Evidence-Centered Harness Engineering for Fail-Closed Neuro-Symbolic Autograding

Omar Ghabayen*, Chujun Tao†, and Kangshuo Li‡
Carnegie Mellon University

***Abstract*—We present Evidence-Centered GradeTrace (NSMC GradeTrace), an evidence-centered harness-engineering methodology that keeps language models in a proposal role. Course materials are compiled into evidence spans and Formal Concept Analysis concepts; candidate tests enter Stable or Shadow suites only after deterministic checks; and missing evidence or gate failure withholds the automated grade rather than guessing. The paper's central contribution is a reportability discipline for LLM-assisted grading: a three-level denominator separates corpus/source-limited exclusions, corpus-valid abstentions, and reportable rows, so coverage is not silently folded into accuracy. In a fixed FalconCode/CS110 run using FCA concept discovery and a Gemini-3-Flash proposer via Vertex, a raw extraction of $44{,}007$ rows yields $7{,}969$ corpus/source-limited exclusions, a $36{,}038$-row corpus-valid denominator, and $30{,}234$ reportable rows. The corresponding rates are 81.9% corpus observability, 83.9% corpus-valid system reportability, and 68.70% raw end-to-end yield. On reportable rows, GradeTrace agrees with FalconCode unit-test reference scores at Pearson $r = 0.805$ and Spearman $\rho = 0.791$, with bias -2.309, median absolute error 0.000, and RMSE 17.800. The remaining large-error tail ($1{,}653$ rows with $|\text{diff}| \geq 25$) is concentrated in all-or-nothing tests lacking partial-credit rubrics. These results define a reproducible single-corpus measurement study, not a production-grader claim or a comparison against external graders or human instructor grades.**

***Index Terms*—automated grading, neuro-symbolic, harness engineering, formal concept analysis, evidence-centered assessment, fail-closed systems, measurement discipline**

I. Introduction

Autograders are central to scalable programming education, returning correctness feedback to large cohorts at near-zero marginal cost. Large language models promise to extend them from black-box pass/fail checkers into formative systems that explain failures, apply rubrics to open-ended work, and generate context-aware feedback [1], [2]. That promise carries a hazard specific to assessment: an LLM asked to grade will almost always return *a* grade, including for submissions whose grounding evidence the system never actually recovered. Reported agreement then silently mixes two unlike quantities—how often the grader is right when it can see the relevant material, and how much of the material it could see at all.

Evidence-Centered GradeTrace takes a neuro-symbolic harness-engineering stance that confronts this hazard directly: the language model proposes, but symbolic structure and executable verification decide what is admissible. The system compiles course materials into a concept inventory anchored to evidence spans, generates and gates candidate tests into Stable and Shadow suites, and scores each submission under a *fail-closed* contract that withholds an automated grade—routing the case to human review—whenever evidence is insufficient or a verification gate fails (Figure 1). Grading thus becomes a measurement problem in which every reportable score is traceable from course evidence to a concept, a test, a gate, and a final grade, and in which the cases the system declines to score are first-class, accounted-for outcomes rather than hidden ones.

Contributions.

- **A measurement and reportability discipline for LLM-assisted autograding.** We define a three-level denominator that separates *corpus/source-limited exclusions* (rows the recovered course subset cannot ground) from within-corpus *eligibility failures* and from genuine fail-closed *system abstentions*, and we report the three rates these levels induce—corpus observability, corpus-valid system reportability, and raw end-to-end yield—so that coverage is never conflated with accuracy. We cast this as evidence-conditioned educational measurement under partial corpus observability.
- **An evidence-centered harness-engineering pattern.** Every reportable score is traced from course evidence to concept, test, gate, and final grade; when the supporting evidence is missing the system withholds rather than guesses, making abstention a recorded decision. The pattern is deliberately operational: provenance files, suite manifests, gate outputs, and score artifacts are treated as part of the grader rather than as after-the-fact reporting.
- **A transparent empirical study with a failure taxonomy.** Instantiated with Formal Concept Analysis concept discovery and a Gemini-3-Flash proposer (via Vertex) on FalconCode/CS110 [3], the system attains Pearson $r = 0.805$ / Spearman $\rho = 0.791$ on $30{,}234$ reportable rows; we then localize the residual error to a single dominant structural cause—all-or-nothing tests that lack partial-credit rubrics—and publish the full discard accounting and diagnostic scoring views behind it.

Scope. This is a single-corpus measurement study, not a production-grading claim. The reference scores are the FalconCode corpus unit-test scores, not human instructor grades, and all results are reported under the explicit denominator boundary above.

Roadmap. We first survey LLM-based autograding and its fragmented evaluation practice (Related Work), then detail the evidence-centered fail-closed pipeline and its FCA concept-discovery method (Methods), present the benchmark construc-

tion and empirical results (Experiments and Results), and close with a failure-mode discussion, limitations, ethics, and reproducibility.

II. Related Work

We organize prior work by where the LLM enters the autograding pipeline, then by the reliability and auditability challenges that entry creates, and finally by how the field measures success.

A. LLMs across the autograding pipeline

Execution and rubric design. Work at the execution layer targets test suites and rubrics aligned with learning objectives [4]. AI-generated multiple-choice questions can match the quality of human-authored items, suggesting models can contribute to assessment design directly [5]. Beyond authoring, [6] proposes a three-stage rubric-generation, grading, and post-grading review loop that refines rubric quality from sampled answers, and [7] adds reflector/refiner agents that optimize grading guidelines by self-reflection and report improved alignment with human graders.

Feedback generation and analytics. A parallel line uses LLMs for personalized feedback: [1] lets instructors control feedback style and depth through prompt pooling, and [8] offers an evaluation framework covering motivational tone, strengths/weaknesses identification, structure, and hallucination. Even so, learner agency and the actionability of feedback remain weak in practice [9].

B. Reliability and auditability challenges

Inserting an LLM raises fairness, reliability, and auditability concerns. Models exhibit latent biases such as stylistic favoritism and self-preference toward familiar outputs [10], and rubric interpretation drifts with model stochasticity [11]. As judges, LLMs are not neutral scorers: they show position and verbosity biases [12], [13], [14], favor their own generations [15], and remain only approximately aligned with human raters even in strong judge frameworks [16]. Aggregating heterogeneous error signals can dilute decision boundaries and produce conflicting updates [17], while black-box reasoning, central-tendency bias, and hallucinated justifications weakly linked to verifiable evidence undermine auditability [18], [19].

Several mitigations have emerged. Deterministic proposer-verifier designs separate semantic interpretation from executable checks to suppress hallucination [2], [20]; self-consistency and majority voting improve stability across repeated runs [21]; post-hoc calibration aligns score distributions with reference distributions, and evidence-grounded extraction constrains rationales to explicit support spans [18], [22], [23]. Human-in-the-loop review remains an important recalibration mechanism, though it is not a universal safeguard, since reviewers can override correct automated grades [24].

C. How the field measures autograders

A standardized benchmark for LLM-based autograders is still missing. As the coverage matrix in Table I shows, current systems report fragmented and inconsistent metric sets: some rely solely on accuracy, others on correlation coefficients, and few report jointly across error, agreement, confidence, and semantic-similarity dimensions. This inconsistency makes cross-system comparison nearly impossible and hides whether a gain in one dimension (say, correlation with reference grades) costs another (fairness or calibration). Read column-wise (Table I), coverage is highly uneven: agreement/correlation and error/regression metrics are widely reported, whereas hypothesis-testing, model-confidence, and semantic-similarity groups are sparse, and fairness or calibration are rarely reported at all—a gap consistent with broader surveys of automated programming-assessment tools [25]. NSMC GradeTrace deliberately spans the accuracy, reliability, quality, cost, and operations families rather than a single metric group, and pairs every score with the evidence that produced it, addressing exactly the auditability and reliability dimensions the LLM-as-judge literature shows to be most fragile.

Methodologically, our approach is neuro-symbolic: a model proposes candidate concepts and tests, but a symbolic concept lattice and executable verification gates decide what is admissible. This mirrors a line of work in which neural components guide, or are constrained by, symbolic search and verification—learning programs from input/output examples [26], graph representations of programs for neural reasoning [27], execution-guided program synthesis [28], and neural-guided loop-invariant generation checked by a symbolic verifier [29]. We import that discipline into grading, where the symbolic layer is what makes the model's judgments auditable and fail-closed.

III. Methods

Evidence-Centered GradeTrace is a six-stage, evidence-centered, *fail-closed* pipeline that turns a corpus of course materials into a defensible, trace-carrying grade. The defining design commitment is a strict separation between *proposal* and *authority*: large language models are used only to *propose* candidate requirements and test behaviors, while every decision that can influence a student's score is made by deterministic, replayable machinery. Whenever the available evidence is insufficient or any internal check fails, the system withholds the automated score and routes the item to human review rather than emitting an optimistic grade. We treat concept discovery and gating as *measurement-validity* problems rather than purely computational ones, in keeping with the requirement that high-stakes scores carry evidence for reliability, fairness, and score meaning [30].

A. Pipeline Overview

Figure 1 summarizes the six stages. Starting from a corpus of course materials $\mathcal{D}$ (specifications, slides, rubrics, and starter code in PDF, DOCX, PPTX, Markdown, and source form), the system performs: **(1)** evidence ingest into deterministic spans, **(2)** an iterative proposer loop that emits evidence-anchored candidate requirements, **(3)** concept discovery via Formal Concept Analysis (FCA), **(4)** test synthesis from

TABLE I

SELECTED EVALUATION-METRIC COVERAGE EXAMPLES FOR LLM-BASED AUTOMATED GRADING. ● = METRIC REPORTED. COLUMN-HEADER COLORS IDENTIFY THE SEVEN METRIC GROUPS (SEE LEGEND). ROWS ARE LIMITED TO CITED WORK PLUS EVIDENCE-CENTERED GRADETRACE; **TG/G/F** COLUMNS INDICATE TEST-ARTIFACT GENERATION / GRADING / FEEDBACK CONTRIBUTIONS. THIS TABLE IS A COMPACT POSITIONING AID, NOT A CITATION-COMPLETE LITERATURE SURVEY.

Paper	*TG*	*G*	*F*	*Sc*	*CL*	*MAE*	*RMSE*	*Len*	R^2	r	ρ	τ	κ	*QWK*	*ICC*	t	*MWU*	*Wil*	*TOST*	d	ψ	ζ	*Var*	*BS*	*Acc*	*PRF*	*EE*	*GE*
Automating Autograding [2]	●	●																							●	●	●	
Grade Like a Human [6]	●	●	●			●	●			●															●			
Confusion-Aware Rubric [17]	●	●		●										●	●						●				●			
RULERS [18]	●	●			●										●													
LLM-Powered Grading (GLO) [7]	●	●		●										●	●						●	●			●			
Rubric Is All You Need [20]		●	●					●		●	●	●	●		●													
CodEv [21]		●	●			●									●													●
Evidence-Centered GradeTrace (this work)	●	●				●	●	●		●	●														●		●	

Focus columns: **TG** = test/artifact generation; **G** = grading; **F** = feedback generation. **(1) LLM-as-Judge:** **Sc** = ordinal scoring scale (e.g. 0–5 self-rating); **CL** = checklist / per-criterion scoring. **(2) Error & Regression:** **MAE** = mean absolute error / mean-abs-diff; **RMSE** = root-mean-squared error (incl. NRMSE); **Len** = leniency (mean-normalised signed bias); $\mathbf{R^2}$ = adjusted R^2. **(3) Agreement & Correlation:** **r** = Pearson; ρ = Spearman; τ = Kendall τ_b; κ = Cohen κ; **QWK** = quadratic weighted κ; **ICC** = intraclass correlation coefficient. **(4) Hypothesis Testing:** $\mathbf{t}$ = paired t-test; **MWU** = Mann–Whitney U / Kruskal–Wallis; **Wil** = Wilcoxon signed-rank; **TOST** = equivalence test; $\mathbf{d}$ = Cohen d effect size. **(5) Model Confidence:** ψ = misconfidence score; ζ = OOD confidence indicator; **Var** = grading variance / min–max spread / IQR. **(6) Semantic Similarity:** **BS** = BERTScore (F1 / P / R). **(7) Classification:** **Acc** = accuracy (exact or ± 1); **PRF** = precision / recall / F1; **EE** = error exceedance rate; **GE** = G-Eval (LLM-as-judge quality score).

committed concepts, **(5)** a nine-gate promotion stage that admits tests to a *Stable* grading suite, and **(6)** fail-closed scoring of student submissions against that suite. Every stage writes versioned, serializable artifacts to the workspace, so each run is reproducible and any two runs (for example, across proposer models) are directly comparable. The remainder of this section describes each stage in execution order.

B. Stage 1: Evidence Ingest and Spans

Ingest converts heterogeneous course materials into a single authoritative substrate of *evidence spans*. Each loaded document is assigned a stable identifier $doc\_id = \text{SHA-256}(relpath : \text{SHA-256}(content))[:16]$, and its text is segmented into spans that record an excerpt, a content quote hash, a physical locator (page or slide), and an *authority rank*. Authority ranks are assigned by glob matching the document path against a policy-configured precedence list (rank 1 is highest, e.g., an official specification; lower ranks for ancillary material), so that downstream stages can prefer evidence drawn from more authoritative sources. All spans are persisted in a content-addressed store. Ingest is fully deterministic: identical inputs yield identical spans and identifiers, which is the precondition for the proof-carrying traces that every later stage extends.

C. Stage 2: Iterative Proposer Loop

Stage 2 produces the set of evidence-anchored *candidate* requirements that becomes the object set of the formal context in Stage 3. The loop is the only place where a language model enters the grading path, and it is deliberately constructed so that the model *proposes* but never *decides*. Each iteration performs four steps.

(i) Root-cause-guided span selection. Spans (and, in later iterations, test "arms") are selected by an upper-confidence-bound (UCB) bandit driven by a root-cause-analysis (RCA) reward that prioritizes evidence likely to close real coverage and discrimination gaps, balancing exploitation of high-reward regions against exploration of under-sampled material.

(ii) Tiered extraction. A cost-tiered language model extracts candidate claims from the selected spans. A cheaper tier (*tier-b*) runs first; on schema-validation failure the request is escalated to a stronger tier (*tier-a*). Each candidate is assigned one of eight schema types—requirement, ban, definition, note, example, API, complexity, or invariant—and must cite at least one supporting span identifier.

(iii) Deterministic validation. Proposed candidates are accepted only if they pass deterministic validators: schema conformance, non-empty evidence-span citations whose identifiers resolve in the span store, and type-consistency checks.

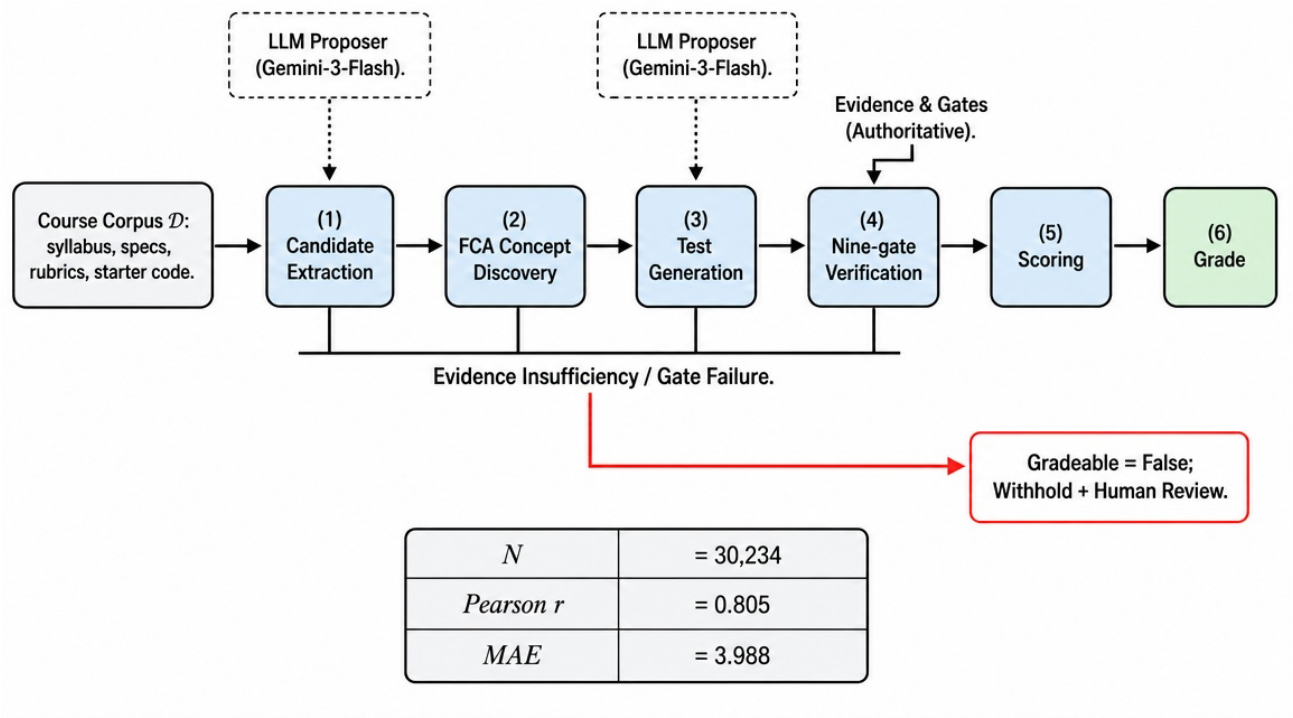

Fig. 1. End-to-end grading pipeline. From a course corpus $\mathcal{D}$ (syllabi, specifications, rubrics, starter code), the system extracts candidate requirements, discovers concepts via Formal Concept Analysis, generates tests, runs nine verification gates, scores, and emits a grade. The pipeline is *fail-closed*: whenever evidence is insufficient or any gate fails, the item is withheld and routed to human review rather than scored automatically.

Validators tier and may reject candidates regardless of model confidence; the model's output is treated as an untrusted proposal that must earn admission. This is the operational meaning of *the proposer is not the authority*.

(iv) Convergence. The loop runs FCA on the accepted candidates and computes convergence signals (document coverage and an out-of-vocabulary term ratio that tightens the extraction prompt when drift is detected). Iteration stops on a deterministic policy (a maximum-iteration cap, a patience window, and no-new-concept / no-new-candidate streaks), so the loop terminates without reference to wall-clock time.

D. Stage 3: Concept Discovery via Formal Concept Analysis

Concept discovery defines the grading vocabulary: a per-scope inventory of *concepts*, each anchored to evidence spans, that conditions every downstream test and score. We construct it with Formal Concept Analysis, which yields concepts whose identity is fixed by a closure operator rather than by a tunable similarity threshold, giving an exact and reproducible notion of "the same concept." Figure 2 illustrates the construction on a small synthetic context.

Formal context. Working per scope (a course–assignment pair, which bounds lattice size), let G be the set of evidence-anchored candidates from Stage 2 and let M be a fixed set of binary attributes. We form a *formal context* $\mathbb{K} = (G, M, I)$ with incidence relation $I \subseteq G \times M$, where $(g, m) \in I$ means candidate g carries attribute m [31]. Attributes are emitted by deterministic extractors organized into families: MARKER (policy signals from candidate type and keyword regex: `MUST`, `FORBIDDEN`, `DEFINITION`, `INVARIANT`, `RAISE_EXCEPTION`, `COMPLEXITY`); REGEX (regular-expression cue indicators); SECTION and SECTIONPATH (the first specification

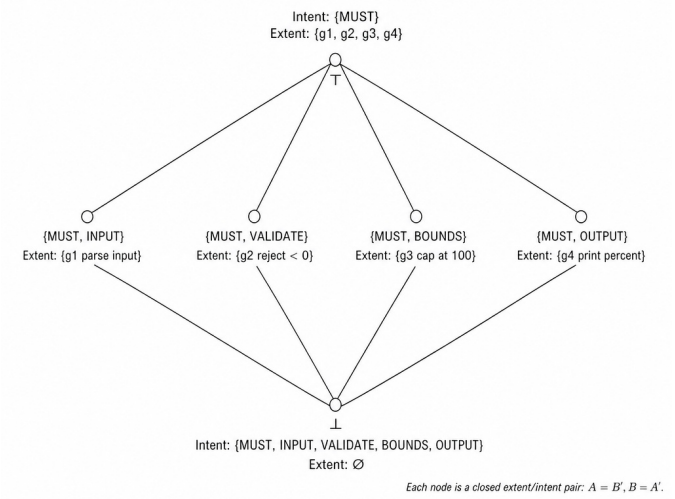

Fig. 2. Concept lattice (Hasse diagram) for a sanitized programming-assignment example. Objects g_1–g_4 are evidence-anchored candidate requirements: parse an input score, reject negative values, cap the score at 100, and print a final percentage. Attributes record deterministic requirement features such as INPUT, VALIDATE, BOUNDS, and OUTPUT; all four candidates share MUST. FCA orders formal concepts (A, B) where $A = B'$ and $B = A'$, so every displayed extent/intent pair is closed under $(\cdot)''$ and can serve as a reproducible grading vocabulary item.

heading and the full heading breadcrumb); LAYOUT (structural formatting cues: bold, bullet, code block); TERM, with a TERM_OTHER out-of-vocabulary catch-all (course-local vocabulary scored by a composite of IDF, PMI, χ^2, TextRank, and RAKE, with adaptive vocabulary sizing clamped to $[80, 800]$); PHRASE (multi-word keyphrases of two to four tokens); AST (lightweight import-derived code-constraint tags, a minor and rarely-decisive family in the current implementation); and SCOPE (week and assignment identifiers). All extraction is deterministic, so identical candidates yield an identical incidence matrix.

Lattice and closure. A *formal concept* of $\mathbb{K}$ is a pair (A, B) with $A \subseteq G$ and $B \subseteq M$ that is closed under the Galois derivation operators

$$\begin{aligned} A' &= \{m \in M \mid \forall\, g \in A\colon (g, m) \in I\}, \\ B' &= \{g \in G \mid \forall\, m \in B\colon (g, m) \in I\}, \end{aligned} \tag{1}$$

that is, $A' = B$ and $B' = A$: the intent B is the set of attributes shared by every candidate in the extent A, and A is exactly the set of candidates carrying every attribute in B. Under the order $(A_1, B_1) \leq (A_2, B_2) \iff A_1 \subseteq A_2$, the set of all formal concepts forms a complete lattice $\underline{\mathfrak{B}}(\mathbb{K})$ [31]. This lattice is *unique* for a given context: a mathematical guarantee, not a heuristic. We enumerate concepts with an off-the-shelf Galois-closure enumerator. Worst-case enumeration is exponential but is bounded in practice by the pruning below and by In-Close–style enumeration [32], with distributed variants extending feasibility to very large contexts [33], [34].

Pruning. We retain a concept node only if it satisfies three constraints: minimum support $|A| \geq 2$; bounded intent $|B| \leq 20$; and an attribute-family requirement that the intent B contain *both* (a) a *marker-like* attribute (a MARKER, AST, or SECTION tag) *and* (b) a *content-bearing* attribute (a TERM, PHRASE, AST, or SECTION tag, or a non-policy REGEX cue). Requiring both a marker and content prevents generic marker-only nodes (e.g., everything tagged `MUST`) from dominating the lattice while still anchoring every retained concept to substantive course vocabulary.

Iceberg fallback. For pathological contexts whose incidence matrix exceeds 1.5×10^6 cells, full enumeration is replaced by a seeded *iceberg* lattice [35]: closures are enumerated from up to 80 structural seed attributes (MARKER, AST, SECTION, REGEX, and SCOPE tags, taken in deterministic lexical order) together with the 300 highest-scoring TERM and PHRASE attributes, with the retained concept set capped at 800 nodes. The fallback trigger and every truncation are logged so the approximation is itself auditable.

Commitment and the subsumption graph. Each retained node yields a *concept draft* carrying a deterministic label (chosen by a priority hierarchy over attribute families), a definition anchored in evidence spans, and three SHA-256 signatures (intent, extent, lexical) truncated to 16 hexadecimal characters; the committed identifier is

$$\begin{aligned} \mathit{concept\_id} = \text{SHA-256}\big(&\mathit{course\_id} + \mathit{scope\_id} \\ &+ \mathit{canon\_def} + \mathit{intent\_sig}\big)[:16]. \end{aligned}$$

Drafts are merged by intent Jaccard similarity,

$$J(B_i, B_j) \;=\; \frac{|B_i \cap B_j|}{|B_i \cup B_j|}, \tag{2}$$

using the headline run's policy values: automatic merge at $J \geq 0.78$ and a review-queue flag for near-duplicates in $0.60 \leq J < 0.78$. The library defaults are stricter (0.85 merge, 0.70 review); the FalconCode run records the override in its policy artifact. A second automatic-merge path fires when two drafts share nearly identical evidence (evidence Jaccard ≥ 0.90) at borderline intent overlap. Intent Jaccard is the sole similarity used for merging. The lattice's cover edges (lower-neighbor relations) directly induce a SUBSUMES concept hierarchy at no additional cost. We emphasize that the lattice induces SUBSUMES edges *only*; REQUIRES relationships, where present, are mined separately from observed test co-failure patterns during audit and are *not* read off the lattice. The committed configuration is recorded as $\mathit{concept\_method} = \texttt{lcm}$.

Why FCA. The result is *proof-carrying*: every committed concept admits a complete trace from evidence span through attribute extraction and lattice node to the concept identifier, and concept identity is fixed by extent–intent closure rather than a threshold. Because all operations use sorted inputs, stable hashing, and deterministic tie-breaking with no random seeds, the inventory is bit-reproducible across runs, which is the property recent work argues makes FCA a suitable basis for interpretable, high-stakes decision structures [36]. We treat this determinism not as a free-standing software virtue but as a defense against construct-irrelevant variance in the resulting scores [37].

E. Stage 4: Test Synthesis

Committed concepts are authoritative; Stage 4 turns them into executable tests without letting a model override that authority. Each concept emits one or more *concept test requests*, which are realized through typed templates: `static_*` (AST and structural checks), `regex_*` (textual-pattern checks), and `runtime_*` (behavioral checks executed in a sandbox). An optional language-model *behavior plan* may enrich a runtime test, but only behind a deterministic firewall: any model output must pass schema validation and span-citation validation against the committing concept before it is admitted, and it can never introduce a test that is not traceable to a committed concept. The stage materializes a `pytest` suite together with a manifest and a concept-to-test map, so that every test carries the concept identifiers and evidence spans that justify it.

F. Stage 5: Nine-Gate Promotion Pipeline

Generated tests are not trusted by default. Stage 5 subjects every test to nine verification gates and admits to the *Stable* grading suite only those that survive all applicable gates; survivors that are sound but not grade-bearing are demoted to a *Shadow* suite (run for observability only), and hard policy violations are *Blocked*. Gates execute sequentially in the fixed order shown in Figure 3: **(1)** enforceability (static AST scan rejecting banned imports, calls, filesystem writes, and syntax

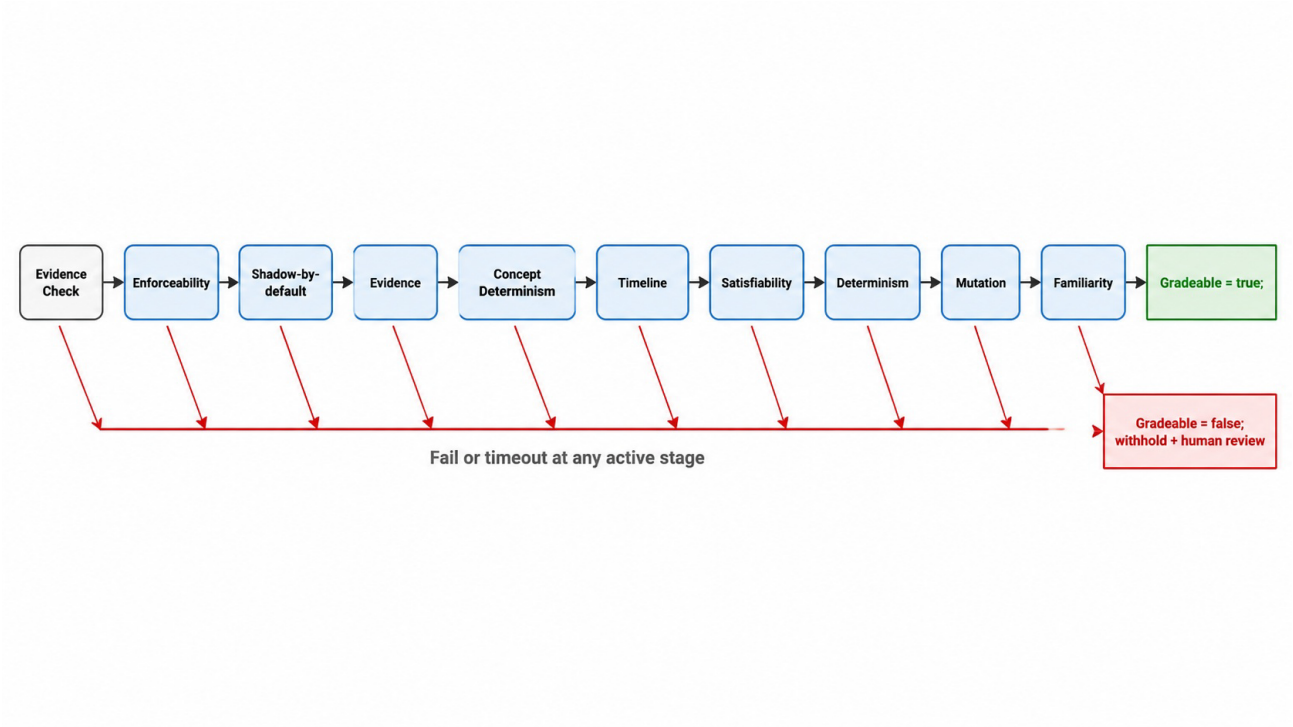

Fig. 3. Fail-closed verification decision flow. A candidate first passes an evidence check, then runs the nine verification gates in order: **(1)** enforceability, **(2)** shadow-by-default, **(3)** evidence, **(4)** concept determinism, **(5)** timeline, **(6)** satisfiability, **(7)** determinism, **(8)** mutation, and **(9)** familiarity. The first three always run; the remaining six are policy-toggleable (enabled by default, a disabled gate passes all tests). Only if every active gate passes does the system emit `gradeable = true`. Any failure or timeout—at the evidence check or any gate—sets `gradeable = false`, withholds the grade, and routes the item to the single human-review sink.

errors; block-only), **(2)** shadow-by-default (metadata-driven demotion), **(3)** evidence (each test must link to at least one valid, non-empty span; block-only), **(4)** concept determinism (suite-level checksum gate that blocks the whole suite if the committed concept artifacts have drifted), **(5)** timeline (a not-yet-taught concept is demoted to Shadow, but a hard-gated concept with unknown instructional week is blocked), **(6)** satisfiability (the test must pass a known-correct witness submission in a hermetic Docker container), **(7)** determinism (identical verdicts across repeated witness runs), **(8)** mutation (kill rate ≥ 0.80 over AST mutants of the witness for applicable `runtime_*` tests), and **(9)** familiarity (a coverage score $\mathit{timeline\_factor} \cdot (0.5 \cdot \mathit{authority} + 0.5 \cdot \mathit{evidence})$, minimized over a test's concepts, must reach 0.70).

The pipeline does not short-circuit: all nine gate reports are always produced (a disabled gate emits a synthetic all-*stable* report), and per-test dispositions are combined *pessimistically*—blocked dominates shadow, which dominates stable, and a stable verdict never re-promotes a test that another gate demoted or blocked. Crucially, gates *fail closed*: any gate that crashes or exceeds its wall-clock timeout blocks every test it touches rather than passing it. A test reaches the Stable suite only when *every* applicable gate independently returns *stable*. Of the nine, seven (enforceability, evidence, timeline, satisfiability, determinism, mutation, and familiarity) are *required-stable*: a test must read *stable* on all seven to be eligible for the Stable suite, while the remaining two govern integrity and metadata-driven demotion. A subsequent deterministic override layer may only demote tests, never promote them, so repair workflows can never bypass a safety gate.

G. Stage 6: Fail-Closed Scoring

Scoring is defined over the *active set* $\mathcal{A}$, which is exactly the Stable suite produced by Stage 5; Shadow and Blocked tests never contribute to a grade. For a submission, the score and its ceiling are

$$\text{Score} = \sum_{i \in \mathcal{A}} s_i, \qquad \text{MaxScore} = \sum_{i \in \mathcal{A}} w_i, \tag{3}$$

where w_i is test i's weight. A passing test scores $s_i = w_i$; a failing test scores $s_i = 0$ unless partial credit is enabled, in which case it earns a deterministic fraction of w_i keyed to its failure mode (timeout, assertion, or output mismatch). The reported percentage is $\text{Score}/\text{MaxScore} \times 100$.

The scoring contract is fail-closed at its core. If the Stable suite is empty, the submission is declared *ungradeable*: the system sets $\mathit{gradeable} = \text{FALSE}$, returns a null percentage with the reason `stable_suite_empty`, and refuses to backfill with Shadow results. A guard raises an error if an empty-Stable run ever produces a numeric score, so the system cannot silently fabricate a grade from unverified evidence. Each grade is accompanied by score components (per-question and per-scope content scores, policy penalties, and rubric components)

that, together with the concept and span provenance carried forward from earlier stages, make every emitted grade auditable end to end.

IV. Experiments

A. Benchmark and corpus

We evaluate Evidence-Centered GradeTrace on the FalconCode/CS110 programming corpus [3], an archived collection of student Python submissions from the 2021 cohort of an introductory programming course. Every result in this paper comes from a *single, fixed* configuration: Formal Concept Analysis (FCA) concept discovery (concept method `lcm`) with a Gemini-3-Flash proposer (`gemini-3-flash-preview`) served via Vertex. We do not tune across methods, models, or courses; the study is a measurement of one system on one corpus, not a comparison.

A point that governs the interpretation of every agreement number below must be stated plainly and early: the *reference scores against which we measure agreement are the FalconCode corpus reference scores produced by the course's own unit tests*—they are not human instructor grades. Agreement with these references therefore bounds how closely the system reproduces a unit-test-based grading target, and any structural property of those tests (most importantly, the absence of partial credit) propagates directly into the error profile we report in Section V. We treat the reference as a measurement target rather than a gold-standard human judgement.

We assess the system along six axes: agreement with the reference scores, stability of the model/prompt configuration, systematic bias, the shape of the error distribution, auditability of each grading decision, and operational efficiency. The full per-criterion benchmark—the metric definitions, the artifacts each metric is read from, and the exact computation for every sub-metric across the accuracy, reliability, quality, cost, and operations groups—is reported in the Appendix (Table VII); the related-work coverage matrix that situates these criteria against prior evaluation practice is Table I. Here we report the measurement outcomes.

V. Results

Evidence-Centered GradeTrace is evaluated here as a research measurement artifact, not a finished production-quality grader. A concentrated RMSE/std-diff tail (Section V-E) still requires generic partial-credit and graphics/runtime repair. Following the denominator boundary of the scoring contract, all score-difference metrics are computed *only* on reportable rows within the corpus-valid denominator (Section V-D); rows discarded for corpus, prompt, source, graphics/runtime, or rubric limitations are accounted for separately and never enter the score-difference population.

A. Metric definitions

Let $\mathrm{diff}_i = s_i^{\mathrm{grader}} - s_i^{\mathrm{ref}}$ be the signed percentage-point difference between the automated score and the corpus reference score on reportable row i. Over the n reportable rows we report

$$\mathrm{bias} = \tfrac{1}{n}\textstyle\sum_i \mathrm{diff}_i, \qquad \mathrm{MAE} = \tfrac{1}{n}\textstyle\sum_i |\mathrm{diff}_i|,$$
$$\mathrm{std\_diff} = \sqrt{\tfrac{1}{n}\textstyle\sum_i (\mathrm{diff}_i - \mathrm{bias})^2}, \quad \mathrm{RMSE} = \sqrt{\tfrac{1}{n}\textstyle\sum_i \mathrm{diff}_i^2}.$$

Only bias is a signed average, so positive and negative differences cancel in it; MAE, RMSE, and $\mathrm{std\_diff}$ do not cancel. A *large error* is a reportable row with $|\mathrm{diff}_i| \geq 25$ points. The std-diff uses the population denominator. These definitions are pinned to `tools/score_error_outlier_audit.py`.

TABLE II
Headline accuracy on the stable, corpus/source-limit-adjusted view ($n = 30{,}234$). Reference scores are FalconCode corpus unit-test scores, not human instructor grades. Source: `FINAL_POST_ADJUDICATION_PROFESSOR_SUMMARY_20260519`.

Metric	Value
Reportable rows (n)	30,234
Corpus-valid denominator	36,038
System reportability (of 36,038)	83.9%
Raw end-to-end yield (of 44,007)	68.70%
Pearson r	0.805
Spearman ρ	0.791
Bias	-2.309
MAE	3.988
RMSE	17.800
Std. diff	17.650
Median $\lvert\mathrm{diff}\rvert$	0.000
P90 $\lvert\mathrm{diff}\rvert$	0.000
P95 $\lvert\mathrm{diff}\rvert$	30.000
Large errors ($\lvert\mathrm{diff}\rvert \geq 25$)	1,653

B. Headline accuracy

On the stable, corpus/source-limit-adjusted view (Table II), GradeTrace attains Pearson $r = 0.805$ and Spearman $\rho = 0.791$ against the corpus reference scores over $n = 30{,}234$ reportable rows, with a small negative bias of -2.309 points. The median and P90 absolute differences are both 0—a majority of reportable rows are graded exactly—while RMSE (17.800) and std-diff (17.650) are an order of magnitude above the MAE (3.988), the hallmark of a tail-dominated error distribution rather than uniform disagreement. Conditional on the 36,038-row corpus-valid denominator, system reportability is 83.9% (Section V-D).

C. Coverage and discard accounting

We deliberately report the denominator at three levels rather than as a single "coverage" number, so that the recoverability of a course subset is never conflated with system behavior (Figure 4, Table III). The raw extraction contains 44,007 rows. Of these, 7,969 rows—spanning 17 problem families—are *corpus/source-limited exclusions*: their prompts refer to material outside the recovered course subset (for example, "refer back to class"), which cannot be grounded in admissible evidence, so they fall outside the evidentiary scope of the method and are not GradeTrace candidates at all. Removing them leaves a *corpus-valid denominator* of 36,038 rows. Within that denominator,

TABLE III
COVERAGE AND DISCARD ACCOUNTING. SOURCE: FINAL_POST_ADJUDICATION_PROFESSOR_SUMMARY_20260519.

Field	Value
Raw denominator	44,007
Corpus/source-limited exclusions (17 families)	7,969
Corpus-valid denominator	36,038
Reportable rows	30,234
Within-corpus-valid withheld	5,804
Reportable family coverage	131 / 131
Corpus observability (36,038/44,007)	81.9%
Corpus-valid system reportability (30,234/36,038)	83.9%
Raw end-to-end yield (30,234/44,007)	68.70%
$44{,}007 = 30{,}234 + 7{,}969 + 5{,}804$	verified

30,234 rows are reportable and 5,804 are withheld—either by eligibility checks (missing executable or scoring artifacts) or by fail-closed verification gates. We count neither withheld group as a grading error, and the per-row gate outcomes that separate the two are retained in the provenance logs. In total 13,773 rows are discarded (7,969 corpus/source-limited + 5,804 within-corpus-valid withheld), so $44{,}007 = 30{,}234 + 13{,}773$.

This decomposition yields three distinct rates rather than one: *corpus observability* $36{,}038/44{,}007 = 81.9\%$ (how much of the raw extraction the course subset can ground); *corpus-valid system reportability* $30{,}234/36{,}038 = 83.9\%$ (how often the system produces a grade once admissible evidence exists); and *raw end-to-end yield* $30{,}234/44{,}007 = 68.70\%$ (their operational composition). The funnel identity

$$44{,}007 \;=\; 30{,}234 \;+\; 7{,}969 \;+\; 5{,}804$$

is verified by the released audit artifacts, and reportable family coverage is complete: $131/131$ corpus-valid families are covered (rate 1.0).

D. Corpus observability and source-limited exclusions

Unlike evaluations that silently collapse missing context, execution failure, and grading failure into a single benchmark denominator, we separate corpus observability from grading reportability. Source-limited rows are not failures of the grader but failures of the corpus to satisfy the evidentiary preconditions for measurement: GradeTrace requires assignment-level source evidence before it will emit a score, so a row whose prompt points outside the recovered course subset is excluded from the corpus-valid denominator and reported separately, not counted as a model abstention or a grading error. This is exactly the distinction the three rates make explicit. A lone coverage number would conflate how much of the archive the course subset can ground (corpus observability, 81.9%) with how often the system grades once evidence exists (corpus-valid system reportability, 83.9%).

Threat to validity. Our agreement results therefore characterize performance *conditional on recoverable course evidence*. They do not imply that GradeTrace can grade arbitrary submissions without assignment context, rubrics, or executable artifacts; the corpus/source-limited exclusions quantify the practical burden of deploying evidence-centered grading on an incomplete historical corpus, and the gap between corpus observability and full coverage is an archival/infrastructure property of the dataset rather than a grading-accuracy failure. Because the exclusions concentrate in 17 problem families, the reportable subset may skew toward better-preserved assignments; a per-family retained-versus-excluded comparison over the available metadata is the right next disclosure and is left to future work, as it requires the full per-row exclusion labels. We emphasize that uneven material completeness across assignments is an archival limitation, not a grading-accuracy failure: the system may grade well within well-documented assignments yet cannot be deployed uniformly across an incompletely preserved archive.

E. Error profile and the residual tail

Although MAE is small (3.988) and both the median and P90 absolute errors are 0.000, the RMSE (17.800) and std-diff (17.650) are an order of magnitude larger, the P95 absolute error is 30.000, and 1,653 rows are large errors (Figure 5). GradeTrace is thus close to unbiased on average yet carries a concentrated high-error tail. That tail is overwhelmingly an *under*-grading phenomenon: among nonzero signed differences in the reportable set, under-scoring errors outnumber over-scoring ones by more than three to one (1,634 versus 501), consistent with the small overall negative bias of -2.309. The error is concentrated rather than diffuse—the top problem families alone hold 623 of the 1,653 large errors, a 37.7% share—which is what we examine in the outlier taxonomy (Section V-G).

F. Diagnostic scoring views

Table IV reports diagnostic views over the scoring and denominator boundaries. These are not all component ablations. Every row marked diagnostic (†) is *not* a promotable scoring configuration; these views exist only to localize error sources. For instance, excluding the partial-credit-labeled families raises r to 0.883 and cuts RMSE to 10.553, but it does so by removing rows and is a counterfactual, not a deployable grader. The two substitution views reach $r = 0.822$ and $r = 0.824$, but the tail-safe best-subset view fails its held-out audit, and the holdout-accepted component policy accepts *zero* replacements. The headline remains the stable post-adjudication view at $r = 0.805$, and post-adjudication produced *zero* score-changing promotions (Section V-H). Full provenance for all views, with verbatim claim-status labels and per-view replaced/excluded row counts, is in the Appendix (Table VI).

G. Outlier taxonomy

The residual tail concentrates in a small number of problem families (Table V, Figure 7). The worst family, `lsn19_100m` ($n = 562$), has RMSE 57.949, MAE 34.541, and bias -31.075: systematic under-scoring where the reference test admits no partial credit. Every one of the top-five families is dominated by all-or-nothing tests with no partial-credit rubric, so a submission that misses a single sub-case can forfeit an entire problem's points even when most of its behavior is correct. This is a

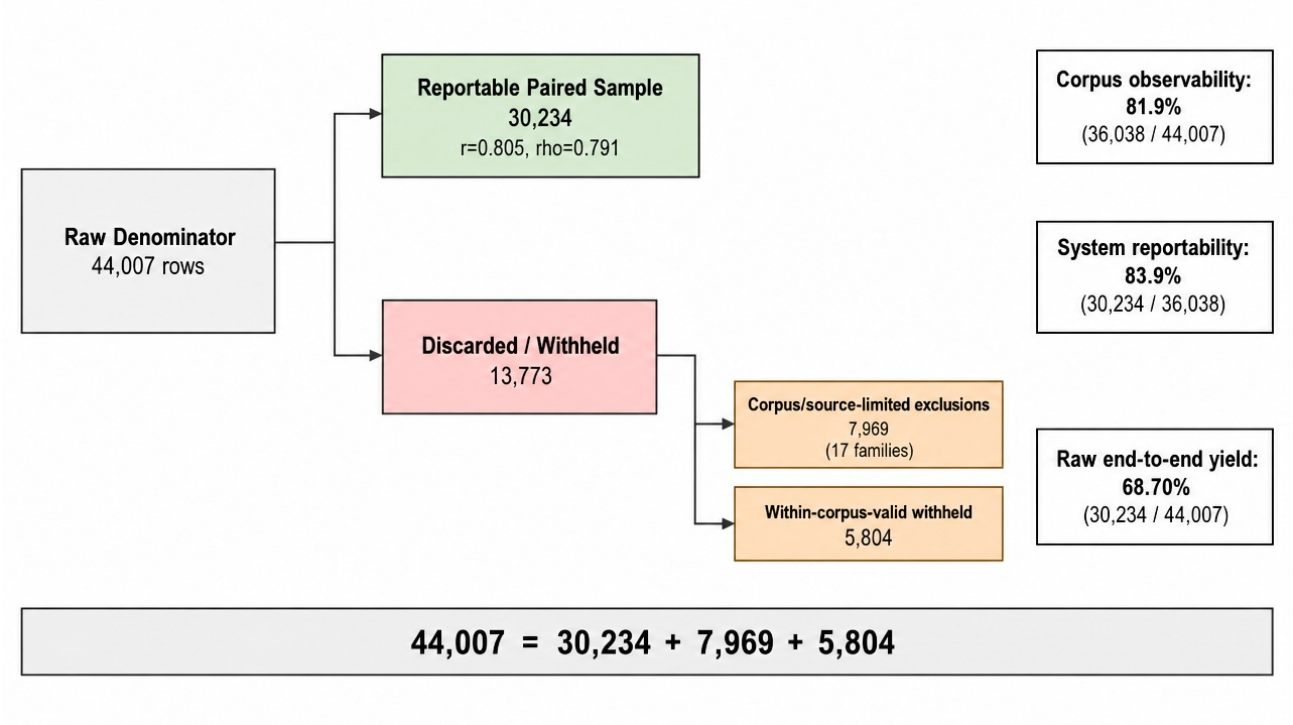

Fig. 4. Denominator flow. The raw 44,007 rows decompose exactly as 30,234 reportable + 7,969 corpus/source-limited exclusions + 5,804 within-corpus-valid withheld (identity verified by the released audit artifacts). Removing the 7,969 corpus/source-limited rows—whose material cannot ground an admissible test—leaves a corpus-valid denominator of 36,038. Three rates summarize the funnel: *corpus observability* 81.9% (how much of the raw extraction the course subset can ground), *system reportability* 83.9% (how often a grade is produced once the evidence exists), and *raw end-to-end yield* 68.70% (their operational combination). The 68.70% is a reportability boundary, not a performance filter.

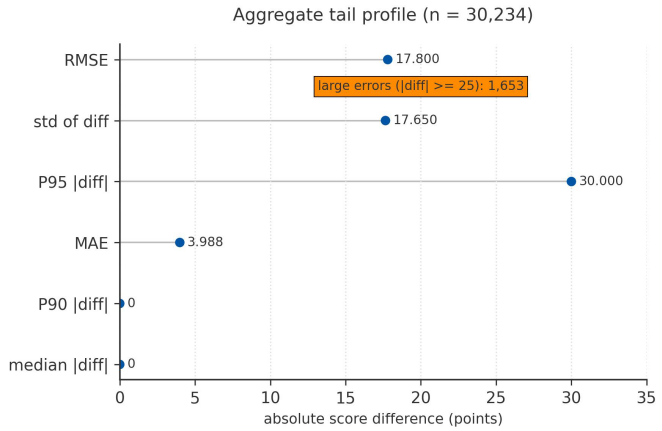

Fig. 5. Aggregate error-tail profile for the headline configuration ($n = 30{,}234$). The plot uses released aggregate statistics, not row-level private records: the median and P90 absolute differences are both 0, while P95 (30), MAE (3.99), std-diff (17.65), and RMSE (17.80) climb sharply. The released aggregates show a center-concentrated profile with a heavy right tail; in total 1,653 reportable rows are large errors ($|\text{diff}| \geq 25$).

TABLE IV
DIAGNOSTIC SCORING VIEWS. † = DIAGNOSTIC ONLY (NOT A PROMOTABLE SCORING CONFIGURATION); THE HEADLINE IS THE STABLE POST-ADJUDICATION CORPUS/SOURCE-LIMIT-ADJUSTED VIEW. "LARGE ERR." COUNTS $|\text{diff}| \geq 25$. SOURCE: PAPER_ABLATION_TABLE_20260518 VIA FINAL_POST_ADJUDICATION_PROFESSOR_SUMMARY_20260519.

View	Status	n	r	RMSE	Lg. err.
Repair-adjusted baseline	baseline	38,203	0.771	19.326	2,361
Stable headline	**headline**	30,234	0.805	17.800	1,653
Broad equal-component sub.	†	38,203	0.822	16.251	1,974
Tail-safe best-subset	† (h/o fail)	38,203	0.824	16.147	1,809
Holdout-accepted policy	bdry (0 acc.)	38,203	0.796	17.820	2,075
Top-tail exclusion	†	33,706	0.853	14.666	1,298
Excl. graphics/PythonGraph	†	35,749	0.797	18.193	2,008
Excl. partial-credit-labeled	†	22,051	0.883	10.553	448

property of the *reference rubric encoding*, not of the concept-discovery or scoring machinery, which is precisely why the median and P90 absolute errors remain 0 while RMSE stays high. It also bounds the remediation path: adding generic partial-credit decomposition for the few high-mass families, rather than a broad model change, is the direction the diagnostic views indicate would most improve the headline.

H. Promotion safety and artifact verification

Post-adjudication, no candidate test was promoted to the stable suite in a way that would change scores. Of 12 ready action rows there were 0 stable promotions: 4 rows failed the label-safety gate, 2 require repair before they can pass a gate, 4 executed with no score change, and 2 carry non-promotable candidate labels. The released artifact index is fully verified—140/140 artifacts present and hash-matched, with zero missing and zero mismatched—and the released audit checks pass the

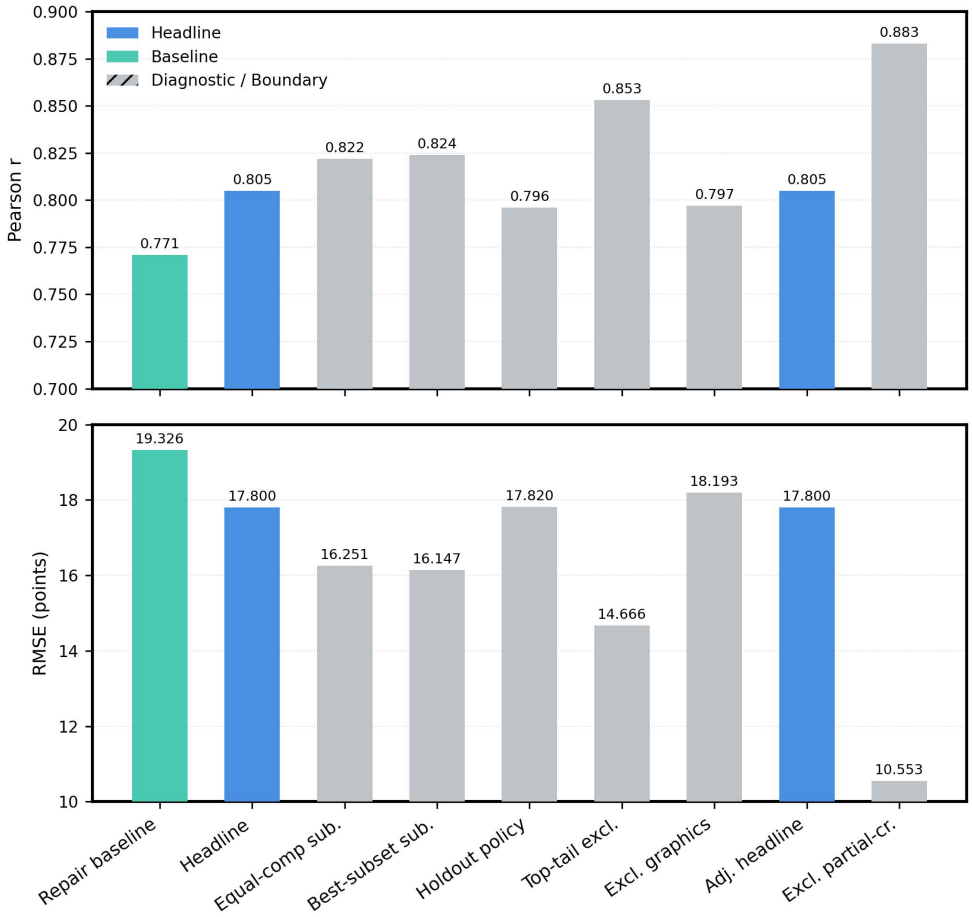

Fig. 6. Diagnostic scoring and denominator views from Table IV. Top: Pearson r against the reference scores; bottom: RMSE. Solid bars are the reportable headline and the stable baseline; hatched bars are diagnostic-only or boundary views we do not promote—several raise apparent agreement only by excluding hard families. The headline is the conservative, corpus/source-limit-adjusted configuration used throughout.

TABLE V
TOP-FIVE HIGHEST-RMSE BOTTLENECK FAMILIES. ALL FIVE ARE DOMINATED BY ALL-OR-NOTHING TESTS WITH NO PARTIAL-CREDIT RUBRIC. SOURCE: `FINAL_POST_ADJUDICATION_PROFESSOR_SUMMARY_20260519` ("TOP BOTTLENECKS BY RMSE").

Family	n	RMSE	MAE	Bias
`lsn19_100m`	562	57.949	34.541	−31.075
`lsn4_madlib`	368	50.271	25.272	−24.185
`lsn14_counting`	213	36.418	24.648	−20.423
`lsn19_movies`	111	36.021	14.387	−13.775
`lsn14_coordinates`	139	35.932	13.612	−13.468

requirements for a reproducible single-corpus measurement study with explicit limitations.

VI. DISCUSSION

The headline numbers (Pearson $r = 0.805$, Spearman $\rho = 0.791$, median $|\text{diff}| = 0$) are not the contribution of this paper. They describe one configuration on one corpus and would change with the proposer model, the course, or the rubric encoding. What is meant to transfer is the *measurement discipline* that produced them: an explicit corpus-valid denominator, a fully reconciled discard ledger ($44{,}007 = 30{,}234 + 7{,}969 + 5{,}804$), and diagnostic scoring views that are never silently folded into the headline. A grade that the system reports is a grade it can defend through the concept → test → gate chain; a row it cannot defend is withheld and counted, not guessed.

Where the error concentrates. The residual error is concentrated, not diffuse (Figure 7). Among the 30,234 reportable rows there are 1,653 large errors ($|\text{diff}| \geq 25$ points), and the few highest-mass problem families hold 623 of them, a 37.7% share of the entire large-error tail. The tail is also strongly directional: among nonzero signed differences in the reportable set, under-scoring errors outnumber over-scoring ones by more than three to one (1,634 versus 501), consistent with the small overall negative bias (−2.309). The worst families by RMSE (e.g. `lsn19_100m`, $n = 562$, RMSE 57.949; `lsn4_madlib`, $n = 368$, RMSE 50.271;

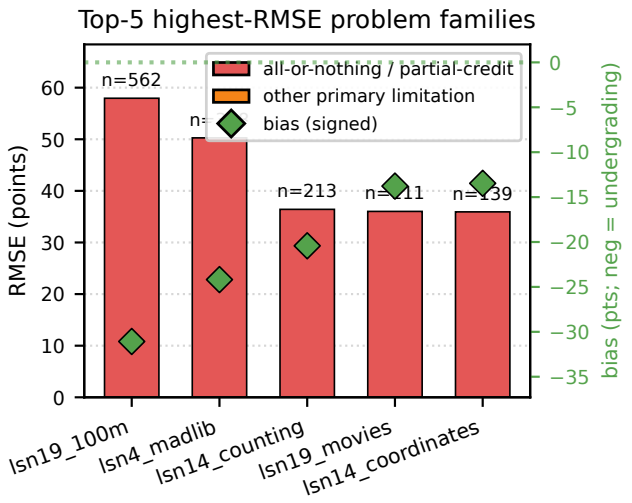

Fig. 7. The five highest-RMSE problem families (cf. Table V), colored by dominant failure mode. Diamonds overlay the signed bias—all negative, i.e. systematic under-grading—and n is the family size. Tail error concentrates in families whose reference unit tests admit no partial credit.

`lsn14_counting`, $n = 213$, RMSE 36.418) share one mechanism: their tests are all-or-nothing and award no partial credit, so a submission that misses a single sub-case can forfeit an entire problem even when most of its behavior is correct. This is a property of the *rubric encoding*, not of the concept-discovery or scoring machinery, which is precisely why the median and P90 absolute errors stay at 0 while RMSE remains high (17.800): most submissions are graded exactly, and the variance lives in a small, identifiable set of binary-scored problems.

The practical reading is that remediation is bounded. Because the mass sits in a handful of families with a shared, named cause, a generic partial-credit decomposition targeted at those families is a more promising lever than any broad change to the model or the discovery procedure. The diagnostic exclusion view that excludes partial-credit-labeled families reaches $r = 0.883$ with RMSE 10.553 and only 448 large errors, which upper-bounds the achievable gain—but it is a counterfactual view, not a deployable configuration, and we treat it as an attribution of error rather than a result.

VII. Limitations

We state the limits of this study plainly, because the measurement framing only holds if its boundaries are explicit.

Single corpus, single configuration. Every number reported here comes from one corpus (FalconCode/CS110, 2021 cohort [3]) and one configuration: Formal Concept Analysis (FCA) concept discovery (`concept_method: lcm`) with a single proposer model (`gemini-3-flash-preview` served via Vertex). Although the corpus-specific logic is kept separate from the generic harness, we do not evaluate generalization to other courses, languages, or proposer models, and a controlled study across cohorts and models is left to future work.

No external baseline. We report *no* comparison against an external grader—neither a vanilla LLM-as-judge nor a conventional test-suite autograder. The agreement figures are therefore *absolute measurements* of this system on a fixed configuration, not evidence of superiority over a simpler approach. $r = 0.805$ should be read as "what this pipeline achieves here," and establishing external baselines is the single most valuable next step.

Reference labels are corpus scores, not human grades. The grading target is the FalconCode corpus reference (unit-test) score, read from the corpus and treated as ground truth. These are not human instructor grades; their provenance and scale bound the ceiling of any agreement metric and should be interpreted as a reference, not a gold standard. This is the familiar validity gap between a measurable proxy and the construct an assessment is meant to capture [37], [38], [39]: a high correlation with corpus scores certifies agreement with the proxy, not construct validity of the underlying grade.

No uncertainty quantification yet. Reportable rows are individual submissions and are not statistically independent—many students solve the same problem, so a single family can dominate a metric. We do not yet report confidence intervals bootstrapped by problem family, nor a risk–coverage (deferral) curve establishing that the fail-closed gate *preferentially* withholds rows it would otherwise grade incorrectly. Both are low-compute analyses we prioritize for the next revision; their absence means the point estimates should be read with the clustering in mind.

Residual partial-credit tail. Evidence-Centered GradeTrace is explicitly not a finished production grader. The dominant residual failure mode is the all-or-nothing partial-credit tail described above, reported here as a diagnostic exclusion rather than a completed intervention. Running that intervention and measuring before/after RMSE, MAE, P95, and rank correlation on the high-tail families is the highest-value empirical next step alongside the external baselines.

VIII. Ethics and Educational Deployment

GradeTrace is fail-closed by construction. When evidence is insufficient or a verification gate fails, it withholds an automated grade (`gradeable=false`) and routes the case to human review rather than guessing; the 5,804 within-corpus rows withheld in our run are exactly such deferrals. This design trades coverage for safety, reducing the risk of unfair automated penalties, and a reported grade should never be acted upon unless the gates pass and `gradeable=true` is emitted. Because every score traces to concrete concept, test, and gate evidence, decisions are auditable and directly support student appeals and dispute resolution. No student submissions or private course data are redistributed: the corpus is externally sourced and referenced by identifier only. Given the partial-credit tail and the absence of external baselines, the system should be deployed strictly as instructor-in-the-loop decision support, not as an autonomous grader.

IX. Reproducibility

All reported numbers come from the single frozen post-adjudication snapshot named in Appendix A-B, not from ad hoc recomputation. The released artifact index is fully verified ($140/140$ artifacts present and hash-matched, 0 missing), and the funnel identity $44{,}007 = 30{,}234 + 7{,}969 + 5{,}804$ reconciles exactly. Metric definitions are pinned to the score-error outlier audit script, so Pearson, Spearman, bias, MAE, RMSE, and the percentile/large-error counts are reproduced from one source of truth. The reported run uses FCA concept discovery (`concept_method: lcm`) with the `gemini-3-flash-preview` proposer served via Vertex, on the FalconCode/CS110 2021 cohort [3]; the proposer is configurable and provider-agnostic. The grading code—concept discovery, the verification gates, and the fail-closed scorer—lives under `src/nsmc/` with deterministic gates and checksummed suite manifests, and the run contract is documented alongside the snapshot.

X. Conclusion

We presented Evidence-Centered GradeTrace, a neuro-symbolic, evidence-centered, fail-closed autograding methodology whose transferable contribution is an accounting discipline rather than a correlation. An explicit corpus-valid denominator, a reconciled discard ledger, and diagnostic scoring views keep the reported agreement explicit about what it does and does not certify. Instantiated with FCA concept discovery and a Gemini-3-Flash proposer on FalconCode/CS110, the system reaches Pearson $r = 0.805$ and Spearman $\rho = 0.791$ with near-zero median error on $30{,}234$ reportable rows, while transparently accounting for $13{,}773$ limitation discards and localizing its residual error to a small set of all-or-nothing problem families. The next steps follow directly from that accounting: a targeted partial-credit repair for the high-mass tail, comparison against external grading baselines, and family-clustered confidence intervals with a risk–coverage curve. We argue that this accounting, not the headline correlation, is what should transfer to the next evaluation of an LLM-assisted grader.

References

[1] V. Sahu, G. R. Gupta, R. Borikar, and N. Mane, "Autograder+: A multi-faceted ai framework for rich pedagogical feedback in programming education," 2025. [Online]. Available: https://arxiv.org/abs/2510.26402

[2] U. Alkafaween, I. Albluwi, and P. Denny, "Automating autograding: Large language models as test suite generators for introductory programming," 2024, later published in Journal of Computer Assisted Learning, 41:e13100 (2025). [Online]. Available: https://arxiv.org/abs/2411.09261

[3] J. de Freitas, B. Gruen, D. Kauchak, and A. J. Ko, "FalconCode: A multiyear dataset of python code samples from an introductory computer science course," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, ser. SIGCSE 2023. New York, NY, USA: Association for Computing Machinery, 2023, pp. 938–944. [Online]. Available: https://huggingface.co/datasets/koutch/falcon_code

[4] B. Char and T. Hewett, "Work in progress: A first-year common course on computational problem solving and programming," in *2014 ASEE Annual Conference & Exposition Proceedings*. Indianapolis, Indiana: ASEE Conferences, 2014, pp. 24.1383.1–24.1383.17. [Online]. Available: https://doi.org/10.18260/1-2--23316

[5] J. Doughty, Z. Wan, A. Bompelli, J. Qayum, T. Wang, J. Zhang, Y. Zheng, A. Doyle, P. Sridhar, A. Agarwal, C. Bogart, E. Keylor, C. Kultur, J. Savelka, and M. Sakr, "A comparative study of AI-generated (GPT-4) and human-crafted MCQs in programming education," in *Proceedings of the 26th Australasian Computing Education Conference (ACE 2024)*. New York, NY, USA: Association for Computing Machinery, 2024, pp. 114–123.

[6] W. Xie, J. Niu, C. J. Xue, and N. Guan, "Grade like a human: Rethinking automated assessment with large language models," 2024. [Online]. Available: https://arxiv.org/abs/2405.19694

[7] Y. Chu, H. Li, K. Yang, H. Shomer, H. Liu, Y. Copur-Gencturk, and J. Tang, "A LLM-powered automatic grading framework with human-level guidelines optimization," in *Proceedings of the 18th International Conference on Educational Data Mining (EDM 2025)*, 2025.

[8] K. Qian, Y. Cheng, R. Guan, W. Dai, F. Jin, K. Yang, S. Nawaz, Z. Swiecki, G. Chen, L. Yan, and D. Gašević, "Dean of LLM tutors: Exploring comprehensive and automated evaluation of LLM-generated educational feedback via LLM feedback evaluators," 2025. [Online]. Available: https://arxiv.org/abs/2508.05952

[9] E. M. AlGhamdi, Y. Li, D. Gašević, and G. Chen, "Leveraging prompt-based LLMs for automated scoring and feedback generation in higher education," *Computers & Education*, vol. 243, p. 105511, 2026.

[10] K. Wataoka, T. Takahashi, and R. Ri, "Self-preference bias in LLM-as-a-judge," 2024, accepted at NeurIPS 2024 Safe Generative AI Workshop. [Online]. Available: https://arxiv.org/abs/2410.21819

[11] R. Stureborg, D. Alikaniotis, and Y. Suhara, "Large language models are inconsistent and biased evaluators," 2024. [Online]. Available: https://arxiv.org/abs/2405.01724

[12] P. Wang, L. Li, L. Chen, Z. Cai, D. Zhu, B. Lin, Y. Cao, Q. Liu, T. Liu, and Z. Sui, "Large language models are not fair evaluators," 2023. [Online]. Available: https://arxiv.org/abs/2305.17926

[13] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena," in *Advances in Neural Information Processing Systems 36 (NeurIPS 2023) Datasets and Benchmarks Track*, 2023. [Online]. Available: https://arxiv.org/abs/2306.05685

[14] L. Shi, C. Ma, W. Liang, X. Diao, W. Ma, and S. Vosoughi, "Judging the judges: A systematic study of position bias in LLM-as-a-judge," in *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2025)*, 2024. [Online]. Available: https://arxiv.org/abs/2406.07791

[15] A. Panickssery, S. R. Bowman, and S. Feng, "LLM evaluators recognize and favor their own generations," in *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024. [Online]. Available: https://arxiv.org/abs/2404.13076

[16] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-Eval: NLG evaluation using GPT-4 with better human alignment," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*. Singapore: Association for Computational Linguistics, 2023, pp. 2511–2522. [Online]. Available: https://aclanthology.org/2023.emnlp-main.153/

[17] Y. Chu, H. Li, K. Yang, Y. Copur-Gencturk, J. Krajcik, N. Shin, and J. Tang, "Confusion-aware rubric optimization for LLM-based automated grading," 2026. [Online]. Available: https://arxiv.org/abs/2603.00451

[18] Y. Hong, H. Yao, B. Shen, W. Xu, H. Wei, and Y. Dong, "RULERS: Locked rubrics and evidence-anchored scoring for robust LLM evaluation," 2026. [Online]. Available: https://arxiv.org/abs/2601.08654v1

[19] L. Xie, S. Huang, Z. Zhang, A. Zou, Y. Zhai, D. Ren, K. Zhang, H. Hu, B. Liu, H. Chen, Z. Liu, and B. Ding, "Auto-rubric: Learning from implicit weights to explicit rubrics for reward modeling," 2025. [Online]. Available: https://arxiv.org/abs/2510.17314

[20] A. Pathak, R. Gandhi, V. Uttam, A. Ramamoorthy, P. Ghosh, A. R. Jindal, S. Verma, A. Mittal, A. Ased, C. Khatri, Y. Nakka, Devansh, J. S. Challa, and D. Kumar, "Rubric is all you need: Improving LLM-based code evaluation with question-specific rubrics," in *Proceedings of the 2025 ACM Conference on International Computing Education Research (ICER '25), Volume 1*. New York, NY, USA: Association for Computing Machinery, 2025.

[21] E.-Q. Tseng, P.-C. Huang, C. Hsu, P.-Y. Wu, C.-T. Ku, and Y. Kang, "CodEv: An automated grading framework leveraging large language models for consistent and constructive feedback," in *2024 IEEE International Conference on Big Data (BigData)*. IEEE, 2024, pp. 5442–5449.

[22] N. Farzi and L. Dietz, "Pencils down! automatic rubric-based evaluation of retrieve/generate systems," in *Proceedings of the 2024 ACM SIGIR International Conference on Theory of Information Retrieval*, ser. ICTIR '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 175–184. [Online]. Available: https://doi.org/10.1145/3664190.3672511

[23] L. Dietz, "A workbench for autograding retrieve/generate systems," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*. Association for Computing Machinery, 2024. [Online]. Available: https://arxiv.org/abs/2405.13177

[24] D. Armfield, E. Chen, A. Omonkulov, X. Tang, J. Lin, E. Thiessen, and K. R. Koedinger, "Avalon: A human-in-the-loop LLM grading system with instructor calibration and student self-assessment," in *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky (AIED 2025 Companion)*. Springer, 2025, pp. 111–118.

[25] M. Messer, N. C. C. Brown, M. Kölling, and M. Shi, "Automated grading and feedback tools for programming education: A systematic review," *ACM Transactions on Computing Education*, vol. 24, no. 1, pp. 10:1–10:43, 2024. [Online]. Available: https://doi.org/10.1145/3636515

[26] M. Balog, A. L. Gaunt, M. Brockschmidt, S. Nowozin, and D. Tarlow, "DeepCoder: Learning to write programs," 2016, published at ICLR 2017. [Online]. Available: https://arxiv.org/abs/1611.01989

[27] M. Allamanis, M. Brockschmidt, and M. Khademi, "Learning to represent programs with graphs," in *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*, 2018. [Online]. Available: https://arxiv.org/abs/1711.00740

[28] X. Chen, C. Liu, and D. Song, "Execution-guided neural program synthesis," in *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019. [Online]. Available: https://openreview.net/forum?id=H1gfOiAqYm

[29] X. Si, H. Dai, M. Raghothaman, M. Naik, and L. Song, "Learning loop invariants for program verification," in *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*. Curran Associates, Inc., 2018, pp. 7762–7773. [Online]. Available: https://papers.nips.cc/paper/2018/hash/65b1e92c585fd4c2159d5f33b5030ff2-Abstract.html

[30] American Educational Research Association, American Psychological Association, and National Council on Measurement in Education, *Standards for Educational and Psychological Testing*. Washington, DC: American Educational Research Association, 2014.

[31] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, 2nd ed. Cham, Switzerland: Springer, 2024.

[32] S. Andrews, "In-close2, a high performance formal concept miner," in *Conceptual Structures for Discovering Knowledge (ICCS 2011)*, ser. Lecture Notes in Computer Science, vol. 6828. Springer, 2011, pp. 50–62.

[33] A. Khaund, A. M. Sharma, A. Tiwari, S. Garg, and S. Kailasam, "RD-FCA: A resilient distributed framework for formal concept analysis," *Journal of Parallel and Distributed Computing*, vol. 179, p. 104710, 2023.

[34] R. K. Chunduri and A. K. Cherukuri, "Scalable formal concept analysis algorithm for large datasets using Spark," *Journal of Ambient Intelligence and Humanized Computing*, 2018. [Online]. Available: https://arxiv.org/abs/1807.02258

[35] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal, "Computing iceberg concept lattices with TITANIC," *Data & Knowledge Engineering*, vol. 42, no. 2, pp. 189–222, 2002.

[36] D.-Y. Niu and J.-S. Mi, "Three-way concept lattice based on boolean formal context," *International Journal of Approximate Reasoning*, vol. 175, p. 109286, 2024.

[37] S. Sinharay, R. E. Bennett, M. Kane, and J. R. Sparks, "Validation for personalized assessments: A threats-to-validity approach," *Journal of Educational Measurement*, vol. 62, no. 2, pp. 282–310, 2025.

[38] P.-B. Degen, "Revisiting generalizability theory in the age of artificial intelligence: Implications for empirical educational research," *Computers and Education Open*, vol. 9, p. 100278, 2025. [Online]. Available: https://doi.org/10.1016/j.caeo.2025.100278

[39] M. van Haastrecht, L. de Groot, M. Jongbloed-Pereboom, F. Buytenhuijs, and J. Kruis, "Artificial intelligence for educational measurement: Where is the value for education?" *Frontiers in Education*, vol. 10, p. 1677255, 2025.

TABLE VI
ALL NINE DIAGNOSTIC VIEWS WITH VERBATIM CLAIM STATUS AND THE ROWS EACH REPLACES OR EXCLUDES RELATIVE TO THE 38,203-ROW STABLE BASELINE. "LG. ERR." COUNTS $|\text{diff}| \geq 25$. SOURCE: `FINAL_POST_ADJUDICATION_PROFESSOR_SUMMARY_20260519`.

View	Claim status (verbatim)	n	R/E	r	RMSE	Lg. err.
Repair-adjusted baseline before official overrides	stable baseline	38,203	0	0.771	19.326	2,361
Stable official overrides with corpus/source-limit denom.	stable headline	30,234	483	0.805	17.800	1,653
Post-adjudication corpus/source-limit-adjusted headline	stable headline	30,234	7,969	0.805	17.800	1,653
Broad equal-component substitution	diagnostic only; do not promote	38,203	2,693	0.822	16.251	1,974
Tail-safe best-subset substitution	diagnostic only; held-out audit failed	38,203	2,540	0.824	16.147	1,809
Holdout-accepted component policy	stable boundary check; zero accepted	38,203	0	0.796	17.820	2,075
Top-tail exclusion diagnostic	diagnostic only; concentration evidence	33,706	4,497	0.853	14.666	1,298
Excluding graphics/PythonGraph-limited families	diagnostic only	35,749	2,454	0.797	18.193	2,008
Excluding partial-credit-labeled families	diagnostic only	22,051	16,152	0.883	10.553	448

APPENDIX A
FULL DIAGNOSTIC-VIEW PROVENANCE

Table VI lists all nine diagnostic views from the canonical post-adjudication summary, each with its *verbatim* claim-status label and the number of rows it replaces or excludes. The replaced/excluded counts are measured against the 38,203-row stable baseline (the full repair-adjusted reportable artifact before the corpus/source-limit denominator rewrite), *not* against the 30,234-row headline. Only the rows labeled *stable baseline* ($n = 38{,}203$) and *stable headline* ($n = 30{,}234$) are scoring configurations; every row labeled *diagnostic only* is a counterfactual reported solely to attribute error and must not be read as a deployable result. The two stable-headline rows are the same configuration described two ways: the override-accounting view (replacing 483 override rows) and the corpus/source-limit denominator view (excluding the 7,969 corpus/source-limited rows from the baseline); both report the identical headline metrics.

A. Metric definitions and the denominator boundary

For reportable row i, let $\text{diff}_i = s_i^{\text{grader}} - s_i^{\text{ref}}$ be the signed percentage-point difference between the automated score and the FalconCode corpus reference (unit-test) score. Over the n reportable rows we report

$$\text{bias} = \tfrac{1}{n}\textstyle\sum_i \text{diff}_i, \qquad \text{MAE} = \tfrac{1}{n}\textstyle\sum_i |\text{diff}_i|,$$
$$\text{std\_diff} = \sqrt{\tfrac{1}{n}\textstyle\sum_i (\text{diff}_i - \text{bias})^2}, \quad \text{RMSE} = \sqrt{\tfrac{1}{n}\textstyle\sum_i \text{diff}_i^2}.$$

Only bias cancels signed errors; MAE, RMSE, and std_diff do not. A *large error* is a reportable row with $|\text{diff}_i| \geq 25$; std_diff uses the population denominator. Metrics are computed *only* over reportable corpus-valid rows; corpus, prompt, source, graphics/runtime, and rubric-limited rows are counted

separately and excluded from the score-difference population. The denominator identity

$$44{,}007 \;=\; 30{,}234 \;+\; 7{,}969 \;+\; 5{,}804$$

(raw = reportable + corpus/source-limited + within-corpus-valid withheld) holds exactly. These definitions are pinned to `tools/score_error_outlier_audit.py`.

B. Reproducibility and artifact verification

The canonical results snapshot is the tracked `full_falcon_post_adjudication_20260519/` directory under `docs/research/results/`; the underlying run JSONs are runtime artifacts kept out of version control by repository policy. The released artifact index is fully verified: $140/140$ artifacts present and hash-matched, with zero missing and zero mismatched. The released audit checks pass, and post-adjudication produced 0 score-changing promotions out of 12 ready action rows (4 failed the label-safety gate, 2 require repair, 4 executed with no score change, and 2 carry non-promotable candidate labels). The reported run uses the Formal Concept Analysis (FCA) concept method (`concept_method: lcm`) with the `gemini-3-flash-preview` proposer served via Vertex, on the FalconCode/CS110 2021 cohort [3].

C. Full benchmark criteria

Table VII details the complete per-criterion benchmark across the accuracy, reliability, quality, cost, and operations groups, giving for each sub-metric its definition and the exact way NSMC computes it from the released artifacts.

TABLE VII

EVIDENCE-CENTERED GRADETRACE BENCHMARK CRITERIA (PART 1 OF 2): ACCURACY AND RELIABILITY GROUPS, AS IMPLEMENTED IN THE REPOSITORY. METRIC NAMES MATCH THE EMITTED AUDIT ARTIFACTS. SEE TABLE VIII FOR THE QUALITY, COST, AND OPERATIONS GROUPS.

Group	Category	Metric	How NSMC computes it
Accuracy	Score alignment	`pearson_r`	Pearson correlation between paired automated and reference scores in `audit/score_comparison.csv`.
		`spearman_rho`	Spearman rank correlation over the same paired scores.
		`mae`	Mean absolute difference between paired automated and reference scores, over submission-level pairs.
		`rmse`	Root mean squared difference between paired automated and reference scores.
		`mean_bias`	Mean signed difference, defined per pair as `auto_score - reference_score`.
		`sd_diff`	Population standard deviation of the signed score differences.
	Grade-boundary accuracy	`false_upgrade_rate`	Fraction of pairs with $\mathrm{bin}(\texttt{auto}) > \mathrm{bin}(\texttt{reference})$, where bin is the configured grade-bin mapping.
		`false_downgrade_rate`	Fraction of pairs with $\mathrm{bin}(\texttt{auto}) < \mathrm{bin}(\texttt{reference})$.
		`exact_rate`	Fraction of pairs with $\mathrm{bin}(\texttt{auto}) = \mathrm{bin}(\texttt{reference})$.
	Borderline accuracy	`borderline_count`	Count of pairs whose reference score lies within $\pm\delta$ of an interior grade boundary.
		`borderline_error_rate`	Fraction of borderline pairs whose automated grade bin differs from the reference bin.
Reliab.	Phantom failure rate	`phantom_failure_rate`	Fraction of presumed-correct submissions that execute a test and fail it.
	Out-of-spec detection	`precision`	Precision on labeled synthetic out-of-spec (OOS) cases (empty repo, non-Python source, or missing required structure).
		`recall`	Recall on the same labeled OOS cases.
		`F1`	Harmonic mean of OOS precision and recall.
	Evidence / scope integrity	`fabricated_risk`	Indicator $= 1$ if evidence is missing, quotes are empty, or `quote_hash` fails SHA-256 verification.
		`relational_distortion_risk`	Indicator $= 1$ if a required qualifier is in the mapped concept but absent from test metadata.
		`scope_inflation_risk`	Indicator $= 1$ if a required qualifier is present but `scope_signature` is missing.
	DEF/REQ/BAN fidelity	`count(exp,act)`	Confusion-matrix counts comparing expected enforcement (from concept kind) with actual enforcement (from test source).
		`enforcement_violation`	Per-test event where expected and actual enforcement differ.
	Qualifier fidelity	`correct`	Per-test indicator that actual enforcement matches expected enforcement.
		`conditional_accuracy`	Fraction of qualified tests with `correct`$= 1$.
		`unconditional_accuracy`	Fraction of unqualified tests with `correct`$= 1$.

TABLE VIII
EVIDENCE-CENTERED GRADETRACE BENCHMARK CRITERIA (PART 2 OF 2): QUALITY, COST, AND OPERATIONS GROUPS. CONTINUATION OF TABLE VII.

Group	Category	Metric	How NSMC computes it
Quality	Test diversity	count	Raw test counts by test_family, concept_kind, matcher_type, relevance_bucket, and policy_channel.
		rate	Within-dimension proportion, count/total for each diversity dimension.
	Test discriminability	discriminability	How well a test separates stronger from weaker submissions, as the gap between witness-good and witness-bad pass rates.
		corr_with_reference	Pearson correlation between the test's binary pass/fail vector and corpus reference scores. Legacy artifacts may emit this as corr_with_instructor.
	Redundancy	same_concept_id_redundancy_rate	Duplicate concept-ID coverage beyond the first occurrence, over total test count.
		same_concept_set_redundancy_rate	Duplicate concept-set coverage beyond the first occurrence, over total test count.
	Concept precision / noise	low_relevance_rate	Fraction of tests tagged with low relevance.
		missing_evidence_rate	Fraction of tests with an empty evidence list.
		concept_kind_other_rate	Fraction of tests whose concept kind resolves to OTHER or empty.
Cost	Run cost	cost_token_units	Total token count read from each run_log_*.json.
		cost_usd	Optional estimate from user-supplied config/pricing.yaml; when that file is absent, the harness leaves this field empty rather than failing.
		pearson_r	Pearson correlation reported in audit/bench/accuracy_summary.json, joined to cost rows when cost logs exist.
		mae	Run-level quality metric copied into cost_vs_accuracy.csv.
		mean_bias	Run-level quality metric copied into cost_vs_accuracy.csv.
	Accuracy per cost	delta_r_per_cost	For adjacent runs sorted by cost, marginal change in Pearson correlation per unit cost.
		delta_mae_per_cost	For adjacent runs sorted by cost, marginal change in MAE per unit cost.
Ops	Human-review trigger rate	trigger_rate	Fraction of registered tests that trigger human-review dispositions.