AGENT AS A BENCH: WHAT DO AI-GENERATED TESTS CONTRIBUTE TO EVALUATION?

Anonymous authors
Paper under double-blind review

ABSTRACT

AI assistance accelerates code and research, increasing the need for reliable evaluation. Generated tests can expand this capacity, but construction success does not establish their value on other implementations. Our study identifies two reasons: detections overlap existing suites, and short packages retain different fault patterns across implementation populations. Agent as a Bench freezes generated packages to separate **complementation**, adding specification-supported detections beyond a complete suite, from **compression**, retaining its detections with fewer inputs. On HumanEval, the stronger suite already rejects 63 of 81 mismatch-selected targets. Generated tests expose one supported suite omission, while a fixed 120-target panel yields no confirmed additional detections. Compression nevertheless provides a measurable benefit: eight generated inputs retain five of six suite-rejected targets, compared with three for a random subset and four for a greedy subset, improving task-macro retention by 25 and 12.5 percentage points. These gains occur on one date task; development rankings differ. A year-range witness contributes 12.5 points against both subsets; the additional 12.5 points against Random8 involve a format-sensitive target. Uniform eight-input sampling from the fixed suite detects the year-range fault with probability 3.16%. The results establish added and retained detection as distinct sources of reuse value and show how fault-level evidence connects construction choices to that value.

1 INTRODUCTION

AI assistance accelerates code and research, but checking its outputs still requires expert effort. Model-written evaluations could expand this capacity (Perez et al., 2023; Bowman et al., 2022). We ask what generated code tests contribute when reused on other implementations (Figure 1).

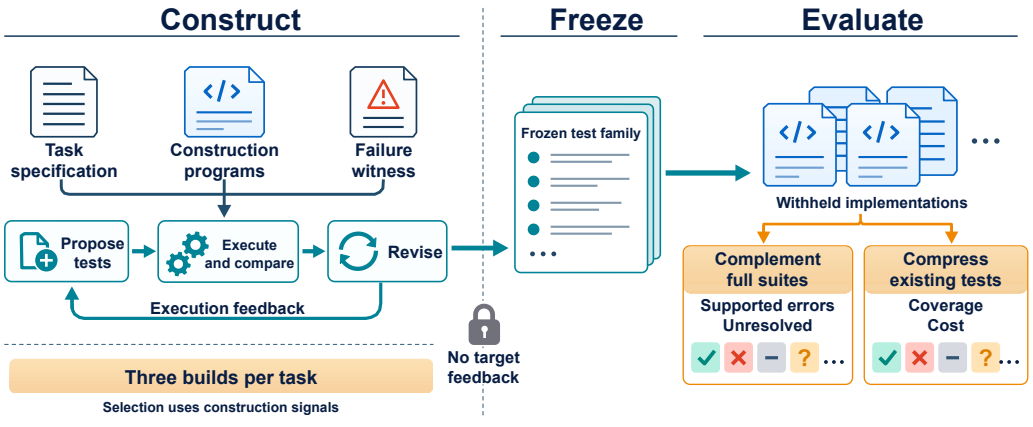


Figure 1: **From test construction to reuse.** Agent as a Bench freezes generated tests and reuses them on other implementations of the same task. Complementation measures specification-supported errors missed by a complete suite; compression measures its detections retained with fewer inputs.

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.

Test generation already delivers substantial capabilities. Coverage-guided systems build useful regression suites, and targeted generators expose errors in supplied programs. Yet these objectives differ from deciding whether a frozen package is useful on other implementations. A package can execute successfully and detect many discrepancies while adding little beyond a complete existing suite. Conversely, it can add no new error and still be valuable if a short input sequence preserves detections that would otherwise require a much larger suite. Evaluating these roles separately is necessary to identify the actual benefit.

Our insight is that reuse value depends on both the comparator and the faults encountered. In our development screening, most selected mismatches are already caught by the stronger suite. In compression, equal task-average scores hide different missed implementations, and a rare year-range witness creates a local advantage over suite subsets. These observations explain why neither a single construction score nor an aggregate detection count fully describes reuse. They motivate measuring added evidence, retained evidence, and the fault patterns behind each result.

HumanEval, introduced with OpenAI's Codex, evaluates whether programs generated from docstrings behave correctly (Chen et al., 2021). Its use in Code Llama evaluation and extension by EvalPlus make it an established setting for code-generation research (Rozière et al., 2023; Liu et al., 2023). We use its specifications, reference implementations and original/expanded suites to trace which errors generated tests add or preserve. Agent as a Bench freezes packages and tests them on other implementations of these known tasks. We ask whether construction signals select better packages, whether the packages complement a complete suite, and which detections survive eight inputs. We separate retrospective diagnostics from prospective comparisons and retain unresolved outcomes.

Our findings are concrete. Acceptance explains 0.50 pp of a 0.74-pp retrospective selection gain. Generated tests expose one specification-supported suite omission, but the fixed 120-target panel yields no confirmed complementation. On six suite-rejected targets across four tasks, C8 retains five, versus three for Random8 and four for Greedy8: macro gains of 25 and 12.5 pp. All favorable task contrasts occur on HE124, and the development population reverses the micro ranking. This local compression benefit has an identifiable source and a scope determined by the observed fault families.

Contributions. Our empirical analysis shows that construction-score gains, additional errors and retained detections answer different reuse decisions. We quantify these outcomes under complete-suite controls and explain them through acceptance, suite overlap and fault-revealing inputs. The resulting evidence identifies both a supported omission and a local compression benefit, while exposing when construction information misses the faults that matter at deployment.

2 Related work: from generating tests to reusing evidence

Regression-oriented generation. Feedback-directed testing predates LLMs (Pacheco et al., 2007). CoverUp and TestForge use execution and coverage feedback; TestGen-LLM filters for buildability, repeated passing and added coverage; TestGenEval measures file-level generation and completion (Altmayer Pizzorno & Berger, 2025; Jain & Le Goues, 2025; Alshahwan et al., 2024; Jain et al., 2025). These approaches serve regression testing, but passing the current implementation does not certify its correctness. Mathews & Nagappan (2024) show that filtering or repairing failing tests can discard bug-revealing cases. Our question is instead which errors a frozen package detects on other implementations, with specification support evaluated separately from agreement.

Targeted counterexamples. ProbeGen searches for distinctions between supplied programs and explicitly discusses probe reuse (Allamanis & Yin, 2025). TrickCatcher generates program variants from the program under test and its specification to expose bugs in plausible programs (Liu et al., 2025); CodeHacker targets particular competition submissions (Shi et al., 2026), and TCGBench evaluates targeted generators (Cao et al., 2025). Their target-conditioned objectives differ from freezing a package before deployment outcomes. Our information boundary measures what remains useful after generation has ended, without target-specific repair.

Suite reduction and reuse. Suite minimization studies retained fault detection (Yoo & Harman, 2012). EvalPlus expands tests and reduces them using coverage, mutant kills and model-sample kills, including leave-one-model-out validation (Liu et al., 2023). Compression and cross-model evaluation therefore have clear precedents. We study both complete-suite complementation and fixed-budget

retention of generated packages, tracing outcomes to specific faults. Random8 and Greedy8 are our specified controls, not a reproduction of the published EvalPlus reducer. CodeT's generated tests select code solutions rather than test packages (Chen et al., 2023).

Proxies and deployment evidence. Mutant detection can correlate with real-fault detection, although results depend on evaluation conditions (Just et al., 2014; Papadakis et al., 2018). For LLM-generated tests, Zhao et al. (2026) find useful inter-model signals in regression settings but unreliable indicators when the supplied code is buggy. These findings make the intended use central to interpreting a metric. Our contribution is to measure two reuse endpoints and explain their differences through frozen-package and fault-level evidence.

3 Deployment validity: two uses and a selection decision

To measure reuse without target-specific repair, we freeze the input generator, reference cache, acceptance decision and seeded input order after construction. A complete-suite comparator separates additional evidence from overlap; task/target weightings and fault witnesses expose what aggregate scores hide. Target outcomes cannot change the package, choice rule or uncertainty treatment (Figure 1).

Selecting a package. A construction-only rule tests whether available scores guide reuse by choosing among three builds, compared with uniform choice (Appendix B). Development uses New@k: the fraction of all targets newly mismatched beyond the initial witness, not specification-certified fault recall. Later endpoints measure specification-supported errors.

Complementing an existing suite. To isolate additional evidence, a supported detection counts only when the complete base-plus suite passes. The primary rate averages over six implementations on each of 20 tasks (Equation 2, Appendix C). All 120 targets remain in the denominator, including suite failures and unknowns; build probabilities are frozen.

Compressing detections. Let $\mathcal{U}_t$ contain targets that pass complete original tests B_0 but are rejected by B_+. Retention is the fraction of $\mathcal{U}_t$ detected by a short package. Task-macro retention gives equal weight to tasks with nonempty support; target-micro retention weights each target equally. Empty support is reported, not assigned zero or perfect retention. The compression criterion was fixed at 95% task-macro retention. A supported subset, when used, is identified separately from suite-policy rejection.

Accounting for unresolved outcomes. Timeouts, missing slots and unresolved semantics yield finite-panel identification bounds. Shared unknowns are completed jointly in paired contrasts. These bounds describe label uncertainty; task-bootstrap intervals describe resampling uncertainty.

4 Study design and reusable packages

Table 1 separates populations and comparison timing. Panels overlap; the exploratory follow-ups reuse the prospective 120 targets (Appendix E). They cannot be pooled as independent samples. All tasks are known HumanEval tasks: unused implementations do not establish unseen-task generalization.

Construction. GPT-5.6-Luna with medium reasoning constructs packages on CodeGen-6B archives (Nijkamp et al., 2023); original targets are Mistral-7B archives (Jiang et al., 2023). Three builds per task share a legal donor/reference mismatch from bounded source-order search. Exact-source, normalized-AST and near-clone exclusions limit construction–target source overlap.

Policy C uses eight construction inputs, one revision informed by program/reference outcomes, and a final eight-input check without repair. Retention requires parseability and at least one executable input. Invalid, repeated and unknown inputs consume slots; unsuccessful builds retain costs and use a frozen mutation fallback. Seeded sequences support 8/16/32-slot prefixes; C8 and C32 denote the corresponding C prefixes. Appendix A details C and the original controls: replay R, mutation M+, paired construction A, on-only ablation B and continued search D.

Deployment controls. To compare generated packages with suite-based selection rules, Random8 uses SHA256-ranked suite inputs and Greedy8 uses construction coverage with deterministic ties and filling. Boundary32 uses AI-assisted specification-only design; SuiteGreedy32 uses suite inputs and construction coverage. All exclude target outcomes. Equal scheduled slots include fallbacks and invalid positions, not equal construction costs.

Table 1: Main populations (implementations / tasks). Panels overlap and cannot be added. Timing concerns the stated comparison; all tasks are known.

Study	Population	Design and conditional support
Selection	2,584 / 162	Retrospective; three builds per task.
Complement, development	81 / 36	Mismatch-selected; 18 complete-suite survivors.
Complement, prospective	120 / 20	Six per task; panel and endpoint fixed before outcomes.
Compression, development	87 / 24	Complete-suite rejection support: 18 / 7.
Producer comparison	39 / 21	Fixed before its outcomes; rejection support: 6 / 4.
Repetition / depth	27 further; 66 exposed	Different conditional supports; depth is retrospective.

Exploratory generator follow-ups compare continuations with and without new execution outcomes on the same 120 targets (Appendix E). Their staged timing and full results remain separate from the original prospective comparison.

Comparisons and semantic support. To avoid counting every mismatch as an error, we separate mismatches, suite-policy rejections and specification-supported violations. Support requires a specification-justified legal input and an observed violating output (Barr et al., 2015). Prospective semantic cards precede target outcomes. Sources execute in restricted containers; numerical reanalysis executes no targets.

5 Build acceptance explains most of the selection gain

We first ask whether construction information helps choose among frozen packages. C, the strongest original policy under the mismatch score, retains 466 of 486 artifacts and reaches task-mean New@32 of 21.23%. The shared witness already rejects 58.74% of targets; these remain in the denominator but receive no additional New credit. Appendix B reports the original policy contrasts and their limitations.

Does the construction score choose a better C package? Our retrospective selector first prefers accepted builds, then maximizes the number of construction programs rejected during the final eight-input check. It averages ties uniformly. The control chooses uniformly among the same three frozen builds. Both have the same sunk construction budget and available artifacts; neither changes a package after target outcomes.

Joint selection raises New@32 from 21.23% to 21.97%: $+0.74$ pp with interval $[+0.30, +1.26]$ pp from 10,000 task-bootstrap resamples. Fourteen tasks improve and none decline in this expected score. The primary 32-slot outcome and secondary 8/16-slot results are reported in Appendix B.

What makes the selector useful? A post-result accepted-only ablation reaches 21.73%. Avoiding unaccepted builds contributes 0.50 pp, or 67.83% of the joint gain. Coverage beyond acceptance contributes 0.24 pp across five tasks. Package validity therefore explains most of the measured selection benefit; constructor coverage supplies a smaller, separately measured increment.

The target oracle is 1.95 pp above uniform choice. Joint selection captures 0.74 pp, leaving 1.22 pp within highest-score ties. All three scores tie on 128 of 162 tasks, including 22 with differing outcomes (Appendix B).

From selection to complementation. Acceptance explains most of the New@32 mismatch gain at the same sunk construction cost. On the prospective specification-supported complementation endpoint, the same selector instead has paired increments of exactly $[0, 0]$ pp over uniform and accepted-only choice (Section 6.1). These are distinct measures of reuse value.

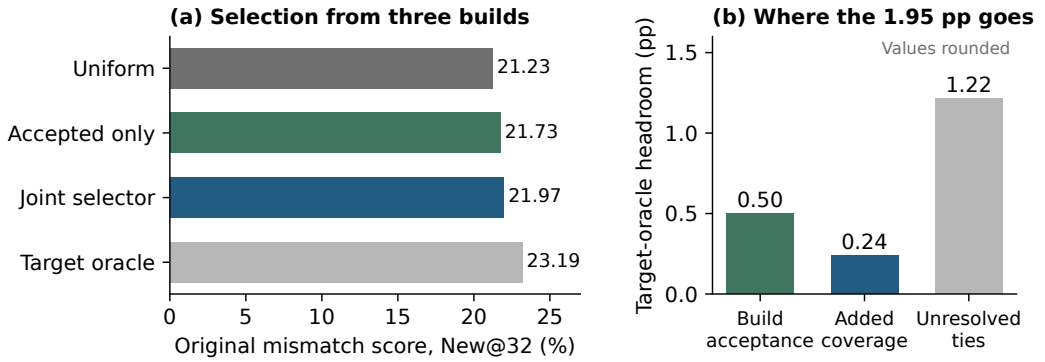


Figure 2: Retrospective selection among three C builds. New@32 is the task-average fraction of all targets newly mismatched beyond the common witness at 32 slots, under the original comparator. Acceptance explains 67.83% of the joint gain; coverage adds 0.24 pp. The post hoc decomposition separates these contributions. The target oracle uses deployment outcomes; its remaining 1.22 pp lies within highest-score ties.

6 Complementation beyond complete test suites

Does a generated test add evidence the complete suite misses? We first screen mismatch-selected targets to locate overlap and omissions. Because that screening selects on observed mismatches, we then use a fixed implementation panel to measure additional detections without that selection.

Most selected detections overlap the stronger suite. Complete original HumanEval tests pass 509 of the 2,584 development targets, fail 2,041, and leave 34 unresolved. Among the passes, the union of previously recorded mismatches across policies, runs and prefixes up to 32 slots selects 81 targets on 36 tasks. Complete EvalPlus base-plus tests reject 63 and pass 18. Thus 63/81 selected candidates are already detected by the stronger baseline. The 63/81 overlap rate is conditional on this mismatch-selected subset.

Table 2: Complete-suite screening of 81 mismatch-selected targets. The 17 unconfirmed survivor labels remain unresolved.

Disposition	Targets
Rejected by complete base-plus tests	63
Passes base-plus; frozen specification-supported witness	1
Passes base-plus; supported witness not established	17

A generated test supplies genuine complementation. HumanEval/12 asks for a longest string from a list, returning `None` only when the list is empty. For the nonempty list `[""]`, the archived target returns `None` instead of the empty string. Both complete suites pass the target. C's frozen run-0 sequence exposes the error at slot 11; D exposes it at slot 5 in run 2. The test is valuable precisely because the specification settles the behavior and the complete comparator misses it.

Replaying all 8,694 frozen slots on the 18 survivors recovers 1,711 previously unobserved suffix slots. Six supported observations identify one distinct target/input error, reached by C after the eight-slot prefix. Other observations include representation-only differences, Unicode-policy ambiguity, invalid inputs and execution uncertainty. With 17 unresolved target labels, the number of erroneous survivors is identified between one and eighteen. Figure 6 exposes the candidate overlap and all survivor slots.

A prospective test of the deployment claim. To estimate complementation on a fixed population, we freeze six unused archival CodeLlama implementations (Rozière et al., 2023) on each of 20 known tasks before new execution outcomes, without selecting for an original-test pass. The task

frame is inherited from earlier depth/selection work. HE22 is structurally missing because its source components link to prior implementations; Appendix C records the frame and freeze timeline.

Each of the 120 targets is evaluated by original tests, complete base-plus tests, three frozen C32 builds, Boundary32 and SuiteGreedy32. Uniform C, accepted-only C and joint selection are pre-specified comparisons. Equation 2, using all six targets per task, is the primary endpoint. All-target supported detection is secondary. The protocol retains unknowns, historical missing slots and fixed tie probabilities; it does not add inputs or retries in response to observed failures.

This design measures both the value of the generated packages and the incremental value of choosing among them, using a common complementation endpoint.

6.1 A FIXED-PANEL ESTIMATE OF BEYOND-SUITE DETECTION

Complete base-plus tests pass 40 of the fixed 120 targets, reject 77, and leave three UNKNOWN. There is no definite specification-supported complementation. Joint C32's primary identification bound is $[0, 2.22]\%$. Its paired complementation increment over both uniform and accepted-only C is exactly $[0, 0]$ pp. All three C choice rules have the same finite-panel bounds; these establish neither a zero population rate nor a statistical confidence upper bound.

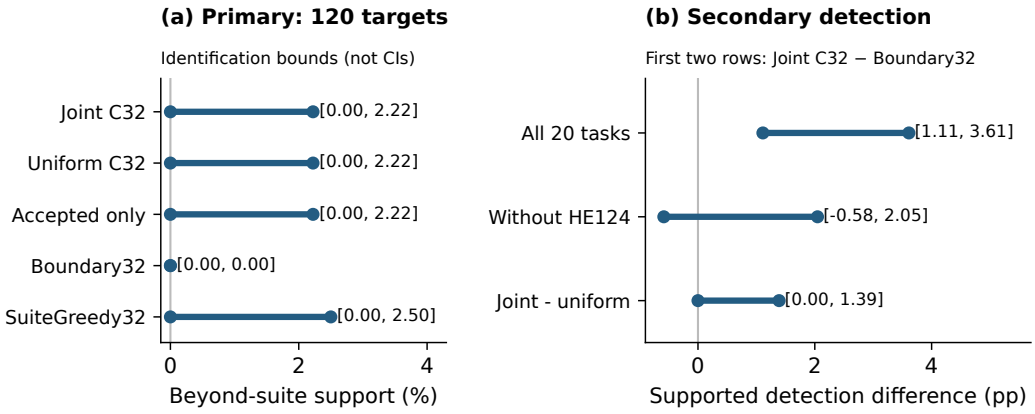


Figure 3: Prospective results and a post hoc sensitivity check. Bars are identification bounds, not confidence intervals. Primary complementation counts supported detections on complete-suite passes, keeping all 120 targets in the denominator. Secondary detection includes suite-caught errors. Right: first two rows, joint C32 minus Boundary32; last, joint minus uniform. Omitting HE124 removes the positive lower bound of the secondary joint-minus-Boundary advantage; the primary analysis is unchanged.

The entire 2.22% upper endpoint comes from unresolved method slots or execution outcomes, not confirmed additional errors. Appendix C decomposes these sources without changing the fixed denominator.

Error coverage across all targets. Across all targets, joint C32 has supported-detection bounds of 61.94–64.44%. Its paired advantage over Boundary32 is 1.11–3.61 pp, and over uniform or accepted-only C it is 0–1.39 pp. These are secondary endpoints that include errors already caught by the complete suite. A post hoc omission of HE124 changes joint-minus-Boundary to $[-0.58, +2.05]$ pp and joint-minus-uniform to exactly zero. HE124 accounts for the positive lower bound of the Boundary32 contrast and the selection increment.

The original-pass/base-plus-reject support contains four targets on two tasks: three on HE82 and one on HE124. Joint retention is 83.33% macro and 75% micro; Boundary32 is 50% macro and 75% micro. The macro difference is +33.33 pp, but the micro difference is zero. The reciprocal witnesses explain how both statements can hold: equal-task weighting gives the single date target as much total weight as the three prime-length targets. The weighting determines which of these two fault families drives the aggregate comparison.

Table 3: Reciprocal supported witnesses in the prospective panel. Both targets already fail the complete suite, so neither is primary complementation.

Target	Specification and observed violation	Frozen-package result
HE124/130	Date `01-1-1900` violates the frozen `mm-dd-yyyy` width rule; target returns `True`.	C32 build 2 detects; Boundary32 passes.
HE82/10	String length 121 is composite (11^2); target returns `True` for the prime-length predicate.	Boundary32 detects; all three C32 builds pass.

7 COMPRESSING: WHICH DETECTIONS SURVIVE A SHORT PACKAGE?

Can a short package preserve detections even when it adds no new errors? We compare frozen eight-slot packages on a development panel and another archival producer. Their small conditional supports reveal which faults survive; reporting both task-macro and target-micro retention exposes differences hidden by either weighting alone.

Development: equal macro scores hide different misses. The separate development panel contains 87 original-test-passing targets on 24 tasks and 96 construction implementations. Complete base-plus tests pass 62 targets, reject 18 on seven tasks and leave seven unknown. C8 detects 9/18 rejected targets, while Random8 and Greedy8 each detect 14/18. All three attain the same task-macro retention, $5/7 = 71.43\%$ (Figure 4).

The equality has a concrete explanation. C and Random8 exchange one detected HE0 target for six HE22 targets; C and Greedy8 exchange one HE124 target for those six HE22 targets. All packages miss three HE123 targets. The C-minus-subset macro difference is 0 pp with paired task-bootstrap interval $[-42.86, +42.86]$ pp. All three fall below the prespecified 95% macro-retention criterion; the interval describes substantial task-level variation around their equal aggregate scores. Figure 7 shows which task detections are exchanged.

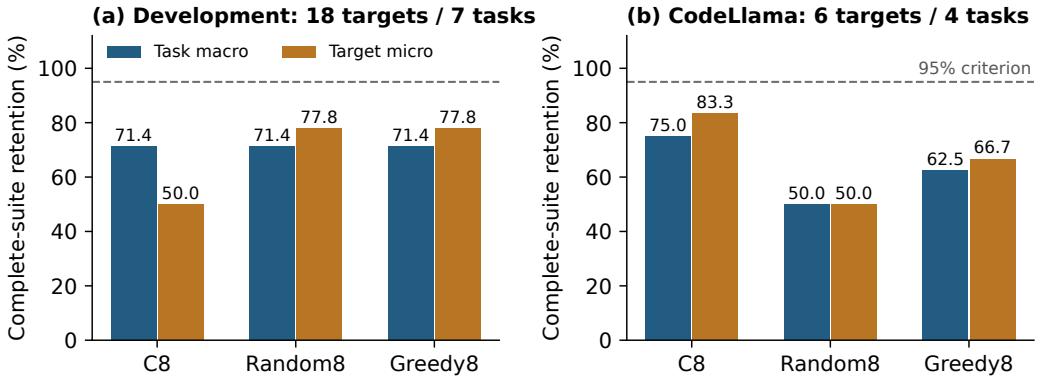


Figure 4: The same eight-slot packages on separate conditional populations. Macro retention weights tasks with suite-rejected targets equally; micro retention weights each such target equally. C's micro ranking improves in the CodeLlama comparison, but all favorable task contrasts there occur on HE124. The populations differ in producer, screening and support.

Retention on CodeLlama implementations. Before further outcomes, we freeze a CodeLlama-7B source panel and the same 24 task specifications and package inputs. Static filtering leaves 186 candidates: eight on 23 tasks and two on HE22. All original-test screens finish first, giving 66 passes, 117 failures and three resource/time unknowns. Taking at most the first two passes per task yields 39 targets on 21 tasks. Three tasks without a pass remain in the coverage report.

Complete base-plus tests reject six targets on four tasks, pass 32 and leave one unknown. C8 detects five of six rejects, Random8 three and Greedy8 four. Their macro retentions are 75%, 50% and 62.5%. The C-minus-control macro differences are +25 and +12.5 pp, with four-task bootstrap intervals

$[0, +75]$ and $[0, +37.5]$ pp; exact enumeration verifies the endpoints. All positive differences occur on HE124. The next analysis examines the date-error inputs responsible for that contrast.

Execution costs on the measured support. On the development 18-target support, C and both suite subsets take approximately six seconds each, versus 14.05 seconds for complete base-plus execution. Across all 87 targets, larger timing differences concentrate on resource-uncertain programs. Historical construction wall time is incomplete, and suite subsets can also be reused. Figure 8 separates these execution costs from the unavailable construction-time accounting.

7.1 Which fault patterns drive short-package retention?

Why does the short generated package retain a detection that suite subsets miss? We measure the density of fault-revealing suite inputs, then examine the generated witness. Repeated builds and boundary controls test how far this local explanation extends.

A rare detection explains a real local advantage. Figure 5 compares observed detection with exact uniform-eight sampling (Equation 3, Appendix D). For HE124/145 in the producer comparison, only 4 of 1,001 suite input IDs reject the target: uniform-eight detection probability is 3.16%. C's input `12-09-8842` is absent from the suite pool: the target returns `False` because it imposes an unrequested 2000–2020 year bound, although the specification admits this date. By contrast, HE105/174 fails 434 of 1,004 IDs. The +25-pp C-minus-Random8 contrast includes this year-range witness and the format-policy-sensitive C8 witness on HE124/123; the +12.5-pp C-minus-Greedy8 contrast differs only on HE124/145. Both contrasts retain the suite-policy denominator (Appendix D). These cases are distinct from the prospective HE124/130 padding witness (Table 3). A post hoc fixed-denominator stress test gives no method credit on format-sensitive HE124/123: C8 still leads both subsets by 12.5 pp macro (Appendix D).

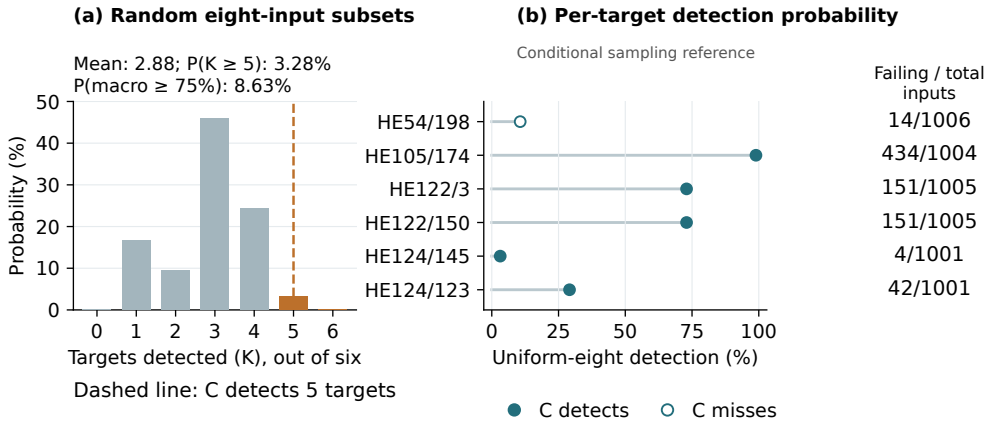


Figure 5: Finite-pool reference on six suite-rejected targets. Left: the exact distribution of detections from uniform-eight sampling, preserving within-task dependence; orange bars mark at least five detections, and the dashed line marks C's observed five. Right: per-target sampling detection probabilities and failing/total suite input counts; filled markers are C's observed rejections. The 3.28% tail is a conditional sampling probability, not a method-comparison p-value.

The exact sampling reference retains dependence between targets sharing a task by grouping input IDs by their joint failure mask. The probability of reaching C's 75% macro retention is 8.63%, distinct from the 3.28% detection-count tail. The calculation conditions on the cached execution matrix and preserves the dependence between targets that share a task.

The useful year range appears before revision. The accepted HE124 generator produces the non-seed input through a broad year-sampling strategy already present in its initial proposal. The rare date detection is therefore associated with that generated input family. The paired continuation study in Section 8 examines the separate contribution of execution feedback.

Coverage across builds and input budgets. Evaluating the remaining 27 original-test passes from the same source screen yields five further suite rejections. A fixed sparsity rule retains three targets but only two non-date task/fault families, below its planned three-family condition. On that support, C's three eight-slot builds average 50% macro detection, matching the boundary packages; the suite greedy package reaches 100%. The wider 32-slot diagnostic improves recurrence of HE0 and HE18 witnesses, but all three C builds still miss HE54/198's case-fold error, which Boundary32 detects.

The HE54 case pinpoints a gap in the construction signal: all nine qualified constructors pass the inputs that expose this target error. Greedy8 similarly increases constructor coverage while its target detection count remains unchanged. Appendix D gives the counts and finite-pool scope.

8 Discussion: implications for AI-assisted evaluation

Choose the deployment role before the score. The HE12 omission establishes that generated tests can complement a suite; the fixed panel yields no confirmed complementation. On six targets across four CodeLlama tasks, C8 improves macro retention by 25 pp over Random8 and 12.5 pp over Greedy8, entirely on HE124. The Random8 contrast includes the format-policy-sensitive HE124/123 witness (Section 7.1). Development has equal macro scores but fewer C8 detections. Reuse therefore requires specifying the comparator, population and retained fault families.

Use construction scores for the decision they support. Acceptance explains 0.50 pp of the 0.74-pp selection gain. Yet all nine qualified HE54 constructors pass the inputs revealing the target error: their coverage cannot reward that distinction. Acceptance is a useful first filter here; deployment comparisons establish what evidence the selected package adds or preserves.

Feedback and input coverage. The exploratory contrasts in Table 8 examine coverage loss. Terra's HE82 feedback packages twice restrict all eight inputs to length five, losing detections reached without new feedback. Omitting HE82 makes Sol's contrast include zero. These staged probes reuse an observed pilot; 360 target–build observations per arm represent 20 tasks and 120 targets. Stochastic continuation variation is not isolated: the observed coverage loss does not establish a causal feedback effect (Appendix E).

Limitations and next steps. The studies concern known HumanEval tasks and archived implementations. Clone filtering reduces source overlap; task familiarity and pretraining exposure remain uncontrolled. Broad selection and compression use one historical generator; additional serving routes cover only 20 tasks. Compression supports are small, and producer comparisons also change screening and support. Selection and several mechanism analyses are retrospective; the stated prospective comparisons precede their new outcomes.

Semantic labels rely on AI-assisted specification review. Checks on one canonical implementation per task leave false-rejection rates across other correct implementations unmeasured. Identification bounds retain unresolved semantics, missing slots and execution uncertainty. They condition on retained semantic labels and do not cover systematic errors in labels marked supported. Input budgets and model usage are reported separately; incomplete construction wall time precludes an end-to-end speedup claim. Further studies should test additional task distributions, broader fault families and repository-level settings such as SWE-bench (Jimenez et al., 2024), with independently checked semantics and complete cost accounting.

Conclusion. Generated tests provide two distinct forms of value: additional detections beyond a complete suite and retained detections under a smaller budget. Our results demonstrate both a supported omission and a local compression gain, and identify the fault patterns that sustain them. Evaluating these roles separately makes reuse benefits measurable and their limits inspectable.

AI USE STATEMENT

AI models generated the executable test artifacts studied here. AI tools also assisted research planning, hypothesis and protocol design, implementation, analysis, semantic review, literature retrieval, manuscript preparation and figures. The original generator is GPT-5.6-Luna; follow-ups use recorded Grok, Terra and Sol routes (Appendix E). Boundary controls and semantic cards are AI-assisted, with costs not matched to the generator. Results derive from preserved records. Verification includes numerical replay, scoring-rule regression tests, arithmetic checks of selected witnesses, citation checks and rendered-PDF inspection. Semantic review is not independent human certification. The authors are responsible for the final text, claims and artifacts.

ETHICS STATEMENT

This study evaluates generated code tests on public programming benchmarks and archived model-generated implementations; it does not recruit human participants or collect personal data. Generated programs and tests can be unsafe to execute, so reuse requires appropriate isolation and resource limits. Passing a finite test suite must not be treated as a correctness or security guarantee. We report unresolved outcomes and the limits of AI-assisted semantic review to reduce the risk of overstating reliability. Benchmark and software reuse remains subject to upstream licenses and attribution requirements; the numerical exports do not grant new rights over upstream artifacts.

REPRODUCIBILITY STATEMENT

The source package compiles the manuscript, regenerates the seven statistical figures and follow-up tables, and checks key numbers from frozen exports without model inference or candidate execution. Figure 1 is supplied as a vector asset. Historical figure versions retain their original bytes and provenance in `RESTORED_FIGURES.json`; the current plotting scripts use the same frozen numerical inputs. Appendices specify construction, selection, uncertainty and conditional supports; Appendix F maps claims to evidence. The numerical package does not recreate the execution campaign: complete execution reproduction additionally requires the preserved program, container and operation archives.

REFERENCES

Miltiadis Allamanis and Pengcheng Yin. Disproving program equivalence with LLMs. *arXiv preprint arXiv:2502.18473*, 2025. URL https://arxiv.org/abs/2502.18473v1.

Nadia Alshahwan, Jubin Chheda, Anastasia Finogenova, Beliz Gokkaya, Mark Harman, Inna Harper, Alexandru Marginean, Shubho Sengupta, and Eddy Wang. Automated unit test improvement using large language models at Meta. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, pp. 185–196, 2024. doi: 10.1145/3663529.3663839.

Juan Altmayer Pizzorno and Emery D. Berger. CoverUp: Effective high coverage test generation for Python. *Proceedings of the ACM on Software Engineering*, 2(FSE):2897–2919, 2025. doi: 10.1145/3729398.

Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering*, 41(5):507–525, 2015. doi: 10.1109/TSE.2014.2372785.

Samuel R. Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022. URL https://arxiv.org/abs/2211.03540v2.

Yuhan Cao, Zian Chen, Kun Quan, Ziliang Zhang, Yu Wang, Xiaoning Dong, Yeqi Feng, Guanzhong He, Jingcheng Huang, Jianhao Li, Yixuan Tan, Jiafu Tang, Yilin Tang, Junlei Wu, Qianyu Xiao, Can Zheng, Shouchen Zhou, Yuxiang Zhu, Yiming Huang, and Tianxing He. Can LLMs generate reliable test case generators? A study on competition-level programming problems. *arXiv preprint arXiv:2506.06821*, 2025. URL https://arxiv.org/abs/2506.06821v4.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ktrw68Cmu9c.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Kush Jain and Claire Le Goues. TestForge: Feedback-driven, agentic test suite generation. *arXiv preprint arXiv:2503.14713*, 2025. URL https://arxiv.org/abs/2503.14713.

Kush Jain, Gabriel Synnaeve, and Baptiste Rozière. TestGenEval: A real world unit test generation and test completion benchmark. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/26ded5c8ee8ec1bc4caced4e1c9b1584-Abstract-Conference.html.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023. URL https://arxiv.org/abs/2310.06825.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html.

René Just, Darioush Jalali, Laura Inozemtseva, Michael D. Ernst, Reid Holmes, and Gordon Fraser. Are mutants a valid substitute for real faults in software testing? In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 654–665, 2014. URL https://homes.cs.washington.edu/~rjust/publ/JustJIEHF2014b-abstract.html.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572, 2023. doi: 10.52202/075280-0943. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Kaibo Liu, Zhenpeng Chen, Yiyang Liu, Jie M. Zhang, Mark Harman, Yudong Han, Yun Ma, Yihong Dong, Ge Li, and Gang Huang. LLM-powered test case generation for detecting bugs in plausible programs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 430–440, 2025. doi: 10.18653/v1/2025.acl-long.20. URL https://aclanthology.org/2025.acl-long.20/.

Noble Saji Mathews and Meiyappan Nagappan. Design choices made by LLM-based test generators prevent them from finding bugs. *arXiv preprint arXiv:2412.14137*, 2024. URL https://arxiv.org/abs/2412.14137v1.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. CodeGen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=iaYcJKpY2B_.

Carlos Pacheco, Shuvendu K. Lahiri, Michael D. Ernst, and Thomas Ball. Feedback-directed random test generation. In *Proceedings of the 29th International Conference on Software Engineering*, pp. 75–84, 2007. URL https://homes.cs.washington.edu/~mernst/pubs/feedback-testgen-icse2007-abstract.html.

Mike Papadakis, Donghwan Shin, Shin Yoo, and Doo-Hwan Bae. Are mutation scores correlated with real fault detection? A large scale empirical study on the relationship between mutants and real faults. In *Proceedings of the 40th International Conference on Software Engineering*, pp. 537–548, 2018. doi: 10.1145/3180155.3180183.

Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, et al. Discovering language model behaviors with model-written evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13387–13434, 2023. doi: 10.18653/v1/2023.findings-acl.847. URL https://aclanthology.org/2023.findings-acl.847/.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. Code Llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023. URL https://arxiv.org/abs/2308.12950.

Jingwei Shi, Xinxiang Yin, Jing Huang, Shengyu Tao, and Jinman Zhao. CodeHacker: Automated test case generation for detecting vulnerabilities in competitive programming solutions. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2352–2382, 2026. doi: 10.18653/v1/2026.acl-long.108. URL https://aclanthology.org/2026.acl-long.108/.

Shin Yoo and Mark Harman. Regression testing minimization, selection and prioritization: A survey. *Software Testing, Verification and Reliability*, 22(2):67–120, 2012. doi: 10.1002/stvr.430.

Junda Zhao, Shurui Zhou, and Eldan Cohen. Do coverage and mutation scores of LLM-generated test suites correlate with their effectiveness? (replicability study). *arXiv preprint arXiv:2607.22880*, 2026. URL https://arxiv.org/abs/2607.22880v1.

A CONSTRUCTION PROTOCOL AND STUDY BOUNDARIES

A.1 SAMPLING, GENERATION, AND APPLICATION

The original construction universe is all 164 HumanEval tasks. HumanEval/50 and /53 lack distinct new target groups after static exclusions, leaving 162 evaluable tasks. Targets are selected in source order from remaining Mistral groups, without target-outcome screening, with at most sixteen per task. HE22 has fourteen and HE45 ten targets. The total is 2,584 distinct implementations. Three separately sampled builds per task share specifications, donor, witness and construction pool; 7,752 build-target applications are not 7,752 independent programs.

Exact source, normalized AST, and token grouping are computed jointly across archived producers within each task. Near-duplicate token 5-gram edges use Jaccard similarity at least 0.85 and size ratio at least 0.85; connected components retain transitive links. Targets exclude prior exposure, scanned donor groups and final construction groups. This prevents known static overlap without proving semantic independence.

The donor is the first program in the bounded source-order scan with successful but unequal donor/reference outputs on a legal input. Remaining constructor slots are filled in source order. If no witness is found, a single shared seed-search request can return at most sixteen bounded inputs. Qualification uses JSON positional arguments with limits on size, nesting, finite numeric magnitude and task-specific types. The implemented contract is a bounded domain, not proof of full natural-language specification coverage. Partial original/extended-reference qualification on HE17 and HE78 remains unresolved.

Construction responses contain bounded executable samplers. For C, initial eight-input execution supplies all construction-pool and reference outcomes to one revision; the final eight-input check has no additional repair. Parseability and at least one executable input suffice for retention. Application freezes a witness-plus-31-input sequence using fresh seeds. Stored hashes link code, metadata, acceptance, domain/comparator version and the concrete sequence. Invalid, repeated and inconclusive positions are consumed without replacement. Definite mismatches stop later batches; already-executed batch tails remain charged. Ordinary exceptions count as mismatches only on legal inputs with supported successful reference outcomes; timeouts, missing dependencies and relevant environment failures remain UNKNOWN.

A.2 ORIGINAL POLICY FAMILY

Table 4: Original policies; retention is policy-specific and not a common semantic-quality threshold.

Policy	Construction rule	Retained	Fallback
R	Replay the shared witness.	—	—
M+	Bounded boundary and failure-preserving mutations; no model revision.	—	—
A	Paired on/off generation with input-property, donor contrast and diversity checks.	184	302
B	Shared A initialization where eligible; on-only generation, donor-error and diversity checks.	360	126
C	Ordinary execution-feedback generation with one revision.	466	20
D	Continue pair search after an initial error, using one revision.	472	14

All retention/fallback denominators are 486 attempts. M+ alternates boundary and failure-preserving channels after the witness; empty channels replay the witness. A declares an input-only predicate, sampler, edit path, edit rule and preserved features. Host-computed task-specific properties, not model-supplied checks alone, validate them. Four final fresh pairs must pass legality/relation checks, contain at least three distinct on-inputs, and meet at least three on-mismatches and at most one off-mismatch. Only the on-generator is deployed. B instead requires eight legal, decidable condition-satisfying inputs, at least six distinct and six donor mismatches. A–B thus changes input allocation, contrast information and acceptance; it is not an isolated causal effect of off-feedback. D selects an untried behavior-cluster pair when available and can reuse an existing pair otherwise. It does not reproduce another paper's trained verifier.

B CONSTRUCTION SELECTION: FULL RULE AND NUMERICAL CHECKS

For task t, build b, and input budget k, let $q_{tb}(k)$ be a deployment score and $\pi_t(b)$ a construction-only choice distribution. Selection utility against uniform choice among three builds is

$$\Delta_{\text{sel}}(k) = \frac{1}{T}\sum_{t}\sum_{b=1}^{3}\left[\pi_t(b) - \frac{1}{3}\right] q_{tb}(k). \tag{1}$$

Let $a_{tb} \in \{0, 1\}$ indicate acceptance, and c_{tb} count constructor programs rejected in the final eight-input check. Joint selection assigns equal probability to builds with the lexicographically maximal pair (a_{tb}, c_{tb}). Uniform assigns $1/3$ to each build. Accepted-only chooses uniformly among accepted builds, reverting to uniform if none are accepted. Original outcomes use the stored lower-credit mismatch policy: unknowns do not become positive detections. The target oracle maximizes the observed deployment score and is only a diagnostic upper comparator over these same three artifacts.

Table 5: Retrospective selection of C packages. Scores are original New@k in percent; increments and intervals are percentage points. New@32 is primary. Bootstrap uses 10,000 task resamples, seed 20260914. Secondary intervals are descriptive and unadjusted.

Slots	Uniform	Joint	Gain [95% interval]	Oracle headroom
8	19.53	20.29	+0.75 [+0.27,+1.29]	2.80
16	20.77	21.48	+0.71 [+0.26,+1.24]	2.07
32	21.23	21.97	+0.74 [+0.30,+1.26]	1.95

At 32 slots, the exported joint gain is 0.739455 pp: 0.501543 pp from acceptance and 0.237912 pp from additional coverage. Their ratio gives the 67.8261% acceptance share. The latter gain comes from HE124, HE141, HE119, HE75 and HE43. The acceptance ablation was requested after reading the primary result and is labeled post hoc. On the 145 tasks where all three builds are accepted, joint gain is 0.244 pp; HE124 and HE141 account for 88.24% of that subgroup increment. This subgroup does not replace the 162-task denominator.

The oracle score is 23.186728%; its headroom over uniform is 1.954733 pp, of which joint selection captures 37.8289%. All-score ties occur on 128 tasks, of which 22 have target variation; 36 tasks have any target-side headroom. Highest-score tie loss at 32 slots is 1.215278 pp and ranking loss is zero. At eight slots there is also approximately 0.08 pp ranking loss. C@8 joint selection improves fifteen tasks and worsens three, precluding an inference of universal no harm from the zero observed losses at 32 slots.

Costs and unknowns. Both selectors over the same three builds share their sunk costs. Relative to drawing one new build, three-build construction has three times the expected historical model-cost attribution. The whole-panel token-equivalent cost bounds are [2.4059,5.0849] USD for three builds and [0.8020,1.6950] USD for one uniformly chosen build. They are accounting proxies, not invoices, and do not include a complete execution-cost comparison. Raw C records retain 992 final-construction unknown cells and 4,265 target unknown slots at 32 inputs; these are different record types and remain unresolved.

Original policy context. At 32 slots, A–B over all 162 tasks is -2.38 pp with interval $[-4.37, -0.55]$; on the 112 paired-eligible tasks it is -1.15 pp with interval $[-2.49, +0.15]$. A–C is -11.57 pp over all tasks. These original-comparator results locate the central C policy; they do not establish semantic fault-coverage differences. Other policies' retrospective selectors are preserved in the numeric export. No outcome-dependent choice of a new winning method or revised scoring rule is introduced in this manuscript.

Replay. `data/t116_RESULT.json`, `t116_ACCEPTANCE_ABLATION.json`, `t116_COMPACT_DATA.json`, and the analysis specification contain the numerical selection evidence. The original independent stdlib/Fraction verification and raw execution-record references remain in the development-selection archive. This paper's figure regeneration reads the numerical exports only.

C PROSPECTIVE CONFIRMATION: ENDPOINTS AND EVIDENCE

Task frame and freeze timeline. The frame was inherited from the earlier 21-task depth/selection manifest, not freshly sampled from the 162-task study. Its IDs coincide with the 21 tasks having selected original-test-passing targets in the preceding CodeLlama compression panel. Earlier development, compression and depth/selection results were therefore available; this is a prospective test on unused implementations of known tasks.

The static preparation specification was fixed on 14 September 2026 at 11:40 UTC, ordering candidates by a fixed salted SHA256 of task ID and archive member. It inspected 200 archived programs per task and excluded components linked to previously exposed programs or references. HE22's 200 programs formed 13 components, all blocked; the other 20 tasks each supplied six unused components. The revised 120-target scope was frozen at 13:04 UTC and authorized at 14:39 UTC before execution. New original-test, complete-suite and method outcomes did not determine inclusion. The inherited frame is not treated as a random task sample; this preparation record does not independently establish how the antecedent study originally chose its task frame.

The eligible HumanEval IDs are 0, 15, 18, 24, 25, 54, 59, 71, 74, 78, 79, 82, 87, 101, 105, 107, 114, 122, 124 and 136. The archived preparation specification, static candidate inventory, revised scope and bound operation manifest preserve this sequence; Appendix F gives the archive entry point.

Let $D_m(j)$ denote supported detection and $B_+(j) = \text{pass}$ a complete base-plus pass. With n_t implementations per task, complementation is

$$G_m = \frac{1}{T} \sum_t \frac{1}{n_t} \sum_{j \in t} \mathbf{1}\{B_+(j) = \text{pass}\}\, D_m(j). \tag{2}$$

Choice rules average $D_m(j)$ over frozen build probabilities. Fixing $n_t = 6$ and $T = 20$ keeps all 120 targets, including suite failures and unknowns.

The prospective study fixes 20 known HumanEval tasks, six unused CodeLlama targets per task, the three C32 builds, Boundary32, SuiteGreedy32 and choice rules before target outcomes. HE22 remains a recorded structural omission from the earlier 21-task universe. Semantic cards cover 3,097 of 3,200 fixed method slots; 63 are unsupported and 40 historical missing/invalid. Cards were fixed before outcomes through AI-assisted specification analysis and checks, not external human adjudication.

All 1,264 operations have receipts: four canaries, 40 reference reproductions, 100 semantic validations, 160 canonical controls and 960 target operations. Original tests pass 47 and reject 73; complete base-plus passes 40, rejects 77 and leaves three unknown. All canonical B0/base/plus controls pass. Of 100 canonical method packages, 97 are supported PASS and three UNKNOWN, with no definite supported rejection. These canonical controls do not represent all correct programs.

Table 6: Fixed-panel identification bounds, in percent. All 120 targets remain in the denominator. Macro and micro coincide because every task has six targets.

Rule	Primary beyond-suite support	Secondary all-target support
Joint C32	[0,2.22]	[61.94,64.44]
Uniform C32	[0,2.22]	[60.56,64.44]
Accepted-only C32	[0,2.22]	[60.56,64.44]
Boundary32	[0,0]	[60.83,60.83]
SuiteGreedy32	[0,2.50]	[57.50,60.83]

Paired bounds. Joint-minus-uniform/accepted-only primary increments are exactly $[0, 0]$; joint-minus-Boundary is $[0, 2.22]$ pp and joint-minus-SuiteGreedy is $[-2.50, 2.22]$ pp. Shared unknowns are completed jointly. Corresponding all-target detection contrasts are $[0, 1.39]$, $[0, 1.39]$, $[1.11, 3.61]$ and $[1.11, 6.94]$ pp; these secondary advantages do not replace the primary endpoint.

Upper-bound decomposition. The five definite suite passes with historical missing method slots are HE25/187, /75, /86 and HE59/30, /82. Each carries at most one-third unresolved build weight,

totaling $5/(3 \cdot 120) = 1/72$. HE25/144 and /18 have both suite uncertainty and missing method slots, totaling $1/180$. HE15/193 has execution uncertainty in both plus and C32 build 2, totaling $1/360$. Thus $1/72 + 1/180 + 1/360 = 1/45 = 2.2222\%$. HE25's affected build lacks eight historical slots and HE59's lacks one. The protocol does not retrospectively fill them.

Omission sensitivity. The post hoc data-only check deletes each of the 20 tasks in turn, keeping equal task weights on the remaining 114 targets. Omitting HE124 gives joint-minus-Boundary all-target detection $[-1/171, 7/342]$, or $[-0.5848, +2.0468]$ pp; joint-minus-uniform is $[0, 0]$. This sensitivity check does not alter the primary 120-target denominator, reclassify unknowns, or constitute a new prospective test.

Conditional support and concrete witnesses. The original-pass/base-plus-reject set comprises three HE82 targets and HE124/130. Joint C detects the date-width violation with the selected-build probabilities in the frozen rule, while Boundary32 detects HE82/10's composite-length failure missed by all three C builds. Both detect the other two HE82 targets. Joint retention is 83.33% macro and 75% micro; Boundary retention is 50% and 75%. Joint's macro advantage over uniform/accepted-only remains $[0, 16.67]$ pp. All four targets have a supported witness in the measured-method union, which is not an independently exhaustive semantic truth set.

D COMPRESSION SUPPORTS, EXACT REFERENCE, AND REPEATABILITY

Development support. The compression panel contains 24 tasks, 87 original-test passes and 96 construction implementations. The complete-suite outcomes are 62 passes, 18 rejections and seven unknowns. The 18 rejections span seven tasks. C8 detects nine, M+8 two, D8 one, and both suite subsets fourteen. C8 and both subsets each have $5/7$ task-macro retention. Omitting any single support task leaves their macro retention between 66.67% and 83.33%. These are finite-panel sensitivities and remain below the 95% target; they do not prove a population-level impossibility of compression.

HE22 depends on the distinction between the original reference's `isinstance(x,int)` and the expanded reference's `type(x)==int`. It remains in the primary suite-policy analysis. Reducing its weight can change rankings; omitting it raises C's macro retention to 83.33%. All 120 development package-prefix reference checks pass, establishing compatibility at those inputs without settling intended bool/int semantics. Only 30 of Greedy8's 192 slots add construction coverage; the remaining 162 follow saturation. Its 78/96 constructor detections exceed Random8's 66/96, but their target counts agree at 14/18.

Producer comparison. The independently selected archival producer is CodeLlama-7B at temperature 0.8. Filtering checks signatures, prior exposure, normalized AST overlap and near-clone components. Supply is eight candidates per task except two on HE22. All 186 original-test screens complete before selecting at most the first two definite passes per task, giving 39 targets over 21 tasks. Complete base-plus results are 32 passes, six rejections on four tasks and one unknown. All 63 package-reference checks pass. C8 detects 5/6, Random8 3/6 and Greedy8 4/6. All positive C-minus-control task contrasts occur on HE124. The four-task bootstrap intervals reported in the main text are verified by enumerating all 4^4 resamples.

An archived semantic spot-check flags C8's HE124/123 witness `04-012-2003` as format-policy-sensitive: the expanded reference returns `False`, whereas the original reference parses the day as 12 and returns `True`. C8 and Greedy8 reject this target; Random8 passes. The reported retention counts remain suite-policy comparisons, not independently certified semantic fault estimates. This case is distinct from HE124/145's normal-format year-range error.

Withholding format-sensitive credit. A new post hoc sensitivity retains the same six targets and four tasks but gives every method zero detection credit on HE124/123. C8 then retains 4/6 targets and both subsets 3/6; macro retention is 62.5% versus 50% for each subset, a +12.5-pp contrast. The positive comparison with both controls therefore remains without credit on that target. This is a diagnostic scoring stress test, not a semantic relabeling, independent certification of the other targets, or a population confidence bound. `verify_format_sensitivity.py` computes it directly from the frozen target-level outcomes.

Only HE122 and HE124 have nonempty rejected-target support in both panels. On these common tasks, C-minus-Random macro retention changes from 0 to +50 pp, while C-minus-Greedy decreases from +50 to +25 pp. Yet the latter whole-panel contrast rises from 0 to +12.5 pp. Thus changes in support can reverse the apparent direction of improvement. An archived symmetric decomposition separates within-common-task differences, common-task reweighting and entry/exit of support; it is descriptive because target implementations also differ.

Exact random-subset reference. If a fixed target fails K of N input IDs, a uniform sample of eight distinct IDs detects it with probability

$$p_8(N,K) = 1 - \frac{\binom{N-K}{8}}{\binom{N}{8}}. \tag{3}$$

The full-matrix six-target support has no unknown cells. IDs with equal arguments remain distinct sampling units. For each task, group IDs by their joint target-failure mask. Sampling eight without replacement induces a multivariate hypergeometric law; aggregating masks and convolving independent task draws yields the joint detection-count and macro-retention distributions. The expected count is 2.8776/6 and expected macro retention 49.67%. The tail at five detections is 3.28%, while the tail at

75% macro is 8.63%. These are finite-pool sampling probabilities conditional on observed outcomes, not inferential evidence that a learned method has a corresponding population error probability.

Table 7: Conditional checks of the local compression result. These populations are not independent additions to one common denominator.

Check	Observed result	What it addresses
Exact suite-sampling reference	Expected 2.8776/6 detections and 49.67% macro; probability of at least C's five detections 3.28%.	A random design over the observed input pool.
Three C8 builds; sparse support 3 targets / 2 tasks	Macro detection 100%, 0%, 50%; mean 50%. Three boundary packages each reach 50%.	Repeatability on a small selected support.
32-slot depth; non-date supported set 5 / 4	C builds reach 75%, 50%, 75%; Boundary32 reaches 100%.	More inputs and a specification boundary control.

Repetition and depth. The remaining 27 of 66 original-test passes are retained before new base-plus outcomes; they yield 21 passes, five policy rejections and one unknown. A fixed sparse-witness rule requires an unknown-free complete matrix, a supported witness and at most 1% failing input IDs. Three targets qualify: two near-clone HE0 distance-comparison errors and one HE18 case-fold error, only two independent non-date families. This does not meet the planned three-family supply condition. C8 macro values are 100%, 0%, 50%; each of three boundary packages obtains 50%, and existing suite Greedy8 obtains 100%.

The subsequent depth analysis uses already exposed targets. Its non-date supported population has five targets on four tasks; C32 builds score 75%, 50%, 75%, versus Boundary32's 100%. All C32 builds miss HE54/198. Adding C inputs to an eight-input construction-coverage selector yields 75%, matching the previous Greedy8; its increment over the equally selected suite-plus-boundary control is $[0, 25]$ pp, with raw policy decisions identical on all 66 targets. The difference is semantic certainty on a boundary input, not an additional observed rejection. No new policy rejection appears among 53 complete-suite-passing targets; eight C32 cells remain unknown from missing/invalid slots.

Construction coverage and missing fault information. The broader lesson is about the source of retention: a package needs inputs for the faults it will encounter. Greater coverage of a small construction pool does not guarantee those inputs. In the development comparison, Greedy8 raises constructor coverage from 66/96 to 78/96 without changing its 14/18 target detections. In the later HE54 diagnostic, all nine available qualified constructors pass inputs that expose the target error. These finite-pool examples explain a limitation of the observed feedback, without showing that specifications, source text or better construction populations cannot supply the missing information.

E EXPLORATORY FOLLOW-UPS ACROSS GENERATOR SERVING ROUTES

E.1 PROTOCOL, TIMING AND REPORTABLE ANALYSIS

The follow-ups reuse the fixed 20-task, 120-target panel from Appendix C; they add generator builds, not tasks or target implementations. Each model has three planned paired builds per task. A pair shares an initial proposal, four construction programs, the task specification, public examples and a common donor/reference witness. The initial generator supplies eight construction inputs. One continuation receives their program/reference execution outcomes; the other is asked to reconsider the same proposal from the specification and construction sources without those new outcomes. Thus "no new feedback" still includes the shared witness and source code, not a feedback-free information state.

Each continuation has one model request and a 4,500-token output cap. The generator must be deterministic in its seed. Thirty-one fresh seeded inputs follow the common witness in a frozen 32-slot sequence; the prespecified follow-up endpoint uses its first eight slots. The two arms share deployment seeds and are dispatched in a fixed hash-based order. An unavailable response, malformed generator or generator with no executable output is retained as an unavailable package, not replaced by the shared witness. Invalid and repeated positions consume slots without replacement. Unlike the original C protocol, failed follow-up arms receive no mutation fallback; these results must not be pooled with original C scores.

Grok requests use `grok-4.6` and successful replies report `grok-4.6-build`. Terra and Sol replies report `gpt-5.6-terra` and `gpt-5.6-sol`, with medium reasoning. These are recorded serving identities; immutable upstream weight revisions are not attested. Terra and Sol use the same paired prompts, construction material, seeds, output limits and target panel. Grok shares those evaluation objects but uses different serving/request semantics. Historical Luna eight-slot prefixes have a different construction protocol. The experiment therefore does not support a controlled four-model ranking or independent-family generalization.

Staged exploration. Grok generation was interrupted by service failures and resumed for precommitted, unissued cells. Some target outcomes had been observed before the later phase; later generator prompts still excluded them. Six failed requests were not retried. Terra/Sol first ran a four-task, two-build pilot on HE18, HE54, HE105 and HE122. The full-panel extension was authorized after these pilot results were observed: each model reuses eight pilot pairs and adds 52, giving 16 reused and 104 new pairs in total. All newly generated Terra/Sol packages were sealed before their new target scoring. The old pilot and historical target results were already exposed. Neither follow-up is described as prospective confirmation, and the pilot is not counted as an independent replication of the extension.

Oracle alignment and failure accounting. A credited detection requires both an actual rejection by the frozen comparator and specification-supported error semantics. A comparator match is not a detection even if an exact-value semantic check disagrees within numerical tolerance. The initial Grok analysis omitted this conjunction; only the corrected oracle-aligned version is reported here, consistently for both arms and historical controls. Superseded outputs remain archived and do not change the original manuscript's scores. Before new Terra/Sol target scoring, a syntax-invalid Terra HE74 repeat-2 no-new-feedback package exposed a separate qualification issue: a failed generator could inherit the common witness. The derived scoring view enforces the original failure rule by treating that entire package as UNKNOWN. Its source and input sequence were not repaired or resampled. The gate affects no earlier Grok or pilot package.

Table 8: Exploratory eight-slot generator follow-ups on the same 20 tasks and 120 targets. Each arm has 60 planned packages and 360 target–build cells. Detection/complementation are percentages; Δ is feedback minus no-new-feedback in percentage points. Brackets are conservative finite-panel identification bounds, not confidence intervals. Terra/Sol use the matched continuation protocol; Grok is archived context with different serving semantics.

Route	Arm (usable/60)	Supported detection	Beyond-suite support	Detection Δ (pp)
Grok	Feedback (59)	[59.72, 61.67]	[0.00, 1.11]	[-2.78, +3.33]
Grok	No new feedback (58)	[58.33, 62.50]	[0.00, 1.39]	—
Terra	Feedback (60)	[55.28, 55.56]	[0.00, 0.28]	[-6.94, -1.94]
Terra	No new feedback (59)	[57.50, 62.22]	[0.00, 3.33]	—
Sol	Feedback (60)	[58.89, 59.17]	[0.00, 0.00]	[-1.39, -0.56]
Sol	No new feedback (60)	[59.72, 60.28]	[0.00, 0.28]	—

E.2 Denominators, uncertainty and per-task results

Let $[l_{tbr}, u_{tbr}]$ denote the eight-slot supported-detection bounds for task t, build b and target r. A definite supported comparator rejection gives $[1, 1]$; an unresolved observation or unavailable package without a definite detection gives $[0, 1]$; otherwise the endpoint is $[0, 0]$. Averaging over $20 \cdot 3 \cdot 6 = 360$ scheduled cells gives each arm's bounds $[L_a, U_a]$. The reported difference uses the conservative enclosure $[L_F - U_N, U_F - L_N]$. It is not a statistical confidence interval or a claim that every endpoint is attainable under all cross-arm semantic dependencies. For complementation, complete-suite rejects contribute zero, definite passes use the method's detection bound, and unresolved suite status contributes $[0, u_{tbr}]$.

All arms have zero confirmed native-base-plus complementation. Grok's detection comparison includes zero; Terra and Sol have negative finite-panel bounds. These outcomes show no consistent benefit from this particular feedback continuation on the observed panel. They do not quantify a population-level effect of feedback, and their widths do not measure variation across new tasks or new model snapshots. The per-task counts below expose where the aggregate differences arise.

Table 9: Per-task detection counts as lower–upper numerators out of 18 planned target–build cells per arm. F: feedback; N: no new feedback. A range preserves uncertain labels or unavailable packages. Each column sums to its 360-cell aggregate; tasks, rather than cells, delimit generalization.

HumanEval	Grok F	Grok N	Terra F	Terra N	Sol F	Sol N
0	9	7	8	8	7	7
15	6	6	6–7	6–15	6	6–7
18	6	6	6	6	6	6
24	12	12	12	12	12	12
25	3	3	3	3	3	3
54	15	15	14	15	15	15
59	12	12	10	9	11	12
71	15	10–16	15	15	15	15
74	7	8	6	5–11	6	6
78	9–10	8–11	9	9–11	8–9	9–10
79	4–10	4–10	6	6	6	6
82	12	13	5	11	9	11
87	18	18	18	18	18	18
101	6	6	6	6	6	6
105	15	15	14	15	15	15
107	12	12	12	12	12	12
114	15	15	15	15	15	15
122	18	18	18	18	18	18
124	15	16	10	12	18	17
136	6	6	6	6	6	6
Total / 360	215–222	210–225	199–200	207–224	212–213	215–217

Historical context. On the same targets, uniformly averaged historical Luna C8 prefixes have supported-detection bounds $[57.50, 62.78]\%$, Boundary8 gives 58.33%, and SuiteGreedy8 gives $[54.17, 57.50]\%$. These are existing packages under the aligned scorer, not new matched Luna experiments. Short historical sequences retain missing slots as unknown. Baseline overlap remains explicit: all-target detections include faults already rejected by the complete suite.

Requests and cost scope. The Grok campaign records 218 requests, 212 successful replies and 1,887,538 reported tokens; this includes semantic-audit work and qualification, so it is not a like-for-like generation-cost denominator. Usage for six failed requests is unknown. Each Terra/Sol campaign has 182 successful requests including qualification: 25 in the pilot and 157 in the extension. Cumulative reported tokens are 844,970 for Terra and 1,025,747 for Sol; the newly generated extension alone accounts for 1,621,998 tokens combined. Equal request caps and scheduled input counts do not equate actual compute, latency or total construction cost.

E.3 A LOCAL COVERAGE LOSS AND SEMANTIC SENSITIVITIES

Prime-length coverage after feedback. HE82 asks whether a string's length is prime. In Terra repeats 1 and 2, all eight feedback-arm inputs have length five. The paired no-new-feedback sequences also reach lengths such as 2, 4, 6, 13, 15, 67 and 81. Across those two builds, feedback loses six target–build detections, contributing $-6/360 = -1.67$ pp to the full-panel contrast. One saved witness is `aa`: length two is prime, the reference returns `True`, and the target returns `False`. This is an inspectable loss of deployment coverage after a particular revision, not proof that feedback generally narrows useful exploration.

Terra has strictly lower feedback bounds on four tasks and higher bounds on one; Sol has lower bounds on two and higher on one. In a post hoc leave-HE82-out check, the remaining 19-task contrasts are $[-5.56, -0.29]$ pp for Terra and $[-0.88, 0.00]$ pp for Sol. Sol's strictly negative full-panel bound thus depends on HE82. The omission analysis does not replace the fixed-panel result or turn its other 19 tasks into a new confirmatory sample. Data-only arithmetic checks corroborate the eight archived cross-arm HE82 discrepancy witnesses; they do not constitute independent human certification.

Date-format dependence. The primary semantic card interprets `mm-dd-yyyy` as fixed-width month and day fields. The HE124/130 witness in Table 3 depends on that width rule. A separate post hoc audit identifies six Grok/historical-prefix slot observations on four targets whose error credit depends only on zero padding; withholding certainty for these observations is a semantic-policy sensitivity, not evidence that the relaxed interpretation is correct. The 32-slot omission result in Figure 3 remains the reported original sensitivity. Date errors involving an unsupported year restriction, such as the `12-09-8842` witness in Section 7.1, are distinct from the padding dispute.

Output-type sensitivity is not a generated-input increment. A post hoc audit of the 8,694 archived survivor slots finds 72 literal return-annotation discrepancies on two implementations: 30 HE4 observations return an integer where `float` is annotated, and 42 HE25 observations return floating-point elements where `List[int]` is annotated. All flagged values equal the archived references numerically. Allowing integer-to-float compatibility removes the former; the latter is an annotation-level distinction, not a primary value-level detection. Missing annotations and unavailable observations remain unassessed.

For HE25, the original suite already contains input 57, expecting `[3,19]`; the archived target returns `[3,19.0]`. Ordinary value equality passes, while an element-type check rejects. Changing the oracle on an existing suite input can therefore expose this discrepancy without generating a new input. Any type-sensitive comparison must apply the same oracle to the complete-suite control before attributing additional value to generated tests. We retain the original primary labels. This selected-panel sensitivity strengthens the requirement to separate input-generation value from comparator changes.

Replay boundary. `verify_followups.py` replays three-model endpoints, per-task tables, pilot reuse and HE82 omission contrasts from frozen slot labels, comparator decisions, package availability and target IDs. `data/FOLLOWUP_SOURCE_MAP.json` records source hashes. Replay verifies numerical consistency; it neither certifies semantic labels nor reruns models or programs.

F REPRODUCTION MAP AND RETAINED EVIDENCE

This appendix records the minimal entry points for checking the paper's claims without reproducing the entire exploration history. Internal study identifiers name archive packages, not additional independent experiments. All source paths below are relative to the research repository; the accompanying presentation package contains copied numerical exports with SHA256 provenance in `data/SOURCE_MAP.json`.

Table 10: Claim-to-artifact map. The source package provides numerical presentation inputs; primary archives supply execution and analysis context.

Claim	Primary archive entry point
Original C scores, selection, ties and acceptance decomposition	`output/deployment-selection-20260914/RESULT.json`; acceptance-ablation and compact records in the same archive.
Development residual survivor and compression comparison	`output/core-audit-20260911/data/CORE_DATA.json`; original audit paths are retained in the deployment exports.
Fixed 120-target prospective result	`output/prospective-confirmation-20260914/recovery/execution/RESULT.json`; raw results and complete reproduction package alongside.
Task omission and upper-bound sources	`output/finalization-20260915/OMISSION_SENSITIVITY.json` and `UNCERTAINTY_DECOMPOSITION.json` in that directory.
Exact subset and producer support	`output/deployment-contribution-20260912/t107-results/RESULT.json`; exact reference in `output/transfer-attribution-20260912/RESULT.json`.
Repetition and depth checks	`output/sparse-witness-milestone-20260913/RESULT.json` and `output/portfolio-mechanism-20260913/RESULT.json`.

Numerical presentation reproduction. From the extracted manuscript directory, run `python3 generate_figures.py` with NumPy and Matplotlib installed, then `latexmk -pdf manuscript.tex`. The three figures are vector PDFs with PNG previews; all study values are read from frozen exports or exact fractions stated in the source. `verify_numbers.py` checks the main selection result, acceptance decomposition, prospective endpoint, paired zero increment, omitted-task sensitivity and upper-bound arithmetic. The provided data map identifies the upstream artifact for each copied input. This checker does not execute programs, rerun model inference, or replace the original study analyzers.

Full execution reproduction. The prospective archive's `T119_complete_reproduction.zip` includes frozen analysis, raw execution records and expected output. Construction programs, containers and operation records are needed for full execution reproduction; receipt completeness alone does not establish complete input sequences or resolved semantic labels.

Information boundaries and excluded inferences. Construction programs, generated artifact hashes, reference versions, target sources and semantic rules are linked in the execution archive. The original scorer's representation limitations remain disclosed; later semantic replays do not silently rewrite its scores. Full-suite passing is a finite test result, not a proof of correctness. Specification-assisted boundary controls have a different construction process and cost basis from C. Neither a high construction score nor a valid canonical run is evidence that every correct implementation will pass.

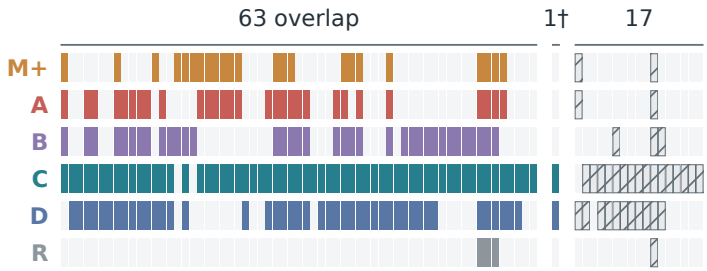
The follow-up projection and hashes are in `data/followup_observations.json` and `data/FOLLOWUP_SOURCE_MAP.json`; `verify_followups.py` provides integer-count replay. This revision adds a data-only format-credit sensitivity, with no new model or target execution; historical records remain unchanged.

G ADDITIONAL VISUAL EVIDENCE

These figures restore the program- and task-level views behind the main summaries. They use the original frozen populations and retain their comparator, uncertainty and weighting qualifications. No new data are introduced.

(a) Every selected candidate, by configuration

One column = one target; any of 3 runs, up to 32 slots



	B+ reject	Spec. error	Uncon-firmed
M+	23	0	2
A	29	0	2
B	36	0	3
C	61	**1**	16
D	44	**1**	11
R	3	0	1

Rows overlap; hatched targets remain unconfirmed.

(b) Exhaustive semantic observation on all 18 survivors

8,694 target slots; 1,711 recovered suffix slots

HE task (targets; slots)	Value compatible	Repr. only	Unicode unknown	Invalid input	Runtime unknown	Spec. error
4 (1; 483)	453	**30**	0	0	0	0
12 (1; 483)	477	0	0	0	0	**6**
16 (4; 1,932)	1,664	0	**268**	0	0	0
25 (1; 483)	433	**42**	0	**8**	0	0
27 (4; 1,932)	1,460	0	**472**	0	0	0
60 (1; 483)	0	**483**	0	0	0	0
66 (1; 483)	296	0	**187**	0	0	0
147 (4; 1,932)	1,831	0	0	0	**101**	0
161 (1; 483)	460	0	**23**	0	0	0
Total	**7,074**	**555**	**950**	**8**	**101**	**6**

† HE12: C first at run 0 / slot 11; D at run 2 / slot 5.
Six supported slots share one target and the input [""].

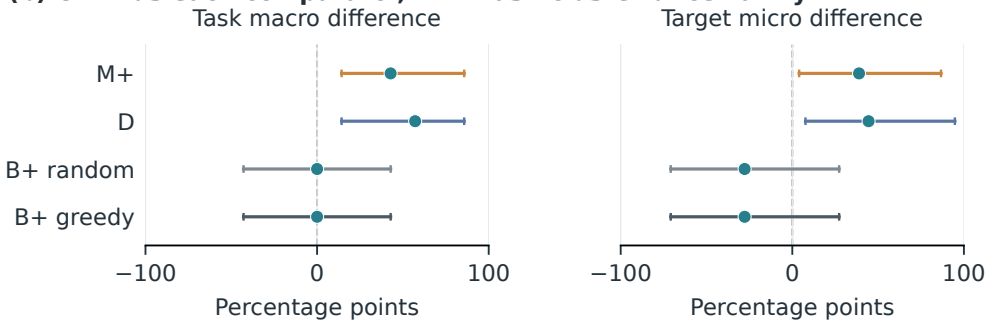
Figure 6: All 81 mismatch-selected development targets and all 8,694 frozen slots on the 18 complete-suite survivors. Rows in the upper panel overlap. The six specification-supported slots in the lower panel refer to one HE12 target and one input, shared by C and D. Hatching retains uncertainty; unconfirmed targets are not certified correct or declared false positives. The prospective panel is separate.

(a) Retaining complete-suite detections with 8 slots
Separate panel: 24 tasks / 87 targets; U = 18 targets in 7 tasks

	0	22†	59	101	122	123	124	Task macro	Target micro
M+	1/1	0/6	0/1	0/1	0/5	0/3	1/1	28.6	11.1
C	1/1	0/6	1/1	1/1	5/5	0/3	1/1	71.4	50.0
D	0/1	0/6	1/1	0/1	0/5	0/3	0/1	14.3	5.6
B+ random	0/1	6/6	1/1	1/1	5/5	0/3	1/1	71.4	77.8
B+ greedy	1/1	6/6	1/1	1/1	5/5	0/3	0/1	71.4	77.8
Full B+	1/1	6/6	1/1	1/1	5/5	3/3	1/1	100.0	100.0

Cells: detected / U targets. Aggregate retention in %.
† HE22 depends on the frozen bool/int oracle policy.

(b) C minus each comparator, with task-cluster uncertainty



95% exact empirical bootstrap intervals; 7 tasks resampled jointly.
Equal macro means do not establish equivalence or 95% retention.

Figure 7: Development compression on 18 suite-rejected targets across seven nonempty tasks. Cells show detected/available targets; macro and micro weightings differ. The dagger marks policy-sensitive HE22. Paired differences use exact enumeration of the empirical seven-task cluster bootstrap. Equal macro estimates establish neither equivalence nor the fixed 95% retention criterion.

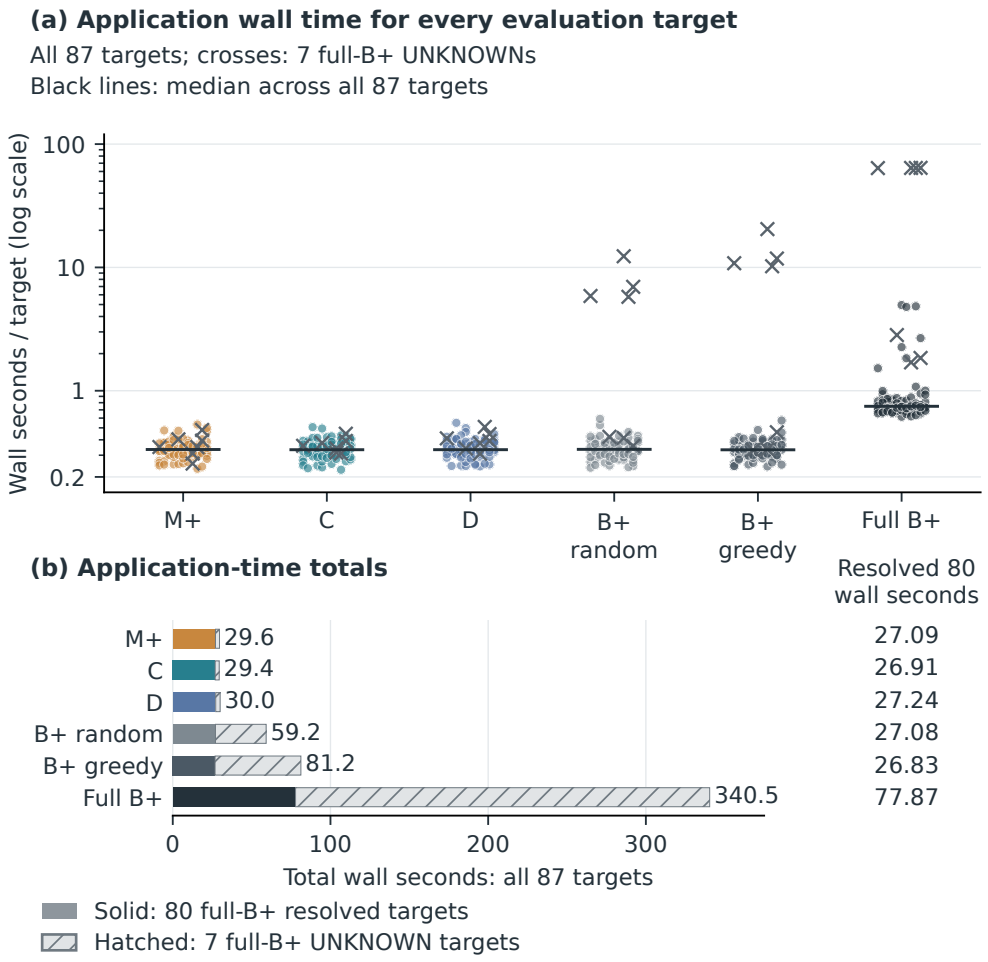


Figure 8: All 87 development-compression targets remain in the application-time plot. Crosses mark the same seven full-suite UNKNOWN targets. Total wall time is separated by that fixed status; the other 80 targets are also reported. Full B+ uses separate base and plus groups; small packages use one. Sandbox overhead is included, while construction wall time and timing-repeat uncertainty are unavailable. These observations do not establish an end-to-end speedup.