What Reviewers Catch and Frontier Models Miss: Statement-Level Errors in Autoformalization

Anonymous authors
Paper under double-blind review

Abstract

A formal statement can typecheck and admit a proof while failing to express the intended mathematics. We study such statement-level errors where they naturally occur: in the human review process of a Lean 4 library of formalized open problems. From squash-merged pull-request history we recover 74 *natural negatives*, erroneous statements corrected during human review. Each is hand-labeled against a 14-class error taxonomy, using the reviewer's fix and comments as the reference. We then evaluate twelve frontier models. When shown the reviewer's correction and comments, the best model distinguishes semantic errors from non-semantic changes at $F_1 = 0.79$ ($\kappa = 0.73$); without the correction, no model exceeds $F_1 = 0.56$. In detection, misses concentrate in the classes that depend on Lean semantics: wrong carriers, definitional mismatches, junk-value traps. Under the same detection prompt, the strongest models achieve drastically higher scores on evaluator-injected errors from the same error classes, with F_1 reaching 1.00. Supplying the file's definitions, forcing a pass through the class list, or saying only that an error is present helps at most modestly; naming the reviewer-assigned error class dramatically improves class-level recall, but wrong hints also elicit substantial label compliance. The corpus of 5,726 reviewed (informal, formal) pairs, the labeled negatives, and the evaluation harness will be released.

1 Introduction

Autoformalization research measures whether models produce formal statements that typecheck or admit proofs (Wu et al., 2022; Azerbayev et al., 2023; Ying et al., 2024). Whether those statements say what the informal source says is harder to assess and is not established by typechecking or proof success alone (Zheng et al., 2022; Tsoukalas et al., 2024). The stakes are real: a machine-checked proof of the wrong statement certifies the wrong theorem, and recent audits have found such defects in widely used formal benchmarks (Ammanamanchi et al., 2026). Synthetic misalignments are one validation source for alignment scorers (Lu et al., 2024); human-annotated formalization attempts provide another (Poiroux et al., 2025; Zhang et al., 2026). We recover errors and corrections from a library's review history to test whether models that detect synthetic misalignments also detect naturally occurring mistakes.

We use reviewer corrections as reference judgments of statement correctness. Our source is `formal-conjectures` (Firsching et al., 2026; The Formal Conjectures Authors, 2025), a community Lean 4 (de Moura & Ullrich, 2021) library of formalized open problems in which each statement pairs a LaTeX docstring with a Lean declaration and undergoes human review before merging. Our mining procedure provides a template for studying other reviewed formal libraries, though this repository's review culture and conventions, such as the `answer()` marker, need not be representative.

We study a three-month window chosen to postdate the evaluated models' likely training cutoffs. Of the 502 pull requests merged during this period, 85 had statement changes during review. Review also missed errors: in the same period, 69 commits marked as fixes corrected 111 declarations that had already passed review. From the squash-merged history, we recover 74 *natural negatives*: statements from the initial pull-request versions that differ semantically from the versions approved during review, each paired with the reviewer's correction as a reference.[1]

[1]The corpus and evaluation harness are submitted as anonymised supplementary material; see Appendix A.

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

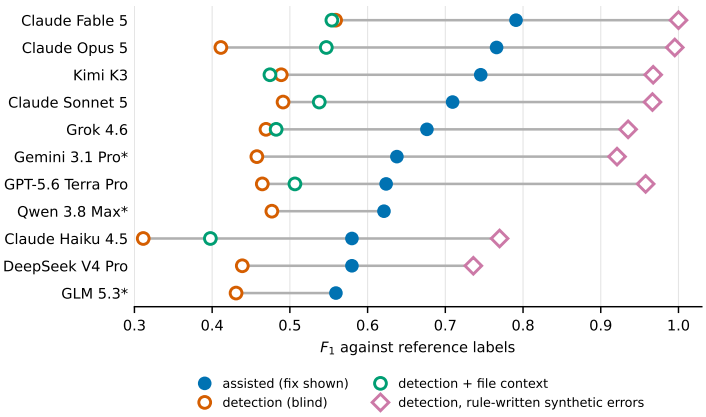

Figure 1: Per model: F_1 against reference labels with the reviewer's correction and comments (assisted, filled), without them (detection, open), with the file's definitions added (+context), and, under the same detection prompt, on rule-written synthetic errors (diamonds, Section 6). Models with complete synthetic evaluations detect the injected errors far better than the natural ones; diamonds are omitted for runs with fewer than 300 scored records. Asterisks indicate models with incomplete response coverage on the natural task. Muse Spark 1.2 is omitted because no responses completed within the output budget (n=0).

Our contributions are (1) a corpus of 5,726 reviewed statement pairs and 74 multi-label natural negatives with provenance, reviewer-approved corrections, and available comments; (2) a twelve-model evaluation showing an assisted-to-detection gap, with best F_1 scores of 0.79 and 0.56 and particularly low recall on Lean-semantics classes (Section 4); (3) diagnostic comparisons of file context, a checklist, and oracle hints, distinguishing unaided detection from agreement when labels are supplied (Section 5); and (4) comparisons with rule-written, model-written, and FormalAlign misalignments showing substantially higher scores on synthetic errors, including class-matched and matched-controls analyses (Section 6). Review history supports continued collection, while timestamps permit temporal contamination checks.

2 Related work

Autoformalization translates informal mathematics into formal statements (Wu et al., 2022), with resources including ProofNet, Lean Workbook, miniF2F, and PutnamBench (Azerbayev et al., 2023; Ying et al., 2024; Zheng et al., 2022; Tsoukalas et al., 2024). Draft, Sketch, and Prove also connects informal reasoning with formal proofs (Jiang et al., 2023). Typechecking and proof success alone do not establish faithfulness to the informal source.

Faithfulness evaluation includes reference-based equivalence checks (Poiroux et al., 2025), trained alignment scorers (Lu et al., 2024), and LLM judges (Zhang et al., 2026). ProofNetVerif supplies human-annotated formalization attempts (Poiroux et al., 2025), and Zhang et al. (2026) calibrate judges against human review. Our distinct source is library review history, including paired corrections and reviewer comments. Benchmark audits document statement defects (Ammanamanchi et al., 2026); review history additionally provides paired versions and correction context. This use of review follows work in software engineering (Bacchelli & Bird, 2013; Sadowski et al., 2018) and informs editorial applications such as Palomar (Tao, 2026). Appendix B expands these connections.

3 A corpus of reviewed statement pairs

Reviewed statement pairs. We parse every top-level declaration in the library and pair each documented declaration with its docstring. We strip proof bodies but retain definition bodies, which specify the objects being defined. At the frozen commit, the repository contains 6,949 declarations,

(a) Docstring: *Can the smallest modulus of a covering system be arbitrarily large?* (The known answer is no.)

```
theorem erdos_2 : answer(False) ↔                                    -- first push
    ∃ B : ℕ, ∀ S, IsDistinctCoveringSystem S → minModulus S ≤ B
theorem erdos_2 : answer(False) ↔                                    -- as merged
    ∀ B : ℕ, ∃ c : StrictCoveringSystem ℤ, ∀ i, ∃ m : ℕ,
      c.moduli i = Ideal.span {(m : ℤ)} ∧ B < m
```

(b) Docstring: *What is the smallest integer $m > 1$ such that $a(10^m)$ is nonzero?*

```
theorem conjecture1 : answer(sorry) =                                 -- first push
    sInf {m : ℕ | 1 < m ∧ a (10 ^ m) ≠ 0}
theorem conjecture1 : answer(sorry) =                                 -- as merged
    if h : ∃ m, 1 < m ∧ a (10 ^ m) ≠ 0
    then some (sInf {m | 1 < m ∧ a (10 ^ m) ≠ 0}) else none
```

Figure 2: Two statement-level errors recovered from review. (a) The `answer(False)` marker must pair with the proposition posed by the question: that the smallest modulus can be arbitrarily large. The initial version instead pairs it with boundedness, asserting that boundedness is false and thus reversing the known result. Labels: wrong-answer-marker, wrong-quantifier. (b) For naturals, `sInf` of the empty set is 0. If no qualifying m exists, the initial version therefore returns 0, which does not satisfy the question's constraint $m > 1$. The merged version distinguishes existence (`some`) from nonexistence (`none`); its binder m is inferred to be a natural from the type of `a`. Label: junk-value-trap. Both original statements typecheck, but human review identified semantic errors in each.

of which **5,726** have docstrings. Each record retains file path, line, commit, status, subject attributes, and docstring citations.

Natural negatives. Squash merging does not preserve individual pull-request commits in the main-branch history. We recover pre-review versions for the 502 pull requests squash-merged between 29 May and 31 August 2026. The window postdates the evaluated models' likely training cutoffs (Section 8). Of the 502 pull requests, 124 were force-pushed. Timeline queries recover pre-push commits for 18; for 101, those commits had been garbage-collected; the remaining 5 had no changes beyond the merge base. For each pull request, we fetch its branch head (`refs/pull/N/head`), identify the first commit after the merge base, and compare declaration statements with their merged versions in files modified by the squash commit. Where pre-push commits were garbage-collected, that first commit may already include the reviewer's fix, leaving no diff; errors corrected by force-push are therefore under-represented. Proof and docstring edits are not candidates; the docstring at merge is the informal statement, and the `informal-changed` label was assigned to no record. Only merged pull requests qualify, because our procedure requires a reviewer-approved version as the reference.

This procedure yields 374 candidates, reduced to 360 after deduplication and rename detection. One author annotated these candidates using a 14-class taxonomy, allowing multiple labels per record, and assigned an error *direction*: whether the initial statement was weaker than, stronger than, or incomparable to the informal intent. Annotation used the reviewer-approved corrections and 528 inline review comments. The final labeling pass hid model suggestions and preliminary labels, and both formalizations were elaborated in Lean where the difference was unclear. Of the 360 candidates, **74** contain semantic errors (Table 1); the remainder involve renames, API migrations, or stylistic changes, including first pushes that encode the docstring's content without the project's `answer()` marker. Figure 2 shows two examples of semantic errors.

4 Can models do the reviewer's job?

We evaluate twelve models: four Claude-family models accessed through the Anthropic API, and GPT-5.6 Terra Pro, Gemini 3.1 Pro, Grok 4.6, Kimi K3, DeepSeek V4 Pro, Qwen 3.8 Max, GLM 5.3, and Muse Spark 1.2 accessed through OpenRouter, each condition under a fixed prompt.

Assisted: the model receives the informal statement, the initial formalization, the reviewer-approved revision, and review comments, and is asked to explain any semantic error (excluding the 33 records that carried a triage flag in the payload does not materially change the scores; Appendix E). Withholding the comments (Table 9, appendix) lowers F_1 by 0.04–0.08 for all seven models in this ablation (paired intervals exclude zero for six), entirely through precision; recall stays at 0.95–0.97. Thus,

error class	meaning	records
wrong-type-or-domain	wrong carrier: $\mathbb{N}$ vs $\mathbb{Z}$ vs $\mathbb{Q}$ vs $\mathbb{R}$, meaning-changing coercion	20
definitional-mismatch	auxiliary definition does not capture the informal notion	15
missing-hypothesis	informal side condition absent (positivity, distinctness, $n \geq 1$)	12
wrong-answer-marker	`answer(True/False)` wrong, or on the wrong side of the $\leftrightarrow$	11
junk-value-trap	meaning shifts through Lean junk values: $x/0$, $\mathbb{N}$ subtraction, $\inf \emptyset$	11
wrong-quantifier	$\exists/\forall$ swapped, wrong scope or order	9
other	none of the above; explained in the record's note	6
wrong-connective	$\wedge/\vee/\rightarrow/\leftrightarrow$ misuse, misplaced negation	3
vacuous-or-trivial	trivially true or false (empty range, wrong index set)	2
extraneous-hypothesis	formal statement assumes more than the informal one	2
wrong-referent	points at the wrong declaration or constant	1
wrong-constant-or-bound	off-by-one, wrong constant, strict vs non-strict inequality	1

Table 1: Error-class frequencies among the 74 natural negatives. Each statement may receive multiple labels (93 assignments over 74 records); the taxonomy's two non-error labels are omitted (Appendix D). Wrong carriers, definitional mismatches, and missing hypotheses are the most frequent classes. A first push that encodes the docstring's content without the project's `answer()` marker is labeled non-semantic.

	assisted				detection			
model	P	R	F1	κ	P	R	F1	κ
Claude Haiku 4.5	0.42	0.93	0.58	0.41	0.24	0.45	0.31	0.06
Claude Sonnet 5	0.56	0.97	0.71	0.61	0.70	0.38	0.49	0.41
Claude Opus 5	0.63	0.97	0.77	0.69	0.67	0.30	0.41	0.33
Claude Fable 5	0.68	0.95	0.79	0.73	0.61	0.51	0.56	0.46
GPT-5.6 Terra Pro	0.47	0.92	0.62	0.48	0.44	0.49	0.46	0.32
Gemini 3.1 Pro$^{(342–351)}$	0.48	0.96	0.64	0.51	0.44	0.47	0.46	0.30
Kimi K3	0.66	0.85	0.75	0.67	0.54	0.45	0.49	0.37
Muse Spark 1.2	–	–	–	–	–	–	–	–
DeepSeek V4 Pro	0.42	0.93	0.58	0.41	0.35	0.58	0.44	0.25
Qwen 3.8 Max$^{(328–335)}$	0.46	0.97	0.62	0.49	0.42	0.55	0.48	0.33
Grok 4.6	0.53	0.95	0.68	0.56	0.40	0.57	0.47	0.30
GLM 5.3$^{(319–328)}$	0.40	0.92	0.56	0.39	0.43	0.43	0.43	0.29

Table 2: Agreement with human labels in the assisted and detection conditions, computed on each condition's scoreable records. Superscripts give the range of scoreable counts when either is below 360; missing outputs result from responses reaching the output-token limit or omitting records from the returned JSON. Muse Spark 1.2 produced no scoreable output within the 32k-token output budget. Detection F_1 is lower than assisted F_1 for every model with scores in both conditions.

models retain high binary recall with the correction but no comments, while comments help them avoid flagging non-semantic edits. **Detection:** the model receives only the informal statement and candidate formalization and is asked to identify any semantic error. The prompts require a JSON-only response with a one-sentence rationale per record and no visible scratchpad; requests use provider default reasoning settings. We present records in fixed batches of 12, with identical batch composition across models. We measure agreement with human labels using precision, recall, F_1, and Cohen's κ for the binary semantic/non-semantic distinction (Table 2); a prediction counts as semantic when its class set contains any class other than `non-semantic`. We also measure exact class-set agreement; scoring compares labels, not rationales.

Findings. *(i)* Detection F_1 is lower than assisted F_1 for every model with scores in both conditions (Figure 1). The highest assisted F_1 is 0.79 (Claude Fable 5; 95% bootstrap confidence interval $[0.72, 0.85]$), whereas the highest detection F_1 is 0.56 (also Claude Fable 5; $[0.46, 0.66]$).

(ii) Class-level results reveal consistently low recall for errors involving Lean semantics (Table 3). The highest counts across models, taken separately for each class, are just 5 of 20 wrong-type-or-domain errors, 4 of 15 definitional mismatches, 7 of 11 junk-value traps, and 6 of 11 answer-marker errors; no model assigns the gold `other` label to any of its 6 examples. We pool these four classes

error class (n)	Claude Haiku 4.5	Claude Sonnet 5	Claude Opus 5	Claude Fable 5	GPT-5.6 Terra Pro	Gemini 3.1 Pro*	Kimi K3	DeepSeek V4 Pro	Qwen 3.8 Max*	Grok 4.6	GLM 5.3*
wrong-type-or-domain (20)	0	2	2	2	4	4	2	1	4/15	5	2/16
definitional-mismatch (15)	0	2	1	4	3	2	3	1	3/14	3	2/14
missing-hypothesis (12)	5	6	5	8	7	5	4	5	8/11	5	3/9
junk-value-trap (11)	2	3	3	6	5	3	5	2	7	3	4/10
wrong-answer-marker (11)	2	1	2	5	3	5	4	6	5	6	3/8
wrong-quantifier (9)	2	3	2	6	4	3	2	2	3/7	5	3/8
other (6)	0	0	0	0	0	0	0	0	0	0	0
wrong-connective (3)	3	1	1	1	2	1	1	2	2	1	1/2
extraneous-hypothesis (2)	1	1	1	0	1	1	0	0	1	2	2
vacuous-or-trivial (2)	0	2	2	2	1	1	2	0	2	2	1
wrong-constant-or-bound (1)	0	1	1	1	1	1	1	0	1	1	0/0
wrong-referent (1)	1	1	1	1	1	1	1	1	1	1	1

Table 3: Class-level detection results. Cells count errors correctly assigned to each class; row labels give the total number of gold examples in that class. For models with incomplete response coverage (asterisked), a cell reads found/answered whenever the model answered fewer than the row's gold examples. Correct-class recall is especially low for wrong types or domains and definitional mismatches. Results on wrong connectives, vacuous statements, extraneous hypotheses, wrong constants, and wrong referents are based on only 1–3 examples per class.

for group-level analysis, yielding 57 gold label assignments across potentially overlapping records. Because the answer marker is a project convention rather than general Lean semantics, we also pool the remaining three classes alone (46 labels): models with full coverage recover 2–12 of the 46, so the deficit is not confined to the project-specific class. Results for classes with few examples are descriptive; Table 13 in the appendix reports confidence intervals for individual classes and the pooled group for selected models.

5 Context, checklists, and supplied labels

We test whether context or explicit guidance helps models find the Lean-semantics errors they miss.

Setup. The context-augmented condition adds a field to the detection prompt containing the file's imports, `open` declarations, and auxiliary declarations referenced by the statement. References are resolved by name through two levels of dependencies. All context is extracted from the file at the record's pre-review commit to avoid introducing the reviewer's correction. We evaluate seven models using the frozen detection prompt with one added paragraph describing the context field. That paragraph also directs the model to reason about how the statement elaborates (what a definition unfolds to, which carrier a coercion lands in, whether an operation can reach a junk value), so the condition combines local context with hazard-oriented guidance; core and Mathlib definitions are not supplied. We do not tune the prompt for this condition, so the results assess this specific context intervention rather than the usefulness of context in general.

Results. Context helps modestly. Changes in F_1 range from -0.01 to $+0.14$ (Table 6, appendix); paired bootstrap intervals exclude zero only for Claude Opus 5 and, marginally, Claude Haiku 4.5. No context-augmented score exceeds the highest unaugmented detection score, and pooled recall on the four Lean-semantics classes rises for every model, from 4–17 to 6–20 of 57 labels, but stays well below the class-hint condition. Paired comparisons on identical records (Table 7) reveal different responses across models. Claude Opus 5 is the only model with a statistically significant increase in recall (13 errors newly found, none lost), accompanied by nearly as many new false positives as newly detected errors. Claude Haiku 4.5 removes 47 false flags while gaining 12 errors and losing 10; Kimi K3 adds 27 false flags. These tests are unadjusted for multiplicity; under a Bonferroni correction for the 14 tests ($\alpha = 0.0036$), only Opus's recall gain and Haiku's reduction in false flags

	blind		error-present		class named	
model	class set	Lean-sem.	class set	Lean-sem.	class set	Lean-sem.
Claude Haiku 4.5$^{(359)}$	0.52	4/57	0.82	17/57	0.92	48/57
Claude Sonnet 5$^{(293)}$	0.79	5/45	0.84	14/45	0.98	44/45
Claude Opus 5$^{(285)}$	0.83	4/35	0.87	17/35	0.94	34/35
Claude Fable 5$^{(253)}$	0.78	12/43	0.85	17/43	0.92	41/43
GPT-5.6 Terra Pro	0.71	15/57	0.82	26/57	0.92	51/57
Grok 4.6	0.66	17/57	0.83	26/57	0.92	50/57

Table 4: Unassisted detection compared with two oracle hints: the binary semantic/non-semantic label and the reviewer-assigned error class. For each model, all three conditions are scored on their shared set of answered records; superscripts give its size. Both hints disclose the binary label, so binary detection metrics are omitted. We report exact class-set agreement and pooled Lean-semantics recall (found / gold). Class hints yield larger gains than binary-label hints, but agreement under supplied class labels does not measure independent detection. Record subsets can differ across models, so rows should not be interpreted as a model ranking. Kimi K3 is omitted: its existence-hint run answered only 143 records, leaving 2 semantic records in the shared set.

remain significant. Overall, supplying definitions raises Lean-semantics recall by a few labels per model without closing the gap.

Two oracles. The context results leave open whether models can verify these errors when given more specific guidance. We compare two oracle conditions. The first supplies the reviewer-assigned error class or classes (or "non-semantic" for records labeled as such) and asks the model to substantiate the supplied labels. The second reveals only whether a semantic error is present, without identifying its class. Both disclose the gold binary label, so we examine class-level recall and exact class-set agreement, which measure what naming the class adds beyond revealing that an error exists.

Results. Naming the error class improves class-level recall substantially more than revealing only that an error exists (Tables 4 and 11). For Haiku, GPT, and Grok, whose shared records cover all 57 Lean-semantics labels, the number returned correctly rises from 4–17 of 57 under unassisted detection to 17–26 when error presence is revealed, and to 48–51 when the class is supplied. For the four models in Table 11, class-hinted recall reaches 10–11 of 11 junk-value labels, 13–15 of 15 definitional mismatches, all 11 answer-marker labels, and 15–20 of 20 carrier errors. These are agreement scores under supplied labels, not independent evidence that the explanations verify the errors. Hints provide information as well as guidance, so the experiment does not isolate attention from capability or validate the annotations.

Distractor control. To test whether models also comply with incorrect class hints, we rerun the hint prompt with a wrong hint on every record: a class the reviewer did not assign on each error, and an asserted error class on each control (Table 10, appendix). Compliance is substantial and model-dependent. Given a wrong class, Claude Haiku 4.5 echoes it on 55 of 74 errors and finds a gold class on 7; Claude Sonnet 5 echoes it on 50 of 74 and finds a gold class on 11; Kimi K3 echoes it on 45 of 73 and finds a gold class on 18. GPT-5.6, Grok 4.6, Claude Opus 5, and Claude Fable 5 echo the wrong class on 27–41 errors and still find a gold class on 25–26 of 70–74. On controls with a falsely asserted error, GPT-5.6 and Grok 4.6, which answered all 286, return non-semantic on 64% and 58%; Claude Fable 5 does so on 73 of the 90 controls its batches completed and Claude Sonnet 5 on 33 of 243, but those subsets are not random samples, so their rates are indicative only. These results show that high class-hinted recall can coexist with substantial acceptance of incorrect labels; it should not be interpreted as verified-error recall.

A non-oracle intervention. The unassisted detection prompt already lists every error class. The checklist condition additionally requires an explicit verdict for each class except the catch-all `other` on each record, without supplying any gold labels. Classes marked true form the predicted label set. Changes in F_1 range from -0.09 to $+0.12$ (Table 18), with intervals excluding zero only for Haiku's gain and Sonnet's loss. The number of correctly identified Lean-semantics labels ranges from 8–20 of 57 under the checklist, compared with 44–56 under the class hint. Requiring a systematic pass through the specified classes therefore does not reproduce the gains from naming the class.

	natural F1		synthetic, rule-written				synthetic, LLM-written				FormalAlign's	
model	all	matched	P	R	F1	Δ	P	R	F1	Δ	F1	Δ
Claude Haiku 4.5	0.31	0.29	0.63	0.98	0.77	+0.48	0.51	1.00	0.68	+0.39	0.65	+0.36
Claude Sonnet 5	0.49	0.45	0.96	0.97	0.97	+0.51	0.90	0.97	0.94	+0.48	0.84	+0.39
Claude Opus 5	0.41	0.40	0.99	1.00	1.00	+0.60	0.99	1.00	1.00	+0.60	0.94	+0.54
Claude Fable 5	0.56	0.52	1.00	1.00	1.00	+0.48	0.98	1.00	0.99	+0.47	0.85	+0.34
GPT-5.6 Terra Pro	0.46	0.46	0.94	0.98	0.96	+0.50	0.89	0.99	0.94	+0.48	0.77	+0.31
Gemini 3.1 Pro$^{(84)}$	0.46	0.44	0.85	1.00	0.92	+0.49	0.97	1.00	0.98	+0.55	–	–
Kimi K3	0.49	0.43	0.94	1.00	0.97	+0.54	0.88	1.00	0.93	+0.51	0.77	+0.35
DeepSeek V4 Pro$^{(229)}$	0.44	0.39	0.60	0.95	0.74	+0.35	0.60	0.84	0.70	+0.31	–	–
Grok 4.6	0.47	0.40	0.90	0.97	0.94	+0.54	0.89	1.00	0.94	+0.54	0.77	+0.37
GLM 5.3$^{(37)}$	0.43	0.39	0.77	0.91	0.83	+0.44	–	–	–	–	–	–

Table 5: Detection performance on natural errors and three synthetic sets: rule-written, model-written, and FormalAlign's released misalignments. The natural "matched" baseline retains controls and errors whose labels all occur in the rule arm; all deltas use this baseline. The other arms are not separately class-matched. For models with incomplete coverage on the rule-written or model-written arm (359 and 360 records respectively), the superscript gives the smaller of the two scored counts. Missing records were not retried on these arms; rows with fewer than 300 scored records (Gemini 3.1 Pro, DeepSeek V4 Pro, GLM 5.3) are shown for completeness and excluded from the ranges.

6 Natural versus synthetic negatives

FormalAlign evaluates alignment on evaluator-constructed misalignments (Lu et al., 2024); we evaluate our model panel under the same detection prompt on synthetic and natural errors.

Rule-written arm. We construct each synthetic negative by applying a single taxonomy-based text transformation to a reviewed statement outside the natural candidate set. Transformations include quantifier swaps, hypothesis deletion or addition, carrier substitution, connective or answer-marker changes, off-by-one bounds, unsatisfiable hypotheses, and changes that expose Lean junk values. Each perturbed declaration is re-elaborated in its original file and retained only if it typechecks.

Among classes covered by the rules, we followed the class distribution of a preliminary labeling pass (Table 17, appendix), which differs from the final distribution. The arm approximately matches the preliminary natural task's ratio of errors to controls, with 104 injected errors and 255 unperturbed reviewed statements as controls (Table 17, appendix). Our rule set does not cover wrong referents, definitional mismatches, or the catch-all class `other`. We therefore also report natural-error performance on a class-matched subset, the records whose classes the rules cover (matching class support, not frequency or difficulty).

Model-written arm. To test whether the results depend on the simplicity of rule-based perturbations, we ask Claude Opus 5 (medium reasoning effort) to modify a reviewed statement by introducing one error from a target class. The prompt requests a plausible initial submission by a competent formalizer: a statement that still elaborates and contains a subtle semantic error, preferably one arising from Lean's interpretation of the statement (prompt and generation parameters in Appendix E). We apply the same typechecking filter as in the rule-written arm; 98% of generated outputs pass; they are not hand-verified, so their classes are intended rather than established. The arm reuses the rule-written controls, draws sources disjoint from the natural candidates and the rule-written sources, and matches class proportions across ten classes: 105 perturbed records plus the 255 shared controls. Because the generator is also an evaluated model, we report its score on this arm but exclude it from our interpretation.

Natural controls underwent non-semantic changes during review, whereas synthetic controls are unperturbed reviewed statements; the matched-controls experiment below addresses this difference.

Results. Models perform substantially better on injected errors than on the rule-class-matched natural baseline (Table 5). On the rule-written arm, F_1 increases by 0.48–0.60 across the models with full coverage. Excluding the generator, the five highest-scoring models on this arm achieve F_1 of 0.94–1.00, compared with 0.40–0.52 on natural errors of matched classes: a factor of 1.9–2.3. The same five achieve 0.93–0.99 on model-written errors, with gains of 0.47–0.54 over the matched natural baseline. The gap persists for model-written errors. For Grok 4.6, the lowest-scoring of these

five on the rule-written arm, the 95% bootstrap confidence interval is $[0.90, 0.97]$, compared with $[0.30, 0.50]$ on matched natural records.

We also evaluate a third set containing 105 misaligned and 255 aligned candidates from FormalAlign's released MiniF2F evaluation data (Table 16, appendix). The six models comprising these five and Opus identify 93–100% of the labeled misalignments (Table 16 also lists Haiku 4.5). Their F_1 scores of 0.77–0.94 are limited primarily by precision against the released labels. Of the 255 candidates labeled aligned, 37 are flagged by at least four of the six models. We audited all 37. As released, 32 are not valid aligned pairs: 16 are statements commented out in the release, inherited from the MiniF2F Lean 4 port, and assert nothing; the others include a fourth root written as $x^{1/4}$ whose exponent elaborates as natural-number division (so the power equals 1), natural-number subtraction in hypotheses, a swapped modulus and residue, and three statements we proved false in Lean. Judging the commented-out statements by their content instead, 21 of the 37 are defects and 16 are model false alarms. Only flagged candidates were audited, so unflagged defects remain unknown and the adjusted labels are partial. Against them the six models' precision is 0.76–0.97 and F_1 0.85–0.94 (Table 16); counting commented-out references as defects gives 0.82–0.99 and 0.90–0.96.

Table 12 (appendix) compares class-level recall across natural errors and both synthetic arms for four selected models; these count errors identified with the correct class, a stricter measure than the binary detection of Table 5. Synthetic answer-marker errors are identified every time, whereas natural examples are identified in at most about half of cases. Synthetic carrier errors are identified nearly every time and their natural counterparts in at most a quarter.

Junk-value traps remain challenging in the rule-written arm: the four models fail to assign the junk-value class to between 4 and 6 of 17 examples. In the model-written arm, however, three of the four models assign that class to all 17 examples. These errors involve reasoning about the values Lean assigns to expressions in boundary cases, but the results show that their difficulty also depends on the examples and generation method.

What the gap measures. The difference in control construction does not fully explain the performance gap. A matched-controls experiment injects rule-based errors into approved statements whose initial versions were labeled non-semantic in the preliminary pass. Controls use initial submissions still labeled non-semantic in the final pass (Table 15, appendix). Every evaluated model achieves 98–100% recall on these injected errors, compared with 30–57% recall on natural errors. Only 3 of the 41 injected errors fall in the junk-value or carrier classes, but natural detection recall reweighted to the injected class mix is 50–71%, so class composition explains only part of the difference.

Together with the class-level comparisons, this result shows that high recall on injected errors need not translate into high recall on errors recovered from human review. One possible explanation is that synthetic perturbations introduce local anomalies, whereas natural errors can reflect a coherent but mistaken interpretation of the informal statement. The experiments establish a difference in detection difficulty, but do not isolate the mechanism behind it.

7 Discussion

Three results together characterize the detection gap in our setting. Models flag the natural errors when shown the correction and largely fail to find them unaided; the misses concentrate in classes that turn on Lean semantics. Supplying the file's definitions or forcing a pass through the class list helps at most modestly. Naming the class substantially increases label agreement, but the distractor control shows that every tested model also accepts incorrect hints. The hint results therefore do not establish that verifying a named error is easy. The strongest models detect nearly all injected errors across the three synthetic sets, while performing substantially worse on review-derived errors, including within shared classes. The comparisons establish these differences without isolating their mechanism; surface anomaly in injected errors is one explanation, not an established cause.

Formal mathematics is a controlled instance of a general evaluation problem: negatives made by perturbing correct instances can carry artifacts that natural mistakes lack, and a detector validated on them may be scored on those artifacts. Review history is what makes the natural population observable; wherever caught-and-corrected mistakes are recorded, the same comparison can be run.

The findings motivate tools that supply concrete evidence about semantic hazards. A static lint targeting division, natural-number subtraction, and related operators flags 9 of 11 junk-value traps, compared with 7 identified by the best unassisted model for that class, but also flags roughly one third of reviewed statements (Table 14); it fires on 122 of the 360 records to find those 9 traps. These are candidate hazards, not verified errors. The hazard is visible to a pattern; the tested detectors do not reliably distinguish actual defects from candidate hazards, and that is the step a tool would have to support. A model with a Lean REPL could probe expressions at concrete arguments; we do not test such a loop. Answer-marker misses may require mathematical knowledge or better interpretation of marker placement and polarity; the experiments do not separate these explanations.

Our results bear on LLM editorial review such as Palomar (Tao, 2026) without evaluating its deployed procedure: the highest detection precision, $P = 0.70$, suggests potential for triage, but low recall limits sole screening, and deployment precision depends on the target error prevalence.

8 Limitations

All data come from one repository and one review culture. Natural negatives include only errors that reviewers identified and authors corrected before merging. The 111 declarations corrected after merge demonstrate that errors can survive review, but do not establish an overall miss rate. Pull requests closed without merging are excluded because they lack a merged reference statement. Of 243 such pull requests in the study window, 18 received review. Errors corrected by force-push are under-represented (Section 3). These counts describe the excluded population but do not establish the extent of selection bias; surviving errors may skew toward local, readily repaired mistakes. The corpus does not support an estimate of the overall statement-error rate.

All labels come from one annotator; an initial pass with model suggestions was discarded for the blind pass. No independent annotator has labeled the corpus, so label reliability is not established; that remains the missing check.

Class-hinted responses include rationales, but scoring checks label agreement, not rationale correctness; the distractor control quantifies acceptance of incorrect labels but does not establish the correctness of individual rationales. All models are evaluated against the same labels, yet annotation errors could still affect models and conditions differently; the bootstrap intervals of Section 4 quantify sampling uncertainty, not systematic annotation error. Records cluster by pull request (two contribute 35 each); intervals that resample pull requests are $[0.61, 0.89]$ for the best assisted F_1 and $[0.46, 0.65]$ for the best detection F_1, and excluding the three largest pull requests (95 records) gives 0.73 and 0.57.

Records are evaluated in batches of 12; a 50-record single-request rerun of the top detector agrees with its batched judgments on 84% of records at the same accuracy, which does not exclude batching effects elsewhere. Missing records were rerun one per request in the assisted and detection conditions, in the two hint conditions on records whose labels changed, and for the errors of the distractor condition; the synthetic arms and the distractor controls were not retried, and rows with small scored counts there are not comparable to full-coverage rows. On the records answered by all models with scoreable outputs (297 assisted, 284 detection), each model's F_1 moves by at most 0.05, and Claude Fable 5 remains the top model in both conditions. Each cell is a single run; run-to-run variability is not characterized.

The source library is public, so training-data contamination cannot be excluded; recent commit dates permit temporal checks but do not establish exposure.

Withholding the reviewers' comments from the assisted condition lowers F_1 by 0.04–0.08 without reducing recall (Table 9), so high assisted recall does not require the comments, although they improve precision. The primary detection condition supplies only the docstring and formal statement. The context experiment adds definitions from the record's file, not from core or Mathlib, under one fixed prompt. Other context presentations, reasoning budgets, and tool access remain untested, and no symbolic equivalence checker or trained alignment scorer was evaluated (Appendix B). The class-hint condition supplies both error presence and class information; its gains over the existence-only condition do not isolate attention from verification capability. The synthetic comparisons do not match class frequencies or prevalence exactly (Section 6).

Reproducibility statement

Reported quantitative results are computed by scripts from the corpus, the blind human labels, and saved model predictions. The artifact includes the extraction and mining code, evaluation harness, and table-generation scripts. The corpus is frozen at a specified upstream commit, evaluation prompts are included verbatim, and each record retains its source commit and timestamp. Appendix A documents the evaluation window, model versions, and cost. The corpus, labels, generated synthetic sets, model predictions, and code are included as anonymised supplementary material.

AI use statement

Claude Fable 5 assisted with drafting, editing, related-work discovery, tooling, and analysis code. Model-generated label suggestions were used in a preliminary labeling pass that was discarded; the final labels were made with neither the suggestions nor the preliminary labels visible. The author verified the claims, numerical results, and citations and read every cited work. Fable is also evaluated and has the highest assisted score (Section 8). Claude Opus 5 generated the model-written synthetic negatives of Section 6; they were typecheck-gated but not hand-verified, and Opus is also an evaluated model. GPT-6 Astra and Gemini 3.1 Pro assisted with proofreading, manuscript review, interpretation of results, and methodological qualifications. Google's Paper Assistant Tool provided feedback on a draft of the manuscript. No model was used to prove mathematical claims.

References

Pawan Sasanka Ammanamanchi, Siddharth Bhat, and Stella Biderman. Faults in our formal benchmarking: Dataset defects and evaluation failures in Lean theorem proving, 2026. arXiv:2606.29493.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics, 2023. arXiv:2302.12433.

Alberto Bacchelli and Christian Bird. Expectations, outcomes, and challenges of modern code review. In *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, 2013.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *CADE*, pp. 625–635, 2021.

Moritz Firsching, Paul Lezeau, Salvatore Mercuri, Miklós Z. Horváth, Yaël Dillies, Calle Sönne, Eric Wieser, Fred Zhang, Thomas Hubert, Blaise Agüera y Arcas, and Pushmeet Kohli. Formal conjectures: An open and evolving benchmark for verified discovery in mathematics, 2026. arXiv:2605.13171.

Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *ICLR*, 2023.

Jianqiao Lu, Yingjia Wan, Yinya Huang, Jing Xiong, Zhengying Liu, and Zhijiang Guo. FormalAlign: Automated alignment evaluation for autoformalization, 2024. ICLR 2025; arXiv:2410.10135.

Auguste Poiroux, Gail Weiss, Viktor Kunčak, and Antoine Bosselut. Reliable evaluation and benchmarks for statement autoformalization. In *EMNLP*, 2025.

Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. Modern code review: a case study at Google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2018.

Terence Tao. Palomar: a registry of Lean verified mathematics, 2026. Registry and announcement at `https://palomar-registry.org`.

The Formal Conjectures Authors. The Formal Conjectures Repository. `https://github.com/google-deepmind/formal-conjectures`, 2025.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating neural theorem-provers on the Putnam mathematical competition. In *NeurIPS Datasets and Benchmarks*, 2024.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *NeurIPS*, 2022.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Zheng Yuan, Dahua Lin, and Kai Chen. Lean workbook: A large-scale Lean problem set formalized from natural language math problems, 2024. arXiv:2406.03847.

Ke Zhang, Patricio Gallardo Candela, Sudhir Murthy, Yi Xie, Zhi Wang, and Maziar Raissi. Beyond compilation: Evaluating faithful natural-language-to-Lean statement formalization, 2026. arXiv:2606.31002.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: a cross-system benchmark for formal Olympiad-level mathematics, 2022. ICLR.

Jin Peng Zhou, Charles Staats, Wenda Li, Christian Szegedy, Kilian Q. Weinberger, and Yuhuai Wu. Don't trust: Verify — grounding LLM quantitative reasoning with autoformalization. In *ICLR*, 2024.

A Reproducibility

All model evaluations were run in August–September 2026. The artifact's `scripts/multi_model_compare.py` specifies the requested API model identifiers and request parameters. These identifiers do not guarantee immutable provider-side model versions. Extraction, mining, triage, synthetic-negative generation, evaluation, and table generation are scripted; human annotation uses a labeling interface. Exact prompts are preserved in the artifact's `docs/prompts.md`, with typeset versions below (Appendix E). Total model spend across every condition, including failed and partial runs, was $636 ($361 Anthropic, $275 OpenRouter) for roughly 32,000 scored judgments. Artifact links are withheld for anonymity; the artifact is attached as anonymised supplementary material.

B Extended related work

Autoformalization and what it measures. Wu et al. (2022) demonstrated the feasibility of using large language models to translate informal mathematics into a proof assistant's language. Related benchmarks and training resources include ProofNet (Azerbayev et al., 2023), Lean Workbook (Ying et al., 2024), and the competition-style suites miniF2F (Zheng et al., 2022) and PutnamBench (Tsoukalas et al., 2024). Pipelines such as Draft, Sketch, and Prove (Jiang et al., 2023) connect informal reasoning with formal proof; Don't Trust: Verify uses autoformalization and theorem proving to check reasoning steps (Zhou et al., 2024). Typechecking and proof success provide useful measures of formal validity, but neither alone establishes that a formal statement faithfully represents its informal source. Our corpus draws on a library (Firsching et al., 2026; The Formal Conjectures Authors, 2025) whose review process explicitly addresses this question, providing human judgments and corrections of statement-level errors.

Faithfulness evaluation. Approaches to evaluating whether a formal statement faithfully represents its informal source include symbolic equivalence checking, trained alignment scorers, and LLM judges. Symbolic equivalence checking (Poiroux et al., 2025) seeks to prove a candidate equivalent to a reference formalization; its interpretation as evidence of faithfulness depends on the reference being faithful. We evaluated no symbolic equivalence checker and no specialized alignment scorer, so whether the best detection scores we observe are specific to prompted general-purpose models remains open. An equivalence baseline would be feasible on part of our corpus: two propositions can always be related by $\leftrightarrow$, and equivalent open conjectures can sometimes be related without proving

either. It would be harder where the correction changes a definition's type (the carrier, the codomain, or the shape of the answer, as in Figure 2b and Appendix F), where direct equality is not available and a bridging statement must be chosen, and automated proofs of either direction are rarely available for open conjectures. We leave this comparison to future work. Trained alignment scorers such as FormalAlign (Lu et al., 2024) learn an alignment signal from paired data. LLM judges (Zhang et al., 2026) assess agreement between informal and formal statements directly. FormalAlign evaluates on constructed misalignments. Other evaluations use human annotations: ProofNetVerif contains 3,752 annotated formalization attempts (Poiroux et al., 2025), and Zhang et al. (2026) calibrate LLM judgments on independently audited outputs. Our contribution is specifically review-derived errors with corrections, not the first source of human-labeled misalignments. Section 6 compares detection of injected errors with detection of review-derived errors. Under the same detection prompt, our evaluated models achieve high recall on injected errors, including FormalAlign's released misalignments, but substantially lower scores on natural errors of matched classes. Our contribution is a review-derived validation set for assessing detectors on naturally occurring statement errors.

Defects in formal benchmarks. Ammanamanchi et al. (2026) audit widely used formal benchmarks and identify statement-level defects in published evaluation sets. Their findings show that such errors can survive the processes used to construct formal benchmarks. Our work complements these audits by recovering errors documented during review, together with their corrected versions and available reviewer comments. These records provide paired examples of erroneous and corrected statements for evaluating error detection.

Editorial review in formal-mathematics infrastructure. The Palomar registry (Tao, 2026) includes an editorial step in which a language model assesses whether a formal statement faithfully represents its informal claim. Our detection experiments provide evidence relevant to this use of LLM judges: given an informal–formal statement pair without the reviewer's correction, how often can a model identify an error documented during human review?

Review history as supervision. Outside formal mathematics, the software-engineering literature has studied code review as a source of evidence about what reviewers catch and why (Bacchelli & Bird, 2013; Sadowski et al., 2018). We apply this perspective to formal statements, using reviewer corrections and comments to identify and label semantic errors. Our mining procedure recovers pre-review versions from a squash-merging repository, whose main-branch history does not preserve the individual pull-request commits. Recovery uses pull-request references and the platform's event log.

C Extended results

Supporting tables for Sections 5 and 6, followed by the taxonomy, the frozen prompts, and sample records.

	detection	context-augmented				Lean-semantics recall	
model	F1	P	R	F1	ΔF1	detection	context-augmented
Claude Haiku 4.5	0.31	0.34	0.47	0.40	+0.09	4/57	6/57
Claude Sonnet 5	0.49	0.71	0.43	0.54	+0.05	8/57	11/57
Claude Opus 5	0.41	0.65	0.47	0.55	+0.14	8/57	16/57
Claude Fable 5	0.56	0.55	0.55	0.55	−0.00	17/57	19/57
GPT-5.6 Terra Pro	0.46	0.49	0.53	0.51	+0.04	15/57	18/57
Kimi K3	0.49	0.45	0.50	0.47	−0.01	14/57	17/57
Grok 4.6	0.47	0.43	0.55	0.48	+0.01	17/57	20/57

Table 6: Context-augmented detection against the blind baseline, with recall on the four Lean-semantics classes pooled (found / gold). Seven models, full coverage, one run each. Paired bootstrap 95% intervals on ΔF_1 (2,000 resamples of records) exclude zero for Claude Opus 5 ($[+0.05, +0.23]$) and marginally for Claude Haiku 4.5 ($[0.00, +0.17]$) only.

	semantic records (74)			non-semantic records (286)		
model	found only w/ context	lost w/ context	p	new false flags	flags removed	p
Claude Haiku 4.5	12	10	0.832	9	47	<0.001
Claude Sonnet 5	10	6	0.454	4	3	1.000
Claude Opus 5	13	0	<0.001	9	1	0.021
Claude Fable 5	6	3	0.508	15	6	0.078
GPT-5.6 Terra Pro	9	6	0.607	13	17	0.585
Kimi K3	10	6	0.454	27	10	0.008
Grok 4.6	4	5	1.000	15	23	0.256

Table 7: Context-augmented against blind detection on the same records: flips in each direction and exact two-sided McNemar p-values, without adjustment for multiple comparisons. With a Bonferroni correction for the 14 tests ($\alpha = 0.0036$), only Opus's recall gain and Haiku's reduction in false flags remain significant.

	detection	oracle class hint				Lean-semantics recall	
model	F1	P	R	F1	ΔF1	detection	oracle class hint
Claude Haiku 4.5$^{(359)}$	0.31	0.90	0.97	0.94	+0.62	4/57	48/57
Claude Sonnet 5	0.49	0.96	0.99	0.97	+0.48	8/57	56/57
Claude Opus 5$^{(346)}$	0.41	0.97	0.94	0.95	+0.54	8/57	44/46
Claude Fable 5	0.56	0.92	0.93	0.93	+0.37	17/57	55/57
GPT-5.6 Terra Pro	0.46	0.87	0.88	0.87	+0.41	15/57	51/57
Kimi K3$^{(357)}$	0.49	0.96	0.89	0.92	+0.43	14/57	51/56
Grok 4.6	0.47	0.86	0.85	0.86	+0.39	17/57	50/57

Table 8: Unassisted detection compared with an oracle class hint. The hint supplies the gold class set, including the binary label. Oracle precision, recall, and F_1 therefore do not measure independent detection; pooled class-level recall measures agreement under supplied labels. Superscripts give records with scoreable output when below 360.

	assisted	answer key only				Lean-semantics recall	
model	F1	P	R	F1	ΔF1	assisted	answer key only
Claude Haiku 4.5$^{(359)}$	0.58	0.38	0.97	0.54	−0.04	27/57	22/57
Claude Sonnet 5	0.71	0.48	0.97	0.64	−0.07	38/57	37/57
Claude Opus 5	0.77	0.56	0.97	0.71	−0.05	40/57	41/57
Claude Fable 5	0.79	0.59	0.95	0.73	−0.07	41/57	35/57
GPT-5.6 Terra Pro	0.62	0.38	0.96	0.54	−0.08	40/57	43/57
Kimi K3	0.75	0.54	0.96	0.69	−0.06	32/57	35/57
Grok 4.6	0.68	0.45	0.95	0.61	−0.07	44/57	38/57

Table 9: Assisted condition with the reviewers' comments (assisted) and with the corrected statement alone (answer key only), with pooled Lean-semantics recall (found / gold). The difference isolates what the reviewers' prose adds beyond the formal diff: recall does not decrease without the comments, and the F_1 losses come from precision on the non-semantic controls. Seven models, one run each.

	controls	errors, wrong class hinted		errors, gold class hinted
model	hint rejected	wrong class echoed	a gold class found	a gold class found
Claude Haiku 4.5	70/283	55/74	7/74	66/74
Claude Sonnet 5	33/243	50/74	11/74	72/74
Claude Opus 5	100/177	41/70	25/70	60/64
Claude Fable 5	73/90	40/73	26/73	67/74
GPT-5.6 Terra Pro	184/286	27/74	26/74	61/74
Kimi K3	30/67	45/73	18/73	63/72
Grok 4.6	165/286	39/74	26/74	62/74

Table 10: Distractor control for the class hint. The hint prompt is unchanged, but every record receives a class the reviewer did not assign (deterministic per record): on the 286 controls the hint asserts an error, on the 74 errors it names the wrong class. Columns count controls on which the model rejected the hint and returned non-semantic, errors on which the wrong class was echoed, and errors on which a gold class was found anyway, with the last column giving the same count under the true class hint. Wrong hints made several models reason past the batched output cap; missing errors were refilled one record per request, controls were not, and denominators give the records answered. The completed control subsets are not random samples, so control-side rates are comparable only among models with full coverage.

error class	Claude Fable 5		Claude Sonnet 5		Kimi K3		Grok 4.6	
	blind	hint	blind	hint	blind	hint	blind	hint
wrong-type-or-domain	2/20	20/20	2/20	19/20	2/20	16/20	5/20	15/20
definitional-mismatch	4/15	14/15	2/15	15/15	3/15	13/14	3/15	13/15
missing-hypothesis	8/12	12/12	6/12	12/12	4/12	9/11	5/12	11/12
junk-value-trap	6/11	10/11	3/11	11/11	5/11	11/11	3/11	11/11
wrong-answer-marker	5/11	11/11	1/11	11/11	4/11	11/11	6/11	11/11
wrong-quantifier	6/9	9/9	3/9	9/9	2/9	8/9	5/9	9/9
other	0/6	1/6	0/6	5/6	0/6	4/6	0/6	2/6
wrong-connective	1/3	3/3	1/3	3/3	1/3	1/3	1/3	2/3
vacuous-or-trivial	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
extraneous-hypothesis	0/2	2/2	1/2	2/2	0/2	2/2	2/2	2/2
wrong-referent	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
wrong-constant-or-bound	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1

Table 11: Recall by class (found / gold), blind and with the oracle class hint, for four selected models. The hint supplies the gold classes, so class-level agreement does not independently establish verification.

error class	Claude Fable 5			Claude Sonnet 5			Kimi K3			Grok 4.6		
	nat.	rule	LLM	nat.	rule	LLM	nat.	rule	LLM	nat.	rule	LLM
wrong-answer-marker	5/11	23/23	21/21	1/11	23/23	21/21	4/11	23/23	21/21	6/11	23/23	21/21
missing-hypothesis	8/12	20/20	19/19	6/12	19/20	19/19	4/12	19/20	19/19	5/12	17/20	19/19
wrong-type-or-domain	2/20	20/20	17/19	2/20	20/20	19/19	2/20	20/20	18/19	5/20	19/20	17/19
junk-value-trap	6/11	12/17	17/17	3/11	11/17	16/17	5/11	13/17	17/17	3/11	13/17	17/17
extraneous-hypothesis	0/2	11/11	11/11	1/2	11/11	10/11	0/2	11/11	11/11	2/2	10/11	11/11
wrong-quantifier	6/9	6/6	6/6	3/9	6/6	6/6	2/9	6/6	6/6	5/9	6/6	6/6
vacuous-or-trivial	2/2	3/3	3/3	2/2	3/3	3/3	2/2	3/3	3/3	2/2	3/3	3/3
wrong-constant-or-bound	1/1	2/2	1/2	1/1	2/2	1/2	1/1	2/2	2/2	1/1	2/2	1/2
wrong-connective	1/3	2/2	2/2	1/3	2/2	2/2	1/3	2/2	2/2	1/3	2/2	2/2

Table 12: Correct-class recall (found / gold) on natural negatives and on both synthetic arms, rule-written and model-written, for four selected models. Synthetic answer-marker errors are always assigned the correct class. Results for synthetic carrier errors and junk-value traps vary across models and generation methods; natural examples have low recall.

error class	Grok 4.6		GPT-5.6 Terra Pro		Claude Fable 5	
	found/gold	95% CI	found/gold	95% CI	found/gold	95% CI
wrong-type-or-domain	5/20	[0.11, 0.47]	4/20	[0.08, 0.42]	2/20	[0.03, 0.30]
definitional-mismatch	3/15	[0.07, 0.45]	3/15	[0.07, 0.45]	4/15	[0.11, 0.52]
missing-hypothesis	5/12	[0.19, 0.68]	7/12	[0.32, 0.81]	8/12	[0.39, 0.86]
junk-value-trap	3/11	[0.10, 0.57]	5/11	[0.21, 0.72]	6/11	[0.28, 0.79]
wrong-answer-marker	6/11	[0.28, 0.79]	3/11	[0.10, 0.57]	5/11	[0.21, 0.72]
wrong-quantifier	5/9	[0.27, 0.81]	4/9	[0.19, 0.73]	6/9	[0.35, 0.88]
other	0/6	[0.00, 0.39]	0/6	[0.00, 0.39]	0/6	[0.00, 0.39]
wrong-connective	1/3	[0.06, 0.79]	2/3	[0.21, 0.94]	1/3	[0.06, 0.79]
vacuous-or-trivial	2/2	[0.34, 1.00]	1/2	[0.09, 0.91]	2/2	[0.34, 1.00]
extraneous-hypothesis	2/2	[0.34, 1.00]	1/2	[0.09, 0.91]	0/2	[0.00, 0.66]
wrong-referent	1/1	[0.21, 1.00]	1/1	[0.21, 1.00]	1/1	[0.21, 1.00]
wrong-constant-or-bound	1/1	[0.21, 1.00]	1/1	[0.21, 1.00]	1/1	[0.21, 1.00]
Lean-semantics, pooled	17/57	[0.20, 0.43]	15/57	[0.17, 0.39]	17/57	[0.20, 0.43]

Table 13: Blind detection recall by class with Wilson 95% intervals, for the three selected models, including the pooled Lean-semantics group of Section 5. Results for classes with few examples are descriptive. Pooled Wilson intervals treat label assignments as independent and do not account for dependence between labels on the same record; they should be interpreted as approximate descriptive intervals.

detector	P	R	F1	junk-value recall	false flags (controls)	flags library stmts
static lint	0.24	0.39	0.30	9/11	93/286	33%
Grok 4.6, blind	0.40	0.57	0.47	3/11	63/286	–
GPT-5.6 Terra Pro, blind	0.44	0.49	0.46	5/11	45/286	–
Claude Fable 5, blind	0.61	0.51	0.56	6/11	24/286	–

Table 14: A static junk-value lint (division, inversion, subtraction with an $\mathbb{N}$ binder, `sInf`-like operators, `Nat.find`) as a detection baseline on the natural task, against three blind detectors. P, R, and F1 are whole-task binary scores over all 74 errors and 286 controls, not junk-value-specific; the false-flag column counts flags on the 286 controls, and the last column is the lint's flag rate over the library's review-approved theorem statements. The three models are those of Table 13; Qwen 3.8 Max, which found 7 of 11 traps, is omitted for incomplete coverage. The lint flags most junk-value traps but also flags safe uses; a match identifies a candidate hazard rather than establishes a semantic error.

	natural errors (74)			rule errors on the same population (41)		
model	P	R	F1	P	R	F1
Claude Haiku 4.5	0.24	0.45	0.31	0.46	0.98	0.62
Claude Sonnet 5	0.70	0.38	0.49	1.00	1.00	1.00
Claude Opus 5	0.67	0.30	0.41	1.00	1.00	1.00
Claude Fable 5	0.61	0.51	0.56	1.00	1.00	1.00
GPT-5.6 Terra Pro	0.44	0.49	0.46	0.76	1.00	0.86
Kimi K3$^{(234)}$	0.54	0.45	0.49	0.57	1.00	0.73
Grok 4.6	0.40	0.57	0.47	0.79	1.00	0.88

Table 15: Matched-controls arm, built from the records a preliminary labeling pass marked non-semantic. Rule-written errors are injected into their approved statements, and the remaining records' initial submissions serve as controls shared with the natural task. The construction yields 41 perturbations: 16 wrong-answer-marker, 8 extraneous-hypothesis, 5 missing-hypothesis, 4 wrong-quantifier, 2 each of junk-value-trap, wrong-connective, and vacuous-or-trivial, and 1 each of wrong-type-or-domain and wrong-constant-or-bound. The arm was built from a preliminary labeling; 12 controls whose first push the final labels mark as an error are excluded, leaving 202, and 2 injected records derive from such sources. Recall on injected errors is 98–100%; each model's natural detection recall reweighted to this class mix is 50–71%. Precision also changes: shared control statements do not fix predictions or precision when the error population and batch composition differ. The comparison does not isolate a batch-composition effect.

model	recall by strategy (found / scored) constant	equality	exponent	unpaired	variable new	variable type	released labels P	released labels F1	audited labels P	audited labels F1
Claude Haiku 4.5	36/38	6/6	7/7	15/15	6/22	16/17	0.54	0.65	0.61	0.68
Claude Sonnet 5	38/38	6/6	7/7	15/15	21/22	17/17	0.73	0.84	0.86	0.91
Claude Opus 5	38/38	6/6	7/7	15/15	22/22	17/17	0.89	0.94	0.97	0.94
Claude Fable 5	38/38	6/6	7/7	15/15	17/22	17/17	0.78	0.85	0.91	0.93
GPT-5.6 Terra Pro	38/38	6/6	7/7	15/15	20/22	17/17	0.64	0.77	0.77	0.86
Kimi K3	38/38	6/6	7/7	15/15	15/22	17/17	0.66	0.77	0.79	0.85
Grok 4.6	38/38	6/6	7/7	15/15	21/22	17/17	0.63	0.77	0.76	0.86

Table 16: Third synthetic arm: FormalAlign's released MiniF2F misalignments. Recall by perturbation strategy (found / scored), with precision and F_1 computed against the released labels, aligned candidates serving as controls, and against labels corrected by our audit of the 37 contested aligned candidates, judging commented-out references by their content (21 defects, 16 false alarms; the audit sheet with verdicts and Lean checks is in the artifact). We do not apply our synthetic typechecking filter to this set.

error class	natural	synthetic
wrong-type-or-domain	20	20
definitional-mismatch	15	–
missing-hypothesis	12	20
wrong-answer-marker	11	23
junk-value-trap	11	17
wrong-quantifier	9	6
other	6	–
wrong-connective	3	2
vacuous-or-trivial	2	3
extraneous-hypothesis	2	11
wrong-referent	1	–
wrong-constant-or-bound	1	2
semantic total	74	104
non-semantic controls	286	255

Table 17: Composition of the synthetic set against the natural gold negatives. Natural records are multi-label, so the natural column's class counts sum to 93 label assignments over the 74 records in the total row; each synthetic record carries one class. The synthetic class proportions follow a preliminary labeling pass, under which answer-marker errors were the most frequent class; reweighting each model's synthetic binary detection recall within each class to the final class mix changes it by at most 0.02 (it stays at 0.97–1.00 for the five strongest non-generator models on both arms, against 0.42–0.60 natural recall on the same classes). Classes without a rule-based operator (wrong referent, definitional mismatch, other) are absent from the rule-written arm; the natural baseline in Table 5 is therefore also reported on the class-matched subset.

model	blind F1	blind Lean-sem.	forced checklist (no oracle) P	R	F1	ΔF1	Lean-sem.	class hint Lean-sem.
Claude Haiku 4.5	0.31	4/57	0.30	0.72	0.43	$+0.12$	14/57	48/57
Claude Sonnet 5$^{(358)}$	0.49	8/57	0.51	0.32	0.40	-0.09	8/57	56/57
Claude Opus 5	0.41	8/57	0.48	0.34	0.40	-0.01	11/57	44/46
Claude Fable 5$^{(343)}$	0.56	17/57	0.53	0.56	0.54	-0.01	16/55	55/57
GPT-5.6 Terra Pro	0.46	15/57	0.39	0.55	0.46	-0.01	19/57	51/57
Kimi K3$^{(348)}$	0.49	14/57	0.52	0.48	0.50	$+0.01$	13/56	51/56
Grok 4.6	0.47	17/57	0.40	0.68	0.50	$+0.03$	20/57	50/57

Table 18: Unassisted detection, a non-oracle per-class checklist, and an oracle error-class hint. Neither unassisted detection nor the checklist receives gold labels; the oracle condition supplies the reviewer-assigned class. The checklist yields only small changes in detection F_1 and leaves recall on the pooled Lean-semantics classes well below the class-hint condition. Superscripts give scoreable records when below 360; the checklist runs of Claude Fable 5 and Kimi K3 lack two and one Lean-semantics labels respectively, hence the smaller denominators.

D Error taxonomy

Labels apply to the first-pushed statement judged against the docstring at merge; the reviewer-approved statement is the answer key, but the label describes what was wrong with the first push, not what the diff did. Classes are multi-label. Each record also carries a *direction*: whether the initial statement was logically weaker than the informal claim (before-weaker), stronger (before-stronger), incomparable, or equivalent. The annotation prompt also permits `unclear` when the direction cannot be determined. Among the 74 negatives the gold directions are 16 before-weaker, 16 before-stronger, 6 incomparable, and 36 unclear. Models return a direction with every prediction, but the main results score classes only. On the errors a model detects and whose gold direction is definite, its direction agrees with the annotator's on 36–81% of records in the assisted condition and 21–82% in detection.

The final annotation pass hid both model suggestions and preliminary labels. Relative to the preliminary pass, 45 semantic labels became non-semantic and 14 non-semantic labels became semantic. The revised procedure incorporated additional Lean elaboration checks and treated omission of the project's answer marker as non-semantic when the statement otherwise preserved the docstring's content. These changes describe a superseded annotation procedure, not an independent reliability estimate.

class	meaning	canonical example
wrong-quantifier	$\exists/\forall$ swapped, wrong scope or order, quantifier over the wrong variable	$\exists B, \forall S$ vs $\forall B, \exists c$ (Figure 2)
missing-hypothesis	side condition in the informal statement absent from the formal one	a positivity or distinctness condition dropped
extraneous-hypothesis	formal statement assumes more than the informal one	a bound $393 < n$ where the question concerns every $n \geq 4$
wrong-type-or-domain	wrong carrier ($\mathbb{N}$ vs $\mathbb{Z}$ vs $\mathbb{Q}$ vs $\mathbb{R}$, `Fin n` vs $\mathbb{N}$), wrong codomain, meaning-changing coercion	a sequence defined into $\mathbb{N}$ instead of $\mathbb{Q}$
wrong-referent	points at the wrong declaration or constant	residues of A counted where the question is about $A + A$
wrong-constant-or-bound	off-by-one, wrong constant or exponent, strict vs non-strict	an exponent θ where $1-\theta$ was meant
wrong-connective	$\wedge/\vee$ / $\rightarrow$ / $\leftrightarrow$ misuse, misplaced negation	a biconditional whose reverse direction fails, where an implication was meant
wrong-answer-marker	`answer(True/False/sorry)` wrong or on the wrong side of the $\leftrightarrow$	Figure 2a
junk-value-trap	meaning diverges through Lean junk values ($x/0$, $\mathbb{N}$ subtraction, $\inf \emptyset$, `tsum` of a divergent series)	Figure 2b ($\inf \emptyset = 0$ in $\mathbb{N}$)
definitional-mismatch	an auxiliary definition does not capture the informal notion	`AdmitsLieGroupStructure` (Appendix F)
vacuous-or-trivial	trivially true or false (empty range, wrong index set) contrary to intent	`[Subsingleton G]` assumed, making a trivial center immediate
non-semantic	rename, namespace, style, whitespace; meaning unchanged (excluded from negatives)	a namespace prefix dropped after `open`
informal-changed	the docstring also changed between first push and merge; assigned to no record	
other	anything else; explained in the record's note	the arguments of a Legendre symbol reversed

E Frozen prompts

The shared preamble is followed by one condition-specific paragraph. Records are sent as JSON in fixed chunks of twelve; the model returns one JSON object keyed by record id. The versions below replace Unicode with ASCII equivalents and wrap long lines for typesetting; the artifact preserves the exact prompts. These formatting changes do not modify the prompts used in evaluation. Conditions that reuse another condition's prompt verbatim (the synthetic, FormalAlign, and matched-controls arms all use the detection prompt on a different dataset) are listed by reference. The assisted condition's `bucket` field is a triage flag that marked 33 records as machine-suspected renames; all 33

are gold non-semantic and every model returned non-semantic for all of them. Excluding them from scoring leaves precision, recall, and F_1 in Table 2 unchanged to two decimals and lowers κ by at most 0.03, as removing correctly classified controls must. This is a sensitivity check; the records were not rerun without the flag, so it does not establish that the flag had no effect. The final paragraph gives the prompt used to generate the model-written synthetic arm.

Condition `assisted`.

```
You label candidate Lean 4 formalization errors for a research corpus.
Each record pairs an informal mathematical statement (a docstring) with a formal
Lean 4 statement `formal_before`. Judge whether `formal_before` faithfully
formalizes the informal statement.

Error classes (multi-label): wrong-quantifier, missing-hypothesis,
    extraneous-hypothesis, wrong-type-or-domain, wrong-referent,
    wrong-constant-or-bound, wrong-connective, wrong-answer-marker,
    junk-value-trap, definitional-mismatch, vacuous-or-trivial, non-semantic,
    informal-changed, other.
- wrong-quantifier: exists/forall swapped, wrong scope or order
- missing-hypothesis: a side condition in the informal statement is absent
- extraneous-hypothesis: the formal statement assumes more than the informal one
- wrong-type-or-domain: wrong carrier (N vs Z vs Q vs R, Fin n vs N, codomain)
- wrong-referent: references the wrong declaration or constant
- wrong-constant-or-bound: off-by-one, wrong constant, strict vs non-strict
- wrong-connective: and/or/implies/iff misuse, misplaced negation
- wrong-answer-marker: the project's answer(...) marker wrong or misplaced
- junk-value-trap: meaning diverges via Lean junk values (x/0, N-subtraction, sInf
    of empty)
- definitional-mismatch: an auxiliary def used by the statement does not capture
    the informal notion
- vacuous-or-trivial: trivially true/false contrary to informal intent
- non-semantic: rename/style only, meaning unchanged (i.e. no error)
- informal-changed / other: as named.

Direction of `formal_before` relative to the informal intent: one of
before-weaker, before-stronger, incomparable, equivalent, unclear
(equivalent only together with non-semantic).

Output: a single JSON object, no prose, no code fences, mapping every input
record id to {"classes": [...], "direction": "...", "confidence":
    "high"|"medium"|"low",
"rationale": "<exactly one sentence naming the decisive difference>"}.
Every id in the input must appear. If `informal` is null, judge on the other
fields and say "no docstring" in the rationale.

`formal_after` is the reviewer-approved corrected statement (answer key) and
`comments` are the PR reviewers' remarks; use both. A record with bucket
"refactor-rename" was machine-suspected non-semantic -- confirm or overrule.
```

Condition `assisted-nocomments`.

```
You label candidate Lean 4 formalization errors for a research corpus.
Each record pairs an informal mathematical statement (a docstring) with a formal
Lean 4 statement `formal_before`. Judge whether `formal_before` faithfully
formalizes the informal statement.

Error classes (multi-label): wrong-quantifier, missing-hypothesis,
    extraneous-hypothesis, wrong-type-or-domain, wrong-referent,
    wrong-constant-or-bound, wrong-connective, wrong-answer-marker,
    junk-value-trap, definitional-mismatch, vacuous-or-trivial, non-semantic,
    informal-changed, other.
- wrong-quantifier: exists/forall swapped, wrong scope or order
- missing-hypothesis: a side condition in the informal statement is absent
- extraneous-hypothesis: the formal statement assumes more than the informal one
```

```
- wrong-type-or-domain: wrong carrier (N vs Z vs Q vs R, Fin n vs N, codomain)
- wrong-referent: references the wrong declaration or constant
- wrong-constant-or-bound: off-by-one, wrong constant, strict vs non-strict
- wrong-connective: and/or/implies/iff misuse, misplaced negation
- wrong-answer-marker: the project's answer(...) marker wrong or misplaced
- junk-value-trap: meaning diverges via Lean junk values (x/0, N-subtraction, sInf
    of empty)
- definitional-mismatch: an auxiliary def used by the statement does not capture
    the informal notion
- vacuous-or-trivial: trivially true/false contrary to informal intent
- non-semantic: rename/style only, meaning unchanged (i.e. no error)
- informal-changed / other: as named.

Direction of `formal_before` relative to the informal intent: one of
before-weaker, before-stronger, incomparable, equivalent, unclear
(equivalent only together with non-semantic).

Output: a single JSON object, no prose, no code fences, mapping every input
record id to {"classes": [...], "direction": "...", "confidence":
    "high"|"medium"|"low",
"rationale": "<exactly one sentence naming the decisive difference>"}.
Every id in the input must appear. If `informal` is null, judge on the other
fields and say "no docstring" in the rationale.

`formal_after` is the reviewer-approved corrected statement (answer key); use
it. You are not shown the reviewers' comments. A record with bucket
"refactor-rename" was machine-suspected non-semantic -- confirm or overrule.
```

Condition `detection`.

```
You label candidate Lean 4 formalization errors for a research corpus.
Each record pairs an informal mathematical statement (a docstring) with a formal
Lean 4 statement `formal_before`. Judge whether `formal_before` faithfully
formalizes the informal statement.

Error classes (multi-label): wrong-quantifier, missing-hypothesis,
    extraneous-hypothesis, wrong-type-or-domain, wrong-referent,
    wrong-constant-or-bound, wrong-connective, wrong-answer-marker,
    junk-value-trap, definitional-mismatch, vacuous-or-trivial, non-semantic,
    informal-changed, other.
- wrong-quantifier: exists/forall swapped, wrong scope or order
- missing-hypothesis: a side condition in the informal statement is absent
- extraneous-hypothesis: the formal statement assumes more than the informal one
- wrong-type-or-domain: wrong carrier (N vs Z vs Q vs R, Fin n vs N, codomain)
- wrong-referent: references the wrong declaration or constant
- wrong-constant-or-bound: off-by-one, wrong constant, strict vs non-strict
- wrong-connective: and/or/implies/iff misuse, misplaced negation
- wrong-answer-marker: the project's answer(...) marker wrong or misplaced
- junk-value-trap: meaning diverges via Lean junk values (x/0, N-subtraction, sInf
    of empty)
- definitional-mismatch: an auxiliary def used by the statement does not capture
    the informal notion
- vacuous-or-trivial: trivially true/false contrary to informal intent
- non-semantic: rename/style only, meaning unchanged (i.e. no error)
- informal-changed / other: as named.

Direction of `formal_before` relative to the informal intent: one of
before-weaker, before-stronger, incomparable, equivalent, unclear
(equivalent only together with non-semantic).

Output: a single JSON object, no prose, no code fences, mapping every input
record id to {"classes": [...], "direction": "...", "confidence":
    "high"|"medium"|"low",
"rationale": "<exactly one sentence naming the decisive difference>"}.
Every id in the input must appear. If `informal` is null, judge on the other
```

```
fields and say "no docstring" in the rationale.

You see ONLY the informal statement and `formal_before` -- no answer key.
If `formal_before` is a faithful formalization, output "classes": ["non-semantic"]
with direction "equivalent".
```

Condition `detection-context`.

```
You label candidate Lean 4 formalization errors for a research corpus.
Each record pairs an informal mathematical statement (a docstring) with a formal
Lean 4 statement `formal_before`. Judge whether `formal_before` faithfully
formalizes the informal statement.

Error classes (multi-label): wrong-quantifier, missing-hypothesis,
    extraneous-hypothesis, wrong-type-or-domain, wrong-referent,
    wrong-constant-or-bound, wrong-connective, wrong-answer-marker,
    junk-value-trap, definitional-mismatch, vacuous-or-trivial, non-semantic,
    informal-changed, other.
- wrong-quantifier: exists/forall swapped, wrong scope or order
- missing-hypothesis: a side condition in the informal statement is absent
- extraneous-hypothesis: the formal statement assumes more than the informal one
- wrong-type-or-domain: wrong carrier (N vs Z vs Q vs R, Fin n vs N, codomain)
- wrong-referent: references the wrong declaration or constant
- wrong-constant-or-bound: off-by-one, wrong constant, strict vs non-strict
- wrong-connective: and/or/implies/iff misuse, misplaced negation
- wrong-answer-marker: the project's answer(...) marker wrong or misplaced
- junk-value-trap: meaning diverges via Lean junk values (x/0, N-subtraction, sInf
    of empty)
- definitional-mismatch: an auxiliary def used by the statement does not capture
    the informal notion
- vacuous-or-trivial: trivially true/false contrary to informal intent
- non-semantic: rename/style only, meaning unchanged (i.e. no error)
- informal-changed / other: as named.

Direction of `formal_before` relative to the informal intent: one of
before-weaker, before-stronger, incomparable, equivalent, unclear
(equivalent only together with non-semantic).

Output: a single JSON object, no prose, no code fences, mapping every input
record id to {"classes": [...], "direction": "...", "confidence":
    "high"|"medium"|"low",
"rationale": "<exactly one sentence naming the decisive difference>"}.
Every id in the input must appear. If `informal` is null, judge on the other
fields and say "no docstring" in the rationale.

You see the informal statement, `formal_before`, and `context`: the imports,
`open`s, and the auxiliary declarations from the same file (at the same commit)
that `formal_before` depends on. There is no answer key. Use the context to
reason about how the statement elaborates rather than how it reads: what an
auxiliary definition actually unfolds to, which carrier a coercion lands in,
and whether an operation can reach a Lean junk value on some input.
If `formal_before` is a faithful formalization, output "classes": ["non-semantic"]
with direction "equivalent".
```

Condition `detection-synthetic`. Identical to condition `detection`; only the input records differ.

Condition `detection-hint`.

```
You label candidate Lean 4 formalization errors for a research corpus.
Each record pairs an informal mathematical statement (a docstring) with a formal
Lean 4 statement `formal_before`. Judge whether `formal_before` faithfully
formalizes the informal statement.

Error classes (multi-label): wrong-quantifier, missing-hypothesis,
```

```
    extraneous-hypothesis, wrong-type-or-domain, wrong-referent,
    wrong-constant-or-bound, wrong-connective, wrong-answer-marker,
    junk-value-trap, definitional-mismatch, vacuous-or-trivial, non-semantic,
    informal-changed, other.
- wrong-quantifier: exists/forall swapped, wrong scope or order
- missing-hypothesis: a side condition in the informal statement is absent
- extraneous-hypothesis: the formal statement assumes more than the informal one
- wrong-type-or-domain: wrong carrier (N vs Z vs Q vs R, Fin n vs N, codomain)
- wrong-referent: references the wrong declaration or constant
- wrong-constant-or-bound: off-by-one, wrong constant, strict vs non-strict
- wrong-connective: and/or/implies/iff misuse, misplaced negation
- wrong-answer-marker: the project's answer(...) marker wrong or misplaced
- junk-value-trap: meaning diverges via Lean junk values (x/0, N-subtraction, sInf
    of empty)
- definitional-mismatch: an auxiliary def used by the statement does not capture
    the informal notion
- vacuous-or-trivial: trivially true/false contrary to informal intent
- non-semantic: rename/style only, meaning unchanged (i.e. no error)
- informal-changed / other: as named.

Direction of `formal_before` relative to the informal intent: one of
before-weaker, before-stronger, incomparable, equivalent, unclear
(equivalent only together with non-semantic).

Output: a single JSON object, no prose, no code fences, mapping every input
record id to {"classes": [...], "direction": "...", "confidence":
    "high"|"medium"|"low",
"rationale": "<exactly one sentence naming the decisive difference>"}.
Every id in the input must appear. If `informal` is null, judge on the other
fields and say "no docstring" in the rationale.

You see the informal statement, `formal_before`, and `hint`: the error class
or classes a human reviewer assigned to this record, or "non-semantic" when
the reviewer found no error. There is no answer key beyond that. The hint
tells you what kind of problem to look for, not where it is or whether the
hint is right; verify it against the statement and output the classes you can
actually substantiate. If you cannot substantiate any error, output
"classes": ["non-semantic"] with direction "equivalent".
```

Condition `detection-exists`.

```
You label candidate Lean 4 formalization errors for a research corpus.
Each record pairs an informal mathematical statement (a docstring) with a formal
Lean 4 statement `formal_before`. Judge whether `formal_before` faithfully
formalizes the informal statement.

Error classes (multi-label): wrong-quantifier, missing-hypothesis,
    extraneous-hypothesis, wrong-type-or-domain, wrong-referent,
    wrong-constant-or-bound, wrong-connective, wrong-answer-marker,
    junk-value-trap, definitional-mismatch, vacuous-or-trivial, non-semantic,
    informal-changed, other.
- wrong-quantifier: exists/forall swapped, wrong scope or order
- missing-hypothesis: a side condition in the informal statement is absent
- extraneous-hypothesis: the formal statement assumes more than the informal one
- wrong-type-or-domain: wrong carrier (N vs Z vs Q vs R, Fin n vs N, codomain)
- wrong-referent: references the wrong declaration or constant
- wrong-constant-or-bound: off-by-one, wrong constant, strict vs non-strict
- wrong-connective: and/or/implies/iff misuse, misplaced negation
- wrong-answer-marker: the project's answer(...) marker wrong or misplaced
- junk-value-trap: meaning diverges via Lean junk values (x/0, N-subtraction, sInf
    of empty)
- definitional-mismatch: an auxiliary def used by the statement does not capture
    the informal notion
- vacuous-or-trivial: trivially true/false contrary to informal intent
- non-semantic: rename/style only, meaning unchanged (i.e. no error)
```

```
- informal-changed / other: as named.

Direction of `formal_before` relative to the informal intent: one of
before-weaker, before-stronger, incomparable, equivalent, unclear
(equivalent only together with non-semantic).

Output: a single JSON object, no prose, no code fences, mapping every input
record id to {"classes": [...], "direction": "...", "confidence":
    "high"|"medium"|"low",
"rationale": "<exactly one sentence naming the decisive difference>"}.
Every id in the input must appear. If `informal` is null, judge on the other
fields and say "no docstring" in the rationale.

You see the informal statement, `formal_before`, and `hint`: either
"error-present", meaning a human reviewer found a semantic error in this
record (of some class you are not told), or "non-semantic", meaning the
reviewer found no error. There is no answer key beyond that. Find and classify
the error if the hint says one is present; verify rather than assume. If you
cannot substantiate any error, output "classes": ["non-semantic"] with
direction "equivalent".
```

Condition `detection-distractor`. Identical to condition `detection-hint`; only the input records differ.

Condition `detection-checklist`.

```
You label candidate Lean 4 formalization errors for a research corpus.
Each record pairs an informal mathematical statement (a docstring) with a formal
Lean 4 statement `formal_before`. Judge whether `formal_before` faithfully
formalizes the informal statement.

Error classes (multi-label): wrong-quantifier, missing-hypothesis,
    extraneous-hypothesis, wrong-type-or-domain, wrong-referent,
    wrong-constant-or-bound, wrong-connective, wrong-answer-marker,
    junk-value-trap, definitional-mismatch, vacuous-or-trivial, non-semantic,
    informal-changed, other.
- wrong-quantifier: exists/forall swapped, wrong scope or order
- missing-hypothesis: a side condition in the informal statement is absent
- extraneous-hypothesis: the formal statement assumes more than the informal one
- wrong-type-or-domain: wrong carrier (N vs Z vs Q vs R, Fin n vs N, codomain)
- wrong-referent: references the wrong declaration or constant
- wrong-constant-or-bound: off-by-one, wrong constant, strict vs non-strict
- wrong-connective: and/or/implies/iff misuse, misplaced negation
- wrong-answer-marker: the project's answer(...) marker wrong or misplaced
- junk-value-trap: meaning diverges via Lean junk values (x/0, N-subtraction, sInf
    of empty)
- definitional-mismatch: an auxiliary def used by the statement does not capture
    the informal notion
- vacuous-or-trivial: trivially true/false contrary to informal intent
- non-semantic: rename/style only, meaning unchanged (i.e. no error)
- informal-changed / other: as named.

Direction of `formal_before` relative to the informal intent: one of
before-weaker, before-stronger, incomparable, equivalent, unclear
(equivalent only together with non-semantic).

Output: a single JSON object, no prose, no code fences, mapping every input
record id to {"classes": [...], "direction": "...", "confidence":
    "high"|"medium"|"low",
"rationale": "<exactly one sentence naming the decisive difference>"}.
Every id in the input must appear. If `informal` is null, judge on the other
fields and say "no docstring" in the rationale.

You see ONLY the informal statement and `formal_before` -- no answer key and
```

```
no hint. Do not judge the record as a whole first. Walk the error classes one
at a time and decide for each whether it applies to this record. Add a field
"checklist" to each record's output: an object mapping every class name
except non-semantic, informal-changed, and other to true or false. Then set
"classes" to exactly the classes marked true (or ["non-semantic"] with
direction "equivalent" if none), and fill direction, confidence, and
rationale as usual.
```

Condition `detection-formalalign`. Identical to condition `detection`; only the input records differ.

Condition `detection-synthetic-matched`. Identical to condition `detection`; only the input records differ.

Condition `detection-synthetic-llm`. Identical to condition `detection`; only the input records differ.

Generation prompt for the model-written arm. System prompt of `scripts/make_synthetic_llm.py`; the user turn is a JSON object with the target class and its gloss, the docstring, the approved statement, and the file context. Each request allows 12,000 output tokens; sampling parameters are provider defaults. Generator: claude-opus-5 at effort medium; seed 0; oversample 2. Controls are the rule-based arm's 255 unperturbed positives, reused verbatim.

```
You are helping build an evaluation set of realistic Lean 4 formalization errors.

You will be given an informal mathematical statement (a docstring), its
reviewer-approved Lean 4 formalization, the file context it lives in (imports,
opens, and the auxiliary declarations it uses), and ONE target error class.

Write the version of the statement that a competent, careful formalizer might
plausibly have pushed by mistake, such that:
  1. it contains exactly one error of the target class and no other change;
  2. it reads as a natural, intended formalization -- not an obvious typo, not a
     nonsensical statement, nothing a reader would flag on sight;
  3. it still elaborates: every identifier you use must exist in Mathlib or in
     the given context, and types must line up;
  4. the error is real: the new statement is NOT equivalent to the original.

Prefer errors that are wrong only on elaboration (what an expression denotes,
which carrier a coercion lands in, what a definition unfolds to) over errors
visible on the surface of the text. Keep the declaration name unchanged.

Output a single JSON object, no prose, no code fences:
{"statement": "<full Lean statement text, from `theorem`/`lemma` through the
 type, without `:= by sorry`>", "direction":
     "before-weaker"|"before-stronger"|"incomparable",
 "why": "<one sentence: what changed and why it is wrong>"}
If no natural error of this class is possible for this statement, output
{"statement": null, "why": "<reason>"}.
```

F Sample records

Three gold negatives from the released sample, ASCII-simplified (binders and operators spelled out as `ALL`, `EX`, `/\`; the Euclidean model with corners $\mathcal{R}$ written `R`; the analytic degree ω written `omega`). Every statement elaborated in its file; reviewer handles are aliased.

AdmitsLieGroupStructure gold: definitional-mismatch; direction: unclear.

```
-- Docstring: A topological group admits a Lie group structure if it has a
    `LieGroupPresentation` in some finite dimension.

-- first push
def AdmitsLieGroupStructure (G : Type*) [Group G] [TopologicalSpace G] :
    Prop :=
  EX (k : Nat) (cs : ChartedSpace (EuclideanSpace Real (Fin k)) G),
    letI := cs
    IsManifold (R k) omega G /\ LieGroup (R k) omega G

-- as merged
def AdmitsLieGroupStructure (G : Type u) [Group G] [TopologicalSpace G] :
    Prop :=
  EX n, Nonempty (LieGroupPresentation G n)
```

conjecture4 gold: wrong-type-or-domain; direction: unclear.

```
-- Docstring: Conjecture (4): Let $b(n) = \mathrm{A004125}(n) = \sum_{k=1}^n (n \bmod k)$.
    Then $\mathrm{ord}_p(a(np)) = \sum_{i \ge 0} b(\lfloor n/p^i \rfloor)$.

-- first push
theorem conjecture4 (n p : Nat) (hn : 0 < n) (hp : p.Prime) :
    (a (n * p)).factorization p = SUM' i : Nat, b (n / p ^ i)

-- as merged
theorem conjecture4 (n p : Nat) (hn : 0 < n) (hp : p.Prime) :
    padicValRat p (a (n * p)) = SUM' i : Nat, (b (n / p ^ i) : Int)
```

conjecture2 gold: wrong-type-or-domain; direction: unclear.

```
-- Docstring: $a(n)$ is odd if and only if $n = 2, 2^2, 2^3, \dots$. - _Zhi-Wei Sun_, Apr
    06 2010

-- first push
theorem conjecture2 (n : Nat) (hn : 1 <= n) :
    Odd (a n) <-> EX m : Nat, 1 <= m /\ n = 2 ^ m

-- as merged
theorem conjecture2 (n : Nat) (hn : 1 <= n) :
    ((a n).den = 1 /\ Odd (a n).num) <-> EX m : Nat, 1 <= m /\ n = 2 ^ m
```

What is wrong, and how one can tell. `AdmitsLieGroupStructure`: a `LieGroupPresentation G n` is a Hausdorff, second-countable analytic Lie group together with a topological-group isomorphism to G. The first push instead asks for a chart structure on G itself and explicitly requires neither Hausdorffness nor second countability, so it admits non-Hausdorff manifold structures that no Lie group in the informal sense has. Claude Fable 5 (blind and with file context) and Grok 4.6 returned non-semantic; with the presentation structure in its context, Fable's rationale reads "existence of some k with a chart structure making G an analytic manifold and analytic Lie group is the unbundled content of a `LieGroupPresentation`". `conjecture4` and `conjecture2` depend on the file's sequence `a`, defined in the first push as `def a (n : ℕ) : ℕ := ... num / den` with natural-number division, so `a n` is a truncated quotient and the p-adic order and parity are computed on the truncated value; the merged version redefines `a` into $\mathbb{Q}$ and uses `padicValRat` and the numerator and denominator. The definition of `a` is in the file context, yet Fable's context-augmented rationale for `conjecture4` reads "`factorization p` is `ord_p`, matching the conjecture".