# The Signal-Coverage Matrix: Stratifying Type and Semantic Errors in Statement Autoformalization

**Chengxiao Dai** [1] **Zhaokun Yan** [2] **Zhanhui Lin** [1]

## Abstract

Headline type-correctness (TC%) of LLM autoformalization has climbed from ∼53% to ∼76% in two years, yet this scalar conceals which errors each method resolves. We propose a signal-coverage matrix that crosses the Lean elaborator (pass/fail) with a semantic-equivalence judgment (equivalent/not), sorting every output into one of four cells: true success (TS), type-only (TO), semantic-only (SO), or both fail (BF). On ProofNet$^{\#}$ and MiniF2F-test with DeepSeek V4-Pro across Vanilla, Lean-Retry, Sample-Filter, and Stratified Autoformalization (SAF): (1) the $+34$ to $+36$ TS gain across the three elab-feedback methods is ∼64% type-stratum recovery, with SO flat on net (87.5% of original semantic errors rescued, 8 newly created). (2) The TO→TS rate is $23/61$ for each method (Wilson 95% CI [26.6%, 50.3%]), and this stratum-level recovery rate predicts $\Delta$TS on held-out methods to within $2/186$ and renders $\Delta TC$ linear in the Vanilla elab-fail rate across six (model, dataset) cells ($R^2$=0.96). (3) The two judges disagree by 26 to 37 pp on elab-feedback outputs (vs. 7 pp on Vanilla), with 30 to 56% of symbolic-judge false negatives traceable to elaborator-forced rewrites. The persistent residual reduces to two gold-formalization errors. TC% gains should be credited by which cell moved, not by the scalar alone.

## 1. Introduction

Statement-level autoformalization maps a natural-language theorem to a Lean 4 signature with a `sorry`-placeholder body. Headline type-correctness (TC%) on ProofNet-class benchmarks (Azerbayev et al., 2023; Poiroux et al., 2024) has climbed from ∼53% (GPT-4o (OpenAI, 2024)) to ∼76% (DeepSeek V4-Pro (DeepSeek-AI, 2026)) in two years, driven by methods that refine against the Lean elaborator. Because TC% does not certify denotational correctness, semantic-faithfulness (SF%) estimators are reported alongside it: back-translation (Zhou et al., 2024) and generalized tree edit distance (GTED, Liu et al., 2025b).

Headline metrics conceal mechanism. A $+15$ pp TC% gain hides which error stratum is repaired and which is not, and the choice of SF% judge can shift the reported value by $\geq 25$ pp on elab-feedback methods (§5.5). No published account decomposes TC% gains by stratum and method. We provide such an account.

Existing methods cluster along three axes. **Single-oracle refinement against the elaborator** (Wu et al., 2022; Poiroux et al., 2024; Guo et al., 2025; Lu et al., 2024) consults only the elaborator (or a close proxy) during refinement, and the class has plateaued at ∼75% TC% on ProofNet$^{\#}$ regardless of topology. **Semantic-faithfulness judging** comprises LLM back-translation (Zhou et al., 2024) and symbolic GTED via the Lean LSP (Liu et al., 2025b), used as standalone SF% estimators but never calibrated against each other at scale. **Typed structured intermediate representations** were introduced for Isabelle by Draft, Sketch, and Prove (Jiang et al., 2023), whose sketch is a textual proof skeleton rather than a typed structural object. The broader formal-math LM literature on premise selection, proof generation, and math pretraining (Polu & Sutskever, 2020; Yang et al., 2023; Azerbayev et al., 2024; Lin et al., 2025) addresses different stages of the pipeline and is complementary.

We instantiate a $2\times2$ signal-coverage matrix (elaborator pass/fail $\times$ judge equivalent/not) on DeepSeek V4-Pro $\times$ ProofNet$^{\#}$ (186 problems) under a dual-judge protocol that pairs Claude Opus 4.7 (Anthropic, 2026) with GTED at $\tau$=0.5, tracking per-problem transitions from Vanilla into Lean-Retry, Sample-Filter, and Stratified Autoformalization (SAF).

This paper makes three contributions:

(1) **A diagnostic framework and the predictive regularity**

[1]School of Computer Science, University of Sydney, Sydney, Australia [2]China Academy of Information and Communications Technology, China. Correspondence to: Chengxiao Dai <cdai0023@uni.sydney.edu.au>.

The $3^{rd}$ *AI for Math Workshop* at the $43^{rd}$ International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

**it yields.** A 2×2 signal-coverage matrix and a 4×4 transition matrix decompose any TC% gain into type-stratum recovery, semantic-stratum turnover, and a durable residual. Under this lens, three algorithmically distinct methods each recover 23/61 of Vanilla TO (Wilson 95% CI [26.6%, 50.3%]), and $\Delta TC$ is linear in the Vanilla elab-fail rate at $R^2$=0.96 across six (model, dataset) cells (§2, §5.2, §5.4).

(2) **A calibrated dual-judge protocol.** Opus paired with GTED at $\tau$=0.5 achieves 94.4% to 95.0% precision across benchmarks, and a four-class taxonomy attributes 30 to 56% of GTED false negatives to elaborator-forced structural rewriting (§5.5).

(3) **SAF as a causal-attribution probe.** Stratified Autoformalization (SAF) refines a typed JSON IR against the elaborator and emits Lean by a deterministic translator. Its determinism makes the candidate's surface form a function of the IR alone, attributing the +37 pp Opus-vs-GTED gap to elaborator-driven surface geometry rather than LLM sampling noise (§5.5). SAF ties Lean-Retry and Sample-Filter on aggregate metrics. Its role here is diagnostic rather than state-of-the-art optimization.

## 2. The Signal-Coverage Matrix

Under statement-level scope (NL statement $S$ → Lean 4 signature $\sigma$ with `:= by sorry`), an output that fails its specification falls into exactly one of two strata. A type-stratum failure means $\sigma$ does not type-check (`Unknown identifier`, `failed to synthesize`, `application type mismatch`, ...), so the math may be understood but the Lean surface form is invalid. A semantic-stratum failure means $\sigma$ type-checks but denotes a different proposition from $S$, for instance hardcoding `ZMod 2` when $S$ refers to "any field of size 2", using `Summable f` when $S$ asserts the sequence has a limit, reversing an inequality, or dropping a hypothesis. (A third, implementation stratum for proof-level autoformalization is out of scope.)

The two refinement signals used in practice are stratum-specialized. The Lean elaborator is complete on the type stratum, since every failure raises an error, and silent on the semantic stratum, since a semantically wrong but type-correct $\sigma$ elaborates without complaint. Semantic judges, namely back-translation (Zhou et al., 2024) and GTED (Liu et al., 2025b), cover the semantic stratum but forgive the type stratum, since a non-compiling fragment back-translates to noisy NL.

Crossing elaboration (pass / fail) with the semantic judgment (equivalent / not-equivalent) partitions $n$ outputs into four cells (Figure 1): true success (TS), type-only failure (TO), semantic-only failure (SO), and both fail (BF). These labels denote failure modes, not correctness modes. The two off-diagonal cells are exactly the per-instance disagreements between elaborator and judge, and their size, sources, and dynamics across methods are this paper's core empirical object. Writing $|c|$ for the count in cell $c$,

$$TC\% = \frac{|\mathrm{TS}|+|\mathrm{SO}|}{n}, \quad SF\% = \frac{|\mathrm{TS}|}{n}, \quad n = \sum_c |c|. \quad (1)$$

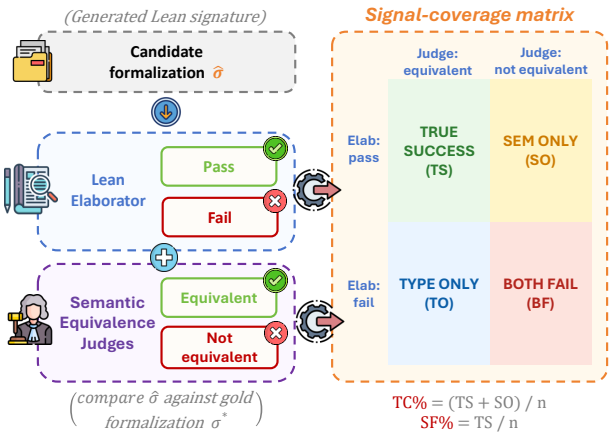


*Figure 1.* Signal-coverage matrix: elaborator (pass/fail) × judge (equivalent/not) gives the 2×2 cell decomposition and TC%/SF%.

## 3. Stratified Autoformalization

Because refinement signals are stratum-specialized (§2), SAF factors autoformalization through a typed intermediate representation (IR). The LLM emits an IR, a deterministic ∼30-line translator produces Lean from it, and on elab failure the IR is refined rather than Lean source. This determinism is central: every refined SAF output is a function of the IR alone. On the aggregate metrics, SAF ties Lean-Retry and Sample-Filter on both TC and Opus-judged SF (§5). Its role in this paper is the causal-attribution probe of §5.5, where the +37 pp Opus-vs-GTED gap is attributed to elaborator-driven surface geometry rather than to LLM sampling noise.

The refinement loop (Algorithm 1) alternates IR generation by the LLM with elaborator checking until the candidate type-checks. The total LLM budget is $K$+1 (one extraction plus $K$ refinements), and we use $K$=3 in all experiments.

A SAF IR is a JSON object with five fields:

```
{
  "imports":   ["Mathlib"],
  "opens":     ["Polynomial", "Real"],
  "objects":   [
    {"name":"G","type":"Type*", "binding":
        "implicit"},
    {"name":"", "type":"Group G","binding"
        :"instance"},
    {"name":"a","type":"G",      "binding"
        :"explicit"}
  ],
```

```
  "hypotheses":[{"id":"h", "expression":"a
      *b = b*a"}],
  "conclusion": "<Lean 4 expression>"
}
```

**Algorithm 1** SAF refinement loop

**Require:** NL statement $S$, retry budget $K$
**Ensure:** type-checked Lean signature, or $\perp$

```
ir ← LLM.extract_IR(S)
for k = 0 to K do
    σ ← ir_to_lean(ir)
    (ok, errors) ← LeanElaborator.check(σ)
    if ok then
        return σ
    end if
    ir ← LLM.refine_IR(S, ir, errors)
end for
return ⊥
```

The `imports` and `opens` fields carry the Mathlib environment, `objects` declares typed binders with their binding mode, `hypotheses` lists named premises, and `conclusion` holds the goal as a Lean 4 expression. The `ir_to_lean` translator is a ∼30-line template substitution that maps each field to its Lean counterpart: `imports` to `import` lines, `opens` to `open` lines, each `objects` entry to a typed binder (`(x : T)` / `{x : T}` / `[T]` per `binding`), `hypotheses` to hypothesis binders, and `conclusion` to the theorem body ending in `:= by sorry`. The translator is purely template-based (no LLM, no fuzzy matching, no error recovery), and malformed IR fails fast.

## 4. Experimental Setup

**Datasets.** ProofNet# (Poiroux et al., 2024) is the type-complex benchmark: 186 undergraduate-textbook problems (Rudin, Munkres, Dummit-Foote, Artin, Axler, Herstein, Pugh, Putnam, Ireland-Rosen, Shakarchi) exercising Mathlib typeclass machinery (`Group`, `MeasurableSpace`, `ContinuousMap`, `Sylow`, ...). MiniF2F-test (Zheng et al., 2022) is the type-simple benchmark: 244 olympiad problems admitting largely primitive-type formalizations. The two bracket the type-system complexity axis.

**Subject models.** DeepSeek V4-Pro (49B active / 1.6T total MoE parameters) is the primary subject, with the full battery on ProofNet# and three methods on MiniF2F. Qwen3.5-Plus (Qwen Team, 2026) (480B-class) and MiMo-v2.5-Pro (Xiaomi MiMo Team, 2026) (Xiaomi, thinking traces disabled) serve as cross-family direction checks. All decoding is at `temperature=0.0` except where $N$=4 sampling is the method itself. Gold-judge analysis is anchored on DeepSeek V4-Pro × ProofNet#, with cross-model and cross-dataset runs serving as mechanism-direction checks at coarser (GTED-only) judge resolution.

**Methods.** Four methods, three at the same ∼4-LLM-call budget. Vanilla: one-shot. Lean-Retry (Guo et al., 2025-style): emit Lean signature, refine by feeding elaborator errors back to the LLM, $K$=3 retries. Sample-Filter (Poiroux et al., 2024-style): sample $N$=4 candidates, retain elab-pass candidates, pick first. SAF (this paper, §3): typed JSON IR refined against the elaborator, translated to Lean by a deterministic ∼30-line procedure, $K$=3.

**Judges.** A dual-judge protocol pairs Claude Opus 4.7 (Anthropic, disjoint from all four subject families) as the gold cross-judge with GTED (Liu et al., 2025b) as the scalable symbolic proxy. Opus compares candidate Lean against gold Lean for semantic equivalence under a STRICT prompt when the candidate elaborates and an EQUIV-IF-FIXED prompt when it does not, yielding 744 verdicts on DeepSeek V4-Pro × ProofNet# × {Vanilla, Lean-Retry, SAF, Sample-Filter}, 244 on DeepSeek V4-Pro × MiniF2F × Vanilla, and 988 in total. GTED computes generalized tree edit distance on the LSP-extracted typed operator tree, returns a similarity in $[0, 1]$, is thresholded at $\tau$=0.5, and runs on all 19 (model, dataset, method) combinations.

**Infrastructure.** Lean 4.19.0 via `lean_interact`, prebuilt Mathlib, ATLAS-based (Liu et al., 2025a) GTED LSP backend, elab timeout 60s, and JSONL resume on all pipelines.

## 5. Experimental Results and Analysis

### 5.1. Per-method cell decomposition

For each of the 186 ProofNet# test problems run on DeepSeek V4-Pro under each of the four methods, we record the elaborator label (pass / fail) and the Opus label (equivalent / not-equivalent), assigning each output to one of the four cells defined in §2. Table 1 reports the four-method matrix, Figure 2 visualizes Vanilla's cell counts, and Figure 3 shows the cross-method TC / SF comparison (computed per Equation (1)). We use DeepSeek V4-Pro × ProofNet# as the running example because it is the only (model, dataset) cell with full four-method Opus dual-judging. Appendix E extends the same ΔTC / ΔSF comparison to the other five cells under GTED. The cell-level view is what lets us separate elaborator-side recovery (TO→TS) from semantic recovery (SO→TS) rather than read both off a single aggregate rate.

Vanilla decomposes as 95 / 16 / 61 / 14. Of the 91 failures, the elaborator catches 75 (TO + BF) and Opus catches 30 (SO + BF), with overlap of 14. The 77 / 186 outputs off-diagonal (41.4%, SO + TO) are the per-instance dis-

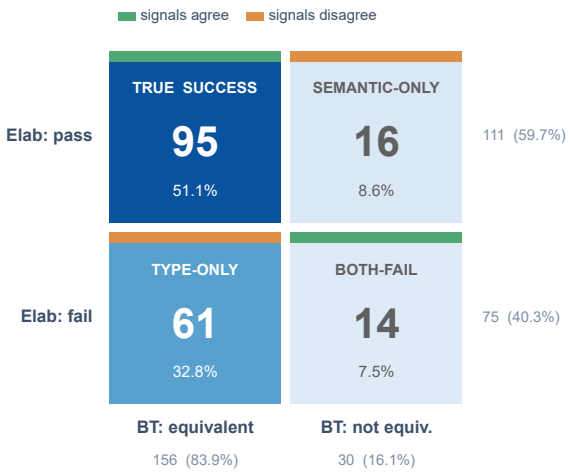


*Figure 2.* Vanilla signal-coverage matrix (DeepSeek V4-Pro × ProofNet#). Rose-outlined off-diagonal cells mark elaborator/Opus disagreements.

*Table 1.* Signal-coverage matrix on DeepSeek V4-Pro × ProofNet# ($n$=186), four methods. Cells are defined in Figure 1, labels denote failure modes.

| **Method** | TS | SO | TO | BF | **TC%** | **SF%** |
|---|---|---|---|---|---|---|
| Vanilla | 95 | 16 | 61 | 14 | 59.7 | 51.1 |
| Lean-Retry | 131 | 10 | 41 | 4 | 75.8 | 70.4 |
| Sample-Filter | 131 | 11 | 37 | 7 | 76.3 | 70.4 |
| SAF | 129 | 11 | 39 | 7 | 75.3 | 69.4 |

agreements between the two signals. The largest single recovery target for any elab-only feedback method is the 61 TO cases: semantically correct outputs that the elaborator nonetheless rejects on surface form.

The three elab-feedback methods add 34 to 36 problems to TS, shrink TO by 20 to 24, shrink BF by 7 to 10, and shrink SO by only 5–6 from its baseline of 16, despite their algorithmic differences (feedback vs. sampling, raw Lean vs. typed IR). A bootstrap 95% CI ($B$=10,000, Table 9) shows ΔTS and ΔTO significant for all three methods (CI does not cross 0), while ΔSO is not significant for any.

### 5.2. Per-problem transitions: TS gains come mostly from TO-rescue

Aggregate Δs answer which cells moved on net. To resolve which problems moved, we join each problem's Vanilla cell to its Lean-Retry cell, giving the 4×4 transition matrix in Figure 4 and the per-cell decomposition of Δ|TS| in Equation (2).

Writing $|c{\to}c'|$ for the number of problems in Vanilla cell $c$

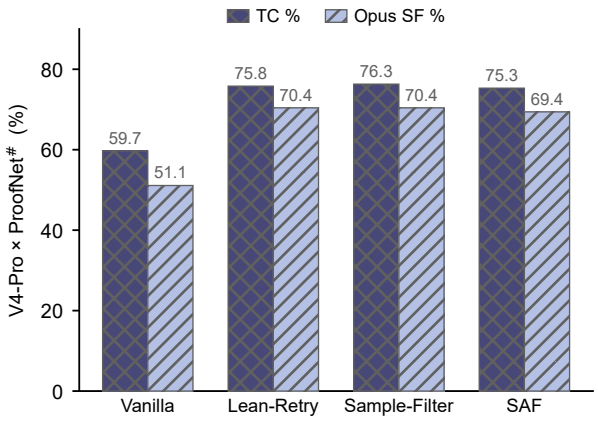


*Figure 3.* TC% and Opus-judged SF% across four methods on DeepSeek V4-Pro × ProofNet#. The three elab-feedback methods (Lean-Retry, Sample-Filter, SAF) cluster within ∼ 1 pp on both axes.

that land in Lean-Retry cell $c'$, the net TS gain is given by

$$\Delta|\mathrm{TS}| = \underbrace{|\mathrm{TO}{\to}\mathrm{TS}|}_{23} + \underbrace{|\mathrm{SO}{\to}\mathrm{TS}|}_{14} + \underbrace{|\mathrm{BF}{\to}\mathrm{TS}|}_{2} - \underbrace{|\mathrm{TS}{\to}\neg\mathrm{TS}|}_{3} = +36, \tag{2}$$

attributing $23/36 \approx 64\%$ to type-stratum recovery and $14/36 \approx 39\%$ to semantic flip on the rescued side. Per-cell individual rescue rates are: 96.8% of Vanilla TS stay TS, 87.5% (14 / 16) of Vanilla SO problems are individually rescued, and only 37.7% (23 / 61) of Vanilla TO are recovered, with the remainder stuck.

### 5.3. Semantic turnover and the gold-error floor

The aggregate SO count drops only from 16 to 10 under Lean-Retry, but this stability masks substantial per-problem turnover. 14 of 16 Vanilla SO are individually rescued (87.5%), while Lean-Retry simultaneously creates 8 new SO (4 from BF, 3 from TS regression, 1 from TO), for a net change of −6: aggregate stability hides a near-total replacement of the underlying error population.

Of the 2 Vanilla SO problems that remain SO under Lean-Retry, both are gold-formalization bugs, verified by Opus reasoning across all four methods. The first, `Ireland-Rosen|exercise_1_30`, asks "Prove that $1/2 + 1/3 + \cdots + 1/n$ is not an integer." Gold's summand $\frac{1}{n+2}$ does not depend on the summation index, so the gold sum evaluates to $n/(n+2)$, a different proposition from the harmonic tail. The candidate faithfully formalizes the harmonic tail, and Opus correctly judges candidate ≠ gold under all four methods (Theorem 1).

**Example 1** (Buggy gold formalization, $n$=1)**.**

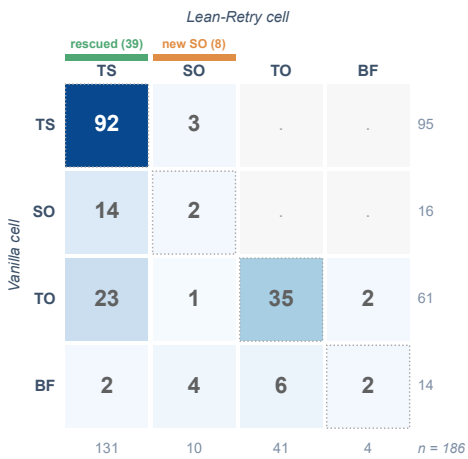


*Figure 4.* Per-problem Vanilla → Lean-Retry transition matrix. Green column: rescue into TS. Orange column: newly created SO. Dotted diagonal: stay-put. Shade encodes count.

```
-- Gold: summand '(1 : Rat) / (n+2)' is
    constant in i,
-- so the sum equals n/(n+2), not the
    harmonic tail.
theorem exercise_1_30 {n : Nat} :
  Not (Exists fun a : Int ⇒
    (Finset.sum (Finset.univ : Finset
        (Fin n))
      fun _ ⇒ (1 : Rat) / (n+2)) = a)
```

```
-- Candidate (elab pass; Opus ≠ gold
    under all 4 methods):
-- faithful formalization of the harmonic
    tail.
theorem exercise_1_30 {n : Nat} (hn : 2 ≤
     n) :
  Not (Exists fun a : Int ⇒
    (Finset.sum (Finset.Icc 2 n)
      fun k ⇒ (1 : Rat) / k) = a)
```

The companion case `Dummit-Foote|exercise_7_2_2` has gold LHS $p \mid 0$, trivially true in any commutative ring, so the biconditional collapses to a one-sided statement (Appendix F, Case D). Both cases are documented with Opus traces and suggested corrections in the supplementary note.

Across all four methods, only 1 of 186 problems is durably SO (`Ireland-Rosen|exercise_1_30`). Under Lean-Retry the count rises to 2, both gold bugs. Excluding the gold bugs, zero of 186 problems are durably semantic-only across the three elab-feedback methods, so the SO plateau approximates the gold-error floor of the benchmark.

*Table 2.* Per-stratum recovery rates on DeepSeek V4-Pro × ProofNet#, dual-judged. $\rho_{c\to\mathrm{TS}}(m)$ is the share of Vanilla cell $c$ that lands in TS under method $m$.

| Method | TO→TS | SO→TS | BF→TS | TS retain |
|---|---|---|---|---|
| Lean-Retry | 37.7% | 87.5% | 14.3% | 96.8% |
| SAF | 37.7% | 75.0% | 42.9% | 92.6% |
| Sample-Filter | 37.7% | 68.8% | 28.6% | 97.9% |
| mean (cross-method) | 37.7% | 77.1% | 28.6% | 95.8% |
| half-spread | 0 pp | 9.4 pp | 14.3 pp | 2.6 pp |

### 5.4. A predictive regularity: stratum-rates are method-invariant

The aggregate gains of §5.1 and the transition matrix of §5.2 naturally raise the question: do the three elab-feedback methods recover the *same* problems at the *same* rate, or do they merely converge in aggregate? For each Vanilla cell $c \in \{\mathrm{TS}, \mathrm{SO}, \mathrm{TO}, \mathrm{BF}\}$ and each elab-feedback method $m$, define the recovery rate

$$\rho_{c\to\mathrm{TS}}(m) \;=\; \frac{|\{p : \mathrm{Vanilla}(p) = c \wedge m(p) = \mathrm{TS}\}|}{|\{p : \mathrm{Vanilla}(p) = c\}|}. \tag{3}$$

Table 2 reports the four rates for each of the three elab-feedback methods.

All three elab-feedback methods recover *exactly* 23 of the 61 Vanilla TO problems, despite differing in algorithm (sequential elaborator feedback, deterministic IR refinement, parallel $N$=4 sampling), in LLM call topology, and in candidate generation entropy. The point estimate has a wide Wilson 95% CI of $[26.6\%, 50.3\%]$ at $n$=61, so we do not claim the true rate is precisely 37.7%. What we claim, and what the predictive regularity of Table 3 rests on, is that the three methods recover statistically indistinguishable populations on the type stratum: per-method CIs overlap entirely, and a Fisher exact test of pairwise homogeneity returns $p$=1.00 for all three pairs. Appendix A gives Wilson CIs for every cell of Table 2 and discusses the limits of the $n$=186 sample. The semantic recovery rate is high and tighter than the BF rate (the latter is small-$n$ noise, 14 BF problems total). The retention rate on TS is uniformly $\geq 92.6\%$, so net regression is small.

Taking any single elab-feedback method's rates and applying them as a mass-balance rule to the Vanilla cell distribution $(|\mathrm{TS}|, |\mathrm{SO}|, |\mathrm{TO}|, |\mathrm{BF}|) = (95, 16, 61, 14)$ yields a testable prediction for the other methods. With Lean-Retry's rates, $\Delta\hat{\mathrm{TS}} = 0.377{\cdot}61 + 0.875{\cdot}16 + 0.143{\cdot}14 - (1 - 0.968){\cdot}95 = +36.0$. Table 3 compares predicted to observed: the calibration on Lean-Retry predicts the other two methods' $\Delta$TS to within 2 of 186 problems, an order of magnitude better than the bootstrap noise band of any cell.

Extending across the six (model, dataset) cells of the cross-

*Table 3.* Predicted vs. observed ΔTS on DeepSeek V4-Pro × ProofNet#, using stratum-rates calibrated from Lean-Retry.

| Method | Predicted ΔTS | Observed ΔTS | Residual |
|---|---|---|---|
| Lean-Retry | +36.0 | +36 | 0.0 |
| SAF | +36.0 | +34 | −2.0 |
| Sample-Filter | +36.0 | +36 | 0.0 |

*Table 4.* Cross-cell fit of Equation (4) on six (model, dataset) cells under Lean-Retry. Rows are grouped by benchmark.

| Model | $V_{\text{fail}}$% | $\Delta TC$ obs. | $\Delta TC$ pred. | residual |
|---|---|---|---|---|
| *ProofNet# (type-complex, n=186)* | | | | |
| DeepSeek V4-Pro | 40.3 | +16.1 | +16.0 | +0.1 |
| Qwen3.5-Plus | 28.5 | +12.9 | +12.1 | +0.8 |
| MiMo-v2.5-Pro | 32.3 | +12.4 | +13.4 | −1.0 |
| *MiniF2F (type-simple, n=244)* | | | | |
| DeepSeek V4-Pro | 12.7 | + 6.1 | + 7.0 | −0.9 |
| Qwen3.5-Plus | 6.1 | + 4.5 | + 4.9 | −0.4 |
| MiMo-v2.5-Pro | 13.5 | + 8.6 | + 7.3 | +1.3 |

*Table 5.* Opus vs. GTED SF% on DeepSeek V4-Pro × ProofNet# outputs.

| Method | TC% | Opus SF% | GTED SF% | Gap |
|---|---|---|---|---|
| Vanilla | 59.7 | 51.1 | 44.1 | +7.0 |
| Lean-Retry | 75.8 | 70.4 | 44.1 | +26.3 |
| SAF | 75.3 | 69.4 | 32.3 | +37.1 |
| Sample-Filter | 76.3 | 70.4 | 43.5 | +26.9 |

*Table 6.* GTED $\tau$ calibration against Opus. $P = P(\text{Opus equiv} \mid \text{GTED} \geq \tau)$, $R$ is recall on the Opus-equiv elab-pass set. Recommended row $\tau$=0.5 in blue underline. Per-method ProofNet# precision in Table 10.

| | *MiniF2F, Vanilla* | | *ProofNet#, 4-method pool* | |
|---|---|---|---|---|
| $\tau$ | $P$ | $R$ | $P$ | $R$ |
| 0.3 | 88.5% | 62.0% | 93.1% | 75.1% |
| 0.5 | 94.4% | 45.5% | 95.0% | 51.0% |
| 0.7 | 97.8% | 23.5% | 98.1% | 21.2% |

model panel, where the test is necessarily at the TC level (Opus dual-judging is anchored on the V4-Pro × ProofNet# cell), the recovery rate of Vanilla elab-fails under Lean-Retry fits a single linear regression in the Vanilla elab-fail rate $V_{\text{fail}}$:

$$\widehat{\Delta TC} = 0.325 \cdot V_{\text{fail}} + 2.88 \quad (R^2 = 0.957), \qquad (4)$$

with per-cell residuals in $[-1.0, +1.3]$ pp (Table 4). The recovery rate of elab-fails ($\Delta TC/V_{\text{fail}}$) is 41.2% on ProofNet# (range $[38.3, 45.3]$) and 61.8% on MiniF2F (range $[48.4, 73.3]$): the same elab-signal recovers a strictly higher share of failures on type-simple benchmarks because type-simple failures concentrate in the easy surface-form band that $K$=3 refinement resolves.

Two consequences for method comparison follow. First, a new method's *aggregate* $\Delta TC$ is partially determined by its target's Vanilla cell distribution, not only by the method's algorithmic innovations: a method evaluated only on a type-complex benchmark will look weaker than the same method on MiniF2F, by a benchmark-modulation factor predicted from $V_{\text{fail}}$ alone. Second, the type-stratum recovery rate of $23/61$ is what current elab-feedback methods deliver, and closing the gap to the elab-only ceiling $(|\text{TS}|+|\text{TO}|)/n = 83.9\%$ would require recovering hard TO cases that survive $K$=3 refinement, a population on which no current method makes progress (§6). The within-$2/186$ agreement is partly self-consistency, since the three methods already converge in aggregate (Table 1).

### 5.5. Opus vs. GTED: the surface-form rewriting gap

On the same 186 DeepSeek V4-Pro × ProofNet# outputs, Claude Opus 4.7 and GTED $\tau$=0.5 give different SF% numbers (Table 5). SAF's deterministic IR translator isolates the cause.

On Vanilla the two judges agree to within +7 pp. On the three elab-feedback methods they disagree by +26 to +37 pp, with Opus reading SF as rising and GTED reading it as flat or falling. The mechanism: GTED penalizes surface-form normalizations (alternative `opens`, `IsCompact univ`↔`CompactSpace`, $\mathrm{Fin}\, n \to \mathbb{R}$ vs. $\forall i : \mathbb{N}$, $\mathrm{Icc}\, 0\, 1$ restatements) that elab-feedback adopts to satisfy the elaborator, while Opus recognizes these as semantically equivalent. Theorem 2 is a representative instance on `Munkres|exercise_29_4`: gold and candidate denote the same Π type, but GTED returns similarity 0.143 because the extracted operator trees place an arrow node where the other places a ∀ binder.

**Example 2** (Surface-form gap, $n$=1)**.**

```
-- Gold
theorem exercise_29_4 :
  Not (LocallyCompactSpace
        (Nat → Set.Icc (0 : Real) 1))
```

```
-- Vanilla candidate (elab pass; Opus
   equiv; GTED sim 0.143)
theorem exercise_29_4 :
  Not (LocallyCompactSpace
        (forall i : Nat, Set.Icc (0 :
           Real) 1))
```

The gap is largest for SAF (+37 pp): with surface form fixed by the deterministic translator, divergence is the IR's choice, not LLM sampling noise. Additional cases (TO→TS recoveries, gold bugs) are catalogued in Appendix F.

*Table 7.* Taxonomy of Opus-vs-GTED disagreements on DeepSeek V4-Pro × ProofNet#. Counts: problems with Opus equiv ∧ GTED $< 0.5$ ∧ elab-pass. Percentages: share of each method's false-negative pool.

| Method | N (%) | I (%) | S (%) | Z (%) | Total |
|---|---|---|---|---|---|
| Vanilla | 4 (10.5) | 6 (15.8) | 14 (36.8) | 14 (36.8) | 38 |
| Lean-Retry | 11 (17.7) | 9 (14.5) | 30 (48.4) | 12 (19.4) | 62 |
| SAF | 11 (14.5) | 8 (10.5) | 23 (30.3) | 34 (44.7) | 76 |
| Sample-Filter | 3 ( 4.8) | 6 ( 9.7) | 35 (56.5) | 18 (29.0) | 62 |

Table 6 confirms that $\tau$=0.5 maintains 95.0% pooled precision on ProofNet# and $\geq 93\%$ on every method individually (per-method breakdown in Table 10). The threshold is stable across distributions, so the gap is a structural property of elab-feedback's surface rewriting rather than of judge calibration.

The Opus-equiv but GTED-non-equiv set ($n$=38 on Vanilla, $n \in [62, 76]$ on the three elab-feedback methods) is what drives the gap. We classify each case by the type of rewrite that distinguishes the candidate from gold (rules and per-method audit in Appendix D):

- **Notational (N).** Surface rewrites that preserve definitional shape: `A->B` vs. `forall _ : A, B`, `Real^n` vs. `Fin n -> Real`, coercion (`ZMod 2` for the unique field of order 2), alternative `open`/namespace prefixes, or cosmetic renaming.
- **Idiomatic (I).** Definitional aliases or Mathlib formulation swaps: `IsConj` vs. `exists g, b = g*a*g⁻¹`, `Summable` vs. `Tendsto` of partial sums, `CompactSpace` vs. `IsCompact Set.univ`, or `LinearEquiv`-via-`Nonempty` typeclass framing.
- **Structural (S).** Binder, quantifier, or typeclass restructuring that changes the post-elaboration term: `let` vs. explicit hypothesis, instance binder vs. hypothesis binder, `[Field]` vs. `(h : Field)`, quantifier moved inside or outside an existential, `Iff` direction swap, biconditional introduced or collapsed, or `subtype` vs. predicate-restricted set.
- **Residual (Z).** Cases that mix multiple categories or resist single-category classification.

Table 7 reads as follows. On Vanilla, 38 of 95 elab-pass outputs (40%) are surface-form rewrites Opus accepts but GTED does not, distributed across all four categories. Under each elab-feedback method the false-negative count grows by 24 to 38 problems. For Lean-Retry and Sample-Filter the growth concentrates in category S (16 of 24 and 21 of 24 added cases, respectively): the elaborator pressures the LLM to restructure binders, typeclass formulations, and quantifier orderings until the output type-checks. SAF's growth is more diffuse: only 9 added in S (the smallest, because SAF's deterministic `ir_to_lean` translator does not freely restructure binders) but +20 in Z, reflecting the multi-category rewrites the IR-mediated path tends to produce. The disagreement gap is therefore not random surface variance but a measurement of elaborator-driven structural rewriting, which is precisely the mechanism that recovers TO into TS (§5.2).

### 5.6. Cross-dataset: SO rate stable, TO rate scales with type-complexity

DeepSeek V4-Pro × MiniF2F-test × Vanilla (244 problems, Opus-judged) decomposes as TS=163, SO=24, TO=51, BF=6. The SO rate is 9.8%, essentially unchanged from 8.6% on type-complex ProofNet#. The TO rate, by contrast, drops from 32.8% to 20.9%. Dataset type-complexity loads the type stratum and not the semantic stratum, providing supporting evidence at Vanilla-only resolution for the mechanism of §5.2. Full MiniF2F dual-judge analysis across the four methods is future work.

### 5.7. Cross-model: TC gain robust to subject model

Figure 5 reports TC% for three subject models on both datasets. Lean-Retry $\geq$ Vanilla in 6/6 cells (mean $\Delta$TC = +10.1 pp). SAF $\geq$ Vanilla in 5/6, the one exception (Qwen × ProofNet#, −12.9 pp) being a high IR-extraction failure rate on Qwen3.5-Plus, reported without ad-hoc patching. Under GTED-only judging, $\Delta$TC $>$ $\Delta$SF holds in 6/6 cells for Lean-Retry (mean +10.1 vs. +1.3 pp, Table 11).

### 5.8. K-saturation: refinement saturates fast and three methods converge

Figure 6 plots SAF's K-saturation on DeepSeek V4-Pro × ProofNet#. TC% rises 50.5 → 72.0 → 74.7 → 75.3% across $K$=0, 1, 2, 3. Refinement saturates at $K$=1 (+21.5 pp), with $K$=2, 3 contributing +3.3 pp combined. All three methods land within 1.0 pp at saturation (SAF 75.3, Lean-Retry 75.8, Sample-Filter 76.3), so different topologies reach the same ceiling. Per-iteration faithfulness remains high: $k^*$=0: 89/94 (94.7%), $k^*$=1: 34/40 (85.0%), $k^*$=2: 5/5 (100%), and $k^*$=3: 1/1 (100%). Of the 46 problems that never pass, 39 (84.8%) are Opus equiv-if-fixed (i.e., TO). The 16 durably TO across all four methods share a common failure mode: plausible but invalid Mathlib API names that elab errors cannot prescribe (Appendix F, Case E).

## 6. Discussion and Limitations

The (TC $\neq$ SF) gap itself is not new, but the per-stratum quantification is. On type-complex benchmarks TC% alone is misleading: the recommended reporting unit is TC%, dual-judge SF%, and the per-cell decomposition

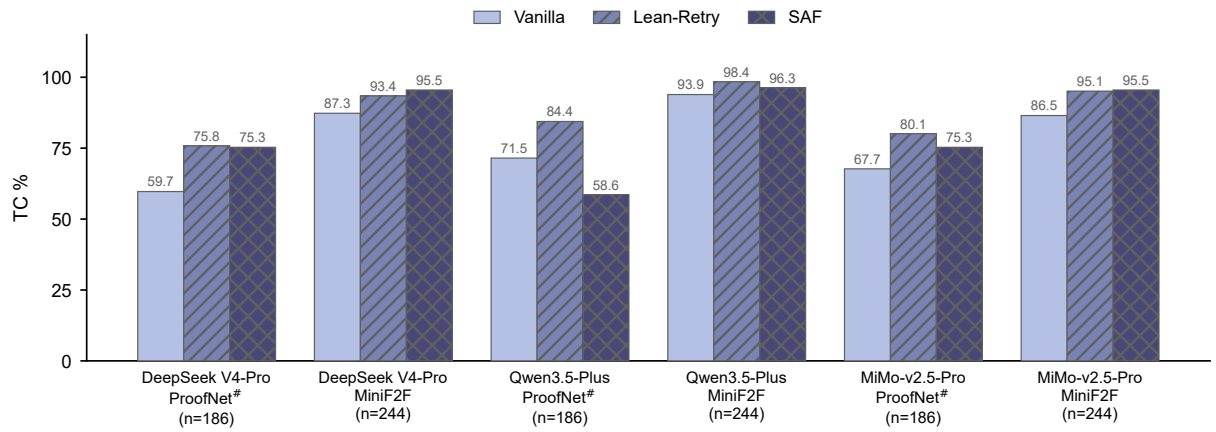


*Figure 5.* Cross-model TC% on 3 models × 2 datasets × 3 methods.

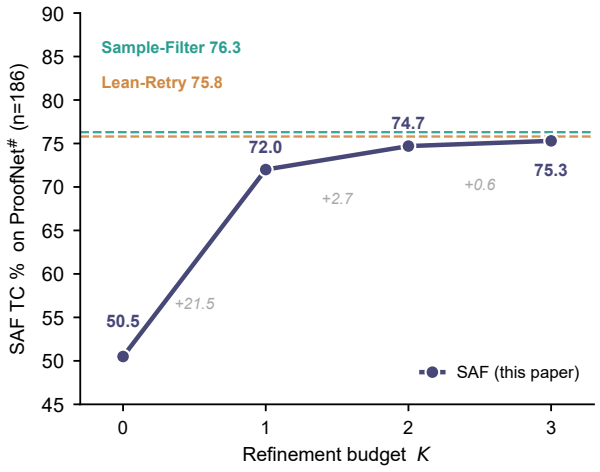


*Figure 6.* SAF K-saturation on DeepSeek V4-Pro × ProofNet#. Dashed bands: Lean-Retry and Sample-Filter saturation levels.

(TS/SO/TO/BF). Two elab-feedback methods compared under a single symbolic judge can differ by $\geq 25$ pp purely from structural rewriting, so independent-LLM verification is required for any SF% claim.

**Why three methods converge.** The collapse to within ∼1 pp TC% at saturation (§5.8), the identical 37.7% TO→TS rate (§5.4), and the shared structural-rewriting signature in the Opus/GTED gap (§5.5) are three views of the same claim: at $K \leq 3$ the elab signal is largely exhausted by the fixed ∼4-call budget, so all three refinement topologies converge to the same family of type-checking surface forms. The remaining axis of progress is the signal itself, not the topology on top of it.

Per-output semantic faithfulness is unreliable under elab-feedback because of turnover (87.5% rescue offset by 8 newly created errors per method). Future signals must correlate with semantic correctness rather than only distinguish elab-pass from elab-fail.

SAF's deterministic IR translation serves as a causal-attribution probe for the surface-form rewriting mechanism (§5.5) and the per-stratum predictive regularity (§5.4). Two ProofNet# gold formalizations contain gold-formalization errors (`Ireland-Rosen|exercise_1_30`, `Dummit-Foote|exercise_7_2_2`) and are documented with suggested corrections in the supplementary note. Benchmarks aiming to measure SO below 1% will need gold audits, not larger LLMs alone.

Our semantic labels rely on Claude Opus 4.7 as a strong cross-judge rather than expert Lean validators ($\geq$ 94% pooled precision in §5.5). A small expert audit and a canonicalization-aware symbolic metric (e.g. operator-tree matching after `whnf` reduction) would absorb most of the N/I taxonomy categories without an LLM judge. Both are natural follow-ups.

## 7. Conclusion

We decompose TC% gains via a 2×2 signal-coverage matrix and a 4×4 per-problem transition matrix. On DeepSeek V4-Pro × ProofNet#, three elab-feedback methods deliver +34 to +36 TS via ∼64% type-stratum recovery (Equation (2)), with SO net-flat but 87.5% per-problem turnover. The TO→TS rate is 23/61 for each method (Wilson 95% CI [26.6%, 50.3%]), cross-method calibration predicts ΔTS to within 2/186, and $\Delta TC$ fits a linear regression in the Vanilla elab-fail rate at $R^2$=0.96 across six (model, dataset) cells. A dual-judge protocol (Opus and GTED at 95% pooled precision) shows that single-judge SF% under-credits elab-feedback methods by $\geq 25$ pp via elaborator-forced structural rewrites.

## References


Anthropic. The Claude 4 model family. https://www.anthropic.com/claude, 2026. Model card.

Azerbayev, Z., Piotrowski, B., Schoelkopf, H., Ayers, E. W., Radev, D., and Avigad, J. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023.

Azerbayev, Z., Schoelkopf, H., Paster, K., Dos Santos, M., McAleer, S., Jiang, A. Q., Deng, J., Biderman, S., and Welleck, S. Llemma: An open language model for mathematics. In *International Conference on Learning Representations*, 2024.

DeepSeek-AI. DeepSeek-V4 technical report. https://huggingface.co/deepseek-ai, 2026. Technical report.

Guo, Q., Wang, J., Zhang, J., Kong, D., Huang, X., Xi, X., Wang, W., Wang, J., Cai, X., Zhang, S., and Ye, W. Autoformalizer with tool feedback. *arXiv preprint arXiv:2510.06857*, 2025.

Jiang, A. Q., Welleck, S., Zhou, J. P., Li, W., Liu, J., Jamnik, M., Lacroix, T., Wu, Y., and Lample, G. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *International Conference on Learning Representations*, 2023.

Lin, H., Sun, Z., Welleck, S., and Yang, Y. Lean-STaR: Learning to interleave thinking and proving. In *International Conference on Learning Representations*, 2025.

Liu, X., Bao, K., Zhang, J., Liu, Y., Chen, Y., Liu, Y., Jiao, Y., and Luo, T. ATLAS: Autoformalizing theorems through lifting, augmentation, and synthesis of data. *arXiv preprint arXiv:2502.05567*, 2025a.

Liu, Y., Zhu, T., Liu, X., Chen, Y., Liu, Z., Guo, Q., Zhang, J., Bao, K., and Luo, T. Generalized tree edit distance (GTED): A faithful evaluation metric for statement autoformalization. In *ICML 2025 Workshop on AI for Math (AI4Math)*, 2025b. arXiv:2507.07399.

Lu, J., Wan, Y., Liu, Z., Huang, Y., Xiong, J., Liu, C., Shen, J., Jin, H., Zhang, J., Wang, H., Yang, Z., Tang, J., and Guo, Z. Process-driven autoformalization in lean 4. *arXiv preprint arXiv:2406.01940*, 2024.

OpenAI. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/, 2024.

Poiroux, A., Weiss, G., Kunčak, V., and Bosselut, A. Improving autoformalization using type checking. *arXiv preprint arXiv:2406.07222*, 2024.

Polu, S. and Sutskever, I. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

Qwen Team. Qwen3.5 technical report. https://qwenlm.github.io, 2026. Technical report.

Wu, Y., Jiang, A. Q., Li, W., Rabe, M., Staats, C., Jamnik, M., and Szegedy, C. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368, 2022.

Xiaomi MiMo Team. MiMo-V2.5-Pro: Reasoning model technical report. https://github.com/XiaomiMiMo, 2026. Technical report.

Yang, K., Swope, A., Gu, A., Chalamala, R., Song, P., Yu, S., Godil, S., Prenger, R., and Anandkumar, A. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, 2023.

Zheng, K., Han, J. M., and Polu, S. MiniF2F: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*, 2022. doi: 10.48550/arXiv.2109.00110. URL https://arxiv.org/abs/2109.00110.

Zhou, J. P., Staats, C., Li, W., Szegedy, C., Weinberger, K. Q., and Wu, Y. Don't trust: Verify – grounding llm quantitative reasoning with autoformalization. In *International Conference on Learning Representations*, 2024.

## A. Wilson 95% CI on per-stratum recovery rates

Table 8 gives Wilson 95% CIs for every entry of Table 2. Pairwise Fisher exact tests for homogeneity of the TO→TS rate return $p$=1.0 for all three pairs (the observed counts are identical, $23/61$ for each method). The semantic and BF recovery rates have overlapping CIs across methods, consistent with a shared underlying rate plus small-sample noise. The wide BF CI ($\pm\sim$20 pp) reflects $n$=14 rather than instability of the recovery mechanism.

*Table 8.* Wilson 95% CI for each entry of Table 2.

| Method | TO→TS | SO→TS | BF→TS | TS retain |
|---|---|---|---|---|
| Lean-Retry | 37.7% [26.6, 50.3] | 87.5% [64.0, 96.5] | 14.3% [ 4.0, 39.9] | 96.8% [91.1, 98.9] |
| SAF | 37.7% [26.6, 50.3] | 75.0% [50.5, 89.8] | 42.9% [21.4, 67.4] | 92.6% [85.6, 96.4] |
| Sample-Filter | 37.7% [26.6, 50.3] | 68.8% [44.4, 85.8] | 28.6% [11.7, 54.6] | 97.9% [92.6, 99.4] |

## B. Bootstrap 95% CI on stratum-shift $\Delta$s

Resampling is at the problem level rather than the cell level because the four cell counts on a given problem are coupled (each problem contributes to exactly one cell per method), and intervals are the percentile method on $B$=10,000 resamples (seed 20260512) on DeepSeek V4-Pro × ProofNet$^{\#}$ ($n$=186). The interval containment pattern is uniform across the three elab-feedback methods: $\Delta$TS and $\Delta$TO are bounded away from 0 for all three, $\Delta$SO CIs all straddle 0, and $\Delta$BF reaches significance only on Lean-Retry, which gives the precise numerical backing for the qualitative claim of §5.1.

*Table 9.* Bootstrap 95% CI on $\Delta$ vs. Vanilla ($B$=10,000). $\Delta$BF significant only for Lean-Retry.

| Method | $\Delta$TS | $\Delta$SO | $\Delta$TO | $\Delta$BF |
|---|---|---|---|---|
| Lean-Retry | $\underline{+36}$ [+25, +48] | $\underline{-6}$ [−15, +3] | −20 [−31, −10] | $\underline{-10}$ [−17, −3] |
| SAF | +34 [+22, +47] | −5 [−14, +4] | −22 [−33, −11] | −7 [−15, 0] |
| Sample-Filter | $\underline{+36}$ [+25, +47] | −5 [−14, +3] | $\underline{-24}$ [−35, −13] | −7 [−14, 0] |

## C. GTED $\tau$ calibration: per-method precision

Table 6 of §5.5 reports the 4-method pooled precision at $\tau$=0.5 (95.0% on ProofNet$^{\#}$). The breakdown below shows that no single method drags the pool: per-method precision stays at $\geq$ 93.4% at $\tau$=0.5 and rises monotonically with $\tau$, so the chosen threshold is robust to which method's elab-pass outputs are pooled into the calibration set.

*Table 10.* GTED precision at three thresholds, per-method on DeepSeek V4-Pro × ProofNet$^{\#}$ elab-pass subset.

| $\tau$ | Vanilla | Lean-Retry | SAF | Sample-Filter |
|---|---|---|---|---|
| 0.3 | 87.4% | 95.2% | $\underline{95.7\%}$ | 93.5% |
| 0.5 | 93.4% | $\underline{97.2\%}$ | 94.6% | 94.5% |
| 0.7 | 96.3% | $\underline{100.0\%}$ | $\underline{100.0\%}$ | 96.8% |

## D. Disagreement taxonomy: rules and audit

Table 7 of §5.5 reports a four-class classification of every Opus-equiv but GTED $<$ 0.5 output on DeepSeek V4-Pro × ProofNet$^{\#}$, across four methods. Classes are defined as follows:

**N (Notational).** The rewrite preserves the operator-tree shape modulo Lean's surface notation: `A -> B` ↔ `forall _ : A, B`, `Real^n` ↔ `Fin n -> Real`, alternative `open`/namespace prefixes, coercion (`ZMod 2` as representative for "any field of order 2"), `Function.Injective` ↔ `Injective`, and cosmetic variable renaming. GTED penalises these because its operator tree is extracted pre-reduction.

**I (Idiomatic).** The rewrite swaps one Mathlib idiom for a definitionally equivalent or near-equivalent one: `IsConj` ↔ `exists g, b = g*a*g⁻¹`, `Summable` ↔ `Tendsto` of partial sums, `CompactSpace` ↔ `IsCompact`

`Set.univ`, `LocallyCompactSpace` ↔ `Nonempty`-wrapped `LinearEquiv`, `generateFrom = sInf` ↔ `generateFrom (sInter ...)`, etc.

**S (Structural).** The rewrite changes term structure: binder ordering, hypothesis hoisting (`let`-binding vs. explicit hypothesis vs. premise), typeclass formulation (`[Field K]` vs. `(h : Field K)`, `[Fact p.Prime]` vs. `Nat.Primes`, `[GCDMonoid R]` vs. explicit gcd predicate), quantifier reordering across an existential or universal, `Iff` direction swap or biconditional collapse, implication-vs-biconditional substitution, subtype-vs-predicate-restricted-set, and hypothesis omission of vacuously-true premises (e.g. `0 < n` when $n$=0 admits no counterexample).

**Z (Residual).** Cases mixing multiple categories (e.g. both a notational and a structural rewrite) or specific to NL ambiguity that do not cleanly map to one class.

Classification rules are encoded as a regex cascade evaluated in the order I → N → S → Z, and the first match wins. Rules are tuned on the Opus rationale text, then frozen before the per-method counts were tabulated. The full classifier ships with the supplementary script `analysis/disagreement_taxonomy.py`, which also dumps the per-case rationale string and assigned label for every entry in Table 7. Limitations of the classifier: (i) the Z bucket absorbs cases that mix two structural levels or that the rationale describes ambiguously, and (ii) borderline I vs. N decisions (e.g. `deriv`$^{[n]}$ vs. `iteratedDeriv n`) are made by the first matching rule rather than by adjudication. The dominant-class claim of §5.5 (that S grows under elab-feedback) is robust to these noise sources because S's share rises by 12 to 20 pp between Vanilla and the elab-feedback methods, well outside any plausible reassignment of the Z residual.

## E. Cross-method TC/SF dissociation under GTED (6 (model, dataset) combos)

This table reports the GTED-judged Δ vs. Vanilla on the five (model, dataset) cells outside the Opus-judged DeepSeek V4-Pro × ProofNet# baseline of §5.1. The dissociation pattern reappears: ΔTC% is positive in 11 of 12 (method, model, dataset) rows (up to +16.1 pp), while ΔSF% stays within ±5 pp in 11 of 12 rows and is non-positive in 6 of 12: elab-feedback moves the elab axis without moving the semantic axis on every cell tested.

*Table 11.* Δ vs. Vanilla under GTED $\tau$=0.5 across the six (model, dataset) cells. Left: ProofNet#. Right: MiniF2F.

*(a)* ProofNet# (type-complex, $n$=186)

| Model | Method | ΔTC% | ΔSF% |
|---|---|---|---|
| DeepSeek V4-Pro | Lean-Retry | +16.1 | 0.0 |
| DeepSeek V4-Pro | SAF | +15.6 | −11.8 |
| Qwen3.5-Plus | Lean-Retry | +12.9 | +4.3 |
| Qwen3.5-Plus | SAF | −12.9 | −5.4 |
| MiMo-v2.5-Pro | Lean-Retry | +12.4 | +1.6 |
| MiMo-v2.5-Pro | SAF | +7.5 | −5.4 |

*(b)* MiniF2F (type-simple, $n$=244)

| Model | Method | ΔTC% | ΔSF% |
|---|---|---|---|
| DeepSeek V4-Pro | Lean-Retry | +6.1 | +0.8 |
| DeepSeek V4-Pro | SAF | +8.2 | 0.0 |
| Qwen3.5-Plus | Lean-Retry | +4.5 | +0.8 |
| Qwen3.5-Plus | SAF | +2.4 | +4.9 |
| MiMo-v2.5-Pro | Lean-Retry | +8.6 | +0.4 |
| MiMo-v2.5-Pro | SAF | +9.0 | −3.7 |

## F. Case studies

Each case below shows the natural-language statement, the gold formalization, and one or more candidate outputs. Gold and candidates are reproduced in Lean 4 syntax with implicit elaboration arguments preserved. Outcome tags use the cell labels of §5.1. The five cases together cover the four mechanisms identified in the body: Case A is a TO→TS recovery (§5.2), Case B a canonical GTED false negative driven by surface-form rewriting (§5.5), Cases C and D the two distinct gold-formalization bug shapes that pin the durable SO floor (§5.3), and Case E a representative TO that survives $K$=3 refinement across all four methods (§5.8).

**Case A: TYPE_ONLY → TRUE_SUCCESS under Lean-Retry (`Dummit-Foote|exercise_11_1_13`).** **NL.** "As vector spaces over $\mathbb{Q}$, $\mathbb{R}^n \cong \mathbb{R}$, for all $n \in \mathbb{Z}^+$."

```
-- Gold (Mathlib idiom)
theorem exercise_11_1_13 {n : Nat} (hn : 0 < n) :
  Nonempty (LinearEquiv (RingHom.id Rat)
                        (Fin n → Real) Real)
```

```
-- Vanilla (elab fail): `Real^n` does not resolve to a type
-- in this position (no `HPow Type Nat Type` instance)
theorem exercise_11_1_13 {n : Nat} (hn : 0 < n) :
  Nonempty (LinearEquiv (RingHom.id Rat)
                        (Real^n) Real)
```

```
-- Lean-Retry K=1 (elab pass; matches gold idiom)
theorem exercise_11_1_13 {n : Nat} (hn : 0 < n) :
  Nonempty (LinearEquiv (RingHom.id Rat)
                        (Fin n → Real) Real)
```

Opus judges equiv to gold under both STRICT and EQUIV-IF-FIXED prompts. One of 23 TO→TS recoveries.

**Case B: GTED false negative on a surface-form rewrite (`Munkres|exercise_29_4`).** **NL.** "Show that $[0,1]^\omega$ is not locally compact in the uniform topology."

```
-- Gold
theorem exercise_29_4 :
  Not (LocallyCompactSpace (Nat → Set.Icc (0 : Real) 1))
```

```
-- Vanilla (elab pass; Opus equiv; GTED sim = 0.143)
theorem exercise_29_4 :
  Not (LocallyCompactSpace
        (forall i : Nat, Set.Icc (0 : Real) 1))
```

The two function types are the same Π type with different surface notations (`A -> B` is itself notation for `forall _ : A, B`). The elaborator treats them identically. The typed operator tree GTED extracts from each is structurally divergent (arrow node vs. ∀ binder), giving the 0.143 similarity. Canonical shape of the +26 to +37 pp Opus-vs-GTED gap on elab-feedback outputs (§5.5).

**Case C: Durable SEM_ONLY from a buggy gold formalization (`Ireland-Rosen|exercise_1_30`).** **NL.** "Prove that $1/2 + 1/3 + \cdots + 1/n$ is not an integer."

```
-- Gold: summand `(1 : Rat) / (n+2)` is constant in i,
-- so the sum equals n/(n+2), not the harmonic tail.
theorem exercise_1_30 {n : Nat} :
  Not (Exists fun a : Int ⇒
    (Finset.sum (Finset.univ : Finset (Fin n))
      fun _ ⇒ (1 : Rat) / (n+2)) = a)
```

```
-- Candidate (elab pass; Opus ≠ gold under all 4 methods):
-- faithfully formalizes the harmonic tail.
theorem exercise_1_30 {n : Nat} (hn : 2 ≤ n) :
  Not (Exists fun a : Int ⇒
    (Finset.sum (Finset.Icc 2 n)
      fun k ⇒ (1 : Rat) / k) = a)
```

The gold theorem and the natural-language statement are different propositions: the gold sum is a rational $n/(n+2)$ rather than the harmonic tail. The only problem durably SO across all four methods on DeepSeek V4-Pro × ProofNet$^\#$.

**Case D: Durable SEM_ONLY from a vacuous gold biconditional (`Dummit-Foote|exercise_7_2_2`).** **NL.** "Let $p(x) = a_n x^n + \cdots + a_0 \in R[x]$. Prove that $p(x)$ is a zero divisor in $R[x]$ iff there is a nonzero $b \in R$ with $b \cdot p(x) = 0$."

```
-- Gold: LHS `Dvd.dvd p 0` is trivially true (p * 0 = 0),
-- so the biconditional collapses to a one-sided Exists.
theorem exercise_7_2_2 {R : Type*} [Ring R]
    (p : Polynomial R) :
  Iff (Dvd.dvd p 0)
      (Exists fun b : R ⇒ b ≠ 0 /\ b * p = 0)
```

```
-- Candidate (elab pass under Lean-Retry / Sample-Filter;
-- Opus ≠ gold): captures zero-divisor non-trivially.
theorem exercise_7_2_2 {R : Type*} [CommRing R]
    (p : Polynomial R) :
  Iff (Exists fun q : Polynomial R ⇒ q ≠ 0 /\ p * q = 0)
      (Exists fun b : R           ⇒ b ≠ 0 /\ b * p = 0)
```

Gold's LHS is vacuous over any commutative ring, collapsing the biconditional to its right-hand side. Durably SO under Lean-Retry and Sample-Filter. Under SAF the IR refinement loop fails to elaborate at all (BF). Both gold-formalization bugs are documented with Opus reasoning traces in the supplementary note accompanying this submission.

**Case E: Durable TO across all four methods (`Dummit-Foote|exercise_3_4_5b`).** **NL.** "Prove that quotient groups of a solvable group are solvable."

```
-- Gold (Mathlib idiom): the class name is `IsSolvable`.
theorem exercise_3_4_5b {G : Type*} [Group G] [IsSolvable G]
    (H : Subgroup G) [Normal H] :
  IsSolvable (G / H)
```

```
-- Vanilla (elab fail; Opus equiv-if-fixed):
-- uses `Solvable`, but the Mathlib class is `IsSolvable`.
theorem solvable_quotient {G : Type*} [Group G] [Solvable G]
    (H : Subgroup G) [H.Normal] :
  Solvable (G / H)
```

```
-- Lean-Retry K=3 (still elab fail; Opus equiv-if-fixed):
-- the elaborator reports `unknown identifier Solvable` but does
-- not surface the correct Mathlib name. The LLM only adjusts
-- binder syntax (`[H.Normal]` → `[h : H.Normal]`).
theorem solvable_quotient {G : Type*} [Group G] [Solvable G]
    (H : Subgroup G) [h : H.Normal] :
  Solvable (G / H)
```

Opus judges both candidates equiv-if-fixed: the mathematical content matches gold, only the Mathlib API name is wrong. SAF and Sample-Filter likewise fail to elaborate at $K{=}3$ on this problem. One of 16 problems that remain TO across all four methods on DeepSeek V4-Pro $\times$ ProofNet$^{\#}$: a canonical instance of the elab signal naming a symptom without surfacing the cure.