Behavioral Delta Ledger: Per-Edit No-Op Detection on a Hidden Basis Derived from an Exercise's Own Tests

DeepForge Research

September 18, 2026 · behavioral-delta-ledger

Preprint · Artifact · Not peer reviewed

Abstract

A learner's repeated Run on an auto-graded exercise usually leaves one bit — pass or fail — and no memory of the previous program. We define a per-edit behavioral signature: a deterministic 48-probe basis built by perturbing the exercise's own test inputs, relative to the shipped reference, with states agree / wrong value / raise-or-timeout, executed under a fresh-copy protocol. Mining the DeepForge corpus (5,730 exercises, 5,682 analyzable, 88,357 sampled single-edit mutants) shows 10.23% of mutants are invisible on the basis (95% CI [10.03, 10.43]), 16.45% pass every shipped test, and 44.40% of problems carry a test-passing slip the basis can see. On 699 simulated edit walks, 34–51% of edits are exact no-ops by signature. In-sample, on the same full-sign statistic used for the held-out figures, the sign of the hidden delta matches the sign of the test delta for 87.78% of climb steps and 83.40% of random steps; with the basis built only from odd-index tests and the delta scored on even-index tests the held-out concordance falls to 84.00% (AUC 0.829) and 82.23% (AUC 0.797) — a drop of 3.78 and 1.17 points, and the number we report. A positive-only in-sample rate, restricted to steps with a nonzero test delta, reads 90.20% (AUC 0.903); that is a different statistic on a different denominator and is not comparable to the held-out figures, so the 6.20-point gap between 90.20% and 84.00% is not a leakage measurement. Break-induced risk of a next-attempt test regression is 1.14× (95% CI [0.88, 1.47]) and 1.36× ([1.04, 1.79]), below the pre-registered 2× prediction, so the ledger is retrospective attribution, not forecasting. Persistent “cold” probes are common under the formal last-three-attempt definition (85.8% of climb walks have >= 4) but almost never concentrated in one perturbation family (1.50%), and the aggregate enrichment over structural transformations is post hoc and modest (reverse z = +3.1, empty z = -11.8 under a within-walk permutation null); all residual routing is therefore excluded from the shipped design. The shipped instrument is a count-only, opt-in, standalone practice route that stores one signature per problem, never sources, never families, and never grades; it reports “k of N hidden checks changed” or “no change on N hidden checks”, because probe blindness makes stronger language false.

Keywords: behavioral delta, differential testing, probe blindness, novice edits, no-op edits, exercise corpus, Python, education

1. Introduction & motivation

On a practice platform, pressing Run on an auto-graded exercise produces one bit of information and destroys the process that produced it. The learner sees pass or fail; the browser keeps only the latest source. A learner on their fifth edit cannot tell which of the five changed anything, whether the last one was a rename, or whether an earlier failing case quietly regressed. The platform cannot tell either, because nothing ever compares two of the learner's own programs.

The Behavioral Delta Ledger (BDL) makes the comparison executable. For a problem with reference

f and shipped tests T, the Ledger builds a deterministic probe basis from the exercise's own test inputs (one argument perturbed at a time, exact shipped inputs excluded), executes the submission on that basis under a fresh-copy protocol, and reduces the run to a ternary signature: agree, wrong value, raise-or-timeout. After each Run it diffs the new signature against the previous one and reports counts only: how many hidden checks changed, whether the signature is byte-identical (a behavioral no-op edit), and whether the visible tests changed. The basis is hidden; the reference is recomputed locally per run; nothing is graded.

This paper reports the measurement program behind that instrument: a full-corpus census of 88,357 sampled single-edit mutants on 5,682 analyzable exercises, a study of 699 simulated edit walks under three policies, a leakage-controlled re-measurement of the in-sample concordance result, and a formal reconciliation of the cold-set object that the first draft measured incorrectly. It also reports what failed: four pre-registered predictions did not survive, and the failures removed copy from the product rather than being smoothed in the text.

- Per-edit hidden signature: a deterministic basis derived from the exercise's own tests, ternary states under a fresh-copy protocol, diffed between two consecutive learner programs.
- Census: 5,682 analyzable exercises, 88,357 sampled single-edit mutants, 10.23% invisible on the 48-probe basis, 16.45% passing all shipped tests, 45.23% of those hidden-visible, and 44.40% of problems carrying at least one visible silent slip.
- Walk study: 699 simulated walks (climb / random / revert) with exact-signature ghost rates of 48.37% / 51.16% / 33.98% after the corrected no-op definition, replacing a definition that missed wrong-to-wrong state flips.
- Held-out collapse: 84.00% (AUC 0.829) and 82.23% (AUC 0.797) under an odd-test-basis, even-test-scoring protocol, a drop of 3.78 and 1.17 points from the like-for-like full-sign in-sample rates of 87.78% and 83.40%; directional copy is cut.
- Formal cold-set reconciliation: the measured object (walk-long) and the defined object (last k = 3) disagree, H6b is dead under both, and the residual routing story is cut from the product.
- Product decision: a count-only, opt-in, standalone route that never states direction and never touches grading, review, certificates, or sync.

> *Novelty, stated honestly — The visibility of hidden-basis divergence is wave 41's result, and this run is consistent with it rather than a replication of it: 44.40% of problems carry a test-passing slip the basis sees, against Alibi Distance's 46.08% P(alpha=1) on a basis that included the shipped inputs. The two statistics are built differently - different basis composition and a different estimand (a test-passing slip against a radius-1 alibi) - so their 1.68-point difference is not a measure of agreement. What this paper adds is the object: the delta between two of a learner's own consecutive programs on a hidden basis derived from the exercise's own tests, the exact no-op edit as a behavioral event, and the measured limits of that object (held-out sign concordance 84.0 / 82.2%, 10.2% probe blindness, cold-set coherence 1.5%). No priority claim is made for the trace object, the no-op definition, or any rate over it.*

> *Four predictions failed and are reported as failures — H2b (family ordering): predicted cmp and int would be the least invisible; cmp is the most invisible at 24.75% and notdel the least at 1.47%. H3a (hidden-visible test-passers): 45.23% misses the 50% target. H5b (next-test regression): 1.14x and 1.36x, below the pre-registered 2x. H6b (family-coherent cold sets): 1.50% and 0.46% against a predicted 40%. Each failure is a measurement that removed product copy, not an anomaly to explain*

> *away.*

The paper also states a limit that applies to every learner-facing number below: there is no human data. Every walk, ghost rate, and improvement number comes from simulated edit policies or synthetic cohorts on the real corpus. Whether real learners produce no-op edits at these rates, and whether the count card helps them, is unmeasured. The first human experiment proposed here is a local shadow log of consecutive signature pairs.

2. Related work

Differential testing (McKeeman, 1998) runs two comparable programs on shared inputs and treats divergence as the oracle. BDL is a differential test in miniature, but three things differ: the setting is a learner's own consecutive submissions rather than independent implementations; the input coordinate system is a deterministic basis derived from the exercise's own tests relative to the shipped reference; and the product is a count of changed checks with a no-op statement, never a verdict. Regression testing supplies the fix/break intuition; the measured held-out concordance bounds how far that intuition can be carried, and the Ledger's copy stops short of it.

Mutation testing is the parent methodology. DeMillo, Lipton, and Sayward (1978) framed test data selection around seeded simple faults; Jia and Harman (2011) and Papadakis et al. (2019) surveyed the field; Just et al. (2014) gave the strongest evidence that mutants correlate with real faults. BDL samples the same single-edit families as the wave-41 census, but its object is not a mutation score: it computes the behavioral delta of an edit rather than the survival of a mutant, and it is relative to the learner's previous program, not to an absolute reference. Inozemtseva and Holmes (2014) showed coverage is a weak proxy for suite effectiveness, which is why the Ledger reports counts over a basis instead of a score. DSpot (Danglot et al., 2019) amplifies developer tests, and the position that the individual surviving mutant matters more than an aggregate adequacy threshold is also the position this paper reaches independently, by measuring the estimator rather than the score. The Ledger is adjacent to both but never modifies a shipped suite or gates a learner.

On the education side, Jadud (2006) characterized novice compilation behaviour and flailing from error streams; the Ledger measures futility behaviorally, judging an edit by whether it changed observable behavior rather than by its text, error count, or timing. Examplar (Prasad et al., 2024) uses curated chaff mutants to test student-authored examples; BugSpotter (Padurean et al., 2025), When AI Is Wrong on Purpose (2026), FPPgen (Caraco et al., 2024), and Katabench all generate buggy code as practice. Those bugs fail by construction or are findable by writing tests; BDL's subject is the learner's own edit, which may change nothing at all. Errorful learning (Kornell, Hays, and Bjork, 2009; Metcalfe, 2017) and desirable difficulties (Bjork and Bjork, 2020) motivate the opt-in replay shelf, but the pedagogy is old and the paper makes no learning-outcome claim. Spacing references (Cepeda et al., 2006; Karpicke and Bauernschmidt, 2011; the FSRS algorithm) describe the scheduler the Ledger deliberately does not touch.

Delta debugging (Zeller and Hildebrandt, 2002) minimizes a known-failing input; a hidden probe exists where no shipped input fails. Machine teaching (Goldman and Kearns, 1995; Zhu, 2015) asks which examples identify a concept; BDL asks a different question, whether an edit changed anything on a fixed hidden sample. STING (2026) augments benchmarks with tests that kill surviving program

variants; BDL's wave-41 neighbor, the closure result, shares that goal, while BDL itself produces a feedback count and no new tests at all.

Source	Taken	Not claimed
Differential testing	Divergence between comparable programs as an oracle	A correctness verdict; BDL counts changed hidden checks
Mutation testing	Single-edit families as a slip model	A mutation score for a program under test
Wave 41 / Alibi Distance	The probe-divergence visibility phenomenon (re-measured here on a different estimator, not replicated)	Discovery of probe blindness or a radius account
Novice flailing (Jadud)	Behavioral futility as a distinct event from error count	An error-quotient replacement or a diagnostic of intent
Errorful learning (Kornell; Metcalfe)	Re-encountering one's own wrong program	Evidence that replay improves learning
DSpot and the survivor-first position	Act on the individual survivor, not the aggregate score	Automatic test augmentation or repair

What BDL takes and what it does not claim

3. Method

A problem is a triple p = (S, f, T): a statement S, a Python reference f, and a shipped test set T with three to six cases. The basis B(p) is built from the first three shipped inputs by perturbing exactly one argument at a time using the frozen perturbation set (empty, singleton, reversed, duplicated, extended, shrunk, negated, sorted, incremented; matrices are also transposed and row-perturbed). Exact shipped inputs and duplicates are excluded; the candidate list is shuffled with a seed derived from the problem id and the first 48 (research) or 24 (product runtime) probes with a reference answer are kept. Probes where the reference itself raises or times out define no answer and are dropped from comparisons.

```
                          Behavioral signature
sigma_j(g) = 0 if g(args_j) returns and deep_eq(g(args_j), f(args_j));
   1 if g returns a different value; 2 if g raises or times out. The
         reference state x means f has no answer at that probe.
```

Equality is the platform's 1e-6 deep-equality semantics; state 2 covers raises and the per-call timeout. x dimensions are not comparable and are excluded from churn counts. 1<->2 transitions count as changes under the corrected definition below.

Execution follows the fresh-copy protocol: every call runs on a fresh deep copy of the probe arguments. This is not a stylistic choice. A dedicated pass over the corpus found 8 of 5,682 references (0.14%) mutate their argument objects; a shared-object protocol flaked on 9 problems and disagreed with fresh-copy on 8, while the fresh-copy protocol flaked on 0. The same pass exposed a second bug in the transform check: consistent local renaming initially reported churn on

two nested-function problems because nonlocal declarations were not renamed with the variable. After the fix, rename churn is 0 on 4,680 of 4,680 applied renames and cosmetic churn is 0 on 5,682 of 5,682.

Corrected churn and ghost edit (E1)

Churn(sigma, sigma') = |{ j : sigma_j != sigma'_j }| over comparable dims; ghost = text_changed and sigma = sigma' exactly and the visible test bitmask is unchanged.

The earlier definition, Churn = |Fix| + |Break|, ignored nonzero-to-nonzero transitions and reported behavior-changing edits as ghosts. It is superseded. Equal signatures prove only no change on the basis, never no behavioral change; product copy says 'no change on N hidden checks'.

The walk study used three policies over a stride-24 sample: climb (edit maximizing tests passed), random (uniform single edit), and revert (alternating edit and undo), each 233 walks over 2,330 / 2,330 / 2,796 steps. Walk RNG seeds are derived from the problem id and the policy index. Sub-basis sufficiency was measured on a stride-12 sample (477 problems) against the 48-probe research basis and on a 600-problem stratified sample against a 96-probe reference. The leakage protocol is separate: the basis is built from odd-index shipped tests only and the delta is scored on even-index tests over independent walks (n = 18,191 steps). The clean corpus run used a truncating writer with no resume path, one fresh record per problem.

Basis, signature, and delta (research form)

```
# p = (id, f, T); f is the reference, T the shipped tests
B = perturb_one_argument(first_three_inputs(T))   # frozen perturbation set
B = dedupe(exclude_shipped(B))
B = shuffle(B, seed=int(md5(id)[:8], 16))          # product seeds with FNV-1a
instead
B = [b for b in B[:cap] if reference_answers(f, b)]  # cap 48 research / 24 product

sig(g) = [state(g, b) for b in B]                  # 0 agree, 1 wrong, 2
raise/timeout

churn = count(sig_before[j] != sig_after[j] for j in comparable)
ghost = text_changed and sig_before == sig_after and mask_before == mask_after
fix   = count(sig_before[j] != 0 and sig_after[j] == 0)
break_ = count(sig_before[j] == 0 and sig_after[j] != 0)
```

Parameter	Research	Product
Basis cap	48 (runtime studies at 24 and 16)	24
Basis seed	int(md5(id)[:8], 16)	fnv1a32(id)
Equality	1e-6 deep equality	1e-6 deep equality
Per-call timeout	0.25 s	wall budget 2 s total; timed-out dims x
Mutant sampling	<= 24/problem, <= 4/family	not shipped
Skip rule	fewer than 5 evaluable probes (48 problems)	same rule
Walk policies	climb / random / revert, 699 walks	not shipped

Frozen research constants. All seeds are deterministic. The product deviates in one documented way: it seeds the

basis shuffle with 32-bit FNV-1a over the problem id instead of the research md5 prefix, so the product does not claim bit-identical bases with the corpus study (see Section 10).

> *What the measurement is relative to — Every rate is relative to the frozen perturbation set, the deterministic basis, the 1e-6 equality semantics, and the eleven-family edit model. Enlarging the basis should find at least as much divergence, so we expect the invisibility rate measured here to be a lower bound for a wider basis; that expectation is not separately measured in this study, and the sibling paper in this series withdrew an analogous monotonicity claim after a counterexample was found, so it is stated as an expectation rather than a property. None of these numbers is a property of learners.*

4. Census: what the hidden basis sees

The clean run processed all 5,730 exercises; 48 have fewer than five reference-evaluable probes and are skipped, leaving 5,682 analyzable problems. Mutant sampling generated 88,357 single-edit mutants, of which 14,534 (16.45%) pass every shipped test. The hidden basis sees 6,574 of those passers (45.23%) and misses the other 54.77%; 9,041 mutants (10.23%) are invisible on the basis, and 2,523 problems (44.40%) carry at least one test-passing, hidden-visible slip. The test-passing rate is close to wave 41's 16.50% (16.45% here), which is a useful coarse consistency check but not a replication: the two runs sample mutants under different caps (at most 24 per problem here, at most 36 there) over slightly different analyzable sets (5,682 against 5,721), so the two percentages are not estimates of the same quantity and their 0.05-point proximity should not be read as agreement.

Metric	Value	95% CI
Analyzable problems / corpus	5,682 / 5,730 (48 skipped)	-
Sampled single-edit mutants	88,357	-
Invisible on the 48-probe basis	9,041 (10.23%)	[10.03, 10.43]
Test-passing mutants	14,534 (16.45%)	[16.21, 16.70]
Of those, hidden-visible	6,574 (45.23%)	[44.42, 46.04]
Problems with a visible silent slip	2,523 / 5,682 (44.40%)	[43.12, 45.70]
Problems with a test-passing mutant	3,600 (63.36%)	-
Problems with all sampled mutants visible	2,818 / 5,627 (50.08%)	-
Reference flakes (fresh-copy protocol)	0 / 5,682	-
Arg-mutating references	8 / 5,682 (0.14%)	-
Cosmetic / rename churn not zero	0 / 5,682; 0 / 4,680 applied	-

Fresh full-corpus census (clean run, 5,730 records, one per problem). The 48 skipped problems have fewer than five reference-evaluable probes. Problem-level rates condition on problems that generated at least one evaluable mutant (55 problems generated none). Confidence intervals are 95% on the binomial rate over the stated denominator. Every count is from clean/census_clean.jsonl, produced by a truncating writer with no resume path.

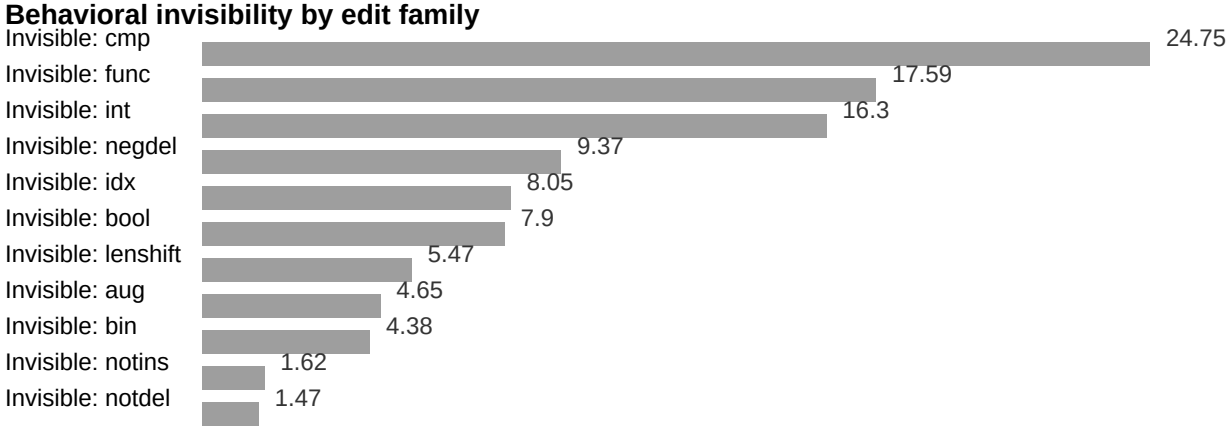
Family structure reverses the pre-registered expectation. Comparison swaps are the blindest family: 24.75% of cmp mutants are invisible, because changing <= to < or == to != often changes nothing on the generated inputs; deletion families are the least blind (notdel 1.47%, notins 1.62%). int, the other family predicted to be nearly fully visible, sits at 16.30%. The prediction H2b is falsified, and

per-family labels were removed from every learner surface as a consequence: family rates are a corpus fact about this edit model, not a diagnosis of a learner.

Family	Generated	Invisible	Rate
cmp	7,740	1,916	24.75%
func	898	158	17.59%
int	23,830	3,884	16.30%
negdel	1,751	164	9.37%
idx	17,424	1,402	8.05%
bool	1,670	132	7.90%
lenshift	8,258	452	5.47%
aug	2,601	121	4.65%
bin	15,267	669	4.38%
notins	8,170	132	1.62%
notdel	748	11	1.47%
Total	88,357	9,041	10.23%

Invisibility by single-edit family (cmp is the blindest). Generated mutants, mutants invisible on the 48-probe basis, and rate, sorted by rate. The pre-registered prediction H2b expected cmp and int to be the least invisible and the deletion families the most; the ordering is reversed, so H2b is reported as falsified. Comparison swaps are the blindest family because <= versus < often changes nothing on the generated inputs. Family labels are a corpus-engineering artifact and never appear on a learner surface.

Behavioral invisibility by edit family



Share of generated single-edit mutants that leave the 48-probe signature unchanged. cmp leads at 24.75% and notdel trails at 1.47%; the pre-registered prediction that cmp would be among the least invisible is falsified. The figure is the census fact behind the always-visible caveat that the hidden checks are a sample.

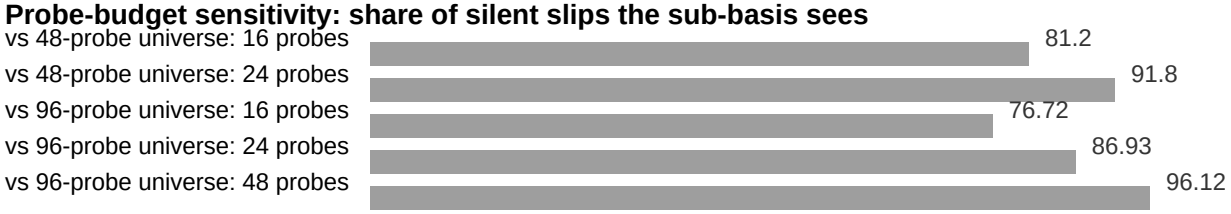
Two denominator corrections matter for reading the tables. 55 of 5,682 problems generated no evaluable mutant at all, and problem-level rates condition on the 5,627 that did; the all-sampled-mutants-visible share is therefore 2,818 / 5,627 = 50.08%, not the 49.6% printed in the first draft with the wrong denominator. Within problems that do have mutants, sensitivity (the mean fraction of a problem's sampled mutants that change the signature) is p10 0.283, median 0.551, p90 0.838.

A smaller basis still sees most of what the research basis sees. Against the 48-probe universe, a 16-probe basis recovers 81.2% of the visible silent slips and a 24-probe basis 91.8%. Against a stricter 96-probe reference on a 600-problem stratified sample, the 48-probe basis sees 96.12% (669/696) of the reference's test-passing slips, the 24-probe basis 86.93%, and the 16-probe basis 76.72%. Sign-level agreement is stronger than coverage: at the step level the 16- and 24-probe bases preserve the sign of the 48-probe impact on 96.19% and 97.83% of steps in the test-positivity sense, and on 92.40% and 95.87% in the full-sign sense; the 24-probe basis estimates the 48-probe agreement rate within +/-0.10 for 88.4% of steps (median error 0.024). The product uses 24 probes because these bounds suffice for a count, not a verdict.

Prediction	Threshold	Observed	Status
H1a fresh-copy flake-free	kill < 99.5%	0 / 5,682 flakes	survived
H2a basis-invisible mutants	kill > 25%	10.23% (9,041 / 88,357)	survived
H2b cmp and int least invisible	kill: ordering reversed	cmp most invisible (24.75%)	falsified
H3a hidden-visible test-passers	kill < 30%	45.23% (below the 50% target)	failed (near)
H3b problems with a visible slip	kill < 20%	44.40% (2,523 / 5,682)	survived
H5a same-edit sign concordance	kill < 60%	84.00% / 82.23% held-out, AUC 0.829 / 0.797	survived held-out
H5b >= 2x next-test regression	kill < 1.3x	1.14 [0.88, 1.47]; 1.36 [1.04, 1.79]	falsified
H6b cold sets family-concentrated	kill < 25%	1.50% / 0.46% under last k = 3	falsified
H8 24-probe latency	kill > 200 ms p99	CPython p99 0.49 ms; Pyodide median 0.271 ms	survived

Pre-registered predictions: what survived and what failed. Prediction and kill thresholds were written before the clean run. Four predictions failed or reversed and are reported as failures, not smoothed: H2b (family ordering reversed), H3a (near miss), H5b (risk ratio below the 2x threshold), and H6b (cold sets almost never family-concentrated). A failed prediction removes copy, not a measurement.

Probe-budget sensitivity: share of silent slips the sub-basis sees



A 16- or 24-probe basis compared with the 48-probe research basis and with a 96-probe reference; the 48-probe basis sees 96.12% (669 / 696) of the 96-probe universe's test-passing slips, the 24-probe basis 86.93%, and the 16-probe basis 76.72%. The shipped product uses 24 probes and reports counts, never a verdict; the 48-probe numbers are the research basis, not a product promise.

5. Edit walks: ghosts, concordance, and prediction

The walk study simulates a learner editing a broken program for ten to twelve steps under three

policies: climb, random, and revert. Exact-signature ghost edits, after the corrected definition, are 48.37% of climb steps, 51.16% of random steps, and 33.98% of revert steps. Almost all ghosts are the same wrong behavior repeated: the walks start broken and the edits keep the program broken in exactly the same way. The old Churn = |Fix| + |Break| definition reported higher rates and, worse, miscounted 6.86-10.44% of its own 'ghosts' - edits that actually flipped a probe from one wrong state to another. Only the corrected definition is used anywhere in this paper or the product.

Policy	Exact-signature ghost (reported)	Old Churn = 0 (superseded)	Old ghosts with changed states
climb	48.37% (1,127 / 2,330)	51.93% (1,210 / 2,330)	6.86% (83 / 1,210)
random	51.16% (1,192 / 2,330)	57.12% (1,331 / 2,330)	10.44% (139 / 1,331)
revert	33.98% (950 / 2,796)	36.77% (1,028 / 2,796)	7.59% (78 / 1,028)

Ghost edits by policy, corrected definition (699 walks). An exact-signature ghost is an edit whose text changed while the signature and the visible test bitmask are byte-identical. The old Churn = |Fix| + |Break| definition is superseded: under it, 6.86-10.44% of its 'ghosts' had actually flipped a probe from one wrong state to another (for example 2 -> 1). The corrected rates are the reported ones.

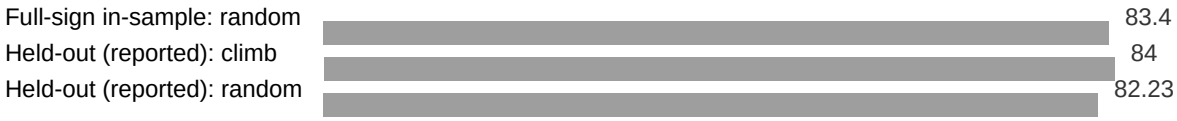
The sign of a hidden delta agrees with the sign of the graded test delta far more often than chance - but only in-sample. On the full-sign statistic used throughout this section, in-sample concordance is 87.78% for climb and 83.40% for random; the positive-only in-sample rate of the first draft (90.20% climb, AUC 0.903) conditions on steps with a nonzero test delta, is a different statistic on a different denominator, and is not used as a reported result. Under the leakage-controlled protocol, with the basis built only from odd-index tests and the delta scored on even-index tests, held-out concordance is 84.00% (AUC 0.829) for climb and 82.23% (AUC 0.797) for random over 18,191 independent steps. The like-for-like drop is 3.78 points for climb and 1.17 for random. The correct reading is that the basis is a genuinely informative proxy for the graded contract, and not informative enough to state a direction to a learner. That is precisely why the product reports whether the signature changed and how many dims changed, and never whether the edit improved the program.

Estimate	climb	random
Full-sign in-sample concordance	87.78%	83.40%
Lag-free in-sample, all steps	89.57%	88.37%
Held-out concordance (odd-test basis, even-test scoring)	84.00% (AUC 0.829)	82.23% (AUC 0.797)
Held-out steps	18,191	18,191

Same-edit sign concordance: in-sample versus held-out. The held-out protocol builds the basis from odd-index shipped tests only and scores the delta on even-index tests, over independent walks (n = 18,191 steps). Held-out performance is what the paper and the product use. The estimand is the full-sign rate: the sign of the hidden delta against the sign of the test delta on every step where both are defined. The positive-only in-sample rate of the withdrawn draft (90.20% climb, AUC 0.903) restricts the same comparison to steps with a nonzero test delta, so it conditions on a subset of steps and is not on the same scale as any held-out row here; it is listed only to record that it exists. Comparing it to a held-out full-sign figure would mix a conditional rate with an unconditional one, and the 6.20-point gap it suggests is 2.42 points of metric change rather than leakage. The like-for-like drop is 87.78 to 84.00 (3.78 points) for climb and 83.40 to 82.23 (1.17 points) for random.

Sign concordance: in-sample versus held-out





Full-sign in-sample concordance (87.78% climb, 83.40% random) against the held-out protocol that builds the basis from odd-index tests and scores on even-index tests (84.00%, AUC 0.829; 82.23%, AUC 0.797). Both rows are the same full-sign statistic, so the in-sample drop is comparable: 3.78 points for climb (87.78% to 84.00%) and 1.17 points for random (83.40% to 82.23%), the measured cost of removing the leak. The held-out bars are the reported numbers. The withdrawn draft's 90.20% positive-only in-sample rate is deliberately not plotted: it conditions on steps with a nonzero test delta, and plotting it beside a full-sign held-out bar would put two different statistics on one axis.

The next-edit version of the question is where the forecasting story dies. Breaking hidden dimensions raises the risk of a next-attempt test regression by 1.14x (95% CI [0.88, 1.47]) under climb and 1.36x ([1.04, 1.79]) under random; the random arm's interval excludes 1, but both are far below the pre-registered 2x threshold. The next-improvement AUC is below chance (0.308 and 0.359) and the next-regression AUC is weak (0.667 and 0.626). The honest use of the delta is retrospective attribution - this edit did change or did not change the measured behavior - not a prediction of what happens next.

Quantity	climb	random
Next-improvement AUC	0.308	0.359
Next-regression AUC	0.667	0.626
Risk ratio for next-attempt test regression	1.14 [0.88, 1.47]	1.36 [1.04, 1.79]
Regression risk, broke vs did not break hidden dims	15.98% vs 14.00% (n = 676 / 1,421)	14.76% vs 10.83% (n = 657 / 1,440)
Always-fix baseline among test-changing steps	51.6%	41.6%
Always-break baseline among test-changing steps	48.4%	58.4%

What a hidden delta does and does not predict. Next-improvement and next-regression AUCs are for the next attempt after a hidden-impact event; the anti-correlation direction stands, and the previously quoted 0.241 anti-correlation figure is not reproduced. The risk ratio is a risk ratio, not an odds ratio. Trivial baselines are computed among test-changing steps only.

```
Next-test-regression risk ratio (H5b)
RR = P(test regression at next attempt | hidden dims broken) / P(test
regression | hidden dims not broken) = 1.14 [0.88, 1.47] (climb); 1.36
[1.04, 1.79] (random).
```

A risk ratio, not an odds ratio; the earlier 'odds' wording is withdrawn. Both intervals sit below the pre-registered 2x threshold, so H5b is falsified and no forecasting copy ships.

> *The number the product uses — The product uses only the identity part of the delta: same signature and same test mask means 'no change on N hidden checks'; a different signature means 'k of N hidden checks changed'. It uses no sign, no direction, and no held-out classifier, because the held-out numbers (84.00 / 82.23, AUC 0.829 / 0.797) support a proxy statement and not a directional one.*

6. Residuals: a negative result and a cut

The first draft defined the cold set on the last k = 3 attempts but measured it over every version of each walk. Those are different objects, and the difference is large. Under the walk-long object, 49.79% of climb walks and 74.68% of random walks end with at least four cold dims; under the formal last k = 3 object the shares rise to 85.84% and 93.56%, because a three-attempt window is easier to keep fully wrong than a ten-step history.

Object	climb, >= 4 cold dims	random, >= 4 cold dims	climb concentration	random concentration
Walk-long (what the first draft measured)	49.79%	74.68%	1.72%	1.15%
Formal last k = 3 (the defined object)	85.84%	93.56%	1.50%	0.46%

Cold sets: walk-long object versus the formal last-three-attempt object. The formal cold set is defined on the last k = 3 attempts; the walk-long analysis averaged over every version of each 10-12 step walk. The two objects disagree, and the residual product story was cut for exactly this reason. H6b is falsified under either object: a single family covering at least 60% of the cold set with at least 1.5x excess over its basis share almost never occurs.

The pre-registered family-coherence prediction H6b is dead under both objects. A single perturbation family covering at least 60% of the cold set with at least 1.5x excess over its basis share occurred in only 1.72% (walk-long) and 1.50% (formal) of climb walks, and 1.15% / 0.46% of random walks, against a predicted 40%. The aggregate enrichment is real but post hoc and modest: under the shipped within-walk permutation null (300 draws, seed 12345) on the formal object, empty is under-indexed at z = -11.8, zero1 at -3.8, and zero at -3.7, while reverse (+3.1), inc (+2.5), inc2 (+2.5), sort (+1.9), dup (+1.8), and extend (+1.7) are over-indexed. The walk-long recompute points the same way (empty -11.7, inc +4.6, reverse +4.1). The old z-values (empty -20.5, inc +8.2, reverse +7.1) came from a null that was never shipped and are withdrawn.

The synthetic routing result is a model, not evidence: with a mastery gain proportional to the visible family slips a problem exposes and a 0.35 scaling factor, a residual-targeted greedy policy reaches mastery in a mean of 3.9 problems against 100.3 for a fixed category track. The model was chosen, not measured; the number is an upper bound on what content routing could buy if practice credit really scaled with discriminating coverage. It is not a measurement of human learning and it is not shipped.

> *Residual coordinates and review routing are cut — Residual cold sets, residual family profiles, and residual-targeted review routing are cut from the product permanently: the defined object (last k = 3) was never the measured object, H6b is dead under both objects, and any learner-level blind-spot copy ('you never diverge on empty inputs') would outrun the in-model aggregate. The aggregate enrichment survives only as a corpus fact about the edit model, stated post hoc.*

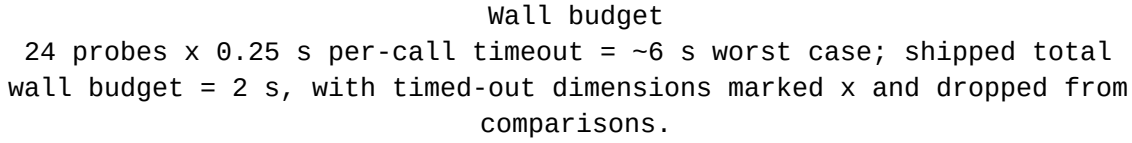
7. Cost and storage

Runtime cost is small but not free, and the difference matters for where the work happens. A 24-probe signature costs a median of 0.271 ms in Pyodide (p95 0.638 ms, max 50.6 ms), about 2.45x the verifier's CPython median (0.111 ms), and well above the original audit's 0.07 ms median.

Building the basis is the expensive part: it exceeded 250 ms on 2 of 200 sampled problems and reached 3.07 s on a reference-timeout case, because it evaluates the reference on every candidate probe. The product therefore builds the basis lazily, once per problem, and caches by a basis fingerprint; it never rebuilds per Run.

Quantity	Value	Note
CPython 24-probe signature, original audit	median 0.07 ms	p99 0.49 ms, max 2.66 ms (n = 200)
CPython 24-probe signature, verifier method	median 0.111 ms	second independent timing
Pyodide 24-probe signature	median 0.271 ms	p95 0.638 ms, max 50.6 ms; ~2.45x the verifier CPython median (0.111 ms)
Basis build cold-start	max 3.07 s	> 250 ms on 2 / 200 sampled; reference-timeout case
Per-call timeout	0.25 s	counted as state 2 where the reference succeeded
Total wall budget per Run	2 s	enforced in the harness via time.monotonic
Stored signature (product)	24 B	24 ASCII chars + <= 64 B metadata per problem
48-trit signature serializations	144 B / 48 B / 12 B	JSON ints / one char per trit / 2-bit packed

Cost, wall budget, and storage. Basis build is not free and must happen lazily once per problem, not on every Run; 24 probes times a 0.25 s per-call timeout is a ~6 s worst case, so the shipped harness enforces a 2 s total wall budget and marks timed-out reference dims x (dropped from comparisons). Storage: the earlier ~100 B/problem figure is withdrawn; the product stores a 24-character signature plus bounded metadata.



The budget is enforced inside the Python harness with time.monotonic plus a per-call line budget raised as an exception learner code cannot catch, so a hostile or nonterminating submission cannot extend a run. A hung worker remains a documented residual risk, recovered by reload.

Storage is bounded and small. The product stores one signature per problem: 24 ASCII characters (24 B) plus at most 64 B of metadata (basis fingerprint, code hash, test mask, timestamp). For the 48-trit research signature, JSON integers cost 144 B, one character per trit costs 48 B, and 2-bit packing costs 12 B; the earlier ~100 B/problem figure is withdrawn as inaccurate. The opt-in replay shelf stores at most three saved sources per problem, each at most 4 KB, with at most 48 total and LRU eviction; sources are saved explicitly, never automatically.

8. Product implications and safety

What ships is deliberately smaller than what was measured. The Ledger is a standalone, opt-in practice route: an index that lists attempted problems and a workspace where the learner edits and runs a program. The Ledger builds its hidden basis from the exercise's own tests, executes the reference and the submission locally under the fresh-copy protocol, and after each run reports counts only - how many hidden checks changed, whether the signature is identical (a no-op edit),

and whether the visible tests changed. It stores one signature per problem, never sources, never families, and never a verdict. The companion 'Your bugs' shelf lets the learner save up to three versions per problem, at most 4 KB each and 48 total, and replay them against the visible tests plus the hidden delta count.

- Changed: 'This edit changed k of N hidden checks.'
- No-op: 'No change on N hidden checks. This edit changed text, not the behavior the checks measure.'
- Always-visible caveat: 'The hidden checks are a sample; an edit can still matter without changing them.'
- Privacy: 'The Ledger stores one signature per problem in this browser. Nothing is synced. You can clear it at any time.'
- Never: 'correct', 'wrong', 'no behavioral change', 'you fixed/broke', 'blind spot', family names, or accuracy percentages.

Cut	Why
In-problem delta card after failed runs	hidden-oracle content in the solve flow; touches ProblemView; nothing in the shipped claim needs it
Fix/break direction ('fixed 5, broke 1')	held-out 84.0 / 82.2 (AUC 0.829 / 0.797) is not strong enough for directional copy
Residual cold sets, profiles, review routing	the formal object (last k = 3) was never the measured object; H6b dead under both; coherence 1.5% / 0.5%
Per-family labels on any learner surface	H2b family ordering is wrong; families are a corpus-engineering artifact
Build-time mining or precomputed per-problem bases	runtime is negligible (0.271 ms per signature Pyodide median); build lazily, cache by content hash
Any hidden-oracle verdict or 'am I done?' meter	blocker; untestable offline against learner intent
Grading, review, certificate, badge, streak, XP, sync coupling	red lines; the route writes only its own two local keys
Storing reference signatures or probe answers	only the learner's last signature is stored; the reference is recomputed locally per run

Cut from the product, with reasons. The count-only claim does not need any of these, and two of them (direction and residual routing) failed or were demoted in the measurements. Every cut is enforced by source-scan tests in the repository, not by editorial discipline.

> *The claim sentence (the ceiling for all product copy) — In an opt-in, standalone practice view, the Behavioral Delta Ledger reports whether an edit changed the program's behavior on up to 24 deterministic hidden checks derived from the exercise's own tests - as a count ('k of N hidden checks changed') and a no-op statement ('no change on N hidden checks') - never which check changed, never whether the edit improved the program, and never as a grade, review signal, certificate input, or synced datum.*

The safety rule is mechanical. The Ledger route imports no grading or progress-writing module and writes only two device-local keys through raw read/write helpers; no store id, no sync registry entry, and no backup-included key exists for it, so it cannot ride a backup or a remote sync. A source-scan

test asserts all of this, asserts that the ledger tree never imports the problem view, the spaced scheduler, the review queue, the certificate module, or any state-writing progress function, and asserts that no ledger source calls a save, mark, or record function outside its own two keys.

The count-only result shape (product engine)

```
export interface BdlRunResult {
  /** One char per probe: "0" agree, "1" wrong value, "2" raise/timeout, "x" no
reference answer. */
  readonly sig: string;
  /** One char per shipped test: "0" fail, "1" pass. */
  readonly mask: string;
  readonly passed: number;
  readonly total: number;
}

export interface BdlDelta {
  readonly changed: number;    // positions where sig chars differ
  readonly comparable: number; // positions comparable on both runs
  readonly sigEqual: boolean;  // same length and every char equal
  readonly maskEqual: boolean; // masks byte-identical
}
```

9. Reproducibility

> *Limitation: what is absent from the release, and what the gate actually checks — Neither the census engine (bdl_engine_clean.py), nor its census_clean.jsonl output, nor analysis/recompute.py, nor the walk and cold-set artifacts is in this release. Absent are therefore every full-corpus count in Table 1 (88,357 / 9,041 / 14,534 / 6,574 / 2,523 / 3,600 / 2,818 / 5,627 / 55), the per-family invisibility table, the 699-walk study, the leakage protocol's 18,191 steps, the sub-basis sensitivity figures, the permutation null, the latency audit, and the synthetic routing model. The permanent gate consequently checks something much narrower than the headline. What it recomputes on every run: the analyzer on a deterministic stratified 240-problem sample (4.2% of the corpus), two byte-identical runs of it, the flake, cosmetic-churn, and rename-churn rows, and the twelve shipped harness fixtures under CPython. What it compares against constants rather than recomputing: the full-corpus counts above, and even for those the exact check runs only when the census artifact is supplied, which CI does not do. The sampled check is a band, not the census: on the last run the sampled hidden-visible rate was 42.33% against the published 45.23% and passed on a printed tolerance of 10.07 points, and problems-with-a-visible-slip sampled 45.34% against 44.40% on 9.92 points. A reader should treat the full-corpus rates as independently re-derived on a 240-problem sample, not as reproduced. Three of the four sampled rates landed within 0.6 points; the fourth was 2.9 points off and still passed. Whether the paper is right or the band is too wide is not settled by anything in this repository.*

> *Recomputed versus replayed — A gate that recomputes a value and compares it with a stored constant proves that the pipeline is deterministic on this corpus and this machine; it does not prove the measurement is correct, and where the expected constant was produced by the same code path the comparison can only catch drift. Applying that distinction here: exact and recomputed are the twelve harness fixtures (each built by the shipped buildBdlHarness, run under CPython, and asserted on its parsed signature and mask), the two committed spot records al-345 and ds-074 (compared against embedded SHA-256 digests produced with CPython 3.13), and the determinism row asserting*

> *that two runs inside one invocation are byte-identical. Band-checked against stored constants are the five sampled rates. Replayed only, with no independent check in this repository, are every full-corpus count, the family table, the walk-study rates, the held-out concordance, the sensitivity figures, the permutation z-values, the latency numbers, and the routing model. Nothing in the gate compares against an external oracle of the kind REPROGPU's Philox and FIPS vectors provide, so nothing in this paper rests on one.*

The census procedure needs no network, no model, and no new dependency, but the engine that produced it is not part of this release. The cited artifact is the clean run under the wave-42 engineer scratch directory: clean/bdl_engine_clean.py wrote clean/census_clean.jsonl with a truncating writer and no resume path, producing 5,730 fresh records, one per problem. Neither the script, the JSONL, nor the scratch directory is in this repository, and the derived statistics come from analysis/recompute.py, which is also absent. The earlier append-resumed artifact is retired: it carried two stale pre-rename-fix records (al-345, ds-074, both with churn_rename null); the fresh records for those problems have churn_rename 0 and identical mutant counts, and every headline total is unchanged. Every number in Sections 4 to 6 is therefore a record of that run rather than an artifact a reader can regenerate.

Reproducing the reported results

```
# the full-corpus census engine is NOT in this release, so these two
# commands cannot be run from a clean checkout; they record how the reported
# numbers were produced
python3 clean/bdl_engine_clean.py problems.json clean/census_clean.jsonl
python3 analysis/recompute.py > analysis/recompute.out

# the permanent Python-backed harness gate (wired as verify:bdl in CI).
# It re-runs the analyzer on a deterministic stratified 240-problem sample and
# requires the four headline rates to fall within 3 sigma of the stored values.
bun run scripts/verify-bdl.ts

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

- Corpus: 5,730 problems; skip a problem when it has fewer than five reference-evaluable probes (48 skipped); require the ast.unparse(ast.parse(solution)) round trip to pass the shipped tests.
- Basis: first three shipped inputs, one argument perturbed at a time, shipped inputs excluded, deduplicated, shuffle seed int(md5(id)[:8], 16), cap 48 (research) / 24 (product), reference-evaluable only.
- Signatures: 1e-6 deep equality, 0.25 s per-call timeout, fresh deep copy of probe arguments on every call.
- Mutants: at most 24 per problem and 4 per family over eleven families, target order shuffled by the problem seed.
- Walks: stride-24 sample, 233 walks per policy, 10-12 steps, walk seed md5(id)[:8] + {climb:1, random:2, revert:3}.
- Permutation null: within-walk, 300 draws, seed 12345. Synthetic cohort: seed 20260918, 400 learners. Latency audit: 200 problems, seed 42.
- Derived statistics, recorded from the clean run and stored as constants in the permanent gate (which compares them exactly only when the census artifact is supplied, and CI does not supply it):

88,357 mutants; 9,041 invisible = 10.23% [10.03, 10.43]; 14,534 test-passing = 16.45% [16.21, 16.70]; 6,574 hidden-visible = 45.23% [44.42, 46.04]; 2,523 / 5,682 visible silent slips = 44.40% [43.12, 45.70]; 0 / 5,682 flakes. In the default self-contained mode the gate instead re-runs the analyzer on a deterministic stratified 240-problem sample and requires the four headline rates to fall within 3 sigma of the stored values, printing the tolerance for each row.

- Permanent gate: scripts/verify-bdl.ts plus scripts/py_bdl_verify.py emit harnesses for fixed fixtures (agree, wrong value, raise, reference timeout), assert the parsed sig and mask exactly, assert the wall-deadline path marks x, and exit non-zero on any mismatch.

> *Seed deviation, stated in the artifact — The research study seeded the basis shuffle with an md5 prefix; the product uses 32-bit FNV-1a over the problem id. The product therefore does not claim bit-identical bases with the corpus numbers; a build-side QA run can recompute FNV-seeded bases to close the gap. This deviation is recorded here and in the engine, not hidden.*

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, and no observational data of any kind: every walk, ghost rate, concordance, and routing result is a simulated edit policy, a synthetic cohort, or a corpus statistic, and the section below that says so is the one a citation should carry. (2) No external replication. Nothing in this paper has been reproduced by anyone outside this project, and the wave-41 comparison in Section 4 is a consistency check between two different estimators rather than a replication, for the reasons given there. (3) The full-corpus census is not reproducible from a checkout. The engine, its JSONL output, and the analysis script are absent, and CI substitutes a 240-problem stratified 3-sigma band check whose printed tolerances reach 10.07 points on a headline row; Section 9 lists exactly which numbers the gate recomputes, which it replays, and which it does not touch. A gate that compares a recomputed value with a stored constant proves determinism, not correctness, and for the full-corpus counts that is the whole of what the gate does. (4) No experiment here measures a generative-AI tutor. The external field evidence on generative tutoring is about other systems and other designs; nothing in Sections 4 to 7 bears on whether a generative tutor produces no-op edits, on the counts this paper defines, or on their value to a learner.*

The first limitation bounds every learner-facing number in this paper: there is no human data. Every walk, ghost rate, concordance, and routing result is a simulated edit policy, a synthetic cohort, or a corpus statistic. Whether real learners make no-op edits at these rates, whether a count card changes behavior, and whether the replay shelf helps retention are all unmeasured. The first human experiment proposed here is deliberately small and local: log consecutive signature pairs in the opt-in route and compare the observed no-op distribution with the simulated one. Nothing in this paper should be cited as a learner-error rate. The programming-trace literature is the natural home for the object this paper studies, and it was not surveyed systematically here, so no priority claim is made for the trace features or the delta object.

- No human data: every learner number is simulated; the shadow log is the first proposed experiment.
- Probe blindness is real and measured: 10.23% of sampled single-edit slips are invisible on the 48-probe basis, and 54.77% of test-passing mutants are invisible to both the tests and the basis; the 24-probe product basis inherits 86.93-91.8% of the visible-slip coverage depending on the

universe.

- Specification ambiguity: hidden probes can leave the stated domain, and a legitimate alternative reading can be counted as divergence; the count is labeled a behavior check and never grades.
- The reference is visible: 'Show solution' ships with every problem, so the Ledger can never certify anything; it is a mirror, not a lock.
- Seed deviation: research md5 vs product FNV-1a means the product basis is not bit-identical with the corpus study.
- The product's 24-probe basis is not measured at corpus scale; the coverage figures come from the 48-probe research basis and stratified samples.
- Wall-time risk: a hostile submission that hits C-level nontermination can bypass line tracing; the 2 s wall budget and input caps bound but do not eliminate this, and a hung worker is recovered by reload.
- No learning-outcome evidence: errorful learning and desirable difficulties motivate the replay shelf, but no study here connects it to outcomes.

Future work follows the failures. The family-ordering failure (H2b) means the edit model's rates are corpus engineering, and a wider perturbation set is the honest way to test them. The held-out collapse (H5a) sets the ceiling for any directional feature; a larger held-out study could raise it, but until then the product stays count-only. The cold-set object mismatch (E2) and its dead coherence prediction (H6b) mean residual routing needs a new measurement before it can be reconsidered, not a new product. The shadow-logging study is the single experiment that would move the most: it would replace the simulated no-op distribution with an observed one and would either support or falsify the premise that behavioral no-ops are common in real learner edits.

> *Honest summary — The Ledger measures a real and large phenomenon - 10.23% of sampled slips are invisible on a 48-probe basis, nearly half of test-passing mutants are visible on it, and simulated walks produce exact no-op edits at 34-51% - and it states a smaller claim than the phenomenon invites: whether your last edit changed anything on up to 24 hidden checks, as a count, on an opt-in standalone route. The direction, the residuals, the forecasts, and the grades were all measured and all removed.*

References

1. DeMillo, Lipton & Sayward 1978, Hints on test data selection: help for the practicing programmer. https://doi.org/10.1109/C-M.1978.218136
2. Jia & Harman 2011, An analysis and survey of the development of mutation testing. https://doi.org/10.1109/TSE.2010.62
3. 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
4. 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
5. Inozemtseva & Holmes 2014, Coverage is not strongly correlated with test suite effectiveness. https://doi.org/10.1145/2568225.2568271
6. Danglot, Vera-Perez, Baudry & Monperrus 2019, Automatic test improvement with DSpot. https://doi.org/10.1007/s10664-019-09692-y
7. McKeeman 1998, Differential testing for software.

https://www.cs.tufts.edu/comp/150FP/archive/bill-mckeeman/DifferentailTesting.pdf
8. Zeller & Hildebrandt 2002, Simplifying and isolating failure-inducing input. https://doi.org/10.1109/32.988498
9. Goldman & Kearns 1995, On the complexity of teaching. https://doi.org/10.1006/jcss.1995.1003
10. Zhu 2015, Machine teaching: an inverse problem to machine learning and an approach toward optimal education. https://doi.org/10.1609/aaai.v29i1.9761
11. 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
12. Cepeda et al. 2006, distributed-practice meta-analysis (839 assessments). https://www.evullab.org/pdf/CepedaPashlerVulWixtedRohrer-PB-2006.pdf
13. Karpicke & Bauernschmidt 2011, absolute versus relative spacing. https://doi.org/10.1037/a0023436
14. Kornell, Hays & Bjork 2009, unsuccessful retrieval with feedback. https://doi.org/10.1037/a0015729
15. Metcalfe 2017, Learning from errors. https://doi.org/10.1146/annurev-psych-010416-044022
16. Jadud 2006, Methods and tools for exploring novice compilation behaviour. https://doi.org/10.1145/1151588.1151600
17. STING 2026, Probe to Generate: Program variant-guided test augmentation for repository-level benchmark repairs. https://arxiv.org/abs/2604.01518
18. Prasad, Greenman, Nelson & Krishnamurthi 2024, Conceptual mutation testing for student programming misconceptions. https://doi.org/10.22152/programming-journal.org/2024/8/7
19. Padurean, Denny & Singla 2025, BugSpotter: automated generation of code debugging exercises. https://doi.org/10.1145/3641554.3701974
20. Padurean et al. 2026, When AI is wrong on purpose: how students respond to buggy GenAI code. https://doi.org/10.1145/3765964.3811667
21. Caraco, Lojo, Verdicchio & Fox 2024, Generating multi-part autogradable faded Parsons problems from code-writing exercises. https://doi.org/10.1145/3626252.3630786
22. Katabench, mutation-graded test-writing exercises (grading documentation). https://katabench.com/docs/grading
23. [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 measurement 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
24. FSRS algorithm explainer (power-law curve, FSRS-6 weights). https://github.com/open-spaced-repetition/awesome-fsrs/wiki/The-Algorithm