoran Sun, Jiale Huang, Likang Wu, and Hongke Zhao. From trajectories to prefixes: Reusing teacher trajectories via replayed prefixes and online continuation. *arXiv preprint arXiv:2607.19395*, 2026c.

You Wang, Michael Pradel, and Zhongxin Liu. Are "solved issues" in SWE-bench really solved correctly? an empirical study. *arXiv preprint arXiv:2503.15223*, 2025c.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Tianyuan Wu, Chaokun Chang, Lunxi Cao, Wei Gao, and Wei Wang. Crab: A semantics-aware checkpoint/restore runtime for agent sandboxes. *arXiv preprint arXiv:2604.28138*, 2026.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying LLM-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.

Chunqiu Steven Xia, Zhe Wang, Yan Yang, Yuxiang Wei, and Lingming Zhang. Live-SWE-agent: Can software engineering agents self-evolve on the fly? *arXiv preprint arXiv:2511.13646*, 2025.

Junjielong Xu, Boyin Tan, Xiaoyuan Liu, Chao Peng, Pengfei Gao, and Pinjia He. Scalable supervision for software agents via patch reasoning. *arXiv preprint arXiv:2510.22775*, 2025a.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b. arXiv:2502.12110.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a. arXiv:2405.15793.

John Yang, Carlos E. Jimenez, Alex L. Zhang, Kilian Lieret, Joyce Yang, Xindi Wu, Ori Press, Niklas Muennighoff, Gabriel Synnaeve, Karthik R. Narasimhan, Diyi Yang, Sida I. Wang, and Ofir Press. SWE-bench multimodal: Do AI systems generalize to visual software domains? *arXiv preprint arXiv:2410.03859*, 2024b.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. arXiv:2504.21798.

Boxi Yu, Yang Cao, Yuzhong Zhang, Liting Lin, Junjielong Xu, Zhiqing Zhong, Qinghua Xu, Guancheng Wang, Jialun Cao, Shing-Chi Cheung, Pinjia He, and Lionel Briand. SWE-ABS: Adversarial benchmark strengthening exposes inflated success rates on test-based benchmark. *arXiv preprint arXiv:2603.00520*, 2026.

Daoguang Zan, Zhirong Huang, Wei Liu, Hanwu Chen, Linhao Zhang, Shulin Xin, Lu Chen, Qi Liu, Xiaojian Zhong, Aoyan Li, Siyao Liu, Yongsheng Xiao, Liangqiang Chen, Yuyu Zhang, Jing Su, Tianyu Liu, Rui Long, Kai Shen, and Liang Xiang. Multi-SWE-bench: A multilingual benchmark for issue resolving. *arXiv preprint arXiv:2504.02605*, 2025. URL `https://arxiv.org/abs/2504.02605v1`. Version 1.

Guangtao Zeng, Maohao Shen, Delin Chen, Zhenting Qi, Subhro Das, Dan Gutfreund, David Cox, Gregory Wornell, Wei Lu, Zhang-Wei Hong, and Chuang Gan. Satori-SWE: Evolutionary test-time scaling for sample-efficient software engineering. *arXiv preprint arXiv:2505.23604*, 2025.

Saber Zerhoudi, Jelena Mitrovic, and Michael Granitzer. The compaction cliff in long-running AI agent memory. *arXiv preprint arXiv:2608.22752*, 2026. URL `https://arxiv.org/abs/2608.22752v1`. Version 1.

Linghao Zhang, Shilin He, Chaoyun Zhang, Yu Kang, Bowen Li, Chengxing Xie, Junhao Wang, Maoquan Wang, Yufan Huang, Shengyu Fu, Elsie Nallipogu, Qingwei Lin, Yingnong Dang, Saravan Rajmohan, and Dongmei Zhang. SWE-bench goes live! In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025a. arXiv:2505.23419.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025b.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö. Arik. Chain of agents: Large language models collaborating on long-context tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.02818.

Bingchen Zhao, Dhruv Srikanth, Yuxiang Wu, and Zhengyao Jiang. SpecBench: Measuring reward hacking in long-horizon coding agents. *arXiv preprint arXiv:2605.21384*, 2026.

Ziqian Zhong, Aditi Raghunathan, and Nicholas Carlini. ImpossibleBench: Measuring LLMs' propensity of exploiting test cases. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.20270.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2310.04406.

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Jasjeet Sekhon, Jacob Steinhardt, Antony Kellermann, Sarah Schwettmann, Matei Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. Establishing best practices for building rigorous agentic benchmarks. *arXiv preprint arXiv:2507.02825*, 2025.

Yu Zhuang, Kefei Chen, Yitong Duan, Shuxin Zheng, Jian Li, and Xu-Yao Zhang. AgentRewind: Recoverable execution for long-horizon LLM agents. *arXiv preprint arXiv:2608.14380*, 2026.

APPENDIX

A EXTENDED RELATED WORK

This appendix expands the comparisons in §2.

Warm starts, takeover benchmarks, and related controls. KC & Budathoki (2026)'s 75 source tasks yield 181 handoff points and 2,172 takeover runs scored by the official SWE-bench harness on the inherited repository, 61 of those points in an "Already solved; preserve" stratum which is operationally $Y^{\text{prefix}} = 1$ and which we therefore re-read as a transition matrix (§1, Appendix R); Wang et al. (2026a)'s overlay starts off, so the successor's source tree is pristine, but the successor can inspect and apply the prior diff. This differs from arm C, which reverts edits without offering that overlay. Ding & Zhang (2026) branch new coding trials from archived trajectories at intermediate steps. Wang et al. (2026c) replay teacher prefixes and continue online in TextCraft, BabyAI, and ALFWorld; their inherited state is not a code repository. Fu et al. (2026) pair warm repair with clean replay to verify cold-start reproducibility of a repository's setup contract, and use a delta-repair sanity check to prevent weakened verification commands. This is a related warm/clean control, with a different intervention and measurement target. Finally, Gulati (2026) resume across `tau`-bench, AppWorld and TerminalBench at three handoff fidelities. Interruption reduces mean task success from 41.7% to 28.0%; they vary context fidelity, whereas we vary code state. They keep each benchmark's *native* evaluator. Applying our proposed accounting to such evaluators would require making the pre-takeover baseline an explicit evaluation input.

Boundaries where state survives. Context compaction and summarisation (Packer et al., 2023; Sun et al., 2025; Kang et al., 2026; Li et al., 2026), agentic memory stores (Xu et al., 2025b; Chhikara et al., 2025) and snapshot/restore runtimes for agent sandboxes (Wu et al., 2026; Zhuang et al., 2026) preserve different parts of agent and environment state. Context-engineering surveys organise these design choices (Mei et al., 2025; Hua et al., 2025), while compaction studies directly measure

information retention and downstream behaviour (Zerhoudi et al., 2026; Chen et al., 2026b). Multi-agent work studies conversation, coordination, and shared versus separate contexts (Wu et al., 2023; Zhang et al., 2024; Fourney et al., 2024; Helmi, 2025); interoperability protocols provide interfaces for tool access and agent handoffs (Ehtesham et al., 2025). These settings differ in what survives the boundary; our intervention isolates filesystem inheritance after conversational context is removed.

Test-time scaling and the independent-sampling baseline. Arm A uses repeated independent sampling (Brown et al., 2024). Broader test-time scaling methods include large-scale sampling with filtering, generated-test selection, iterative refinement, and adaptive compute allocation (Li et al., 2022; Chen et al., 2023; Snell et al., 2024; Ehrlich et al., 2025; Zeng et al., 2025; Wang et al., 2025a; Agarwal et al., 2025; Kim et al., 2026; Chen et al., 2026a; Zhang et al., 2025b); arm A does not implement those refinements. Reported gains depend sharply on verifier quality (Stroebl et al., 2024), and practical verifiers and patch critics remain imperfect (Saad-Falcon et al., 2025; Xu et al., 2025a); we therefore report oracle `pass@8` alongside `avg@8`. The former assumes oracle selection among the sampled candidates; the latter describes their mean test-based success rate. CodeR (Chen et al., 2024a) instead coordinates multiple agents through predefined task graphs, another way to organise issue resolution.

The benchmark family and its scaffolds. Beyond SWE-bench (Jimenez et al., 2024) and its human-validated subset (OpenAI, 2024), related benchmarks and data-generation pipelines extend coverage to multimodal, multilingual, live and synthesised tasks (Yang et al., 2024b; Zan et al., 2025; Rashid et al., 2025; Badertdinov et al., 2025; Zhang et al., 2025a; Yang et al., 2025) alongside the repository-level tasks of SWE-bench Pro (Deng et al., 2025). Coding scaffolds provide tools and execution environments (Yang et al., 2024a; Wang et al., 2025b; 2026b; SWE-agent Team, 2025; Xia et al., 2025), and recent work studies their harness layer directly (Rombaut, 2026; Liu et al., 2026; Galster et al., 2026; Barbaste et al., 2026)—which is the level at which our artifact lives. Whether iterative agentic interaction is where a score comes from is contested independently of our setting: Xia et al. (2024) report that a localise–repair–validate pipeline outperforms the open-source agents they compare on SWE-bench Lite. Leaderboard-level artefacts and evaluation infrastructure are studied by Singh et al. (2025) and Kapoor et al. (2026). Our proposal to report zero-edit rates extends this concern for transparent evaluation to attribution under state inheritance; we do not attribute that specific reporting requirement to those papers.

B POOL PROVENANCE

All tasks derive from the SWE-bench Pro public split (731 tasks). Our filters first exclude tasks lacking container images or failing gold/empty-patch sentinel checks. They then exclude collection-infeasible tasks and tasks for which every candidate has an empty test report. The resulting formal pool has 519 tasks; Table 3 records each stage and its exclusions. The historical 260-task pool used in the main analyses is a *simple uniform* random subsample of the 519—`random.sample` with seed `20260805`, drawn once and frozen, not stratified—whose repository proportions happen to track the 519-pool's within 2 points on every repository. The draw was recorded when the campaign held 16 candidates and zero verdicts, before judging results were available. Both models and all four arms use these same 260 tasks in the historical analysis; the 519-task results and the 100-task sweep are reported separately.

Two distinctions matter for interpreting these exclusions. A whole task was excluded only when every candidate was unjudgeable; if any candidate returned a test list, other unjudgeable candidates remained in the pool and scored unresolved under the frozen rule. This establishes that the environment can produce tests for at least one candidate, not that every candidate can be evaluated reliably.

A *pending* candidate has no readable verdict and has not reached an applicable terminal rule. A *terminal ITT outcome* may instead come from readable tests, an empty or oversized patch, `G1Infeasible`, or exhausted attempts without readable tests. A set restricted to candidates with readable tests is therefore different from a set with no pending candidates. The completed historical ledger removes pending-candidate filtering; it does not remove selection caused by requiring readable tests. We do not reconstruct a strict-set history here because the earlier counts were not bound

Table 3: The exclusion ledger from the SWE-bench Pro public mirror to the 260-task pool. Every step records a task or container eligibility criterion. This does not guarantee that excluded tasks would have affected all strategies equally. The two exclusion lists are stored with the project records (`EVAL_INFEASIBLE_candidates.txt`, 48 tasks; `EXCLUDED_INFRA_TASKS.txt`, 23 tasks).

Tasks	Removed	Reason
731	—	SWE-bench Pro public mirror, as distributed.
590	141	**Sentinel validation.** A task is kept only if the dataset's own gold patch *passes* `fail_to_pass` in our container *and* the empty patch *fails*. Removed: 84 with no container image, 53 `gold_broken` (the shipped gold patch does not pass in our container either), 4 `empty_passes` (the tests already pass unpatched, so the bug is absent in this environment).
567	23	**Collection-infeasible.** Every candidate fails after ≈110 s with no traceback, reproduced by both our collector and the unmodified single-arm collector—a task- or image-side fault, not an agent outcome.
519	48	**Unjudgeable environment.** *All* candidates return `{"tests": []}`: the judging container cannot start (these tasks need redis, postgres or a `pnpm` build). Concentrated in the same heavy-environment repositories the sentinel already thinned: `flipt-io` 20, `navidrome` 13, `vuls` 5, `teleport` 4, `protonmail` 3.
260	259	**Uniform random subsample**, one fixed-seed draw, taken when the campaign held 16 candidates and zero verdicts, never re-drawn. This is the historical subsample, not the denominator of the separate 519-task and sweep analyses.

to a common model, pool, candidate set, timestamp and predicate. The per-arm terminal routes and sensitivity to excluding unjudgeable pairs are reported separately in Appendix I.

C THE EDIT DETECTOR AND ITS VALIDATION

Every branch trajectory is scanned for the repository-modifying actions the branch issued *itself*. The detector splits each executed action into shell segments (on `&&`, `||`, `;`, `|`), recognises the editor tool's `create` / `str_replace` / `insert` / `replace` commands, and additionally flags shell writes (`sed -i`, `patch -p`*n*, `tee`, output redirection, Python `open(..., 'w')`, `apply_patch`) whose resolved target lies inside the repository. Writes to scratch paths (`/tmp`, `/var`, `/dev`, `/proc`, `/root`, ...) do not count.

Two directions of error matter differently. Under-detection would misclassify a branch that worked as a free-rider and would *inflate* the free-ride stratum; over-detection would dilute it. Because one model's agent writes nearly every shell action as `cd /app && <cmd>`, a naive first-token match under-detects badly on it, which is why the detector is segment-structured rather than regex-only.

Arm C provides a validation check: after `git checkout -- .` and `git clean -fd`, a branch with no detected edits should submit an empty patch unless later commands have uncounted side effects. The table reports that check and its observed exceptions.

D THE TWO SCALES, DERIVED

The decisive region, and where the two scales come from. A branch arm is scored as $\{A_{s_{0..}s_3}\} \cup \{h_{0..3}\}$ against A's $\{A_{s_{0..}s_7}\}$. The systems share the first four samples verbatim. Their `pass@8` scores can differ only where those four all failed: 89 tasks for deepseek-v4-flash and 148 for gpt-oss. Their `avg@8` scores can differ outside that region as well. **For `avg@8`** the shared half cancels in the

Table 4: Detector validation. In arm C "zero own edits" should imply "empty patch"; in arm B it should not, and the gap is the free-ride channel. The two deepseek-v4-flash counterexamples in C were inspected individually and are side effects of running the repository's own test suite (generated fixture files), not undetected agent edits.

Model	Arm	zero own edits	of which non-empty patch	consistent
deepseek-v4-flash	C revert-in-place	40	2	95.0%
	B resume	136	99	27.2%
gpt-oss	C revert-in-place	819	0	100.0%
	B resume	839	310	63.1%

difference, which is what produces the factor of two in Eq. 3. **For `pass@8` it does not**: `pass@8` is a non-linear function of the eight indicators, so the decisive region is where the arms *can* differ at all, but no factorisation follows from it (see the derivation below).

Fix a task and write $r(\cdot) \in \{0, 1\}$ for a candidate's resolved indicator. The arm-level `avg@8` of A is $\frac{1}{8}\sum_{i=0}^{7} r(A_{s_i})$; that of a branch arm $X \in \{\mathsf{B}, \mathsf{C}\}$ is $\frac{1}{8}\big(\sum_{i=0}^{3} r(A_{s_i}) + \sum_{i=0}^{3} r(h_i^X)\big)$, because the design scores the branch arm as its four branches *plus the four donor samples they descend from*, held verbatim. Subtracting, the four shared terms cancel:

$$\Delta_{\text{arm}} = \tfrac{1}{8}\Big(\textstyle\sum_{i=0}^{3} r(h_i^X) - \sum_{i=4}^{7} r(A_{s_i})\Big) = \tfrac{1}{2}\Big(\tfrac{1}{4}\textstyle\sum_{i=0}^{3} r(h_i^X) - \tfrac{1}{4}\sum_{i=4}^{7} r(A_{s_i})\Big) = \tfrac{1}{2}\,\Delta_{\text{branch}}.$$

Every arm-level Δ against A in this paper is therefore exactly half the corresponding per-branch effect, and so is every interval around it, since the bootstrap resamples the same per-task quantity scaled by $\frac{1}{2}$. The conversion is exact, not approximate, and it applies to `avg@8` only: `pass@8` is a non-linear function of the eight indicators and has no such factorisation, which is why Table 20 gives a per-branch column for `avg@8` and not for `pass@8`.

A consequence worth stating: the decomposition $\mathsf{B} - \mathsf{A} = (\mathsf{C} - \mathsf{A}) + (\mathsf{B} - \mathsf{C})$ closes exactly only when all three terms are measured on the same candidate set. At the per-branch scale over all 1,040 deepseek-v4-flash candidates, $(+1.44) + (-0.38) = +1.06$; at the arm level, $(+0.72) + (-0.19) = +0.53$. The $\mathsf{B} - \mathsf{C}$ readings of -2.47 and -3.17 elsewhere in the paper are conditional on post-treatment strata (`dirty`>0; both branches edited) and are *not* the term that closes the decomposition; conflating the two is the error this appendix exists to prevent.

Table 5: The arm-level block withheld from Tables 20 and 17: the deepseek-v4-flash contrasts at the 50% cut read as Δ_{arm}, i.e. what happens to a fixed eight-candidate budget when four of its candidates become branches. Every entry is exactly half its per-branch counterpart, interval included, and all three are equivalent to zero within the ± 3-point margin. This is the same evidence at a different scale, not additional evidence. The corresponding arm `avg@8` levels are 52.98 (A), 53.51 (B) and 53.70 (C), with 95% CIs $[-1.01, +2.07]$ for $\mathsf{B} - \mathsf{A}$ and $[-0.82, +2.31]$ for $\mathsf{C} - \mathsf{A}$.

Contrast (arm-level, diluted $\times\frac{1}{2}$)	n	mean	90% CI	equivalent at ± 3?	MDE
$\mathsf{C} - \mathsf{A}$ environment	260	$+0.72$	$[-0.58, +2.02]$	**yes** (± 2.02)	2.24
$\mathsf{B} - \mathsf{A}$ warm start as reported	260	$+0.53$	$[-0.77, +1.83]$	**yes** (± 1.83)	2.20
$\mathsf{B} - \mathsf{C}$ orphaned code, all candidates	260	-0.19	$[-1.54, +1.11]$	**yes** (± 1.54)	2.28

E $\mathsf{B} - \mathsf{C}$ CONDITIONAL ON THE DELIVERED DOSE

Table 17 reports the three unconditional per-branch contrasts. The rows below condition instead on what the replay delivered and on what the branch then did, and are the equivalence-table form of the deepseek-v4-flash strata of §T.7. Both conditioning variables are post-treatment (Appendix J),

so these are descriptive robustness checks on the measurement, not causal estimates, and no equivalence claim is made from them; the 90% upper bounds of $+0.92$ and $+0.00$ quoted in §5.3 are read off the first two rows.

Table 6: deepseek-v4-flash, 50% cut, per-candidate B − C within post-treatment strata. 10,000-draw paired bootstrap (seed 12345), resampling tasks rather than slots (§3); MDE is the effect detectable at 80% power.

Contrast	n	mean	90% CI	MDE
B − C treated (`dirty>0`)	648	-2.47	$[-5.96, +0.92]$	5.83
B − C both branches edited	537	-3.17	$[-6.57, +0.00]$	5.69
B − C placebo (`dirty=0`)	380	$+3.16$	$[-0.25, +6.50]$	5.72

F THE GPT-OSS PROTOCOL MISMATCH

Arm A for gpt-oss was collected on an earlier campaign that used the Responses API in a direct high-effort mode (`mode = oss-high-direct-responses`); the two branch arms were collected later through the chat-completions path (`mode = flash-litellm-chat`), which is also what both deepseek-v4-flash arms use. The shard rows record the mode and protocol per candidate, and the behavioural gap is large: a median A session makes 26 model calls at a median 108 completion tokens per call, against 4 calls at 1,875 tokens per call in the branch arms. We discovered this in a post-hoc audit rather than by design and report it as a defect.

Its consequence is bounded but real. Any gpt-oss contrast *against* A—C − A, B − A, and the branch-versus-fresh-sample rows of Appendix H—confounds the treatment with the protocol. These contrasts describe the collected systems but do not identify a strategy effect. The B − C contrast does not have this mismatch: both branch arms were produced by the same collector in the same campaign from the same donor prefixes, and differ only in the restore command.

The mechanism narrative of §T.7 also survives, because gpt-oss's format fragility is visible *within* each protocol. Its A arm, on its own protocol, exits `exit_format` 50.9% of the time, against 46.5% (B) and 45.1% (C) on the other; deepseek-v4-flash exits `exit_format` 3.3%, 4.6% and 3.0% on the same three arms. **The branch-arm rates are over all** 1040 **candidate slots and the arm A rates over its** 2080 (8 candidates × 260 tasks); Table 12 reports the same counts over the 975 non-`G1Infeasible` slots, which is why the gpt-oss figures there read 49.6% and 48.1% for the same 484 and 469 exits. The branch-arm comparison uses a common protocol and still shows a large format-error difference between the models. It does not isolate protocol effects on other behaviours or outcomes.

Table 7: The gpt-oss counterpart of Table 20, withheld from the main text because every Δ in it is confounded with the protocol difference above. Columns as in Table 20; CI is a 10,000-draw paired bootstrap on the per-task arm-level `avg@8` difference. These are descriptive scores for the collected systems, not estimates of a pure strategy effect. The A row gives `pass@8A` and the B row `pass@4A+4spread`.

	`pass@8`				`avg@8`			
Arm	rate	Δ	excl.	p	rate	Δ_{arm}	CI_{95}	Δ_{branch}
A pristine	49.23	—	—	—	23.46	—	—	—
B resume	44.62	-4.62	13:1	.002	15.82	-7.64	$[-9.33, -6.06]$	-15.29
C revert-in-place	43.46	-5.77	15:0	.0001	12.50	-10.96	$[-12.88, -9.09]$	-21.92

G COLLECTION AND JUDGING DIAGNOSTICS

The two branch arms' `G1Infeasible` sets coincide candidate-for-candidate (65 of 65, over 51 tasks), as expected since constructibility is decided at replay time before the arms diverge. Exactly one task has all four branches infeasible; arm A does not solve it.

Table 8: Per-arm collection status, gpt-oss (deepseek-v4-flash is complete with 0 `G1Infeasible` and is omitted). Under the frozen ITT rule, `G1Infeasible`, empty patches, oversized patches and uncollected candidates all score as unresolved with the full pool in the denominator. "Terminal" counts settled ITT outcomes, not candidates with readable tests. Arm C submits an empty patch 88.8% of the time and arm B 60.1%; the 28.7-point gap motivates the artifact-level attribution checks.

Arm	candidates	collected	`G1Infeasible`	empty patch	terminal
A pristine (8/task)	2080	2080	0 (0.0%)	592 (28.5%)	2080
B resume (4/task)	1040	1040	65 (6.2%)	625 (60.1%)	1040
C revert (4/task)	1040	1040	65 (6.2%)	924 (88.8%)	1040

H BRANCH VERSUS FRESH SAMPLE, PER CANDIDATE

Table 18 compares the two branch arms to each other. This appendix compares each branch to a fresh sample it does not descend from (h_i against $A_{s_{4+i}}$ on the same task), which is the contrast that speaks to the warm start as a whole rather than to its decomposition. It is per-branch by construction and needs no dilution correction; averaging its B rows over the two strata reproduces the Δ_{branch} column of Table 20.

Table 9: Branch versus same-task fresh sample, paired per candidate. Intervals are 10,000-draw **task-clustered** paired bootstraps (seed 12345) on the paired difference, as used for the main inferences (Appendix T.1). Strata here use the *single-arm* `dirty` criterion, which is why the denominators are 386/654 rather than the body's 380/648; see the note below the table.

Model	Arm	Stratum	n	branch	fresh A	Δ (95% CI)
deepseek-v4-flash	B	`dirty=0`	386	43.0	36.8	+6.2 [+1.6, +11.1]
	B	`dirty>0`	654	60.2	62.2	−2.0 [−5.8, +1.7]
	C	`dirty=0`	386	39.9	36.8	+3.1 [−1.5, +7.9]
	C	`dirty>0`	654	62.7	62.2	+0.5 [−3.5, +4.4]
gpt-oss	B	`dirty=0`	564	1.4	20.6	−19.1 [−23.8, −14.7]
	B	`dirty>0`	411	18.5	28.0	−9.5 [−14.4, −4.6]
	C	`dirty=0`	564	1.6	20.6	−19.0 [−23.4, −14.6]
	C	`dirty>0`	411	1.5	28.0	−26.5 [−32.3, −21.0]

Two `dirty` criteria, and why their denominators differ. This table stratifies on a *single* arm's `postreplay_dirty_n`, giving 386 + 654 = 1,040 for deepseek-v4-flash and 564 + 411 = 975 for gpt-oss (the remaining 65 being `G1Infeasible`, which have no branch session and so no per-candidate pairing). Every B-versus-C contrast in the body instead requires *both* arms to be dirty (or both clean), giving 380 + 648 = 1,028 for deepseek-v4-flash. **The difference is 12 deepseek-v4-flash slots in which one arm's count is zero and the other's is not, and the two criteria are therefore not interchangeable**—we never mix them in one row. The 12 are an artefact of `postreplay_dirty_n` counting *untracked* files: in all 12 the tracked tree is the same and the discrepancy is run-time debris (`core.python.<pid>`, `dump.rdb`) that the two arms' successors happened to leave or not leave. gpt-oss has none, which is why its 411 is the same under both criteria. The gpt-oss rows are contaminated by the protocol difference of §4—they contrast a chat-protocol branch against a Responses-protocol fresh sample—and are reported only to show the `dirty=0` pattern: branches resolve 1.4% and 1.6% against 20.6%, alongside the approximately 86% zero-edit rate in Table 12. These are related observations, not interchangeable measurements.

I THE UNJUDGEABLE CLAUSE, AND A PER-PROTOCOL SENSITIVITY

The frozen ITT rule scores a candidate unresolved if the evaluation harness has not produced a test list after three attempts. Table 10 counts where that happens. It is not a negligible clause: 519 candidates across the six A/B/C arms—a count of candidates, not of tasks, and unrelated to the

519-task split—carry a non-empty, normally sized patch and are scored unresolved on this route alone.

Table 10: Candidates whose patch is non-empty and within the size limit but for which the harness never returned a test list, scored unresolved under the frozen ITT rule.

Model	Arm	non-empty patches	unjudgeable → unresolved	share
deepseek-v4-flash	A pristine	1850	67	3.6%
	B resume	986	40	4.1%
	C revert-in-place	992	41	4.1%
gpt-oss	A pristine	1487	214	14.4%
	B resume	415	123	**29.6%**
	C revert-in-place	116	34	**29.3%**

What the unjudgeable candidates actually are. We read every one of these candidates' {candidate}_output.json and stderr rather than infer the failure mode. On gpt-oss the composition is **a single category with no exceptions**: all 123 arm-B and all 34 arm-C cases are {"tests": [], "attempts": 3, "last_rc": 0}, with stderr stopping after Applied patch cleanly. and the test phase producing no output. **Zero** are oversized, **zero** fail to apply, **zero** hit the container timeout (rc=−9), and **zero** are missing a log. The patch applies; the tests then yield nothing.

The two arms' unjudgeable rates are close once normalised. Normalised by the candidates that can reach judging at all—non-empty, within the size limit—the two arms are within a third of a point of each other: $123/415 = 29.6\%$ for B against $34/116 = 29.3\%$ for C. Arm C contributes fewer unjudgeable candidates in absolute terms while submitting fewer non-empty patches. This is what one would expect if the clause acted on patches rather than on arms, but we report it as the observed pattern: **we measure the two rates, not the selection mechanism that produces them**, and we do not claim to have ruled out a judging bias.

Unjudgeable patches are larger in the resumed arm: an association, not an identified cause. Within arm B, the candidates the harness could not judge carry a median patch of **5236** B against **2386** B for those it did judge—more than double. In arm C the same comparison shows nothing: 2734 B unjudgeable against 2920 B judged, if anything slightly smaller.

This is *consistent with* a large inherited diff breaking test *collection*, so that the harness returns no test list at all. **We do not claim to have identified that mechanism, and arm C is not a clean control for it.** The two arms differ in more than patch size: the restore removes the entire inherited source state, not merely its volume, and the arms' container paths differ as well. Any of these could produce the same contrast. What we have is an association in the arm that carries donor residue and its absence in the arm that does not; isolating the size channel would need an experiment that varies size while holding inherited state fixed, which we did not run. §3.3 cites this as a reason for caution in naming the $1 \to 0$ cell, not as an attribution.

We did not re-run these candidates, deliberately. CE_MAX_ATTEMPTS=3 was declared before analysis. Retrying only the candidates that failed, until they return a number, is selective retry: it would keep adding judgements until the arms stopped looking short, with no principled stopping rule, and—for the reason given in the next paragraph—we could not say in advance which way that would move the contrast. B has the larger number of unjudgeable candidates, but a count of what is missing does not determine how many of them would have resolved. We therefore report the frozen-ITT figure as the headline and the per-protocol figure below as the sensitivity, and we do not report a re-run point estimate. **We do not claim the frozen rule makes our estimate conservative in general**: the direction in which missing judgements move a contrast is set by how many of each arm's missing candidates *would have resolved*, and those two counts are unobservable—their absence is precisely what hides them. An arm's unjudgeable count bounds the number of those candidates that would have resolved, but does not determine it. Neither the rates nor their totals

therefore establish a direction; only an arm with no missing relevant outcome contributes a known zero. Terminal ITT status alone does not establish that condition. On this pool the rates are in any case within a third of a point of each other (29.6% against 29.3%), which we report as an observation and not as an argument. The per-protocol figure below is what the shift is on this data, not a bound.

Per-protocol sensitivity. Dropping every treated pair in which either arm is unjudgeable gives the following, against the frozen ITT figures of Table 18:

B − C, treated stratum	ITT	per-protocol	pairs dropped
deepseek-v4-flash	-2.47	-1.98	41 of 648 (B 28, C 19)
gpt-oss	$+17.03$	$+\mathbf{24.05}$	120 of 411 (B 116, C 11)

For deepseek-v4-flash, the point estimate remains negative under both readings. On gpt-oss the two readings differ more: the per-protocol reading of the contrast a warm-start study would report is $+24.1$ against the frozen rule's $+17.0$. **That is the shift on this data, not a bound.** Normalised by the candidates that can reach judging at all, the two arms' unjudgeable rates are within a third of a point of each other (29.6% against 29.3%, above)—which we record as an observation and not as an argument, since the unjudgeable counts bound the unknown numbers of missed solutions without determining them or the direction of the contrast's shift. Arm B contributes more unjudgeable candidates in absolute terms while submitting more non-empty patches. We keep the ITT number in the main text because it is the pre-registered one.

J THE FOUR STRUCTURAL RESERVATIONS, IN FULL

(i) Post-treatment stratification. Every "both branches really edited" stratum conditions on a variable the treatment itself moves: orphaned code raises the branch's zero-edit rate from 3.8% to 13.1% (Table 12) and lowers its edit count, so selecting pairs in which both branches edited selects differentially on the treated arm. Those rows describe whether the contrast persists among pairs where both branches edited; they do not isolate a causal effect or rule out inherited solutions. *(ii) What the placebo validates.* The `dirty=0` stratum validates B against C only: the branch arms are byte-identical there, but both containers are still warm while A starts cold, which is exactly what C − A is for. A placebo for C − A would need a cold container replayed with an empty action list, which the harness cannot produce without changing the provisioning path. *(iii) Cross-model incomparability.* gpt-oss emits 65 `G1Infeasible` candidates per branch arm and deepseek-v4-flash none, and the frozen ITT rule charges those to the branch arms only (§P), so the two models' environment components differ by a branch-construction failure rate as well as by model behaviour and are not to be differenced. *(iv) Frozen ITT under differential coverage.* An uncollected candidate scores unresolved, so incomplete collection can move a contrast. Write u_X for the number of candidates in arm X that would have resolved but are scored unresolved because they were never collected; the contribution to a recorded B − C is then $(u_C - u_B)/N$. **Coverage bounds each u_X from above but does not determine its size**, so in general *the direction of the shift does not follow from which arm is less complete*: the thinner arm's missing candidates may simply be ones that would not have resolved. We state this because the converse is tempting and we have made the corresponding error elsewhere (§3.4 and the retraction it cites).

For this *uncollected-candidate* component, the direction is determined when one arm is fully collected: its u is zero, so the shift can only favour that arm. This statement does not cover terminal outcomes with missing tests, whose effects remain separate. That was the case we met on the 519-task split, and it is worth recording as history rather than as a present state. At the frozen snapshot of 2026-09-25 00:01:28, deepseek-v4-flash's arm D stood at 100% of its candidates while its reference half stood at 51.3%, so $u_{\mathsf{D}} = 0$ and D − A could only be pushed upward. **Both arms have since closed at** $2076/2076$, so the narrow case no longer applies to this contrast: the reading on the completed split is in Table 22. A separate project note records a full-pool per-branch value of -23.17 on 519 tasks, alongside the historical 260-task value of -23.65. The direct 519-task output underlying that note has not been located among the retained sources reviewed for this draft (Appendix U). The treated contrast and the full-pool contrast use different selected populations and must not be joined into a collection-progress trajectory. **Which coverage figure matters here is not the arm's: D − A pairs each branch h_i against $A_{s_{4+i}}$, so only the four candidates $A_{s_4}, \ldots, A_{s_7}$ serve as its**

reference. At that snapshot, the reference half was less completely collected than the shared half, so the eight-candidate average overstated collection coverage for the four candidates used by this contrast. Quoting the arm-level figure therefore understates the gap the contrast is exposed to. **An arm's coverage is not automatically the coverage of the subset that enters a given contrast; report the one that does.** This is inert in the body, where all four arms are complete on the 260-task pool. For pending collection, check the exact reference set before interpreting coverage. If both arms are incomplete, their collection rates alone do not determine the direction of the shift. For terminal ITT outcomes, report readable-test coverage separately. We withheld the deepseek-v4-flash 519 contrasts on exactly that ground until every arm closed, and report them below now that they have.

K BRANCH-SESSION BEHAVIOUR

Table 11: Model configurations and two distinct evidence sets. Both models are self-hosted on SGLang with controlled sampling settings. The first panel reports system success on all 519 tasks (action50%, frozen ITT; Table 1). gpt-oss uses different APIs for fresh and branch sessions, so its score difference does not isolate a warm-start strategy effect. The second panel gives historical branch-activity medians, pooled over B and C on 260 tasks (2,080 candidates per model), from the recorded `n_calls` field and branch-edit ledger. These describe the zero-edit exposure in §3.5; they do not explain the formal-pool score differences. Referenced from §4.

System success: all 519 tasks, formal pool

Model	`pass@8A`	`pass@4A+4spread`
deepseek-v4-flash	68.98%	69.17%
gpt-oss-120b	47.78%	42.77%

Historical branch activity: 260 tasks, 2,080 candidates per model

Model	Sampling	Median calls	Median own edits
deepseek-v4-flash	T=1.0, top-p 1.0, thinking max	138	7
gpt-oss-120b	T=1.0, reasoning effort high	4	0

Table 12: Branch-session behaviour, per candidate: deepseek-v4-flash over all 1,040 candidates per arm, gpt-oss over the 975 non-`G1Infeasible` ones. Model calls are the recorded `n_calls` field; "zero-edit rate" is the fraction of branch candidates that never issued a repository-modifying action. Medians are over the stated candidate set, so 135.5 is the true median of an even-sized set and is not rounded. Referenced from §T.7 and §5.5.

Model	Arm	median model calls	median own edits	zero edit	`submitted`	`exit_format`
gpt-oss	B resume	4	0	86.1%	36.5%	49.6%
	C revert-in-place	4	0	84.0%	37.7%	48.1%
deepseek-v4-flash	B resume	140	6	13.1%	69.3%	4.6%
	C revert-in-place	135.5	8	3.8%	76.7%	3.0%

L FREE-RIDE DECOMPOSITION: DEEPSEEK-V4-FLASH, AND AN ALTERNATIVE CRITERION

Table 18 splits on the branch's own edit actions. A weaker, cruder criterion—did the branch make more than five model calls—gives the same picture on gpt-oss: free-riders ($n = 293$) show $+16.4$ $[+11.3, +21.8]$, apparent workers ($n = 118$) show $+18.6$ $[+10.8, +27.2]$, and restricting to pairs where *both* arms exceed five calls leaves $n = 32$ with $+18.8$ $[+3.1, +35.5]$. The call-count criterion does not separate the artifact, because a branch can burn calls without editing; the edit-action

Table 13: deepseek-v4-flash counterpart to the indented rows of Table 18, on its 648 treated pairs. The B-idle-/-C-worked cell that is decisive for gpt-oss is here only 100 pairs and moves the other way; the cell that dominates gpt-oss (both idle) is a single pair. Intervals are 95% task-clustered paired bootstraps (Appendix T.1).

Cell	n	B	C	B − C (95% CI)
all treated pairs	648	60.8	63.3	-2.5 [$-6.68, +1.56$]
B idle / C idle	1	—	—	—
B worked / C idle	10	80.0	10.0	$+70.0$ [$+44.4, +90.0$]
B idle / C worked	100	67.0	74.0	-7.0 [$-17.65, +3.06$]
B worked / C worked (net effect)	537	59.2	62.4	-3.2 [$-7.24, +0.74$]

criterion is more directly tied to this artifact, subject to its detector limits. On deepseek-v4-flash the call-count criterion is degenerate (only 2 pairs have a branch under six calls), so that threshold provides little separation there. The edit-action split on deepseek-v4-flash gives -5.9 [$-16.5, +4.1$] for free-riding branches ($n = 101$) and -1.8 [$-5.8, +2.0$] for working ones ($n = 547$).

M INFRASTRUCTURE VALIDATION: CONCURRENCY

CPU concurrency silently degrades rollout quality. In an earlier campaign on this cluster, raising the cap from 20 to 52 simultaneous containers dropped the same-task paired `submitted` rate from 81% to 14% and the median model calls per run from 49 to 9. The mechanism is indirect: ordinary shell commands (`grep`, `find`, test runners) are dragged past the per-command timeout by contention, and the resulting terminations cascade. No explicit overall failure flag announces this—the jobs complete and the harness records them as finished; only the quality metrics move. All results in this paper run under the validated cap, identical across arms, with paired candidates scheduled under the same cap.

N THE REPLAY-FIDELITY ASSERTION SUITE

Replay is the load-bearing step of this design and it can fail silently. We encountered exactly that: in an early collection every `str_replace_editor` invocation in the replayed prefix exited with `rc=127`—the tool binary was not on the path inside the replay container—so the donor's edits never landed. The harness reported success throughout: the replay completed, the sessions ran, the patches were collected and judged, and the resulting "resume" arm produced a plausible near-null. It was in fact a degraded copy of the revert arm. The near-null score alone did not reveal the replay failure.

Every branch record now carries the following fields, written by reading back `git status` inside the container after replay and again after the restore:

field	assertion
`postreplay_is_repo`	the replay target is a valid git repository
`postreplay_dirty_n`	modified-file count after replay, *before* any restore
`postreplay_diffstat`	the diffstat, for manual inspection of any anomaly
`revert_dirty_before`	modified-file count immediately before the restore
`revert_dirty_after`	modified-file count immediately after the restore
`revert_clean`	true iff `revert_dirty_after` $= 0$
`revert_residual`	the surviving `git status` entries, if any

An arm is rejected before scoring if its post-replay dirty rate does not match its donors' edit rate. All data in this paper pass. Measured outcomes:

- `postreplay_is_repo` holds for 1,040/1,040 deepseek-v4-flash and 975/975 non-`G1Infeasible` gpt-oss branch candidates in each arm.

- Before the restore, the recorded tracked-file fingerprints match between B and C in 1040/1040 deepseek-v4-flash and 975/975 gpt-oss slots. This supports the checkpoint comparison without proving byte identity. **The recorded count `postreplay_dirty_n` is not identical, because it also counts untracked files**: it agrees in 1,009/1,040 deepseek-v4-flash slots and 975/975 gpt-oss slots, every deepseek-v4-flash disagreement being run-time debris. This is why the both-arms criterion yields 1,028 rather than 1,040 deepseek-v4-flash slots, and why we do not present that split as exhaustive.
- `revert_clean` holds for 936/1,040 deepseek-v4-flash (90.0%) and 882/975 gpt-oss (90.5%) arm C candidates. Every residual is a git submodule pointer—103 of 104 deepseek-v4-flash residuals and all 93 gpt-oss residuals are `M vendor/...`—which `git checkout -- .` cannot restore. Submodule pointers are not source code the agent could be credited for, and they are present in both arms, but we report the figure rather than describing C as perfectly pristine.

We report all of this because the replay failure and the free-ride artifact of §3.5 are the same kind of error: a pipeline that keeps running while the quantity it measures quietly becomes something else. Scores alone do not establish replay fidelity or edit attribution. Dirty-tree and own-edit checks can expose these failure modes, with the detector and fingerprint limits described above.

O CUT-RATIO SWEEP: FULL NUMBERS

Figure 3 in the main text shows the full-slot and conditional dose comparisons. The tables below retain their complete numerical results and checks.

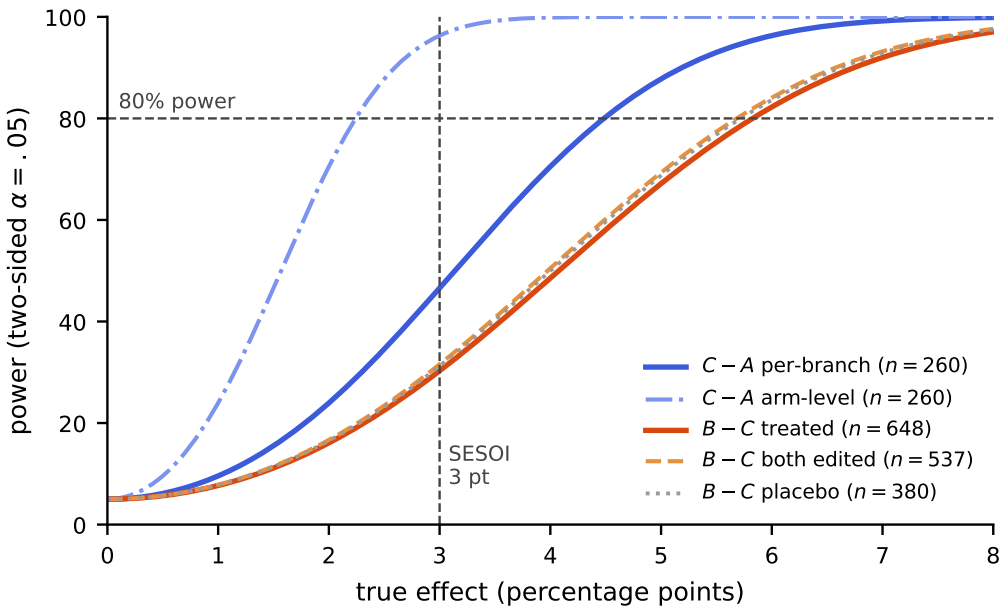

Figure 4: Power to detect a true effect at the sample sizes we have, for the 50% cell of Table 17. The per-task arm-level contrast (light blue) resolves effects below the 3-point margin; the same contrast read per-branch (dark blue) and the candidate-level B − C contrasts do not, which is why we report a one-sided bound for those rather than equivalence.

The sweep pool is the first 100 tasks of the frozen 260-task pool in canonical order. It covers only 5 of the pool's 11 repositories and is dominated by one of them—`ansible` 39, `flipt-io` 21, `NodeBB` 18, `element-hq` 15, `future-architect` 7, so 39% of the sweep is a single repository against 15% of the full pool. It is a deterministic subset, not a fresh random draw, and it is harder than the full pool (A `pass@8` 64.00% against 70.38%, `avg@8` 47.88% against 52.98%). Both facts limit generalisation of either the curve shape or its levels beyond this subset. Every sweep cell reuses the 50% cell's configuration verbatim—same donor root, same 250 model-call budget,

Table 14: Dose–response on the 100-task sweep pool, frozen ITT throughout, 10,000-draw paired bootstrap (seed 12345). *Delivered dose* is the fraction of the 400 branch candidates whose tree was dirty after replay. Block (a) reports one estimand—per-branch effects over all 400 candidates—so that $(\mathsf{C}-\mathsf{A})+(\mathsf{B}-\mathsf{C})=(\mathsf{B}-\mathsf{A})$ holds at every dose; it is what Figure 3(a) plots. Block (b) reports the candidate-level $\mathsf{B}-\mathsf{C}$ restricted to the treated stratum (`dirty`>0 in both arms) beside the same-cut placebo (`dirty`$=0$, where the arms are byte-identical and the true value is 0); the placebo column reports a contrast whose true value is zero, and is read as a descriptive check on that column, not as a threshold for the column to its left—both are judged against the same pre-declared ±3 SESOI. At 0% there is no branch arm. The 90% *equivalence* bounds (the smallest symmetric margin the data support, not interval half-widths) for treated $\mathsf{B}-\mathsf{C}$ are ±9.09, ±4.80 and ±4.40 at 25/50/75%—all wider than the SESOI, so equivalence is *not* established at any cut—and ±2.96 for the pooled placebo; those for $\mathsf{C}-\mathsf{A}$ are in Table 15.

(a) Per-branch, all 400 candidates: the two components sum to $\mathsf{B}-\mathsf{A}$.

Cut	Delivered	$\mathsf{C}-\mathsf{A}$ environment			$\mathsf{B}-\mathsf{C}$ orphaned code		
ratio	dose	n	pt	CI_{95}	n	pt	CI_{95}
0%	0.0%	100	0	*by constr.*		undefined (no branch arm)	
25%	15.2%	100	+7.25	[+2.75, +12.25]	400	−0.50	[−4.00, +3.00]
50%	57.5%	100	+4.00	[−1.25, +9.50]	400	+1.25	[−3.00, +5.50]
75%	82.2%	100	+6.00	[+1.50, +10.75]	400	−2.50	[−6.75, +1.75]

(b) Conditional on delivered dose: candidate-level $\mathsf{B}-\mathsf{C}$ on treated pairs, beside the same-cut placebo.

Cut	Delivered	$\mathsf{B}-\mathsf{C}$ treated			$\mathsf{B}-\mathsf{C}$ placebo (`dirty`$=0$)		
ratio	dose	n	pt	CI_{95}	n	pt	CI_{95}
25%	15.2%	61	−1.64	[−10.71, +6.67]	339	−0.29	[−4.01, +3.88]
50%	57.5%	230	+0.00	[−5.78, +5.61]	169	+2.96	[−2.67, +8.72]
75%	82.2%	329	−0.61	[−5.14, +3.83]	71	−11.27	[−20.55, −1.69]
pooled placebo				—	579	−0.69	[−3.39, +2.23]

Table 15: Equivalence bounds for the cut sweep, in the format of Table 17 and on the same per-branch scale. At $n=100$ tasks none of the three $\mathsf{C}-\mathsf{A}$ cells establishes equivalence within ±3 points. Each interval describes the effect at its own cut, not a cross-cut comparison. The one cell that *does* is the pooled zero-dose placebo (bound ±2.96 on $n=579$ candidate pairs): the arms are byte-identical there, so its true value is zero and the equivalence verdict is a statement about resolution, not about a treatment effect.

Contrast	n	mean	90% CI	equivalent at ±3?	MDE
$\mathsf{C}-\mathsf{A}$ environment, 25% cut (dose 15.2%)	100	+7.25	[+3.50, +11.50]	no (±11.50)	6.80
$\mathsf{C}-\mathsf{A}$ environment, 50% cut (dose 57.5%)	100	+4.00	[−0.25, +8.50]	no (±8.50)	7.55
$\mathsf{C}-\mathsf{A}$ environment, 75% cut (dose 82.2%)	100	+6.00	[+2.25, +10.00]	no (±10.00)	6.59
$\mathsf{B}-\mathsf{C}$ zero-dose placebo, pooled	579	−0.69	[−2.96, +1.71]	**yes** (±2.96)	4.55

same thinking settings, same execution timeout, same judging harness—with only the cut fraction changed, so the three doses differ in one variable.

Free-ride exposure over the treated pairs, by cut ratio: the B zero-edit rate is 11.5% / 11.7% / 20.7% at 25/50/75%, pure free-rides (non-empty patch from a branch with no own edits) are 11.9% / 11.5% / 20.4% of B's non-empty patches, and 9.8% / 12.8% / 22.2% of B's solves come from zero-

edit branches. The C zero-edit rate is 1.6% / 1.3% / 1.5% and its non-empty-patch rate 98.4% / 98.7% / 97.0%, consistent with low zero-edit exposure after the restore across these cuts.

Additional conditional readings of the cut sweep. Across the same 400 slots, B − C point estimates are within 2.5 points of zero, with all three intervals covering zero. C − A ranges from +4.0 to +7.3 without a monotonic pattern; at the 50% cut it is +4.00 here and +1.44 on the full historical 260-task pool. These observations do not establish equality across cuts. On byte-identical `dirty=0` pairs, the pooled placebo is −0.69 points ($n = 579$, 95% CI $[-3.39, +2.23]$, 90% bound ±2.96), versus −0.29, +2.96 and −11.27 by cut. The treated 90% interval envelopes are 9.09, 4.80 and 4.40 points at 25/50/75%, all wider than the pre-declared ±3 SESOI. Each covers zero, so none establishes a nonzero effect or practical equivalence. These envelopes describe the data; they are not independently specified thresholds. The editing-based exposure series above conditions on treated subsets whose composition changes across cuts, and is not itself a scaling estimate.

P WHERE THE FROZEN ITT RULE BITES

deepseek-v4-flash has no `G1Infeasible` candidates in either branch arm, so that clause of its ITT rule is inert (the "unjudgeable" clause is not; Appendix I). gpt-oss has 65 in each branch arm (6.2% of 1,040, over 51 tasks), and the two arms' `G1Infeasible` sets coincide candidate-for-candidate, all 65 of 65—constructibility is decided at replay time, before the arms diverge—so the rule subtracts symmetrically and cannot generate the B − C contrast §T.7 is about. Where the rule is *not* inert is C − A: A has zero `G1Infeasible` by construction, so part of gpt-oss's environment reading is branch-construction failure. That reading is in any case uninterpretable for the larger reason of §4—its A arm ran on a different inference protocol—and the two models' C − A values are not to be differenced (Appendix G).

Q PROCESS METRICS

The protocol-phase wall-clock difference, and which way its offset points. Wall clock here is the elapsed time of the whole protocol phase under shared load, not a measure of model effort; we name it accordingly and do not read it as thinking time. It carries one structural offset, and that offset points *against* the treated finding. All 386 `dirty=0` C candidates in the pool have `revert_dirty_before=0`, so arm C runs `git checkout -- .` and `git clean -fd` there with nothing to restore—two extra commands arm B does not run—and this adds overhead to C. It does not guarantee the sign of total wall-clock time under shared load. The measured placebo difference is indeed negative, −180.8 s, 95% CI $[-406.3, +53.8]$, an interval containing zero. The treated difference runs the other way at +586.6 s, $[+416.8, +758.9]$: the structural offset would have pushed it *down*, so it is not what produces the result. Both intervals are visibly asymmetric, which is why we quote endpoints rather than half-widths.

R THE HANDOFF DEBT RE-READING: CONSISTENCY CHECKS

KC & Budathoki (2026) (we cite arXiv v2) interrupt a coding agent at deterministic handoff points, freeze the repository, and hand it to a successor under several views—of which *repository state only* is the warm start in its purest form—scoring takeover success as a final repository state the official SWE-bench harness resolves. Crucially for us, they stratify the handoff points by what the successor inherits, exactly so as to "separate two takeover cases that can share the same final solved label" (Appendix Table 10 in v2). In one stratum of 61 of their 181 points the checkpointed repository "is already resolved and a successful takeover keeps it resolved"—the label comes from the same official validation used for the final score, so that stratum *is* $Y^{\text{prefix}} = 1$. Read as a transition matrix, Eq. 1 says no takeover there can add a solution: the reported repository-state-only figure of 77.0% (47/61, the Qwen→Devstral successor) is both a final solved rate and a *preservation* rate, and 14 of the 61 takeovers left a task unresolved after it had been resolved at handoff—23.0%, on a single attempt per point, and this despite their prompt instructing the successor to "preserve" changes that already satisfy the task. The integer counts reconstruct the published one-decimal percentages under binary, equally weighted outcomes; they are not newly observed run-level data. **We report those counts descriptively, without an interval.** The full set contains 181 handoff

Table 16: deepseek-v4-flash: process metrics, B − C, over exactly the strata of Table 18, referenced from §5.5. *Clean finish* is the fraction exiting via `submitted`; p is a two-sided exact sign test; brackets are 95% **task-clustered** paired bootstrap intervals, as used for the main inferences (Appendix T.1). Wall clock is measured under shared load and carries a structural offset from the two extra `git` commands in C. That overhead favours an earlier B finish, but does not determine total wall-clock time; see the text. The Δ model-calls interval contains zero in the treated stratum, which is why the process claim is led by clean finish. **Arm levels behind each** Δ (B versus C): placebo, calls 173.3 vs 172.2, wall 3235 vs 3416 s, clean finish 59.2 vs 62.1; treated, 133.6 vs 129.1, 3098 vs 2512 s, 75.2 vs 85.0; both really edited, 141.3 vs 133.5, 3212 vs 2556 s, 79.9 vs 85.9.

Stratum	n	Δ model calls	Δ wall clock (s)	Δ clean finish (pt)
`dirty`=0 (placebo)	380	+1.1 [−5.1, +6.8]	−181 [−406, +54]	−2.9 [−7.8, +2.0] p=.32
`dirty`>0 (treated)	648	+4.5 [−1.1, +10.4]	**+587** [+417, +759]	**−6.9** [−14.1, −5.7] $p=2\times10^{-6}$
both really edited	537	**+7.8** [+2.3, +13.4]	**+655** [+481, +831]	**−6.0** [−10.1, −1.9] p=.006

points from 75 source tasks, but the tables we inspected neither establish independence within the 61-point stratum nor provide its point-to-task mapping. We therefore cannot substantiate a binomial independence assumption or construct a task-clustered interval from those tables. Two consistency checks make the reconstruction auditable (Appendix R). The same already-solved stratum reports repository-only final solved rates of 88.5% for Qwen→Qwen and 95.1% for Qwen→Gemma. We use Qwen→Devstral as the worked preservation-loss example because it has the largest observed loss among these three successors, not as an estimate of typical takeover behaviour. The authors already distinguish these outcomes from completing unresolved tasks. Our contribution here is a common accounting notation for that distinction; it does not contradict their efficiency claim.

Two checks make the re-reading of KC & Budathoki (2026)'s stratum table (§1) auditable at the published numerical precision.

The reconstructed strata reproduce each headline at the published precision. Successes and points add separately. **Worked here for their Qwen→Qwen successor**: $(26+54+4)/(110+61+10) = 84/181 = 46.4\%$, and likewise 42.5% for Qwen→Gemma and 34.3% for Qwen→Devstral. A re-reading that did not reproduce the headline would indicate we had misidentified the strata.

One trap worth naming, since the two figures look interchangeable. The 77.0% we re-read in §1 is the *Qwen→Devstral* row of the already-solved stratum, so it recovers $\mathrm{round}(0.770 \times 61) = \mathrm{round}(46.97) = 47$ preserved solutions. The worked example just above is *Qwen→Qwen*, whose same stratum reads 88.5% and recovers 54. Checking the Qwen→Devstral 77.0% against the Qwen→Qwen numbers in this paragraph therefore yields 54, not 47, and would suggest our re-reading is wrong; both counts are correct and they belong to different successors.

The inherited share is large in every model. Across the three models, 64.3–75.8% of the repository-state-only successes fall in the already-resolved stratum, while genuine additions over the 120 not-yet-resolved points run 25.0%, 15.8% and 12.5%.

In that stratum the best result any view achieved was 60/61—an observed maximum, not a logical bound, since 61/61 is attainable in principle—so the +16 to +21 point advantages the context-bearing views enjoy on it are reductions in loss, not additional solutions. We deliberately do *not* claim that those views are net-negative there: the claim would rest on a single failure out of 61 single attempts, and we have no interval for it.

S ARM D: MEASUREMENT DETAILS

S.1 CHECKPOINT IDENTITY, IN FULL

Slot-by-slot comparison of `cut_step`, `replayed_actions`, `replayed_edit_actions`, the post-replay diffstat and the post-replay dirty-file list restricted to tracked files. Arm D's fingerprint matches *both* warm-started arms' in $975/975$ gpt-oss slots (100.00%) and $1039/1040$ deepseek-v4-flash slots (99.90%); the B-versus-C control comparison is $975/975$ and $1040/1040$. The gpt-oss denominator is 975 and not 1040 because 65 slots are `G1Infeasible` in all three arms—the donor is too short, or contains no edit action, so there is no post-replay state to compare—and the infeasible sets coincide exactly across arms. deepseek-v4-flash's pool is fully feasible and arm-D collection is closed at $1040/1040$. Figures are from `armd_identity.json`, `generated_at 2026-09-24 17:53:49`.

The one real mismatch, kept on the record. In slot `navidrome-09ae41a2 / h1` the source file the donor edited (`server/subsonic/middlewares.go`, 382 lines changed, 191 insertions and 191 deletions) has the same fingerprint on both sides, and the warm-started arms additionally carry one line of `go.mod` that the Go toolchain rewrote during replay. Toolchain-generated or not, it is a difference in the tracked tree the successor started from, and we have not established that it is harmless.

The remaining disagreements are untracked debris, and they do not flatter the successor. Before restricting to tracked files, arm D's raw dirty-file *count* differs from B's in $25/1040$ deepseek-v4-flash slots and from C's in $12/1040$. Every one is run-time debris (`core.python.<pid>`, `dump.rdb`), and the direction is that the warm-started arms have *more* of it: `core.python` occurs in 53 recorded shard rows for B, 23 for C and 0 for D, because arm D runs no successor and so never crashes one.

Disclosure: the identity is checked on a truncated fingerprint. The collector's `worktree_state()` stores the first five lines of `git status --porcelain` and `git diff --stat | tail -3` truncated at 200 characters. deepseek-v4-flash has $51/1040$ slots whose dirty-file list exceeds five entries and 33 whose diffstat hits the character limit, so for those slots the comparison is on a prefix of the true state. The mitigation is that the diffstat's final line is the global summary `N files changed, X insertions(+), Y deletions(-)`, which therefore remains inside the comparison for the $612/1040$ deepseek-v4-flash and $383/975$ gpt-oss slots that carry it: total insertions and deletions are compared even where the per-file list is clipped. This is why we write "fingerprint agreement" and not "byte-identical checkpoints".

T PROTOCOL, STATISTICS AND SUPERSEDED EVIDENCE

T.1 METRICS, STATISTICS, AND EQUIVALENCE TESTING

Exactly two outcome metrics, never mixed in one number: `pass@8`, the per-task indicator that at least one of the eight candidates resolved the task, and `avg@8`, the mean per-candidate resolve rate. Tests are two-sided exact paired sign tests on discordant pairs; p-values are uncorrected. The testing unit is a task for `pass@8` and a slot for candidate-level contrasts. The slot sign tests do not adjust for within-task dependence; the task-clustered intervals are the main basis for inference.

The primary inferential intervals are $10{,}000$-draw paired bootstraps (seed 12345) over sorted pair lists, resampling tasks so that a task's four branches enter and leave together. Candidate slots share task-level conditions, so independent slot resampling need not capture their dependence. We retain slot-bootstrap and normal-approximation comparisons below to show the effect of that choice; task clustering does not necessarily widen every interval. Independent slot resampling narrows the treated $\mathsf{B}-\mathsf{C}$ 95% interval from $[-6.68, +1.56]$ to $[-5.71, +0.77]$ on deepseek-v4-flash and from $[+12.4, +22.1]$ to $[+13.4, +20.9]$ on gpt-oss, and it narrows $\mathsf{B}-\mathsf{D}$'s deepseek-v4-flash interval from $[+9.5, +18.7]$ to $[+10.7, +17.6]$ and $\mathsf{C}-\mathsf{A}$'s 90% interval from $[-1.15, +4.04]$ to $[-0.67, +3.65]$. We therefore quote explicit interval endpoints throughout rather than $\pm$ half-widths, which would assert a symmetry the clustered bootstrap does not have. The exception is the equivalence *bound* column, which is symmetric by definition.

Clustering does not uniformly widen. On the placebo strata, where the two arms are byte-identical and the within-task correlation of the difference is near zero, **clustering narrows** the interval by a factor of about 0.88 — that is, the clustered interval there is *tighter* than the slot one, not a re-widening of ours. On gpt-oss's $\mathsf{B} - \mathsf{D}$ it slightly reduces the standard error ($0.64 \to 0.63$), because only seven of 411 slots are discordant and they fall on seven distinct tasks.

"Not significant" communicates nothing about what has been excluded, so for the contrasts that close the channel we pre-declare a smallest effect size of interest of ± 3 percentage points—the range in which warm-start gains would motivate the engineering—and test equivalence by two one-sided tests through the paired bootstrap (the 90% interval lies inside $\pm\delta$ exactly when TOST rejects non-equivalence at $\alpha = .05$ against margin δ), reporting per contrast the smallest symmetric margin the data support and the effect detectable at 80% power. Where equivalence fails we say so and fall back to the one-sided bound.

Sparse discordance and uncertainty. In the historical gpt-oss treated analysis, 404 of 411 slots retain the same prefix and final score; six change positively and one negatively. Seven discordant pairs could produce a two-sided exact p as small as 0.0156, so the observed $p = 0.125$ is not a test-resolution floor. Sparse observed changes do not remove uncertainty about the population effect. We do not estimate a cross-model effect ratio: the treated sets differ and this gpt-oss interval includes zero.

T.2 The untracked-file exposure of the patch convention

Because the patch convention is `git add -A`, untracked files a session happened to leave in the repository are staged into the scored diff, and the frozen rule discards a patch above 5 MB. Both are routes by which run-time debris could change a score, so we measured them rather than assume they are inert.

Arm D is clean: of its 600 (deepseek-v4-flash) and 366 (gpt-oss) non-empty patches, zero contain `core.*`, `*.rdb` or `*.pyc` entries, zero exceed the threshold, and the 99th-percentile patch is 463 KB—an order of magnitude below it.

The exposure is on the successor side: 34 deepseek-v4-flash B slots and 17 deepseek-v4-flash C slots carry debris, and gpt-oss has none in either arm. The effect on the reported numbers is nil: of the 51 affected slots, 26 are in the treated subset and none of those exceeds the threshold. The one slot that does exceed it—an `element-web` session that unpacked an npm tarball into the repository root, 11.7 MB—is not treated and consists entirely of new files, so filtering it leaves an empty patch, which the frozen rule already scores unresolved. Filtering and not filtering give identical scores.

T.3 Judging coverage: the per-cell composition

A closed slot reaches its outcome by one of three routes: a readable test report, no readable tests after the third attempt (closed unresolved by the frozen rule), or an empty patch (unresolved mechanically). For arm D the split is $540/60/48$ of deepseek-v4-flash's 648 treated slots and $254/112/45$ of gpt-oss's 411. Per cell:

- $1 \to 1$ *inherited and preserved* is entirely test-backed on both sides—$276/276$ on deepseek-v4-flash and $70/70$ on gpt-oss have a readable test report in arm D *and* in arm B. The recorded cases are therefore supported on both sides. This does not exclude additional inherited solutions hidden by missing tests in other cells.
- $0 \to 1$ *added by the successor* is not uniformly a test-confirmed new fix. Of deepseek-v4-flash's 118, 64 have readable tests on both sides; for 28 arm D returned no readable tests and for 26 it submitted an empty patch, so those 54 are "the checkpoint did not score" rather than "the checkpoint was tested and failed". On gpt-oss the asymmetry is starker and we state it plainly: of the 6, 3 have readable tests on both sides and **the other** 3 **are arm-D empty patches**—the checkpoint submitted nothing, so it was never tested. Two counts should be kept separate here: 3 **of the** 6 $0 \to 1$ **slots are arm-D empty patches**, and the *net* numerator of $\mathsf{B} - \mathsf{D}$ is $6 - 1 = 5$. The point is not a ratio between them but the composition of the 6: in half of the slots where the successor is credited with adding a solve, the inherited tree submitted nothing and was therefore never tested against it.

- $1 \to 0$ *ended unresolved* is the cell the name was chosen for. Of deepseek-v4-flash's 27, only 19 have a readable failing test report in arm B; 5 submitted an empty patch and 3 closed without readable tests. gpt-oss's single slot closed without readable tests in arm B and is therefore not an observed regression at all.

We do not change the scoring rule to accommodate this: the frozen rule was declared before the data and charges every arm identically. The reported cells are complete for this scoring rule. Their interpretation as true-correctness transitions, including the overall inherited share, remains limited by missing tests and evaluator validity even though every observed $1 \to 1$ case has tests on both sides.

T.4 D − A: THE PREFIX ON ITS OWN

The -23.7 points read as: submitting the 50% prefix on its own, with no successor, loses to one fresh full-budget sample by about a quarter of the scale. Arm D spends zero model calls and runs no session, so this is not a policy comparison at equal cost. It compares the measured inherited baseline with fresh sampling; B − D separately measures net score change from that baseline. The negative D − A contrast shows that the checkpoints score below fresh full-budget sessions.

Read per warm-started branch over the full 260-task pool, D − A $= -23.65$ points, 90% CI $[-27.02, -20.38]$, 95% CI $[-27.69, -19.81]$, discordant counts 12:129. At the arm level (diluted by a factor of two, Eq. 3) it is -11.83, 90% CI $[-13.51, -10.19]$. On the treated slots alone, read candidate-level, it is -15.59, 95% CI $[-20.12, -11.01]$; the two populations are not interchangeable and we do not mix them in one row. Because arm D runs no successor it consumes no model-call budget, so none is a cost-matched comparison against arm A. These contrasts compare the inherited tree with fresh sampling; B − D instead compares the successor's final score with the measured checkpoint score.

T.5 SCAFFOLD, MODEL AND CONTAINER CONFIGURATION

The scaffold is SWE-agent (Yang et al., 2024a) with a 250 model-call budget, a per-command execution timeout, and the standard `bash` + `str_replace_editor` tool set. Branch construction, replay, revert-in-place, prefix-only extraction and budget accounting are a wrapper around the unmodified scaffold; no scaffold source was patched.

Both models are self-hosted on SGLang so that sampling settings are under our control and no vendor-side change can occur mid-experiment: deepseek-v4-flash at $T = 1.0$, top-p 1.0, thinking budget maximal; gpt-oss-120b at $T = 1.0$, reasoning effort high. Per-model detail is in Table 11.

All runs execute in CPU containers, one per candidate, under a concurrency cap validated by same-task paired comparison (Appendix M) and identical across arms. §5.4 adds 25% and 75% cells on a 100-task subset (400 candidates per arm per cut), reusing the 50% configuration verbatim with only the cut fraction changed.

Post-replay, 100% of branch containers are valid repositories; post-restore, the arm C tree is byte-clean in 90.0% (deepseek-v4-flash) and 90.5% (gpt-oss) of candidates, every residual being a git submodule pointer that `git checkout -- .` cannot restore and that is not source the agent could be credited for (Appendix N).

The unjudgeable clause bites unevenly: the 519 affected candidates are 4.1% of each deepseek-v4-flash branch arm's non-empty patches but 29.6% and 29.3% of gpt-oss's, which is why the sensitivity in Appendix I matters more for gpt-oss than for deepseek-v4-flash.

T.6 EQUIVALENCE TESTING ON DEEPSEEK-V4-FLASH AT THE 50% CUT

Three readings, stated separately because they license different claims, with the equivalence table itself in Appendix T.6. *(i)* On the per-branch scale—what the practitioner buys—none of the three unconditional contrasts establishes equivalence within ± 3. At $n = 260$ tasks their normal-approximation MDEs exceed 4.4 points; equivalence itself is assessed from the 90% intervals. What we can say is one-sided and is the claim we make: **a per-branch environment-advancement benefit above $+4.04$ points is excluded, and the warm start as a whole buys no more than $+3.65$**

Table 17: Equivalence testing on deepseek-v4-flash at the 50% cut, 260-task pool, SESOI = ± 3 points, 10,000-draw task-clustered paired bootstrap (seed 12345). Parenthesised is the smallest symmetric margin the data support; MDE is the effect the contrast has 80% power to detect (Figure 4). The upper block is the primary per-branch scale, on which the decomposition closes exactly; read arm-level each entry is exactly halved (Eq. 3) and all three are then equivalent at ± 3 (bounds ± 2.02, ± 1.83, ± 1.54; Appendix D). All three rows are per-branch and unconditional; the B − C contrasts conditioned on the delivered dose are in Appendix E, and the cut sweep in Table 15.

Contrast	n	mean	90% CI	equivalent at ± 3?	MDE
C − A environment	260	$+1.44$	$[-1.15, +4.04]$	no (± 4.04)	4.48
B − A warm start as reported	260	$+1.06$	$[-1.54, +3.65]$	no (± 3.65)	4.41
B − C orphaned code, all candidates	260	-0.38	$[-3.08, +2.21]$	no (± 3.08)	4.55

(90% upper bounds), against point estimates of $+1.44$ and $+1.06$. *(ii)* At the system level, the same data establish equivalence to zero within ± 3, with bound ± 2.02. The system-level difference is half the per-branch difference by construction, so this is the same evidence on a different scale. *(iii)* Conditional on the dose delivered, orphaned source code's 90% upper bound is $+0.92$ points on the treated stratum and $+0.00$ where both branches really edited. These intervals rule out gains larger than about one point, but they **no longer exclude zero**; a harm of 6–7 points is not excluded either. (In an earlier version we computed these by resampling candidate slots and reported upper bounds of $+0.31$ and -0.37, the second strictly negative. Resampling tasks instead—the correct unit, since a task's four branches are not independent—widens both and moves the second to exactly zero. We report the clustered numbers and the weaker claim they support.)

T.7 The gpt-oss free-ride chain, in full

The early analysis used behavioural proxies for inheritance. This appendix reports those proxies alongside the later checks against arm D and the four-cell accounting in §5.2.

Five audited examples, with hashes. These are the cases §5.2 refers to: gpt-oss slots in which the resumed arm was credited with a solve the control did not get, while the branch itself made no repository-modifying edit and its submitted patch was byte-identical to the prefix-only one. Each satisfies all five conditions simultaneously: (i) arm B resolved the task; (ii) the branch's own edit count is zero; (iii) $Y^{\text{prefix}} = 1$, so the checkpoint already resolved it; (iv) arm C did not resolve it; and (v) $\text{sha256}(\text{B patch}) = \text{sha256}(\text{D patch})$. They fall on two tasks and five donor indices:

Task	Donor index	SHA-256 prefix (B patch = D patch)
`ansible__ansible-3b823d908e`	h_0	`edda3c85cb`
	h_1	`a9c21ccb1d`
	h_2	`9d16e0346a`
`ansible__ansible-502270c804`	h_0	`0d7a850e63`
	h_1	`66eaf8c1ad`

We report five audited cases rather than the full census deliberately: the same five conditions hold on 58 gpt-oss slots over 42 tasks and on 8 deepseek-v4-flash slots over 7, so the audited list illustrates the larger census; it is not a separate estimate of its prevalence. "Inspected" here refers to examination of the recorded artifacts.

Table 18 contains the number a warm-start study would report: on gpt-oss, over the 411 paired branches whose replay landed edits, the resume arm resolves 18.5% against the revert arm's 1.5%—B−C = +17.0 points, 95% CI $[+12.4, +22.1]$, with an interval excluding zero. It is a valid estimate of the total effect of retaining code state, and §5.2 shows how little of it the successor produced.

Composition, and whose solves these are. B submits a non-empty patch in 373 of the 411 pairs (90.8%) against C's 43 (10.5%), issues zero edit actions of its own in 347 (84.4%), and draws 310 of those 373 patches (**83.1**%) and 62 of its 76 solves (**81.6**%) from branches that edited nothing—

Table 18: Candidate-level B versus C on gpt-oss, paired on (task, candidate index), frozen ITT. `dirty` counts modified files after replay, before the restore; these counts agree across both arms on the gpt-oss slots reported here; "worked" means the branch issued an edit action of its own, and the indented rows partition the treated pairs. Intervals are 10,000-draw paired bootstraps resampling *tasks*, not slots (§3); endpoints are given explicitly because the clustered intervals are not symmetric. Rows conditioning on what a branch did are post-treatment: descriptive, not causal (§3.5). deepseek-v4-flash's counterpart—placebo $+3.2$ ($n = 380$, $[-0.80, +7.16]$), treated -2.5 ($n = 648$, $[-6.68, +1.56]$), both edited -3.2 ($n = 537$, $[-7.24, +0.74]$)—is decomposed in Appendix L.

Stratum	n	B	C	B − C (95% CI)
`dirty`=0 (placebo; identical arms)	564	1.4	1.6	-0.2 $[-1.44, +1.06]$
`dirty`>0 (treated, *contaminated*)	411	18.5	1.5	$+\mathbf{17.0}$ $[+12.41, +22.06]$
B idle / C idle	300	16.0	0.0	$+16.0$ $[+11.44, +20.89]$
B worked / C idle	55	21.8	0.0	$+21.8$ $[+11.32, +33.33]$
B idle / C **worked**	47	29.8	10.6	$+\mathbf{19.1}$ $[+4.44, +34.69]$
B worked / C worked (net)	9	22.2	11.1	$+11.1$ $[-22.22, +44.44]$
own-edit gate applied to the treated row	411	3.4	1.5	$+1.95$ $[-0.24, +4.08]$

73.8% of its 84 solves over all 1,040 candidates. C resolves 6 here and 15 overall, **0** of them zero-edit. The strata below condition on what a branch did and are therefore post-treatment: descriptive, not causal (§3.5).

The zero-edit versus edited cell. In 47 pairs, B issued no detected edit while C did. Arm B scored 19.1 points higher (95% CI $[+4.4, +34.7]$). This shows that a positive final-score contrast can occur without newly detected edits by the resumed branch; zero edits does not mean no model calls or inspection. Because the cell is selected by post-treatment behaviour, it does not establish whether inherited state helps successors in general. In the largest cell, 300 of 411 pairs, neither branch issued a detected edit and the scores were 16.0 for B and 0.0 for C.

What a scoring rule can and cannot recover. Keeping only pairs where both branches edited leaves $n = 9$ and a 95% interval of $[-22.2, +44.4]$: this conditional effect is too imprecise to support a directional claim. The own-edit gate is computable on every pair and removes 62 of B's 76 solves and *none* of C's 6—not because C rarely idles (86.4% of its treated branches issue no edit) but because every one of its solves came from a branch that worked. The contrast falls from $+17.03$ to $+1.95$ points (95% CI $[-0.24, +4.08]$, 90% $[+0.00, +3.73]$; sign test 14:6, $p = .115$); the 95% interval contains zero and we do not call the remainder significant. The gate does not isolate newly added solutions: of the 14 surviving B solves, 4 submit a patch touching files the branch never edited and 11 submit one whose files were already dirty after the replay, so the donor's hunks ride along into the scored diff even where the branch edited the same file. Section 5.2 measures the score transitions directly. Meanwhile the placebo stratum, where the replay landed nothing and the arms are byte-identical, gives -0.2 (95% CI $[-1.4, +1.1]$) on 564 pairs: this check is consistent with a zero contrast on that stratum, without certifying the whole pipeline.

Branch activity and observed exposure. A median gpt-oss branch makes four model calls and zero own edits and about half terminate on a malformed tool call, so its submitted diff is essentially a readout of the replayed prefix; a median deepseek-v4-flash arm-B branch makes 140 model calls and 6 own edits and exits via `submitted` in 69.3% of candidates (Table 12). deepseek-v4-flash is not immune, only less exposed: on its treated stratum 68 of B's 394 solves (17.3%) come from branches with no own edits against 1 of C's 410 (0.2%); over all 1,040 candidates the counts are 69 of 560 and 1 of 564. Hence a cheap pre-registrable screen: report the branch arm's zero-edit rate and dominant exit status—both already in the logs—before running a warm-start comparison.

Table 19: What each contrast identifies, *with the population and scale it is measured on*. These four numbers are **not** additive components of a single decomposition: they differ in model, in population (all candidates versus the treated subset) and aggregation. Identities can be applied within a model after matching the candidate set, pairing and scale; they do not justify combining all rows into one decomposition. In particular the $+17.0$ is bound to gpt-oss, the 50% cut, the `dirty>0` treated subset, candidate-level, $n = 411$; it is not a warm-start benefit against fresh sampling, whose gpt-oss comparison is confounded by inference protocol (Appendix F). *Paired-bootstrap 95% CIs for the two $\mathsf{B} - \mathsf{D}$ rows: $[0.00, +2.54]$ for gpt-oss (seven discordant pairs, sign test $p = 0.125$) and $[+9.5, +18.7]$ for deepseek-v4-flash (sign test $p < 10^{-4}$). †$\mathsf{B} - \mathsf{C}$ on gpt-oss is $+17.0$ points, 95% CI $[+12.4, +22.1]$, sign test 72:2.

Contrast	Estimand	Question answered	Model	Population	Scale	Reading
$\mathsf{B} - \mathsf{D}$	$\mathbb{E}[Y^{\text{final}} - Y^{\text{prefix}}]$	What did the successor add on top of the inherited tree, net of what ended unresolved?	gpt-oss	treated, n=411	cand.	$+1.2^{*}$
			deepseek-v4-flash	treated, n=648	cand.	$+14.0^{*}$
$\mathsf{B} - \mathsf{C}$	$\mathbb{E}[Y_{\mathsf{B}}^{\text{final}} - Y_{\mathsf{C}}^{\text{final}}]$	What is retaining the donor's code state worth, *given* a successor runs?	gpt-oss	treated, n=411	cand.	$+17.0^{\dagger}$
$\mathsf{C} - \mathsf{A}$	$\mathbb{E}[Y_{\mathsf{C}}^{\text{final}} - Y_{\mathsf{A}}^{\text{final}}]$	What is the warm container worth once the source state is removed?	deepseek-v4-flash	pool, n=260	branch	$+1.44$
$\mathsf{D} - \mathsf{A}$	$\mathbb{E}[Y^{\text{prefix}} - Y_{\mathsf{A}}^{\text{final}}]$	What is the donor prefix worth on its own, at zero model calls?	deepseek-v4-flash	pool, n=260	branch	-23.7

Table 20: Three-arm comparison on deepseek-v4-flash, frozen ITT, denominator 260 tasks. Δ is against A; "excl." counts tasks won exclusively by A versus by the branch arm; p is a two-sided exact paired sign test on those pairs. The `avg@8` Δ is the *per-branch* reading—one warm-started branch against one fresh sample—with a task-clustered paired bootstrap CI; the arm-level reading is exactly half of it (Eq. 3) and, with the closure check for both scales, is in Appendix D. `pass@8` admits no such factorisation. gpt-oss's counterpart is in Appendix F: its A ran on a different inference protocol from its branch arms, so its Δ-against-A rows do not isolate strategy effects. Here the A row gives `pass@8A` and the B row `pass@4A+4spread`.

	`pass@8`				`avg@8`		
Arm	rate	Δ	excl.	p	rate	Δ_{branch}	CI_{95}
A pristine	70.38	—	—	—	52.98	—	—
B resume	71.15	$+0.77$	6:8	.791	53.51	$+1.06$	$[-2.02, +4.13]$
C revert-in-place	70.77	$+0.38$	6:7	1.00	53.70	$+1.44$	$[-1.63, +4.62]$

T.8 THE FOUR CONTRASTS, TABULATED

T.9 DEEPSEEK-V4-FLASH THREE-ARM LEVELS

T.10 THE OWN-EDIT GATE CROSSED WITH Y^{prefix}

On the treated slots, crossing the own-edit gate with the judged outcome of the checkpoint shows the gate's observed errors in this sample. Of gpt-oss's 76 resumed-arm solves the gate discards 62, and every one of those has $Y^{\text{prefix}} = 1$; of the 14 it keeps, 6 have $Y^{\text{prefix}} = 0$ and 8 have $Y^{\text{prefix}} = 1$; it discards no solve with $Y^{\text{prefix}} = 0$. On deepseek-v4-flash's 394 solves the same four counts are 68, 118, 208 and 0. Over the full treated sets the 2×2 is $55/9/285/62$ on gpt-oss's 411 slots and $319/228/26/75$ on deepseek-v4-flash's 648 (reading: own-edits>0 with Y^{prefix}=0, own-edits>0 with Y^{prefix}=1, own-edits=0 with Y^{prefix}=0, own-edits=0 with Y^{prefix}=1).

The gate removes no recorded added solve on these slots but retains inherited solves: $8/14$ of the retained gpt-oss solves and $208/326$ of the retained deepseek-v4-flash solves have $Y^{\text{prefix}} = 1$. This is an empirical calibration for this detector and frozen scoring rule, not a guarantee of the same error direction on other data. The proxies can flag exposure; arm D measures the score transitions that the proxies cannot separate.

U REPRODUCTION ON THE FULL 519-TASK SPLIT

The 260-task pool is a fixed-seed uniform subsample of the 519-task filtered split (Appendix B), and the main-text historical analyses use that subsample. For gpt-oss the four arms have terminal ITT outcomes on all 519 tasks. The results below report the full pool alongside the historical 260-task analysis; they do not relabel the earlier numbers. System-success panels use all tasks in each stated pool and display both eight-candidate systems. The separate accounting panels condition on treated slots, so their denominators and scores differ from those system panels.

Which records support which results. The following saved outputs are relative to the project harness directory. Full-pool system scores come from `flash519_pass8a_result.out` and `paper/frozen_519_final_20260924_2356/oss_revert_arm_table.out`; historical system scores come from `paper/frozen_20260924_raw/revert_arm_table.out` and its `oss_revert_arm_table.out` counterpart. The completed treated-slot contrasts for both models are in `flash519_bd_result.out` and `flash519_bd_cluster.json`. Their terminal ITT coverage counts do not certify readable tests for every candidate.

The gpt-oss block of `paper/frozen_519_final_20260924_2356/arm_d_four_cell.out` supplies the $658/11/2/136$ cells on 807 treated slots. Its $2076/2076$ cut-step agreement compares D, B and C; its diffstat agreement compares only D with B in the inspected producer. Missing fields can agree, and these counts include infeasible slots. These limited checks do not establish full-tree identity. The deepseek-v4-flash block in that same saved output is an earlier incomplete snapshot with a mismatch warning; a final full-pool four-cell output has not been located among the retained sources reviewed here. We therefore report deepseek-v4-flash's completed treated contrasts without promoting the earlier four-cell snapshot to a final result.

The detailed multi-field fingerprint record, `armd_identity.json`, explicitly covers 260 tasks; so does the per-cell readable-test composition in `paper/frozen_20260924_raw/coverage_composition.out`. Those historical checks do not establish full-pool checkpoint agreement or readable-test composition. Source inspection at review time also does not by itself bind an output to the exact producer version used when it was generated.

Point estimates and precision on the two nested pools. The gpt-oss interval excludes zero on the 519-task pool but not on the 260-task subsample. The point estimates are similar ($+1.12$ and $+1.22$), as are the observed discordance rates ($13/807 = 1.61\%$ and $7/411 = 1.70\%$). On the larger pool, 794 of 807 treated slots retain the same score, and the inherited share is 92.5% compared with 92.1%. The larger pool contains more discordant pairs (13 versus 7) and gives a narrower interval. Because the pools are nested and their composition differs, this comparison neither provides independent replication nor isolates sample size as the cause of the increased precision.

Table 21: gpt-oss on the historical 260-task pool and formal 519-task pool. The first panel gives system success over all tasks in each pool; fresh and branch sessions use different APIs, so their difference does not isolate the warm-start strategy. The second panel gives accounting on treated slots, where replay landed edits in both warm-started arms. Intervals are 95% task-clustered paired bootstraps, as in the main analysis. The two pools are listed side by side; we do not subtract one column from the other, and neither validates the other—they are nested, the 260 pool being a subsample of the 519 split, so the columns share candidates and agreement between them is not independent corroboration.

System success: all tasks in each pool		
Pool	`pass@8A`	`pass@4A+4spread`
260 tasks (historical)	49.23%	44.62%
519 tasks (formal)	47.78%	42.77%
Checkpoint accounting: treated slots		
	260 pool (body)	519 split
treated slots (n)	411	807
tasks contributing	187	363
$0 \to 0$ neither	334	658
$0 \to 1$ added by the successor	6	11
$1 \to 0$ ended unresolved	1	2
$1 \to 1$ inherited and preserved	70	136
inherited share of B's solves	$70/76 = 92.1\%$	$136/147 = \mathbf{92.5}\%$
B − D	+1.22	+1.12
95% CI	$[0.00, +2.54]$	$[+\mathbf{0.25}, +\mathbf{2.01}]$
discordant pairs	7 (6:1)	13 (11:2)
sign test	$p = 0.125$	$\mathbf{p = 0.023}$

deepseek-v4-flash on the 519 split. All expected deepseek-v4-flash candidates have terminal ITT outcomes on this split: arms B, C and D at $2076/2076$ candidates each, and arm A at $4152/4152$ across its eight. The reported contrasts below use these completed ITT records; terminal status does not establish readable-test coverage for every candidate. Collection closed progressively, and until it did we withheld each contrast whose arms were short rather than publish a partial reading—the coverage argument of Appendix J applied to our own numbers.

Table 22: deepseek-v4-flash on the formal 519-task pool and historical 260-task pool. The first panel gives system success over all tasks in each pool. The second panel gives completed candidate-level contrasts (run at 2026-09-25 13:27) over *treated* slots (replay landed edits in both warm-started arms): $n = 1256$ slots over 416 tasks on 519, $n = 648$ over 213 on 260. Intervals are 95% task-clustered paired bootstraps. The two pools are listed side by side; we do not subtract one column from the other, and neither validates the other—they are nested, the 260 pool being a subsample of the 519 split, so the columns share candidates and agreement between them is not independent corroboration.

System success: all tasks in each pool		
Pool	`pass@8A`	`pass@4A+4spread`
260 tasks (historical)	70.38%	71.15%
519 tasks (formal)	68.98%	69.17%
Candidate-level contrasts: treated slots		
	260 pool (body)	519 split
B − D successor's contribution	+14.04	+12.58
95% CI (task-clustered)	[+9.48, +18.73]	[+9.57, +15.55]
discordant pairs (B>D : B<D)	118:27	199:41
B − C orphaned code	−2.47	−0.16 ($p = 0.946$)
D − A prefix vs fresh sample	−15.59	−14.89

Full-pool D − A: an unresolved source link. The historical per-branch result over all 260 tasks is -23.65 points (95% task-clustered CI $[-27.69, -19.81]$). Section 6e of the project note `paper/FROZEN_NUMBERS_20260924.md` also records -23.17 ($[-25.87, -20.42]$) over all 519 tasks. The direct 519-task stdout or JSON behind that entry has not been located among the retained sources reviewed for this draft; the available `armd_contrasts.json` identifies the 260-task pool. We retain the 519 entry as a project-note-only record with an unresolved source link, rather than an independently reproduced result. It does not confirm the historical estimate. The two stated pools are nested, and the completed treated-slot D − A in Table 22 uses a different selected population; neither supplies the missing source for this full-pool entry.

Eight-candidate system results on the 519 split. Over the 519-task pool, `pass@8A` is 68.98% and `pass@4A+4spread` is 69.17% ($n = 519$ tasks). The corresponding mean candidate scores are 51.16% and 50.99% ($n = 4152$ slots per system). Substituting resume moves `pass@8` by $+0.19$ points (per task, $n = 519$; sign test $p = 1.000$) and `avg@8` by -0.17 points (per candidate, $n = 4152$; 95% CI $[-1.23, +0.84]$). Substituting revert moves `pass@8` by -0.39 (per task, $p = 0.832$) and `avg@8` by -0.26 (per candidate, $[-1.37, +0.79]$). The 187 tasks on which all of $A_{s_0}, \ldots, A_{s_3}$ fail are reported as the subset where the systems' `pass@8` scores could differ; `avg@8` can differ elsewhere as well; they are *not* the denominator of either statistic.

The mean-score intervals do not establish no effect. For `avg@8`, both substitution intervals include zero and extend roughly a point in either direction. These intervals describe uncertainty in `avg@8`, not `pass@8`, for which no corresponding interval is reported here. We make no claim about what this design could or could not have resolved—that would require a minimum detectable effect or an equivalence bound for `pass@8` and `avg@8`, and we computed neither. We deliberately quote no external "noise floor" beside these numbers: the zero-value control we have, the B-versus-C placebo, is a conditional estimand on the `dirty=0` subset with its own pairing and its own statistic, so its interval describes that contrast and not this one. An interval borrowed from a different quantity would misstate this one's resolution in either direction.

deepseek-v4-flash's other 519 contrasts are reported alongside, and one of them shows what the coverage rule buys. The candidate-level D − A on the *treated* subset—a different estimand from the per-branch figure above, on a different population and denominator—reads $+12.13$ on the 519 split at the 2026-09-25 00:01:28 snapshot ($n = 1212$ slots over 409 tasks, with the reference half at 51.3% coverage), against -15.59 on the 260 pool ($n = 648$ over 213). **We do not place those two beside the per-branch pair, and we draw no trajectory through them**: the treated subset is defined by both warm-started arms being dirty, so at that snapshot the quantity could not be computed at all, and the run that produced $+12.13$ prints its own warning that contrasts involving the short arms were unreadable on this pool. Arm C has since closed, and the value on the completed split is reported in Table 22. Under the frozen intent-to-treat rule an uncollected candidate scores unresolved, so a short arm is depressed by its own incompleteness rather than by anything the experiment did. This is a historical incomplete-collection example, not a current estimate; the completed treated contrast is the one reported in Table 22.

V Reproduction details and artifact status

The following records and procedures are needed to reproduce the reported analyses. They are maintained in the project records; this draft does not yet provide an anonymous artifact access link. Complete reproduction requires the exact input artifacts, producer versions, task lists and parameters associated with each reported result.

Task pool. The formal 519-task identifier list is `EVAL_POOL_519.txt`; `EVAL_POOL_519_RATIONALE.md` records its filtering provenance from the 731-task SWE-bench Pro public split (Appendix B). Both are stored in the project harness directory. The separate `flash_gen_priority_260.txt` records the historical fixed-seed subsample shared by both models and all four arms. The 100-task sweep uses the first 100 historical tasks in canonical order (§5.4). These lists retain their stated identities: historical 260-task analyses and the 100-task sweep are not relabelled as results on the formal 519-task pool.

Arm construction. The cut policy (`actions`, at 25/50/75%), the budget rule (`edits`: $B' = 250 - |\text{prefix edit actions}|$), and the revert-in-place step (`git checkout -- .` followed by

`git clean -fd`, applied after replay and before the session starts) are specified by the protocol. Arms B and C share cut and budget settings; they differ in whether the restore is applied before the successor begins.

Scoring rule. The frozen intent-to-treat rule is stated in full in §3.4: `G1Infeasible`, empty patch, oversized patch, unjudgeable after three attempts, and uncollected all score as unresolved, with the full stated task pool retained in every arm: 260 for the historical main analyses, 519 for the full-pool results, and 100 for the sweep. Conditional quantities additionally state their selected candidate-set denominators. The rule was fixed before these runs and applies identically to all four arms. As collected, no candidate is uncollected, but the "unjudgeable" clause is *not* inert: in the historical 260-task analysis it absorbs 519 candidates across the six A/B/C arms—a count of candidates, not of tasks, and unrelated to the 519-task split— and Appendix I gives the per-arm counts and a per-protocol sensitivity analysis for the contrasts it could affect.

Statistics. Sign tests are two-sided exact paired tests on discordant outcomes, with uncorrected p-values. The unit is a task for `pass@8` and a candidate slot for candidate-level contrasts; slot-level tests do not adjust for within-task dependence. Primary inferential intervals are 10,000-draw paired bootstraps (seed 12345) over deterministically ordered pairs, resampling tasks together with all their slots. Repeating an analysis with the same inputs, implementation and parameters makes these bootstrap draws reproducible. Slot-bootstrap and normal-approximation intervals are retained only as comparisons. For example, deepseek-v4-flash's treated B − C is -2.47 ± 3.3 under the normal approximation, $[-5.71, +0.77]$ under a slot bootstrap and $[-6.68, +1.56]$ under the task-clustered bootstrap. We use explicit endpoints for inferential intervals; the symmetric $\pm$ equivalence bound has a different meaning.

Equivalence testing. The SESOI of ± 3 percentage points is declared in §3 before the equivalence analysis and is not tuned to the data. TOST is implemented as the 90% paired bootstrap interval on the same pairing used for the corresponding point estimate, with the same seed; the reported "bound" is $\max(|\text{lo}|, |\text{hi}|)$ and the reported MDE is 2.802 times the bootstrap standard error of the paired difference. The power curves of Figure 4 use the same standard errors and a normal approximation. The project records include the script used to produce the equivalence table and figure.

Edit attribution. The branch-own-edit detector is specified in Appendix C, together with the arm C validation that constrains its false-negative rate. Reproducing the decomposition requires this detector, its input trajectories, the checkpoint and final scores, and the candidate-selection metadata.

Determinism caveats. Agent rollouts are not deterministic: both models are sampled at temperature 1.0 and the container environment introduces timing variation, so re-running will not reproduce individual trajectories. Replay re-executes recorded action strings but is not guaranteed to reproduce identical state. The measured fingerprint agreement is $975/975$ for gpt-oss and $1039/1040$ for deepseek-v4-flash, with one tracked-state difference: a `go.mod` line rewritten by the Go toolchain whose harmlessness has not been established (Appendix S.1).

Infrastructure sensitivity. Appendix M documents a confound we encountered and controlled: CPU contention above a concurrency threshold silently degrades rollout quality by pushing ordinary shell commands past the per-command timeout. Anyone reproducing this work at scale should validate their concurrency setting with a same-task paired comparison before collecting confirmatory data.

A negative result about our own pipeline. Appendix N documents a silent replay failure that voided an entire earlier collection of arm B, and the invariant now checked before any scoring. This failure and its diagnostic checks are part of the reproducibility record.