Alibi Distance: measuring and closing the silent-failure aperture of a verified exercise corpus

DeepForge Research

September 18, 2026 · alibi-distance

Preprint · Artifact · Not peer reviewed

Abstract

Auto-graded exercises ship a reference program and a handful of tests. A program can pass every shipped test and still be wrong. We define the alibi distance alpha(p) as the minimum number of single-edit mutations of the reference that yields a program passing all shipped tests while diverging from the reference on a deterministic probe bank. Mining the DeepForge corpus (5,730 exercises; 106,081 sampled mutants; 7,727 silent survivors) shows that 46.1% of analyzable problems admit a radius-1 alibi. A second, separately written CPython engine, run by this paper's author over the same corpus, agrees closely but not exactly: within Table 7 the census-rate rows differ by at most 0.14 percentage points (16.50% against 16.57% test-passing, 46.08% against 45.94% affected), the length-quartile aperture rows by at most 0.32 (56.66% against 56.34%), and the largest single discrepancy anywhere in that table is 0.44 (cross-validated held-out kill, 48.35% against 48.79%). Neither engine, nor any raw mining artifact, is part of this release, so that agreement is an internal provenance record rather than an external check, and the census figures below are not independently reproducible by an outside reader. A witness chosen on the full mined alibi set (sampled mutants over the fixed probe bank) closes every mined alibi in 59.7% of affected problems and kills 73.2% of them, but a cross-validated witness (chosen on half the alibis) kills only 48.4%; the deployable, oracle-free estimate is therefore ~23.5% of problems still affected, not the oracle-chosen 18.6%. Two negative results: probe selection policies do not transfer better than random (63.3% vs 59.5% conviction within ten probes), and <=2-literal input predicates reach 0.9 balanced accuracy for only 32.2% of alibis against a permutation control of 0.881. A blind audit of ten alibis sampled independently of the author's first ten, judged by this paper's author without sight of the first audit's verdicts, found 10/10 plainly wrong. We frame alpha as a property of the corpus and the edit model, not of learners: mutation–slip coupling is assumed rather than measured in this corpus, and a shadow-logging study is proposed. No grading path was modified; the shipped artifact is a re-mined set of 96 verified silent-bug practice puzzles that resist an uncapped union lazy suite and an independently constructed held-out suite.

Keywords: mutation testing, test adequacy, exercise grading, mutation augmentation, counterexample, Python, education

1. Introduction & motivation

DeepForge is a practice platform whose catalog is machine-verified: 5,730 Python exercises, each with a reference solution that provably passes its shipped tests under real execution, a 1e-6 deep-equality harness, and a per-call time budget. That verification is a claim about the catalog, not about the tests. A shipped suite has three to six cases, so passing it is a finite condition. A program that agrees with the reference on every shipped case and disagrees somewhere else is wrong in the only sense a grader can see, and the grader cannot see it at all. We call such a program an alibi: it was at the scene, but the evidence cannot convict it.

This paper measures how close the nearest alibi sits. The alibi distance alpha(p) is the minimum number of single-edit mutations of the reference that produces a program passing every shipped test and diverging from the reference somewhere in a deterministic, per-problem probe bank. alpha = 1 means a one-line slip passes the entire suite and is behaviorally wrong somewhere the suite never looks. This is a radius/weakest-link framing: it asks not how many faults a suite kills on average, but how far the nearest silent nearest-miss is.

The measurement is a build-time operation. It needs the reference, the tests, real Python execution, and a fixed seed; it needs no network, no model, and no telemetry about learners. That is a deliberate fit for a server-free, LLM-free product: the same resources that make the catalog verifiable make its aperture measurable. The measurement reported here was constructed in DeepForge research wave 41 and was run over the whole corpus by two separately written engines, which is why almost every number below arrives twice. The components it uses — single-edit mutants, the surviving-mutant set, a witness plus its reference output, greedy set cover over witnesses, and ordinary cross-validation — are standard mutation-analysis and differential-testing practice; the assembled procedure and the minimum-radius aggregation are what this paper describes.

The paper is candid about what is new and what is not. Closing a suite against a surviving mutant is textbook mutation-test augmentation; the witness plus the reference output is the generated test, and no generation step is needed. What we contribute is: (i) a census of radius-1 silent survivors over this platform's own corpus of 5,730 verified exercises, computed with real execution and cross-checked by a second, separately written engine — not a priority claim, because survival of CS autograding suites has been measured in the published literature for over a decade and the aggregation, not the phenomenon, is the difference; (ii) the radius/weakest-link metric as a corpus statistic rather than a per-program adequacy score; (iii) two clean negative results that stop a plausible next step before it is built; and (iv) two audits of ten alibis each, one sampled blind and one not, that give a first estimate of how often the mined objects are defects, with a wide interval at that sample size. The contributions in full:

- Census: 106,081 sampled mutants over 5,730 exercises; 17,502 pass all shipped tests (16.50%); 7,727 of those diverge in the probe bank; 46.08% of analyzable problems admit a radius-1 alibi. These are records of a mining run whose programs are not in the release (Section 9), so they are not independently reproducible from a checkout.
- Second engine: a from-scratch engine reports 105,425 mutants, 16.57% pass, 45.94% affected, 73.25% best-probe kill; across Table 7 the census-rate rows agree to 0.14 percentage points, the length-quartile rows to 0.32, and the largest single gap is 0.44 on cross-validated held-out kill. Both engines are by this paper's author and neither is shipped, so this is an internal agreement record rather than external replication.
- Closure accounting: one oracle-chosen witness kills 73.22% of alibis and fully closes 59.71% of affected problems; a cross-validated witness kills 48.35% of held-out alibis, making the deployable estimate 23.5-23.8% still affected, not the in-sample 18.6%.
- Negative result: feature-guided probe policies transfer no better than random (63.3% random vs 59.5% guided conviction within ten probes).
- Negative result: best <=2-literal predicates reach 0.9 balanced accuracy for only 32.2% of alibis, against a permutation control of 0.881.
- Audits: a blind audit of ten alibis sampled independently of the author's first ten found 10/10 plainly

wrong, judged by this paper's author; a separate non-blind author audit found 8 wrong, 1 immaterial, 1 subtle. Neither audit record is committed.

- Artifact: a frozen, machine-verified set of 96 mined alibis that resists an uncapped union lazy suite and an independently constructed held-out suite, shipped as the Silent Bug Hunt practice route, with no grading, review, or certificate integration.

> *Scope — alpha is relative to the edit model E, the probe domain D, and the sampled mutant family M. All closure statements read as statements about the mined sample over a fixed (D, M), never about all programs one edit from the reference. The census is a property of the corpus and the edit model; it is not a claim about the rate at which learners write silently-passing code. Mutation-slip coupling is assumed rather than measured in this corpus, and Section 10 proposes the study that would measure it.*

2. Related work

Mutation testing is the parent technique. DeMillo, Lipton, and Sayward framed test data selection around deliberately seeded simple faults; Jia and Harman surveyed the field's first decades; Papadakis et al. surveyed the following decade and the equivalent-mutant problem that dominates its practice. An alibi is a surviving mutant under a fixed operator family and a deterministic execution probe; what differs here is that the survival set is used to build a minimum-distance statistic over an exercise corpus rather than a mutation score for program p under test.

The closest published system is STING (2026), which generates surviving variants of a reference patch on SWE-bench and synthesizes targeted regression tests to kill them, reporting that 77% of instances admit at least one surviving variant. The goal is the same as witness closure here; the mechanism differs in two ways. STING generates tests with a model and validates them against behavior-preserving transformations, while our closure appends the witness input and the reference's own output directly, so the added test cannot reject the reference and no test synthesis is involved. STING targets agent benchmarks; this paper targets a verified education corpus and reports the closure's generalization honestly, including its loss. Its aperture and ours are not comparable and should not be lined up: its 77% is over SWE-bench Verified instances under LLM-based mutation, while 46.08% is over exercises under eleven fixed operator families with no generated variants, and the denominators are different objects. The ordering of the two numbers says nothing about either study.

Education-side work shares the motivating phenomenon but inverts the object. Examplar and conceptual mutation testing (Prasad, Greenman, Nelson, and Krishnamurthi, 2024) check whether student-authored examples pin a problem down using curated chaff mutants; our mining checks whether the platform-shipped suite pins the problem down and produces a corpus metric, not a student-facing exercise. BugSpotter (Padurean, Denny, and Singla, SIGCSE 2025) and When AI Is Wrong on Purpose (2026) generate buggy code and have students produce failing tests. Their bugs fail the suite by construction and are findable; an alibi passes the suite, which is exactly the silent case a finite test set can miss. FPPgen (Caraco, Lojo, Verdicchio, and Fox, 2024) and Katabench both ship mutant-as-practice: students write tests graded by whether planted faults are caught. Those products are practice surfaces; neither reports a corpus census or a minimum-distance measure.

On the testing side, Just et al. (2014) is the strongest available evidence that mutants correlate with real faults, which is the coupling assumption this paper keeps at arm's length: their result is about fault detection correlation in Java programs with real fault data, while this corpus has no human false-pass data at all. Inozemtseva and Holmes (2014) showed coverage is a weak proxy for suite effectiveness, which is the same warning this paper's mutant-count-vs-length confound echoes in miniature. DSpot (Danglot et al., 2019) amplifies developer-written tests, and differential testing (McKeeman, 1998) runs two comparable programs against the same inputs and treats divergence as the oracle. The alibi duel in the shipped practice route is differential testing in miniature: reference and ghost run locally, and the divergence fact is the verdict.

Delta debugging (Zeller and Hildebrandt, 2002) minimizes a known-failing input; there, an input is known to fail, while an alibi witness exists precisely because no shipped input fails. The minimal pair in linguistics is the conceptual ancestor of the witness, and machine teaching (Goldman and Kearns, 1995; Zhu, 2015) asks which examples identify a concept, but that literature concerns example selection for a learner rather than measuring the aperture of a shipped suite. The engineering position this paper takes — act on the individual surviving mutant rather than on an aggregate score threshold — is also the position Section 5 reaches independently, and for a different reason: the score's apparent strength is partly in-sample, which is a statement about the estimator and not about the aggregate. Finally, near-miss exposure is a plausible practice object under the desirable-difficulties account (Bjork and Bjork, 2020), but this paper makes no learning-outcome claim.

Source	Taken	Not claimed
Mutation testing	Surviving single-edit mutants as the fault model; equivalent-mutant caveat	A mutation score for a program under test
STING 2026	Witness closure is the same goal as surviving-variant augmentation	Test synthesis, LLM generation, agent benchmarks
Examplar 2024	Chaff mutants as a way to pin a problem statement down	Student-authored examples as the object of measurement
BugSpotter / AI Wrong on Purpose	Near-miss buggy code as a practice object	Their bugs fail the suite; alibis pass it
DSpot / differential testing	Divergence between comparable programs as the oracle	Automatic repair or generation of new behavior
Delta debugging / minimal pairs	The witness as the smallest revealing input	A known-failing input: shipped tests all pass

What alibi mining takes and what it does not claim

3. Method

A problem is a triple p = (S, f, T): a natural-language statement S, a partial reference function f in Python, and a shipped test set T = {(x_i, y_i)} with y_i = f(x_i) and, in this corpus, three to six cases. The platform supplies two step functions: eval_f(x), which runs f under the harness semantics (1e-6 relative-or-absolute deep equality, a 0.25 s per-call alarm, first def as entry point), and pass(g, T), which runs g against every shipped case. Nothing about S is machine-readable; that fact bounds

everything below.

The edit model E is a set of AST rewrites, each applied once. The census used eleven families. One direction applies per family for cmp, bool, bin, aug, func, notins, notdel, and negdel; int, idx, and lenshift are two-directional per target node. The number of concrete rewrites for a given problem therefore depends on the reference AST, and no fixed rewrite count is claimed. Distance d(f, g) is the minimum number of family applications taking f to g; it is slip distance, the size of the human error class being modeled, not semantic distance.

Family	Rewrite
cmp	< <-> <=, > <-> >=, == <-> !=
bool	and <-> or
bin	+ <-> -, // <-> /, % -> //, * -> +
aug	+= <-> -=, *= -> +=
int	integer constant n -> n +- 1
func	min <-> max
idx	subscript index i -> i +- 1
lenshift	len(x) -> len(x) +- 1
negdel	unary minus deletion: -x -> x
notins	wrap a comparison in not (...)
notdel	delete a not

The eleven single-edit families. One direction per family for cmp, bool, bin, aug, func, notins, notdel, negdel; two directions per target node for int, idx, lenshift. The concrete rewrite count per problem depends on the reference AST, so the operator set is described by family, not by a fixed number of rewrites.

The probe domain D(p) is deterministic and per-problem. The first three shipped inputs are perturbed one argument at a time: empty, singleton, reversed, duplicated, extended, shrunk, negated, sorted, and incremented variants; matrices are also transposed and row-perturbed. Duplicates are removed, shipped inputs come first, and the bank is shuffled with seed int(md5(problem_id)[:8], 16) and capped at 48 entries. Across the corpus the mean bank holds 39.05 probes, of which 89.22% are reference-evaluable: only probes on which the reference succeeds can serve as witnesses, because the comparison needs a reference output to diverge from. A 300-problem robustness rerun with the cap raised to 120 moved the aperture from 45.3% to 46.7% (4 problems gained, 0 lost), so the 48-probe cap is not what produces the headline rate.

Mining samples at most 36 single-edit mutants per problem, at most 6 per family, with the target order shuffled by seed int(md5(problem_id)[:6], 16). A control step first requires ast.unparse(ast.parse(f)) to pass T, so formatting round-trips cannot inflate survival. The pass set, the divergent programs, and the metric are then defined as follows, where D_eval is the reference-evaluable part of the bank and A1(p) is the mined radius-1 alibi set.

```
                    Alibi set and alibi distance
 Pass(p) = { g in E(f) : g compiles and pass(g, T) } ; D_eval(p) = { x
in D(p) : eval_f(x) succeeds } ; Diverge(g) = { x in D_eval : eval_g(x)
!= eval_f(x) or eval_g(x) fails/times out } ; Alibi(p) = { g in Pass(p)
```

```
: Diverge(g) != empty } ; alpha(p) = min { d(f, g) : g in Alibi(p) }
```

Equality is the harness deep-equality with rel_tol = abs_tol = 1e-6; a raise or timeout where the reference returned counts as divergence; a reference failure on a probe drops that probe. Near-miss aperture is the event alpha(p) = 1.

Alibi mining, one problem

```
# input: p = (id, f, T)
assert pass(unparse(parse(f)), T)                    # round-trip control
D  = probe_bank([x for x, _ in T], seed=int(md5(id)[:8], 16), cap=48)
O  = { i: eval_f(D[i]) for i in range(len(D)) if ok(eval_f(D[i])) }
M  = sample_mutants(f, cap=36, per_family=6, seed=int(md5(id)[:6], 16))
P  = [m for m in M if compiles(m) and pass(m, T)]       # survivors
W  = { m: [x for x in O if diverges(eval_m(x), O[x])] for m in P }
A1 = { m: W[m] for m in W if W[m] }                     # radius-1 alibis
best_probe = argmax_x |{ m : x in W[m] }|
closure = greedy_set_cover([set(W[m]) for m in A1])     # 1-closure probes
```

Cost on the real corpus: 106,081 mutants generated, 17,502 survivors, 7,727 alibis, roughly 4.5-5 million Python function calls, about 27 CPU-minutes, and about 4 minutes wall-clock on 8 CPython workers. This is a build-time content pipeline, not runtime work. Seeds are deterministic throughout: the problem id seeds the mutant target order and the probe shuffle, and the resampling and audit scripts use the fixed seeds 11, 5, 7, 99, 2026, 3, and 41.

Two scope facts must be stated before any guarantee is read. First, mining samples mutants: no statement below is about the full single-edit space, only about the sampled family M. Second, the probe bank is rebuilt whenever the shipped test set changes, so closure statements are relative to a fixed (D, M). An early draft claimed monotonicity across test sets; that claim is false and is withdrawn. The corrected statement holds only when the bank and the sample are held fixed.

```
            Corrected monotonicity (fixed D and M only)
  For T subseteq T' with D and M fixed: Alibi_{D,M}(p') subseteq
       Alibi_{D,M}(p) and alpha_{D,M}(p') >= alpha_{D,M}(p)
```

Across different test sets the statement fails because D is rebuilt from T. Counterexample, reproduced: problem la-029, mutant idx-@5, with T' = [x0] + T on a probe x0 where every sampled passer agrees, D(T') \ D(T) grows by 17 probes and the mutant passes T' while diverging at [[[4, 5, 6]], 1, 2]. There is no monotonicity across test-set changes.

> *What 1-closure guarantees — 1-closure means: no surviving alibi among the sampled mutants over the fixed probe bank. It is not a guarantee over all one-edit programs, and it does not eliminate radius-2 alibis. Section 5 measures the leakage directly: after one added witness, random two-edit programs still survive at 8.56% against 12.52% before.*

4. Census results

The author engine analyzed 5,721 of the 5,730 problems: five have fewer than five reference-evaluable probes and four raised during mining (ds-038, ds-235, ds-236, ds-237). The second engine analyzed 5,725: it handled the four raising problems and skipped the same five for the same probe reason. Over its analyzable set the author engine sampled 106,081 mutants; 17,502

pass every shipped test, a 16.50% survival rate; 7,727 of those survivors diverge somewhere in the probe bank, so they are alibis, which is 7.28% of all sampled mutants and 44.15% of survivors. The second engine found 105,425 mutants, 17,471 passers (16.57%), and 7,704 alibis (7.31% of mutants, 44.10% of survivors).

At the problem level, 2,636 of 5,721 analyzable problems admit at least one radius-1 alibi, 46.08%; the second engine found 2,630 of 5,725, 45.94%. That agreement is the load-bearing cross-implementation check in the paper, and it is not the only one: the singleton-witness share is 22.78% (author) against 22.83% (second engine); the best-probe corpus kill is 73.22% against 73.25%; the fully-closed share is 59.71% against 59.77%. Row by row in Table 7, the five census-rate rows differ by at most 0.14 percentage points, the closure and derived rows by at most 0.44 (cross-validated held-out kill, 48.35% against 48.79%), and the four length-quartile aperture rows by at most 0.32 (56.66% against 56.34%). Two qualifications belong with that number. Both engines were written by this paper's author, in the same project, against the same corpus and the same seed discipline, so the check establishes that two implementations of one specification agree and not that the measurement has been reproduced by anyone else. And neither engine, its inputs, or its output is in this repository, so the agreement cannot be re-checked from a checkout by an outside reader. The remaining problems include 1,051 (18.37%) whose sampled survivors never diverged anywhere in the bank; the second engine counted 1,054 (18.41%). Those probe-equivalent survivors are reported as unresolved, not as alibis: their true status is unknown, and an alibi without a witness is a claim this method cannot make.

Family structure is the clearest qualitative result. The classic boundary and arithmetic slips dominate: cmp survives at 35.26% and int at 25.93%, while the destructive rewrites nearly always die: notins at 2.26% and notdel at 1.90%. Which is the most that can be said without human false-pass data: the ordering is consistent with a slip-relevance reading of the families, but a family-ordering argument cannot separate a plausible-slip effect from a rewrite that is simply mechanically destructive, and no human false-pass data exists here to separate them.

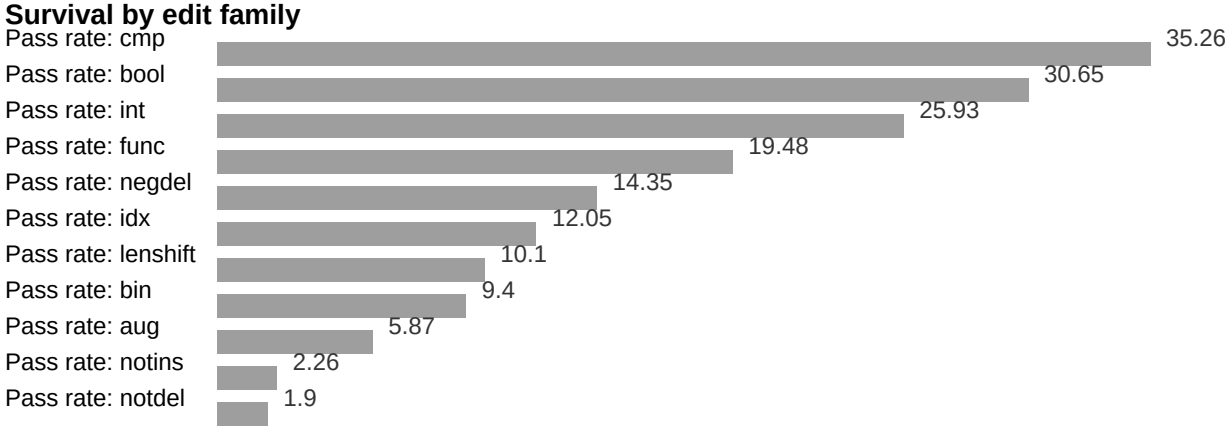
Family	Generated	Pass	Pass rate
cmp	8,576	3,024	35.26%
bool	1,801	552	30.65%
int	29,624	7,682	25.93%
func	955	186	19.48%
negdel	1,924	276	14.35%
idx	22,414	2,700	12.05%
lenshift	8,778	887	10.10%
bin	19,218	1,807	9.40%
aug	2,812	165	5.87%
notins	9,189	208	2.26%
notdel	790	15	1.90%
Total	106,081	17,502	16.50%

Mutant survival by family (author census, all analyzable problems). Generated, passing all shipped tests, and pass rate per family. The ordering is a slip-relevance ordering: the classic boundary and arithmetic slips survive most often, the destructive rewrites least.

The aperture rises with both difficulty and length. Easy problems sit at 35.77%, Medium at 49.39%, and Hard at 57.48%. By line quartile the share runs 20.24%, 41.45%, 56.66%, and 69.95%, with per-mutant survival rising from 11.45% to 18.57% (1.62x), the same direction independently (11.50% to 18.63%). Test count, read marginally, is associated with aperture (35.35% at three tests, 43.63% at four, 57.62% at five), but that association is largely length: holding the solution to 5-8 lines, the shares are 36.73%, 42.98%, and 41.83%, and per-mutant survival is actually higher at five tests (15.76%) than at three (12.06%). More shipped tests is not a proxy for a tighter suite in this corpus.

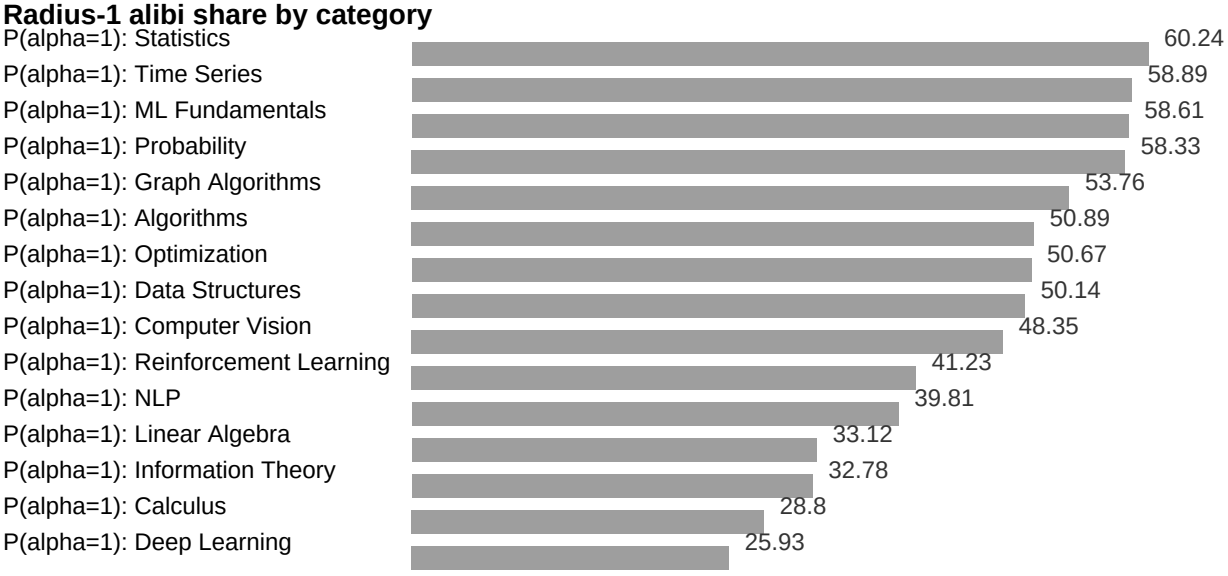
Slice	n	P(alpha=1)	Mutant survival
Easy	2,055	35.77%	-
Medium	2,549	49.39%	-
Hard	1,117	57.48%	-
Lines <= 4	1,502	20.24%	11.45%
Lines 5-8	1,503	41.45%	14.10%
Lines 9-14	1,435	56.66%	16.55%
Lines > 14	1,281	69.95%	18.57%
Tests 3 (marginal)	1,089	35.35%	-
Tests 4 (marginal)	2,998	43.63%	-
Tests 5 (marginal)	1,581	57.62%	-
Tests 3, lines 5-8	294	36.73%	12.06%
Tests 4, lines 5-8	840	42.98%	14.13%
Tests 5, lines 5-8	361	41.83%	15.76%

Aperture by slice (author census). P(alpha=1) is the share of the slice's problems with at least one radius-1 alibi. Test-count rows are marginal; the last three rows hold solution length fixed at 5-8 lines to separate test count from length.



Share of generated mutants that pass all shipped tests, by family. The boundary (cmp), boolean, and integer families dominate; the destructive rewrites (notins, notdel) almost never survive, which is what a slip-relevance ordering should look like.

Radius-1 alibi share by category



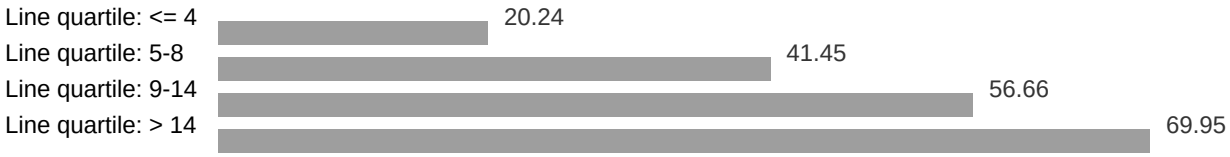
Share of problems with at least one radius-1 alibi, by category, sorted descending. The spread (25.93% to 60.24%) is far wider than the largest between-engine difference in Table 7 (0.44 percentage points), but it remains confounded with solution length and branching, which the census does not separate; the ordering is a hypothesis about where to look, not a validated triage signal.

Category spread is wide and larger than any between-engine difference reported in Table 7: Statistics 60.24%, Time Series 58.89%, ML Fundamentals 58.61%, Probability 58.33%, Graph Algorithms 53.76%, Algorithms 50.89%, Optimization 50.67%, Data Structures 50.14%, Computer Vision 48.35%, Reinforcement Learning 41.23%, NLP 39.81%, Linear Algebra 33.12%, Information Theory 32.78%, Calculus 28.80%, Deep Learning 25.93%. The ordering is a starting point for an authoring review queue, not a validated signal: a review queue that looks at Statistics first would encounter roughly twice the aperture of one that starts with Deep Learning. It is not a statement that one category's authors did worse work; higher-aperture categories are also the ones whose solutions are longer and more branch-heavy, and the census does not separate those effects, so the ordering may be a proxy for solution shape rather than for suite quality.

Witness sets are small. Across the 7,727 alibis the mean witness fraction is 11.03% of evaluable probes, the median is 7.69%, and the p90 is 24.00%; the second engine measured a mean of 11.04%. Affected problems hold a mean of 2.93 alibis, with a maximum of 19. A singleton witness set holds for 22.78% of alibis (1,760 of 7,727; second engine 22.83%), which means nearly a quarter of the mined silent failures are separated from the reference by exactly one bank input. An earlier draft of the wave-41 report understated this share by more than an order of magnitude, because a per-alibi count was divided by the wrong denominator; that figure is retracted. The corrected share strengthens, rather than weakens, the case that a witness is hard to guess: the median alibi reveals itself on well under one in ten probes, even though nearly a quarter of them reveal themselves on a single bank input.

Radius-1 alibi share by difficulty and by length quartile





The two slices from the census with the clearest structure: difficulty rises monotonically, and solution length rises across quartiles. Length is mechanically confounded with the number of sampled mutants per problem, so the length bars are reported as an association, not a causal claim.

An earlier draft also proposed a geometric per-line law, P(alpha = 1) growing as 1 - (1 - q)^L. It is withdrawn. The corrected length-decile recompute is 8.4 / 15.0 / 42.3 / 39.7 / 46.0 / 53.5 / 53.2 / 60.5 / 65.9 / 76.3% across deciles with mean lines 2.0 / 2.8 / 4.4 / 5.7 / 7.4 / 8.9 / 10.9 / 13.7 / 17.9 / 29.9. There is a step at decile three and no clean single-q curve; the decile ladder co-moves with the number of sampled mutants per problem (5.4 / 16.3 / 24.7 / 39.0 / 47.9 / 56.1 / 60.5 / 59.8 / 69.6 / 81.5), so length, branching, and sampling are entangled. What survives is the robust regularity already reported: survival rises with solution length, mechanically confounded by how many mutants a longer program can host.

Decile	Mean lines	P(alpha=1)	Mean mutants generated
1	2.0	8.4%	5.4
2	2.8	15.0%	16.3
3	4.4	42.3%	24.7
4	5.7	39.7%	39.0
5	7.4	46.0%	47.9
6	8.9	53.5%	56.1
7	10.9	53.2%	60.5
8	13.7	60.5%	59.8
9	17.9	65.9%	69.6
10	29.9	76.3%	81.5

Corrected length deciles (replaces the withdrawn per-line law). Mean lines, P(alpha=1), and mean sampled mutants generated per decile. The non-monotone step at deciles three and four and the co-movement with mutant count are why no single-q geometric law is claimed.

5. Closure: oracle versus deployable

For each affected problem, a witness chosen to maximize coverage over the problem's own alibis (an oracle choice) kills 73.22% of all alibis corpus-wide: 5,658 of 7,727. The second engine records 73.25%. That witness fully closes 59.71% of affected problems (1,574 of 2,636; second engine 59.77%), and after one such added test the affected share falls from 46.08% to 18.56% while aggregate mutant survival falls from 16.50% to 11.17%, a relative reduction of 32.3%. The second engine lands on 18.48% and 11.22%.

This is where the wave-41 draft overclaimed. Those one-probe numbers are oracle-chosen: the witness is selected on the same alibi set it is scored against, so 18.56% is an in-sample bound, not a deployable expectation. The honest experiment is cross-validation: choose the witness on half the

alibis and score it on the other half. The split is by md5 parity of the alibi label, and it only works when both halves are nonempty. Of 1,786 problems with at least two alibis, 1,332 enter the cross-validated cohort; the other 454 have all their alibis on one side of the parity split and are dropped. On the held-out half the witness kills 48.35%, against 71.96% for the oracle choice and 10.80% for a random probe. The second engine's cohort is 1,320 problems with a held-out kill of 48.79%. Closure generalizes to unseen alibis at roughly two-thirds of its in-sample effect, not all of it.

```
                    Deployable affected estimate
author: 46.08% x (1 - 0.4835) = 23.80% ; independent: 45.94% x (1 -
                                0.4879) = 23.53%
```

Reported as roughly 23.5-23.8% of problems still affected after one added test. The oracle number 18.56% (independent 18.48%) is an in-sample bound and must be labeled as such wherever it appears.

Full greedy 1-closure costs a median of one probe per affected problem, a mean of 1.594, a p90 of 3, and a maximum of 7 (independent: mean 1.593, maximum 6). Cumulative coverage: one probe fully closes 59.71% of affected problems, two close 86.57%, and three close 95.98%; four, five, six, and seven reach 98.79%, 99.62%, 99.96%, and 100%. The denominator for all of these is the mined sample over the fixed bank.

The prediction table from the research plan scored one split verdict that is worth keeping. Prediction P4 asked whether one oracle witness kills at least 75% of a problem's radius-1 alibis. The aggregate 73.2% is below that threshold, so the prediction as written fails; the mean per-problem kill rate, 82.94%, is above it, and 59.71% of affected problems have 100% of their mined alibis killed. The two statistics are not averaged: aggregate and per-problem readings answer different questions, and the split is reported as a split.

Radius-2 leakage was measured directly. On 120 affected problems with 24 random two-edit mutants each, silent survival was 12.52%; adding one oracle radius-1 witness reduced it to 8.56%, a 31.6% relative reduction, where a random probe reduced it only to 11.75%, a 6.1% reduction. Conditioned on problems with any radius-2 survivor, survival moved from 13.78% to 9.42%. The second engine measured the same direction and size: 11.91% to 8.42% (-29.3%) against a random probe's 11.31% (-5.0%). The radius guarantee does not transfer past radius 1; the directional benefit does.

Quantity	Author	Independent
Best single probe, corpus alibis killed	5,658/7,727 (73.22%)	73.25%
Best single probe, problems fully closed	1,574/2,636 (59.71%)	59.77%
Problems affected after one oracle probe	18.56% (in-sample)	18.48% (in-sample)
Aggregate mutant survival after one oracle probe	16.50% -> 11.17%	16.57% -> 11.22%
Cross-validated held-out kill (n = 1,332 / 1,320)	48.35%	48.79%
CV oracle kill / random probe	71.96% / 10.80%	-
Deployable affected estimate after one test	23.80%	23.53%
Greedy 1-closure cost (mean / median / p90 / max)	1.594 / 1 / 3 / 7	1.593 / 1 / 3 / 6

Quantity	Author	Independent
Fully closed by <= 1 / <= 2 / <= 3 probes	59.71 / 86.57 / 95.98%	59.77 / 86.62 / 95.93%

Witness closure: oracle probe and deployable estimate. The one-probe rows are oracle-chosen: the witness is selected on the same alibi set it is scored on, so 18.56% and 18.48% are in-sample bounds. The deployable estimate applies the cross-validated held-out kill rate to the affected share.

> *Nothing is auto-applied — Witness closure was not shipped as an automatic operation on tests or grading. Three blockers were reproduced: specification ambiguity means every witness needs author judgment, because an alibi can be a legitimate alternative reading of the statement; hidden tests plus no telemetry make a false wrong verdict silent; and timeout witnesses are hardware-dependent across CPython and Pyodide. The closure numbers in this section are measurements of an offline operation, and any future adoption requires a visible, labeled witness that never gates a certificate.*

6. Two negative results

The research plan predicted that a learner-probing duel would become an information-gain skill game once probe selection was guided by features learned from witness sets. It is not. With a 60/40 train/test split by problem, feature weights were learned on training witness sets (rate of each structural feature among witness probes, smoothed), and used to rank probes on held-out problems; a uniform-random probe order was the control. Within ten probes the random policy convicted 63.3% of problems, with a mean of 4.36 probes when successful, while the feature-guided policy convicted 59.5%, with a mean of 4.42. The ratio is 0.94x on the success rate: guided selection is not better than random. The predicted 1.5x advantage is falsified.

The consequence is a design fact: the duel's difficulty comes from the small witness fraction (median 7.7% of probes), not from hidden structure that a policy can learn. It is a short exposure exercise, not a strategic search game. This is one of the two results the paper most wants on the record, because the plausible next step, building a guided duel with a learned probe ranker, is already known to be wasted work.

The second negative result concerns explanations. For each alibi, the best conjunction of at most two literals over 21 structural input features was scored by balanced accuracy against the alibi's witness set. The plan predicted that at least 70% of alibis would admit a predicate with balanced accuracy at or above 0.9. Only 32.2% did (n = 671 alibis; median balanced accuracy 0.807). A permutation control, the best predicate for random subsets of the same size, reached a mean-max balanced accuracy of 0.881, and only 26.2% of real predicates beat the control maximum. A held-out split, predicate chosen on even probes and scored on odd probes over 743 alibis with at least three witnesses, gave 32.8% at the 0.9 threshold, with 53.8% beating the control maximum. The mechanism is real for a few recurring families (second_zero, second_one, any_empty, any_neg recur across problems), but an automatic explanation at the 70% threshold is dead.

- Probe-policy transfer: random 63.3% vs feature-guided 59.5% conviction within ten probes (0.94x success-rate ratio); not better than random.
- Predicate explanations: 32.2% of 671 alibis reach balanced accuracy 0.9 with <=2 literal conjunctions; permutation control mean-max 0.881; held-out 32.8% of 743, with 53.8% beating control.

Both results are reported with their thresholds because both were falsifiable predictions formed before the corpus run. Both survived only as dead ends; neither weakens the census, and both were run only on the author engine's data, so the second engine did not run them and no agreement on this section is claimed. No script or log for either sweep is committed.

7. Audits and validity

A census of divergence facts is only as valuable as the share of those facts that are actually bugs rather than legitimate alternative readings of the statement. Two audits address this, and neither is large. The author audit inspected ten randomly sampled alibis with their diffs and witnesses, not blind, and judged eight plainly wrong, one immaterial, and one subtle. The examples: pr-039 a dead empty-guard; op-183 a sign error in Expected Improvement (best - mu becoming best + mu); rl-259 a boundary change (<= 0 to <= 1); rl-318 a sentinel sign flip; st-291 an off-by-one guard that leads to division by zero; ml-346 a 1.0 - 1e-6 to 1.0 + 1e-6 clip change (immaterial, inside the tolerance scale); graph-043 a queue bound head < len(queue) becoming <=, producing an infinite loop and a timeout divergence; graph-218 a numerical threshold theta >= 0 becoming >= 1 (subtle); ts-282 a negative-value guard < 0 becoming < -1; ds-270 a ceil-division adjustment of -1 becoming -2. Eight of ten are plainly wrong, and the two exceptions illustrate the boundaries: tolerance-scale changes and numerically subtle thresholds.

The blind audit was sampled blind but not judged by anyone other than this paper's author: a script drew ten problems from its own ledger with a fixed seed (8675309), excluding the author's ten problem ids, chose one alibi per problem at random, printed the diff and up to two witnesses, and the author judged each case against the statement without having seen the first audit's labels or verdicts. Only the sample was independent of the first audit; the judge is the same person, and no audit transcript is committed. All ten were judged clearly wrong, zero spec-valid. Two examples: al-108 changes an empty-input guard from return 0 to return -1, diverging on [[], 2]; dl-064 replaces true division by floor division in a checkpoint average, diverging on [[[0, 0], [2, 2], [4, 4], [0, 0]]], where the reference returns [1.5, 1.5] and the ghost returns [1, 1]. Both audits sample only problems that already have an alibi, so they estimate precision within the mined set, not the aperture itself. They bear on the coupling assumption only indirectly: what they establish is that most mined objects are defects, not that those defects resemble human slips. That is a coupling question, and it is unmeasured at scale.

Several validity threats are structural and are not fixed by larger n. Specification ambiguity is undecidable from the machine-readable triple (S is natural language), so some alibis may be legitimate readings; this is the strongest reason not to auto-apply closure. Timeout divergences depend on the execution budget and the runtime and were excluded from the shipped practice set for that reason. Probe domains can leave the statement's promised domain (for example, non-square matrices), and the audits did not sample specifically for that property; the curated set excludes such witnesses by curation rather than by a measured rate. The withdrawn geometric law and the length confound in Section 4 are an example of the same discipline: when a regularity turned out to be partly an artifact of how many mutants a problem can host, it was deleted rather than smoothed.

The relativization caveat is the one to carry into any citation of the headline number. alpha is relative

to the edit model E and the probe domain D, and the census used eleven families and at most 48 probes. Enlarging either can only find more alibis, so 46.08% is a lower bound on any wider-domain notion of radius-1 aperture. It is not semantic distance, and it is not a property of learners. The census says: in this corpus, under this edit model, this fraction of exercises has a one-edit program that the shipped tests cannot distinguish from the reference anywhere they look.

- Author audit (n = 10, not blind): 8 plainly wrong, 1 immaterial (tolerance-scale clip), 1 subtle (numeric threshold).
- Second audit (n = 10, blind sampling of the ten ids, judged by this paper's author, seed 8675309, selection method above): 10/10 clearly wrong, 0 spec-valid.
- Coupling: the mutation-slip assumption is unmeasured at scale; the audits bound the precision of the mined set, not the coupling. No human false-pass data exists offline, and no audit transcript is committed.
- Spec ambiguity: undecidable from (S, f, T); every witness needs author judgment or a conservative class rule.
- Timeouts: machine-dependent; excluded from shipped content.
- Relativization: alpha is a lower bound relative to (E, D); both are fixed at census settings.

8. Product implications

The wave produced no change to grading, review, certificates, or any learner state. Three candidate product units were cut after reproduction. Automatic witness closure was cut because a witness needs author judgment, hidden tests make false verdicts silent, and timeouts are hardware-dependent. The examined solve and its Ghost Card were cut because alibis are mined from the reference, not from the learner's accepted code, so the claim that a learner cannot distinguish their own solution from its nearest alibi is false for any accepted g != f; random probing already convicts 63.3% of problems within ten probes, so the ritual would not be the skill game it was pitched as. Certificate or ledger integration was cut for the same broken premise and because a third post-solve gate has no evidence base. The negative results in Section 6 are the reason those cuts are cheap: the interesting UI, a guided duel with learned probe selection, is already known not to transfer.

What did ship is a standalone, optional practice route: Silent Bug Hunt at /alibi, built from 96 machine-verified mined alibis that survived both an uncapped union lazy suite and an independently constructed held-out suite, frozen into the data layer. Each puzzle is a pair of programs: the reference, which passes every shipped test, and the ghost, which also passes every shipped test and diverges from the reference at the stored witness input. The learner types a Python literal, the platform runs both programs locally, and the verdict is the raw divergence fact: value divergence beyond 1e-6, an exception, or a timeout, with the programs shown side by side and exactly one changed line between them. The route keeps all state in session memory; it writes nothing, persists nothing, and is reachable only as an optional exercise.

The bank has been rebuilt twice, and the honest reason matters. The first curated set selected mines by witness size, but a resistance check run against the visible shipped tests showed that 74-78% of its 96 puzzles were solvable within five lazy probes: single-element jitters of the shipped

inputs, the first thing a learner tries. That failed the practice claim the route was built on. The first re-curation gated on a capped 128-probe lazy sample; when the uncapped union suite was re-run against that bank, the union alone caught 84 of its 96 puzzles and alternate unseen shuffles of the same strategies caught 47-64, so only 3 of its 96 cleared both filters (4 cleared the union alone), 93 were dropped, and the bank was re-mined. The re-mine was exhaustive over the engine's radius-1 targets: 89,622 single-edit mutants across the 2,636 affected problems, 19,030 of which pass every shipped test; 4,060 of those are clean under the full union suite; 268 still diverge at some hidden probe; 217 of those are also clean under the held-out suite. The legacy pool contributed 9 union-clean candidates, 8 of them valid. The 253 combined candidates were re-validated end to end with fresh deep copies per program; 211 survived both suites, and 96 were shipped by category round-robin. Every shipped puzzle diverges on zero union and zero held-out probes while still diverging at the stored hidden-engine witness.

> *Practice, not proof of skill — Resistance is a gate on this specific union and held-out suite, not a claim that a ghost is unguessable. The shipped bank diverges on zero of 189,658 union probes and zero of 24,560 held-out probes, and the verifier's own S2/S3 sequences catch nothing at N = 1, 5, 10, or 25; alternate unseen shuffles caught 1 of 768 S2 sequences (at probe 21) and 0 of 768 S3 and S3-leaf sequences. That scope is load-bearing: a separately written attack script's seven novel probe families (cross-test draws, pairwise perturbations, magnitudes from 10^0 to 10^18, container restructures, string patterns, boundary-adjacent values, and type jitter) caught 6 of the 96 puzzles at N = 1, 11 at N = 5, 22 at N = 25, and 35 at N = 100. It was written by this paper's author, so "independent" there describes a separate program, not a separate person. Systematic input variation still solves a meaningful share. The design intent behind the route is that a shipped suite is not the behavior and that not every ghost demands a leap; no learner has been observed doing either, so that intent is unevaluated. Those sweep figures are records of attacks run at authoring time with no script or log committed. The game remains practice, not proof: human behavior is unmeasured, and an adversarial solver can still convict a ghost with an input no fixed suite contains.*

Property	Value
Puzzles	96 (one per affected problem, unique)
Re-mine funnel	89,622 single-edit mutants over 2,636 affected problems -> 19,030 shipped-test passers -> 4,060 clean under the full union lazy suite -> 268 with a hidden witness -> 217 also clean under the independently constructed held-out suite; legacy pool 9 -> 8 valid; combined 253 re-validated -> 211 valid -> 96 shipped by category round-robin
Resistance	96/96 resistant: zero divergence over 189,658 union probes and 24,560 held-out probes (fresh deep copies per program)
Divergence class at stored witness	32 crash, 64 value, 0 timeout
Difficulty	Easy 24 / Medium 49 / Hard 23
Categories	all 15, 3-10 each

Property	Value
Shipped tests	412 (reference and ghost pass all of them)
Text diff vs reference	exactly 1 changed line in all 96
Re-verification	96/96: reference passes tests, ghost passes tests, ghost diverges at witness, zero divergent union and held-out probes
Harness semantics	1e-6 deep equality, 0.25 s alarm
Serialized size	152,696 B raw / 27,510 B gzip -9 (JSON.stringify, compact; gzip -9 -n)

Silent Bug Hunt: the re-mined, union- and held-out-resistant practice set. The shipped artifact is a frozen, machine-verified set of mined alibis re-mined under an uncapped union-suite resistance gate and filtered by an independently constructed held-out suite. Each record stores the reference, the ghost, the shipped tests, the witness input, and the union and held-out probe counts it survived; a panel of one changed line separates the two programs.

The curated set is the census turned into content: 32 crash-divergence witnesses and 64 value-divergence witnesses, no timeouts; difficulty Easy 24, Medium 49, Hard 23; all 15 categories represented, 3-10 puzzles each; 412 shipped tests, every one passed by both the reference and the ghost; exactly one changed line in all 96, because the stored references are ast.unparsed so formatting is identical; and 96/96 re-verified for the five load-bearing claims (reference passes the shipped tests, ghost passes the shipped tests, ghost diverges from reference at the stored witness under a 1e-6 comparison and a 1.0 s hardware-tolerant gate alarm, and ghost diverges on zero of its union probes and zero of its held-out probes). The whole set serializes to 152,696 bytes raw and 27,510 bytes gzip -9. The permanent gate scripts/verify-alibis.ts plus scripts/py_alibi_verify.py re-checks, in real Python outside the browser, the claims attached to the 96 shipped puzzles — reference passes the shipped tests, ghost passes the shipped tests, divergence at the witness, a one-line diff, and zero divergence over 189,658 union and 24,560 held-out probes — and exits non-zero on any failure; the last gate run passed 96/96 with zero union, held-out, metadata, or witness failures. It does not recompute the census, the mining funnel, the closure estimates, the cross-validated numbers, the negative results, or the audits.

The safety rule is mechanical, not editorial: alibis never gate grading or certificates. The practice route imports no module that can write progress, submissions, review state, certificates, or bug-hunt history, and a source-scan test enforces that none of the route, the client component, or the pure harness module imports a stateful module or calls a state-writing function. No witness is hidden, none is added to any graded suite, and no learner-visible statement calls a ghost wrong; the copy states divergence as a fact and leaves interpretation to the reader. The census tables, not a UI, are the other shipped artifact: they are the aperture report an author can use to decide where a boundary test is worth adding. If that ever happens it is a manual, reviewed change to shipped tests, one extra test call per submission, not a runtime gate.

9. Reproducibility

> *Limitation: what is absent from the release, and what the gate actually checks — Neither the alibi mining engine nor any raw artifact of the mining run is in this release, and the machine-local scratch trees named at the end of this section no longer exist. Absent from the release are: the author engine and the from-scratch second engine; the 106,081-mutant sample, the 17,502 survivors, the 7,727 alibi records and their witness sets; the per-family, per-slice, per-decile and per-category tables derived from them; the probe-bank statistics (mean 39.05 probes, 89.22% reference-evaluable) and the 300-problem cap-120 robustness rerun; the closure and cross-validation computations; the 89,622-mutant re-mine funnel; the two audits and their transcripts; the S2/S3 shuffle sweeps and the seven-family attack sweeps; and the cost measurements (roughly 4.5-5 million Python calls, about 27 CPU-minutes, about 4 minutes wall-clock on 8 workers), which are machine-local and were not instrumented. Consequently the permanent gate checks something strictly narrower than this paper's headline. What the gate computes from a clean checkout is the 96 shipped puzzles — 189,658 union and 24,560 held-out probes in real CPython, the one-line diff, the witness divergence, the 412 shipped tests, and the eight stored funnel constants, which it re-prints without re-deriving them. What an outside reader can therefore verify is that the 96 shipped puzzles are sound. What an outside reader cannot verify is every census, closure, cross-validation, negative-result, and audit number in Sections 4 to 8; those are the record of a run whose programs are not shipped. The paper reports them unchanged because they are measurements, not because they are re-derivable, and this distinction should travel with any citation of 46.08%.*

> *Recomputed versus replayed — A gate that recomputes a value and compares it with a stored constant proves that the pipeline is deterministic; it does not by itself prove the measurement is correct. The distinction applies unevenly across this paper. Recomputed on every gate run, against real CPython execution and no stored constant: the 96/96 result, the 189,658 + 24,560 probe counts, the 32 crash / 64 value / 0 timeout split, the 24 / 49 / 23 difficulty mix, the 15-category coverage, the 412 shipped tests, the one-line diff in all 96, and the 152,696 B / 27,510 B serialization. Replayed only — held in the gate as literals and compared for equality: the eight re-mine funnel counts, which the gate prints without re-deriving any step. Not gated at all: every census, closure, cross-validation, witness-statistic, decile, category, cost, and audit number. Where correctness for the replayed set comes from an external oracle rather than from the constant, the paper says so; where it does not, no independent check exists in this repository.*

The census procedure requires no network access, no model, and no new dependency, and both engines ran against the repository corpus: the author engine over 5,721 analyzable problems and a from-scratch second engine over 5,725. Both used the platform harness semantics (real Python, 1e-6 deep equality, first def as entry point, 0.25 s per-call alarm) and the same seed discipline: the mutant target order is seeded by int(md5(problem_id)[:6], 16), the probe shuffle by int(md5(problem_id)[:8], 16), and the resampling and audit scripts use fixed seeds 11, 5, 7, 99, 2026, 3, and 41. The blind audit used random.Random(8675309). Two limits belong here rather than later. Neither engine, nor its input manifest, nor its chunked output is part of this repository, so the census is not reproducible from a checkout by an outside reader; the determinism result reported at the time (byte-identical chunk files on a re-run) is a property of code a reader cannot obtain and cannot be re-checked. And the agreement recorded in Table 7 is between two implementations written by this paper's author, so it is an internal cross-implementation record, not external replication.

• Probe bank: first three shipped inputs perturbed one argument at a time, deduplicated, shuffled by

md5(id), cap 48 (robustness rerun at 120), reference-evaluable only; mean 39.05 probes, 89.22% evaluable.

- Mutants: at most 36 per problem and at most 6 per family; unparse round-trip must pass T or the problem is skipped.
- Divergence: 1e-6 deep equality; raise or timeout where the reference returned counts as divergence; reference failure drops the probe.
- Timeout: 0.25 s per call; the practice set excludes timeout-only witnesses.
- Skip rules: at least five reference-evaluable probes are required; four problems that raised during mining were handled by the second engine.

Metric	Author	Independent
Problems / analyzable	5,730 / 5,721	5,730 / 5,725
Sampled mutants	106,081	105,425
Mutants passing all shipped tests	17,502 (16.50%)	17,471 (16.57%)
Probe-divergent survivors (alibis)	7,727 (7.28% of mutants)	7,704 (7.31% of mutants)
Problems with at least one alibi	2,636 (46.08%)	2,630 (45.94%)
Probe-equivalent survivors	1,051 (18.37%)	1,054 (18.41%)
Singleton-witness alibis	1,760/7,727 (22.78%)	22.83%
Witness fraction mean / median / p90	11.03% / 7.69% / 24.00%	11.04% / 7.69% / 24.00%
Best single probe kill / fully closes	73.22% / 59.71%	73.25% / 59.77%
After one oracle probe, affected / survival	18.56% / 11.17%	18.48% / 11.22%
CV held-out kill	48.35% (n = 1,332)	48.79% (n = 1,320)
Deployable affected estimate	23.80%	23.53%
Length quartiles P(alpha=1)	20.24 / 41.45 / 56.66 / 69.95%	20.11 / 41.38 / 56.34 / 69.86%
Length quartile survival, shortest -> longest	11.45% -> 18.57% (1.62x)	11.50% -> 18.63%
Corrected length deciles	8.4 / 15.0 / 42.3 / 39.7 / 46.0 / 53.5 / 53.2 / 60.5 / 65.9 / 76.3%	same recompute
Monotonicity counterexample	la-029 idx-@5; D grows by 17 probes	reproduced

Two-engine agreement: census rates and closure estimates. Author engine and a second engine written from scratch over the same corpus. Observed agreement, row by row: the five census-rate rows differ by at most 0.14 percentage points; the closure and derived rows by at most 0.44 (cross-validated held-out kill); the four length-quartile aperture rows by at most 0.32 (56.66 against 56.34). Both engines were written by this paper's author, and neither engine, its inputs, nor its output is part of this release, so this table is a provenance record of a past run rather than a reproducible check; the permanent gate does not read it. The second engine analyzed four problems the author run skipped.

The shipped artifact has its own permanent gate. scripts/verify-alibis.ts serializes each puzzle's

reference, ghost, entry function, shipped tests, witness, and union and held-out probe counts to a temporary JSON, shells into scripts/py_alibi_verify.py, and asserts per puzzle that the reference passes all shipped tests, the ghost passes all shipped tests, the two diverge at the witness under 1e-6 deep equality with a 1.0 s hardware-tolerant gate alarm, the text diff is exactly one changed line, and the ghost diverges on zero probes of the full union suite and zero probes of the independently constructed held-out suite. Any failure prints a per-puzzle line and exits non-zero; success prints PASS 96/96 with zero union, held-out, metadata, and witness failures. The gate never runs in the browser. The re-mined set was generated once from the scratch artifacts by a deterministic rule recorded with it: every radius-1 engine target of every affected problem was evaluated (89,622 mutants); shipped-test passers were run against the union suite; candidates clean under the union and still diverging at a hidden-engine witness were filtered by the held-out suite; after combining with the legacy pool, 253 unique candidates were re-validated end to end with fresh deep copies, 211 survived both suites, and 96 were taken by category round-robin, sorted Easy to Hard then by id. The gate stores those eight funnel counts as constants and re-prints them; it does not re-derive any step of the funnel, and the mining programs are not part of this artifact. What the gate does recompute is every per-puzzle claim in the rows above. Re-running the gate is the single command that keeps the shipped content honest.

Reproducing the reported results

```
# permanent integrity gate for the 96 shipped puzzles
bun run scripts/verify-alibis.ts

# same gate via the package script (once wired)
bun run verify:alibis

# paper and PDF projections (all registered inventions)
bun test tests/inventions.test.ts tests/pdf.test.ts
```

The raw mining run is not committed. Its artifacts live in scratch directories outside the repository: the author chunks and engine under /var/folders/lc/r25hfwjs4j963w2f40s9rc5h0000gn/T/opencode/w41/, the second engine's shards and stats under .../w41-verify-empirical/, and the theory verifier's recomputes under .../w41-verify-theory/. The final selection artifacts are .../w41-remine2/: ex4-*.jsonl with their .stats files (the exhaustive funnel), phase1.py and phase1-survivors.json (the legacy-pool re-evaluation that showed the old bank was overfit), suites.py (both suite builders), remine4_driver.py and remine4_slot.py (the exhaustive re-mine), and build_bank.py plus selection.json and build-data/alibi-bank.json (the 96 shipped puzzles and their metadata). The superseded 128-probe bank is .../w41-remine/selected-resistant.json, kept only as the retraction record. The typed records in src/data/alibis/puzzles.ts are generated from selection.json with a stored provenance header, and scripts/verify-alibis.ts is the permanent gate. Nothing in the scratch directories is needed at runtime; the checked-in paper and the verified puzzle records are the durable outputs. Those directories are machine-local temporary paths and are not expected to survive the authoring session: a reader who follows them will find nothing there. The durable outputs are the paper, the verified puzzle records, and this provenance header, and the census numbers above are therefore records rather than re-derivable artifacts.

10. Limitations & future work

> *Limitations that apply to every number in this paper — Four of them, stated once so they cannot be lost in a citation. (1) No human-participant data. Nothing in this paper was measured on a learner, a submission, or a graded run; the mutation-slip coupling that would make 46.08% educationally meaningful is an assumption, and the two audits measure the precision of the mined set rather than the coupling. (2) No external replication. The second engine was written by this paper's author, in the same project, against the same corpus; the agreement in Table 7 is a cross-implementation record, not a replication by an independent party, and no party outside this project has attempted to reproduce any number here. (3) The census is not reproducible from a checkout. Neither mining engine, nor any raw artifact, is in this release; the gate recomputes the 96 shipped puzzles and re-prints the eight stored funnel constants, and it does not recompute the census, the closure, the cross-validation, the negative results, or the audits. A gate that compares a recomputed value against a stored constant proves determinism, not correctness, and for the funnel constants that is the whole of what the gate does. (4) No experiment here measures a generative-AI tutor. Where this paper mentions a model it does so to say that no model is used in the pipeline, and nothing in Sections 4 to 8 bears on how a generative tutor behaves, teaches, or is graded; the external evidence on generative tutoring cited elsewhere in this portfolio is about other systems and other designs.*

The most important limitation is the one no offline method in this wave can fix: mutation-slip coupling is assumed rather than measured in this corpus. The census asks whether a single-edit mutant can pass the suite; the product implication we care about is whether a human slip can. The n = 10 blind audit bounds the precision of the mined set and nothing more; it is a precision check on ten items, with a wide interval. The proposed study is shadow logging: record learner submissions that pass every shipped test and are later contradicted by a run or a review, then compare those false passes against the mined alibis for the same problems, by edit family and by distance. That study needs telemetry the product does not have, so it is recommended, not shipped. Until it exists, alpha is a property of the corpus and the edit model, and the paper should never be cited as a learner-error rate. The gap is a data gap rather than an open theoretical question: Section 2 notes that mutant-to-fault coupling has been measured on other data, including on student implementations in CS courses, and this corpus simply has none of the human false-pass records that such a study needs.

The measurement is also narrow in ways the corpus dictated. It is Python-only, first-def-only, and limited to pure functions with no classes, I/O, or randomness; the edit families are plausible slips but a small, non-exhaustive set; the probe bank is a deterministic construct whose 89.22% reference-evaluable mean leaves real gaps in the diverge set; and the census 46.08% is a lower bound relative to (E, D), not a domain-wide constant. Nine problems are absent from at least one engine: four raised during mining and five have fewer than five reference-evaluable probes. One problem in five has probe-equivalent survivors whose status is unknown, and those are reported as unresolved rather than folded into the alibi count. The compute, about 27 CPU-minutes plus probe execution, is fine at build time and impossible per interaction.

- Shadow logging: join learner false-passes to mined alibis; the only test of coupling at scale.
- Wider edit models and probe domains: alpha is a lower bound, so every expansion is measurable; report how much aperture the new families find.
- r-closure accounting: 1-closure leaves radius-2 survival at 8.56%; measure whether greedy

multi-witness closure is worth its test cost.

- Maintenance: content-hash caching and CI re-mining only for problems whose solution or tests changed; the full run is only needed for a new corpus snapshot.
- Audits: expand to n >= 100 only if the sampling and judgment cost is justified by a decision that depends on the interval.
- Learner outcomes: no study yet exists on whether silent-bug practice transfers; the duel probe-policy result says the practice is exposure, not search training.

The honest summary of closure is bounded. One added test, chosen from the mined alibis, removes the mined near-miss class for a median cost of one probe and fully closes 59.71% of affected problems; roughly half of what it removes, 48.35%, generalizes to alibis chosen on held-out halves, giving a deployable estimate of 23.5-23.8% still affected instead of the oracle's in-sample 18.6%. The guaranteed quantity is a radius-1 guarantee over the sampled edit family and the fixed probe bank; it is not full coverage, and the paper does not present it as such. Closing the remaining aperture needs either a wider radius, richer edit models, or authored tests written with judgment that no offline method here supplies.

References

1. [withdrawn] speccle ADR-0012 — the repository this pointed to could not be located, so the sentence in Section 2 that leaned on it was reworded to stand on this paper's own argument and this entry is not to be cited. https://github.com/matthewalton/speccle/blob/main/docs/adr/0012-strengthen-routes-on-the-survivor-not-the-score.md
2. DeMillo, Lipton & Sayward 1978, Hints on test data selection: help for the practicing programmer. https://doi.org/10.1109/C-M.1978.218136
3. Jia & Harman 2011, An analysis and survey of the development of mutation testing. https://doi.org/10.1109/TSE.2010.62
4. Papadakis, Kintis, Zhang, Jia, Le Traon & Harman 2019, Mutation testing advances: an analysis and survey. https://doi.org/10.1016/bs.adcom.2018.03.015
5. Just, Jalali, Inozemtseva, Ernst, Holmes & Fraser 2014, Are mutants a valid substitute for real faults in software testing?. https://doi.org/10.1145/2635868.2635929
6. Inozemtseva & Holmes 2014, Coverage is not strongly correlated with test suite effectiveness. https://doi.org/10.1145/2568225.2568271
7. Danglot, Vera-Perez, Baudry & Monperrus 2019, Automatic test improvement with DSpot. https://doi.org/10.1007/s10664-019-09692-y
8. McKeeman 1998, Differential testing for software. https://www.cs.tufts.edu/comp/150FP/archive/bill-mckeeman/DifferentailTesting.pdf
9. Zeller & Hildebrandt 2002, Simplifying and isolating failure-inducing input. https://doi.org/10.1109/32.988498
10. Goldman & Kearns 1995, On the complexity of teaching. https://doi.org/10.1006/jcss.1995.1003
11. Zhu 2015, Machine teaching: an inverse problem to machine learning and an approach toward optimal education. https://doi.org/10.1609/aaai.v29i1.9761
12. Bjork & Bjork 2020, Desirable difficulties in theory and practice. https://sites.lifesci.ucla.edu/psych-bjorklab/wp-content/uploads/sites/13/2021/01/RABjorkELBjorkJARMAC2020ForPostingSingleSpaced.pdf
13. STING 2026, Probe to Generate: Program variant-guided test augmentation for repository-level benchmark

repairs. https://arxiv.org/abs/2604.01518

14. Prasad, Greenman, Nelson & Krishnamurthi 2024, Conceptual mutation testing for student programming misconceptions. https://doi.org/10.22152/programming-journal.org/2024/8/7
15. Padurean, Denny & Singla 2025, BugSpotter: automated generation of code debugging exercises. https://doi.org/10.1145/3641554.3701974
16. Padurean et al. 2026, When AI is wrong on purpose: how students respond to buggy GenAI code. https://doi.org/10.1145/3765964.3811667
17. Caraco, Lojo, Verdicchio & Fox 2024, Generating multi-part autogradable faded Parsons problems from code-writing exercises. https://doi.org/10.1145/3626252.3630786
18. Katabench, mutation-graded test-writing exercises (grading documentation). https://katabench.com/docs/grading
19. speccle ADR-0012, strengthen routes on the survivor, not the score. https://github.com/matthewalton/speccle/blob/main/docs/adr/0012-strengthen-routes-on-the-survivor-not-the-score.md