BEYOND BINARY TESTS: AUDITING EXECUTION-DERIVED REWARDS FOR REPOSITORY-LEVEL AGENT RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Repository-level coding agents are attractive targets for reinforcement learning because software can be executed, yet training commonly reduces that evidence to a single test-pass bit. We study two limitations of this paradigm: verifier trustworthiness, because passing tests need not imply that the intended task was solved, and reward starvation, because uniformly failing rollout groups provide no within-group learning signal. We audit execution-derived rewards for selectively certified repository tasks by separating two choices: which executable evidence the environment collects and how that evidence is aggregated. Holding hidden reference executions fixed, we characterize exactly when strict all-pass and fractional aggregation can select differently, then test semantic validity and held-out utility. Across 42 prospectively evaluated model-repair tasks, raw selection opportunity is 0/42 at the predeclared $K = 4$ endpoint (an exact zero empirical bound on aggregation-only selection gain for this set) and 1/42 at $K = 8$ and $K = 16$. In 288 natural agent patches, the sole raw opportunity among 18 comparable pools disappears under repository-native semantics. Controlled studies exhibit positive held-out differences on some raw-opportunity pools, but gains are concentrated; a task-disjoint replication has 2/16 raw opportunities and an average-effect interval including zero. Complementary historical real-agent data show that an earlier execution-derived signal recovers variation in 19/22 terminal-binary-flat rollout groups versus 8/22 for repository-test fraction. A frozen adversarial campaign withholds full credit from 21/22 qualifying visible-verifier exploits, while a separate optimizer-interface check verifies a nondegenerate optimizer signal without supporting a causal policy comparison. These results motivate selective executable supervision: determine whether the bottleneck lies in reward semantics, aggregation, evidence construction, or candidate generation rather than forcing every task into a binary or pseudo-dense reward template.

1 INTRODUCTION

Repository-level coding agents are attractive targets for reinforcement learning because software execution supplies cheap, automatically checkable feedback. Yet a terminal pass/fail reward creates two distinct failure modes. *Verifier trustworthiness* asks whether the executed checks measure the intended behavior; *reward starvation* asks whether the resulting reward retains useful distinctions among sampled candidates. We study both without conflating them.

An executable reward therefore has two conceptually distinct layers: the *evidence* collected from the environment and the *aggregation* that reduces those observations to a training or selection signal. Comparing visible binary tests with a graded hidden verifier changes both at once. Our main experiments hold executable evidence and candidate pools fixed and ask when finer aggregation can change candidate selection and when that change predicts held-out behavior. Complementary historical real-agent data ask the other question: when terminal evidence is flat, can a richer executable interface recover distinctions at all?

For the matched aggregation comparison, strict all-pass and fractional scores consume the same hidden reference executions. A finer score can improve the specified candidate selector only if

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

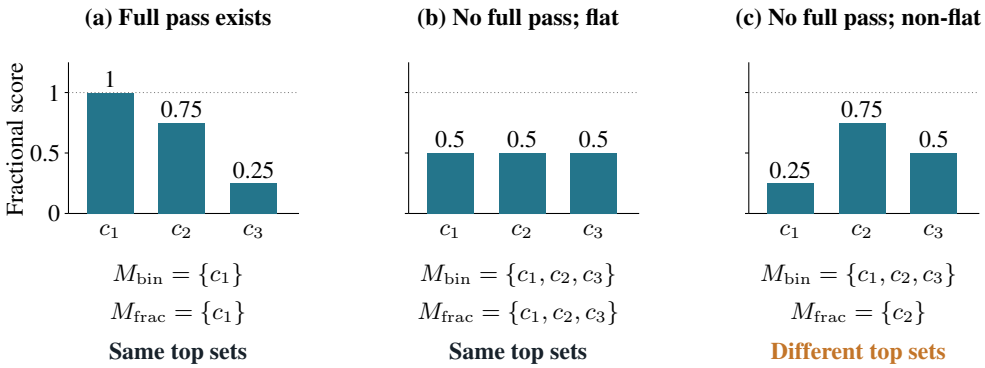

Figure 1: When reward aggregation changes selection. M_{bin} and M_{frac} denote the top-scoring sets. Scores are illustrative, not experimental observations. Binary assigns one only to full-pass candidates. Both rules therefore select the same top set in (a) and (b), while only (c) creates raw opportunity. Different top sets alone guarantee neither semantic validity nor positive held-out utility.

its extra resolution can change which candidates are chosen. Figure 1 shows the three possible candidate-pool regimes.

If a full-pass candidate exists, both rules select exactly the full-pass set. If no candidate fully passes and all fractional scores tie, both select the entire pool. Only a non-flat fractional profile without a full-pass candidate allows the top sets to differ. We call this condition *raw opportunity*. With uniform expected tie-breaking, the empirical task-macro held-out utility difference is exactly opportunity prevalence times mean difference conditional on opportunity. The algebra is elementary; its value is operational: under fixed evidence and candidates, opportunity prevalence upper-bounds the absolute selection gain obtainable by changing aggregation alone.

Opportunity alone is insufficient. In the natural Xarray case, stored fractional scores distinguish candidates that satisfy the repository's own semantic equivalence check. The apparent opportunity disappears under that equivalence (Figure 3). Even a semantically meaningful ordering need not predict held-out behavior. We therefore separate three questions: are score distinctions real, can they change selection, and does that change improve held-out utility? The prospective study measures a sparse opportunity regime; the natural audit exposes spurious resolution; controlled studies test conditional utility.

This diagnostic is motivated by executable supervision for software agents (Jimenez et al., 2024; Pan et al., 2025; Jain et al., 2025) and by methods using pass rates or process feedback as rewards (Wang et al., 2026a; Li et al., 2026; Setlur et al., 2025; Tao et al., 2026; Kim and Yun, 2026; Dai et al., 2026). A binary group can be flat because the environment does not measure behavior that distinguishes candidates, or because binary aggregation discards resolution already present in the evidence. Our matched theorem and experiments diagnose the second lever; a historical rollout study (Appendix F) supplies boundary evidence for the first. Reference behavior comes from a privately retained known-correct implementation, without human-authored process labels, a reference trajectory, or a learned reward model. We audit the reward channel rather than claiming an end-to-end policy-quality comparison.

Contributions. We provide (i) an exact matched-evidence diagnostic that factorizes empirical selection gain into opportunity prevalence and conditional utility; (ii) an evidence–aggregation audit separating score semantics, selection opportunity, and held-out utility; (iii) prospective and natural measurements showing when, under fixed evidence, matched aggregation-only gains are zero, tightly bounded, or semantically spurious; and (iv) complementary boundary evidence showing that richer executable evidence can restore within-group variation under binary collapse and withhold full credit from 21/22 qualifying visible-verifier exploits.

2 Related Work

Executable environments. SWE-bench and subsequent systems make repository execution central to agent evaluation and training (Jimenez et al., 2024; Pan et al., 2025; Jain et al., 2025; Yang et al., 2025; Zhang et al., 2026b). Other work broadens executable evaluation to feature development and environment configuration (Zhou et al., 2026; Kuang et al., 2026), runtime-faithful codebase extraction (Lee et al., 2026), semantic reasoning (Roy et al., 2026), and cryptographic code generation (Kumar et al., 2026). We instead audit the selection information supplied by reward aggregation when candidate behavior can be compared with a private executable reference.

Verifier feedback and dense rewards. VerifyBench studies reference-based reward reliability (Yan et al., 2026); reward-hacking research distinguishes verifier success from intended correctness (Wang et al., 2026b). VeRPO uses partial execution success for dense verifiable RL rewards (Wang et al., 2026a), while pass-rate rewards do not reliably outperform binary rewards in the critic-free RL study of Li et al. (2026). Other approaches use progress or process-level feedback to guide optimization (Setlur et al., 2025; Tao et al., 2026; Kim and Yun, 2026; Dai et al., 2026); SWE-RL applies reinforcement learning to open software evolution (Wei et al., 2025), while RobustTests combines faulty-code-driven test synthesis with dense pass-rate reward shaping (Zhang et al., 2026a). Those studies examine how reward formulations or training pipelines guide optimization. Our comparison instead fixes executable observations and candidate pools, then measures whether aggregation changes the selected set and its utility on withheld behavior. This identifies the selection channel under a specified rule without attributing the mechanisms of other systems' training gains. Neither fractional scoring nor differential/property-based testing is itself our contribution (Claessen and Hughes, 2000; McKeeman, 1998). The contribution is the joint audit of selection opportunity, semantic validity, and conditional held-out utility, including their operational limits on real-agent tasks.

3 A Selection-Level Diagnostic for Execution-Derived Rewards

Let task t contain candidate pool $\mathcal{C}_t$. On n frozen hidden reference calls, define candidate agreement

$$f_t(c) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{c_i = r_i\}. \tag{1}$$

The matched strict and fractional scores are

$$s_{\mathrm{bin},t}(c) = \mathbf{1}\{f_t(c) = 1\}, \qquad s_{\mathrm{frac},t}(c) = f_t(c). \tag{2}$$

They consume the same observations; only aggregation differs.

For a score s, let

$$M_t(s) = \arg\max_{c \in \mathcal{C}_t} s_t(c) \tag{3}$$

be its top-scoring set. If $b_t(c) \in [0, 1]$ denotes held-out utility, uniform expected tie-breaking gives

$$U_t(s) = \frac{1}{|M_t(s)|} \sum_{c \in M_t(s)} b_t(c). \tag{4}$$

Raw opportunity. For complete equal-length traces,

$$O_t = \mathbf{1}\left\{ \max_{c \in \mathcal{C}_t} f_t(c) < 1 \right\} \mathbf{1}\left\{ \max_{c \in \mathcal{C}_t} f_t(c) > \min_{c \in \mathcal{C}_t} f_t(c) \right\}. \tag{5}$$

If any candidate has $f_t(c) = 1$, binary and fraction have the same top set: exactly the full-pass candidates. If no candidate has full credit, binary ties the entire pool, and fractional aggregation changes the top set if and only if the fractional scores are nonconstant. Thus $O_t = 1$ if and only if the two matched selectors have different top sets (Figure 1).

Opportunity decomposition. Define the task-level held-out difference

$$\Delta_t = U_t(s_{\text{frac}}) - U_t(s_{\text{bin}}). \tag{6}$$

Whenever $O_t = 0$, the top sets are identical and therefore $\Delta_t = 0$. For an evaluated set of T tasks, let

$$\widehat{\pi}_O = \frac{1}{T}\sum_{t=1}^{T} O_t \tag{7}$$

and, when at least one opportunity exists,

$$\overline{\Delta}_O = \frac{\sum_t O_t \Delta_t}{\sum_t O_t}. \tag{8}$$

With $\overline{\Delta}_O = 0$ by convention when no opportunity exists,

$$\boxed{\overline{\Delta} = \widehat{\pi}_O \, \overline{\Delta}_O.} \tag{9}$$

Because $U_t \in [0, 1]$, $|\Delta_t| \leq 1$, and hence

$$\boxed{|\overline{\Delta}| \leq \widehat{\pi}_O.} \tag{10}$$

Equations 9–10 are exact identities for the *evaluated finite task set* and the specific matched selectors above. They are not confidence bounds on a broader task population. For a random task drawn from a fixed distribution under the same setup, the analogous population identity is $\mathbb{E}[\Delta] = \Pr(O = 1)\mathbb{E}[\Delta \mid O = 1]$ and therefore $|\mathbb{E}[\Delta]| \leq \Pr(O = 1)$; the observed $\widehat{\pi}_O$ estimates but does not determine that population probability.

Raw opportunity is not semantic validity. Equation 5 is mechanical: it describes whether the *implemented scores* can alter the decision. A raw opportunity need not represent a meaningful behavioral distinction. We therefore keep two stages separate. First compute raw O_t from frozen scores. Then, where interpretation depends on a non-flat profile, inspect whether the underlying disagreement is semantically meaningful under repository-native or task-appropriate equivalence. The exact algebra above uses raw opportunity; semantic validation determines whether that numerical opportunity deserves scientific interpretation as behavioral information.

Evidence-relative scope. Opportunity is also relative to the frozen evidence and candidate pool. A flat profile does not prove that candidates are behaviorally equivalent outside those probes. It says only that changing aggregation cannot create a selection difference from information that the current evidence did not expose. In such a regime, the next intervention should target evidence construction or candidate generation rather than aggregation.

Connection to group-relative RL. For a sampled group with rewards $r_1, \ldots, r_K$, a centered reward advantage $A_i = r_i - \bar{r}$ (or centered-and-scaled normalization) is zero for every candidate when the reward vector is constant. Thus an all-fail binary group is reward-starved through that channel, while a nonconstant execution-derived reward can restore relative signal even without changing the top set. This is only an opportunity for optimization, not evidence of a better learned policy; we therefore keep within-group signal recovery distinct from selection utility.

4 Evaluation Design

The studies estimate different quantities and are not pooled into a single efficacy number. The prospective model-repair and nested-K studies measure aggregation opportunity while preserving evidence. The natural audit tests operational applicability and semantic validity, and the controlled studies measure held-out utility when matched aggregation can differ. A historical boundary track (Appendix F) instead changes the evidence/scoring interface on real agent rollouts; it is reported separately because it does not isolate aggregation.

For the constructed-task studies, hidden evidence is produced by reference execution rather than an LLM judge. The generator derives argument strategies from function signatures and declared

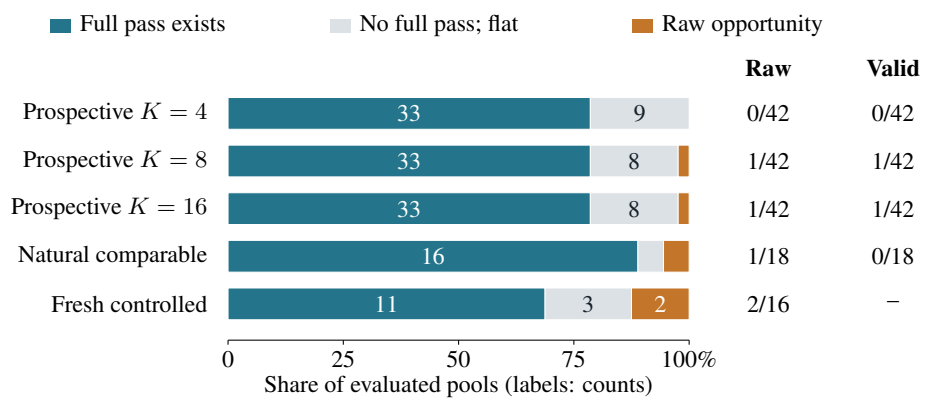

Figure 2: Observed opportunity anatomy. Colored segments partition each population into full-pass, no-full-pass flat, and raw-opportunity pools. Right columns separate raw from semantically validated prevalence; "–" means validation is not reported. The prospective rows use nested pools on the same 42 tasks, not independent samples. Populations are not pooled.

or inferred types, then uses Hypothesis-based property search to find concrete calls. Only distinct calls that execute successfully against the known-correct reference are retained. A is generated first; B uses a separate seed while excluding every concrete A call. Thus A and B are input-disjoint but are produced by the same reference-only generation mechanism. We do not claim structural independence of the behaviors they exercise.

Candidate information is never used to construct A or B. Reference and candidate executions use the same subprocess harness and output normalization. Exceptions, non-zero subprocess returns, timeouts, or unparseable outputs are execution failures. Pool-construction attrition is reference-side: tasks can fail because insufficient distinct valid A/B calls are found or because reference self-consistency fails before candidate outcomes are used.

For the strict held-out endpoint,

$$q_t(c) = \mathbf{1}\{b_t(c) = 1\}, \qquad V_t(s) = \frac{1}{|M_t(s)|} \sum_{c \in M_t(s)} q_t(c). \tag{11}$$

The task is the inferential unit; reported intervals use 10,000 paired task-bootstrap resamples. Uniform tie-breaking is an explicit mechanism baseline, not a claim of superiority over every secondary heuristic that could be layered onto a binary verifier.

Figure 2 shows the reconstructable opportunity counts; Appendix B retains the exact summary table and evaluation pipeline. The original controlled study is omitted because its retained authoritative summary establishes held-out deltas but not the A16 profiles needed to reconstruct O_t. Nonzero held-out deltas do not identify all opportunity pools.

5 Prospective Model-Repair Study: Measuring the Bottleneck

An outcome-exposure audit identified 82 eligible constructed repository defects, and all 82 were frozen rather than outcome-subsampled. The primary protocol fixed four single-shot GPT-5.6-Sol target-function repairs per scoreable task, generic reference-only A32/B64 construction, A16 as the selection endpoint, B64 as input-disjoint held-out evaluation, and the same tie rule as Eq. 4. Pool construction succeeded on 45 tasks; three additional tasks failed a repeated reference self-consistency gate using reference-side evidence only, leaving 42 primary tasks and 168 scored repairs. These are target-function repairs on constructed defects, not unrestricted multi-file agent patches or natural GitHub issues.

At the predeclared $K = 4$ endpoint, 33/42 pools contain an A16 full-pass candidate. The remaining 9 contain no full pass but are fractionally flat. Thus raw opportunity prevalence is exactly 0/42 on this evaluated set, the binary and fractional top sets are identical on every task, and Eq. 10 forces

Table 1: Candidate-pool-size stress test. The empirical opportunity bound is the maximum absolute task-macro utility difference possible for the matched binary-versus-fraction selectors on the evaluated set, not a population confidence bound.

K	Full pass	Flat	Opportunity	Emp. bound	ΔU	ΔV
4	33/42	9/42	0/42	0.0000	0.000000	0.000000
8	33/42	8/42	1/42	0.0238	0.001302	0.000000
16	33/42	8/42	1/42	0.0238	0.001395	0.000000

the empirical aggregation-only task-macro difference to zero. The observed B64 utility is 0.906 and strict B64 full-pass selection probability is 0.786 for both selectors. All 33 A16 full-pass selected sets also fully pass B64; none of the nine no-full-pass pools yields a B64-full-pass selection.

Frozen-protocol deviations. The protocol planned 328 generations across 82 tasks, but generation followed reference-pool construction in execution, so only the 45 pool-ready tasks were generated (180 attempts). A post-freeze, reference-only self-consistency gate then excluded three nondeterministic `backoff_iter` tasks. Retaining them gives 33 full-pass, 11 flat, and 1 opportunity among 45 tasks, with task-macro $\Delta U \approx 8.68 \times 10^{-5}$. Appendix E documents both deviations and the sensitivity.

5.1 Nested-K stress test

Because four candidates might artificially suppress opportunity, we froze a separate post-hoc stress-test protocol before generating additional repairs. The same 42 tasks, model path, frozen A32/B64 evidence, A16 endpoint, scorer, and tie rule were retained. Candidate pools are nested: $C_{t,4} \subset C_{t,8} \subset C_{t,16}$. The original four attempts were not regenerated; 12 additional attempts per task produced 504/504 new generations without generation or infrastructure failures.

Only one task enters the opportunity regime at $K = 8$, and it remains the sole opportunity at $K = 16$. The task is a `boltons chunk_ranges` variant. Its non-flat A16 profile survives a semantic-validity audit: the differing repair omits a real `input_size == 0` edge-case guard rather than differing only in representation order. At $K = 16$, the sole opportunity's task-level held-out difference is therefore 0.0586, while multiplication by the observed prevalence $1/42$ yields the overall 0.001395 difference.

The sweep also rules out exact duplicate generation as the main explanation for flatness. Eight of the nine original no-full-pass flat tasks already contain textually and AST-distinct repairs at $K = 4$, and those same eight remain flat through $K = 16$. This does not establish semantic equivalence outside the frozen evidence; it shows that the observed bottleneck is not explained merely by identical text or ASTs.

The prevalence estimates remain sample-specific. The same model, prompt, task population, and evidence generator are used throughout the stress test, and the paired task-bootstrap interval for the $K = 16$ minus $K = 4$ opportunity-prevalence change is $[0.000, 0.0714]$. We therefore do not interpret 0/42 or 1/42 as exact population rates.

6 Natural Real-Agent Study: Verifier Trustworthiness and Spurious Resolution

The natural study contains 52 repository-repair tasks with 288 recoverable candidate patches generated before the present hidden-probe analysis. Forty-one tasks come from a Qwen3-27B source pool and eleven from GPT-5.6-Sol. Under the frozen capability taxonomy, 23/52 tasks are probe-capable. One task is excluded from primary statistics because its final probe design was candidate-informed, leaving 22 primary-capable tasks. Fifteen have both candidate-level hidden-binary and repository-test binary fields.

On those 15 paired tasks, task-macro hidden-versus-repository binary agreement is

$$0.9667, \qquad 95\%\ \text{CI}\ [0.900, 1.000].$$

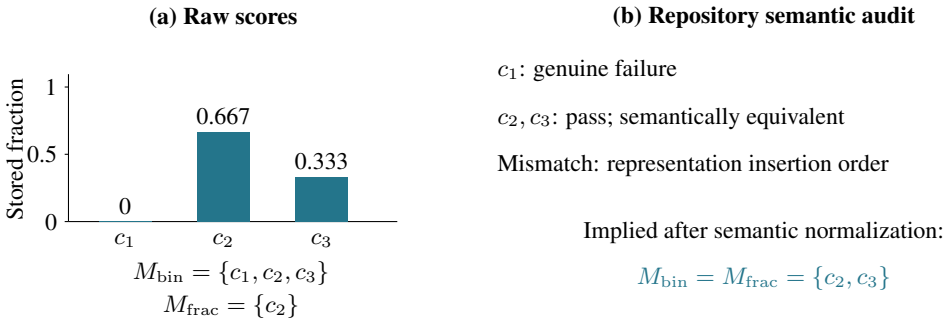

Figure 3: The Xarray raw opportunity fails semantic validation. Candidate labels are local to this illustration. Panel (a) shows the stored scores; panel (b) summarizes the reported `assert_identical` audit. The corrected top-set equality is its semantic implication, not a newly executed scoring result. This study has no standardized held-out B endpoint.

Across the 22 primary-capable tasks, 64 candidate records receive full credit from the repository baseline; two fail hidden reference behavior, both on `pylint-7114`, with hidden fraction 0.4. This establishes a concrete repository-pass/reference-disagreement case, not by itself a proof of semantic incorrectness and not a population error-rate estimate.

The aggregation anatomy is more revealing. Eighteen primary-capable tasks contain at least two candidates with both hidden binary and hidden fraction. Sixteen contain at least one full-pass candidate, forcing identical top sets. `astropy-14365` has no full-pass candidate but every scored candidate has hidden fraction 0.9318, so both selectors again tie the same pool. The remaining task, `pydata__xarray-3364`, is the sole raw opportunity: hidden binary scores are $0, 0, 0$ while stored hidden fractions are $0, 0.667, 0.333$.

That apparent extra resolution is not semantically trustworthy (Figure 3). Inspection of the retained task artifact shows that the 0.667- and 0.333-scoring candidates both pass the relevant repository FAIL_TO_PASS tests and satisfy the repository's own `assert_identical` equivalence check. Their fractional mismatch comes from coordinate/dictionary insertion-order differences in captured return representations. Only the zero-scoring candidate is a genuine failure. Under repository-native equivalence, the two passing candidates would both receive full semantic credit, making the strict-binary and fractional top sets identical; the raw opportunity therefore disappears after semantic normalization. Thus this population has raw opportunity 1/18 but validated opportunity 0/18.

The Xarray case is important because it prevents a common shortcut: non-binary numerical spread is not itself evidence of useful information. It also clarifies the order of the audit. Score semantics must be checked before an apparent opportunity is interpreted as behavioral resolution. The natural pools lack a standardized input-disjoint B endpoint, so this study does not estimate natural-task selection gain.

Operational reach is selective. Usable reference evidence appears across several repositories in this 52-task audit, but a separate frozen harvesting stress test is substantially less portable: 16/87 attempted tasks certify and all 16 are Django, while the other ten repositories contribute 0/51 certifications. We treat that result as a limitation of the current generic automatic harvesting implementation, not as evidence that the decision-relevance decomposition itself is Django-specific.

Targeted natural-rollout hidden-A/B follow-up. A post-hoc follow-up selected two replayable SymPy rollout groups by prior call-reconstruction fidelity, not candidate outcomes; no protocol was frozen. On `sympy__sympy-24102` (8/8 usable), one candidate fully passes 96 recorded A calls while the other seven score 95/96, so strict and fractional aggregation select the same full-pass candidate. On `sympy__sympy-15346` (7/8 usable; one rollout submitted no patch), all seven score 238/239 on A, leaving an all-fail but fractionally flat pool. Both have $O_t = 0$, and every usable candidate scores 1.0 on disjoint B. This is boundary validation, not a prevalence estimate; no existing denominator changes. Appendix J and the artifact give the full provenance.

7 CONTROLLED STUDIES: CONDITIONAL REWARD UTILITY ONCE OPPORTUNITY EXISTS

The prevalence studies above answer whether the matched reward formulations have room to induce different selections. Controlled mutation pools answer the complementary question: when the frozen evidence and candidate pool do expose an ordering that binary reward aggregation may collapse, does the fractional ordering predict held-out behavior?

7.1 ORIGINAL CONTROLLED STUDY

The original frozen track targeted within-task A/B concordance rather than the selector estimand used here. We therefore treat the selector result below as a post-hoc reanalysis of frozen, reward-blind mutation pools. The manifest contains 25 constructed single-function defects in real repositories; 15 have sufficient hidden A/B evidence and at least two selectable deterministic AST-level mutation candidates, yielding 70 candidates. Probe B contains 64 separately generated calls whose concrete inputs are disjoint from A, but A and B use the same reference-only generation mechanism.

At A16, fractional aggregation exceeds hidden binary by 0.106875 in task-macro held-out utility (95% CI $[0.014583, 0.218333]$; 4W/11T/0L). The result is concentrated: exactly four tasks carry nonzero deltas, and removing any one of those four moves the bootstrap lower bound to zero. We therefore treat this as fragile historical evidence that conditional utility can occur, not as a stable population estimate; a recovered per-task delta table lets the task-macro mean and win/tie/loss be recomputed exactly, and the CI closely but not exactly, while the underlying per-candidate selector records remain unreconstructed (Appendix C).

The exact original A16 raw-opportunity count is not reconstructed from the retained authoritative candidate-level profiles in this manuscript. We do not equate the four nonzero held-out deltas with four opportunities: opportunity is defined from A-side scores, whereas nonzero Δ_t depends on B-side outcomes. The full A4/A8/A16/A32 budget curve, coverage-based repository comparison, common-subset selector table, leave-one-task-out analysis, and provenance sensitivity are retained in Appendix C.

7.2 FRESH, TASK-DISJOINT REPLICATION

To test the concentrated original result without reusing exposed tasks, we ran a prospectively frozen replication. Its manifest contains 24 mechanically mined tasks with zero overlap with all 247 previously exposed constructed tasks and spans 18 source repositories. Reference-only gating leaves 16/24 primary tasks with complete A32/B64 pools; eight fail predeclared reference-side attrition.

The replication yields

$$\overline{\Delta} = 0.042643, \qquad 95\%\ \text{CI}\ [0.000000, 0.105469], \qquad 2W/14T/0L.$$

Thus it does not reproduce a CI-excluding-zero population-wide average effect. Its anatomy is nevertheless directly informative for the decomposition: 11/16 pools contain a full-pass candidate, 3/16 have no full pass but are fractionally flat, and 2/16 are raw opportunities. Those two raw opportunities are exactly the two tasks with nonzero held-out deltas, `mbstrdecoder` (+0.3229) and `geomet` (+0.3594). Hence

$$0.042643 = \frac{2}{16} \times 0.341144$$

up to rounding, where 0.341144 is the observed mean Δ_t conditional on the two *raw* opportunity pools.

We label that conditional quantity raw-opportunity utility. Frozen secondary additions—semantic validation, a repository-coverage tie-break, and a developer-test endpoint—are absent from the released run, so no claims are made from them; leave-one-repository-out means range from 0.0215 to 0.0525. The full-sample interval includes zero, and the original and fresh estimates remain separate; no pooled estimate replaces either study.

8 DISCUSSION

Opportunity prevalence limits aggregation gain. Prospective opportunity is absent at $K = 4$ and scarce at $K = 8, 16$; the natural study's sole raw opportunity is a representation artifact. In the fresh replication, two raw opportunities carry all nonzero deltas, but the overall interval includes zero; the decomposition explains concentration, not significance.

A practical three-stage audit. Before engineering denser executable supervision: (1) validate score semantics; (2) measure whether aggregation can change the decision under frozen evidence and candidates; and (3) test held-out utility on valid opportunities. Prospective $K = 4$ stops at stage 2; Xarray fails stage 1; the fresh `mbstrdecoder/` `geomet` raw opportunities have positive held-out differences but remain semantically unvalidated because that predeclared audit was not completed. Applied to all 18 comparable real-agent pools in Section 6 (Qwen3-27B/GPT-5.6-Sol patches), 17 have no raw selection opportunity (16 contain a full-pass candidate and 1 is no-full-pass/flat). The sole raw opportunity, Xarray, fails semantic validation, leaving 0/18 validated opportunities. Thus no pool here both passes semantic validation and has $O_t = 1$; a held-out check on a validated real-agent opportunity remains untested.

Evidence, aggregation, and optimization are distinct channels. Top-set opportunity is stricter than within-group RL signal: below-top-set rewards may differ even when selectors agree. Historical boundary data recover variation in 19/22 terminal-binary-flat groups versus 8/22 for repository-test fraction, and withhold full credit from 21/22 qualifying exploits. The optimizer run is only an interface check (Appendix F).

9 LIMITATIONS

The bottleneck is evidence- and candidate-relative: a flat A16 profile does not imply behavioral equivalence beyond those probes. The prospective 0/42 result is conditional on 42 reference-valid tasks from 82 frozen tasks, one model/prompt, one evidence generator, and $K = 4$; the nested-K extension is post-hoc.

A and B are input-disjoint but share a reference-only generator, so B tests unseen calls rather than independently designed behaviors. The primary 52-task natural audit has no common B endpoint; the targeted two-task follow-up supplies disjoint B only for those groups. Uniform tie-breaking isolates aggregation but not production tie-breakers, so no dominance over repository fraction, critics, or other heuristics is claimed.

Non-flat scores need not be semantically valid. Controlled single-function defects and deterministic mutations are narrower than unrestricted agents; the fresh study has only 16 primary tasks and a CI including zero, while the original 15-task effect is concentrated in four tasks. Harvesting also certifies only 16/87 attempted tasks, all Django (0/51 across the other ten repositories).

Historical provenance further limits interpretation: omitting unresolved `ingest_rsa` moves the original controlled CI lower bound to zero; the optimizer arms use separately trained checkpoints; the verifier audit lacks complete inspected patches; and the 21/22 adversarial result covers a frozen, non-adaptive attack population (Appendices F–G).

10 CONCLUSION

Binary reward can fail because executable evidence is too weak or because its aggregation discards useful distinctions. For strict all-pass and fractional scoring on fixed traces, we characterize when aggregation can change the specified selector, factorize empirical selection gain into opportunity prevalence and conditional utility, and derive an observable aggregation-only bound. Prospective and natural studies expose sparse or spurious aggregation opportunities; controlled studies show useful conditional ordering can exist but is concentrated; and historical real-agent data show richer evidence can restore within-group signal under binary collapse. For repository-level agent reinforcement learning, the implication is diagnostic, not a blanket prescription for dense reward: validate semantics, measure reward starvation and selection opportunity, then target aggregation, evidence, or candidate generation according to the bottleneck. End-to-end policy improvement remains open.

AI USE STATEMENT

AI Use Statement. Generative AI tools were used to assist with drafting, editing, and polishing the manuscript and with final claim-to-evidence, methodology, and reproducibility auditing. Language models evaluated in this work were also used to generate candidate repairs or trajectories as part of the experimental methodology. Generative AI was not used to formulate or prove the mathematical claims reported in this work. AI-assisted feedback was checked against the manuscript, code, and retained experimental records before inclusion. The author takes responsibility for the final content and artifacts.

ETHICS STATEMENT

The experiments use public open-source repositories and isolated local execution environments. The exploit-seeking and adversarial visible-verifier audits were restricted to local or network-isolated Docker environments and did not target deployed services, private systems, or user data. Reported verification disagreements, representation artifacts, and verifier exploits are studied to improve the reliability of training and evaluation infrastructure.

REPRODUCIBILITY STATEMENT

The accompanying anonymous artifact contains frozen raw records for the prospective, fresh-replication, natural, and harvesting studies, plus historical records and audit notes for the original controlled track. Appendix A gives the score properties and decomposition; Appendix B gives summary accounting and selection pseudocode; Appendices C–E describe the controlled, fresh-replication, and prospective analyses, sensitivity checks, and protocol fingerprints. The released original-controlled files let the task-macro mean and win/tie/loss of its A16 selection reanalysis be recomputed from a recovered historical task-level delta summary, but the underlying per-candidate A4/A8/A16/A32 selector records and the original bootstrap implementation remain unavailable. Appendices H– K document natural-agent capability, semantic auditing, and harvesting coverage. Appendix G documents the ordinary-rollout and exploit-seeking verifier audits, and Appendix F documents the adversarial visible-verifier campaign and optimizer-interface check. Appendix J and `data/natural_ab_followup/` document the post-hoc two-task natural A/B follow-up. Historical tracks with incomplete execution provenance are explicitly identified; original controlled opportunity counts are not inferred from nonzero held-out deltas.

REFERENCES

Koen Claessen and John Hughes. QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs. In *Proceedings of the 5th ACM SIGPLAN International Conference on Functional Programming*, pp. 268–279, 2000.

Silong Dai, Changzhi Sun, Haolun Wu, Huanran Zheng, Tao Ji, Junchi Yan, Yuanbin Wu, Dell Zhang, Xiaoling Wang, and Xuelong Li. Group Verification-based Policy Optimization for Interactive Coding Agents. In *International Conference on Learning Representations*, 2026.

Naman Jain, Jaskirat Singh, Manish Shetty, Tianjun Zhang, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural Environment Generation and Hybrid Verifiers for Scaling Open-Weights SWE Agents. In *Conference on Language Modeling*, 2025.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? In *International Conference on Learning Representations*, 2024.

Minsu Kim and Se-Young Yun. Process-Verified Reinforcement Learning for Theorem Proving via Lean. In *International Conference on Learning Representations*, 2026.

Jiayi Kuang, Yinghui Li, Xin Zhang, Yangning Li, Di Yin, Xing Sun, Ying Shen, and Philip S. Yu. Process-Level Trajectory Evaluation for Environment Configuration in Software Engineering Agents. In *International Conference on Learning Representations*, 2026.

Mayank Kumar, Jiaqi Xue, Mengxin Zheng, and Qian Lou. FHE-Coder: Benchmarking Secure Agentic Code Generation for Fully Homomorphic Encryption. In *International Conference on Learning Representations*, 2026.

Hyunji Lee, Minseon Kim, Chinmay Singh, Matheus Pereira, Atharv Sonwane, Isadora White, Elias Stengel-Eskin, Mohit Bansal, Zhengxiang Shi, Alessandro Sordoni, Marc-Alexandre Côté, Eric Yuan, and Lucas Caccia. Gistify: Codebase-Level Understanding via Runtime Execution. In *International Conference on Learning Representations*, 2026.

Xin-Ye Li, Ren-Biao Liu, Yun-Ji Zhang, Hui Sun, Zheng Xie, and Ming Li. Exploring Pass-Rate Reward in Reinforcement Learning for Code Generation. *arXiv preprint arXiv:2605.02944*, 2026.

William M. McKeeman. Differential Testing for Software. *Digital Technical Journal*, 10(1):100–107, 1998.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training Software Engineering Agents and Verifiers with SWE-Gym. In *International Conference on Machine Learning*, 2025.

Monoshi Kumar Roy, Simin Chen, Benjamin Steenhoek, Jinjun Peng, Gail Kaiser, Baishakhi Ray, and Wei Le. CodeSense: A Real-World Benchmark and Dataset for Code Semantic Reasoning. In *International Conference on Learning Representations*, 2026.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding Progress: Scaling Automated Process Verifiers for LLM Reasoning. In *International Conference on Learning Representations*, 2025.

Leitian Tao, Ilia Kulikov, Swarnadeep Saha, Tianlu Wang, Jing Xu, Sharon Li, Jason E. Weston, and Ping Yu. Hybrid Reinforcement: When Reward Is Sparse, Better to Be Dense. In *International Conference on Learning Representations*, 2026.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. SWE-RL: Advancing LLM Reasoning via Reinforcement Learning on Open Software Evolution. In *Advances in Neural Information Processing Systems*, 2025.

Longwen Wang, Yirui Liu, Xuan'er Wu, Xiaohui Hu, Yuankai Fan, Kaidong Yu, Qizhen Weng, Wei Xi, and Xuelong Li. Beyond Binary: Turning Partial Success into Dense Verifiable Rewards for Reinforcement Learning in Code Generation. *arXiv preprint arXiv:2601.03525*, 2026.

Xinpeng Wang, Nitish Joshi, Barbara Plank, Rico Angell, and He He. Is It Thinking or Cheating? Detecting Implicit Reward Hacking by Measuring Reasoning Effort. In *International Conference on Learning Representations*, 2026.

Yuchen Yan, Jin Jiang, Zhenbang Ren, Yijun Li, Xudong Cai, Yang Liu, Xin Xu, Mengdi Zhang, Jian Shao, Yongliang Shen, Jun Xiao, and Yueting Zhuang. VerifyBench: Benchmarking Reference-based Reward Systems for Large Language Models. In *International Conference on Learning Representations*, 2026.

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*, 2025 (Datasets and Benchmarks Track).

Yiwen Zhang, Xiaodong Yan, Zhenyu Huang, Deng Zhao, Liang Jiang, Qing Cui, Zujie Wen, Zhiqiang Zhang, and Jun Zhou. Robust Code RL via Faulty-Code-Driven Test Case Synthesis and Dense Reward Shaping. *arXiv preprint arXiv:2608.24135*, 2026.

Zhe Zhang, Runlin Liu, Aishan Liu, Xingyu Liu, Xiang Gao, and Hailong Sun. Code2Bench: Scaling Source and Rigor for Dynamic Benchmark Construction. In *International Conference on Learning Representations*, 2026.

Qixing Zhou, Jiacheng Zhang, Haiyang Wang, Rui Hao, Jiahe Wang, Minghao Han, Yuxue Yang, Shuzhe Wu, Feiyang Pan, Lue Fan, Dandan Tu, and Zhaoxiang Zhang. FeatureBench: Benchmarking Agentic Coding for Complex Feature Development. In *International Conference on Learning Representations*, 2026.

A Score Properties and Opportunity Decomposition

A.1 Top-set characterization

For complete equal-length hidden traces,

$$s_{\mathrm{bin}}(c) = 1 \quad \Longleftrightarrow \quad s_{\mathrm{frac}}(c) = 1.$$

Therefore, if at least one candidate satisfies every hidden check, the binary and fractional top-scoring sets are identical. If no candidate satisfies every hidden check, binary assigns zero to the entire pool and fraction changes the top set exactly when fractional scores are nonconstant.

No dominance result follows merely from having an opportunity. For example, if candidate X has A-fraction 0.75 and held-out utility 0 while candidate Y has A-fraction 0.25 and held-out utility 1, binary tie selection has expected utility 0.5 whereas fraction selects X and obtains 0. This is why opportunity and conditional held-out utility are separate quantities.

A.2 Finite-sample factorization

Let $\Delta_t = U_t(s_{\mathrm{frac}}) - U_t(s_{\mathrm{bin}})$. Since $O_t = 0$ implies identical top sets, $O_t = 0$ implies $\Delta_t = 0$. Hence, for any fixed set of T evaluated tasks,

$$\overline{\Delta} = \frac{1}{T}\sum_{t} \Delta_t = \frac{1}{T}\sum_{t:O_t=1} \Delta_t = \widehat{\pi}_O \overline{\Delta}_O.$$

Because $U_t \in [0,1]$, $|\Delta_t| \le 1$ and $|\overline{\Delta}| \le \widehat{\pi}_O$.

This is a deterministic identity for the evaluated sample under the specified selectors, not a confidence statement. If tasks are random draws from a fixed distribution under the same setup, the population analogue is $\mathbb{E}[\Delta] = \Pr(O=1)\mathbb{E}[\Delta \mid O=1]$, with $|\mathbb{E}[\Delta]| \le \Pr(O=1)$. The observed opportunity frequency is an estimate of, not an equality to, that population probability.

A.3 Raw versus semantically valid opportunity

Raw O_t is defined from implemented A-side scores and therefore exactly controls whether the implemented top sets can differ. Semantic validation is an additional scientific filter on whether that numerical distinction corresponds to meaningful behavior. We do not replace raw O_t by a semantic label inside the algebra above; instead we report raw and validated opportunity separately where validation is available.

B Visual Accounting and Selection Procedure

Table 2 retains the exact summary underlying Figure 2; the additional $K = 8$ chart row comes from Table 1. Figure 4 distinguishes changing an evidence source from changing its aggregation. No results are pooled across these populations.

Table 2: Opportunity anatomy for the populations where raw opportunity is directly reconstructable from retained A-side scores. "Valid" reports semantic validation only where such an audit was performed; a dash means we do not infer semantic validity from the raw score profile alone.

Population	Full pass	Flat	Raw opp.	Valid opp.	Selection result
Prospective $K = 4$	33/42	9/42	0/42	0/42	$\Delta U = 0$
Prospective $K = 16$	33/42	8/42	1/42	1/42	$\Delta U = 0.001395$; raw/valid conditional $\Delta U = 0.0586$
Natural comparable	16/18	1/18	1/18	0/18	No standardized held-out B
Fresh controlled	11/16	3/16	2/16	–	$\Delta U = 0.042643$; raw-opportunity conditional $\Delta U = 0.341$

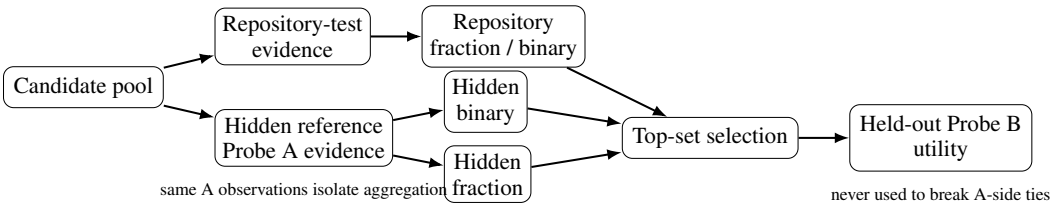

Figure 4: Experimental decomposition. Repository versus hidden checks changes the evidence source; hidden binary versus hidden fraction changes only aggregation because both consume the same Probe A observations. Probe B is input-disjoint from A and evaluates the resulting selection.

B.1 Selection and expected tie handling

The following Python-style pseudocode implements Eqs. 5, 4, and 6 for one task with nonempty candidate pools and complete, equal-length binary A traces. Integer match counts avoid deciding full-pass status by rounded decimal scores. The input `b` contains held-out B utilities and is read only after the top sets have been fixed. This is explanatory pseudocode, not an additional empirical implementation or a substitute for the released experimental artifact.

```
# a[j][i]: frozen A agreement, candidate j on call i
# b[j]: held-out B utility; never used for selection
m = [sum(trace) for trace in a]
n = len(a[0])
all_candidates = set(range(len(a)))
full = {j for j in all_candidates if m[j] == n}
M_bin = full if full else all_candidates
M_frac = {j for j in all_candidates if m[j] == max(m)}
O = int(max(m) < n and max(m) > min(m))
U_bin = sum(b[j] for j in M_bin) / len(M_bin)
U_frac = sum(b[j] for j in M_frac) / len(M_frac)
delta = U_frac - U_bin
```

Semantic validation is a separate audit of the equivalence relation behind `a`; it is not inferred from `O` or from a positive `delta`. Figure 3 illustrates why that separation matters. The pseudocode describes the matched hidden-evidence comparison; repository-test selectors use their own evidence source.

C Controlled Study Accounting

The controlled manifest contains 25 tasks. The frozen track's original primary statistic was within-task A/B concordance; the selector comparison reported here is a post-hoc reanalysis. Fifteen tasks support that hidden binary-versus-fraction reanalysis and contain 70 selectable mutation candidates. The stronger coverage-based repository-test baseline is available on 11 of these tasks. Task-level bootstrap intervals use 10,000 resamples with fixed seed 314159.

Coverage-based repository-test baseline. For each task, `pytest` is run with dynamic per-test coverage instrumentation (`--cov-context=test`) against the reference candidate's own checkout, scoped to test files under the repository's standard test-file naming conventions. The set of test node IDs whose executed lines fall inside the target function's real AST body range is the coverage-confirmed repository-test surface, determined by dynamic execution rather than lexical name matching. Those node IDs must first pass a reference gate (running against the reference candidate, exiting with only pass/fail, not a collection error) before being applied to any scored candidate. Repository binary is 1 iff every coverage-confirmed node ID passes for that candidate, 0 otherwise; repository fraction is the fraction of coverage-confirmed node IDs that pass, matching the hidden-fraction convention. If coverage instrumentation cannot be collected for a task (sandboxed-checkout limitation, no line hits inside the function from any test, or timeout), that task falls back to the earlier lexical method and is flagged accordingly; such tasks are excluded from the 11-task coverage-based comparison reported here. The selector-level results are reproduced here to make the distinction from

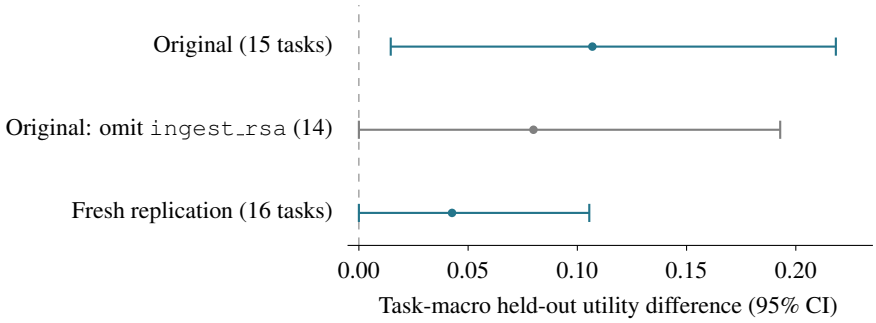

Figure 5: Reported controlled-study utility differences and 95% paired task-bootstrap intervals. The original result is concentrated and its `ingest_rsa`-omission sensitivity includes zero. The task-disjoint fresh replication also includes zero. These are separate analyses, not a pooled effect. The sensitivity addresses the provenance issue documented below.

the natural-agent study explicit. The released `data/controlled_original/` files retain the earlier concordance-track records, not the later budgeted A4/A8/A16/A32 per-candidate selector table; a separately recovered per-task delta table at budget 16 (sourced from a since-lost intermediate analysis file, preserved only in project provenance notes) lets the archive recompute the task-macro mean (0.106875) and win/tie/loss (4/11/0) exactly, and a paired task-bootstrap over the same seed and resample count reproduces the CI's lower bound exactly and its upper bound closely but not identically (the original bootstrap code itself is not preserved). The underlying per-candidate, per-call execution traces do not survive, nor does the original bootstrap implementation, so this remains a reported historical result at the sub-task level even though its task-macro aggregate is now recomputed from a recovered historical task-level summary, not independently reconstructed from raw execution the way the fresh replication is.

Full leave-one-task-out (all 15 tasks). Only 4 of the 15 primary tasks have a nonzero budget-16 delta (`boltons` $\times 2$ at 0.219 each, `ingest_rsa` at 0.484, `jsonpickle` at 0.681); the remaining 11 are exact ties. Removing any of the 11 zero-delta tasks leaves the mean and CI effectively unchanged (mean 0.1145, CI $[0.0156, 0.2324]$ or $[0.0156, 0.2308]$ depending on which repository baseline the removed task carried). Removing any of the 4 nonzero-delta tasks lowers the mean and, in every one of those 4 cases, drops the bootstrap CI's lower bound to exactly 0.0: dropping `jsonpickle` gives mean 0.066, CI $[0.000, 0.151]$; dropping `ingest_rsa` gives mean 0.080, CI $[0.000, 0.193]$; dropping either `boltons` instance gives mean 0.099, CI $[0.000, 0.218]$. The full 15-task LOO point-estimate range is $[0.0658, 0.1145]$. We disclose this plainly: the 15-task primary result's CI excludes zero, but that property is not robust to removing any one of the 4 tasks that carry the entire observed effect, and it is not merely a `jsonpickle`-specific artifact.

Audit-motivated sensitivity. Because `ingest_rsa`'s per-task delta changed between an earlier, superseded internal run and the current authoritative run (see below), we additionally recompute the primary statistic with `ingest_rsa` omitted, using the identical selection rule and bootstrap seed as the reported 15-task selector reanalysis. Result: 14-task mean 0.079911, CI $[0.000, 0.192857]$, W/T/L 3/11/0. This is reported because of unresolved historical provenance for that one task, not because any task was excluded from the retained analysis population; the 15-task reported result (Table 3, reproduced below) is unchanged.

Historical-number provenance. An earlier, superseded internal analysis produced slightly different budget-8 and budget-16 means. A per-task comparison localizes the difference to `ingest_rsa`; the other 14 primary tasks are unchanged at the per-task level. The retained historical artifacts do not establish why that task's B-side scores changed, so we do not attribute a cause. The current retained selector analysis is the source for the reported numbers. Because the discrepancy cannot be independently reconstructed from retained raw B executions, we also report the frozen audit-motivated sensitivity that omits `ingest_rsa`: 14-task mean 0.079911, 95% CI $[0.000, 0.192857]$,

W/T/L 3/11/0. This provenance issue is unrelated to the separate lexical-versus-coverage repository-baseline change.

Table 3: Controlled-study paired differences.

Budget	Fraction − hidden binary	Fraction − repo fraction
4	0.046 $[0.000, 0.110]$	0.103 $[-0.033, 0.262]$
8	0.078 $[0.009, 0.165]$	0.105 $[-0.028, 0.264]$
16	0.107 $[0.015, 0.218]$	0.145 $[0.019, 0.290]$
32	0.107 $[0.015, 0.218]$	0.145 $[0.019, 0.290]$

Table 4: Task-macro held-out Probe B utility for every selector on the identical 11-task common subset at A16. This table is descriptive across selectors because some comparisons change both evidence source and aggregation.

	Random	Repo bin.	Repo frac.	Hidden bin.	Hidden frac.
Utility	0.522	0.754	0.789	0.832	**0.933**

Exploratory cross-evidence check. As a secondary sanity check using only existing retained records (no new execution), we select candidates via hidden-A binary and hidden-A fraction as usual but evaluate the selection using repository-coverage fraction instead of Probe B, on the same 11-task subset. This tests whether the fraction-over-binary advantage depends on A/B sharing generation machinery. Result: mean 0.048, CI $[0.000, 0.143]$, 1W/10T/0L — positive but the interval touches zero, so this is not treated as independent confirmation, only as a weak consistency check.

D Fresh Controlled Replication Accounting

The replication task manifest was frozen before candidate mutation generation or scoring. It contains 24 mechanically mined tasks with zero overlap with the 247 previously exposed constructed tasks and spans 18 source repositories (maximum four tasks from any one repository). Reference-only gating leaves 16/24 primary tasks with complete A32/B64 pools; eight fail because the frozen search cannot construct sufficient distinct valid inputs or encounters a reference-side search error. No candidate outcome is used in this attrition.

The frozen protocol uses new evidence seeds (A=135791, B=246802), A16 as the primary selection endpoint, B64 as input-disjoint held-out evaluation, the same reward-blind deterministic mutation machinery as the original controlled study, uniform expected tie-breaking, and 10,000 task-bootstrap resamples with fixed seed `20260925b`. Only mutation candidates are selectable; the reference and defective slots retained in the raw file are harness controls and are excluded from selector pools. The frozen task-manifest SHA256 is `f5eb08fdd9ed272c0b86c826d96954b2a889dc8ee52612d10e8dc50eba47a704`; the replication-protocol SHA256 is `d28d5697ce4f43d6745ce4cf1bd7cb4f0ae039e3e0643770ca7edb08520758c8`.

The primary replication estimate is mean $\Delta U = 0.042643$, 95% task-bootstrap CI $[0.000000, 0.105469]$, W/T/L $= 2/14/0$. Only `mbstrdecoder` ($+0.3229$) and `geomet` ($+0.3594$) have nonzero task deltas. Leave-one-task-out point estimates range over $[0.0215, 0.0455]$; leave-one-repository-out means reconstructed from the released per-task deltas range over $[0.0215, 0.0525]$. The full-sample interval already includes zero. The protocol also predeclared semantic validation of each raw opportunity, a repository-coverage tie-break, and a developer-test endpoint; those outputs are absent from the released result files and are not claimed here. The protocol sentence calling candidate construction identical to the prospective study is a wording error: the prospective study uses GPT-5.6-Sol repairs, whereas this controlled study uses deterministic mutations.

At A16, 11/16 pools contain a full-pass candidate, 3/16 have no full pass and a flat fractional profile, and 2/16 satisfy the opportunity condition. The two opportunity pools are exactly the two pools with nonzero fractional-minus-binary held-out deltas. We therefore interpret this replication as in-

dependent evidence that held-out gains are concentrated on sparse raw-opportunity pools, not as a replication of the original CI-excluding-zero average effect.

New-corpus stopping provenance. The frozen new-corpus mining protocol specifies a stopping rule of 30 reference-eligible fresh primary tasks or 500 attempted fresh repositories, whichever comes first, and an attempts ledger for that process. The ledger was not retained. The released artifact therefore verifies the resulting 24-task manifest, its 18-repository composition, and zero overlap with the 247-task exclusion set, but it cannot independently verify which stopping condition ended mining. We treat that stopping-condition detail as unverifiable rather than artifact-confirmed.

E PROSPECTIVE MODEL-REPAIR ACCOUNTING

The prospective protocol was frozen before official candidate generation; its SHA256 digest is recorded in the released anonymous artifact. All 82 tasks that passed the outcome-exposure audit were frozen; no outcome-based subsampling was performed. Generic reference-only A32/B64 construction succeeded on 45 tasks. Thirty-seven tasks failed the predeclared pool-generation gate (35 insufficient A, one insufficient B, one A-search error), and three additional tasks failed a repeated reference self-consistency check, leaving 42 primary tasks. The protocol fixed four single-shot GPT-5.6-Sol target-function repairs per scoreable task, A16 as the primary selection budget, B64 as held-out evaluation, and the tie rule in Eq. 4. The primary population therefore contains 168 scored candidate repairs.

A final recomputation from the retained attempt-level A16 and B64 values applies Eq. 4 literally. Thirty-three tasks contain at least one A16 full-pass candidate; on each of them binary and fractional top sets are identical, and every selected top-set candidate also fully passes B64. The remaining nine tasks contain no full-pass candidate, but all four candidates have identical A16 fractions, so both selectors again tie the same pool. On each of those nine tasks, all four attempts also have identical B64 agreement; their task-macro B64 utility is 0.562500 and none fully passes B64. Hence binary and fractional top sets are identical on all 42 tasks, task-macro B64 selection utility is 0.906250 for both selectors, and strict B64 full-pass selection probability is 0.785714 for both selectors.

The canonical score table used for this recomputation is released with one row per task and attempt. The table itself contains the primary A16/B64 score fields rather than full repair text. For the subsequent nested-K stress test, the preserved raw prospective generation file was reused without modifying attempts 0–3 and additional attempts were appended. This raw record makes it possible to audit textual and AST diversity of the original $K = 4$ pools directly.

Exploratory nested-K stress test. A separate post-hoc stress-test protocol was frozen before any new generation; its SHA256 digest is `8d7254c5045041b7940476096ae8a1d4eb752a6eacff5827f67989ee71403bc7`. The experiment retains the same 42 primary tasks, model/prompt path, frozen A32/B64 evidence, A16 selection endpoint, scorer, and tie rule. Candidate pools are nested by attempt order. Twelve additional attempts per task yield 504/504 successful new generations with no API or infrastructure failures. The original four attempts are byte-identical to the pre-sweep raw file, and the reference-pool file is unchanged.

Independent K-sweep analysis exactly reproduces the primary $K = 4$ result: 33 full-pass, 9 flat, 0 opportunity, 0 different top sets, $U = 0.906250$ for both selectors, and $V = 0.785714$ for both. At $K = 8$, the anatomy becomes 33 full-pass, 8 flat, and 1 opportunity; at $K = 16$ it remains 33/8/1. The same single task has different binary/fractional top sets at both larger pool sizes. Task-macro fractional-minus-binary held-out utility is 0.001302 at $K = 8$ and 0.001395 at $K = 16$; strict B64 full-pass difference is 0 at both. The $K = 16$ minus $K = 4$ opportunity-prevalence difference is $1/42 = 0.0238$, with a paired task-bootstrap 95% CI $[0.000, 0.0714]$ (10,000 resamples, seed 20260925).

The sole opportunity is `boltons__boltons-chunk_ranges-arg_order_kwarg-3-9109`. It first appears at attempt 4 and remains the sole opportunity through $K = 16$. The protocol-mandated semantic-validity audit finds a genuine algorithmic difference: the target returns plain integer tuples, and the differing repair omits an `input_size == 0` edge-case guard. Thus this opportunity is not attributable to representation ordering. Finally, eight of the nine original no-full-pass flat tasks already contain textually and AST-distinct repairs at $K = 4$, and those eight remain

flat through $K = 16$. Exact candidate duplication therefore does not explain most of the flatness, although semantic equivalence outside the frozen evidence is not established.

Superseded prospective result. An earlier generated summary reported a prospective mean "fraction-minus-binary" value of 0.111607 and 8W/34T/0L. Auditing the retained fields shows that this quantity was the task-macro mean of the A16 selector-score difference (`u_frac_A16 − u_bin_A16`), not the held-out selection utility defined by Eq. 4. In the affected tasks, binary assigned zero to all four candidates and therefore tied the full pool; treating that tie as zero utility contradicted the manuscript's tie convention. The superseded statistic is not used anywhere in the paper. The anonymous artifact retains the canonical attempt-level score table and corrected top-set recomputation needed to reproduce the 0/42/0 result.

Generation-coverage deviation from the frozen stopping rule. The frozen protocol's stopping rule reads: "Run all 328 planned generations regardless of interim results." Model-repair generation was in fact executed only on the 45 tasks whose reference-only A32/B64 pool had already succeeded (180 official attempts, not 328), because scoring a repair requires that task's pool to exist first; the remaining 37 tasks failed the predeclared pool-generation gate (35 insufficient A, one insufficient B, one A-search error) before any repair was generated, using reference-side evidence only. No task's inclusion depended on a model-repair outcome. We disclose this as a real deviation from the literal frozen stopping rule rather than silently reconciling the two, and we do not amend the hashed protocol document itself.

Self-consistency exclusion: predeclaration status and sensitivity. The three-task self-consistency exclusion described in Section 5 was a post-freeze reference-validity safeguard, not a rule stated in the frozen protocol; it used only reference-side repeated-execution evidence (never a model outcome) to identify tasks whose reference implementation disagrees with itself. All three excluded tasks are `boltons__boltons-backoff_iter` variants: the same A/B calls produce reference outputs that differ across repeated executions, and every such disagreement occurs on calls with `jitter=True` – a parameter `boltons`' own implementation documents as introducing `random.random()`-based randomization, making those three tasks unsuitable for an exact deterministic reference comparator regardless of any model outcome. Retaining all three gives a sensitivity anatomy over 45 tasks of 33 full-pass, 11 flat, and 1 opportunity (the retained `cur-2012` variant, hidden fractions $0.5625, 0.5625, 0.5, 0.5625$, task-level $\Delta_t = 0.0039$), with task-macro $\Delta U \approx 0.0000868$ over all 45 tasks – unchanged in substance from the primary 0/42 result. We report this sensitivity because the exclusion itself was not predeclared, even though its scientific rationale (reference nondeterminism) is independently verifiable from the retained records.

F Historical Agent-Rollout and Optimizer Boundary Evidence

This section retains earlier studies because they illuminate channels that the matched top-set analysis intentionally holds fixed. They are not pooled with the prospective or controlled estimates above and do not establish end-to-end policy improvement.

Real-agent signal recovery under visible binary collapse. An earlier frozen rollout study sampled groups of eight real agent repairs on constructed single-function defects in real repositories. The retained primary analysis contains 22 groups (176 scored rollouts) spanning 13 tasks for which terminal binary reward is identically zero. The earlier reference-based execution score combines matching-prefix and matching-fraction information and therefore differs from the pure fractional score used in the matched main-paper analysis. It has nonzero within-group variation in 19/22 groups; repository-test fraction has nonzero variation in 8/22, and those eight are a strict subset of the 19. This comparison changes both evidence and scoring, so it does not estimate the aggregation-only quantity Δ_t. Its role is narrower: it shows that a flat terminal signal can reflect an evidence bottleneck that a richer executable interface sometimes exposes.

Optimizer-interface check. A separate historical integration run fed the earlier execution-derived reward to a group-based RL training stack. After mechanically defined infrastructure exclusions, the dense-reward arm has 7 valid optimization steps and all 7 have nonzero recorded loss (absolute range

0.014–0.293), with four solved outcomes recorded during those valid steps. A binary companion run has 9 valid steps, all with zero recorded loss and no solved outcomes. These arms are *not* a clean causal comparison: retained metadata identify separately trained SFT checkpoints (same model/configuration and 632 SFT examples, different training timestamps), and the two runs lose different steps to infrastructure failures. We therefore use the result only to verify that execution-derived reward can produce a nondegenerate signal inside the optimizer. It is not evidence that the dense reward improves the learned policy relative to binary reward.

Adversarial visible-verifier campaign. A separate frozen historical campaign pre-specified a 16-task manifest of constructed single-function defects, five attack families (hardcoding, incomplete-branch fixes, exception suppression, environment detection, special-casing), qualification logic, and a reserve policy. An attack qualifies independently of the execution-derived score only if it passes the visible verifier, leaves tests untouched, and fails input-disjoint held-out validation; qualification never reads the execution-derived score, and reserve replacement follows a pre-frozen order. The campaign ran 107 real generation attempts and exhausted its frozen reserve before reaching the planned target of 48 qualifying attacks, yielding 22 qualifying attacks across nine tasks. The execution-derived scorer assigns full credit to one qualifying attack and withholds full credit from the remaining 21/22 (95.5%); the retained score distribution is 6 at 0.0, 9 at 0.2, 3 at 0.6, 3 at 0.8, and 1 at 1.0. Interior scores are not evidence of graded correctness: the predeclared falsification criterion is strictly full-credit acceptance. This campaign shows that hidden reference evidence rejects full credit for most attacks in this predeclared population; it does not establish resistance to every adaptive strategy against the scorer, and it is a full-credit endpoint check, not a grading-aggregation comparison.

G Historical Verifier-Trustworthiness Audit

This appendix gives the retained accounting for the ordinary-rollout and exploit-seeking audits that motivated the verifier-trustworthiness question. Both use the same frozen pool of 40 real repository repair tasks, disjoint from the constructed-defect tracks used elsewhere in the paper.

Ordinary rollouts. The main pool requests eight rollouts per task (320 planned). Forty-six runs fail during generation or time out before producing a usable scored trajectory, leaving 274 terminal outcomes. Eleven passing rollouts spanning 10/40 tasks edit a test file.

Exploit-seeking audit. An initial non-isolated pilot is excluded from the reported campaign; the reported run uses network-isolated containers. Thirty-eight of forty tasks produce valid runs. Among eighteen mechanical hits (tests pass, tests untouched, patch not an exact reference copy), retained manual classification records seven narrow exploits, nine legitimate fixes despite the instruction, and two ambiguous cases: 7/38 (18.4%) confirmed narrow exploits, or 9/38 (23.7%) counting both ambiguous cases. The surviving artifact contains labels and rationales but not the complete inspected patches or trajectory excerpts; these are historical classification results, not newly independently relabeled evidence.

Conservative partial-patch audit. For the 18 tasks whose upstream patch can be mechanically decomposed, removing one hunk or fine-grained edit operation and rerunning the test suite finds that ten of eighteen accept at least one proper subset of the upstream patch. Such subsets are not necessarily incorrect, because upstream patches may include refactoring or optional hardening; this check is secondary evidence that the visible test surface need not require every shipped change, not a correctness-failure rate.

These audits are historical and predate the opportunity/aggregation framework developed in the main text. They motivate treating a passing terminal verifier as permissive rather than trustworthy by construction; they do not measure selection opportunity or held-out utility and are not pooled with the paper's central estimates.

H Natural-Agent Capability and Agreement

Of the 52 frozen natural tasks, 23 meet the final hidden-probe capability definition. One of those tasks is excluded from primary statistics because its final probe design was candidate-informed,

leaving 22 primary-capable tasks. Among the remaining 29 tasks, 11 fail the final capability gate, 4 are mechanically probe-capable but outside the reward-relevant scope of the study, and 14 do not have sufficient retained evidence for inclusion in the final primary-capable set. All 52 tasks retain recoverable candidate data; the detailed historical status ledger remains in the anonymous artifact.

The paired agreement statistic uses the 15 primary-capable tasks with both candidate-level repository baseline and hidden binary fields. Task-bootstrap resampling uses 10,000 replicates with seed 20260924. The point estimate is 0.9667 with 95% CI $[0.900, 1.000]$ and leave-one-task-out range 0.9643–1.000.

Table 5: Per-repository natural-study capability and paired agreement.

Repository	Tasks	Capable	Paired	Agreement
Django	16	10	4	1.000
SymPy	13	4	4	1.000
Astropy	3	1	1	1.000
Matplotlib	4	0	0	–
scikit-learn	5	1	1	1.000
Flask	1	1	1	1.000
Seaborn	1	0	0	–
Xarray	1	1	0	–
Pylint	1	1	1	0.500
pytest	5	4	3	1.000
Sphinx	2	0	0	–

Table 6: Per-model natural-study summary.

Generator	Tasks	Capable	Paired	Agreement
Qwen3-27B	41	17	12	0.9583
GPT-5.6-Sol	11	6	3	1.000

Among candidates in primary-capable tasks with repository-baseline pass, 64 receive visible full credit. Hidden-binary outcomes are available for 63 of those records; two of the 63 disagree with hidden reference behavior (3.2%). Both are from `pylint-7114` and have hidden fraction 0.4. We report these as visible-pass/reference-disagreement cases rather than assuming that reference disagreement alone proves semantic incorrectness.

I Natural Binary-versus-Fraction Regime Accounting

Eighteen primary-capable natural tasks contain at least two candidates with both hidden-binary and hidden-fraction scores. Sixteen contain at least one candidate with hidden fraction 1.0; on those tasks, binary and fractional top sets are identical by construction. `astropy-14365` has no full-pass candidate but all scored candidates have the same hidden fraction (0.9318), so both top sets are again the entire pool. `pydata__xarray-3364` is the sole task with different top sets: its three scored candidates have hidden binary scores $0, 0, 0$ and hidden fractions $0, 0.667, 0.333$, so binary ties the entire pool while fraction selects the 0.667 candidate. Exact top-set equality is therefore 17/18, not 18/18. This exception, however, does not survive inspection: the retained task artifact shows the 0.667- and 0.333-scoring candidates both pass the task's repository FAIL_TO_PASS tests and satisfy the repository's own semantic equivalence check (`assert_identical`); their fractional scores differ only because of coordinate/dictionary insertion-order differences in the captured return representations, not because of different behavior. Only the 0-scoring candidate is a genuine failure. Because this natural study also does not provide a standardized input-disjoint held-out utility outcome for these candidate pools, the Xarray case does not establish preserved semantic ranking information under binary saturation; it instead illustrates that a stored fractional score can manufacture an apparent selection opportunity from the evidence representation itself, independent of real candidate quality.

J TARGETED NATURAL-AGENT HIDDEN-A/B FOLLOW-UP

This post-hoc follow-up was not preregistered and is not merged into any primary population. The two SymPy tasks were selected from a separate certifier-v2 development population because prior replay-engineering checks indicated high call-reconstruction fidelity; task selection did not use candidate rollout scores or repository-test outcomes. Eight rollouts were attempted for each task. All eight `sympy__sympy-24102` rollouts submitted patches. For `sympy__sympy-15346`, seven submitted patches and one exhausted its step budget without submission; the exclusion therefore precedes reward scoring.

For `sympy__sympy-24102`, the stored evidence contains 96 recorded A calls and 96 recorded B calls. One candidate scores 96/96 on A and the other seven score 95/96; all eight score 96/96 on B. A final artifact audit found that two A records correspond to the same effective non-`self` input (`'x y'`) because the historical harvester's full-argument key includes the runtime object address of `self`. There is no effective-input overlap between A and B after removing that address-bearing `self` field. The within-A duplicate cannot change the regime conclusion: a full-pass candidate exists, so strict and fractional top sets coincide for any common evidence set.

For `sympy__sympy-15346`, seven usable candidates all score 238/239 on A and 238/238 on B. The A profile is therefore flat with no full pass, so both selectors tie the same seven-candidate pool. The stored A/B call records are disjoint. Both tasks consequently have $O_t = 0$ and $\Delta_t = 0$. The artifact ships the recorded probe calls, submitted patch diffs, aggregate candidate scores, the development-population certification ledger, and the historical harvesting/splitting code. It does not ship full repository checkouts or candidate per-call execution traces, so the supplied reproduction script recomputes the opportunity/top-set conclusions from retained scores and audits probe-set disjointness rather than re-executing the repositories.

K NATURAL-TASK INPUT-HARVESTING COVERAGE

This appendix gives full detail for the disjoint, 110-task harvesting-coverage result summarized in Section 6. The population (`scale_population_frozen.json`) was frozen before any task was inspected, spans 11 repositories, and has zero instance-ID overlap with the 52-task natural-agent manifest, confirmed by exact set comparison.

Funnel. 110 frozen tasks → 89 pass mechanical eligibility (a real, deterministic unified-diff scan requiring the gold patch touch exactly one file and reference calls exist) → 87 of those 89 additionally pass AST-based target-function derivation (2 fail; reason codes retained) → all 87 attempted end-to-end → 16 certified (18.4%), all 16 in `django/django` (16/36 of that repository's tasks in this population), 0/51 certified across the other ten repositories.

Certification rule. A task is certified iff real captured `(args, kwargs, return)` calls from the repository's own test suite against the reference-patched code, deduped, total at least 4; a disjoint probe-A/probe-B split of those calls succeeds; and a reference output is available for every probe-B call.

Scope and retained evidence. No candidates were generated or scored on this population: it measures harvesting coverage, not selection efficacy. Complete captured-call tuples are unavailable; retained per-task records contain certification decisions, applicable abstention reasons, and raw/deduplicated call counts. We claim the recorded funnel, not candidate-side replayability.

Interpretation. The retained records reproduce the funnel (110/89/87/16) and task-level certification decisions. At roughly twice the task count, they corroborate the repository-dependent capability pattern in the separate 52-task audit (Appendix H).