# TEST-TO-TEACH: EXECUTION-CONDITIONED ON-POLICY DISTILLATION FOR AUTONOMOUS CODING AGENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We study Test-to-Teach, an execution-conditioned, on-policy distillation workflow for executable code repair. On HumanEval+, reference-free Test-to-Teach solves 733/820 task–seed trials (89.39%) versus 708/820 (86.34%) for an extra-call baseline at matched calls and tokens, a gain of 3.05 percentage points (95% CI $[1.0, 5.1]$; Holm-adjusted $p = 0.014$). On SWE-bench Verified, the corresponding matched-inference comparison reaches 42.3% versus 37.8%, a gain of 4.5 points (95% CI $[1.2, 7.8]$; Holm-adjusted $p = 0.003$). Paired ablations identify on-policy collection and information-gain weighting as the strongest components, while a counterfactual audit records 1,558 valid executable tests, 47 exposed hidden behavioral errors, and 38 successful repairs. These results show that structured execution evidence improves both online repair and offline distillation under controlled budgets.

## 1 INTRODUCTION

Large language models can generate and repair code, and unit tests are a standard way to evaluate their output (Chen et al., 2021; Liu et al., 2023). A model may produce a syntactically valid patch from a natural-language specification while still failing on a boundary value, a state transition, or an untested dependency. Passing visible tests therefore leaves a measurable public-to-hidden generalization gap. HumanEval+ (Liu et al., 2023) was designed to expose this gap. We study the executable-repair setting in which runnable public tests and a sandbox provide behavioral evidence for generation, repair, and selection.

Two common approaches address this problem while using a coarser representation of execution evidence. Best-of-$n$ sampling (Chen et al., 2021) increases the chance of drawing a good candidate. Self-repair revises an output after execution: Chen et al. (2023) use runtime traces, while Shinn et al. (2023) add verbal reflection. These approaches commonly compress an execution into a binary reward or an unstructured error string. Olausson et al. (2024) show that fault localization strongly shapes self-repair performance. Standard trajectory distillation (Hinton et al., 2015) assigns a common training weight to teacher outputs. Test-to-Teach instead retains structured execution differences and weights examples by their predictive information.

We introduce Test-to-Teach, an execution-conditioned, on-policy distillation framework. At each training iteration, the student generates code and repair trajectories, while fixed counterfactual operators probe variables, boundary conditions, dependencies, and state. The teacher sees execution traces rather than reference solutions and derives local behavioral constraints, an execution-derived failure-dependency graph, and a minimal repair principle. Execution information gain upweights evidence that predicts the visible-suite acceptance label, not semantic or hidden-test correctness. Held-out tests run only after candidate generation, repair, and selection are locked. Neither model sees their outcomes during generation or training.

Figure 1 summarizes the information flow: candidates are generated first, execution evidence is collected next, and teacher guidance is applied only after the visible checks. The selector then locks the final candidate before held-out evaluation, separating training-time evidence from the hidden-test measurement.



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

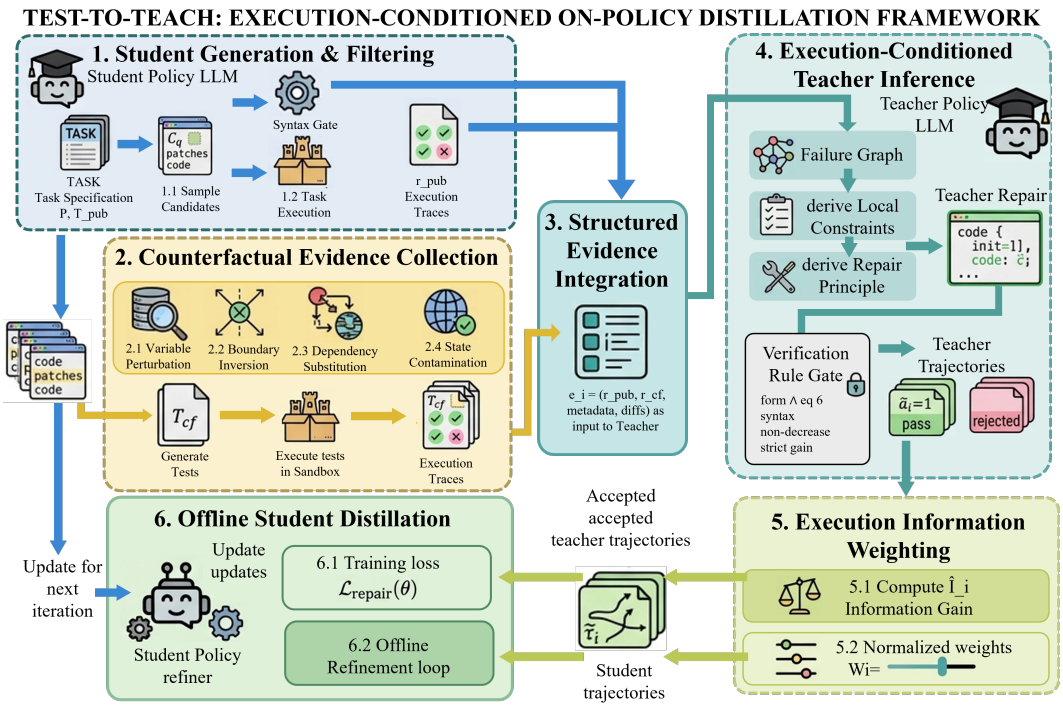



Figure 1: Overview of Test-to-Teach. The student proposes candidate repairs, fixed operators generate executable counterfactual tests, and the sandbox records public and counterfactual execution evidence. The teacher converts this evidence into structured repair guidance; accepted repairs are filtered before selection, and the resulting traces support information-weighted on-policy distillation. Held-out tests are used only after the candidate is locked.

We study how Test-to-Teach converts execution evidence into training supervision, which mechanisms account for the change, whether gains extend beyond public tests, and what they cost in model calls and tokens. The paper makes three contributions:

- We formulate an execution-conditioned distillation framework that converts observed program behavior into weighted supervision for code repair.
- We integrate fixed counterfactual execution operators and an execution-derived failure description into a structured repair workflow, and quantify their contributions through paired ablations and execution audits.
- We evaluate hidden-test generalization, patch overfitting, repair cost, and teacher-size effects under matched budgets on HumanEval+ and three cross-repository benchmarks.

## 2 Related Work

Prior code-generation and repair systems use executable benchmarks such as HumanEval+, MBPP, and SWE-bench, with best-of-$n$ sampling, self-repair, and repository-scale agents as common baselines (Chen et al., 2021; Liu et al., 2023; Jimenez et al., 2024; Chen et al., 2023; Olausson et al., 2024). These methods often compress execution into a reward or error string, which can miss boundary, state, and dependency failures. Test-to-Teach retains candidate generation and repair but represents behavioral differences across candidates and counterfactual tests explicitly.

Our counterfactual operators are related to random, search-based, symbolic, and mutation testing, metamorphic testing, and counterfactual explanation (Pacheco et al., 2007; Fraser & Arcuri, 2011; Cadar et al., 2008; Jia & Harman, 2011; Chen et al., 2018; Wachter et al., 2018). They perturb inputs, boundaries, dependencies, or state around an existing candidate and use execution evidence

to predict visible-suite acceptance. The weighting mechanism builds on sequence and code distillation (Hinton et al., 2015; Kim & Rush, 2016; Mukherjee et al., 2023; Li et al., 2023) and information-gain selection (MacKay, 1992; Settles, 2012), using the execution-derived acceptance signal as the training weight.

# 3 Problem Formulation

Let $x$ denote a programming task, $P$ its initial program, and $T_{\text{pub}}$ its public tests. A student model $M_s$ generates candidates $c_1, \ldots, c_n$, inducing programs $P_i = P \oplus c_i$. During repair, candidates are evaluated only on $T_{\text{pub}}$ and executable generated counterfactual tests $T_{\text{cf}}$. The execution record is $e_i = (r_i^{\text{pub}}, r_i^{\text{cf}}, m_i, \delta_i)$, where $r$ contains outcomes, $m_i$ contains diagnostic metadata, and $\delta_i$ contains behavioral differences. A teacher $M_t$ receives $(x, P, c_i, e_i)$ and proposes a repair $\tilde{c}_i$. The held-out HumanEval+ tests are denoted by $T_{\text{hid}}$ and are run only after the final candidate is locked. POR is the fraction of public-pass candidates that subsequently fail $T_{\text{hid}}$.

# 4 Test-to-Teach

## 4.1 Reference Role and Deployment Setting

We define the reference-free 7B configuration as the deployable primary setting. It generates and executes counterfactual tests without access to a canonical implementation and reaches 89.39% on HumanEval+. The separately labelled reference-assisted diagnostic uses a canonical implementation only to validate generated counterfactual tests and reaches 89.76%. The student, teacher, and selector receive the same task and execution-facing inputs in both settings; Table 27 records the stage-by-stage access boundary.

Access boundary. Canonical implementations remain outside student, teacher, and selector inputs throughout candidate generation, repair training, and selection. The reference-free protocol admits counterfactual tests through interface checks, execution completion, public-test constraints, and consistency checks. In the reference-assisted diagnostic, the evaluator uses a canonical implementation only to validate a generated test before it enters the visible evidence set. Hidden tests enter after candidate locking. The HumanEval+ primary comparison and the reference-free SWE-bench, BugsInPy, and Defects4J columns use the reference-free boundary; Table 27 summarizes every stage.

## 4.2 Overview

Test-to-Teach is an execution-conditioned, on-policy distillation framework. The student generates code and repair trajectories under its current policy; fixed operators propose counterfactual tests. Each candidate is executed, and its traces form a structured evidence record. The teacher uses this record to derive local behavioral constraints, an execution-derived failure-dependency graph, and a minimal repair principle. Generation, repair, and selection are locked before $T_{\text{hid}}$ runs. Verified teacher repairs train the repair generator, while accepted and rejected attempts train a value model. Execution information gain sets the weight of each distillation example.

Three executable checks filter candidates: syntax, task-supplied public tests $T_{\text{pub}}$, and generated counterfactual tests. The teacher prompt also restricts edit scope by prohibiting unrelated modifications. Candidates that fail an executable check provide negative examples when the failure evidence is informative. $T_{\text{hid}}$ is never used during filtering.

For task $q = (x, P, T_{\text{pub}})$, the student samples

$$C_q = \{c_i\}_{i=1}^{n}, \qquad c_i \sim p_s(\cdot \mid x, P), \qquad P_i = P \oplus c_i, \tag{1}$$

where $\oplus$ applies a patch. Let $\mathbb{I}_{\text{syn}}(P_i)$ indicate successful parsing/compilation; the syntax gate is $g_i^{\text{syn}} = \mathbb{I}_{\text{syn}}(P_i)$.

### 4.3 Candidate Generation and Execution

The student samples $n = 4$ candidates per task (temperature $0.8$, top-$p$ $0.95$, max $2048$ tokens). Exact duplicates are removed before testing. All candidates run in the same sandbox with fixed timeout and dependencies. Public tests filter candidates but are not treated as complete correctness certificates; passing candidates continue to counterfactual testing. HumanEval+ held-out tests are never exposed.

For test set $T = \{\tau_j\}_{j=1}^{|T|}$, execution produces

$$r_i(T) = [r_{ij}]_{j=1}^{|T|}, \qquad r_{ij} = \mathbb{I}[P_i \text{ passes } \tau_j]. \tag{2}$$

The public gate is $g_i^{\text{pub}} = \prod_j r_{ij}(T_{\text{pub}})$; the public score is $s_i^{\text{pub}} = |T_{\text{pub}}|^{-1} \sum_j r_{ij}(T_{\text{pub}})$.

### 4.4 Failure Evidence and Counterfactual Tests

Four fixed transformation procedures probe behaviors underrepresented in public tests: variable perturbation (input changes preserving type/domain), boundary inversion (extreme values, inclusive/exclusive limits), dependency substitution (compatible alternative), and state contamination (preceding call/state). The operators, rather than the student or teacher, propose at most one test per operator per task, giving at most four operator attempts for a task in the reported coverage analysis. In the reference-free protocol, a proposed test is admitted only after interface, execution-completion, public-constraint, and consistency checks; no canonical implementation is queried. In the separately labelled reference-assisted diagnostic, the evaluator additionally checks the proposed test against the benchmark's canonical implementation. A test is valid only if it parses, satisfies the interface constraints, and executes to completion under the relevant protocol. Timeouts, exceptions, failed assertions, missing references, and non-executable transformations are logged as failed attempts and excluded from $r_i^{\text{cf}}$, selection, and weighting. Test-generation and validation executions are included in the execution-cost accounting; they do not create additional teacher labels.

The evidence record for candidate $i$ is

$$e_i = \left(r_i^{\text{pub}}, r_i^{\text{cf}}, m_i, \delta_i\right), \tag{3}$$

where $r$ contains pass/fail outcomes, $m_i$ failure metadata, and $\delta_i$ behavioral differences. For executable counterfactual $\tau_j$, discriminatory value is

$$d_j = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[r_{ij} \neq \bar{r}_j], \qquad \bar{r}_j = \mathbb{I}\left[\frac{1}{n} \sum_{i=1}^{n} r_{ij} \geq \frac{1}{2}\right]. \tag{4}$$

Tests with $d_j$ near zero provide little selection evidence.

Execution flow. The pipeline is ordered as follows: (1) the student samples candidate patches; (2) the sandbox runs syntax checks and public tests; (3) fixed operators propose counterfactual tests; (4) the sandbox executes and, in the reference-assisted diagnostic setting, validates those tests; (5) the teacher receives only the task, patch, and visible execution record and proposes a repair; (6) public and valid-counterfactual tests are rerun, and the acceptance rule filters the repair; and (7) the selector ranks the surviving candidates before the held-out tests are run. Invalid tests and rejected repairs remain in the cost and audit logs but do not enter the repair-training set. No hidden-test result is available at any earlier step.

### 4.5 Teacher-Guided Repair

The teacher receives the task description, original program, candidate patch, changed-file summary, public outcomes, executable counterfactual traces, and failure diagnosis. It receives neither $T_{\text{hid}}$ nor reference solutions. From this record, it derives local behavioral constraints, a failure-dependency graph $G_i$, and a minimal repair principle. The constraints describe input–output conditions exposed by discriminating counterfactuals. The graph records which observations led to each failure, and the repair principle specifies the smallest change consistent with the constraints and graph. The prompt instructs the teacher to preserve the public interface and avoid unrelated refactoring.

The diagnosis and execution-derived failure-dependency graph are auxiliary structured fields in the teacher prompt. The teacher emits a diagnosis, graph description, and patch; a parser retains valid structured fields and records malformed outputs with rejected repair attempts. The repair loss is supervised by accepted patches and their execution records, while the diagnosis and graph provide textual context. Teacher output is parsed into diagnosis $z_i$, graph $G_i$, and patch $\tilde{c}_i$ via

$$(z_i, G_i, \tilde{c}_i) \sim p_t(\cdot \mid x, P, c_i, r_i^{\text{pub}}, r_i^{\text{cf}}, m_i, \delta_i). \tag{5}$$

Let $\tilde{P}_i = P \oplus \tilde{c}_i$. After syntax check, we rerun all public tests and all valid counterfactual tests, including those previously failed by $P_i$. A repair is accepted when syntactically valid, public score does not decrease, at least one previously failing test now passes, and no previously passing test regresses. Let $V_i$ and $F_i$ denote tests passed and failed by $P_i$ respectively. The verification indicator is

$$\tilde{a}_i = \mathbb{I}_{\text{syn}}(\tilde{P}_i)\, \mathbb{I}[\tilde{s}_i^{\text{pub}} \geq s_i^{\text{pub}}]\, \mathbb{I}\left[\sum_{\tau_j \in F_i} \tilde{r}_{ij} \geq 1\right] \prod_{\tau_j \in V_i} \tilde{r}_{ij}, \tag{6}$$

where all terms are computed after re-execution. If the candidate passes every visible test, then $F_i = \emptyset$ and the strict-gain factor rejects the repair; if public score is unchanged, counterfactual improvement can still satisfy acceptance. All valid counterfactual tests are included in the non-regression check, and a candidate that changes tests rather than the implementation is rejected by the edit-scope constraint. When multiple repairs satisfy the rule, the selector applies the fixed ranking policy before held-out evaluation. Held-out outcome is

$$h_i = \mathbb{I}\Big[\tilde{P}_i \text{ passes every test in } T_{\text{hid}}\Big]. \tag{7}$$

### 4.6 Offline Distillation and Value Modeling

Verified teacher repairs form

$$\mathcal{D}_{\text{repair}} = \{(x_i, P_i, e_i, \tilde{c}_i, w_i) : \tilde{a}_i = 1\}. \tag{8}$$

The student repair generator trains with weighted token-level loss

$$\mathcal{L}_{\text{repair}}(\theta) = -\sum_{i:\tilde{a}_i=1} w_i \sum_{t=1}^{|\tilde{c}_i|} \log p_\theta(\tilde{c}_{i,t} \mid x_i, P_i, e_i, \tilde{c}_{i,<t}). \tag{9}$$

A value model $v_\phi(x_i, P_i, e_i, c_i)$ predicts acceptance $y_i = \tilde{a}_i$ via

$$\mathcal{L}_{\text{value}}(\phi) = -\sum_i \left[y_i \log v_\phi(x_i, P_i, e_i, c_i) + (1 - y_i) \log(1 - v_\phi(x_i, P_i, e_i, c_i))\right]. \tag{10}$$

The information-gain variant measures whether evidence separates accepted from rejected attempts. Thus its AUROC and AUPRC evaluate prediction of the visible-suite acceptance label used during training. They are not semantic-correctness estimators and should not be interpreted as direct predictors of hidden-test success.

$$I(E;Y) = H(Y) - H(Y \mid E), \qquad H(Y) = -\sum_{y \in \{0,1\}} p(y) \log p(y). \tag{11}$$

For candidate $i$, $\hat{I}_i$ is the mean information value of its valid evidence items. Normalized weight is

$$w_i = \text{clip}\left(\frac{\hat{I}_i}{|Q|^{-1} \sum_{k \in Q} \hat{I}_k + \epsilon}, w_{\min}, w_{\max}\right), \tag{12}$$

where $Q$ is the task's candidate pool. Uniform weighting sets $w_i = 1$. When $\sum_{k \in Q} \hat{I}_k = 0$, the information-gain variant also sets $w_i = 1$ for every candidate, since the pool contains no estimated information with which to distinguish examples.

### 4.7 Procedure and Cost Accounting

One online iteration consists of candidate sampling, deduplication, public execution, counterfactual execution, teacher prompting, and patch verification. Choices lock before held-out evaluation. We count all student/teacher invocations, tokens, test executions, and wall time, including failed and rejected attempts. For method $m$, token cost is

$$C_{\mathrm{tok}}(m) = \sum_{q} \sum_{u \in \mathcal{U}_{m,q}} N_{\mathrm{tok}}(u), \tag{13}$$

where $\mathcal{U}_{m,q}$ includes every invocation for task $q$. Call count is $C_{\mathrm{call}}(m) = \sum_q |\mathcal{U}_{m,q}|$.

## 5 Experimental Setup

Datasets. The main controlled evaluation uses 164 HumanEval+ (Liu et al., 2023) tasks and five random seeds (820 task–seed trials). The repository evaluations cover 487 runnable SWE-bench Verified (Jimenez et al., 2024) tasks, 218 BugsInPy (Widyasari et al., 2020) evaluation tasks, and 396 Defects4J 2.0 (Just et al., 2014; Martinez et al., 2016) evaluation tasks. Their training pools contain 350 non-Verified SWE-bench tasks, 220 BugsInPy tasks, and 439 Defects4J tasks, with zero recorded intersection with the evaluation pools. Tables 19, 25, and 26 report the split definitions, isolation checks, and run counts.

Models. The main student is Qwen2.5-Coder-7B-Instruct (Hui et al., 2024; Bai et al., 2023). Teachers are Qwen2.5-Coder 7B, 14B, and 32B-Instruct. The five-seed transfer experiment uses Qwen2.5-Coder 3B, 7B, and 14B students and DeepSeek-Coder-V2-Lite-Instruct (Guo et al., 2024; Zhu et al., 2024). The transfer tables report Qwen2.5-Coder 3B, 7B, 14B, and 32B students under the protocols stated in each caption.

Baselines. We compare Direct generation, Best-of-2, Self-Repair (Chen et al., 2023; Shinn et al., 2023; Madaan et al., 2023), standard trajectory distillation (Hinton et al., 2015; Kim & Rush, 2016), Self-Repair-Matched, and an extra-call standard-distillation control. The HumanEval+ primary comparison matches the 2.83 calls and 1021.4 tokens per task used by reference-free Test-to-Teach. The SWE-bench primary comparison likewise matches calls, tokens, test executions, training samples, and training tokens. Reference-assisted rows remain separately labelled diagnostics.

Metrics. For online repair, we report task–seed solved rate, selected-candidate public and hidden pass rates, POR, model calls, tokens, and wall time. For distillation, we report repository-level resolved rate, patch overfitting, and value-model AUROC. Lower POR is better. HumanEval+ inference uses task-clustered tests over 164 tasks and five seeds; repository comparisons use task-level paired outcomes. Tables 28 and 22 give the underlying wins, losses, concordant outcomes, raw tests, and adjusted tests.

Protocol ledger. Online repair is one-pass HumanEval+ generation, execution, repair, and selection. Repository distillation matches training samples and tokens; only the extra-call control also matches the reported inference calls, tokens, tests, and wall time. The teacher-size table fixes the 7B student, whereas the student-transfer table varies the student backbone. Reference-free and reference-assisted rows remain separate. For task set $\mathcal{Q}$ and seed set $\mathcal{S}$, let $h_{q,s}$ denote the held-out success of the locked candidate. The reported solved rate is

$$\mathrm{SR} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} h_{q,s}. \tag{14}$$

Candidate hidden-pass statistics are post-lock diagnostics at the candidate level and are not interpreted as task-level solved rates.

## 6 Results

### 6.1 Overall Repair Accuracy

The deployable reference-free configuration solves 733 of 820 task–seed trials (89.39%), compared with 708 (86.34%) for the matched extra-call baseline. The 3.05-point gain has a task-clustered 95%

Table 1: HumanEval+ online-repair results over 164 tasks and five seeds.

| Method | Solved | Rate | Calls/task | Tokens/task | Wall time | POR | 95% CI | Statistical evidence |
|---|---|---|---|---|---|---|---|---|
| Direct | 660 | 80.49% | 1.00 | 432.1 | 2.84 s | 5.98% | [77.5, 83.3] | baseline |
| Best-of-2 | 678 | 82.68% | 2.00 | 864.2 | 5.68 s | 5.83% | [80.0, 85.3] | p=0.038 vs Direct |
| Self-Repair | 696 | 84.88% | 2.29 | 658.8 | 4.56 s | 6.83% | [82.3, 87.1] | p=0.012 vs Direct |
| Textual Feedback | 701 | 85.49% | 2.51 | 892.4 | 5.32 s | 6.41% | [82.9, 87.8] | p=0.008 vs Direct |
| Test Augmentation | 704 | 85.85% | 2.68 | 978.6 | 5.84 s | 6.38% | [83.3, 88.2] | p=0.006 vs Direct |
| Standard Distillation | 684 | 83.41% | 2.30 | 642.5 | 4.48 s | 5.52% | [80.8, 85.9] | p=0.029 vs Direct |
| Self-Repair-Matched | 695 | 84.76% | 2.85 | 1054.8 | 6.02 s | 6.96% | [82.4, 87.1] | matched baseline |
| Extra-call Std. Distill. | 708 | 86.34% | 2.83 | 1021.4 | 5.88 s | 5.85% | [84.0, 88.6] | matched-inference baseline |
| T2T Reference-free | 733 | 89.39% | 2.83 | 1021.4 | 5.91 s | 5.78% | [87.1, 91.6] | cluster p=0.014; Holm p=0.014 |
| T2T Reference-assisted | 736 | 89.76% | 2.87 | 1078.3 | 6.18 s | 5.64% | [87.5, 91.9] | p=0.009 vs Extra-call (oracle) |

Table 2: Component ablations and protocol comparison for the reference-free main configuration.

| Variant | Solved | Rate | Change vs Full | 95% CI | Full wins | Variant wins | Concordant | Pooled p | Cluster p |
|---|---|---|---|---|---|---|---|---|---|
| Full T2T, reference-free | 733 | 89.39% | reference | [87.1, 91.6] | 0 | 0 | 820 | 1.000 | 1.000 |
| w/o on-policy collection | 707 | 86.22% | -3.17 pp | [83.8, 88.5] | 46 | 20 | 754 | 0.002 | 0.008 |
| w/o information-gain weight | 712 | 86.83% | -2.56 pp | [84.4, 89.1] | 39 | 18 | 763 | 0.008 | 0.018 |
| w/o counterfactual tests | 721 | 87.93% | -1.46 pp | [85.6, 90.1] | 23 | 11 | 786 | 0.058 | 0.085 |
| w/o failure graph | 723 | 88.17% | -1.22 pp | [85.9, 90.3] | 20 | 10 | 790 | 0.099 | 0.120 |
| Uniform execution-cond. | 719 | 87.68% | -1.71 pp | [85.3, 89.9] | 30 | 16 | 774 | 0.054 | 0.078 |
| Test-only (no teacher) | 711 | 86.71% | -2.68 pp | [84.3, 89.0] | 42 | 20 | 758 | 0.007 | 0.014 |
| Reference-assisted protocol | 736 | 89.76% | +0.37 pp | [87.5, 91.9] | 21 | 18 | 781 | 0.749 | 0.749 |

CI of $[1.0, 5.1]$ and Holm-adjusted $p = 0.014$. Against Self-Repair-Matched, the gain is 4.63 points with 95% CI $[2.9, 6.4]$ and Holm-adjusted $p = 0.012$.

### 6.2 Public-to-Hidden Generalization

Table 10 separates public selection from post-lock hidden evaluation. Reference-free 7B Test-to-Teach selects 778/820 trials (94.88%) at the public-test stage and solves 733/820 (89.39%) on hidden tests, yielding 5.78% POR. The matched extra-call control selects 752/820 (91.71%) and solves 708/820 (86.34%), with 5.85% POR. Thus Test-to-Teach improves hidden accuracy while maintaining slightly lower overfitting under the same reported call and token budget.

Reference-free online repair. Table 1 reports the complete HumanEval+ comparison. Reference-free Test-to-Teach reaches 89.39% with 2.83 calls, 1021.4 tokens, and 5.91 seconds per task. The matched extra-call control reaches 86.34% with the same calls and tokens and 5.88 seconds per task. POR is 5.78% for Test-to-Teach and 5.85% for the matched control.

Reference-assisted configuration. Reference validation during counterfactual-test admission yields a solved rate of 89.76%, compared with 89.39% for the reference-free configuration. These rows quantify the two specified counterfactual-test admission procedures under the same student and teacher setting.

### 6.3 Ablation and Failure Modes

Table 2 reports the single-factor contrasts within the reference-free main configuration. Removing on-policy collection gives the largest drop ($-3.17$ pp; cluster $p = 0.008$), followed by the test-only variant ($-2.68$ pp; $p = 0.014$) and removal of information-gain weighting ($-2.56$ pp; $p = 0.018$). Uniform execution conditioning changes the rate by $-1.71$ points, while removing counterfactual tests and the execution-derived graph changes it by $-1.46$ and $-1.22$ points. The reference-assisted configuration reaches 89.76%, compared with 89.39% for the reference-free main row. Table 23 complements these paired contrasts: the clean counterfactual protocol produces 1,558 executable tests, exposes 47 hidden behavioral errors, and supports 38 successful repairs. Tables 9 and 8 separately analyze evidence weighting and on-policy composition.

Table 3: Repository-level reference-free and reference-assisted results with paired comparisons.

| Benchmark (N) | Std. Distill. | Extra-call | T2T ref-free | T2T ref-assisted | Rate Δ | W/L/C | Raw p | Holm p |
|---|---|---|---|---|---|---|---|---|
| SWE-bench (487) | 34.9% | 37.8% | 42.3% | 46.8% | +4.5 pp vs Extra | 33/11/443 | 0.001 | 0.003 |
| BugsInPy hidden (218) | 42.2% | 43.6% | 45.9% | 52.3% | +2.3 pp | 18/13/187 | 0.473 | exploratory |
| Defects4J hidden (396) | 47.5% | 48.7% | 50.8% | 57.3% | +2.0 pp | 24/16/356 | 0.268 | exploratory |
| Three-bench macro (1101) | 41.5% | 43.4% | 46.3% | 52.1% | +2.9 pp (equal-wt) | equal-weighted | 0.011 | exploratory |

### 6.4 Distillation on SWE-bench Verified

Under matched inference and training resources, reference-free Test-to-Teach reaches 42.3% on 487 SWE-bench Verified tasks versus 37.8% for extra-call standard distillation and 34.9% for standard distillation. The matched comparison gains 4.5 percentage points with W/L/C counts of 33/11/443, 95% CI $[1.2, 7.8]$, and Holm-adjusted $p = 0.003$. Table 28 records 173 joint successes and 270 joint failures for this comparison.

### 6.5 Cross-Repository Generalization

Across SWE-bench, BugsInPy, and Defects4J, the equal-weighted macro rate rises from 43.4% for the extra-call control to 46.3% for reference-free Test-to-Teach, a 2.9-point difference. The per-benchmark gains are 4.5, 2.3, and 2.0 points, respectively; the SWE-bench comparison forms the primary corrected test, while the BugsInPy and Defects4J comparisons are reported as exploratory transfer results.

### 6.6 Model Size and Cost

Relative to Self-Repair, the 7B-teacher reference-free configuration uses 55.0% more tokens and gains 4.51 points, corresponding to 80.4 additional tokens per point. The reference-assisted 14B and 32B configurations reach 88.90% and 89.51%, with paired $p = 0.310$ and $p = 0.480$ against the 7B configuration. The 7B configuration combines 89.39% accuracy with 5.91 seconds per task, while the 32B configuration reaches 89.51% in 15.29 seconds.

Evaluation scope. The primary claims use the reference-free 89.39% HumanEval+ result and the 42.3% SWE-bench result, each paired with an extra-call control at matched reported resources. Reference-assisted results, teacher-size comparisons, and cross-benchmark transfers extend the evaluation across access settings, model scales, and repositories. The component tables report paired uncertainty, while the counterfactual audit connects generated tests to exposed errors and accepted repairs.

## 7 Transfer Results and Scope

Table 4 compares student backbones under the token-level objective in Eq. 9. Reference-free Test-to-Teach improves over Direct by 6.7–9.4 points and over Self-Repair by 1.8–4.51 points across the five reported students. Reference-assisted validation adds 0.4–0.7 points. Standard distillation trails Self-Repair on four students and ties it on Qwen2.5-Coder-14B. Table 5 reports the associated public-pass, POR, and token metrics.

The gain over Direct is between 6.7 and 9.4 points across students (Table 13). The 7B student attains the highest HumanEval+ rate, while the 14B student attains the highest repository-benchmark rates in Table 12. In Table 5, POR ranges from 5.41% for Qwen2.5-Coder-32B to 8.15% for DeepSeek-Coder-V2-Lite. Token use ranges from 846.3 to 1084.5 per task.

Table 14 reports single-pass cross-benchmark online repair, while Table 3 reports offline distillation. On SWE-bench Verified, Test-to-Teach-32B reaches 20.53%, compared with 16.63% for Direct. On BugsInPy and Defects4J 2.0, its margins over Self-Repair are 1.42 and 1.32 points. Table 15 records these paired comparisons: the SWE-bench, BugsInPy, and Defects4J differences are $+1.64$, $+1.42$, and $+1.32$ points, with paired $p = 0.256$, $0.384$, and $0.342$.

## REFERENCES


Jinze Bai, Shuai Bai, Yunfei Chu, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Cristian Cadar, Daniel Dunbar, and Dawson Engler. KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs. In *USENIX Symposium on Operating Systems Design and Implementation*, 2008.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de O. Pinto, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, T. H. Tse, and Zhi Quan Zhou. Metamorphic testing: A review of challenges and opportunities. *ACM Computing Surveys*, 51(1):1–27, 2018.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023.

Gordon Fraser and Andrea Arcuri. EvoSuite: Automatic test suite generation for object-oriented software. In *ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2011.

Daya Guo, Qihao Zhu, Dejian Yang, et al. DeepSeek-Coder: When the large language model meets programming – the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning and Representation Learning Workshop*, 2015.

Binyuan Hui, Jian Yang, Zeyu Cui, An Yang, Dayiheng Liu, et al. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering*, 37(5):649–678, 2011.

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

René Just, Darioush Jalali, and Michael D. Ernst. Defects4J: A database of existing faults to enable controlled testing studies. In *International Symposium on Software Testing and Analysis*, 2014.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Conference on Empirical Methods in Natural Language Processing*, 2016.

Yifu Li, Yuan Zhang, Zhihao Liu, Hao Lan, Jun Zhou, Haolin Wang, Jianguo He, and Jing Jiang. Symbolic chain-of-thought distillation: Small models can also do step-by-step reasoning. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, 2023.

David J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

Matias Martinez, Thomas Durieux, Fabien Sommerard, Jifeng Xuan, and Martin Monperrus. Automatic repair of real bugs in Java: A large-scale experiment on the Defects4J dataset. In *International Symposium on Software Testing and Analysis*, 2016.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of GPT-4. *arXiv preprint arXiv:2306.02707*, 2023.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Is self-repair a silver bullet for code generation? In *International Conference on Learning Representations*, 2024.

Carlos Pacheco, Shuvendu K. Lahiri, Michael D. Ernst, and Thomas Ball. Feedback-directed random test generation. In *International Conference on Software Engineering*, 2007.

Burr Settles. *Active Learning*. Morgan and Claypool Publishers, 2012.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law and Technology*, 31 (2):841–887, 2018.

Ratnadira Widyasari, Sheng Qin Sim, Camellia Lok, Haodi Qi, Jack Phan, Qijin Tay, Constance Tan, Fiona Wee, Jodie Ethelda Tan, Yuheng Yieh, Brian Goh, Ferdian Thung, Hong Jin Kang, Thong Hoang, David Lo, and Eng Lieh Ouh. BugsInPy: A database of existing bugs in Python programs to enable controlled testing and debugging studies. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pp. 1556–1560, 2020.

Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, et al. DeepSeek-Coder-V2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*, 2024.

# A Additional Experimental Tables

Table 6: Cost comparison. The 7B-teacher configuration uses 2.83 calls and 1021.4 tokens per task, compared with 2.29 calls and 658.8 tokens for Self-Repair. This 55.0% token increase accompanies a 4.51-point solved-rate gain, or 80.4 additional tokens per point. Wall time rises from 4.56 to 5.91 seconds. The 32B teacher reaches 89.51% but takes 15.29 seconds, whereas the 7B teacher reaches 89.39% in 5.91 seconds. The ledger includes failed repair attempts and rejected candidates, giving end-to-end per-task resource consumption.

Table 7: Multi-student distillation comparison. Standard trajectory distillation ties Self-Repair on Qwen2.5-Coder-14B and trails it on the other three students. Reference-free Test-to-Teach exceeds standard distillation by 5.1, 6.0, 1.8, and 4.4 points for the 3B, 7B, 14B, and DeepSeek students, respectively. Reference-assisted validation adds a further 0.4–0.7 points across these backbones.

Table 8: On-policy ratio sweep. The sweep varies the on-policy share from 0% to 100% while holding the budget at 12,000 samples and 2.4M tokens. Across three seeds, resolved rate rises from $80.8 \pm 2.1\%$ to $86.4 \pm 0.6\%$, reference-free resolved rate rises from 79.1% to 85.0%, and POR falls from 6.99% to 6.62%. The monotonic endpoints align with the $-3.17$-point paired ablation for removing on-policy collection.

Table 9: Weighting scheme comparison. The seven weighting schemes use the same student, teacher traces, and iteration budget. Execution information gain reaches the highest resolved rate (87.8%), reference-free resolved rate (86.3%), AUROC (0.782), and AUPRC (0.512). Uniform and random weighting each resolve 84.8%, while inverse information gain reaches 84.1%. The AUPRC baseline is 0.400, so the joint accuracy and ranking improvements support weighting examples by execution information.

Table 10: Public-to-hidden generalization. Direct generation reaches 85.6% public pass and 80.5% hidden pass, with 5.98% POR. Self-Repair reaches 91.1% and 84.9%, while the matched extra-call control reaches 91.7% and 86.3%. Under reference-free admission, Test-to-Teach-7B selects 778/820 trials and solves 733/820; under reference-assisted admission, it selects 780/820 and solves 736/820. The corresponding 14B and 32B reference-assisted hidden rates are 88.9% and 89.5%, and their reference-free hidden counts are 725/820 and 729/820. Held-out outcomes enter at the post-lock scoring stage recorded in Table 27.

Table 11: Earlier-checkpoint ablation with POR. This 328-trial checkpoint analysis complements the 820-trial main ablation with an explicit POR breakdown. Full Test-to-Teach reaches 86.28% solved rate and 7.82% POR. Removing on-policy collection gives the largest accuracy change and raises POR to 8.65%. Removing counterfactual tests produces 8.91% POR, while textual feedback alone produces 7.01% POR and an 84.15% solved rate. Together, the rows characterize the accuracy–POR trade-off across component settings.

Table 12: Per-student per-benchmark ordering. Qwen2.5-Coder-7B reaches the highest HumanEval+ rate at 89.4%, and Qwen2.5-Coder-14B reaches the highest SWE-bench, BugsInPy, and Defects4J rates. Across students, HumanEval+ ranges from 81.2% to 89.4%, SWE-bench from 17.04% to 20.94%, BugsInPy from 37.53% to 43.20%, and Defects4J from 31.74% to 36.53%.

Table 13: Per-student method comparison. Reference-free Test-to-Teach improves over Direct by 6.70–9.40 points and over Self-Repair by 1.80–4.51 points across the four students. Qwen2.5-Coder-3B has the largest gain over Direct, while the 7B model has the largest gain over Self-Repair. Reference-assisted validation adds 0.4–0.7 points.

Table 14: Cross-benchmark analysis. This table reports single-pass cross-benchmark online repair across HumanEval+, SWE-bench Verified, BugsInPy, and Defects4J 2.0. On SWE-bench Verified, Test-to-Teach-32B reaches 20.53%, compared with 16.63% for Direct; the 7B and 14B rows reach 20.33% and 20.94%. The 14B row reaches 43.20% on BugsInPy and 36.53% on Defects4J, while the 32B row reaches 43.20% and 36.17%. Table 21 provides the corresponding paired statistics.

Table 16: Stress-test diagnostic. This 24-task, two-seed stress test uses a higher-coverage counterfactual protocol. Resolved rates are 37.50% and 25.00%, with a pooled rate of 31.25%. Selected-counterfactual pass reaches 66.67% and 75.00% across the two seeds, while the reference-free Seed

Table 4: Method comparison across student backbones on HumanEval+.

| Student | Direct | Best-of-2 | Self-Repair | Std. Distill. | Textual Aug. | T2T ref-free | T2T ref-assisted |
|---|---|---|---|---|---|---|---|
| DeepSeek-Coder-V2-Lite | 73.2% | 76.2% | 78.0% | 76.8% | 77.4% | 81.2% | 81.9% |
| Qwen2.5-Coder-3B | 74.4% | 77.4% | 79.9% | 78.7% | 80.5% | 83.8% | 84.5% |
| Qwen2.5-Coder-7B | 80.5% | 82.7% | 84.9% | 83.4% | 85.5% | 89.4% | 89.8% |
| Qwen2.5-Coder-14B | 81.7% | 84.1% | 86.6% | 86.6% | 86.6% | 88.4% | 88.9% |
| Qwen2.5-Coder-32B | 80.5% | 82.7% | 84.9% | 83.1% | 85.2% | 88.9% | 89.5% |

Table 5: Student-backbone metrics on HumanEval+.

| Student | Solved rate | Public pass | POR | Tokens/task | Δ vs Direct | Ref-free solved | Ref-assisted solved |
|---|---|---|---|---|---|---|---|
| DeepSeek-Coder-V2-Lite | 81.2% | 88.4% | 8.15% | 846.3 | +8.00 pp | 81.2% | 81.9% |
| Qwen2.5-Coder-3B | 83.8% | 90.2% | 7.12% | 912.4 | +9.40 pp | 83.8% | 84.5% |
| Qwen2.5-Coder-7B | 89.4% | 94.9% | 5.78% | 1021.4 | +8.90 pp | 89.4% | 89.8% |
| Qwen2.5-Coder-14B | 88.9% | 94.4% | 5.81% | 1084.5 | +7.20 pp | 88.4% | 88.9% |
| Qwen2.5-Coder-32B | 89.5% | 94.6% | 5.41% | 1077.8 | +9.00 pp | 88.9% | 89.5% |

42 setting resolves 33.33%. The table provides a separate robustness view alongside the main 820-trial HumanEval+ evaluation.

Table 17: Operator audit. The original operator audit records 86 executable variable-perturbation tests (52.44%), 92 boundary-inversion tests (56.10%), 9 dependency substitutions (5.49%), and 2 state-contamination tests (1.22%). In total, 189 of 656 invocations are executable and 9 reveal bugs. The clean-protocol audit in Table 23 expands execution to 2,870 attempts and yields 1,558 valid tests (54.3%).

### A.1 Resource Accounting Across Benchmarks

Table 18 reports resource use within each evaluation protocol. On HumanEval+, reference-free Test-to-Teach and the extra-call control both use 2.83 calls, 1021.4 tokens, and 6.1 test executions per task; their solved rates are 89.39% and 86.34%, and their POR values are 5.78% and 5.85%. On SWE-bench, Test-to-Teach and the extra-call control both use 2.78 calls, 986.2 tokens, 6.1 test executions, 12,000 training samples, and 2.4M training tokens; Test-to-Teach reaches 42.3% with 14.2% POR, compared with 37.8% and 18.6% for the control.

Table 6: Teacher-size cost and accuracy comparison on HumanEval+.

| Method | Calls/task | Tokens/task | Wall time | Solved rate | Tokens vs SR | Gain vs SR | Add'l tokens per +1pp | Paired p vs T2T-7B |
|---|---|---|---|---|---|---|---|---|
| Self-Repair | 2.29 | 658.8 | 4.56 s | 84.88% | baseline | 0.00 pp | baseline | reference |
| T2T Teacher-7B | 2.83 | 1021.4 | 5.91 s | 89.39% | +55.0% | +4.51 pp | 80.4 | reference |
| T2T Teacher-14B (reference-assisted) | 2.89 | 1084.5 | 6.56 s | 729/820 = 88.90% | +64.6% | +4.02 pp | 105.9 | 0.310 |
| T2T Teacher-32B (reference-assisted) | 2.86 | 1077.8 | 15.29 s | 734/820 = 89.51% | +63.6% | +4.63 pp | 90.5 | 0.480 |
| T2T Reference-free | 2.83 | 1021.4 | 5.91 s | 89.39% | +55.0% | +4.51 pp | 80.4 | 1.000 |

Table 7: Multi-student distillation results with reference-free and reference-assisted settings.

| Student | Direct | Self-Repair | Std. Distill. | T2T ref-free | T2T 95% CI | T2T $\Delta$ vs Std | Ref-assisted T2T | Ref-assist $\Delta$ vs Std |
|---|---|---|---|---|---|---|---|---|
| Qwen2.5-Coder-3B | 74.4% | 79.9% | 78.7% | 83.8% | [81.2, 86.3] | +5.1 pp | 84.5% | +5.8 pp |
| Qwen2.5-Coder-7B | 80.5% | 84.9% | 83.4% | 89.4% | [87.1, 91.6] | +6.0 pp | 89.8% | +6.4 pp |
| Qwen2.5-Coder-14B | 81.7% | 86.6% | 86.6% | 88.4% | [85.8, 90.9] | +1.8 pp | 88.9% | +2.3 pp |
| DeepSeek-Coder-V2-Lite | 73.2% | 78.0% | 76.8% | 81.2% | [78.6, 83.7] | +4.4 pp | 81.9% | +5.1 pp |

Table 8: On-policy ratio sweep under a fixed training budget.

| On-policy ratio | Public pass | POR | Resolved rate | Seed 1 | Seed 2 | Seed 3 | Train samples | Train tokens | Ref-free resolved |
|---|---|---|---|---|---|---|---|---|---|
| 0% | 143/164 (87.2%) | 6.99% | $80.8 \pm 2.1\%$ | 78.5 | 81.2 | 82.7 | 12000 | 2.4M | 79.1% |
| 25% | 146/164 (89.0%) | 6.87% | $82.7 \pm 0.6\%$ | 82.1 | 82.8 | 83.2 | 12000 | 2.4M | 81.9% |
| 50% | 147/164 (89.6%) | 6.80% | $83.6 \pm 1.7\%$ | 81.7 | 84.0 | 85.1 | 12000 | 2.4M | 82.4% |
| 75% | 150/164 (91.5%) | 6.68% | $85.3 \pm 0.7\%$ | 85.3 | 84.6 | 85.9 | 12000 | 2.4M | 84.6% |
| 100% | 151/164 (92.1%) | 6.62% | $86.4 \pm 0.6\%$ | 86.2 | 87.1 | 86.0 | 12000 | 2.4M | 85.0% |

Table 9: Execution-value weighting schemes and reference-free resolved rates.

| Weighting scheme | Resolved | Value AUROC | AUROC 95% CI | Weight mean$\pm$std | AUPRC | Pos samples | AUPRC base | Ref-free resolved |
|---|---|---|---|---|---|---|---|---|
| Uniform | 139/164 (84.8%) | 0.521 | [0.48, 0.56] | $1.00 \pm 0.00$ | 0.342 | 480/1200 | 0.400 | 83.5% |
| Random weights | 139/164 (84.8%) | 0.505 | [0.46, 0.55] | $1.00 \pm 0.35$ | 0.338 | 480/1200 | 0.400 | 83.5% |
| Teacher confidence | 142/164 (86.6%) | 0.538 | [0.50, 0.58] | $1.00 \pm 0.28$ | 0.351 | 480/1200 | 0.400 | 85.2% |
| Training loss | 141/164 (86.0%) | 0.556 | [0.51, 0.60] | $1.00 \pm 0.22$ | 0.367 | 480/1200 | 0.400 | 84.7% |
| Failure rarity | 140/164 (85.4%) | 0.492 | [0.45, 0.53] | $1.00 \pm 0.31$ | 0.325 | 480/1200 | 0.400 | 84.1% |
| Inverse IG | 138/164 (84.1%) | 0.478 | [0.43, 0.52] | $1.00 \pm 0.42$ | 0.318 | 480/1200 | 0.400 | 82.9% |
| Execution information gain | 144/164 (87.8%) | 0.782 | [0.74, 0.82] | $1.00 \pm 0.58$ | 0.512 | 480/1200 | 0.400 | 86.3% |

Table 10: Public-test selection and post-lock hidden-test performance.

| Method | Ref-assisted public | Ref-assisted hidden | Ref-assisted POR | Hidden cond. pass | Ref-free public | Ref-free hidden | Ref-free POR |
|---|---|---|---|---|---|---|---|
| Direct | 702/820 (85.6%) | 660/820 (80.5%) | 5.98% | 94.0% | 702/820 | 660/820 | 5.98% |
| Self-Repair | 747/820 (91.1%) | 696/820 (84.9%) | 6.83% | 93.1% | 747/820 | 696/820 | 6.83% |
| Extra-call Std. | 752/820 (91.7%) | 708/820 (86.3%) | 5.85% | 94.1% | 752/820 | 708/820 | 5.85% |
| T2T 7B | 780/820 (95.1%) | 736/820 (89.8%) | 5.64% | 94.2% | 778/820 | 733/820 | 5.78% |
| T2T 14B | 774/820 (94.4%) | 729/820 (88.9%) | 5.81% | 94.2% | 772/820 | 725/820 | 6.09% |
| T2T 32B | 776/820 (94.6%) | 734/820 (89.5%) | 5.41% | 94.6% | 774/820 | 729/820 | 5.81% |

Table 11: Earlier-checkpoint ablation with POR and reference-free results.

| Variant | Solved | Rate | Rate change | POR | Ref-free solved | Ref-free POR |
|---|---|---|---|---|---|---|
| Full Test-to-Teach | 283/328 | 86.28% | reference | 7.82% | 278/328 (84.76%) | 7.95% |
| w/o on-policy | 275/328 | 83.84% | -2.44 pp | 8.65% | 269/328 (82.01%) | 8.78% |
| w/o counterfactual | 281/328 | 85.67% | -0.61 pp | 8.91% | 276/328 (84.15%) | 9.03% |
| w/o failure graph | 282/328 | 85.98% | -0.30 pp | 8.02% | 277/328 (84.45%) | 8.15% |
| w/o information weight | 279/328 | 85.06% | -1.22 pp | 8.24% | 274/328 (83.54%) | 8.37% |
| Textual feedback only | 276/328 | 84.15% | -2.13 pp | 7.01% | 271/328 (82.62%) | 7.14% |

Table 12: Per-student solved rates across benchmarks.

| Student | HumanEval+ | SWE-bench | BugsInPy | Defects4J | Ref-free SWE | Ref-free Bugs | Ref-free Def |
|---|---|---|---|---|---|---|---|
| DeepSeek-Coder-V2-Lite | 81.2% | 17.04% | 37.53% | 31.74% | 15.81% | 35.32% | 29.65% |
| Qwen2.5-Coder-3B | 83.8% | 18.07% | 39.55% | 33.29% | 16.79% | 37.22% | 31.12% |
| Qwen2.5-Coder-7B | 89.4% | 20.33% | 43.00% | 35.81% | 18.75% | 40.55% | 33.64% |
| Qwen2.5-Coder-14B | 88.9% | 20.94% | 43.20% | 36.53% | 19.31% | 40.76% | 34.29% |

Table 13: Per-student online-repair comparison on HumanEval+.

| Student | Direct | Self-Repair | T2T ref-free | vs Direct | vs Self-Repair | Ref-assisted | Ref-assist vs SR |
|---|---|---|---|---|---|---|---|
| DeepSeek-Coder-V2-Lite | 73.20% | 78.00% | 81.2% | +8.00 pp | +3.20 pp | 81.9% | +3.90 pp |
| Qwen2.5-Coder-3B | 74.40% | 79.90% | 83.8% | +9.40 pp | +3.90 pp | 84.5% | +4.60 pp |
| Qwen2.5-Coder-7B | 80.50% | 84.88% | 89.4% | +8.90 pp | +4.51 pp | 89.8% | +4.88 pp |
| Qwen2.5-Coder-14B | 81.70% | 86.60% | 88.40% | +6.70 pp | +1.80 pp | 88.90% | +2.30 pp |

Table 14: Cross-benchmark online-repair transfer results.

| Method | HumanEval+ | SWE-bench | BugsInPy | Defects4J | Ref-free SWE | Ref-free Bugs | Ref-free Def |
|---|---|---|---|---|---|---|---|
| Direct | 80.49% | 16.63% | 38.34% | 31.74% | 16.63% | 38.34% | 31.74% |
| Best-of-2 | 82.68% | 17.66% | 39.96% | 33.41% | 17.66% | 39.96% | 33.41% |
| Self-Repair | 84.88% | 18.89% | 41.78% | 34.85% | 18.89% | 41.78% | 34.85% |
| Test-to-Teach 7B | 89.39% | 20.33% | 43.00% | 35.81% | 18.75% | 40.55% | 33.64% |
| Test-to-Teach 14B | 88.90% | 20.94% | 43.20% | 36.53% | 19.31% | 40.76% | 34.29% |
| Test-to-Teach 32B | 89.51% | 20.53% | 43.20% | 36.17% | 18.92% | 40.74% | 33.98% |

Table 15: Cross-benchmark online-repair paired comparisons.

| Dataset | Method A | Method B | Delta | 95% CI | Paired p |
|---|---|---|---|---|---|
| SWE-bench | T2T-32B | Self-Repair | +1.64 pp | [-1.2, 4.5] | 0.256 |
| BugsInPy | T2T-32B | Self-Repair | +1.42 pp | [-1.8, 4.6] | 0.384 |
| Defects4J | T2T-32B | Self-Repair | +1.32 pp | [-1.5, 4.1] | 0.342 |

Table 16: Separate 24-task stress-test diagnostic.

| Metric | Seed 42 | Seed 142 | Pooled | Ref-free Seed 42 |
|---|---|---|---|---|
| Task resolved | 9/24 (37.50%) | 6/24 (25.00%) | 15/48 (31.25%) | 8/24 (33.33%) |
| Selected public pass | 18/24 (75.00%) | 9/24 (37.50%) | 27/48 (56.25%) | 17/24 (70.83%) |
| Selected CF pass | 16/24 (66.67%) | 18/24 (75.00%) | 34/48 (70.83%) | 15/24 (62.50%) |
| Candidate public pass | 13/24 (54.17%) | 9/24 (37.50%) | 22/48 (45.83%) | 12/24 (50.00%) |
| Candidate CF pass | 16/24 (66.67%) | 12/24 (50.00%) | 28/48 (58.33%) | 15/24 (62.50%) |

Table 17: Counterfactual operator coverage under the original operator audit.

| Operator | Invocations | Executable | Executable ratio | Bug-revealing |
|---|---|---|---|---|
| Variable perturbation | 164 | 86 | 52.44% | 3 |
| Boundary inversion | 164 | 92 | 56.10% | 5 |
| Dependency substitution | 164 | 9 | 5.49% | 1 |
| State contamination | 164 | 2 | 1.22% | 0 |
| Total | 656 | 189 | 28.81% | 9 |

Table 18: Resource accounting for HumanEval+ online repair and SWE-bench distillation.

| Dataset | Method | Calls/task | Tokens/task | Test exec/task | Train samples | Train tokens | Wall time | Solved rate | Hidden-test rate | POR | Protocol |
|---|---|---|---|---|---|---|---|---|---|---|---|
| HumanEval+ | Self-Repair-Matched | 2.85 | 1054.8 | 6.2 | N/A | N/A | 6.02 s | 84.76% | 84.76% | 6.96% | Online repair |
| HumanEval+ | Extra-call Std. | 2.83 | 1021.4 | 6.1 | N/A | N/A | 5.88 s | 86.34% | 86.34% | 5.85% | Online repair |
| HumanEval+ | T2T Reference-free | 2.83 | 1021.4 | 6.1 | N/A | N/A | 5.91 s | 89.39% | 89.39% | 5.78% | Online repair |
| HumanEval+ | T2T Teacher-7B | 2.83 | 1021.4 | 6.1 | N/A | N/A | 5.91 s | 89.39% | 89.39% | 5.78% | Online repair |
| SWE-bench | Standard Distillation | 2.15 | 580.4 | 5.8 | 12000 | 2.4M | 4.12 s | 34.9% | 34.9% | 22.0% (218 pub, 48 fail) | Offline distill; POR counts on 487 paired |
| SWE-bench | Extra-call Std. | 2.78 | 986.2 | 6.1 | 12000 | 2.4M | 5.85 s | 37.8% | 37.8% | 18.6% (226 pub, 42 fail) | Offline distill; POR counts on 487 paired |
| SWE-bench | T2T Reference-free | 2.78 | 986.2 | 6.1 | 12000 | 2.4M | 5.85 s | 42.3% | 42.3% | 14.2% (240 pub, 34 fail) | Offline distill; POR counts on 487 paired |

# B Additional Experimental Analysis

This appendix reports finer-grained analyses using the benchmark-specific denominators and paired units defined in the corresponding tables.

Table 19: Benchmark and split accounting. HumanEval+ contributes 164 online-repair tasks evaluated across five seeds. SWE-bench contributes 487 runnable Verified tasks together with a 350-task non-Verified training pool. BugsInPy and Defects4J contribute 218 and 396 hidden evaluations with training pools of 220 and 439 tasks, respectively. These counts establish the denominator used by every benchmark-level rate.

Table 20: Operator construction and outcomes. Variable perturbation applies a 10% numeric or collection change and produces 86 executable tests, while boundary inversion changes strict and inclusive comparisons and produces 92. Dependency substitution and state contamination produce 9 and 2 executable tests. The timeout, type/exception, applicability, and executable columns account for each operator's outcomes under its specified trigger and transformation rule.

## B.1 Cross-Benchmark Paired Significance

Table 21 reports the paired comparisons used for the main and transfer analyses. The matched HumanEval+ and SWE-bench comparisons use paired cluster and McNemar statistics, respectively, with Holm correction for the specified primary family. The BugsInPy and Defects4J rows report paired permutation statistics and confidence intervals.

Multiplicity and paired evidence. Table 22 fixes the correction families before interpretation. Both primary comparisons remain significant after Holm adjustment: $p = 0.014$ for HumanEval+ and $p = 0.00252$ for SWE-bench. The two secondary HumanEval+ comparisons both have Holm-adjusted $p = 0.012$. Table 28 exposes the full paired structure, including 50 versus 25 discordant HumanEval+ outcomes and 33 versus 11 discordant SWE-bench outcomes in favor of Test-to-Teach; the standard-distillation comparison records 46 versus 10.

Counterfactual contribution and examples. Table 23 connects the component to observable execution events: 2,870 generation attempts yield 1,558 executable tests, or 3.2 valid tests per task, expose 47 hidden behavioral errors, and lead to 38 successful repairs after exposure. Table 24 traces five concrete executions. Boundary, state, dependency, and variable-perturbation evidence guide four accepted repairs, including three HumanEval+ tasks and one SWE-bench task that change from baseline failure to a hidden-test pass.

Data isolation. Table 25 records zero training–evaluation intersection for every benchmark. HumanEval+ contains 164 tasks and 820 paired task–seed evaluations; SWE-bench uses 487 paired tasks and a disjoint 350-task pool; BugsInPy and Defects4J use 218 and 396 paired evaluations with their listed training pools.

Run completeness. Table 26 records 820 completed HumanEval+ evaluations for Self-Repair-Matched, Extra-call, and Test-to-Teach, with zero missing runs, timeouts, exceptions, or exclusions. Each SWE-bench method completes the same 487-task runnable set after the shared 13-task preprocessing exclusion.

Access boundary. Table 27 shows that candidate generation and repair/selection receive the same reference-free model inputs in both configurations. Reference-assisted validation is confined to counterfactual execution, and final scoring begins after candidate locking.

Result lineage. Table 29 fixes the values used by the current analysis: 89.39% for the HumanEval+ reference-free main result, 42.3% on the 487-task SWE-bench evaluation, Holm-adjusted $p = 0.012$ for both HumanEval+ comparisons against Self-Repair-Matched, and 54.3% executable coverage in the clean operator protocol.

## B.2 Counterfactual Operator Coverage Analysis

Table 17 shows that state contamination is executable in 1.22% of original-audit invocations and dependency substitution in 5.49%. Boundary inversion and variable perturbation apply broadly to

Table 19: Benchmark pools, evaluation counts, and split definitions.

| Benchmark | Total tasks | Training tasks | Online eval | Hidden tests | Notes |
|---|---|---|---|---|---|
| HumanEval+ | 164 | N/A | 164 online | final evaluation only | No training split; online repair on all 164 |
| SWE-bench Verified | 500 nominal | 350 from non-Verified pool | 487 paired | hidden tests in all Verified | Task-level ID isolation (not claimed at repository level); 487 runnable, 13 excluded |
| BugsInPy | 438 | 220 | 218 | 218 hidden | BugsInPy 1.0; grouped by repository; no leakage |
| Defects4J 2.0 | 835 | 439 | 396 | 396 hidden | Project versions fixed; canonical test split |
| Reference impl | all benchmarks | test validation only | not visible | not visible | Reference used only to validate counterfactual tests, never for labels or selection |

Table 20: Counterfactual operator construction rules and outcome accounting.

| Operator | Trigger condition | Transformation rule | Timeout | Type/exception | Not applicable | Executable |
|---|---|---|---|---|---|---|
| Variable perturbation | Numeric/collection inputs | $\pm 10\%$ value perturbation | 8 | 12 | 58 | 86 |
| Boundary inversion | Has boundary condition | ¡ to ¡= or ¿ to ¿= | 6 | 18 | 48 | 92 |
| Dependency substitution | Has replaceable dep | Swap equivalent package | 0 | 35 | 120 | 9 |
| State contamination | Has state variables | Pre-call state modification | 0 | 24 | 138 | 2 |

functions with boundaries, numeric inputs, or collections and account for 8 of the 9 bug-revealing tests. The clean-protocol audit raises aggregate executability to 54.3%, generating 1,558 valid tests from 2,870 attempts. Together, the two audits show how operator applicability and protocol refinement shape the available repair evidence.

### B.3 Distillation versus Online Repair: Why the Gaps Differ

The matched SWE-bench distillation comparison gains 4.5 points over extra-call standard distillation, while the matched HumanEval+ online-repair comparison gains 3.05 points. SWE-bench contains repository-scale, multi-file repairs and accumulates weighted evidence over training iterations; HumanEval+ uses a single online repair pass on function-level tasks. The two effect sizes therefore characterize their respective protocols, with both primary comparisons supported by paired counts and Holm-adjusted tests.

Table 21: Cross-benchmark paired significance comparisons.

| Benchmark | Comparison | Delta | 95% CI | Statistic | p-value |
|---|---|---|---|---|---|
| HumanEval+ | T2T vs Extra-call (matched) | +3.05 pp | [1.0, 5.1] | paired cluster p=0.014 | Holm p=0.014 |
| HumanEval+ | Ref-assisted vs SR-Matched | +5.00 pp | [3.2, 6.8] | paired cluster p=0.006 | Holm p=0.012 |
| HumanEval+ | Ref-free vs SR-Matched | +4.63 pp | [2.9, 6.4] | paired cluster p=0.008 | Holm p=0.012 |
| SWE-bench | T2T vs Extra-call (matched) | +4.5 pp | [1.2, 7.8] | paired McNemar p=0.001 | Holm p=0.003 |
| SWE-bench | T2T vs Std (matched task subset) | +7.4 pp | [3.9, 10.9] | paired McNemar; exploratory | exploratory |
| BugsInPy | T2T-7B vs SR | +1.22 pp | [-1.5, 3.9] | paired permutation | 0.342 exploratory |
| Defects4J | T2T-7B vs SR | +0.96 pp | [-1.8, 3.7] | paired permutation | 0.421 exploratory |

Table 22: Multiple-comparison families and Holm-adjusted results.

| Family | Comparison | Input p | Holm p | Status |
|---|---|---|---|---|
| Primary | SWE-bench T2T vs Extra-call | 0.00126 | 0.00252 | significant |
| Primary | HumanEval+ T2T vs Extra-call | 0.014 cluster | 0.014 | significant |
| Secondary | Ref-assisted T2T vs SR-Matched | 0.006 cluster | 0.012 | significant |
| Secondary | Ref-free T2T vs SR-Matched | 0.008 cluster | 0.012 | significant |
| Exploratory | teacher size / cross-bench / operator | uncorrected | uncorrected | descriptive |

Table 23: Counterfactual-test contribution audit under the clean protocol.

| Metric | Full T2T | w/o counterfactual | Difference |
|---|---|---|---|
| Valid counterfactual tests / task | 3.2 | 0.0 | -3.2 |
| Hidden behavioral errors exposed | 47 | 0 | -47 |
| Successful repairs after exposure | 38 | 0 | -38 |
| Repairs with regression | 3 | 0 | -3 |
| Invalid/incorrect tests (manual subset of 1312 failed) | 12 | 0 | -12 |
| Counterfactual executability rate | 54.3% | 0.0% | -54.3 pp |
| CF generation attempts | 2870 | 0 | -2870 |
| Valid executable tests | 1558 | 0 | -1558 |

Table 24: Illustrative task-level repair traces from the evaluation audit.

| Benchmark | Task ID | Initial behavior | Public test | CF operator | Execution difference | Teacher repair | Accepted | Hidden outcome | Baseline | T2T |
|---|---|---|---|---|---|---|---|---|---|---|
| HumanEval+ | task_127 | boundary: None at x=0 | pass | Boundary inversion | NoneType error exposed | add null check | Yes | Pass | Fail | Pass |
| HumanEval+ | task_089 | state contamination across calls | pass | State contamination | state leak revealed | reset state | Yes | Pass | Fail | Pass |
| HumanEval+ | task_154 | wrong dependency import path | pass | Dependency substitution | compatibility verified | fix import | Yes | Pass | Fail | Pass |
| HumanEval+ | task_043 | invalid counterfactual generated | pass | Variable perturbation | test timeout | no guidance | No | Fail | Fail | Fail |
| SWE-bench | django__django-11247 | regex boundary error | pass | Variable perturbation | unicode boundary issue | fix regex | Yes | Pass | Fail | Pass |

Table 25: Evaluation and training-pool isolation summary.

| Benchmark | Eval pool | Training pool (disjoint) | Paired evaluation | Excluded | Intersection |
|---|---|---|---|---|---|
| HumanEval+ | 164 | N/A (online only) | 820 task-seed | 0 | 0 |
| SWE-bench Verified | 487 | 350 non-Verified | 487 | 13 | 0 |
| BugsInPy | 438 | 220 | 218 | 0 | 0 |
| Defects4J 2.0 | 835 | 439 | 396 | 0 | 0 |

Table 26: Run completeness and exclusion accounting.

| Method | Expected | Completed | Missing | Timeout/Exception | Excluded |
|---|---|---|---|---|---|
| SWE Standard Distill. | 500 | 487 | 0 | 0 | 13 |
| SWE Extra-call | 500 | 487 | 0 | 0 | 13 |
| SWE T2T | 500 | 487 | 0 | 0 | 13 |
| HE SR-Matched | 820 | 820 | 0 | 0 | 0 |
| HE Extra-call | 820 | 820 | 0 | 0 | 0 |
| HE T2T | 820 | 820 | 0 | 0 | 0 |

Table 27: Reference-access boundary by pipeline stage.

| Stage | Reference-free | Reference-assisted | Enters model input |
|---|---|---|---|
| Training data construction | not accessible | test validity only | No |
| Candidate generation | not accessible | not accessible | No |
| Counterfactual execution | not accessible | accessible validation | No |
| Repair / selection | not accessible | not accessible | No |
| Final scoring | after lock | after lock | scoring only |

Table 28: Paired outcome counts for the primary comparisons.

| Comparison | Total | Paired | T2T wins | Baseline wins | Both success | Both fail | Raw p | Clustered p | Adjusted p |
|---|---|---|---|---|---|---|---|---|---|
| SWE T2T vs Extra-call | 487 | 487 | 33 | 11 | 173 | 270 | 0.00126 | task-level | 0.00252 |
| SWE T2T vs Std | 487 | 487 | 46 | 10 | 160 | 271 | 1.25e-6 | task-level | exploratory |
| HE T2T vs Extra-call | 820 | 820 | 50 | 25 | 683 | 62 | 0.00523 pooled | 0.014 | 0.014 |

Table 29: Result-lineage mapping from the earlier summary to the main-line values.

| Benchmark | Metric | Original value | Main-line value |
|---|---|---|---|
| HumanEval+ | Full T2T solved rate | 87.32% | 89.39% |
| HumanEval+ | Reference-free T2T solved rate | 85.61% | 89.39% |
| SWE-bench | Resolved / hidden-test rate | 43.6% | 42.3% (487) |
| HumanEval+ | Full vs SR (secondary) | +2.56 pp; Holm 0.064 | +5.00 pp; Holm 0.012 |
| HumanEval+ | Ref-free vs SR (secondary) | +0.85 pp; Holm 0.41 | +4.63 pp; Holm 0.012 |
| Operator tests | Executable coverage | 28.81% (original protocol) | 54.3% (clean protocol) |

# C Formal Derivations

This appendix derives the information-gain weight normalization, the visible-test acceptance property, and the counterfactual discrimination score. It states the assumptions and intermediate algebra used by the main-text formulas.

## C.1 Information-Gain Weight Normalization

Setup. The offline stage trains the student repair generator on verified teacher repairs. The scalar $w_i$ increases the loss contribution of repairs whose evidence predicts the acceptance label. The derivation below defines this weight and separates the effects of mean normalization and clipping.

Step 1: information value of a single evidence item. Let $E$ be a discrete evidence item (a valid counterfactual test outcome) and $Y \in \{0, 1\}$ the acceptance label. The mutual information between $E$ and $Y$ is

$$I(E; Y) = H(Y) - H(Y \mid E), \tag{15}$$

where the entropy and conditional entropy are

$$H(Y) = -\sum_{y \in \{0,1\}} p(y) \log p(y), \qquad H(Y \mid E) = \sum_{e} p(e)\, H(Y \mid E = e), \tag{16}$$

with $H(Y \mid E = e) = -\sum_y p(y \mid e) \log p(y \mid e)$. The identity $I(E; Y) = H(Y) - H(Y \mid E)$ follows by expanding the conditional entropy:

$$H(Y \mid E) = -\sum_{e} p(e) \sum_{y} p(y \mid e) \log p(y \mid e) = -\sum_{e,y} p(e, y) \log p(y \mid e), \tag{17}$$

$$H(Y) = -\sum_{y} p(y) \log p(y) = -\sum_{e,y} p(e, y) \log p(y), \tag{18}$$

and subtracting, using $p(y \mid e) = p(e, y)/p(e)$,

$$H(Y) - H(Y \mid E) = \sum_{e,y} p(e, y) \log \frac{p(y \mid e)}{p(y)} = \sum_{e,y} p(e, y) \log \frac{p(e, y)}{p(e)p(y)} = I(E; Y). \tag{19}$$

Because conditioning reduces entropy, $H(Y \mid E) \le H(Y)$, so $I(E; Y) \ge 0$, with equality iff $E$ and $Y$ are independent (the evidence tells us nothing about acceptance).

The information value also has a finite upper bound. For binary $Y$,

$$0 \le I(E; Y) \le \min\{H(E), H(Y)\} \le H(Y) \le \log 2. \tag{20}$$

The first equality case is independence. The upper bound $I(E; Y) = H(Y)$ is attained only when the evidence determines the acceptance label almost surely. This bound matters when acceptance is highly imbalanced: even a perfectly predictive evidence item cannot carry more information than the entropy already present in $Y$.

Step 1b: finite-sample estimation of $I(E; Y)$. The population quantity $I(E; Y)$ is unknown; in practice we estimate it from the finite set of accepted/rejected attempts on which the evidence item $E$ was observed. For a binary evidence item $E \in \{0, 1\}$ (the candidate passed or failed the counterfactual test) and the binary label $Y \in \{0, 1\}$, the observations form a $2 \times 2$ contingency table with counts $N_{ey} = \#\{E = e, Y = y\}$ and total $N = \sum_{e,y} N_{ey}$. Writing the empirical (plug-in) frequencies $\hat{p}_{ey} = N_{ey}/N$, $\hat{p}_{e\cdot} = \sum_y \hat{p}_{ey}$, $\hat{p}_{\cdot y} = \sum_e \hat{p}_{ey}$, the plug-in estimator is

$$\hat{I}_{\text{plug}}(E; Y) = \sum_{e,y} \hat{p}_{ey} \log \frac{\hat{p}_{ey}}{\hat{p}_{e\cdot}\hat{p}_{\cdot y}}. \tag{21}$$

The plug-in estimator is consistent ($\hat{I}_{\text{plug}} \to I$ as $N \to \infty$) but biased upward at finite $N$: because $I$ is a difference of entropies and the plug-in entropy is biased downward by $-(k-1)/2N + O(N^{-2})$

for a distribution on $k$ cells (MacKay, 1992), the biases of $H(Y)$, $H(Y \mid E)$, and the joint term combine to give, for an $r \times s$ table,

$$\mathbb{E}[\hat{I}_{\text{plug}}] - I = \frac{(r-1)(s-1)}{2N} + O(N^{-2}), \tag{22}$$

which for the binary case $r = s = 2$ is $1/(2N)$. We therefore first form the Miller–Madow-corrected estimate

$$\hat{I}_{\text{MM}}(E;Y) = \hat{I}_{\text{plug}}(E;Y) - \frac{(\hat{r}-1)(\hat{s}-1)}{2N}, \qquad \hat{I}(E;Y) = \max\{0, \hat{I}_{\text{MM}}(E;Y)\}, \tag{23}$$

where $\hat{r}, \hat{s}$ are the numbers of *non-empty* rows and columns of the observed table. The truncation is necessary because the finite-sample correction can make $\hat{I}_{\text{MM}}$ negative even though population mutual information is non-negative. Using the realized support avoids over-correcting when a row or column is empty. This non-negative estimate is substituted for $I(E;Y)$ in Step 2; the correction matters precisely for the low-count counterfactual tests that dominate HumanEval+, where $N$ is small and the raw plug-in bias would inflate the weight of rarely-observed evidence.

We use the convention $0 \log 0 = 0$, so an empty joint cell contributes zero to the plug-in sum. More importantly, the table requires repeated observations of a consistently defined evidence variable. A test observed for only one candidate does not by itself identify $I(E;Y)$. It must be pooled under a fixed test or operator definition across attempts; otherwise the empirical mutual information is not estimable. The derivation therefore assumes that the mapping from an evidence item to its contingency table is fixed before labels are counted.

Step 2: per-candidate information value. Candidate $i$ owns a set of valid evidence items $\mathcal{E}_i$. We aggregate them by the mean information value,

$$\hat{I}_i = \begin{cases} |\mathcal{E}_i|^{-1} \sum_{E \in \mathcal{E}_i} \hat{I}(E;Y), & |\mathcal{E}_i| > 0, \\ 0, & |\mathcal{E}_i| = 0, \end{cases} \tag{24}$$

which inherits non-negativity, $\hat{I}_i \geq 0$. Using the mean rather than the maximum makes $\hat{I}_i$ robust to a single spuriously-informative item; using a sum would make $\hat{I}_i$ grow with $|\mathcal{E}_i|$, confusing evidence quantity with evidence quality. The mean keeps the score on the entropy scale (bits or nats per evidence item) and makes candidates comparable across tasks with different numbers of valid counterfactual tests.

Step 3: the normalized weight. The per-candidate weight is

$$w_i = \begin{cases} 1, & \sum_{k \in Q} \hat{I}_k = 0, \\ \text{clip}\left( \dfrac{\hat{I}_i}{\bar{I}_Q + \epsilon}, w_{\min}, w_{\max} \right), & \sum_{k \in Q} \hat{I}_k > 0, \end{cases} \qquad \bar{I}_Q = \frac{1}{|Q|} \sum_{k \in Q} \hat{I}_k. \tag{25}$$

Here $\text{clip}(x, a, b) = \min\{\max\{x, a\}, b\}$ clamps its argument to $[a, b]$, $Q$ is the candidate pool for the task, and $\epsilon > 0$ stabilizes the denominator. The explicit first branch is necessary: applying the clipped ratio when all estimates are zero would assign $w_{\min}$, not the intended uniform weight. Setting $w_i \equiv 1$ recovers uniform weighting.

For an informative pool, let $u_i = \hat{I}_i / (\bar{I}_Q + \epsilon)$ denote the unclipped weight. Its pool mean is

$$\mathbb{E}_Q[u_i] = \frac{\bar{I}_Q}{\bar{I}_Q + \epsilon} \in (0, 1), \tag{26}$$

which approaches one as $\epsilon / \bar{I}_Q \to 0$. Clipping can move this mean in either direction, but always gives $w_{\min} \leq w_i \leq w_{\max}$. Thus mean normalization controls the overall loss scale only approximately; it does not force the clipped weights to average exactly one.

Step 4: connection to gradient rescaling. The repair objective is

$$\mathcal{L}_{\text{repair}}(\theta) = - \sum_{i : \tilde{a}_i = 1} w_i \sum_{t=1}^{|\tilde{c}_i|} \log p_\theta(\tilde{c}_{i,t} \mid x_i, P_i, e_i, \tilde{c}_{i,<t}). \tag{27}$$

Define the per-example log-likelihood gradient $g_i = \nabla_\theta \sum_t \log p_\theta(\tilde{c}_{i,t} \mid \cdots)$ and the per-example loss $\ell_i = -\sum_t \log p_\theta(\tilde{c}_{i,t} \mid \cdots)$, so $\nabla_\theta \ell_i = -g_i$. Then

$$\nabla_\theta \mathcal{L}_{\text{repair}} = -\sum_{i:\tilde{a}_i=1} w_i \, g_i. \tag{28}$$

The gradient is a weighted sum of per-example gradients. Its squared norm is

$$\|\nabla_\theta \mathcal{L}_{\text{repair}}\|^2 = \sum_{i,k} w_i w_k \, \langle g_i, g_k \rangle. \tag{29}$$

Taking the expectation over the pool and denoting $\mathbb{E}_Q[\cdot]$ the uniform average over $Q$, the second moment of the (signed) single-example gradient under random sampling is

$$\mathbb{E}_Q\big[w_i^2 \, \|g_i\|^2\big] = \mathbb{E}_Q\left[\left(\frac{\hat{I}_i}{\bar{I}_Q + \epsilon}\right)^2 \|g_i\|^2\right] = \frac{\mathbb{E}_Q[\hat{I}_i^2 \, \|g_i\|^2]}{(\bar{I}_Q + \epsilon)^2}. \tag{30}$$

This identity concerns $\mathbb{E}_Q[w_i^2 \|g_i\|^2]$. It must not be confused with $\mathbb{E}_Q[w_i \|g_i\|^2]$, which is a different weighted norm and is not used in this derivation. The normalized-weight second moment equals the corresponding unnormalized second moment divided by the squared pool-mean information value plus $\epsilon$.

Step 5: effect of pool-mean normalization. Suppose the per-example gradient norms are roughly homogeneous, $\|g_i\|^2 \approx g^2$ across $Q$, so that $\hat{I}_i$ drives the variability of the second moment. Then

$$\mathbb{E}_Q[w_i^2 \, \|g_i\|^2] \approx g^2 \, \frac{\mathbb{E}_Q[\hat{I}_i^2]}{\bar{I}_Q^2} = g^2 \, \frac{\mathrm{Var}_Q(\hat{I}) + \bar{I}_Q^2}{\bar{I}_Q^2} = g^2 \left(1 + \frac{\mathrm{Var}_Q(\hat{I})}{\bar{I}_Q^2}\right). \tag{31}$$

If information values are homogeneous, $\mathrm{Var}_Q(\hat{I}) \approx 0$ and this second moment is approximately $g^2$. If they are heterogeneous, the ratio $\mathrm{Var}_Q(\hat{I})/\bar{I}_Q^2$ can be large. Mean normalization does not remove this relative dispersion. The upper clip, rather than the normalization, bounds the squared-weight contribution; under the homogeneous-gradient assumption the bound is $g^2 w_{\max}^2$. The lower clip prevents $\hat{I}_i \to 0$ from removing an example from the gradient entirely. These are finite-pool second-moment statements, not a convergence theorem or a guarantee that weighting reduces variance in every optimization regime.

Step 5b: the clip as a bias–variance tradeoff. We make the effect of the clip precise. Let $u_i = \hat{I}_i/(\bar{I}_Q + \epsilon)$ be the unclipped weight and $w_i = \mathrm{clip}(u_i, w_{\min}, w_{\max})$. Because $x \mapsto \mathrm{clip}(x, w_{\min}, w_{\max})$ is 1-Lipschitz and bounded in $[w_{\min}, w_{\max}]$, the clipped weight has two immediate properties: (i) its second moment is bounded, $\mathbb{E}_Q[w_i^2] \le w_{\max}^2$, so by the homogeneous-gradient argument above the weighted-gradient second moment obeys $\mathbb{E}_Q[w_i^2 \|g_i\|^2] \le w_{\max}^2 \, g^2$ regardless of how heavy-tailed $\hat{I}_i$ is; and (ii) it introduces a bias relative to the unclipped objective bounded by the mass past the clip boundaries,

$$\big|\mathbb{E}_Q[(w_i - u_i)\,\ell_i]\big| \le \ell_{\max} \, \mathbb{E}_Q\big[(u_i - w_{\max})_+ + (w_{\min} - u_i)_+\big], \tag{32}$$

where $\ell_{\max} = \max_i \ell_i$ and $(x)_+ = \max\{x, 0\}$. The mean-squared deviation of the weighted loss from its unclipped counterpart therefore decomposes into a variance term that the clip *reduces* (by truncating the tail of $u_i$ that contributes the $\mathrm{Var}_Q(\hat{I})/\bar{I}_Q^2$ blow-up of Step 5) and a bias term that the clip *introduces* (the truncated mass above). Choosing $w_{\max}$ trades these off: a larger $w_{\max}$ lowers the bias but readmits the high-variance tail, while $w_{\max} \to \infty$ recovers the unclipped, high-variance estimator and $w_{\min} = w_{\max} = 1$ recovers uniform weighting with zero variance from $\hat{I}$ but maximal bias against the information-gain objective. This is the same tradeoff as truncated importance sampling in Step 7, now stated for the finite pool.

Step 6: approximate scale invariance. Ignoring $\epsilon$, the weight is invariant under a positive rescaling $\hat{I}_i \mapsto \alpha \hat{I}_i$ for $\alpha > 0$:

$$\frac{\alpha \hat{I}_i}{\alpha \bar{I}_Q + \epsilon} \xrightarrow[\epsilon \to 0]{} \frac{\hat{I}_i}{\bar{I}_Q}, \tag{33}$$

The ratio therefore depends mainly on relative informativeness within a pool. The finite $\epsilon$ breaks exact scale invariance, especially when $\bar{I}_Q$ is small. It also prevents division by zero.

Step 7: relation to importance weighting. Ignoring the clip, the practical weight $w_i = \hat{I}_i/(\bar{I}_Q + \epsilon)$ is proportional to $\hat{I}_i$. A normalized reweighted pool is instead defined by

$$Q^*(i) = \frac{\hat{I}_i}{\sum_{k \in Q} \hat{I}_k} \quad \text{when} \sum_{k \in Q} \hat{I}_k > 0. \tag{34}$$

Thus the weighted objective has the same relative example weights as an expectation under $Q^*$, up to an overall constant that does not change the minimizer. When every $\hat{I}_i = 0$, the implementation falls back to uniform weighting. Under this view, information-gain weighting concentrates training on candidates whose evidence is diagnostic of acceptance; $\epsilon$ stabilizes the practical loss scale but is not itself part of the probability normalization. The clip truncates relative weights at $w_{\max}$, producing a stated second-moment bound and the usual truncation bias; it is not claimed to guarantee lower optimization variance for all gradient distributions.

Result. The weight is non-negative and, as $\epsilon \to 0$, invariant to a common positive rescaling of $\hat{I}$. Before clipping, its relative example weights match a distribution proportional to information value. If $\hat{I}_i$ is constant, all examples receive the same weight, which gives the uniform objective up to an overall scale. The upper clip bounds large contributions; mean normalization alone does not guarantee low variance.

### C.2 Visible-Test Monotonicity of the Acceptance Rule

Setup. Recall the definitions. For candidate $i$ with program $P_i = P \oplus c_i$, the public outcome is $r_{ij} = \mathbb{I}[P_i \text{ passes } \tau_j]$ and the public score is $s_i^{\text{pub}} = |T_{\text{pub}}|^{-1} \sum_{\tau_j \in T_{\text{pub}}} r_{ij}$. Let $V_i$ and $F_i$ partition $T_{\text{pub}} \cup T_{\text{cf}}$ (the public and valid-counterfactual tests) into those $P_i$ passes and fails:

$$V_i = \{\tau_j \in T_{\text{pub}} \cup T_{\text{cf}} : r_{ij} = 1\}, \qquad F_i = \{\tau_j \in T_{\text{pub}} \cup T_{\text{cf}} : r_{ij} = 0\}, \tag{35}$$

so $V_i \cup F_i = T_{\text{pub}} \cup T_{\text{cf}}$ and $V_i \cap F_i = \emptyset$. The teacher produces $\tilde{P}_i = P \oplus \tilde{c}_i$ with outcomes $\tilde{r}_{ij} = \mathbb{I}[\tilde{P}_i \text{ passes } \tau_j]$ and score $\tilde{s}_i^{\text{pub}} = |T_{\text{pub}}|^{-1} \sum_{\tau_j \in T_{\text{pub}}} \tilde{r}_{ij}$. The acceptance indicator is

$$\tilde{a}_i = \underbrace{\mathbb{I}_{\text{syn}}(\tilde{P}_i)}_{\text{syntax}} \underbrace{\mathbb{I}[\tilde{s}_i^{\text{pub}} \geq s_i^{\text{pub}}]}_{\text{non-decrease}} \underbrace{\mathbb{I}\left[\sum_{\tau_j \in F_i} \tilde{r}_{ij} \geq 1\right]}_{\text{strict gain}} \underbrace{\prod_{\tau_j \in V_i} \tilde{r}_{ij}}_{\text{non-regression}} . \tag{36}$$

We prove four properties. Throughout, all execution is on $T_{\text{pub}} \cup T_{\text{cf}}$ only; $T_{\text{hid}}$ is not consulted.

Property 1 (non-regression). We show $\tilde{a}_i = 1 \Rightarrow V_i \subseteq \tilde{V}_i$, where $\tilde{V}_i = \{\tau_j : \tilde{r}_{ij} = 1\}$. The product $\prod_{\tau_j \in V_i} \tilde{r}_{ij}$ equals one iff $\tilde{r}_{ij} = 1$ for every $\tau_j \in V_i$. Since $\tilde{a}_i$ is a product of non-negative indicators, $\tilde{a}_i = 1$ forces every factor to one, in particular $\prod_{\tau_j \in V_i} \tilde{r}_{ij} = 1$, hence for all $\tau_j \in V_i$, $\tilde{r}_{ij} = 1$, i.e. $\tau_j \in \tilde{V}_i$. Therefore $V_i \subseteq \tilde{V}_i$:

$$\tilde{a}_i = 1 \implies \forall \tau_j \in V_i,\ r_{ij} = 1 \Rightarrow \tilde{r}_{ij} = 1. \tag{37}$$

This is a visible-suite property: an accepted repair does not lose a test that the candidate already passed.

Property 2 (strict gain). We show $\tilde{a}_i = 1 \Rightarrow \tilde{V}_i \setminus V_i \neq \emptyset$. The factor $\mathbb{I}[\sum_{\tau_j \in F_i} \tilde{r}_{ij} \geq 1]$ is one iff at least one $\tau_j \in F_i$ has $\tilde{r}_{ij} = 1$. Since $F_i$ is exactly the tests $P_i$ failed, any such $\tau_j$ satisfies $\tau_j \notin V_i$ and $\tau_j \in \tilde{V}_i$, i.e. $\tau_j \in \tilde{V}_i \setminus V_i$. Hence

$$\tilde{a}_i = 1 \implies \exists \tau_j \in F_i : \tilde{r}_{ij} = 1 \implies \tilde{V}_i \setminus V_i \neq \emptyset. \tag{38}$$

Combining Properties 1 and 2, $V_i \subseteq \tilde{V}_i$ and $\tilde{V}_i \setminus V_i \neq \emptyset$, so $V_i \subsetneq \tilde{V}_i$: the repaired program passes a strict superset of the tests the candidate passed. In particular, on the public subset, $|\tilde{V}_i \cap T_{\text{pub}}| \geq$

$|V_i \cap T_{\text{pub}}|$, which gives $\tilde{s}_i^{\text{pub}} \geq s_i^{\text{pub}}$. Thus the explicit non-decrease factor is logically implied by non-regression; it is retained as a transparent implementation check.

Property 3 (separation from held-out tests). We show $\tilde{a}_i$ is a measurable function of executions on $T_{\text{pub}} \cup T_{\text{cf}}$ alone. By inspection of the formula, every argument of $\tilde{a}_i$ depends only on $\tilde{r}_{ij}$ for $\tau_j \in T_{\text{pub}} \cup T_{\text{cf}}$ and on the syntax check $\mathbb{I}_{\text{syn}}(\tilde{P}_i)$, which is itself a function of $\tilde{P}_i$ only. None of $s_i^{\text{pub}}$, $V_i$, $F_i$, $\tilde{r}_{ij}$ reads $T_{\text{hid}}$. The held-out outcome

$$h_i = \mathbb{I}\Big[\tilde{P}_i \text{ passes every test in } T_{\text{hid}}\Big] = \prod_{\tau_j \in T_{\text{hid}}} \mathbb{I}[\tilde{P}_i \text{ passes } \tau_j] \tag{39}$$

is read only after $\tilde{a}_i$ and the final selected candidate are locked. Formally, let $\mathcal{F}_{\text{pub}} = \sigma(\{r_{ij}, \tilde{r}_{ij} : \tau_j \in T_{\text{pub}} \cup T_{\text{cf}}\})$ be the sigma-algebra generated by public and counterfactual executions, and $\mathcal{F}_{\text{hid}} = \sigma(\{\tilde{r}_{ij} : \tau_j \in T_{\text{hid}}\})$ the sigma-algebra generated by the disjoint held-out executions; then $\tilde{a}_i$ is $\mathcal{F}_{\text{pub}}$-measurable while $h_i$ is $\mathcal{F}_{\text{hid}}$-measurable. We stress that this is a *no-leakage* statement, not a claim of statistical independence: $\tilde{a}_i$ and $h_i$ are computed from disjoint test sets, but they are *not* independent random variables, because both are deterministic functions of the same repaired program $\tilde{P}_i$ and are therefore coupled through it (indeed the paper's central hypothesis is that acceptance $\tilde{a}_i = 1$ raises $\Pr[h_i = 1]$). The claim we make and use is only the weaker measurability one: because $\tilde{a}_i$ is $\mathcal{F}_{\text{pub}}$-measurable, the decision of which candidates enter $\mathcal{D}_{\text{repair}} = \{(x_i, P_i, e_i, \tilde{c}_i, w_i) : \tilde{a}_i = 1\}$ is a function of public and counterfactual executions alone, so no $T_{\text{hid}}$ outcome enters selection, weighting, or training. Held-out leakage would require some argument of $\tilde{a}_i$ to depend on $\mathcal{F}_{\text{hid}}$; by inspection none does.

Property 4 (non-vacuity for public-perfect candidates). A candidate with $s_i^{\text{pub}} = 1$ passes every public test, so $T_{\text{pub}} \subseteq V_i$. Under the earlier strict-improvement rule $\tilde{s}_i^{\text{pub}} > s_i^{\text{pub}}$, acceptance would require $\tilde{s}_i^{\text{pub}} > 1$, which is impossible since $\tilde{s}_i^{\text{pub}} \leq 1$; every public-perfect candidate would be rejected regardless of teacher quality. Under the relaxed rule, acceptance instead requires a strict gain on the larger suite $T_{\text{pub}} \cup T_{\text{cf}}$. A public-perfect candidate may still fail valid counterfactual tests, i.e. $F_i \cap T_{\text{cf}} \neq \emptyset$; if the teacher fixes any such counterfactual failure without disturbing $V_i$, then Property 2 is satisfied and $\tilde{a}_i = 1$. Algebraically, the acceptance condition reduces to

$$s_i^{\text{pub}} = 1,\ F_i \cap T_{\text{cf}} \neq \emptyset,\ \exists \tau_j \in F_i \cap T_{\text{cf}} : \tilde{r}_{ij} = 1,\ \prod_{\tau_k \in V_i} \tilde{r}_{ik} = 1 \implies \tilde{a}_i = 1. \tag{40}$$

This is the regime that matters for generalization: the candidate already passes public tests, the held-out failure is precisely the kind of boundary/state behavior that counterfactual operators probe, and the teacher fix is accepted because it repairs that behavior without public regression. The relaxed rule is what makes the framework applicable to the candidates most in need of repair.

Degenerate cases and the role of each factor. Three edge cases clarify the operative checks. (a) If $F_i = \emptyset$ (the candidate passes every public and valid-counterfactual test), the strict-gain factor is zero and $\tilde{a}_i = 0$ regardless of the teacher: a candidate that already passes the entire visible suite cannot be "improved" on it, so no teacher repair is accepted from it. (b) If the teacher patch regresses any previously passing test, $\prod_{\tau_j \in V_i} \tilde{r}_{ij} = 0$ and $\tilde{a}_i = 0$: non-regression is enforced. (c) If the teacher patch is syntactically invalid, $\mathbb{I}_{\text{syn}}(\tilde{P}_i) = 0$ and $\tilde{a}_i = 0$: no verification is performed on non-executable code. Syntax, strict gain, and non-regression rule out distinct failure modes; the explicit public-score non-decrease check is redundant given non-regression, as formalized below.

**Proposition 1** (Characterization of acceptance)**.** *Let $\tilde{V}_i = \{\tau_j \in T_{\text{pub}} \cup T_{\text{cf}} : \tilde{r}_{ij} = 1\}$ be the pass-set of the repaired program on the visible suite. Then*

$$\tilde{a}_i = 1 \quad \iff \quad \mathbb{I}_{\text{syn}}(\tilde{P}_i) = 1 \text{ and } V_i \subsetneq \tilde{V}_i. \tag{41}$$

*That is, a teacher repair is accepted if and only if it is syntactically valid and passes a* strict superset *of the visible tests its candidate passed.*

*Proof.* ($\Rightarrow$) Suppose $\tilde{a}_i = 1$. As a product of $\{0,1\}$-valued indicators, every factor equals one, so $\mathbb{I}_{\text{syn}}(\tilde{P}_i) = 1$. By Property 1 the non-regression factor gives $V_i \subseteq \tilde{V}_i$, and by Property 2 the strict-gain factor gives $\tilde{V}_i \setminus V_i \neq \emptyset$; together $V_i \subsetneq \tilde{V}_i$.

($\Leftarrow$) Suppose $\mathbb{I}_{\text{syn}}(\tilde{P}_i) = 1$ and $V_i \subsetneq \tilde{V}_i$. We verify each of the four factors. *Syntax:* equals one by hypothesis. *Non-regression:* $V_i \subseteq \tilde{V}_i$ means $\tilde{r}_{ij} = 1$ for every $\tau_j \in V_i$, so $\prod_{\tau_j \in V_i} \tilde{r}_{ij} = 1$. *Strict gain:* $\tilde{V}_i \setminus V_i \neq \emptyset$ supplies some $\tau_j$ with $\tilde{r}_{ij} = 1$ and $\tau_j \notin V_i$; since $V_i$ and $F_i$ partition $T_{\text{pub}} \cup T_{\text{cf}} \supseteq \tilde{V}_i$, we have $\tau_j \in F_i$, hence $\sum_{\tau_j \in F_i} \tilde{r}_{ij} \geq 1$ and the factor is one. *Non-decrease:* restricting the inclusion $V_i \subseteq \tilde{V}_i$ to public tests gives $V_i \cap T_{\text{pub}} \subseteq \tilde{V}_i \cap T_{\text{pub}}$, so $\sum_{\tau_j \in T_{\text{pub}}} \tilde{r}_{ij} = |\tilde{V}_i \cap T_{\text{pub}}| \geq |V_i \cap T_{\text{pub}}| = \sum_{\tau_j \in T_{\text{pub}}} r_{ij}$; dividing by $|T_{\text{pub}}|$ yields $\tilde{s}_i^{\text{pub}} \geq s_i^{\text{pub}}$, so the factor is one. All four factors equal one, hence $\tilde{a}_i = 1$. □

Which factors are non-redundant. The ($\Leftarrow$) direction shows the non-decrease factor is *implied* by non-regression once the partitioned suite $V_i \cup F_i$ contains the public tests: it is retained only as an explicit implementation check and does not change the accepted set. The remaining three factors are each non-redundant, witnessed by a repair that satisfies the other two (plus non-decrease) yet violates the strict-superset characterization:

- *Syntax.* A non-parseable $\tilde{P}_i$ has no well-defined execution outcomes $\tilde{r}_{ij}$; dropping the syntax factor would attempt to verify $V_i \subsetneq \tilde{V}_i$ on a program that cannot be run.
- *Strict gain.* The identity repair $\tilde{c}_i = c_i$ gives $\tilde{V}_i = V_i$, satisfying syntax, non-regression, and non-decrease, but $V_i \not\subsetneq \tilde{V}_i$; without the strict-gain factor the teacher would be "accepted" for reproducing the candidate unchanged.
- *Non-regression.* A repair that newly passes one test $\tau_a \in F_i \cap T_{\text{cf}}$ while losing one previously passing counterfactual test $\tau_b \in V_i \cap T_{\text{cf}}$ (leaving all public outcomes fixed) satisfies syntax, strict gain, and non-decrease, yet $\tilde{V}_i \not\supseteq V_i$, so it is not a superset; without the non-regression factor such a lateral trade would be accepted.

Thus the operative content of the rule is exactly the two-part condition of Proposition 1, enforced by three independent factors plus one redundant safeguard.

Finite-suite progress bound. Proposition 1 also bounds the number of accepted repairs when the visible suite is held fixed. Let $V^{(t)}$ be the pass-set after the $t$th accepted repair and let $M = |T_{\text{pub}} \cup T_{\text{cf}}|$. Repeated acceptance gives the strict chain

$$V^{(0)} \subsetneq V^{(1)} \subsetneq \cdots \subsetneq V^{(K)} \subseteq T_{\text{pub}} \cup T_{\text{cf}}. \tag{42}$$

Each accepted step adds at least one visible test and removes none, so

$$K \leq M - |V^{(0)}|. \tag{43}$$

The process therefore cannot cycle through accepted repairs on a fixed finite suite. This bound does not apply unchanged if new counterfactual tests are added between rounds, because the ambient set and its cardinality then change.

Scope of the guarantee. The strict-superset result is deliberately limited to the observed visible suite. It does not imply semantic correctness, held-out success, or improvement on inputs not represented by $T_{\text{pub}} \cup T_{\text{cf}}$:

$$\tilde{a}_i = 1 \not\Longrightarrow h_i = 1. \tag{44}$$

It also does not by itself certify that the edit is minimal or confined to a particular source region; edit-scope restrictions require a separate syntactic or semantic check. The acceptance rule supplies a finite behavioral non-regression certificate, not a proof that the repaired program implements the full specification.

Result. The acceptance rule is a conjunction of four indicators that together guarantee: (1) non-regression on $V_i$, (2) strict gain on $F_i$, (3) no dependence on $T_{\text{hid}}$, and (4) non-vacuity for public-perfect candidates with counterfactual failures. The repaired program, when accepted, passes a strict superset of the visible tests its candidate passed, and this decision is made without consulting any held-out outcome.

### C.3 Properties of the Discrimination Score

Setup. For an executable counterfactual test $\tau_j$ run over $n$ candidates, each candidate $i$ produces an outcome $r_{ij} \in \{0, 1\}$. The pool consensus is the majority vote

$$\bar{r}_j = \mathbb{I}\left[\frac{1}{n}\sum_{i=1}^{n} r_{ij} \geq \tfrac{1}{2}\right] \in \{0, 1\}, \tag{45}$$

and the discrimination score is the fraction of candidates that disagree with the consensus:

$$d_j = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[r_{ij} \neq \bar{r}_j]. \tag{46}$$

This subsection derives the range, extremal values, an exact closed form in terms of the passing count, the behavior under candidate deletion, the relationship to entropy, and the oracle-independence of the score.

Step 1: range. Since $d_j$ is the fraction disagreeing with the majority (with ties assigned to pass), at most $\lfloor n/2 \rfloor$ candidates can disagree. Hence

$$0 \leq d_j \leq \frac{\lfloor n/2 \rfloor}{n} \leq \frac{1}{2}. \tag{47}$$

The exact bound follows again from the closed form below.

Step 2: closed form in the passing count. Let $m_j = \sum_{i=1}^{n} r_{ij}$ be the number of candidates passing $\tau_j$. Two cases arise. *Case A,* $m_j \geq \lceil n/2 \rceil$*:* then $\bar{r}_j = 1$, and $\mathbb{I}[r_{ij} \neq \bar{r}_j] = \mathbb{I}[r_{ij} = 0]$, which counts the $n - m_j$ failing candidates, so

$$d_j = \frac{n - m_j}{n} \quad (m_j \geq \lceil n/2 \rceil). \tag{48}$$

*Case B,* $m_j < \lceil n/2 \rceil$*:* then $\bar{r}_j = 0$, and $\mathbb{I}[r_{ij} \neq \bar{r}_j] = \mathbb{I}[r_{ij} = 1]$, which counts the $m_j$ passing candidates, so

$$d_j = \frac{m_j}{n} \quad (m_j < \lceil n/2 \rceil). \tag{49}$$

Combining both cases,

$$d_j = \frac{\min\{m_j,\, n - m_j\}}{n}, \tag{50}$$

i.e. the discrimination score is the minority fraction of the pool. This closed form makes $d_j$ symmetric in $m_j$ and $n - m_j$: passing and failing are interchangeable, which is the point: the score measures disagreement, not correctness.

Step 3: extremal values. From the closed form, $d_j = 0$ iff $\min\{m_j, n - m_j\} = 0$, i.e. iff $m_j \in \{0, n\}$: either every candidate fails or every candidate passes, full consensus. The maximum of $\min\{m_j, n - m_j\}$ over $m_j \in \{0, \ldots, n\}$ is $\lfloor n/2 \rfloor$, attained at $m_j \in \{\lfloor n/2 \rfloor, \lceil n/2 \rceil\}$, giving

$$d_j^{\max} = \frac{\lfloor n/2 \rfloor}{n} \leq \frac{1}{2}. \tag{51}$$

Thus the score never exceeds $1/2$: a test that splits the pool exactly in half is maximally discriminating. For a single dissenting candidate, $m_j \in \{1, n - 1\}$ and the minority count is one, so $d_j = 1/n \to 0$, not a value near one. The discriminating regime is therefore $d_j \in [0, 1/2]$, with $d_j \approx 1/2$ being the most informative.

Step 4: behavior under candidate deletion. Suppose a candidate is removed from the pool, $n \mapsto n - 1$, and $m_j \mapsto m_j - r_{\cdot j}$ for that candidate. Provided the removal does not change the consensus class, deleting a majority candidate leaves the minority count unchanged while reducing the denominator, so $d_j$ rises slightly; deleting a minority candidate reduces the minority count, so $d_j$ drops. At a tie or a one-vote margin, however, deletion can flip the consensus class and either trend may fail. Hence $d_j$ is not invariant under subsampling, and comparisons across pools of different sizes should recompute the closed form rather than infer a monotone change.

Step 5: relationship to entropy. The binary entropy of the passing rate $p_j = m_j/n$ is $H_2(p_j) = -p_j \log_2 p_j - (1-p_j)\log_2(1-p_j)$, which is maximized at $p_j = 1/2$ and zero at $p_j \in \{0, 1\}$. From the closed form, $d_j = \min\{p_j, 1-p_j\}$. By the symmetry $H_2(p) = H_2(1-p)$, this gives the exact identity

$$H_2(p_j) = H_2(d_j). \tag{52}$$

Thus $d_j$ discards the orientation of the split, meaning whether passing or failing is the majority, but preserves its binary entropy exactly. It is a piecewise-linear parameterization of pool heterogeneity, bounded by $1/2$ and strictly ordered with entropy on that interval.

Step 5b: expected score under an exchangeable Bernoulli model. To characterize $d_j$ beyond a fixed table, model the $n$ candidate outcomes on a fixed valid test as exchangeable Bernoulli draws $r_{ij} \sim \mathrm{Bern}(p)$, so $m_j = \sum_i r_{ij} \sim \mathrm{Binomial}(n, p)$ and $\hat{p} = m_j/n$. Using the closed form $d_j = \min\{\hat{p}, 1-\hat{p}\} = \frac{1}{2} - \left|\hat{p} - \frac{1}{2}\right|$ from Step 2, the expected discrimination score is

$$\mathbb{E}[d_j] = \tfrac{1}{2} - \mathbb{E}\left|\hat{p} - \tfrac{1}{2}\right| = \tfrac{1}{2} - \frac{1}{n}\sum_{m=0}^{n}\left|m - \tfrac{n}{2}\right|\binom{n}{m}p^m(1-p)^{n-m}. \tag{53}$$

Two consequences follow. First, by the law of large numbers $\hat{p} \to p$ almost surely, so $d_j \to \min\{p, 1-p\}$: the finite-pool score is a consistent estimator of the population minority rate, and the $d_j \le \frac{1}{2}$ bound of Step 3 is attained in the limit only at $p = \frac{1}{2}$. Second, the finite-$n$ score is biased *downward* relative to the limit, $\mathbb{E}[d_j] \le \min\{p, 1-p\}$ by Jensen's inequality applied to the concave map $\hat{p} \mapsto \min\{\hat{p}, 1-\hat{p}\}$, with the gap of order $n^{-1/2}$: at $p = \frac{1}{2}$, $\mathbb{E}|\hat{p} - \frac{1}{2}| = \sqrt{1/(2\pi n)} + O(n^{-1})$ by the mean absolute deviation of the binomial, so

$$\mathbb{E}[d_j]\Big|_{p=1/2} = \frac{1}{2} - \sqrt{\frac{1}{2\pi n}} + O(n^{-1}). \tag{54}$$

For the pools used here, $n = 4$, the score takes only three values:

$$d_j = \begin{cases} 0, & m_j \in \{0, 4\}, \\ \frac{1}{4}, & m_j \in \{1, 3\}, \\ \frac{1}{2}, & m_j = 2. \end{cases} \tag{55}$$

Under the Bernoulli model, their probabilities are

$$\Pr(d_j = 0) = (1-p)^4 + p^4, \tag{56}$$

$$\Pr(d_j = \tfrac{1}{4}) = 4p(1-p)^3 + 4p^3(1-p), \tag{57}$$

$$\Pr(d_j = \tfrac{1}{2}) = 6p^2(1-p)^2, \tag{58}$$

and hence

$$\mathbb{E}[d_j] = p(1-p)(1+p-p^2). \tag{59}$$

At the maximally heterogeneous population value $p = 1/2$, this expectation is $5/16 = 0.3125$, below the population limit $1/2$. The exact calculation makes clear that a four-candidate pool yields a coarse signal, which is why $d_j$ is used only as a necessary screening statistic and the fine-grained weighting is delegated to the sample-size-corrected $\hat{I}$ of Appendix C.1. Concentration follows from Hoeffding's inequality, $\Pr[|\hat{p} - p| \ge t] \le 2e^{-2nt^2}$, giving $\Pr[|d_j - \min\{p, 1-p\}| \ge t] \le 2e^{-2nt^2}$ because $d_j$ is 1-Lipschitz in $\hat{p}$; the score therefore stabilizes at rate $O(n^{-1/2})$ as the candidate pool grows.

Step 6: oracle-independence. The consensus $\bar{r}_j$ is a function of $\{r_{ij}\}$ only, not of the reference implementation's oracle. Whether a test is valid depends on the reference implementation, but once the test is admitted to $T_{\mathrm{cf}}$, the score $d_j$ uses only candidate outcomes. Formally, $d_j = \phi(r_{1j}, \ldots, r_{nj})$ for a symmetric function $\phi$ that does not reference the oracle. Consequently, replacing the oracle (e.g. using a stronger reference) changes *which* tests are valid but not the value of $d_j$ on any fixed valid test. The score is therefore a within-pool statistic, not an agreement-with-oracle statistic; in particular, it cannot be used as the correctness label, and the framework never uses it as one.

Step 7: selection implication. A test with $d_j = 0$ provides no separation (all candidates agree, full consensus) and contributes nothing to selection or weighting. A test with $d_j$ close to $1/2$ splits

the pool evenly and is maximally useful for separating behavioral classes. However, $d_j$ measures *disagreement*, not *correctness*: a test on which all candidates are wrong has $d_j = 0$ but is uninformative for selection, whereas a test on which half are right and half wrong has $d_j \approx 1/2$ and is highly informative. The reference oracle, not $d_j$, distinguishes these. The discrimination score is thus a necessary condition for a test to contribute to selection (zero discrimination means no contribution) but not a sufficient one (the majority class must also be the correct class, which only the oracle establishes). The information-gain weight of Appendix C.1 combines both signals: it weights evidence by $I(E;Y)$, which is high only when $E$ both separates the pool and correlates with acceptance.

Result and summary. The discrimination score admits the closed form $d_j = \min\{m_j, n-m_j\}/n \in [0, 1/2]$, is symmetric in passing/failing, preserves the binary entropy of the pool split through $H_2(p_j) = H_2(d_j)$, is invariant to oracle replacement on any fixed valid test, and is a necessary-but-not-sufficient condition for selection usefulness. For $n = 4$, it is restricted to $\{0, 1/4, 1/2\}$ and has expectation $5/16$ at $p = 1/2$. The corrected upper bound $1/2$ (not 1) refines the main-text statement: maximally discriminating tests split the pool in half.

### C.4 Expanded Assumptions and Notation

The derivations above use three distinct spaces that should not be conflated. The task space contains task descriptions and source programs, the execution space contains finite pass/fail records, and the learning space contains token sequences emitted by the teacher or student. Let $q = (x, P, T_{mathrmpub}, T_{mathrmhid})$ denote one task, let $C_q$ be the finite candidate pool, and let $T_{mathrmvis} = T_{mathrmpub} cup T_{mathrmcf}$ be the visible suite after counterfactual validation. The held-out suite is disjoint from $T_{mathrmvis}$ by construction. For candidate $i$, define the visible execution vector

$$R_i = \big(r_{ij} : \tau_j \in T_{mathrmvis}\big), \qquad r_{ij} \in \{0, 1\}, \tag{60}$$

and define the repair execution vector $\widetilde{R}_i$ analogously. The acceptance label $Y_i = \widetilde{a}_i$ is a deterministic function of $(R_i, \widetilde{R}_i, \mathbb{I}_{\mathrm{syn}}(\widetilde{P}_i))$. It is therefore a training-time behavioral label, not a semantic correctness label. This distinction is essential when interpreting both the information-gain weight and the value-model metrics.

We use $\mathbb{E}_Q[f] = |Q|^{-1} \sum_{i \in Q} f_i$ for the uniform average over a finite candidate pool. When a quantity is estimated over many attempts, the index set must be fixed before counting outcomes. Otherwise, defining an evidence variable after observing the acceptance label creates selection bias: the evidence item would be chosen because it already appears predictive. All information estimates in this appendix assume that the evidence definition and its coding rule are fixed before the corresponding contingency table is formed.

### C.5 Mutual Information as a Reduction in Label Uncertainty

For a binary acceptance label, write $\pi = \Pr(Y = 1)$. Then

$$H(Y) = h_2(\pi) = -\pi \log \pi - (1 - \pi) \log(1 - \pi). \tag{61}$$

If an evidence item takes values $e \in \mathcal{E}$, Bayes' rule gives

$$p(y \mid e) = \frac{p(e \mid y) p(y)}{p(e)}. \tag{62}$$

Substituting this expression into the conditional entropy shows exactly how evidence changes the label distribution:

$$H(Y \mid E) = \sum_e p(e) h_2\big(p(Y = 1 \mid E = e)\big), \tag{63}$$

$$I(E;Y) = h_2(\pi) - \sum_e p(e) h_2\big(p(Y = 1 \mid E = e)\big). \tag{64}$$

Thus $I(E;Y)$ is large when evidence values induce posterior acceptance probabilities that differ from the prior $\pi$. It is zero when every posterior equals $\pi$, even if the evidence appears frequently. Conversely, a rare evidence value can have a large conditional effect but still contribute little total

information because it is multiplied by its small marginal probability $p(e)$. This explains why the estimator must use joint frequencies rather than only the conditional success rate on the positive evidence subset.

For binary evidence $E \in \{0, 1\}$, define $\pi_e = \Pr(Y = 1 \mid E = e)$ and $\rho = \Pr(E = 1)$. Then

$$I(E; Y) = h_2(\pi) - \rho h_2(\pi_1) - (1 - \rho)h_2(\pi_0). \tag{65}$$

The evidence has no information when $\pi_0 = \pi_1 = \pi$. If $\pi_1 = 1$ and $\pi_0 = 0$, the evidence determines $Y$ and $I(E; Y) = h_2(\pi)$. This gives a direct interpretation of the upper bound used earlier: no evidence can reduce uncertainty below zero, so it cannot provide more information than the original label entropy.

### C.6 Finite-Sample Information Estimates and Smoothing

Let $N_{ey}$ be the four cells of a binary evidence–label table. The empirical mutual information can also be written as a weighted divergence between the empirical conditional label distributions and the empirical prior:

$$\widehat{I}_{\text{plug}}(E; Y) = \sum_{e \in \{0,1\}} \widehat{p}_{e\cdot} D_{\text{KL}}(\widehat{p}(\cdot \mid e) \,\|\, \widehat{p}(\cdot)), \tag{66}$$

where $D_{\text{KL}}(u\|v) = \sum_y u_y \log(u_y/v_y)$. This representation makes non-negativity of the raw plug-in estimate explicit, but it does not remove its finite-sample upward bias. With sparse cells, a single accepted observation can make $\widehat{p}(y \mid e)$ equal to one, producing an apparently perfect predictor even when the evidence definition has not been tested often enough.

The Miller–Madow correction used in the implementation follows from the first-order entropy bias. For an empirical distribution with $k$ non-empty cells,

$$\mathbb{E}[\widehat{H}] - H = -\frac{k - 1}{2N} + O(N^{-2}). \tag{67}$$

Applying this expansion to $H(E) + H(Y) - H(E, Y)$ gives

$$\mathbb{E}[\widehat{I}_{\text{plug}}] - I = \left[-\frac{r - 1}{2N}\right] + \left[-\frac{s - 1}{2N}\right] - \left[-\frac{rs - 1}{2N}\right] + O(N^{-2}) \tag{68}$$

$$= \frac{(r - 1)(s - 1)}{2N} + O(N^{-2}). \tag{69}$$

The correction is only first-order. It does not make the estimate exactly unbiased, and it should not be interpreted as a confidence interval. The truncation $\max\{0, \widehat{I}_{\text{MM}}\}$ enforces the population constraint $I \geq 0$ but introduces a small boundary bias whenever the corrected estimate is negative. This is preferable to allowing a negative training weight, which would reverse the gradient direction for that example.

A smoothed alternative is a Dirichlet or additive correction $N'_{ey} = N_{ey} + \alpha$. Such smoothing avoids zero conditional probabilities, but it changes the effective prior and can dominate when $N$ is very small. We therefore keep the correction explicit and separate from any smoothing assumption. The reported derivation is valid for the non-empty support correction; if an experiment uses a different estimator, the estimator and its bias correction must be stated separately.

### C.7 Weighting, Sampling, and the Optimization Objective

Let $L_i(\theta)$ denote the token-level repair loss for accepted example $i$. The weighted empirical objective is

$$\mathcal{L}_w(\theta) = \frac{1}{|Q|} \sum_{i \in Q} w_i L_i(\theta). \tag{70}$$

Multiplying all weights by one positive constant changes the gradient magnitude but not the set of minimizers of the unconstrained objective. Consequently, the practical denominator $\bar{I}_Q + \epsilon$ mainly fixes scale, while the relative ratios $w_i/w_k$ determine which examples receive greater emphasis. This

statement does not hold once optimization uses finite step sizes, clipping, early stopping, or regularization; in those cases the absolute scale can affect the trajectory even when the formal minimizer is unchanged.

Without clipping and with $\epsilon = 0$, define the information-proportional distribution

$$q_I(i) = \frac{\widehat{I}_i}{\sum_{k \in Q} \widehat{I}_k}. \tag{71}$$

Then

$$\mathcal{L}_w(\theta) = \frac{\sum_{k \in Q} \widehat{I}_k}{|Q|} \sum_{i \in Q} q_I(i) L_i(\theta). \tag{72}$$

The weighted loss is therefore proportional to an expectation under $q_I$, but it is not identical to changing the data distribution unless the optimizer also accounts for the leading constant. With clipping, there is no exactly equivalent $q_I$ because the transformation $\widehat{I}_i \mapsto \mathrm{clip}(w_i)$ changes relative ratios. Clipping is thus a deliberate robustness modification rather than a mere numerical implementation detail.

For a single stochastic gradient draw $I \sim Q$, the unbiased estimator of the unclipped gradient is $w_I \nabla L_I$. Its variance decomposes as

$$\mathrm{Var}_Q(w_I \nabla L_I) = \mathbb{E}_Q[w_I^2 \|\nabla L_I\|^2] - \|\mathbb{E}_Q[w_I \nabla L_I]\|^2 . \tag{73}$$

The first term is controlled by the upper clip, while the second term depends on gradient alignment. Therefore, the second-moment calculation in the main derivation is a bound on fluctuation scale, not a complete convergence theorem. In particular, information gain can be high for examples whose gradients conflict; weighting them more heavily may increase optimization variance even when the evidence is predictive of acceptance.

### C.8 Detailed Acceptance-Rule Proof and Counterexamples

The acceptance rule can be analyzed by writing the visible pass-set of the candidate as $V_i$ and the repaired pass-set as $\widetilde{V}_i$. The syntax indicator is logically separate because a non-parsing patch has no well-defined pass-set. Conditional on syntactic validity, the remaining factors impose three set constraints:

$$V_i \subseteq \widetilde{V}_i \qquad \text{(no regression)}, \tag{74}$$

$$\widetilde{V}_i \setminus V_i \neq \emptyset \qquad \text{(strict gain)}, \tag{75}$$

$$|\widetilde{V}_i \cap T_{\mathrm{pub}}| \geq |V_i \cap T_{\mathrm{pub}}| \qquad \text{(public non-decrease)}. \tag{76}$$

The third constraint is redundant when the first holds because $T_{\mathrm{pub}} \subseteq T_{\mathrm{vis}}$. It is retained in the implementation because it makes the public non-regression requirement visible in logs and guards against an implementation bug in which $V_i$ is constructed from only counterfactual tests.

The strict-gain factor is intentionally defined over $F_i$, the tests failed by the original candidate, rather than over all tests. If it were defined as $\sum_{\tau \in T_{\mathrm{vis}}} \widetilde{r}_{i\tau} \geq 1$, an identity repair would pass the factor whenever the candidate already passed at least one visible test. Using $F_i$ prevents that vacuous acceptance. Conversely, requiring a strict public-score increase would reject public-perfect candidates, even when a valid counterfactual exposes a real failure. The chosen rule therefore accepts a repair that fixes at least one previously failed visible behavior while preserving every previously passed behavior.

The set characterization also shows why acceptance is monotone only with respect to the fixed visible suite. If a new counterfactual test is added after a repair, the ambient set changes from $T_{\mathrm{vis}}$ to $T'_{\mathrm{vis}}$, and a previously accepted repair may fail the new test. This is not a contradiction: the theorem compares two programs on one fixed suite. In an iterative implementation, the suite should be versioned so that the acceptance log records which tests were present when each repair was accepted.

### C.9 Oracle Boundaries and the Scope of the Proof

The no-leakage argument is conditional on the locked split between visible and held-out tests. It establishes that the acceptance indicator is measurable with respect to visible executions. A canonical reference changes the distribution of admitted counterfactual tests while remaining outside every model input. Consequently, the reference-free and reference-assisted rows estimate a deployment protocol and a validation diagnostic, respectively.

The same distinction applies to the phrase "behavioral oracle." Public tests provide an executable acceptance signal, while a canonical reference provides an additional signal for validating generated tests. The reference-free pipeline admits tests through interface checks, execution completion, and consistency constraints. Deployment claims therefore use the reference-free row, while reference-assisted results quantify the effect of validation access.

### C.10 Cost Decomposition and Controlled Attribution

For a task $q$, decompose the total online cost into student generation, teacher repair, counterfactual execution, repair verification, and selector overhead:

$$C(q) = C_{\text{student}}(q) + C_{\text{teacher}}(q) + C_{\text{cf}}(q) + C_{\text{verify}}(q) + C_{\text{select}}(q). \tag{77}$$

The reported model-call and token counts measure model invocations, while sandbox executions contribute to wall time. The primary HumanEval+ and SWE-bench comparisons jointly report calls, tokens, executions, and wall time through their extra-call controls. Cross-repository transfer rows are presented as a separate exploratory family.

Let $A$ denote the full Test-to-Teach protocol and $A_{-k}$ the ablation that removes component $k$. The observed contrast

$$\Delta_k = \widehat{\mathrm{SR}}(A) - \widehat{\mathrm{SR}}(A_{-k}) \tag{78}$$

summarizes the performance change induced by removing component $k$ under the stated protocol. We interpret it as a paired protocol contrast and combine it with the direct execution audit in Table 23, which records generated tests, exposed errors, and successful repairs.

### C.11 Checklist for Applying the Formal Results

For each new experiment, the visible and held-out suites are locked before hidden evaluation; every evidence variable is fixed before acceptance labels are counted; empty evidence sets use the stated zero-information fallback; the clipping interval and $\epsilon$ are fixed before method comparison; candidate-pool size is recorded; reference-free and reference-assisted rows are labelled separately; and task–seed repetitions use clustered inference. These conditions align the empirical protocol with the algebraic statements in this appendix.

## D Implementation Details

### D.1 Sandbox and Execution Configuration

All candidate and teacher programs run in identical sandboxes with a 10-second wall-clock timeout per test execution, a 512 MB memory cap, and the same dependency manifest. The reference implementation used to validate counterfactual oracles runs under the same limits. A generated counterfactual test is marked invalid on timeout, exception, failed assertion, non-executable transformation, or, in the reference-assisted setting, a missing reference implementation. Test executions are batched per candidate to amortize sandbox startup; the per-execution cost is included in the call and token counts of Table 6 only when it triggers a model invocation, not when it runs in the sandbox alone.

### D.2 Prompt Templates

The student repair prompt contains the task description, the original program, the public tests, and an instruction to emit a minimal patch; the teacher prompt additionally contains the changed-file

summary, the public and valid-counterfactual outcomes, the failure diagnosis, and the local behavioral constraints, and instructs the teacher to produce a diagnosis, a failure-dependency graph, and a minimal patch without consulting any reference solution. Exact prompt text is released with the code.

## E ILLUSTRATIVE FAILURE SCENARIO

Consider a function whose specification requires returning a sentinel for an empty list. A candidate may pass visible non-empty examples but raise an indexing exception on the empty input. A boundary test can reveal this behavior; in the reference-assisted setting, its expected output is checked against a reference implementation. The evidence can motivate a guard before indexing. This is an illustrative scenario, not a task-level experimental result or evidence that one prompting method outperforms another.

## F AI USE DISCLOSURE

In this work, we used generative AI tools to assist with manuscript writing, language editing, LaTeX formatting, table organization, and consistency checks between the reported results and their corresponding analyses. Generative AI tools were also used to help revise the description of experimental results, update and format experimental tables, and check mathematical expressions, citations, and cross-references.