CO-PEER: AUDITING MATHEMATICS WITH PROOFS OF ERROR

Anonymous authors
Paper under double-blind review

ABSTRACT

A precise counterexample can reveal a recurring failure pattern, motivate a correction, and lead to a theorem about an entire family. We introduce Co-PEER, a workflow for developing and checking such counterexamples, alongside BugBank, a collection of 147 dossiers from 137 source units. Each dossier connects a source claim and its assumptions to a counterexample, a Lean-checked proof, and a separate assessment of whether the objection applies to the source. We also introduce a novel benchmark based off BugBank. On 100 texts known to contain an error, the best-performing model selects a line marked incorrect as its first choice in 73.0% of cases, compared with 18.0% for the strongest rule-based baseline. In author-led mathematical follow-ups with AI-assisted proofs, we extend individual witnesses into complete characterizations and sharp bounds. We characterize and count frozen six-colorings of doubled odd cycles, where no vertex can legally change color: for odd cycle length $n \geq 5$, they exist exactly when three divides n, yielding $90 \cdot 2^n$ colorings for labeled vertices and colors. We also prove that $9/22$ is the optimal limiting fraction of ordered triples $(x, y, x + y)$ with three different colors within a two-threshold integer-coloring family, exceeding the conjectured bound of $2/5$.

1 INTRODUCTION

Constructing a complete proof can demand substantial reasoning, while a counterexample or false step may provide a shorter route to identifying an error. This raises a practical question: how effectively can smaller, locally run models serve as mathematical critics? To use an objection effectively requires knowing exactly what the example establishes: which source claim fails, why the example satisfies its assumptions, and what follows for the surrounding argument.

We introduce **Co-PEER** (Collaborative Proof Expert Evaluation Review), a workflow that makes this evidence formal, provable, and inspectable. A *critic* is a language model asked to identify and explain an error; its *witness* is the object used as a counterexample. Each *objection dossier* connects a quoted source claim and its assumptions to the witness, a proof checked by Lean, and a separate review of whether the objection applies to the source and what it establishes. This checked local failure and its source-level interpretation form a *proof of error*. **BugBank** contains 147 such dossiers from 137 source units (Sections 2–3).

Novel mathematical highlights. Author-led, AI-assisted follow-ups develop individual witnesses into results about entire families. The two specific informal statements, which prove novel results, below illustrate what this can establish.

Exactly when a coloring gets stuck. Replace each vertex of an odd cycle of length $n \geq 5$ by two adjacent vertices, and fully connect neighboring pairs. Although five colors suffice for this graph, a six-coloring in which no vertex can legally change color exists exactly when n is divisible by three. When they exist, there are exactly $90 \cdot 2^n$ such colorings with labeled vertices and colors. Building on the known existence construction of Shaebani (2009), we prove necessity and the exact count, and show how these graphs refute the source conjecture (Proposition 2; Appendices K and S.2).

A better construction, with a sharp limit. Color the integers from one to n with three colors and consider triples $(x, y, x + y)$ whose entries have different colors. As n grows, a two-threshold

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.

construction attains a limiting fraction of $9/22$, exceeding the conjectured maximum of $2/5$ (Parczyk & Spiegel, 2026); independent work gives the same construction and lower bound (Hegde et al., 2026). We prove that its thresholds are uniquely optimal within this family: improving the fraction further requires changing the construction (Theorem 3; Appendix S.1).

Section 6 also proves a complete correction of an integer square-root program: the original fails exactly at positive perfect squares, and the source's corrected rule (Answer to "Relating a proof to a Haskell program") works on every nonnegative integer input (Proposition 1; Appendix B).

Contributions. We contribute BugBank, a benchmark built from its checked errors, and general mathematical results developed from checked counterexamples. BugBank provides 147 human-approved objection dossiers from 137 source units, each connecting a source claim and its assumptions to a counterexample, a Lean proof, and review of the source interpretation. Its 100-text benchmark tests whether LLM critics can identify the checked incorrect lines. Across 13 configurations, the highest first-choice score is 73.0%, versus 18.0% for the strongest rule-based baseline; a 20B local configuration achieves 43.0%. The counterexample procedure also supports general mathematical results: exact error classifications, verified repairs, and optimality theorems. The workflow explains how this reusable evidence is constructed and checked (Appendices T, N, and P).

2 What Makes a Counterexample a Valid Objection?

2.1 Three questions that must be answered separately

Does the mathematics check out? A counterexample must satisfy the claim's assumptions and violate its conclusion. For a program, this might mean supplying an allowed input and proving that the output is wrong. A *formalization* expresses these objects and statements in the precise language of a proof assistant. Lean's *kernel*, its core proof checker, checks a proof of that formal statement. We call the checked proof a *certificate*. The certificate establishes the formal statement; source review checks its correspondence to the original claim.

Does the checked statement match the source? This question asks whether the translation accurately represents the source claim and its assumptions. The translation must preserve the assumptions relevant to the witness. For example, consider an intermediate claim in a sorting argument that assumes the list's first element is smaller than its last. The list $(3, 4, 2, 5, 1)$ violates this assumption because $3 > 1$. The list $(3, 5, 2, 6, 1, 7, 4)$ satisfies it because $3 < 4$. Both lists contradict the claimed intermediate conclusion. Only the second is a valid counterexample to this claim, because it satisfies the required input condition (Appendix D.6.1).

What follows from the error? This is the objection's *scope*. A false intermediate step, an incomplete construction, and a false final theorem are different findings. For example, a proposed proof of the handshaking lemma tries to build every graph by adding two new vertices joined only to each other. This step cannot produce the connected path on three vertices, so the construction does not cover every graph. The path still satisfies the lemma: its vertex degrees sum to $1 + 2 + 1 = 4$, twice the number of edges. Appendix C.1 gives this example and a separate matrix claim refuted by the same path.

These questions also explain what a translation must accomplish. The formal witness must correspond to an object allowed by the source; its formal failure must imply failure of the source conclusion. Appendix Q.1 states and proves these requirements precisely. They do not require translating every sentence in the source, but they do require explaining how the quoted source text supports the assumptions and conclusion attributed to it.

2.2 What a BugBank record contains

Each dossier is an error report organized around the questions above. A file hash is a fingerprint used to identify exact file contents. Hashes and build logs help establish *which artifact was checked*; they do not establish that its meaning matches the source. We preserve earlier counterexamples and their proof and review records, and document corrections separately. Appendix N describes the released ledgers, rendered dossiers and build instructions.

Record component	What a reader can inspect
Source and claim	The saved source version, quoted passage, selected assertion, and the assumptions under which that assertion is claimed to hold.
Counterexample and proof	The witness, mathematical definitions, and exact named Lean theorem that checks the failure.
Checking record	File identifiers, dependencies, build logs, and the assumptions used by the formal proof.
Review and consequence	Reasons the witness does or does not apply to the source, unresolved concerns, and the conclusion the objection supports.

2.3 How the record is constructed

Co-PEER selects a claim and identifies its assumptions, then searches for a counterexample satisfying those assumptions. The original auditing protocol favors short objects: at most 12 graph vertices, 10 objects, or 10 significant trace steps. For each research conjecture, we separately specify the types of examples to examine and the ranges of parameter values to test. A separately written checking program tests the proposed witness using exact arithmetic. Human verifiers and coding agents translate surviving failures into Lean and inspect the selected theorem and its proof dependencies. A separate, model-assisted review evaluates the source connection and records the supported consequence. These stages follow the principle of producing a result and checking its certificate separately (McConnell et al., 2011; de Moura & Ullrich, 2021; The mathlib Community, 2020).

A successful exhaustive check establishes the claim for the example types and sizes it covers. Separately written checking programs and separate model calls can share mistaken assumptions and make the same errors. We record the procedures, search limits, unsuccessful outcomes, and version histories (Appendices T and N).

3 BugBank: The Collection and Its Evidence

The current approved core contains **147 dossiers from 137 source units**, each with a Lean certificate. A single source problem may have multiple dossiers. Every retained dossier has a selected Lean theorem with a reported successful build and inspection of the assumptions used by its proof. A separate review checks whether each objection accurately represents the source claim and its assumptions, and records any limitations or unresolved questions. The collection brings together these verified mathematical objections for detailed analysis. Within a single dossier, a reader can inspect the source passage, rerun its certificate, and examine its human reviews. Table 1 separates the questions supported by the different studies.

The collection includes 109 objections to intermediate assertions, and 74 of the 100 benchmark inputs test such steps. These records make the reasoning inside mathematical arguments directly inspectable: each challenged step is linked to its source assumptions and a checked counterexample. Appendix N describes the retained 147-dossier resource, the 100-input primary evaluation set, and the categories used to describe their contents.

Collection or study	Question it supports
147 retained dossiers	What claim, counterexample, formal proof, and source review are available?
100 texts in the main evaluation set; 31 additional texts for supplementary evaluation	Can a model select a line marked incorrect in a text?
100 proofs; 60 statements	How do instructions change the objections produced and the tokens used?
16 prescribed repairs	Can a model complete and prove a specified correction, with or without a supplied witness?
50 paired tasks per model	How does retrieving earlier objections affect proof development?
13 explanation packages	What further mathematics follows from every case in an earlier selected family-and-repair group?

Table 1: Each study reports results for its own question and evaluation set. The benchmark measures false-assertion identification, while the mathematical case studies establish structural theorems and verified repairs.

A comprehensive multistage human review. A human review must answer three questions affirmatively for a dossier to enter the retained core: does the objection accurately represent the original source claim and its assumptions; does the counterexample satisfy those assumptions and make the claim false; and does the Lean formalization correctly represent the counterexample.

Nine human reviewers contributed to this multistage process. First, an initial review of 100 dossiers identified specific source, witness, and Lean failure modes; this feedback informed a revised automated filtering prompt applied across 425 dossiers. Further random and targeted human reviews checked the revised material, and separate reference-line reviews corrected annotation labels. A final stratified audit of 100 dossiers assessed the earlier 284-dossier collection, yielding an estimated 91.0% approval rate (95% interval: 81.0%–96.5%) for that population.

The resulting 147-dossier core requires an applicable human review approving all three criteria, with no unresolved negative or uncertain human judgment or source-scope concern. Each retained dossier combines a Lean-checked refutation with human approval of its source interpretation, witness and formalization. Appendices N and D document the stages, review counts and version-specific inclusion decisions.

Checking counterexamples and grading references. In a study covering 604 executable tasks, separately written programs checked 351 proposed counterexamples, confirming 348 and rejecting three. A separate analysis examined grading references that source or counterexample review later found invalid. Model graders had credited 60 of 78 outputs as matching those references. These findings highlight the value of checking both proposed counterexamples and the references used to grade model responses.

The release also includes objections from previously vetted community answers and two model-generated proofs from QEDBench (Gonzalez et al., 2026) that had received perfect human rubric scores. Even expert reviewers can overlook an error. Explicit counterexamples and formal checks complement their judgment by providing inspectable evidence. Appendices C and T describe these examples and explain how they were selected.

4 Can an LLM Critic Find an Accepted Error Line?

4.1 The task and what its score means

We call the task *certified-claim localization*, abbreviated CCL. Here *localization* means selecting a location in the source text. Each stored reference counterexample has a Lean certificate.

Figure 1 illustrates the certified-claim localization task. A critic receives the numbered source text. The prompt states that the text contains an error and gives instructions to return up to three ranked candidate line numbers with a brief explanation. The prompt does not reveal the dossier's selected passage, witness, or Lean certificate.

The *reference lines*, also called *gold lines*, record the set of accepted error lines for each input. A selected line earns credit if it belongs to the input's reviewed set of accepted error lines; 57 of the 100 primary inputs have more than one accepted line. **Hit@1** is the percentage of inputs where the first choice earns credit. **Hit@3** allows a match anywhere in the top three. Missing answers, format errors, and budget exhaustion yield a score of zero. Invalid or duplicate labels do not advance later choices (Appendix P).

Inputs and comparisons. The main evaluation set contains 100 texts, each with at least ten nonblank source-text lines. Related inputs are grouped into 92 problem clusters. Appendix P describes the records excluded from scoring. We report 13 evaluated model configurations.

We compare model configurations using their recorded execution settings and computation budgets. Simple baseline rules select lines by length, word count, mathematical-symbol count, or similarity in word use to the full text (Salton & Buckley, 1988). We compare the models against the strongest of these rules. To calculate 95% confidence intervals, we repeatedly sample groups of related texts, keeping each group together (Efron, 1979; Field & Welsh, 2007). Table 27 gives the execution settings. Appendix P gives the statistical methods.

Numbered input (excerpt)
(10 nonblank numbered lines total)

[1] Q1: Suppose a function has the property that whenever "x" is in the domain of "f", then so is "-x". Show that f can be written as the sum of an even function and an odd function.

[3] By definition, an even function has the equation $f(x) = f(-x)$ and by definition, an odd function satisfies the equation $f(-x) = -f(x)$.

[5] Therefore, f can be written such that $f(x) = f(-x) + (-f(x))$.

[7] Applying the second definition above gives $f(x) = -f(x) + -f(x)$ or $f(x) = -2f(x)$.

...

Recorded GPT-OSS 20B selection

`"candidates": [5, 7]`
Score: Hit@1 = 1, Hit@3 = 1

Held-out reference evidence
(not shown to the critic)

Gold target: Line 5
Witness: $D = \{-1, 1\}$, $f \equiv 1$, $x = 1$. Then $f(1) = 1$ while $f(-1) + (-f(1)) = 1 - 1 = 0 \neq 1$.

Lean-checked statement:
The domain is closed under negation, 1 belongs to it, and $f(1) \neq f(-1) + (-f(1))$.

Figure 1: A primary localization input with a false intermediate deduction at line 5 (`reddit_learnmath_dm6itc`). The witness refutes line 5's intermediate identity; the even-plus-odd decomposition theorem is valid. The critic model does not see the held-out reference evidence. The critic selects the incorrect assertion at line 5 as its first choice.

Model configuration	First-choice match (%) Hit@1 [95% interval]	Top-three (%) Hit@3 [95% interval]	Recorded tokens (lower bound)	Usable out of 100
Sonnet 5	61.0 [52.0, 69.6]	76.0 [67.7, 83.7]	697,281	96
Opus 5.5	68.0 [59.8, 76.2]	85.0 [78.6, 91.0]	510,187	96
DeepSeek R1 Qwen	22.0 [14.4, 30.1]	28.0 [20.0, 36.4]	428,199	87
Gemma 3 27B	26.0 [18.3, 33.7]	44.0 [35.7, 52.1]	340,082	100
GLM 4.5 Air	10.0 [4.9, 16.0]	11.0 [5.8, 17.2]	1,658,280	23
GPT-OSS 120B	49.0 [39.6, 58.3]	60.0 [51.0, 69.0]	450,420	97
GPT-OSS 20B	43.0 [34.3, 52.0]	47.0 [37.8, 56.6]	708,462	89
Kimi Linear	14.0 [7.2, 21.4]	27.0 [19.0, 35.7]	241,035	94
Nemotron 3.5 Lightning	27.0 [19.0, 35.6]	27.0 [18.8, 35.4]	1,676,079	41
Nemotron 3 Super	48.0 [40.2, 56.1]	48.0 [40.2, 56.0]	740,935	90
Gemini 3.1 Pro	73.0 [65.1, 80.6]	85.0 [78.2, 91.2]	1,446,420	93
GPT-5.6 Luna	63.0 [54.6, 71.1]	74.0 [66.7, 80.8]	390,142	97
TF-IDF centroid similarity	18.0 [11.0, 25.3]	—	—	—
Random-line expectation	6.7 [5.7, 7.8]	—	—	—

Table 2: Reference selection on the 100 primary inputs within the human-approved core. Both match rates count unusable answers as misses. "Usable" counts answers that meet the execution and response rules. The token usage column reports recorded primary full tokens (lower bound). Results for all 13 configurations, token components, and failures appear in Appendix P and the public core release.

4.2 Results: finding the reference also requires returning a usable answer

Table 2 reports results for 12 model configurations on the 100-text core. Gemini 3.1 Pro achieves the highest observed Hit@1 score, selecting a line marked incorrect on its first choice in 73 of 100 texts (73.0%). The strongest rule-based baseline is word-profile similarity (TF-IDF centroid), matching 18 texts (18.0%), while random line selection has an expected score of 6.7%. Gemini and Opus 5.5 tie for the highest observed Hit@3 score, finding a reference line among their first three choices in 85 of 100 texts (85.0%). Considering all three choices reveals useful candidates below the first choice: Gemini rises from 73 to 85 matches, and Opus from 68 to 85. Finding an accepted error among several candidates and ranking it first are therefore distinct parts of this task.

Response failures remain a substantial part of the result. Nemotron 3.5 Lightning returned an answer meeting the protocol on only 41 inputs, exceeding the allowed budget on the others. Scoring all 100 texts measures how often the model both returns an answer within the budget and selects a line marked incorrect. Because each model's usable answers cover a different subset, conditional accuracies are

not directly matched comparisons. GPT-OSS 120B and 20B achieve 49 and 43 first-choice matches respectively, showing that smaller local configurations can recover many of these error lines. The 20B model matches the checked annotations on 25 more texts than the strongest rule-based baseline. This makes error finding a meaningful evaluation task for local as well as frontier models. Appendix P gives the complete results and execution settings.

4.3 Reference errors and other valid targets

Human reviewers explicitly checked the source assumptions, formal witness, and line correspondence. This confirmed that the reference lines accurately mark the accepted error. Exact human-review cohort intersections and detailed dispositions are listed in the appendix.

A saved earlier-cohort AI-assisted analysis examined 30 of the 58 Sonnet responses whose readable first-choice line missed the reference within the 191-input collection. The review judged that 12 selected another false assertion, 15 selected a line that was not false as written, and three exposed source or annotation problems. For the 12 responses identifying another false assertion, the review also judged the supplied counterexample mathematically sound in 11 cases. This analysis examines objections beyond the benchmark's designated reference lines.

Experiment B: LLM verification of saved earlier-cohort responses. Experiment B examined 48 saved responses drawn from the earlier cohort: one from Gemini 3.1 Pro and one from Nemotron 3 Super 120B for each of 24 sampled texts. Two LLM verifiers separately reviewed anonymized packets, checking whether each explanation challenged the selected claim (J1) and whether its mathematical evidence was valid (J3).

We ask whether each response both identifies the assertion it challenges and supplies valid evidence under that assertion's assumptions. Table 3 reports these joint endpoints after recorded LLM adjudications and source-scope checks. The original first-reader judgments chosen in advance remain in Appendix Q.3. Six of the 10 reference misses supplied valid objections under this joint criterion.

Model	Reference matches	Valid obj. among matches	Valid obj. among misses	Total valid objections
Gemini 3.1 Pro	20	18	2	20 / 24
Nemotron 3 Super 120B	18	15	4	19 / 24

Table 3: Secondary LLM-reader analysis of joint target identification and evidence validity on 24 paired texts. A counted valid objection is a protocol-valid response that challenges its selected claim and supplies mathematically valid evidence under the source assumptions, using recorded adjudications and source-scope checks. Appendix Q.3 reports the original first-reader validity judgements.

These model-assisted judgments assess the validity of the 48 saved objections. Appendix Q.3 reports reader disagreements, adjudications, and the bounds for comparing critics on this panel.

5 Complementary Studies: Instructions, Repairs, and Retrieval

Beyond identifying false assertions, our studies examine search instructions, proof correction, and the reuse of earlier objections. Sonnet 5 completed 15 of 16 repair tasks on the first attempt with a checked counterexample (found in BugBank via our search method), compared with 13 of 16 without one. Both conditions supplied the identified error, a correction template, and the statements to prove. Opus 5.5 succeeded on 15 of 16 tasks in both conditions. Appendix G and the linked study appendices provide the methods, results, and recorded token usage.

6 What Can We Learn After Finding One Error?

From single counterexamples to general theorems. Finding an isolated error is only the beginning of mathematical verification; the next step is to map exactly where the same failure occurs. By

varying an algorithm's input, scaling a graph's size, or adjusting a coloring's thresholds, we can identify the underlying structural pattern.

The three cases below demonstrate how this progression unfolds in practice. For the square-root algorithm, we characterize every failure and prove a one-comparison repair. Second, we generalize a specific counterexample graph into an exact structural characterization, yielding an explicit count of frozen graph colorings for an entire infinite family. The geometry of the counterexample provides a template that forces the local color choices into a globally rigid structure. Finally, we extend an integer-coloring counterexample into a sharp optimal limit on a combinatorial construction, establishing the maximum limiting fraction of ordered triples within a specific threshold family. Optimizing the limiting density over all threshold pairs proves the sharp bound within this two-threshold family.

These follow-ups illustrate the direct path from identifying an isolated error to proving a rigorous new theorem. The original graph and integer-coloring counterexamples were discovered during neutral model runs prior to their inclusion in BugBank; the subsequent mathematical extensions are author-led and AI-assisted. Their formal results have Lean certificates; Appendices B, K and S give the statements and proofs.

6.1 A SQUARE-ROOT ALGORITHM: FROM ONE FAILURE TO A COMPLETE CORRECTION

Source claim and counterexample. Let x be a nonnegative integer input. We analyze the final printed algorithm in Answer to "Relating a proof to a Haskell program" for computing its integer square root (the greatest integer whose square does not exceed x). The algorithm starts with answer zero for input zero and computes answers for successive integers. If its answer for $x - 1$ is b, it chooses b when $x \leq (b + 1)^2$, and $b + 1$ otherwise. The first failure is at $x = 1$, where the algorithm returns zero instead of one. The original algorithm increases its answer one input too late: it returns one at input four and reaches two at input five.

Mathematical advancement. The failure at $x = 1$ suggests a one-input delay: for positive x, the original answer equals the corrected answer at $x - 1$. Induction proves this identity and the corrected algorithm's correctness. Since the integer square root changes exactly at positive perfect squares, the delay identifies every failure. Earlier annotations noted this square pattern, and the source supplied the correction replacing $\leq$ with $<$; Appendices A and B give the proofs.

Proposition 1 (Exact failure set and repair)**.** *The corrected algorithm returns the integer square root of every nonnegative integer. For a positive input x, the original algorithm returns the integer square root of $x - 1$. Its errors therefore occur exactly at the positive perfect squares:* $1, 4, 9, 16, \ldots$.

6.2 GRAPH COLORINGS: A REPEATING LOCAL PATTERN EXPLAINS AN ENTIRE FAMILY

Source claim and counterexample. In a proper graph coloring, vertices joined by an edge must have different colors. A graph is *recolorable* if, for every allowed palette with more than the minimum number of colors, any proper coloring can be changed into any other by changing one vertex's color at a time while keeping adjacent vertices differently colored. A proper coloring is *frozen* if changing any single vertex to another of the available colors would give it the same color as a neighbor (Cereceda et al., 2008). A frozen coloring with an extra color prevents recolorability.

We examine a conjecture about preserving recolorability when vertices are replaced by cliques (sets of mutually adjacent vertices, called a *bag*). Join every vertex in each bag to every vertex in the neighboring bags. Replacing the n vertices of an n-cycle with 2-vertex clique bags creates $C_n[K_2]$, the doubled n-cycle. The source states that if every induced subgraph of a prime graph is recolorable, then all its clique blowups are also recolorable (Belavadi et al., 2026, Conjecture 1, §6); the originally reviewed input was arXiv v2 (Belavadi et al., 2024). Odd cycles of length at least five satisfy the antecedent. The counterexample uses $n = 9$, creating $C_9[K_2]$. This graph has 18 vertices, requires a minimum of 5 colors, and has a frozen 6-coloring, satisfying the source conditions but refuting the conjecture.

Mathematical advancement. The 18-vertex witness reveals a local constraint: every vertex must see all five alternative colors among its neighbors. Thus each three-bag window contains all six colors exactly once (Figure 2). Overlapping windows force the color pairs to repeat every three bags.

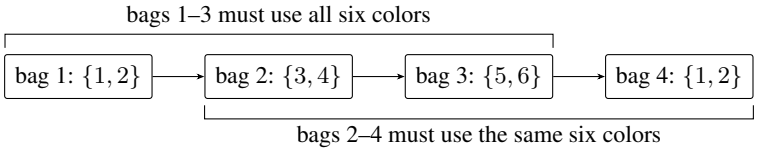


Figure 2: Why the color pairs must repeat every three bags. The two windows share bags 2 and 3, so bag 4 must use the same pair as bag 1. Arrows show the order of the bags, not directed graph edges. The numbers name six distinct colors.

Extending this constraint around the cycle determines which lengths work; counting the patterns gives the theorem below. Existence follows from Shaebani (2009); our advancement proves necessity and the exact count.

Proposition 2 (When the repeating pattern closes)**.** *Let $n \geq 5$ be odd. The doubled n-cycle has a frozen six-coloring exactly when n is a multiple of three. For each such n, there are exactly $90 \cdot 2^n$ such colorings when the vertices and six colors are labeled.*

There are $\binom{6}{2}\binom{4}{2} = 90$ ways to choose the color pairs for the first three bags. Each pair can then be assigned to its two labeled vertices in two ways, giving the factor 2^n.

Separately, we establish larger-bag families: $C_{11}[K_3]$ requires a minimum of 7 colors and has a frozen 8-coloring. For every odd $n \geq 9$, defining q as the least integer greater than or equal to $n/3$, replacing each cycle vertex with a q-vertex clique creates $C_n[K_q]$. This graph requires a minimum of $2q + 1 < n$ colors and has a frozen n-coloring. This shows the preservation implication fails for every odd $n \geq 9$.

6.3 Integer colorings: a better construction with a provable limit

Source claim and counterexample. Consider coloring the integers from one to n, where $n \geq 2$, with three colors. Consider all ordered Schur triples: triples $(x, y, x + y)$ with positive integers x, y and $x + y \leq n$. A triple is rainbow when its three entries have different colors. The goal is to maximize the fraction of rainbow triples as n grows. The source proposes that the maximum limiting fraction is $2/5$, or 40% (Parczyk & Spiegel, 2026).

The counterexample is a specific construction using two thresholds, $a = 4/11$ and $b = 10/11$. All even integers are red. An odd integer k is blue when $k/n \leq a$ or $k/n > b$, and green in between. For every positive integer m, using these optimal thresholds on the integers up to $n = 22m$ gives exactly $99m^2 + 3m$ rainbow triples. Dividing by the total $n(n-1)/2$ ordered triples and taking larger m gives the limit $9/22$, approximately 40.9%, exceeding the conjectured $2/5$. Independent work gives the same construction and lower bound (Hegde et al., 2026).

Mathematical advancement. The counterexample raises a further question: how much can this coloring rule achieve? We keep its even–odd structure and let the two thresholds vary. Counting triples gives a formula $D(a, b)$ for the limiting rainbow fraction over $0 \leq a \leq b \leq 1$. Maximizing this formula identifies the unique best thresholds, and examining changes around the optimum quantifies the loss from moving them.

Theorem 3 (The best thresholds for this construction)**.** *The unique maximum of $D(a, b)$ is $9/22$, approximately 40.9%, at $a = 4/11$ and $b = 10/11$. The simpler rules with $a = 0$ or $b = 1$, which omit one blue end block, reach at most $2/5$, or 40%.*

The theorem establishes $9/22$ as the largest limiting fraction attainable by this exact two-threshold construction. A higher fraction requires a different coloring rule. Appendices M and S.1 give the exact counts when $n = 22m$ and derive the limiting formula.

Mapping further families of counterexamples. Appendix K maps two spaces of constructions: cycle lengths and replacement-clique sizes determine where frozen colorings occur; the two integer-coloring thresholds determine the achievable density. Appendix M develops the same witness-to-family approach in other problems. In the supertoken graph $F_2(K_n)$, vertices represent two tokens,

possibly together, on a complete graph with n vertices. Six reference placements distinguish every placement at $n = 8$ by its distances to them. Keeping these references and adding the placement with both tokens on each new vertex gives resolving sets of size at most $n - 2$ for every $n \geq 8$, refuting the proposed metric dimension (Baskoro et al., 2025). A polynomial witness similarly leads to a coefficient obstruction for every $m \geq 4$ (Sun, 2026b). Further certificates establish a negative coefficient in the fixed-m positivity conjecture (Gatzweiler & Krattenthaler, 2025) and three infinite projective-cube-free families exceeding the proposed density bound (Meng, 2026); the $d = 5$ witness rediscovered a pullback of a published Long–Wagner set (Long & Wagner, 2019) (Appendix L). Appendix I also records 13 family-and-repair packages with explicit assumptions and Lean-checked proofs; three recover the intended final conclusion by repairing an intermediate argument.

7 Related Work and Open Questions

Relation to existing evaluations. FaithSieve (Wang et al., 2026d) breaks informal proofs into smaller claims and combines Lean checks with assessments of whether the formal statements match the original text. It uses these checks to locate the first error. ProofFlow (Cabral et al., 2026) tracks dependencies between proof steps and checks whether formal refutations identify errors in the written proof. Critique-Resilient Benchmarking (Marro et al., 2026) compares models through adversarial exchanges in which objections to generated answers are checked.

Recent step-level resources such as ProcessBench (Zheng et al., 2025), PRMBench (Song et al., 2025), and Hard2Verify (Pandit et al., 2026) evaluate step-level correctness or error localization in natural-language reasoning. Co-PEER's demonstrated advance is a reusable, human-reviewed resource that connects these local checks to the original mathematical claims. Each retained objection links a source passage and its assumptions to a counterexample, a Lean-checked refutation, and human approval of the source interpretation. This evidence supports both reproducible evaluation of LLM critics and mathematical follow-ups that turn individual counterexamples into classification, repair, and optimality theorems.

CounterMath studies counterexample-driven conceptual reasoning (Li et al., 2025), while Learning to Disprove (Li et al., 2026b) and Lean Disprove (Ondras & Freer, 2026) evaluate automated counterexample search. Co-PEER keeps each counterexample together with the Lean statement it verifies, the review of its connection to the source claim, and the mathematical conclusions it supports. FormalAlign and audits of formal benchmarks distinguish a correct formal proof from a faithful translation of a source text (Lu et al., 2024b; Ammanamanchi et al., 2026); Co-PEER applies this distinction to mathematical criticism and documents whether its checked refutations match the source's claims.

Appendix V gives the broader comparison, including connections to property testing, proof-carrying code, and AI-assisted construction.

Open questions. BugBank brings together selected mathematical objections, formal certificates, model-assisted assessments, and human reviews, including a blinded random audit. How well do critics identify errors, recognize correct arguments, and support their objections across a broader range of sources? How do models compare with matched inference budgets on previously unseen texts? Which forms of human–AI collaboration most effectively turn a counterexample into a correction, classification, or sharp bound? The released records and protocols provide a starting point for studying these questions.

Conclusion. Co-PEER turns counterexamples into inspectable evidence and starting points for mathematical discovery. BugBank connects 147 source-linked objections to witnesses, Lean proofs, and reviews of whether each objection applies to its source. Its 100-text benchmark measures how well LLM critics identify the challenged claims. The mathematical follow-ups show what can follow: exact error classifications, proved corrections, and a sharp optimum for a family of constructions. Together, these contributions provide a concrete path from finding an error to understanding its cause and proving something new.

AI USE STATEMENT

Language models contributed to source triage, counterexample search, executable checkers, Lean development, semantic reviews, qualitative coding, experimental grading, mathematical follow-ups, and manuscript revision. Their contributions included mathematical reasoning and interpretation of source claims. The records identify which assessments were made by human reviewers and which by language models. The analytic propositions in Appendix Q were developed with AI assistance and are established by the written proofs provided there. The authors take responsibility for the claims, citations, and released evidence.

ETHICS STATEMENT

Local objections are not blanket judgments about a source author's competence or the validity of an entire work. Source versions, prior corrections, known constructions, and limits of consequence are retained to avoid misleading discovery attribution. Reviewers' identities should remain deidentified in public evidence; source excerpts and redistributed artifacts require appropriate permissions and license compliance. This paper does not claim a representative estimate of source-platform error rates or an independent human endorsement of every retained record.

REPRODUCIBILITY STATEMENT

The instruction-study prompts, grading rubrics and token accounting are in Appendices O, H and F. Corpus construction and earlier audit procedures are in Appendices T and U. Current membership, review evidence, certificates, and score replay are documented in Appendix N and the anonymous repository. Earlier census annotations and coding checks are preserved in `censuses/`. Appendix S gives complete analytic proofs of the structural propositions.

Data availability. Code, data, Lean proofs, review evidence, and experimental results, with replay instructions and file manifests, are available in the anonymized repository at `https://anonymous.4open.science/r/co-peer-ECFB/README.md`.

REFERENCES

Tudor Achim, Alex Best, Alberto Bietti, Kevin Der, Mathïs Fédérico, Sergei Gukov, Daniel Halpern-Leistner, Kirsten Henningsgard, Yury Kudryashov, Alexander Meiburg, Martin Michelsen, Riley Patterson, Eric Rodriguez, Laura Scharff, Vikram Shanker, Vladmir Sicca, Hari Sowrirajan, Aidan Swope, Matyas Tamas, Vlad Tenev, Jonathan Thomm, Harold Williams, and Lawrence Wu. Aristotle: IMO-level automated theorem proving. *arXiv preprint arXiv:2510.01346*, 2025.

Mark Adams. HOL Zero's solutions for Pollack-inconsistency. In *Interactive Theorem Proving (ITP 2016)*, volume 9807 of *Lecture Notes in Computer Science*, pp. 20–35, 2016. doi: 10.1007/978-3-319-43144-4_2. URL `https://doi.org/10.1007/978-3-319-43144-4_2`.

Ammar Ahmed, Azal Ahmad Khan, Ayaan Ahmad, Sheng Di, Zirui Liu, and Ali Anwar. Retrieval-of-Thought: Efficient reasoning via reusing thoughts. *arXiv preprint arXiv:2509.21743*, 2025. ICLR 2026.

Musa Al-hassy. Answer to "relating a proof to a Haskell program". Computer Science Stack Exchange, answer 97647, 2018. URL `https://cs.stackexchange.com/a/97647`. Posted 22 September 2018.

Lisa Alazraki, Maximilian Mozes, Jon Ander Campos, Tan Yi-Chern, Marek Rei, and Max Bartolo. No need for explanations: LLMs can implicitly learn from mistakes in-context. *arXiv preprint arXiv:2502.08550*, 2025. EMNLP 2025.

Eyad Alkassar, Sascha Böhme, Kurt Mehlhorn, and Christine Rizkallah. Verification of certifying computations. Computer Aided Verification (CAV), 2011. `https://doi.org/10.1007/978-3-642-22110-1_7`.

Miltiadis Allamanis and Pengcheng Yin. Disproving program equivalence with LLMs. *arXiv preprint arXiv:2502.18473*, 2025.

Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4): 181–185, October 1985. doi: 10.1016/0020-0190(85)90056-0. URL https://www.cs.cornell.edu/fbs/publications/DefLiveness.pdf.

AlphaProof and AlphaGeometry teams. AI achieves silver-medal standard solving International Mathematical Olympiad problems. DeepMind blog, https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/, 2024.

Pawan Sasanka Ammanamanchi, Siddharth Bhat, and Stella Biderman. Faults in our formal benchmarking: Dataset defects and evaluation failures in Lean theorem proving. *arXiv preprint arXiv:2606.29493*, 2026. URL https://arxiv.org/abs/2606.29493.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Daneshvar Amrollahi, Jerry Lopez, and Clark Barrett. Faithful autoformalization via roundtrip verification and repair. *arXiv preprint arXiv:2604.25031*, 2026.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Learning from mistakes makes LLM better reasoner. *arXiv preprint arXiv:2310.20689*, 2023.

James H. Andrews, Lionel C. Briand, and Yvan Labiche. Is mutation an appropriate tool for testing experiments? Proceedings of the 27th International Conference on Software Engineering (ICSE 2005), 2005. https://dl.acm.org/doi/10.1145/1062455.1062530.

Cem Anil, Guodong Zhang, Yuhuai Wu, and Roger Grosse. Learning to give checkable answers with prover-verifier games. *arXiv preprint arXiv:2108.12099*, 2021. URL https://arxiv.org/abs/2108.12099.

Andrew W. Appel, Neophytos G. Michael, Aaron Stump, and Roberto Virga. A trustworthy proof checker. Journal of Automated Reasoning, 2003. https://doi.org/10.1023/B:JARS.0000021013.61329.58.

Negar Arabzadeh, Wenjie Ma, Sewon Min, and Matei Zaharia. RAG over thinking traces can improve reasoning tasks. *arXiv preprint arXiv:2605.03344*, 2026.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023. URL https://arxiv.org/abs/2302.12433.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*, 2025. URL https://arxiv.org/abs/2503.11926.

Agustín Balestra, Agustín Nolasco, Facundo Molina, Diego Garbervetsky, Renzo Degiovanni, and Nazareno Aguirre. Improving dynamic specification inference with LLM-generated counterexamples. *arXiv preprint arXiv:2604.10761*, 2026. URL https://arxiv.org/abs/2604.10761.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025. URL https://arxiv.org/abs/2505.23281.

Slim Barkallah, Luke Bailey, Kaiyue Wen, Mohammed Abouzaid, and Tengyu Ma. Pseudo-formalization for automatic proof verification. *arXiv preprint arXiv:2605.20531*, 2026.

Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. The oracle problem in software testing: A survey. IEEE Transactions on Software Engineering, 2015. https://dl.acm.org/doi/10.1109/TSE.2014.2372785.

Massimo Bartoletti and Enrico Lipparini. Neuroforger: certified violation witnesses for smart contracts verification via LLMs. *arXiv preprint arXiv:2605.31389*, 2026.

E. T. Baskoro, C. Dalfó, M. A. Fiol, and R. Simanjuntak. On some metric properties of supertoken graphs. *Utilitas Mathematica*, 123:19–33, 2025. doi: 10.61091/um123-02. URL https://doi.org/10.61091/um123-02.

Matthias Bastian. Leading OpenAI researcher announced a GPT-5 math breakthrough that never happened. The Decoder (news report), 2025. https://the-decoder.com/leading-openai-researcher-announced-a-gpt-5-math-breakthrough-that-never-happened/.

Joeran Beel, Min-Yen Kan, and Moritz Baumgart. Evaluating Sakana's AI Scientist: Bold claims, mixed results, and a promising future? *arXiv preprint arXiv:2502.14297*, 2025.

Manoj Belavadi, Kathie Cameron, and Ni Luh Dewi Sintiari. Recoloring via modular decomposition, 2024. URL https://arxiv.org/abs/2405.06446v2. Version 2, 14 November 2024; Figure 3 and Conjecture 1 in Section 6.

Manoj Belavadi, Kathie Cameron, and Ni Luh Dewi Sintiari. Recoloring via modular decomposition. *Journal of Graph Theory*, 111(4):113–123, 2026. doi: 10.1002/jgt.70005. URL https://onlinelibrary.wiley.com/doi/10.1002/jgt.70005.

Joe Benton, Misha Wagner, Eric Christiansen, Cem Anil, Ethan Perez, Jai Srivastav, Esin Durmus, Deep Ganguli, Shauna Kravec, Buck Shlegeris, Jared Kaplan, Holden Karnofsky, Evan Hubinger, Roger Grosse, Samuel R. Bowman, and David Duvenaud. Sabotage evaluations for frontier models. *arXiv preprint arXiv:2410.21514*, 2024.

Adam Betts, Nathan Chong, Alastair Donaldson, Shaz Qadeer, and Paul Thomson. GPUVerify: a verifier for GPU kernels. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA)*, pp. 113–132, 2012.

Federico Bianchi, Yongchan Kwon, Zachary Izzo, Linjun Zhang, and James Zou. To err is human: Systematic quantification of errors in published AI papers via LLM analysis. *arXiv preprint arXiv:2512.05925*, 2025.

Jasmin Christian Blanchette and Tobias Nipkow. Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In *Interactive Theorem Proving (ITP 2010)*, volume 6172 of *Lecture Notes in Computer Science*, pp. 131–146. Springer, 2010. doi: 10.1007/978-3-642-14052-5_11.

Matthew Bolan, Joachim Breitner, Jose Brox, Nicholas Carlini, Mario Carneiro, Floris van Doorn, Martin Dvorak, Andrés Goens, Aaron Hill, Harald Husum, Hernán Ibarra Mejia, Zoltan A. Kocsis, Bruno Le Floch, Amir Livne Bar-on, Lorenzo Luccioli, Douglas McNeil, Alex Meiburg, Pietro Monticone, Pace P. Nielsen, Emmanuel Osalotioman Osazuwa, Giovanni Paolini, Marco Petracci, Bernhard Reinke, David Renshaw, Marcus Rossel, Cody Roux, Jérémy Scanvic, Shreyas Srinivas, Anand Rao Tadipatri, Terence Tao, Vlad Tsyrklevich, Fernando Vaquerizo-Villar, Daniel Weber, and Fan Zheng. The Equational Theories Project: Advancing collaborative mathematical research at scale. *arXiv preprint arXiv:2512.07087*, 2025.

Nicholas Boucher and Ross Anderson. Trojan Source: Invisible vulnerabilities. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 6507–6524. USENIX Association, 2023. URL https://www.usenix.org/conference/usenixsecurity23/presentation/boucher.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Sébastien Bubeck, Christian Coester, Ronen Eldan, Timothy Gowers, Yin Tat Lee, Alexandru Lupsasca, Mehtaab Sawhney, Robert Scherrer, Mark Sellke, Brian K. Spears, Derya Unutmaz, Kevin Weil, Steven Yin, and Nikita Zhivotovskiy. Early science acceleration experiments with GPT-5. *arXiv preprint arXiv:2511.16072*, 2025.

Kevin Buzzard. What is the point of computers? a question for pure mathematicians. *arXiv preprint arXiv:2112.11598*, 2021.

Rafael Cabral, Tuan Manh Do, Xuejun Yu, Wai Ming Tai, Zijin Feng, and Xin Shen. ProofFlow: A dependency graph approach to faithful proof autoformalization. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/5fce5198dbf92d5ff45f74504431e2ff-Paper-Conference.pdf.

Ahmet Çelik, Karl Palmskog, Marinela Parovic, Emilio Jesús Gallego Arias, and Milos Gligoric. Mutation analysis for Coq. Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019. https://2019.ase-conferences.org/details/ase-2019-papers/90/Mutation-Analysis-for-Coq.

Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Connectedness of the graph of vertex-colourings. *Discrete Mathematics*, 308(5–6):913–919, 2008. doi: 10.1016/j.disc.2007.07.028. URL https://doi.org/10.1016/j.disc.2007.07.028.

Jonathan Chappelon, Jorge L. Ramírez Alfonsín, and Dumitru I. Stamate. Numerical semigroup tree: type-representation, 2025. URL https://arxiv.org/html/2507.15006v1. Version 1, Conjecture 4.4.

François Charton, Jordan S. Ellenberg, Adam Zsolt Wagner, and Geordie Williamson. PatternBoost: Constructions in mathematics with a little help from AI. *arXiv preprint arXiv:2411.00566*, 2024.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. *arXiv preprint arXiv:2207.10397*, 2022. ICLR 2023.

Leheng Chen, Zihao Liu, Wanyi He, and Bin Dong. Iteris: Agentic research loops for computational mathematics, 2026a. URL https://arxiv.org/abs/2606.02484v1.

Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, Cheng Ren, Jiawei Shen, Wenlei Shi, Tong Sun, He Sun, Jiahui Wang, Siran Wang, Zhihong Wang, Chenrui Wei, Shufa Wei, Yonghui Wu, Yuchen Wu, Yihang Xia, Huajian Xin, Fan Yang, Huaiyuan Ying, Hongyi Yuan, Zheng Yuan, Tianyang Zhan, Chi Zhang, Yue Zhang, Ge Zhang, Tianyun Zhao, Jianqiu Zhao, Yichi Zhou, and Thomas Hanwen Zhu. Seed-Prover: Deep and broad reasoning for automated theorem proving. *arXiv preprint arXiv:2507.23726*, 2025a.

Shuze Chen, Kunal Marwaha, Xiaoyang Lu, Henry Yuen, and Tianyi Peng. Prove2Me: An open collaborative platform for scaling math formalization. *arXiv preprint arXiv:2608.28433*, 2026b. URL https://arxiv.org/abs/2608.28433. https://prove2.me.

Tsong Y Chen, Shing C Cheung, and Siu Ming Yiu. Metamorphic testing: A new approach for generating next test cases. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, 1998.

Yongqiang Chen, Gang Niu, James Cheng, Bo Han, and Masashi Sugiyama. When and why does multi-agent debate fail and does it really underperform? *arXiv preprint arXiv:2510.20963*, 2025b.

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. FacTool: Factuality detection in generative AI – a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*, 2023.

Yuri Chervonyi, Trieu H. Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang Nguyen, Marcelo Menegali, Junehyuk Jung, Junsu Kim, Vikas Verma, Quoc V. Le, and Thang Luong. Gold-medalist performance in solving olympiad geometry with AlphaGeometry2. *arXiv preprint arXiv:2502.03544*, 2025.

Koen Claessen and John Hughes. QuickCheck: a lightweight tool for random testing of Haskell programs. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP)*, pp. 268–279, 2000. doi: 10.1145/351240.351266.

Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. Computer Aided Verification, 12th International Conference, CAV 2000, 2000. https://link.springer.com/chapter/10.1007/10722167_15.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Katherine M. Collins, Albert Q. Jiang, Simon Frieder, Lionel Wong, Miri Zilka, Umang Bhatt, Thomas Lukasiewicz, Yuhuai Wu, Joshua B. Tenenbaum, William Hart, Timothy Gowers, Wenda Li, Adrian Weller, and Mateja Jamnik. Evaluating language models for mathematics through interactions. Proceedings of the National Academy of Sciences, 2024. https://www.pnas.org/doi/10.1073/pnas.2318124121.

Simon Colton and Alison Pease. The TM system for repairing non-theorems. *Electronic Notes in Theoretical Computer Science*, 125(3):87–101, 2005. doi: 10.1016/j.entcs.2004.07.021.

Contributors to the Erdős Problems AI Wiki. AI contributions to Erdős problems. GitHub wiki, teorth/erdosproblems, 2026. URL https://github.com/teorth/erdosproblems/wiki/AI-contributions-to-Erd%C5%91s-problems. Community-maintained wiki; data as of June 30, 2026; accessed September 25, 2026.

Luís Cruz-Filipe, Marijn J. H. Heule, Warren A. Hunt, Jr., Matt Kaufmann, and Peter Schneider-Kamp. Efficient certified RAT verification. Automated Deduction – CADE 26, 2017. https://link.springer.com/chapter/10.1007/978-3-319-63046-5_14.

Nigel Cutland. *Computability: An Introduction to Recursive Function Theory*. Cambridge University Press, 1980. ISBN 9780521294652. doi: 10.1017/CBO9781139171496. URL https://doi.org/10.1017/CBO9781139171496.

Arghavan Moradi Dakhel, Amin Nikanjam, Vahid Majdinasab, Foutse Khomh, and Michel C. Desmarais. Effective test generation using pre-trained large language models and mutation testing. *arXiv preprint arXiv:2308.16557*, 2023.

Randy Davila. Automated conjecturing with *TxGraffiti*. *arXiv preprint arXiv:2409.19379*, 2024.

Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 337–340, 2008.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *International Conference on Automated Deduction (CADE)*, pp. 625–635, 2021. doi: 10.1007/978-3-030-79876-5_37.

Jasper Dekoninck, Ivo Petrov, Kristian Minchev, Mislav Balunovic, Martin Vechev, Miroslav Marinov, Maria Drencheva, Lyuba Konova, Milen Shumanov, Kaloyan Tsvetkov, Nikolay Drenchev, Lazar Todorov, Kalina Nikolova, Nikolay Georgiev, Vanesa Kalinkova, and Margulan Ismoldayev. The Open Proof Corpus: A large-scale study of LLM-generated mathematical proofs. *arXiv preprint arXiv:2506.21621*, 2025. ICLR 2026.

Carson Denison, Monte MacDiarmid, Fazl Barez, David Duvenaud, Shauna Kravec, Samuel Marks, Nicholas Schiefer, Ryan Soklaski, Alex Tamkin, Jared Kaplan, Buck Shlegeris, Samuel R. Bowman, Ethan Perez, and Evan Hubinger. Sycophancy to subterfuge: Investigating reward-tampering in large language models. *arXiv preprint arXiv:2406.10162*, 2024.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. Findings of the Association for Computational Linguistics: ACL 2024, 2024. https://aclanthology.org/2024.findings-acl.212/.

Elizabeth Dinella, Gabriel Ryan, Todd Mytkowicz, and Shuvendu K. Lahiri. TOGA: A neural method for test oracle generation. *arXiv preprint arXiv:2109.09262*, 2021. ICSE 2022.

Long Doan and ThanhVu Nguyen. AI-assisted autoformalization of combinatorics problems in proof assistants. In *IEEE/ACM International Conference on Software Engineering: New Ideas and Emerging Results*, pp. 1–5, 2025. URL https://roars.dev/pubs/doan2025ai.pdf.

Steven T. Dougherty, Suat Karadeniz, and Bahattin Yildiz. The automorphism group of the doubly-even [72,36,16] code can only be of order 1, 3 or 5, 2013. URL https://arxiv.org/abs/1303.4920v1. Version 1, Theorem 3.3; withdrawn in version 2.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023. ICML 2024.

J. E. Dunbar, S. M. Hedetniemi, S. T. Hedetniemi, D. P. Jacobs, J. Knisely, R. C. Laskar, and D. F. Rall. Fall colorings of graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 33:257–273, 2000. URL https://combinatorialpress.com/article/jcmcc/Volume%20033/vol-033-paper%2018.pdf.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979. doi: 10.1214/aos/1176344552.

Siemion Fajtlowicz. On conjectures of Graffiti. *Discrete Mathematics*, 72(1–3):113–118, 1988.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. Nature, 2024. https://www.nature.com/articles/s41586-024-07421-0.

Tony Feng, Junehyuk Jung, Sang hyun Kim, Carlo Pagano, Sergei Gukov, Chiang-Chiang Tsai, David Woodruff, Adel Javanmard, Aryan Mokhtari, Dawsen Hwang, Yuri Chervonyi, Jonathan N. Lee, Garrett Bingham, Trieu H. Trinh, Vahab Mirrokni, Quoc V. Le, and Thang Luong. Aletheia tackles FirstProof autonomously. *arXiv preprint arXiv:2602.21201*, 2026a.

Tony Feng, Trieu H. Trinh, Garrett Bingham, Dawsen Hwang, Yuri Chervonyi, Junehyuk Jung, Joonkyung Lee, Carlo Pagano, Sang hyun Kim, Federico Pasqualotto, Sergei Gukov, Jonathan N. Lee, Junsu Kim, Kaiying Hou, Golnaz Ghiasi, Yi Tay, YaGuang Li, Chenkai Kuang, Yuan Liu, Hanzhao Lin, Evan Zheran Liu, Nigamaa Nayakanti, Xiaomeng Yang, Heng-Tze Cheng, Demis Hassabis, Koray Kavukcuoglu, Quoc V. Le, and Thang Luong. Towards autonomous mathematics research. *arXiv preprint arXiv:2602.10177*, 2026b.

Christopher A. Field and Alan H. Welsh. Bootstrapping clustered data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(3):369–390, 2007. doi: 10.1111/j.1467-9868.2007.00593.x.

Anna Filighera, Tim Steuer, and Christoph Rensing. Fooling automatic short answer grading systems. Proceedings of the 21st International Conference on Artificial Intelligence in Education (AIED), 2020. https://doi.org/10.1007/978-3-030-52237-7_15.

Jean-Christophe Filliâtre and Andrei Paskevich. Why3 — where programs meet provers. In *European Symposium on Programming (ESOP)*, pp. 125–128, 2013.

Yuval Filmus. Answer to "why is this sequence of recurrence relevant?". Computer Science Stack Exchange, answer 13122, 2013. URL https://cs.stackexchange.com/a/13122. Posted 7 July 2013.

Emily First, Markus N Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-proof generation and repair with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pp. 1229–1241, 2023. doi: 10.1145/3611643.3616243.

Michael Freeze and Wolfgang A. Schmid. Remarks on a generalization of the Davenport constant. *Discrete Mathematics*, 310(23):3373–3389, 2010. doi: 10.1016/j.disc.2010.07.028. URL https://www.math.univ-paris13.fr/~schmid/personal/schmid_28t.pdf.

Jonathan Gabor, Jayson Lynch, and Jonathan Rosenfeld. EvilGenie: A reward hacking benchmark. *arXiv preprint arXiv:2511.21654*, 2025.

Bofei Gao, Zefan Cai, Runxin Xu, Peiyi Wang, Ce Zheng, Runji Lin, Keming Lu, Dayiheng Liu, Chang Zhou, Wen Xiao, Tianyu Liu, and Baobao Chang. LLM critics help catch bugs in mathematics: Towards a better mathematical verifier with natural language feedback. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 14588–14604. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.753. URL https://aclanthology.org/2025.findings-acl.753/.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. *arXiv preprint arXiv:2210.10760*, 2022. ICML 2023.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. RARR: Researching and revising what language models say, using language models. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2023a. https://aclanthology.org/2023.acl-long.910/.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023b. https://aclanthology.org/2023.emnlp-main.398/.

Mona Gatzweiler and Christian Krattenthaler. A positivity conjecture for a quotient of q-binomial coefficients, 2025. URL https://arxiv.org/pdf/2502.06032v3. Version 3, Conjecture 4.

Bogdan Georgiev, Javier Gómez-Serrano, Terence Tao, and Adam Zsolt Wagner. Mathematical exploration and discovery at scale. *arXiv preprint arXiv:2511.02864*, 2025.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, Natalie Stewart, Bogdan Grechuk, Tetiana Grechuk, Shreepranav Varma Enugandla, and Mark Wildon. FrontierMath: A benchmark for evaluating advanced mathematical reasoning in AI. *arXiv preprint arXiv:2411.04872*, 2024.

Max Glockner, Yufang Hou, Preslav Nakov, and Iryna Gurevych. Missci: Reconstructing fallacies in misrepresented science. *arXiv preprint arXiv:2406.03181*, 2024. ACL 2024.

Santiago Gonzalez, Alireza Amiri Bavandpour, Peter Ye, Edward Zhang, Ruslans Aleksejevs, Todor Antić, Polina Baron, Sujeet Bhalerao, Shubhrajit Bhattacharya, Zachary Burton, John Byrne, Hyungjun Choi, Nujhat Ahmed Disha, Koppány István Encz, Yuchen Fang, Robert Joseph George, Ebrahim Ghorbani, Alan Goldfarb, Jing Guo, Meghal Gupta, Stefano Huber, Annika Kanckos, Minjung Kang, Hyun Jong Kim, Dino Lorenzini, Levi Lorenzo, Tianyi Mao, Giovanni Marzenta, Ariane M. Masuda, Lukas Mauth, Ana Mickovic, Andrés Miniguano-Trujillo, Antoine Moulin, Wenqi Ni, Tomos Parry, Kevin Ren, Hossein Roodbarani, Mathieu Rundström, Manjil Saikia, Detchat Samart, Rebecca Steiner, Connor Stewart, Dhara Thakkar, Jeffrey Tse, Vasiliki Velona, Yunhai Xiang, Sibel Yalçın, Jun Yan, Ji Zeng, Arman Cohan, and Quanquan C. Liu. QEDBENCH: Quantifying the alignment gap in automated evaluation of university-level mathematical proofs. *arXiv preprint arXiv:2602.20629*, 2026. URL https://arxiv.org/abs/2602.20629v3. arXiv:2602.20629v3, 6 July 2026.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023. ICLR 2024.

Sébastien Gouëzel and Vladimir Shchur. A corrected quantitative version of the Morse lemma. *arXiv preprint arXiv:1810.04579*, 2018. Journal of Functional Analysis 277(4).

Zhengtao Gui, Sheng Yang, and Zhouxing Shi. Evaluating the robustness of proof autoformalization in Lean 4, 2026. URL https://arxiv.org/abs/2606.14867.

Dadi Guo, Jiayu Liu, Zhiyuan Fan, Zhitao He, Haoran Li, Yuxin Li, Yumeng Wang, and Yi R. Fung. Mathematical proof as a litmus test: Revealing failure modes of advanced large reasoning models. *arXiv preprint arXiv:2506.17114*, 2025a.

Jiaxing Guo, Wenjie Yang, Shengzhong Zhang, Tongshan Xu, Lun Du, Da Zheng, and Zengfeng Huang. Right is not enough: The pitfalls of outcome supervision in training LLMs for math reasoning. *arXiv preprint arXiv:2506.06877*, 2025b.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. DS-Agent: Automated data science by empowering large language models with case-based reasoning. *arXiv preprint arXiv:2402.17453*, 2024. ICML 2024.

Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, Quang Truong Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, Thi Hoai An Ta, Nam Trung Tran, Thi Diep Trieu, Josef Urban, Ky Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture. Forum of Mathematics, Pi, 2017. https://doi.org/10.1017/fmp.2017.1.

Richard H. Hammack and Gregory D. Smith. Cycle bases of reduced powers of graphs. *Ars Mathematica Contemporanea*, 12(1):183–203, 2017. doi: 10.26493/1855-3974.856.4d2. URL https://doi.org/10.26493/1855-3974.856.4d2.

Hojae Han, Jongyoon Kim, Sanghyeok Park, Dongwook Cheon, Yeachan Park, Myeong Jae Jeon, Sunjong Choe, Soonho Kong, Wonseok Hur, Seung won Hwang, and Donghoon Hyeon. ShadowBench: Toward reliable automatic evaluation of semantic alignment in autoformalization, 2026. URL https://arxiv.org/abs/2608.29270. To appear at EMNLP 2026.

Frank Harary and Robert A. Melter. On the metric dimension of a graph. *Ars Combinatoria*, 2: 191–195, 1976.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Kostas Hatalis, Despina Christou, and Vyshnavi Kondapalli. Review of case-based reasoning for LLM agents: Theoretical foundations, architectural components, and cognitive integration. *arXiv preprint arXiv:2504.06943*, 2025.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zhongyuan Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, and Bo Zheng. Can large language models detect errors in long chain-of-thought reasoning? *arXiv preprint arXiv:2502.19361*, 2025. ACL 2025.

Swaroop Hegde, Hitesh Kumar, and Pratibha. A somewhat sure note on an un-Schur problem, 2026. URL https://arxiv.org/abs/2609.18474v1. Version 1, submitted 16 September 2026.

Lukas Helff, Quentin Delfosse, David Steinmann, Ruben Härle, Hikaru Shindo, Patrick Schramowski, Wolfgang Stammer, Kristian Kersting, and Felix Friedrich. LLMs gaming verifiers: RLVR can lead to reward hacking. *arXiv preprint arXiv:2604.15149*, 2026.

Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek. Solving and verifying the boolean Pythagorean triples problem via Cube-and-Conquer. *arXiv preprint arXiv:1605.00723*, 2016. SAT 2016.

Jiewen Hu, Thomas Zhu, and Sean Welleck. miniCTX: Neural theorem proving with (long-)contexts. *arXiv preprint arXiv:2408.03350*, 2024a. ICLR 2025.

Xiangkun Hu, Dongyu Ru, Lin Qiu, Qipeng Guo, Tianhang Zhang, Yang Xu, Yun Luo, Pengfei Liu, Yue Zhang, and Zheng Zhang. RefChecker: Reference-based fine-grained hallucination checker and benchmark for large language models. *arXiv preprint arXiv:2405.14486*, 2024b.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, Ottavia Bertolli, Tom Zahavy, Amol Mandhane, Jessica Yung, Iuliya Beloshapka, Borja Ibarz, Vivek Veeriah, Lei Yu, Oliver Nash, Paul Lezeau, Salvatore Mercuri, Calle Sönne, Bhavik Mehta, Alex Davies, Daniel Zheng, Fabian Pedregosa, Yin Li, Ingrid von Glehn, Mark Rowland, Samuel Albanie, Ameya Velingker, Simon Schmitt, Edward Lockhart, Edward Hughes, Henryk Michalewski, Nicolas Sonnerat, Demis Hassabis, Pushmeet Kohli, and David Silver. Olympiad-level formal mathematical reasoning with reinforcement learning. Nature, 2025. https://www.nature.com/articles/s41586-025-09833-y.

hypnopump. Counterexample to Conjecture 10 of Meng, "On cube-free problems". GitHub gist, revision 664769b, July 2026. URL https://gist.github.com/hypnopump/8a369af584796b25c9af586b4fe39385.

Matthew Inglis, Juan Pablo Mejia-Ramos, Keith Weber, and Lara Alcock. On mathematicians' different standards when evaluating elementary proofs. Topics in Cognitive Science, 2013. https://onlinelibrary.wiley.com/doi/10.1111/tops.12019.

Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.

Alon Jacovi, Yonatan Bitton, Bernd Bohnet, Jonathan Herzig, Or Honovich, Michael Tseng, Michael Collins, Roee Aharoni, and Mor Geva. A chain-of-thought is as strong as its weakest link: A benchmark for verifiers of reasoning chains. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2024. https://aclanthology.org/2024.acl-long.254/.

Rajesh Jayaram, Drew Tyler, David Woodruff, Corinna Cortes, Yossi Matias, Vahab Mirrokni, and Vincent Cohen-Addad. Towards automating scientific review with Google's Paper Assistant Tool. *arXiv preprint arXiv:2606.28277*, 2026. URL https://arxiv.org/abs/2606.28277.

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

Albert Q. 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. *arXiv preprint arXiv:2210.12283*, 2022. ICLR 2023.

Dongwei Jiang, Jingyu Zhang, Orion Weller, Nathaniel Weir, Benjamin Van Durme, and Daniel Khashabi. SELF-[IN]CORRECT: LLMs struggle with discriminating self-generated responses. *arXiv preprint arXiv:2404.04298*, 2024.

Eric Jiang, Xiao Liang, Yikai Zhang, Yingjia Wan, Mengting Li, Haikang Deng, Alexander K. Taylor, Justin Baker, Rushil Raghavan, Junyi Zhang, Ying Nian Wu, Andrea L. Bertozzi, Kai-Wei Chang, Raghu Meka, Matthew Sottile, Nanyun Peng, Amit Sahai, Terence Tao, and Wei Wang. From solvers to research: Large language model-driven formal mathematics at the research frontier. *arXiv preprint arXiv:2607.07779*, 2026.

Lucas Jing, Xinqi Wang, Liao Zhang, and Simon S. Du. PBT-Bench: Benchmarking AI agents on property-based testing. *arXiv preprint arXiv:2605.15229*, 2026.

Haocheng Ju, Guoxiong Gao, Jiedong Jiang, Bin Wu, Zeming Sun, Shurui Liu, Leheng Chen, Yutong Wang, Yuefeng Wang, Zichen Wang, Wanyi He, Peihao Wu, Liang Xiao, Ruochuan Liu, Bryan Dai, and Bin Dong. Automated conjecture resolution with formal verification. *arXiv preprint arXiv:2604.03789*, 2026.

Jaehun Jung, Faeze Brahman, and Yejin Choi. Trust or escalate: LLM judges with provable guarantees for human agreement. *arXiv preprint arXiv:2407.18370*, 2024. ICLR 2025.

René Just, Darioush Jalali, and Michael D Ernst. Defects4J: A database of existing faults to enable controlled testing studies for Java programs. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)*, pp. 437–440, 2014a.

René Just, Darioush Jalali, Laura Inozemtseva, Michael D. Ernst, Reid Holmes, and Gordon Fraser. Are mutants a valid substitute for real faults in software testing? Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), 2014b. https://dl.acm.org/doi/10.1145/2635868.2635929.

Ryo Kamoi, Sarkar Snigdha Sarathi Das, Renze Lou, Jihyun Janice Ahn, Yilun Zhao, Xiaoxin Lu, Nan Zhang, Yusen Zhang, Ranran Haoran Zhang, Sujeeth Reddy Vummanthala, Salika Dave, Shaobo Qin, Arman Cohan, Wenpeng Yin, and Rui Zhang. Evaluating LLMs at detecting errors in LLM responses. *arXiv preprint arXiv:2404.03602*, 2024a. COLM 2024.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs. Transactions of the Association for Computational Linguistics, 2024b. https://aclanthology.org/2024.tacl-1.78/.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Sarkar Snigdha Sarathi Das, Ranran Haoran Zhang, Wenpeng Yin, and Rui Zhang. Efficient PRM training data synthesis via formal verification. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8246–8265, 2026. doi: 10.18653/v1/2026.findings-acl.403. URL https://aclanthology.org/2026.findings-acl.403/.

Amin Karbasi, Omar Montasser, John Sous, and Grigoris Velegkas. (im)possibility of automated hallucination detection in large language models. *arXiv preprint arXiv:2504.17004*, 2025.

Omar Kettani. An algorithm for solving the independent set problem. *arXiv preprint arXiv:0709.2689*, 2007. URL https://arxiv.org/abs/0709.2689v1. Claim 1 in version 1; version 3 withdrawn owing to an error in Claim 1 (https://arxiv.org/abs/0709.2689v3).

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

Akbir Khan, John Hughes, Dan Valentine, Laura Ruis, Kshitij Sachan, Ansh Radhakrishnan, Edward Grefenstette, Samuel R. Bowman, Tim Rocktäschel, and Ethan Perez. Debating with more persuasive LLMs leads to more truthful answers. *arXiv preprint arXiv:2402.06782*, 2024. ICML 2024.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *arXiv preprint arXiv:2303.17491*, 2023. NeurIPS 2023.

Kyuhee Kim, Auguste Poiroux, and Antoine Bosselut. Do LLMs game formalization? evaluating faithfulness in logical reasoning. *arXiv preprint arXiv:2604.19459*, 2026.

Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-C: A software analysis perspective. *Formal Aspects of Computing*, 27(3):573–609, 2015.

Jan Hendrik Kirchner, Yining Chen, Harri Edwards, Jan Leike, Nat McAleese, and Yuri Burda. Prover-verifier games improve legibility of LLM outputs. *arXiv preprint arXiv:2407.13692*, 2024.

Lingkai Kong, Zijian Wu, Yuzhe Gu, Haiteng Zhao, Wenyong Huang, Shuang Sun, Zhicheng Xiong, Xiaotian Zhang, Shuya Zhao, Yan Wang, Disheng Xu, Wenwei Zhang, and Kai Chen. AdvancedMathBench: A benchmark suite for advanced mathematical proof generation and verification. *arXiv preprint arXiv:2607.11849*, 2026. URL https://arxiv.org/abs/2607.11849.

Daniel Kroening and Michael Tautschnig. CBMC – C bounded model checker. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 389–391, 2014.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, JD Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/871ac99fdc5282d0301934d23945ebaa-Abstract-Conference.html.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP 2023)*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Antoine Labelle. On a specialization of Toda eigenfunctions, 2026. URL https://arxiv.org/html/2502.10655v3. Version 3, Conjecture 7.3.

Imre Lakatos. *Proofs and Refutations: The Logic of Mathematical Discovery*. Cambridge University Press, 1976. doi: 10.1017/CBO9781139171472. Edited by John Worrall and Elie Zahar.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. RewardBench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024. NAACL Findings 2025.

Peter Lammich. Efficient verified (UN)SAT certificate checking. *Journal of Automated Reasoning*, 64:513–532, 2020. doi: 10.1007/s10817-019-09525-z. URL https://link.springer.com/article/10.1007/s10817-019-09525-z.

Juho Lauri and Christodoulos Mitillos. Complexity of fall coloring for restricted graph classes, 2019. URL https://arxiv.org/abs/1905.04695v1. Version 1; cycle fall-coloring criterion in Section 4.

Vu Le, Mehrdad Afshari, and Zhendong Su. Compiler validation via equivalence modulo inputs. Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2014. https://dl.acm.org/doi/10.1145/2594291.2594334.

Joshua Ong Jun Leang, Haonan Li, Zheng Zhao, Xinyi Shang, Wenda Li, Zhengzhong Liu, Eric Xing, Shay B. Cohen, and Eleonora Giunchiglia. Magenta: Closing the loop between mathematical reasoning and Lean verification, 2026. URL https://arxiv.org/abs/2609.11319.

leanprover. lean4checker: Replay the Environment for a given Lean module. GitHub repository (leanprover/lean4checker), 2023. https://github.com/leanprover/lean4checker.

leanprover community. axiom-audit: fast axiom-allowlist audit for Lean libraries. GitHub repository (leanprover-community/axiom-audit), 2026. https://github.com/leanprover-community/axiom-audit.

K Rustan M Leino. Dafny: An automatic program verifier for functional correctness. In *International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, pp. 348–370, 2010.

Xavier Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7): 107–115, 2009.

Joel Brewster Lewis and Robert Won. Non-attacking rook placements on crossword grids, 2026. URL https://arxiv.org/abs/2609.03081v1. Version 1, Conjecture 3.10.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv preprint arXiv:2005.11401*, 2020. NeurIPS 2020.

Tianchi Li, Zhenyu Yan, Junhao Liu, Peng Di, and Xin Zhang. Guiding LLM-based loop invariant synthesis via feedback on local reasoning errors. *ACM Transactions on Programming Languages and Systems*, 48(2):8:1–8:30, 2026a. doi: 10.1145/3806652. URL https://arxiv.org/abs/2605.17914.

Yinghui Li, Jiayi Kuang, Haojing Huang, Zhikun Xu, Xinnian Liang, Yi Yu, Wenlian Lu, Yangning Li, Xiaoyu Tan, Chao Qu, Ying Shen, Hai-Tao Zheng, and Philip S. Yu. One example shown, many concepts known! counterexample-driven conceptual reasoning in mathematical LLMs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 35168–35185, 2025. URL https://proceedings.mlr.press/v267/li25ax.html.

Zenan Li, Zhaoyu Li, Kaiyu Yang, Xiaoxing Ma, and Zhendong Su. Learning to disprove: Formal counterexample generation with large language models. *arXiv preprint arXiv:2603.19514*, 2026b.

Hao Liang, Meiyi Qiang, Yuying Li, Zefeng He, Xiaochen Ma, Ruichuan An, Yongzhen Guo, Zhengzhou Zhu, Bin Cui, and Wentao Zhang. MathDebugger: Detecting and diagnosing errors in synthetic mathematical data. *arXiv preprint arXiv:2502.19058*, 2025. URL https://arxiv.org/abs/2502.19058.

Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Ding, Xinyu Yang, Kailas Vodrahalli, Siyu He, Daniel Smith, Yian Yin, Daniel McFarland, and James Zou. Can large language models provide useful feedback on research papers? a large-scale empirical analysis. *arXiv preprint arXiv:2310.01783*, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-Prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*, 2025a.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang, Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-Prover-V2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *arXiv preprint arXiv:2508.03613*, 2025b.

Zicheng Lin, Zhibin Gou, Tian Liang, Ruilin Luo, Haowei Liu, and Yujiu Yang. CriticBench: Benchmarking LLMs for critique-correct reasoning. Findings of the Association for Computational Linguistics: ACL 2024, 2024. https://aclanthology.org/2024.findings-acl.91/.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. Deductive verification of chain-of-thought reasoning. *arXiv preprint arXiv:2306.03872*, 2023. NeurIPS 2023.

Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, Ming Zhang, and Qun Liu. FIMO: A challenge formal dataset for automated theorem proving. *arXiv preprint arXiv:2309.04295*, 2023a.

Chengwu Liu, Ye Yuan, Yichun Yin, Yan Xu, Xin Xu, Zaoyu Chen, Yasheng Wang, Lifeng Shang, Qun Liu, and Ming Zhang. Safe: Enhancing mathematical reasoning in large language models via retrospective step-aware formal verification. *arXiv preprint arXiv:2506.04592*, 2025a.

Hongyi Liu, Frederic Sala, Thomas Reps, and Adithya Murali. Counterexample guided learning in the large using reasoning agents. *arXiv preprint arXiv:2606.11521*, 2026.

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

Lily L. Liu and Yi Wang. On the log-convexity of combinatorial sequences. *Advances in Applied Mathematics*, 39(4):453–476, 2007. doi: 10.1016/j.aam.2006.11.002. URL https://doi.org/10.1016/j.aam.2006.11.002.

Nelson F. Liu, Tianyi Zhang, and Percy Liang. Evaluating verifiability in generative search engines. Findings of the Association for Computational Linguistics: EMNLP 2023, 2023c. https://aclanthology.org/2023.findings-emnlp.467/.

Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. Rethinking and improving autoformalization: Towards a faithful metric and a dependency retrieval-based approach. The Thirteenth International Conference on Learning Representations (ICLR), 2025b. https://openreview.net/forum?id=hUb2At2DsQ.

Ryan Liu and Nihar B. Shah. ReviewerGPT? an exploratory study on using large language models for paper reviewing. *arXiv preprint arXiv:2306.00622*, 2023.

Jason Long and Adam Zsolt Wagner. The largest projective cube-free subsets of $\mathbb{Z}_{2^n}$. *European Journal of Combinatorics*, 81:156–171, 2019. doi: 10.1016/j.ejc.2019.05.005. arXiv:1810.01225.

Nuno P. Lopes, Juneyoung Lee, Chung-Kil Hur, Zhengyang Liu, and John Regehr. Alive2: Bounded translation validation for LLVM. Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI), 2021. https://dl.acm.org/doi/10.1145/3453483.3454030.

Chloe Loughridge, Qinyi Sun, Seth Ahrenbach, Federico Cassano, Chuyue Sun, Ying Sheng, Anish Mudide, Md Rakib Hossain Misu, Nada Amin, and Max Tegmark. DafnyBench: A benchmark for formal software verification. *arXiv preprint arXiv:2406.08467*, 2024.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024a.

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

Pan Lu, Jiayi Sheng, Luna Lyu, Jikai Jin, Tony Xia, Alex Gu, and James Zou. Solving inequality proofs with large language models. *arXiv preprint arXiv:2506.07927*, 2025.

Wenjie Ma, Andrei Cojocaru, Neel Kolhe, Bradley Louie, Robin Said Sharif, Haihan Zhang, Vincent Zhuang, Matei Zaharia, and Sewon Min. Reliable fine-grained evaluation of natural language math proofs. *arXiv preprint arXiv:2510.13888*, 2025. URL https://arxiv.org/abs/2510.13888.

Zeyao Ma, Jing Zhang, Xiaokang Zhang, Jiaxi Yang, Zongmeng Zhang, Jiajun Zhang, Yuheng Jing, Lei Zhang, Hao Zheng, Wenting Zhao, Junyang Lin, and Binyuan Hui. Scaling agentic verifier for competitive coding. *arXiv preprint arXiv:2602.04254*, 2026.

David R MacIver, Zac Hatfield-Dodds, and many other contributors. Hypothesis: A new approach to property-based testing. *Journal of Open Source Software*, 4(43):1891, 2019.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. Memory-assisted prompt editing to improve GPT-3 after deployment. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, 2022. https://aclanthology.org/2022.emnlp-main.183/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023. NeurIPS 2023.

Hamed Mahdavi, Alireza Hashemi, Majid Daliri, Pegah Mohammadipour, Alireza Farhadi, Samira Malek, Yekta Yazdanifard, Amir Khasahmadi, and Vasant Honavar. Brains vs. bytes: Evaluating LLM proficiency in olympiad mathematics. *arXiv preprint arXiv:2504.01995*, 2025a.

Hamed Mahdavi, Pouria Mahdavinia, Samira Malek, Pegah Mohammadipour, Alireza Hashemi, Majid Daliri, Alireza Farhadi, Amir Khasahmadi, Niloofar Mireshghallah, and Vasant Honavar. RefGrader: Automated grading of mathematical competition proofs using agentic workflows. *arXiv preprint arXiv:2510.09021*, 2025b.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2023. https://aclanthology.org/2023.emnlp-main.557/.

Charles F. Manski. *Partial Identification of Probability Distributions*. Springer Series in Statistics. Springer, New York, 2003. ISBN 978-0-387-00454-9. doi: 10.1007/b97478. URL https://link.springer.com/book/10.1007/b97478.

Samuele Marro, Jialin Yu, Emanuele La Malfa, Oishi Deb, Jiawei Li, Yibo Yang, Ebey Abraham, Sunando Sengupta, Eric Sommerlade, Michael Wooldridge, and Philip Torr. Benchmarking at the edge of comprehension. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2602.14307.

Math, Inc. strongpnt: AI-generated Lean formalization of the strong prime number theorem (Gauss autoformalization agent). GitHub repository / public announcement, 2025. https://github.com/math-inc/strongpnt.

Fateme Mazdarani and Carlos Toxtli. Beyond the answer key: Robustness evaluation of large language models for step-level mathematical verification. *arXiv preprint arXiv:2608.28725*, 2026. URL https://arxiv.org/abs/2608.28725.

Nat McAleese, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. LLM critics help catch LLM bugs. *arXiv preprint arXiv:2407.00215*, 2024.

Ross M McConnell, Kurt Mehlhorn, Stefan Näher, and Pascal Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011. doi: 10.1016/j.cosrev.2010.09.009.

William M McKeeman. Differential testing for software. *Digital Technical Journal*, 10(1):100–107, 1998.

Alexander Meinke, Bronson Schoen, Jérémy Scheurer, Mikita Balesni, Rusheb Shah, and Marius Hobbhahn. Frontier models are capable of in-context scheming. *arXiv preprint arXiv:2412.04984*, 2024.

Yuchen Meng. On cube-free problems. *The Electronic Journal of Combinatorics*, 33(1):P1.16, 2026. doi: 10.37236/14052. URL https://doi.org/10.37236/14052.

Dasha Metropolitansky and Jonathan Larson. VeriTrail: Closed-domain hallucination detection with traceability. *arXiv preprint arXiv:2505.21786*, 2025.

Julian Michael, Salsabila Mahdi, David Rein, Jackson Petty, Julien Dirani, Vishakh Padmakumar, and Samuel R. Bowman. Debate helps supervise unreliable experts. *arXiv preprint arXiv:2311.08702*, 2023.

Maciej Mikuła, Szymon Tworkowski, Szymon Antoniak, Bartosz Piotrowski, Albert Qiaochu Jiang, Jin Peng Zhou, Christian Szegedy, Łukasz Kuciński, Piotr Miłoś, and Yuhuai Wu. Magnushammer: A transformer-based approach to premise selection. *arXiv preprint arXiv:2303.04488*, 2023. ICLR 2024.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint arXiv:2411.00640*, 2024.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2023. https://aclanthology.org/2023.emnlp-main.741/.

Ludovico Mitchener, Angela Yiu, Benjamin Chang, Mathieu Bourdenx, Tyler Nadolski, Arvis Sulovari, Eric C. Landsness, Daniel L. Barabasi, Siddharth Narayanan, Nicky Evans, Shriya Reddy, Martha Foiani, Aizad Kamal, Leah P. Shriver, Fang Cao, Asmamaw T. Wassie, Jon M. Laurent, Edwin Melville-Green, Mayk Caldas, Albert Bou, Kaleigh F. Roberts, Sladjana Zagorac, Timothy C. Orr, Miranda E. Orr, Kevin J. Zwezdaryk, Ali E. Ghareeb, Laurie McCoy, Bruna Gomes, Euan A. Ashley, Karen E. Duff, Tonio Buonassisi, Tom Rainforth, Randall J. Bateman, Michael Skarlinski, Samuel G. Rodriques, Michaela M. Hinks, and Andrew D. White. Kosmos: An AI scientist for autonomous discovery. *arXiv preprint arXiv:2511.02824*, 2025.

Brahim Mittou. A new classification of positive integers via new divisor functions, 2025. URL https://arxiv.org/abs/2509.08844v2. Version 2, Conjecture 13.

Brahim Mittou. A new classification of positive integers via new divisor functions, 2026. URL https://arxiv.org/abs/2509.08844v3. arXiv:2509.08844v3, 7 September 2026, Remark 23.

Noor Islam S. Mohammad and Tamim Sheikh. The faithfulness gap: Certifying semantic equivalence between natural-language and formal mathematical statements. *arXiv preprint arXiv:2606.16541*, 2026.

Sherin Muckatira, Jesse Geneson, Slava Gerovitch, Pavel Etingof, Mikhail Gronas, and Anna Rumshisky. CrowdMath: A dataset of crowdsourced mathematical research discussions. *arXiv preprint arXiv:2606.06526*, 2026. URL https://arxiv.org/abs/2606.06526. Accepted to Findings of EMNLP 2026; available preprint.

Ali Hassaan Mughal and Muhammad Bilal. LLM-based test oracles: Source-of-authority taxonomy – a systematic literature review. *arXiv preprint arXiv:2607.05031*, 2026.

George C Necula. Proof-carrying code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pp. 106–119, 1997. doi: 10.1145/263699.263712.

Alexander Novikov, Ngân V u, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. https://doi.org/10.48550/arXiv.2506.13131.

Logan Nye. TIME[t] $\subseteq$ SPACE[$O(\sqrt{t})$] via tree height compression, 2025. URL https://arxiv.org/abs/2508.14831v1. arXiv:2508.14831v1, 20 August 2025 (Appendix B.2, Lemma 21). Version 4 (1 January 2026) is a withdrawal notice without a PDF.

Jan Ondras and Cameron Freer. Lean Disprove: Certified counterexample search for AI-assisted formal mathematics. In *Third AI for Math Workshop at ICML*, 2026. URL https://openreview.net/forum?id=5ck1jRE65S.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. *arXiv preprint arXiv:2201.03544*, 2022. ICLR 2022.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*, 2023a. TACL 2024.

Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. Fact-checking complex claims with program-guided reasoning. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2023b. https://aclanthology.org/2023.acl-long.386/.

Shrey Pandit, Austin Xu, Xuan-Phi Nguyen, Yifei Ming, Caiming Xiong, and Shafiq Joty. Hard2Verify: A step-level verification benchmark for open-ended frontier math. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026)*, pp. 22502–22517, 2026. doi: 10.18653/v1/2026.acl-long.1031. arXiv:2510.13744.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. *arXiv preprint arXiv:2404.13076*, 2024. NeurIPS 2024.

Mike Papadakis, Marinos Kintis, Jie Zhang, Yue Jia, Yves Le Traon, and Mark Harman. Mutation testing advances: An analysis and survey. Advances in Computers, 2019. https://doi.org/10.1016/bs.adcom.2018.03.015.

Zoe Paraskevopoulou, Cătălin Hriţcu, Maxime Dénès, Leonidas Lampropoulos, and Benjamin C. Pierce. Foundational property-based testing. Interactive Theorem Proving (ITP 2015), 2015. https://lemonidas.github.io/pdf/Foundational.pdf.

Olaf Parczyk and Christoph Spiegel. An unsure note on an un-Schur problem. *The Electronic Journal of Combinatorics*, 33(1):P1.45, 2026. doi: 10.37236/13554. URL https://doi.org/10.37236/13554.

Manooshree Patel, Rayna Bhattacharyya, Thomas Lu, Arnav Mehta, Niels Voss, Narges Norouzi, and Gireeja Ranade. LeanTutor: Towards a verified AI mathematical proof tutor. *arXiv preprint arXiv:2506.08321*, 2025.

Ivo Petrov, Jasper Dekoninck, Lyuben Baltadzhiev, Maria Drencheva, Kristian Minchev, Mislav Balunović, Nikola Jovanović, and Martin Vechev. Proof or bluff? evaluating LLMs on 2025 USA Math Olympiad. *arXiv preprint arXiv:2503.21934*, 2025a.

Ivo Petrov, Jasper Dekoninck, and Martin Vechev. BrokenMath: A benchmark for sycophancy in theorem proving with LLMs. *arXiv preprint arXiv:2510.04721*, 2025b. URL https://arxiv.org/abs/2510.04721.

Amir Pnueli, Michael Siegel, and Eli Singerman. Translation validation. Tools and Algorithms for Construction and Analysis of Systems, 4th International Conference, TACAS '98, 1998. https://doi.org/10.1007/BFb0054170.

Auguste Poiroux, Gail Weiss, Viktor Kunčak, and Antoine Bosselut. Reliable evaluation and benchmarks for statement autoformalization. *arXiv preprint arXiv:2406.07222*, 2024. EMNLP 2025.

Robert Pollack. How to believe a machine-checked proof. Twenty-Five Years of Constructive Type Theory (OUP); BRICS RS-97-18, 1998. https://www.brics.dk/RS/97/18/BRICS-RS-97-18.pdf.

Delip Rao, Jonathan Young, Thomas Dietterich, and Chris Callison-Burch. WithdrarXiv: A large-scale dataset for retraction study. *arXiv preprint arXiv:2412.03775*, 2024.

Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. Measuring attribution in natural language generation models. *arXiv preprint arXiv:2112.12870*, 2021. Computational Linguistics 49(4).

Gabriel Recchia, Chatrik Singh Mangat, Issac Li, and Gayatri Krishnakumar. FindTheFlaws: Annotated errors for detecting flawed reasoning and scalable oversight research. *arXiv preprint arXiv:2503.22989*, 2025.

Patrik Reizinger and Wieland Brendel. HALLMARK: Diagnosing three failure modes in LLM citation verifiers. *arXiv preprint arXiv:2607.18360*, 2026.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

Anka Reuel, Amelia Hardy, Chandler Smith, Max Lamparth, Malcolm Hardy, and Mykel J. Kochenderfer. BetterBench: Assessing AI benchmarks, uncovering issues, and establishing best practices. *arXiv preprint arXiv:2411.12990*, 2024. NeurIPS 2024 Datasets & Benchmarks.

Talia Ringer, Karl Palmskog, Ilya Sergey, Milos Gligoric, and Zachary Tatlock. QED at large: A survey of engineering of formally verified software. *Foundations and Trends in Programming Languages*, 5(2–3):102–281, 2019.

Igor Rivin. From Euler to today: Universal mathematical fallibility a large-scale computational analysis of errors in ArXiv papers. *arXiv preprint arXiv:2511.10543*, 2025.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024.

Herbert John Ryser. *Combinatorial Mathematics*. Number 14 in Carus Mathematical Monographs. Mathematical Association of America, 1963. doi: 10.5948/UPO9781614440147. URL https://doi.org/10.5948/UPO9781614440147.

Jon Saad-Falcon, E. Kelly Buchanan, Mayee F. Chen, Tzu-Heng Huang, Brendan McLaughlin, Tanvir Bhathal, Shang Zhu, Ben Athiwaratkun, Frederic Sala, Scott Linderman, Azalia Mirhoseini, and Christopher Ré. Shrinking the generation-verification gap with weak verifiers. *arXiv preprint arXiv:2506.18203*, 2025. NeurIPS 2025.

Adil Salim. Accelerating mathematical research with language models: A case study of an interaction with GPT-5-Pro on a convex analysis problem. *arXiv preprint arXiv:2510.26647*, 2025.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988. doi: 10.1016/0306-4573(88)90021-0. URL https://doi.org/10.1016/0306-4573(88)90021-0.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

Michael Schlichtkrull, Zhijiang Guo, and Andreas Vlachos. AVeriTeC: A dataset for real-world claim verification with evidence from the web. *arXiv preprint arXiv:2305.13117*, 2023. NeurIPS 2023 Datasets and Benchmarks.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent Laboratory: Using LLM agents as research assistants. *arXiv preprint arXiv:2501.04227*, 2025.

Johannes Schmitt, Gergely Bérczi, Jasper Dekoninck, Jeremy Feusi, Tim Gehrunger, Raphael Appenzeller, Pieter Belmans, Alessio Bottini, Jim Bryan, João Camarneiro, Ana Cannas da Silva, Niklas Canova, Ana-Maria Castravet, Timo de Wolff, Claudio Fontanari, Filippo Gaia, Baran Hashemi, Daniel Holmes, David Holmes, Aitor Iribar Lopez, Victor Jaeck, Martina Jørgensen, Steven Kelk, Martijn Kool, Stefan Kuhlmann, Adam Kurpisz, Johannes Lengler, Chiara Meroni, Ingmar Metzler, Martin Möller, Samuel Muñoz-Echániz, David Muñoz-Lahoz, Robert Nowak, Georg Oberdieck, Daniel Platt, Dylan Possamaï, Gabriel Ribeiro, Aluna Rizzoli, Daria Sakhanda, Raúl Sánchez Galán, Zheming Sun, Diaaeldin Taha, Josef Teichmann, Richard P. Thomas, Henk van der Pol, Michel van Garrel, Charles Vial, Ignacio Barros, Benjamin Doerr, Peter Grünwald, Henry Liu, David Martins, Aleksandar Mijatović, Sergej Monavari, Marc Roth, Patrick Schnider, Yannik Schuler, Pim Spelier, Yuuji Tanaka, and Ronald van Luijk. IMProofBench: Benchmarking AI on research-level mathematical proof generation. *arXiv preprint arXiv:2509.26076*, 2025.

Johannes Schmitt, Tim Gehrunger, Jasper Dekoninck, Gergely Bérczi, Uri Kreitner, Liam Price, and David Holmes. ProofCouncil: An LLM agent for solving open mathematical problems. *arXiv preprint arXiv:2607.09474*, 2026.

Peter Scholze. Liquid Tensor Experiment. *Experimental Mathematics*, 31(2):349–354, 2022. doi: 10.1080/10586458.2021.1926016. URL https://doi.org/10.1080/10586458.2021.1926016.

Sergio Segura, Gordon Fraser, Ana B. Sánchez, and Antonio Ruiz-Cortés. A survey on metamorphic testing. *IEEE Transactions on Software Engineering*, 42(9):805–824, 2016. doi: 10.1109/TSE.2016.2532875.

David Seka and Stefan Szeider. Neurosymbolic discovery of algebraic graph constructions. *arXiv preprint arXiv:2608.08118*, 2026.

Saeed Shaebani. On fall colorings of graphs, 2009. URL https://arxiv.org/abs/0909.2769v1. Version 1; Theorem 2 and the example following Corollary 3.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious rewards: Rethinking training signals in RLVR. *arXiv preprint arXiv:2506.10947*, 2025.

Hao Shen, Junyu Guo, Tian Cui, Yuxuan Xiao, and Lihong Zhi. MechGeo: Autoformalizing and proving Euclidean geometry in Lean 4, 2026. URL https://arxiv.org/abs/2608.02295.

Jiawen Shi, Zenghui Yuan, Yinuo Liu, Yue Huang, Pan Zhou, Lichao Sun, and Neil Zhenqiang Gong. Optimization-based prompt injection attack to LLM-as-a-judge. *arXiv preprint arXiv:2403.17710*, 2024. ACM CCS 2024.

Wenlei Shi and Xing Jin. Heimdall: test-time scaling on the generative verification. *arXiv preprint arXiv:2504.10337*, 2025.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. NeurIPS 2023.

Stephen F Siegel, Manchun Zheng, Ziqing Luo, Timothy K Zirkel, Andre V Marianiello, John G Edenhofner, Matthew B Dwyer, and Michael S Rogers. CIVL: the concurrency intermediate verification language. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1–12, 2015.

Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. When to solve, when to verify: Compute-optimal problem solving and generative verification for LLM reasoning. *arXiv preprint arXiv:2504.01005*, 2025. COLM 2025.

Shiven Sinha, Shashwat Goel, Ponnurangam Kumaraguru, Jonas Geiping, Matthias Bethge, and Ameya Prabhu. Can language models falsify? evaluating algorithmic reasoning with counterexample creation. *arXiv preprint arXiv:2502.19414*, 2025. URL https://arxiv.org/abs/2502.19414.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Peter J. Slater. Leaves of trees. In *Proceedings of the Sixth Southeastern Conference on Combinatorics, Graph Theory, and Computing*, volume 14 of *Congressus Numerantium*, pp. 549–559, 1975.

Andries Smit, Paul Duckworth, Nathan Grinsztajn, Thomas D. Barrett, and Arnu Pretorius. Should we be going MAD? a look at multi-agent debate strategies for LLMs. *arXiv preprint arXiv:2311.17371*, 2023. ICML 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. ICLR 2025.

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit A. Seshia, and Vijay A. Saraswat. Combinatorial sketching for finite programs. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 404–415, 2006. doi: 10.1145/1168857.1168907. URL https://doi.org/10.1145/1168857.1168907.

Guijin Son, Jiwoo Hong, Honglu Fan, Heejeong Nam, Hyunwoo Ko, Seungwon Lim, Jinyeop Song, Jinha Choi, Gonçalo Paulo, Youngjae Yu, and Stella Biderman. When AI co-scientists fail: SPOT-a benchmark for automated verification of scientific research. *arXiv preprint arXiv:2505.11855*, 2025.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. PRMBench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, pp. 25299–25346, 2025. doi: 10.18653/v1/2025.acl-long.1230. arXiv:2501.03124.

Nat Sothanaphan. Resolution of Erdős problem #728: a writeup of Aristotle's Lean proof. *arXiv preprint arXiv:2601.07421*, 2026.

Suvrit Sra. GPT, the counterexample machine. *arXiv preprint arXiv:2608.29595*, 2026. URL https://arxiv.org/abs/2608.29595.

KV Aditya Srivatsa, Kaushal Kumar Maurya, and Ekaterina Kochmar. LLMs cannot spot math errors, even when allowed to peek into the solution. *arXiv preprint arXiv:2509.01395*, 2025.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. *arXiv preprint arXiv:2402.08115*, 2024.

Hao Sun, Yong Jiang, Bo Wang, Yingyan Hou, Yan Zhang, Pengjun Xie, and Fei Huang. Retrieved in-context principles from previous mistakes. Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, 2024. https://aclanthology.org/2024.emnlp-main.465/.

Yifeng Sun. Evaluating research-level math proofs via strict step-level verification. *arXiv preprint arXiv:2606.10799*, 2026a.

Zhi-Wei Sun. A family of polynomials and related congruences and series. *Frontiers in Combinatorics and Number Theory*, 2:1–22, 2026b. URL https://arxiv.org/abs/2505.02767v4. arXiv:2505.02767v4 (10 February 2026; first posted 2025). Conjecture 1.1(i), second assertion.

Zhouhao Sun, Xuan Zhang, Xiao Ding, Bibo Cai, Li Du, Kai Xiong, Xinran Dai, Fei Zhang, weidi tang, Zhiyuan Kan, Yang Zhao, Bing Qin, and Ting Liu. GR-Ben: A general reasoning benchmark for evaluating process reward models. *arXiv preprint arXiv:2605.01203*, 2026.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic Cheatsheet: Test-time learning with adaptive memory. *arXiv preprint arXiv:2504.07952*, 2025.

Stefan Szeider. LRAT-Catcher: Importing SAT solver certificates into Lean4 by reflection. *arXiv preprint arXiv:2607.00815v1*, 2026. URL https://arxiv.org/abs/2607.00815v1.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. *arXiv preprint arXiv:2410.12784*, 2024. ICLR 2025.

Yong Kiam Tan, Marijn J. H. Heule, and Magnus O. Myreen. cake_lpr: Verified propagation redundancy checking in CakeML. Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, 2021. https://doi.org/10.1007/978-3-030-72013-1_12.

Amitayush Thakur, George Tsoukalas, Yeming Wen, Jimmy Xin, and Swarat Chaudhuri. An in-context learning agent for formal theorem-proving. *arXiv preprint arXiv:2310.04353*, 2023. COLM 2024.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*, pp. 367–381, 2020. doi: 10.1145/3372885.3373824.

Ken Thompson. Reflections on trusting trust. Communications of the ACM 27(8), 1984. https://dl.acm.org/doi/10.1145/358198.358210.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018. https://aclanthology.org/N18-1074/.

Rishabh Tiwari, Aditya Tomar, Udbhav Bamba, Monishwaran Maheswaran, Heng Yang, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Reward under attack: Analyzing the robustness and hackability of process reward models. *arXiv preprint arXiv:2603.06621*, 2026. URL https://arxiv.org/abs/2603.06621.

Alexis Akira Toda. Can AI refute economic theory? evidence from beyond the knowledge cutoff, 2026. URL https://arxiv.org/abs/2606.05383.

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. Can LLMs learn from previous mistakes? investigating LLMs' errors to boost for reasoning. *arXiv preprint arXiv:2403.20046*, 2024. ACL 2024.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625:476–482, 2024.

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. *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks*, 2024.

Ken Tsui. Self-Correction Bench: Uncovering and addressing the self-correction blind spot in large language models. *arXiv preprint arXiv:2507.02778*, 2025. COLM 2026.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. LLMs cannot find reasoning errors, but can correct them given the error location. Findings of the Association for Computational Linguistics: ACL 2024, 2024. https://aclanthology.org/2024.findings-acl.826/.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022. URL https://arxiv.org/abs/2211.14275.

Teun van der Weij, Felix Hofstätter, Ollie Jaffe, Samuel F. Brown, and Francis Rhys Ward. AI sandbagging: Language models can strategically underperform on evaluations. *arXiv preprint arXiv:2406.07358*, 2024.

Joshua Vendrow, Edward Vendrow, Sara Beery, and Aleksander Madry. Do large language model benchmarks test reliability? *arXiv preprint arXiv:2502.03461*, 2025.

Vasudev Vikram, Caroline Lemieux, Joshua Sunshine, and Rohan Padhye. Can large language models write good property-based tests? *arXiv preprint arXiv:2307.04346*, 2023.

Sydney Von Arx, Lawrence Chan, and Beth Barnes. Recent frontier models are reward hacking. METR blog, June 5, 2025, 2025. https://metr.org/blog/2025-06-05-recent-reward-hacking/.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020. https://aclanthology.org/2020.emnlp-main.609/.

Adam Zsolt Wagner. Constructions in combinatorics via neural networks. *arXiv preprint arXiv:2104.14516*, 2021.

Danqing Wang and Lei Li. Learning from mistakes via cooperative study assistant for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10667–10685, 2023. doi: 10.18653/v1/2023.emnlp-main.659. URL https://aclanthology.org/2023.emnlp-main.659/.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a. TMLR 2024.

Haiming Wang, Huajian Xin, Chuanyang Zheng, Lin Li, Zhengying Liu, Qingxing Cao, Yinya Huang, Jing Xiong, Han Shi, Enze Xie, Jian Yin, Zhenguo Li, Heng Liao, and Xiaodan Liang. LEGO-Prover: Neural theorem proving with growing libraries. *arXiv preprint arXiv:2310.00656*, 2023b. ICLR 2024.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey, Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu, Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani, Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Barroyer, Wen-Ding Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying Liu, and Jia Li. Kimina-Prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025a.

Haocheng Wang, Baiyu Huang, Yingjia Wan, Xiao Zhu, Xiaoyang Liu, Yinya Huang, and Zhijiang Guo. FormalRx: Rectify and eXamine semantic failures in autoformalization, 2026a. URL https://arxiv.org/abs/2607.04655.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*, 2023c. ACL 2024.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023d. ACL 2024.

Ran Wang, Jie Xiong, and Lihu Xu. Large deviation principle of occupation measure for stochastic real Ginzburg-Landau equation driven by α-stable noises, 2015. URL https://arxiv.org/abs/1510.03522v1. arXiv:1510.03522v1, 13 October 2015.

Shuai Wang, Jiayi Kuang, Yinghui Li, Haojing Huang, Xinnian Liang, Ying Shen, and Liang Lin. INSPIRE: An internalize-then-improve approach for example-driven mathematical reasoning. *arXiv preprint arXiv:2608.27501*, 2026b.

Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. Shepherd: A critic for language model generation. *arXiv preprint arXiv:2308.04592*, 2023e.

Xiangfeng Wang, Hangyu Guo, Yanlin Lai, Mitt Huang, Liang Zhao, Chengyuan Yao, Yinmin Zhang, Qi Han, Xiaoxiao Ren, Chun Yuan, Tong Xu, Zheng Ge, Xiangyu Zhang, and Daxin Jiang. PRIME: A process-outcome alignment benchmark for verifiable reasoning in mathematics and engineering. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14973–14985. Association for Computational Linguistics, 2026c. doi: 10.18653/v1/2026.acl-long.683. URL https://aclanthology.org/2026.acl-long.683/.

Xinpeng Wang, Nitish Joshi, Barbara Plank, Rico Angell, and He He. Is it thinking or cheating? detecting implicit reward hacking by measuring reasoning effort. *arXiv preprint arXiv:2510.01367*, 2025b.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. The Eleventh International Conference on Learning Representations (ICLR), 2023f. https://openreview.net/forum?id=1PL1NIMMrw.

Ziyu Wang, Qiming Dai, Yishan Wu, and Zaiwen Wen. FaithSieve: Fine-grained evaluation of math proofs with faithful formal evidence. *arXiv preprint arXiv:2608.26310*, 2026d. URL https://arxiv.org/abs/2608.26310.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory. *arXiv preprint arXiv:2409.07429*, 2024.

S. Ole Warnaar and Wadim Zudilin. q-rious unimodality, 2025. URL https://arxiv.org/html/2502.03993v2. Version 2, Conjecture 5.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. Long-form factuality in large language models. *arXiv preprint arXiv:2403.18802*, 2024. NeurIPS 2024.

Ke Weng, Lun Du, Sirui Li, Wangyue Lu, Haozhe Sun, Hengyu Liu, and Tiancheng Zhang. Autoformalization in the era of large language models: A survey. *arXiv preprint arXiv:2505.23486*, 2025.

Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt, Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pp. 422–429, 2014.

Freek Wiedijk. Pollack-inconsistency. UITP 2010 / ENTCS 285, 2012. https://doi.org/10.1016/j.entcs.2012.06.008.

Dominik Winterer, Chengyu Zhang, and Zhendong Su. Validating SMT solvers via semantic fusion. Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2020), 2020. https://dl.acm.org/doi/10.1145/3385412.3385985.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus N Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Ziang Wu. COVER: Counterexample-guided repair for natural-language constraint modelling. In *New Trends in Intelligent Software Methodologies, Tools and Techniques: Proceedings of the 25th International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques (SoMeT_26)*, Frontiers in Artificial Intelligence and Applications, 2026. doi: 10.3233/FAIA260737. URL https://journals.sagepub.com/doi/10.3233/FAIA260737.

Sarina Xi, Vishisht Rao, Justin Payan, and Nihar B. Shah. FLAWS: A benchmark for error identification and localization in scientific papers. *arXiv preprint arXiv:2511.21843*, 2025a.

Zhiheng Xi, Jixuan Huang, Xin Guo, Boyang Hong, Dingwen Yang, Xiaoran Fan, Shuo Li, Zehui Chen, Junjie Ye, Siyu Yuan, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. Critique-RL: Training language models for critiquing through two-stage reinforcement learning. *arXiv preprint arXiv:2510.24320*, 2025b.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. Evaluating mathematical reasoning beyond accuracy. *arXiv preprint arXiv:2404.05692*, 2024.

Lihan Xie, Zhicheng Hui, Yingjun Lan, Zhehao Li, Xingzhi Qi, Siyue Huang, Jirui Liu, Chuxiao Zeng, Bohan Zhao, and Qinxiang Cao. ProofGap: Benchmarking step-level formal reasoning with local obligations derived from natural-language solutions, 2026. URL https://arxiv.org/abs/2609.29296.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. DeepSeek-Prover: Advancing theorem proving in LLMs through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*, 2024a.

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *arXiv preprint arXiv:2408.08152*, 2024b. ICLR 2025.

Zhaopan Xu, Pengfei Zhou, Jiaxin Ai, Wangbo Zhao, Kai Wang, Xiaojiang Peng, Wenqi Shao, Hongxun Yao, and Kaipeng Zhang. MPBench: A comprehensive multimodal reasoning benchmark for process errors identification. *arXiv preprint arXiv:2503.12505*, 2025.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Yibo Yan, Shen Wang, Jiahao Huo, Hang Li, Boyan Li, Jiamin Su, Xiong Gao, Yi-Fan Zhang, Tianlong Xu, Zhendong Chu, Aoxiao Zhong, Kun Wang, Hui Xiong, Philip S. Yu, Xuming Hu, and Qingsong Wen. ErrorRadar: Benchmarking complex mathematical reasoning of multimodal large language models via error detection. *arXiv preprint arXiv:2410.04509*, 2024. Findings of ACL 2026.

Yuchen Yan, Jin Jiang, Zhenbang Ren, Yijun Li, Xudong Cai, Yang Liu, Xin Xu, Mengdi Zhang, Jian Shao, Yongliang Shen, Jun Xiao, and Yueting Zhuang. VerifyBench: Benchmarking reference-based reward systems for large language models. *arXiv preprint arXiv:2505.15801*, 2025. ICLR 2026.

Jun Yang, Yuechun Sun, Yi Wu, Rodrigo Caridad, Yongwei Yuan, Jianan Yao, Shan Lu, and Kexin Pei. ExVerus: Verus proof repair via counterexample reasoning. *arXiv preprint arXiv:2603.25810*, 2026.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. Buffer of Thoughts: Thought-augmented reasoning with large language models. *arXiv preprint arXiv:2406.04271*, 2024a. NeurIPS 2024.

Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. ReasonFlux: Hierarchical LLM reasoning via scaling thought templates. *arXiv preprint arXiv:2502.06772*, 2025.

Xuejun Yang, Yang Chen, Eric Eide, and John Regehr. Finding and understanding bugs in C compilers. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pp. 283–294, 2011.

Zhe Yang, Yichang Zhang, Yudong Wang, Ziyao Xu, Junyang Lin, and Zhifang Sui. Confidence v.s. critique: A decomposition of self-correction capability for LLMs. *arXiv preprint arXiv:2412.19513*, 2024b.

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. *arXiv preprint arXiv:2406.03847*, 2024a. NeurIPS 2024 Datasets and Benchmarks.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, Yudong Wang, Zijian Wu, Shuaibin Li, Fengzhe Zhou, Hongwei Liu, Songyang Zhang, Wenwei Zhang, Hang Yan, Xipeng Qiu, Jiayu Wang, Kai Chen, and Dahua Lin. InternLM-Math: Open math large language models toward verifiable reasoning. *arXiv preprint arXiv:2402.06332*, 2024b.

Fei Yu, Yingru Li, and Benyou Wang. Scaling flaws of verifier-guided search in mathematical reasoning. *arXiv preprint arXiv:2502.00271*, 2025a.

Xiao Yu, Baolin Peng, Michel Galley, Jianfeng Gao, and Zhou Yu. Teaching language models to self-improve through interactive demonstrations. Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), 2024. https://aclanthology.org/2024.naacl-long.287/.

Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, Yandong Wen, Ge Zhang, and Weiyang Liu. FormalMATH: Benchmarking formal mathematical reasoning of large language models. *arXiv preprint arXiv:2505.02735*, 2025b.

Zhongshen Zeng, Pengguang Chen, Shu Liu, Haiyun Jiang, and Jiaya Jia. MR-GSM8K: A meta-reasoning benchmark for large language model evaluation. *arXiv preprint arXiv:2312.17080*, 2023. ICLR 2025.

Zhongshen Zeng, Yinhong Liu, Yingjia Wan, Jingyao Li, Pengguang Chen, Jianbo Dai, Yuxuan Yao, Rongwu Xu, Zehan Qi, Wanru Zhao, Linling Shen, Jianqiao Lu, Haochen Tan, Yukang Chen, Hao Zhang, Zhan Shi, Bailin Wang, Zhijiang Guo, and Jiaya Jia. MR-Ben: A meta-reasoning benchmark for evaluating system-2 thinking in LLMs. Advances in Neural Information Processing Systems 37 (NeurIPS 2024), 2024. https://openreview.net/forum?id=GN2qbxZlni.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Charlotte Zhuang, Dylan Slack, Qin Lyu, Sean Hendryx, Russell Kaplan, Michele Lunati, and Summer Yue. A careful examination of large language model performance on grade school arithmetic. *arXiv preprint arXiv:2405.00332*, 2024a.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024b. ICLR 2025.

Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zheng, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. In-context principle learning from mistakes. *arXiv preprint arXiv:2402.05403*, 2024c. ICML 2024.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2023. AAAI 2024.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. GenPRM: Scaling test-time compute of process reward models via generative reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(41):34932–34940, 2026. doi: 10.1609/aaai.v40i41.40797. URL https://ojs.aaai.org/index.php/AAAI/article/view/40797.

Kevin Zhao. On zero-sum subsequences in a finite abelian group of length not exceeding a given number, 2025. URL https://arxiv.org/abs/2506.21383v1. Version 1, Conjecture 6.2; Lemma 1.7(iii) and remark following Theorem 1.11.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, pp. 1009–1024, 2025. doi: 10.18653/v1/2025.acl-long.50. arXiv:2412.06559.

Daniel Zheng, Ingrid von Glehn, Yori Zwols, Iuliya Beloshapka, Lars Buesing, Daniel M. Roy, Martin Wattenberg, Bogdan Georgiev, Tatiana Schmidt, Andrew Cowie, Fernanda Viegas, Dimitri Kanevsky, Vineet Kahlon, Hartmut Maennel, Sophia Alj, George Holland, Alex Davies, and Pushmeet Kohli. AI co-mathematician: Accelerating mathematicians with agentic AI. *arXiv preprint arXiv:2605.06651*, 2026a.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations (ICLR)*, 2022.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks*, 2023.

Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Jing Jiang, and Min Lin. Cheating automatic LLM benchmarks: Null models achieve high win rates. *arXiv preprint arXiv:2410.07137*, 2024. ICLR 2025.

Xinyi Zheng, Ningke Li, Xiaokun Luan, Kailong Wang, Ling Shi, Meng Sun, and Haoyu Wang. Beyond correctness: Exposing LLM-generated logical flaws in reasoning via multi-step automated theorem proving. In *48th IEEE/ACM International Conference on Software Engineering*, 2026b. URL https://arxiv.org/abs/2512.23511.

Ziqian Zhong, Aditi Raghunathan, and Nicholas Carlini. ImpossibleBench: Measuring LLMs' propensity of exploiting test cases. *arXiv preprint arXiv:2510.20270*, 2025.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning LLM agents without fine-tuning LLMs. *arXiv preprint arXiv:2508.16153*, 2025.

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. *arXiv preprint arXiv:2403.18120*, 2024a. ICLR 2024.

Zihao Zhou, Shudong Liu, Maizhen Ning, Wei Liu, Jindong Wang, Derek F. Wong, Xiaowei Huang, Qiufeng Wang, and Kaizhu Huang. Is your model really a good math reasoner? evaluating mathematical reasoning with checklist. *arXiv preprint arXiv:2407.08733*, 2024b. ICLR 2025.

Appendix guide. Appendix N describes the retained resource, and Appendix D presents the human reviews. Appendix P gives the benchmark protocol and evaluation details. The mathematical developments are in Appendices B, K, L, and M, with further proofs in Appendix S.

A SOURCE-LINKED COUNTEREXAMPLES AND THEIR MATHEMATICAL CONSEQUENCES

The examples below connect a source assertion to a concrete counterexample and its Lean certificate. Some also show how that counterexample leads to a corrected formula or a more general theorem. The formula, spanning-tree, coding-theory, and two QEDBench examples belong to the 147-dossier human-verified core. The three-vertex graph example and square-root follow-up are supplementary examples, with their review status stated below.

Finding the source and certificate. The anonymized repository's README and artifact guide provide navigation. Each dossier identifier below locates three files: `resource/dossiers/<record-id>.json`, `resource/sources/<record-id>.json`, and `formal/certificates/<record-id>.lean`. The first records the objection, the second preserves the source text, and the third contains the formal proof. Current core membership is recorded in `reviews/verified_core/membership.json`; `benchmark/verified_core/README.md` describes the 100-input main evaluation. The selected examples and their recorded build evidence are indexed in `reviews/source_continuity/source_trust/auditing_exhibits.json`.

A three-vertex counterexample to a graph criterion. Dossier `arxiv_0709.2689v3__source_enriched_codex` examines Claim 1 in version 1 of Kettani (2007). For a graph on n vertices, let U be the strictly upper triangular part of its adjacency matrix. The claim says that the maximum independent-set size is k if and only if $U^{n-k+1} = 0$. The objection uses the vertex order supplied to the algorithm, which tests this matrix power without searching over vertex orderings.

Consider the path numbered 1–2–3. Its maximum independent set has size two, but $U^2 \neq 0$. Numbering matters: U^2 is zero for this path unless its middle vertex receives the middle number. The theorem `Arxiv07092689v3SourceEnrichedCodex.refutes_claim1_forward_implication_on_ordered_path` formalizes this example using Mathlib independent sets and matrix powers. With vertices numbered $0, 1, 2$, it proves that $\{0, 2\}$ is independent, every independent set has at most two vertices, and the specified matrix power is nonzero. The saved build and axiom audit passed with `propext`, `Classical.choice`, and `Quot.sound`.

The version-3 identifier refers to the later withdrawal notice, dated 21 September 2007, which attributes withdrawal to an error in Claim 1. The notice is preserved in `reviews/source_continuity/source_trust/source_notice_0709_2689.json`. This supplementary dossier has a formal certificate but no saved human review approving all three criteria, so it is outside the human-verified core.

A community formula with an explicit correction. Dossier `SE_cs_13118__a13122` concerns the CS Stack Exchange answer (Answer to "Why is this sequence of recurrence relevant?"). The source sums the powers of c from 1 to $n-1$, but gives a geometric-series quotient with a different lower endpoint. At $c = 2, n = 3$, the sum is 6 and the printed quotient is 7. The theorem `QEDCert.SE_cs_13118__a13122.real_sum_counterexample` verifies these values and their inequality over the reals. Its saved build and axiom audit passed with the same three standard axioms listed above.

The discrepancy identifies the correction: summing from 1 rather than 0 gives $(c^n - c)/(c-1)$ for $n \geq 1$ and $c \neq 1$. This general identity follows from the finite geometric sum; the selected Lean certificate establishes the concrete instance that exposes the endpoint error.

A program error with a universal certified repair. The CS Stack Exchange answer (Answer to "Relating a proof to a Haskell program"), stored as `SE_cs_97637__a97647`, derives an integer-square-root algorithm from the specification

$$R(x, y) \equiv y^2 \leq x \ \wedge \ x < (y+1)^2.$$

The printed algorithm, `prog`, keeps its previous answer b when $x \leq (b+1)^2$. At $x = 1$, this rule returns zero instead of one. The theorem `SE_cs_97637_a97647.refutation` certifies the failure on a natural-number input, which is also allowed by the source's integer specification.

Changing the comparison to $x < (b+1)^2$ gives the corrected algorithm `progStrict`. The file `mathematics/additions/sqrt_repair/SE_cs_97637_a97647_repair.lean` preserves both algorithms and the specification. Its theorem `strict_guard_correct` proves by induction that

$$\forall x \in \mathbb{N}, \quad R(x, \mathrm{progStrict}(x)).$$

The theorem `error_and_repair` pairs this universal repair with the original failure.

The next question is exactly when the original algorithm fails. The file `mathematics/additions/sqrt_boundary/SqrtFailureBoundary.lean` proves

$$\mathrm{prog}(n+1) = \mathrm{progStrict}(n) \qquad (n \in \mathbb{N}).$$

The original algorithm is always one input behind the correction. This identity and the corrected specification yield the complete failure set for natural inputs x:

$$\neg R(x, \mathrm{prog}(x)) \quad \Longleftrightarrow \quad \exists k \in \mathbb{N},\ k > 0 \ \wedge\ x = k^2.$$

The theorem `complete_diagnosis` combines this equivalence with universal correctness of the repair. The source annotations had identified the positive-square pattern; the follow-up establishes that the pattern accounts for every failure. Both follow-up files have recorded successful builds under Lean 4.30.0 and 4.34.0 using only `propext` and `Quot.sound`, with receipts beside the sources. These are mathematical follow-ups. The original dossier is excluded from the human-verified core because review E01 questioned its interpretation of the source (Appendix D).

A tied cut explains a failed exchange. Dossier `Algorithms_19__sol3` from QED-Bench (Gonzalez et al., 2026) concerns a spanning-tree proof. Consider a triangle with edge weights $w_{01} = w_{02} = 1$ and $w_{12} = 2$. Its unique minimum spanning tree is $\{01, 02\}$, while the cut $\{0\} \mid \{1, 2\}$ has two tied minimum edges. Deleting 01 and choosing 02 as its replacement leaves a disconnected graph, because 02 was already present.

The theorem `QEDReview19cut.refuted` uses Mathlib graph objects to verify the unique minimum spanning tree, the tied cut minima, and the failed exchange. The counterexample pinpoints the proof's mistake: the components created by deleting a tree edge need not be the two sides of an arbitrarily chosen cut.

A false converse in coding theory. Dossier `arxiv_1303.4920v2__source_enriched_codex` targets Theorem 3.3 in version 1 of Dougherty et al. (2013). Let

$$C = \langle (1,1) \rangle \leq \mathbb{Z}_4^2.$$

Its standard Gray image is the binary self-dual code $\{0000, 0101, 1111, 1010\}$. However, C is not quaternary self-dual, since $(1,1) \cdot (1,1) = 2$ modulo four. The theorem `Arxiv13034920Mathlib.original_witness_refutes` proves this using submodules and orthogonal complements.

Thus binary self-duality of the Gray image does not imply quaternary self-duality of the original code. This is the converse asserted for every $\mathbb{Z}_4$ code in the numbered theorem; the manuscript also considers a more restricted length-72 application. The dossier's version-2 identifier points to the later withdrawal notice, which cites a different lemma.

Objections in previously reviewed source collections. The completed 425-record audit reconciled the selected source collections and retained eight local objections from a 200-answer vetted CS collection and five from 43 accepted math.SE answers with no recorded flaw comments. These are counts from that audit, before construction of the current human-verified core. The file `reviews/source_continuity/source_trust/source_trust_reconciliation.json` records the identifiers, source denominators, and decisions.

Two examples in the current core, `Algorithms_29__sol5` and `Algorithms_36__sol3`, received perfect recorded human rubric scores in QEDBench (Gonzalez et al., 2026). Their later counterexamples illustrate why even highly rated proofs benefit from checking individual assertions. The reconciliation's denominator of 19 counts retained QEDBench records with linked scores; it is not the number of perfect-scored source proofs. We therefore report these as two examples. The original rubric scores and the later reviews of their counterexample dossiers are separate assessments.

From one graph coloring to a family theorem. Replace each vertex of the nine-cycle by two adjacent vertices, joining every vertex in one pair to every vertex in each neighboring pair. The resulting graph, $C_9[K_2]$, has 18 vertices. The file `mathematics/supporting/sources/C9K2_frozen_coloring.lean` formalizes its adjacency rule and explicit five- and six-colorings. Its theorem `C9K2.frozen_six_colouring_certificate` verifies a proper six-coloring in which no vertex can change color while keeping the coloring proper. It also verifies a proper coloring with at most five colors and a second proper six-coloring. The first six-coloring is therefore isolated under single-vertex recoloring, even though another proper six-coloring exists.

This finite example motivates two broader questions: does the base cycle satisfy every hypothesis of the source conjecture, and which cycle lengths admit the same obstruction? The theorem `Recol.not_conjecture1`, in `mathematics/supporting/sources/PrimeBlowup.lean`, checks the remaining hypotheses and refutes the conjecture. The theorem `Recol.cycle_blowup_frozen`, in `mathematics/sources/CycleBlowupAllN.lean`, establishes every clause of Proposition 2 for odd $n \geq 5$, including the exact chromatic number and the classification and count of isolated six-colorings. The number of such colorings is exact; the resulting count of connected components in the recoloring graph is a lower bound.

Recorded builds for the finite example and the conjecture refutation are indexed in `mathematics/supporting/index.json`; the family theorem's build evidence is in `mathematics/recorded_builds/cycle_k2.json`. The finite certificate has successful recorded builds under Lean 4.30.0 and 4.34.0 using only `propext` and `Quot.sound`. The later formalizations received the scoped human reviews reported in Appendix D. These follow-up theorems develop the counterexample's mathematical consequences separately from the benchmark's dossier and input counts.

B WHY THE SQUARE-ROOT ALGORITHM FAILS, AND WHY THE CORRECTION WORKS

The first failing input, $x = 1$, reveals a boundary error: the algorithm keeps zero when it should return one. The same error occurs whenever the correct answer increases. We prove that these inputs are exactly the positive perfect squares, and that changing one comparison corrects the algorithm for every nonnegative integer input. This is the written proof of Proposition 1; Appendix A gives the source example and its Lean certificates.

Let $f(x)$ and $g(x)$ denote the answers returned by the original and corrected algorithms. Both compute the answer for x from the answer for $x-1$, either keeping it or increasing it by one. Their common starting value is zero:

$$f(0) = g(0) = 0, \qquad f(x) = \begin{cases} f(x-1), & x \leq (f(x-1)+1)^2, \\ f(x-1)+1, & \text{otherwise}, \end{cases}$$

$$g(x) = \begin{cases} g(x-1), & x < (g(x-1)+1)^2, \\ g(x-1)+1, & \text{otherwise}, \end{cases} \qquad x \geq 1.$$

The comparison changes from $\leq$ to $<$ in the source algorithm (Answer to "Relating a proof to a Haskell program"). An answer y is correct when $R(x, y)$ holds, where $R(x, y)$ means $y^2 \leq x < (y+1)^2$. Equivalently, $y = \lfloor \sqrt{x} \rfloor$, the greatest integer whose square is at most x. Proposition 1 states that $g(x) = \lfloor \sqrt{x} \rfloor$ for every nonnegative integer x, and that $f(x) = g(x-1)$ for every $x \geq 1$. Thus the original algorithm is one input behind the corrected algorithm. Its exact failure set is

$$\neg R(x, f(x)) \quad \Longleftrightarrow \quad \exists k \in \mathbb{N},\ k \geq 1 :\ x = k^2. \tag{1}$$

All inputs in this statement are nonnegative integers, the domain of the recurrences above.

Proof of Proposition 1. **Step 1: the corrected algorithm returns the integer square root.** We prove $g(x) = \lfloor \sqrt{x} \rfloor$ by induction on x. At zero, the claim follows from $g(0) = 0$. For $x \geq 1$, let $b = g(x-1)$. The induction hypothesis gives

$$b^2 \leq x - 1 < (b+1)^2.$$

If $x < (b+1)^2$, keeping b gives $b^2 \le x < (b+1)^2$, as required. Otherwise, the integer $(b+1)^2$ is greater than $x-1$ and at most x, so it equals x. Increasing the answer to $b+1$ then gives $(b+1)^2 \le x < (b+2)^2$. This proves correctness at every input.

Step 2: the original algorithm returns the preceding correct answer. We prove $f(x) = g(x-1)$ by induction for $x \ge 1$. At $x = 1$, $f(1) = 0 = g(0)$. For $x \ge 2$, the induction hypothesis gives $f(x-1) = g(x-2) = b$. The updates for $f(x)$ and $g(x-1)$ make the same choice, because, for integers,

$$x \le (b+1)^2 \quad \Longleftrightarrow \quad x - 1 < (b+1)^2.$$

Both therefore keep b or both return $b+1$. Hence $f(x) = g(x-1) = \lfloor \sqrt{x-1} \rfloor$.

Step 3: the delay causes an error exactly at positive squares. For $x \ge 1$, let $b = f(x) = \lfloor \sqrt{x-1} \rfloor$. We have $b^2 \le x$ and $x \le (b+1)^2$. The required strict upper bound fails exactly when $x = (b+1)^2$. Conversely, at any positive square $x = k^2$, the original algorithm returns $k-1$, so its upper-bound condition fails by equality. At zero, the answer is correct. This proves both the exact failure classification and the correction for every allowed input. □

The anonymous repository provides the formal proofs in `mathematics/additions/sqrt_boundary/SqrtFailureBoundary.lean` and `mathematics/additions/sqrt_repair/SE_cs_97637_a97647_repair.lean`, with toolchains and replay instructions in `formal/README.md`.

C WHAT A COUNTEREXAMPLE ESTABLISHES

The examples below show how a counterexample can correct a formula or identify a faulty step in a proof. Appendix N describes the current 147-dossier resource and its 100-input primary evaluation set; Appendix D explains the human checks supporting admission.

Checking mathematics that has already been accepted. Consider the "Calculus of differences" passage in Answer to "Why is this sequence of recurrence relevant?". Under its summation convention, $\Sigma(T)(n) = \sum_{k=1}^{n-1} T(k)$, the printed formula $(c^n - 1)/(c-1)$ gives 7 at $c = 2, n = 3$, while the sum is 6. Lean checks the numerical discrepancy. The finite geometric-sum identity gives the corrected numerator $c^n - c$ (Appendix A). Two QEDBench records in the current core concern proofs that previously received perfect human rubric scores. These examples show the value of checking individual steps even after expert review.

C.1 ONE GRAPH, TWO DIFFERENT CONCLUSIONS

The same small graph can explain different kinds of mathematical error. Figure 3 uses the path on three vertices. In one source it contradicts a matrix assertion. In another, it shows why a proposed proof step fails, while the handshaking identity being proved is true. Section 2 uses this distinction to explain what an objection establishes. The related formal statements and reviews are in Appendices I and D.

An independent set is a set of vertices with no edges between them; its largest possible size is the independence number $\alpha(G)$. For a graph with numbered vertices, the adjacency matrix has entry one when two vertices share an edge and zero otherwise. Its strictly upper part U keeps only the entries above the diagonal. The power U^2 uses ordinary matrix multiplication, and the subscript 13 selects row one, column three.

The matrix assertion. The saved record `arxiv_0709.2689v3__source_enriched_codex` concerns Claim 1 of version 1 of Kettani (2007); the version-three suffix is part of its record identifier. For the ordering along the path, the selected theorem `refutes_claim1_forward_implication_on_ordered_path` checks both the independent-set bound and the nonzero matrix square. The statement concerns this specified vertex order. The record belongs to the earlier collection; it is preserved here as a concrete illustration of the scope of a formal refutation.

A false matrix assertion

The path 1–2–3 has $n = 3$ vertices and a largest independent set $\{1, 3\}$ of size $k = 2$. Its strictly upper adjacency matrix is

$$U = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

The entry $(U^2)_{13} = 1$. This contradicts the printed assertion that an independent set of size k forces $U^{n-k+1} = 0$, for this vertex order. The source's withdrawal notice also identifies a fault in that claim.

A failed step in a proof of a true identity

A leaf is a vertex with one neighbor. Deleting the path's two leaves removes two edges, whereas reversing the proposed construction would require removing just one. Thus the construction misses this graph.

The degrees, or numbers of neighbors, are one, two, and one. With two edges, the handshaking identity holds:

$$1 + 2 + 1 = 4 = 2 \times 2.$$

The graph pinpoints an error in the proposed proof step.

Figure 3: A counterexample's conclusion depends on the assertion being checked. The path contradicts the matrix assertion on the left and the proposed construction on the right. Its degree sum satisfies the handshaking identity.

The proposed handshaking proof. Math Stack Exchange question 782216 proposes adding two degree-one vertices and one edge. Such an operation adds an isolated K_2 component: two vertices joined to each other and to no other vertex. Reversing it therefore requires an isolated K_2, which the connected three-vertex path does not contain. The original certificate checked a separate base-case error. The subsequent mathematical follow-up proves the construction's exact boundary, including the path example (Appendix I). This source dossier is in the current human-reviewed core.

The anonymous repository's `ARTIFACTS.md` links the source survey, quoted passages, formal statements, and review records. The source counts are reproduced in `reviews/source_continuity/source_trust/source_trust_reconciliation.json`; current membership is recorded in `reviews/verified_core/membership.json`.

D HUMAN REVIEW STAGES AND FINDINGS

Nine human reviewers contributed to a three-stage process of checking and refining the collection. A human review must answer three questions affirmatively for a dossier to enter the retained core:

1. Does the objection accurately represent the original source claim and its assumptions?
2. Does the counterexample satisfy those assumptions and make the claim false?
3. Does the Lean formalization correctly represent the counterexample?

The reference-line checks and consequence questions were evaluated separately. The current 147-dossier core (137 source units) requires at least one completed review answering YES to all three questions, evidence that the review applies to the current version, no retained negative or uncertain human judgment, and no unresolved source-scope concern. The individual review records and current inclusion decisions are preserved in `reviews/verified_core/membership.json`.

D.1 STAGE 1: INITIAL 100-DOSSIER REVIEW AND DERIVED FILTERS

Eight human reviewers evaluated an initial set of 100 artifacts. This initial phase identified common source-interpretation and formalization issues. The human observations recorded in these reviews were subsequently used to write a revised guidance prompt (preserved in `reviews/initial100/feedback_filter/`), which drove broader automated filtering across the corpus. Of the 100 completed reviews, 61 answered YES to all three questions, 25 contained a NO, and 14 were uncertain without a NO. The resulting automated audit examined 425 dossiers and retained 285; a subsequent withdrawal left 284. These initial review counts are preserved in `reviews/original100/answers.json` and remain separate from the final 147-record selection.

Table 4: Source judgments among reviews that approved both the counterexample's mathematics and its Lean translation. Attribution-clean rows remove the two reciprocal comment-assignment anomalies without reassigning them. These are selected human judgments from the initial evaluation.

Review group	Returns	Both YES	Source YES	NO	Unclear
Human reviews	100	76	61	7	8
Human reviews (clean)	98	74	59	7	8

Of the 76 reviews that approved both the counterexample's mathematics and its Lean translation, 61 also found that the objection accurately represented the original claim and its assumptions; seven did not, and eight were uncertain. The 15 reviews that did not approve the source interpretation raised four reported hypothesis or domain omissions, one missing definition, one asymptotic-versus-literal interpretation dispute, one initial-value discrepancy, one wrong passage highlight, six source-availability limitations, and one notation/encoding uncertainty. All 15 dossiers are excluded from the current 147-dossier core.

Table 5: All 15 eligible concerns from the initial human reviews. "?" is CANNOT DETERMINE. Exact workbook cells, comments and qualifications are retained in the provenance bundle.

Record	Q1	Reported issue and qualification
`Graph_Theory_20__sol4`	NO	Omitted disconnectedness assumption; Q2 accepts the extracted claim conditionally.
`SE_cs_1957__a2388`	NO	Missing definition of g; the comment does not give the alternative reading.
`SE_math_4219732__q`	NO	Complexity-analysis equality read literally; passage location accepted.
`mse_4217205__ans4217514`	NO	Omitted $0 < \theta < 180$; reviewer says the witness satisfies it.
`proofwiki_43948__codex_expansion`	NO	Wrong passage highlighted; counterexample explicitly accepted.
`proofwiki_56736`	NO	Omitted range 118–157; model said to have retained it.
`reddit_askmath_1kp2tga`	NO	Source reportedly has initial value 4, packet 0.
`reddit_askmath_chxjkr`	?	Deleted source; surviving context provisionally accepted.
`reddit_askmath_g7lcfx`	?	Deleted source and uncertain exponentiation notation; interpretation provisionally accepted.
`reddit_learnmath_1hi0nrt`	?	Deleted source; surviving context provisionally accepted.
`reddit_learnmath_b5iobo`	?	Notation uncertain; Q3 YES qualified by an encoding-recheck request.
`reddit_learnmath_egpgy9`	?	Additional conditions such as $BC = 18$ omitted; Q2 still YES.
`reddit_learnmath_rv8znd`	?	Deleted source; surviving context provisionally accepted.
`reddit_math_18hrdbw`	?	Deleted source; surviving context provisionally accepted.
`reddit_math_3osy6o`	?	Called archived and deleted in different comments; context accepted.

The human comments also motivated comparisons of the original and replacement counterexamples. Appendix D.6.1 examines all 16 replacement pairs and explains how checking the source assumptions changes the objection, even when both formal proofs compile.

D.2 Stage 2: Further random review and pruning

Following the prompt-driven filtering, a single human reviewer evaluated a second stratified random sample of 30 artifacts drawn from the 221 retained dossiers with no earlier review. The reviewer disclosed AI assistance. This stage returned 27 dossiers with YES on all three primary questions, guiding further pruning. Specifically, it recorded 28 YES and 2 NO for source connection (R012, R026), 30 YES for valid mathematics, and 29 YES and 1 NO for Lean statement match (R019).

Supplementary targeted reviews. During this refinement period, several supplementary evidence cohorts were collected. One human reviewer returned judgments for 33 explicitly selected artifacts: 32 mathematical artifacts and the documentary supplement D05. Thirty received YES answers to all three primary questions: 29 mathematical artifacts and D05. Table 6 records the detailed judgments from this selected supplementary round.

Table 6: Supplementary human review judgments for 33 explicitly selected artifacts.

Record (ID)	Source Connection	Mathematics Valid	Lean statement match
E01	NO	YES	YES
E02	YES	NO	YES
E03	YES	YES	YES
E04	YES	YES	YES
E05	YES	YES	YES
E06	YES	YES	YES
E07	YES	YES	YES
E08	YES	YES	YES
E09	YES	YES	YES
E10	YES	YES	YES
E11	YES	YES	YES
E12	YES	YES	YES
E13	YES	YES	YES
R01	YES	YES	YES
R02	YES	YES	YES
R03	YES	YES	YES
R04	YES	YES	YES
R05	YES	YES	YES
R06	YES	YES	YES
R07	YES	YES	YES
R08	YES	YES	YES
R09	YES	YES	YES
R10	YES	YES	YES
R12	YES	YES	YES
R13	YES	YES	YES
R14	YES	YES	YES
S01	YES	N/A	YES
B01	YES	YES	YES
C01	YES	YES	YES
S02	YES	YES	YES
T11	YES	YES	YES
Z12	YES	YES	YES
D05 (documentary)	YES	YES	YES

In a separate Fixed30 review round, one human reviewer evaluated 30 selected current artifacts. The return includes 29 dossiers with YES on all three primary questions. For I025, the reviewer found that the objection did not accurately represent the source claim. Records E01, E02, R012, R019, R026, and I025 are all excluded from the 147-dossier core. Three mathematical supplements (L01,

H09, G07) received all-YES judgments on the three primary questions and the optional consequence from one reviewer.

The reference-line checks asked whether the marked lines stated the challenged claim, whether any such line was missing, and whether an unrelated line was marked. The criterion was at least 38 of 40 complete line selections, corresponding to YES/NO/NO answers. The initial human review approved 35 of 40. After corrections, a fresh sample of 40 was drawn from the 239 remaining eligible records, excluding all 40 from the first sample. This second review approved 38 of 40 and met the criterion. The two remaining concerns were an omitted line (G14) and an extraneous marked line (G12). Both sets of original judgments are preserved in `reviews/reference_lines/initial40.json` and `reviews/reference_lines/fresh40.json`.

D.3 Stage 3: Blinded random audit

Finally, we drew a stratified random sample without replacement from the 284 retained dossiers: 67 of the 191 primary records and 33 of the 93 other records. One human reviewer, using AI assistance and personally checking the judgments, assessed accurate source interpretation, witness validity, and correspondence with the Lean statement. The review packets displayed the source and certificate text and masked earlier verdicts and interpretive commentary. The reviewer reported prior project exposure for 28 records.

All three criteria received a YES for 63/67 primary and 28/33 other records, totaling 91/100. The sample contains two groups: primary benchmark records and other records. Weighting each group by its share of the 284-dossier collection gives an estimated approval rate of 91.0%, with a simultaneous 95% sampling interval of [81.0%, 96.5%]. We combine two exact 97.5% finite-population intervals by Bonferroni to obtain at least 95% joint coverage. Individual component approval counts were 95/100 for accurate source interpretation, 98/100 for witness validity, and 94/100 for Lean correspondence. The optional reference-line check received 98 YES and two not-applicable responses. The interval summarizes sampling uncertainty for this review procedure on the 284-record population. The sampling plan and interval method were specified before review; the reviewer-admissibility clarification and weighted point estimate followed the returned judgments. The design, chronology, and analysis are preserved in `reviews/random100/PROTOCOL.md`, `reviews/random100/design.json`, and `reviews/random100/saved_analysis.json`.

D.4 Final dispositions and exclusions

All nine nonapprovals in the final 100-record audit are excluded from the 147-dossier core. Four came from its primary stratum (A001, A002, A023, A043) and five from its other stratum (A072, A075, A085, A093, A099). Table 7 preserves their three answers and recorded follow-up categories. A previously recorded concern retains its original NO judgment.

The version ledger at `reviews/version_resolution/README.md` links every case to its full artifact hashes, source text, selected Lean declaration, original review and final disposition. For all nine, the certificate bytes and source-line arrays match the reviewed versions; exclusion does not overturn the negative judgments. Seven scored cases received YES on the separate line check, which did not resolve their source, witness or Lean concerns.

Of the 147 retained dossiers, 113 have exact matches between the reviewed scientific contents and the current contents. For the other 34, the authors confirmed that the saved reviews apply to the current versions; this confirmation is recorded separately from exact content matching. Among 162 dossiers with at least one all-YES review, 13 are excluded because of adverse human review history and two further dossiers because of source-scope concerns. The membership ledger and continuity receipt are provided in `reviews/verified_core/membership.json` and `reviews/verified_core/author_continuity_receipt.json`. Both evaluation subsets exclude the 31 dossiers flagged for unresolved concerns about source interpretation, witness validity, or the Lean translation, and the three further dossiers with unresolved reference-line concerns.

The 100 primary benchmark inputs include 69 with a positive check of the current reference-line selection: 55 have a positive check with the exact current source text and labels, and 27 have a separate positive line-selection review, with overlap between these groups. The older line-review exports identify the reviewed labels but do not establish complete source-version identity. Three dossiers with

Table 7: The nine nonapprovals in the final random audit. Q1, Q2 and Q3 check the source interpretation, witness and Lean statement. All nine are excluded from the current 147 dossiers and both benchmark subsets. "Earlier set" refers to the original 191 primary and 255 scorable inputs. Certificate versions are identified by SHA-256 prefixes; the linked version ledger gives full source, packet and certificate hashes. Concerns summarize the recorded reviews, not new adjudications.

Packet	Earlier set	Negative checks	Recorded concern	Certificate version
A001	Primary	Q3	Connection from the source's group construction to Lean is left to prose.	`51284638c1b1`
A002	Primary	Q2, Q3	Zero witness conflicts with an implicit nonzero assumption.	`af06bbfe5425`
A023	Primary	Q1, Q2, Q3	Category-label mismatch is treated as a refutation under a disputed translation.	`c8e308ad5c7d`
A043	Primary	Q3	Lean supplies the size-one transition law rather than deriving it.	`76b30ad2964d`
A072	Other scored	Q3	Boolean instance lacks formal syntax and substitution linking it to the source.	`d47105809371`
A075	Other scored	Q1, Q3	Humorous runtime claim is unspecified; Lean checks a supplied runtime function.	`af1c203b5790`
A085	Other scored	Q1	True equalities are displayed; the false step is implied rather than asserted.	`b0893e69a1ae`
A093	Not scored	Q1	Target is a typing requirement in the model setup, rather than an assertion.	`0e7eea1a1635`
A099	Not scored	Q1	Quoted supposition introduces a case; the gap is its later generalization.	`cf26d14808fe`

adverse judgments on the current line selections remain in the 147-record resource and are excluded from both evaluation subsets: `arxiv_2207.06033v2__source_enriched_codex`, `proofwiki_12198__codex_expansion`, and `reddit_mathematics_1dz93wr`. This leaves 100 primary and 131 total scorable inputs. The reference-line evidence is preserved in `benchmark/verified_core/reference_line_coverage.json`.

All cohort answers are available under `reviews/original100/`, `reviews/earlier_selected30/`, `reviews/selected33/`, `reviews/selected30/`, `reviews/scoped_supplements/`, and `reviews/random100/`; `reviews/README.md` explains the files and reproduction commands. Appendix N describes the retained core.

D.5 The initial 100 human reviews

Eight human reviewers returned 100 completed reviews from 110 assignments on 5–12 September 2026. These were the first round of the broader review process, which involved nine human reviewers overall (Appendix D). All 100 reviewed identifiers belong to the earlier 401-dossier census. One returned workbook covers 30 records across two batches, and the other seven cover ten each. The ten assignment workbooks are blank templates; only completed returns contribute judgments.

Each reviewer answered three questions: does the objection correctly interpret the source, is its counterexample valid, and does its Lean statement express that counterexample? We recompute the outcome from these explicit answers. Three YES answers count as approval; any NO counts as rejection; CANNOT DETERMINE without a NO counts as inconclusive. The table preserves the answers as returned, with accompanying comments kept in the review evidence.

Initial returned judgment	Records
All three answers YES	61
At least one NO	25
Inconclusive, without NO	14
Completed reviews	100
Assigned without a completed return	10

What the comments contributed. Human comments identified issues that a calculation or successful Lean build could not resolve on its own. For `Algorithms_35__sol5`, exchanging the two sides of a bipartition did not create the distinct bipartition required by the claim. For `Graph_Theory_20__sol4`, the objection omitted a disconnectedness assumption. For `Graph_Theory_12__sol5`, the selected Lean theorem refuted a conjunction, whereas the source asserted an implication; the reviewer pointed to another theorem in the same file that addressed the implication. These are the reviewers' recorded observations. They helped shape the later filtering prompt and review process described in Appendix D.

Two approvals with ambiguous comments. Two rows in the batch-01 return have three YES answers but apparently exchanged comments. The comments for `mse_3703719__ans3707488` (Answers row 8) describe the singleton matroid-rank example in `proofwiki_132447`; the ProofWiki row (Answers row 4) describes the MSE record's complementary-half-set example. We preserve both returns as written. Excluding these two rows gives 98 reviews with unambiguous comment attribution, including 59 approvals. The full and attribution-clean counts are both available in the public review tables.

Review conditions and version records. The 110 assigned packets included earlier automated assessments in a background appendix. Instructions asked reviewers to finish their own judgment before reading it. The returns do not record when that material was read, so this first round is reported as an unblinded review. Each completed row supplies a reviewer name and date. It contains one reviewer's judgment, and its date records completion rather than time spent. Ten assignments have no returned judgment.

The saved ledger preserves workbook hashes, sheet and row references, comments, and the attribution issue. The initial returns did not include reviewer-supplied hashes of the particular PDF or Lean file they read. Current-version applicability is therefore documented separately: the 147-dossier core has 113 exact scientific-content matches and 34 author-confirmed continuities with earlier positive reviews. The current inclusion rule also checks all retained adverse judgments and source assumptions; 44 core members are supported by this initial review group. The complete process and the later review rounds are in Appendix D.

Public evidence. Anonymous row-level answers, attribution flags, and cohort information are available in `reviews/original100/answers.json` and `reviews/cohorts.json`. From the anonymous repository root, `python -B reviews/reproduce.py --check` recomputes the recorded counts, including the attribution-clean sensitivity. The review README explains the exported evidence. Original votes are preserved alongside the separate current-core membership and version checks in `reviews/verified_core/`.

D.6 Source assumptions and checked replacements

The initial human judgments above and the following saved certificate comparisons concern the earlier construction process. The paired sources, witnesses, certificates, and receipts are indexed in `reviews/source_continuity/PAIR_INDEX.json`.

D.6.1 Comparing original and replacement counterexamples

We compare all 16 records whose counterexamples were replaced during the earlier checks. Five belong to the current 147-dossier core, including the sorting and Gaussian-integer examples below. All 16 pairs retain the same stored problem and source context; 15 also retain the same challenged quote. The comparison therefore makes the change in the counterexample directly inspectable.

The original Lean files have been recovered for all 16 records. Four have saved original build and axiom-check receipts; the other 12 were recompiled unchanged under Lean 4.30.0 and 4.34.0, with all 24 compilations succeeding. All 16 replacement certificates also have build receipts. The new replays establish that the recovered files compile; the original receipts document which checks were performed earlier.

Three pairs preserve the same quote and have both original and replacement build receipts: the endpoint-order example below, an equality interpreted as an eventual asymptotic claim, and a

covariance claim tested with an ensemble different from the source's. The fourth pair with original receipts changes its challenged quote. The public pair index records these distinctions alongside the source excerpts, witnesses and audit reasons.

An inspection of all 123 theorem and lemma declarations in the 12 recompiled files found 78 using no axioms, 42 using only `propext`, and three using `propext` and `Quot.sound`. None uses `sorryAx` or native reduction. Three inspection drivers also queried comment text as declaration names; independent review checked that their outputs still covered every genuine declaration. The files, commands and complete outputs are available in `reviews/source_continuity/historical_replay_20260921/`.

The replacements address several different issues. Four original modules already contain an alternative that satisfies the source assumptions: the arXiv probability family, the ProofWiki induction instance, the six-team tournament and the limit calculation at $x = 3/2$. The first two already select that stronger formal statement; the other two contain it alongside the disputed statement. Other replacements restore omitted assumptions or conventions. The examples below explain two such changes. Their interpretation follows the recorded audits; their membership in this comparison is determined by the recorded replacement, rather than by random sampling.

An executable check of the omitted source condition. The gap-sorting source explicitly assumes $x_0 < x_n$. Its challenged local step says that after the first exchange with x_m, the final gap-m sorted array begins with that entry x_m. The original witness uses $m = 2$ and $A = (3, 4, 2, 5, 1)$. Its compiled theorem correctly shows that the final first entry is 1 rather than $x_2 = 2$. But $3 < 1$ is false: this array violates the source's endpoint hypothesis.

The replacement $A' = (3, 5, 2, 6, 1, 7, 4)$ satisfies $3 < 4$; its first gap-2 insertion exchanges 3 and 2, and the final array is $(1, 5, 2, 6, 3, 7, 4)$, again beginning with 1 rather than $x_2 = 2$. The selected replacement theorem proves the endpoint inequality, exchange hypothesis and failed conclusion together.

A supplementary Lean file retains the current certificate's exact bytes as its prefix and adds three axiom-free lemmas. For the explicit predicate $H(a) := a_0 < a_{|a|-1}$, they prove $\neg H(A)$, rule out the conjunction of $H(A)$ with the encoded objection conditions, and prove that A' satisfies those conditions. The file compiles under Lean 4.30.0 with exit code zero. This checks the particular source condition's role; correspondence between the quoted prose and H remains an inspectable interpretation. It does not certify the entire source proof or claim that its final theorem is false.

An equality case that satisfies the source assumptions. In `mse_605342__ans609345`, the Gaussian-integer source first chooses α from the lattice generated by $1 + i$ with $N(a - \alpha b) \leq N(b)$, then claims that coprimality makes this strict. The old certificate takes $a = b = -1$, $\alpha = i$, and correctly computes $2 > 1$, outside that weak-bound premise. The current Mathlib certificate keeps $a = b = -1$ and uses $\alpha = 1 + i$. Its theorem explicitly establishes coprimality, opposite coordinate parities, the allowed multiplier and the weak bound, but proves

$$N(a - \alpha b) = N(b) = 1, \qquad \neg\big(N(a - \alpha b) < N(b)\big).$$

A companion theorem proves $b \mid a$ together with coprimality: divisibility by this unit is no contradiction. The repaired objection therefore identifies the equality boundary in the local strict-descent argument; it does not establish nontermination of the algorithm.

Together, these comparisons explain the source check used in Co-PEER: state the relevant assumptions, prove that the counterexample satisfies them, and establish the contradiction. The human judgments and certificate comparisons provide complementary evidence for these three steps.

E CHECKING THE REFERENCE OBJECTIONS

We revisited the reference objections for the 100 proofs used in the instruction comparison. Twenty-two cases had matching full source texts and unchanged witnesses supported by the earlier automated checks. On this subset, counterexample search and witness-required sequential review received 22 and 21 supported-objection grades for Sol, 21 and 20 for Gemini, and 19 and 19 for Opus 5. Table 8

reports the paired token comparisons. These are results from the earlier experiment; Appendix N defines the current collection.

The 22 cases comprise 15 exact text matches and seven complete-text matches after removing line numbers and normalizing whitespace. Cases with only a matching excerpt or a replacement witness were excluded. The subset was selected using these source-matching rules, not sampled at random.

Table 8: The 22 source-matched cases from the instruction experiment. Counts compare search with witness-required sequential review, using the common September grades. Ratios compare accepted solver-attempt token sums. Intervals use the paired 20,000-resample analysis in Table 15.

Model	Catches	Δ catches (pp)	Output ratio	Full-token ratio
GPT-5.6 Sol	22/21	+4.5 [+0.0, +13.6]	0.759 [0.633, 0.919]	0.986 [0.981, 0.993]
Gemini 3.1 Pro	21/20	+4.5 [-9.1, +18.2]	0.721 [0.626, 0.819]	0.909 [0.863, 0.951]
Opus 5	19/19	+0.0 [-13.6, +13.6]	0.703 [0.627, 0.784]	0.948 [0.866, 1.031]

What changed when the references were checked? Thirteen original references failed later checks: ten had a recorded source-interpretation problem, and three used witnesses that were replaced as invalid. The saved grader nevertheless marked 60 of their 78 model-and-instruction responses as matching the reference objection. The other responses received six supported-alternative, one unsupported-objection, and eleven missed grades. This illustrates why agreement with a stored reference must be checked against the source.

Table 9 preserves the response grades across all recorded groups. The groups overlap. Scores also change with the selected subset: Gemini has 84/89 supported objections on all 100 inputs, 35/39 on the 42 cases supported by the recorded automated checks, and 21/20 on the 22 source-matched cases, with search listed first.

Three examples. The following cases show why an objection's exact target matters. They were examined in the earlier source audit.

`mse_25914__ans25916`: a local proof step.
The proposed step requires a product of strictly smaller positive integers. Prime 7 has no such factorization. The source's assertion about expressing an integer as an odd number times a power of two is not disproved. The selected Lean theorem addresses the product claim, including possible factors 1.

`mse_1513740__ans1513768`: a local size comparison.
The proposed strict size comparison fails at $k = 2$: the reduced system has two equations, not fewer than two. This refutes the printed comparison, not the underlying solvability theorem.

`SE_cs_12812__a12823`: an algorithmic claim.
The source proposes satisfying the first disjunct as its procedure. It does not assume that every disjunct is internally consistent. The witness tests that stated procedure rather than adding such an assumption.

Reproduction. The instruction-experiment reproduction guide links the source-matching rules, record lists, saved responses, grades, token components, and example evidence. The original 100-input comparison and every group in the tables remain available there.

F Token accounting for the paired witness-required study

This appendix explains how the instruction study counts supported objections and token usage. Its 100 proof inputs are the earlier instruction-study panel; the current 100-input claim-selection benchmark is a separate evaluation drawn from the 147-dossier human-approved core. Both instruction conditions require a concrete counterexample: search chooses a claim to challenge, while sequential review examines the proof in order.

Answers, grades, and reproducibility. Each model has 100 paired grades and 100 paired accepted-attempt usage records. A supported objection receives `CAUGHT_MATCHING` when it challenges the reference error, or `CAUGHT_DIFFERENT` when the grader accepts another error. The comparison uses the common September grading pass, including the regraded Opus 5 responses. The anonymous repository provides the source audit in `experiments/instructions/reference_validity/audit.json` and the input manifest in `experiments/instructions/reproduction/manifest.json`. `experiments/instructions/README.md` gives the portable replay commands.

Paired estimates. A token ratio divides the sum used by search by the sum used by sequential review on the same tasks. A ratio below one therefore means lower recorded usage for search. For token ratios and differences in supported objections, we resample task pairs 20,000 times with seed 20260917 and report the 2.5th and 97.5th percentiles (Efron, 1979). These exploratory, pointwise intervals quantify uncertainty for each comparison. The original output-token summaries in Appendix H use 2,000 resamples; this difference explains small changes in their interval endpoints.

Search yields 95 supported and 3 unsupported objections for Sol, compared with 93 and 4 for sequential review. The corresponding counts are 84 and 1 versus 89 and 0 for Gemini, and 81 and 2 versus 76 and 3 for Opus 5. The saved grade name `FALSE_CATCH` denotes an objection the model grader found unsupported on an input labelled flawed. Checking the underlying reference is a separate part of the analysis, reported in Appendix E.

Input and cache accounting. Gemini reports uncached input separately from cache reads, so its full-token total adds both components to output. This corrects an earlier summary that subtracted cache reads from uncached input. The saved receipts preserve the original components. Accepted-attempt ratios measure recorded solver usage; the ledgers separately identify additional attempts and unknown usage. Monetary comparisons would also require prices and billing rules for each execution system.

Connecting the study to the source audit. The saved crosswalk matches each study input to the 425-record source audit by `record_id` or its explicit `original_record_id` alias. Two Graph Theory records use the latter. This produces 46 retained, 25 excluded, and 29 unmatched records in that earlier audit. The literal-ID-only version gave 46, 23, and 31, respectively. Both versions have the same 21 human-review overlaps. These counts describe the earlier audit crosswalk; current core membership is recorded separately. Establishing that a review applies to an experiment's saved reference also requires comparing the source text and witness. Appendix E provides that comparison; the original human returns identify their review packets but do not record hashes of the files the reviewers opened.

Recorded tool use. The instructions allow reading the input and prohibit other code execution. The accepted-attempt logs nevertheless record six additional executions across four Sol task pairs and 48 across six Gemini pairs. For Sol, 1 search cell and 3 review cells contain such executions, with execution totals of 2 and 4. For Gemini, the corresponding cell counts are 2 and 5, and the execution totals are 6 and 42. Here an execution means a recorded non-read command or tool call; it can include work other than mathematical computation.

The main comparison includes all 100 task pairs. Table 11 also reports the pairs with zero additional executions in both conditions, so every comparison keeps the two answers to a task together. This provides a view of the recorded runs under a narrower execution condition. Opus 5 has no matching execution telemetry. A controlled test of tool permissions would require assigning those permissions before running the models.

The per-task data are in `experiments/instructions/reference_validity/toolfree_sensitivity.json`.

How a large execution affects the average. For Gemini, the median per-task full-token ratio is 0.952, while the ratio of sums is 0.650. The difference arises in part from one large sequential review, `arxiv_1904.10482v2__source_enriched_codex`. It records 24 additional executions and 1,639,229 input tokens, or 1,686,890 including output. Omitting that pair gives a ratio of 0.830.

The complete 100-pair result and this sensitivity together show how long individual runs affect aggregate usage.

G SEARCH, REPAIR, AND REUSE: WHAT THE STUDIES TEST

The claim-selection benchmark asks where a critic identifies an error. The studies summarized in Section 5 examine three further uses of counterexamples: directing a search, completing a supplied correction, and helping revise a proof. Each study uses its own paired tasks and checking procedure. Their results describe these different activities; the current benchmark evaluates 100 inputs from the 147-dossier approved core.

G.1 DOES SEARCHING FOR A COUNTEREXAMPLE USE FEWER TOKENS?

The instruction study uses 100 proofs selected for errors expected to admit concrete examples. *Counterexample-first search* asks a model to find a claim it can refute. *Sequential review with a witness* asks it to read the proof in order and stop at an error supported by a concrete example. Both conditions receive the same proof. A model grader (Zheng et al., 2023) uses the saved reference to assess the answer, crediting either a supported objection to that error or a supported objection to another error. The generated answers are evaluated by this grading procedure.

Search uses 15–31% fewer output tokens than sequential review requiring a witness. The intervals for supported-objection differences cover both gains and losses, leaving the accuracy tradeoff open. The comparison instruction matters: search uses 6–11% more output than ordinary early stopping, which allows an objection without a concrete witness. On a separate panel of 60 false statements, presented without proofs, search also uses less output than a proof attempt that must supply a witness when it rejects a statement.

Reading the cost results. Output tokens count generated text and reported reasoning. *Full tokens* add reported input and cached context. These are usage counts, with each model compared against itself under two instructions. Ratios below one in Table 12 mean lower usage for search. Output and full usage can differ substantially: Gemini's full-token ratio is 0.650 across all proof pairs and 0.955 on pairs with no recorded extra execution. These two views show how the observed execution pattern affects total usage. The ratios cover accepted solver attempts; the saved ledgers also account for grading and additional attempts. Appendices H and O give all four instructions and token components, and Appendix E checks the saved references against the source texts. Each instruction condition operates on its own input, without retrieved BugBank records.

G.2 CAN A CHECKED COUNTEREXAMPLE HELP COMPLETE A SUPPLIED CORRECTION?

On 16 Lean tasks, both conditions receive the defect, definitions, a repair template, and the conclusions the corrected program or proof must establish. One condition also receives a checked counterexample. A successful answer must use permitted edits, prove the required conclusions, and preserve the allowed inputs. This design isolates completion of a specified correction.

On the first attempt, Opus 5.5 succeeds on 15 of 16 tasks in each condition. Sonnet 5 succeeds on 13 without the witness and 15 with it; GPT-OSS 120B succeeds on one and two, respectively. Missing submissions count as failures. For Sonnet, the two task pairs with different outcomes both favor the witness. These paired results motivate a larger test of when a counterexample makes a correction easier to complete. Appendix P.5 reports results with up to three attempts and the recorded resource usage.

G.3 CAN EARLIER OBJECTIONS HELP DEVELOP A NEW PROOF?

The retrieval study compares two revision conditions on 50 paired tasks per model. One retrieves BugBank records during review; the other uses self-review. Both share a revision limit and stopping rule. With retrieval listed first, the model-graded correct-proof counts are 45 versus 47 for Sol, 43 versus 42 for Gemini, and 44 versus 43 for Opus 5. Full-token usage increases for Sol and Gemini and decreases for Opus. These results identify two questions for further study: which retrieved

objections apply to the new argument, and when their value offsets the work required to read and use them. Appendix R reports both stopping protocols and all cost components.

H INSTRUCTION COMPARISONS: COMPLETE RESULTS

This appendix compares four ways to assess a proof or statement: search for a counterexample, review every step, stop at the first error, and stop at an error only after constructing a counterexample. The last instruction gives sequential review the same required output as counterexample search. Appendix O supplies the exact prompts and execution settings.

Which grades appear in each table? Table 15 compares search with witness-required review using a common September grading pass. For Opus 5, that pass gives supported-objection counts of 81/83/76/76 for search, full review, early stopping, and witness-required review. The complete four-instruction table also preserves the original Opus 5 grades, 83/84/78/79, under their original grading procedure. This is a regrading of the same saved answers. Original token summaries use 2,000 bootstrap resamples; the common paired comparison uses 20,000 (Efron, 1979).

Table 14 gives every token component for the four instructions on the 100-proof panel. Table 22 does the same for the separate 60-statement panel. The proof totals are reconstructed from saved native usage and Opus 5 telemetry, with a publication receipt covering 1,709 source files. The reconstruction source is `experiments/instructions/source/recompute_proofs100_full_components.py`; portable replay instructions are in `experiments/instructions/README.md`.

H.1 WHAT CHANGES WHEN BOTH INSTRUCTIONS REQUIRE A WITNESS?

The paired study uses 100 proofs sampled from 560 inputs that the construction pipeline had labelled as containing errors with concrete witnesses. This is the instruction-study panel, distinct from the current 100-input benchmark in the 147-dossier human-approved core. Counterexample search chooses a claim to challenge; sequential review inspects the proof in order. The model grader, `gpt-6-astra`, sees the reference defect and can also credit a supported objection to another claim. Table 15 reports the common September grades for all 100 inputs, including the regraded Opus 5 responses. Appendix E checks reference validity and gives the separate 22-record comparison with matching full source texts, where the lower-output pattern also appears.

Search uses 31%, 15%, and 29% less recorded output for Sol, Gemini, and Opus 5, respectively. The corresponding supported-objection differences are $+2$, -5, and $+5$ percentage points. Each interval includes zero and allows both gains and losses, so the accuracy tradeoff remains an open question. Against ordinary early stopping, which allows an objection without a concrete witness, search uses 6–11% more output. Requiring the same form of evidence is therefore central to interpreting the instruction comparison.

How input usage and tool execution affect the result. Adding reported input, cache, and output gives full-token ratios of 0.958 for Sol and 0.650 for Gemini; the Sol interval includes one. The corresponding ratios are 0.977 and 0.955 on the 96 Sol and 94 Gemini task pairs with no recorded additional execution in either condition. These subsets show the usage pattern among the observed runs with that property. Full tokens count usage rather than money, and accepted-attempt totals cover the solver stage. Appendix O reports the false-statement study; Table 28 reports the separate 16-task Lean repair study.

I FROM A COUNTEREXAMPLE TO A COMPLETE EXPLANATION

A counterexample identifies one failure. The next question is whether the same mechanism explains a whole family of failures and suggests a correction. This study followed every record in the earlier 285-dossier collection whose existing annotations described both a failure family and a repair: thirteen records in total. The square-root result had already been formalized. For the other twelve, human verifiers and coding agents developed formal proofs, which then received separate adversarial reviews

by language models. All thirteen packages contain family and repair statements checked by Lean, with their assumptions stated explicitly.

Table 16 gives the complete study roster. Ten of its source dossiers are in the current 147-dossier human-reviewed core; the square-root, array-assignment, and recurrence-sum dossiers are outside that core. The formal follow-ups are provided as mathematical developments of the selected examples, with their review records and individual scope (Appendix D).

From an incorrect total to an exact positivity condition. The signed-cut example illustrates how a correction can expose the next mathematical question. At the source-compatible values $n = 3$, $T = 2$, $d = 20$, $f = -1$, and $a = 19$, the source's total $T + f$ is 1, while the correct additive total $T + a - 2(n-2)d$ is -19. Replacing the incorrect identity resolves the accounting error. The surrounding argument also requires a positive total, so the follow-up proves an exact condition for positivity. The Lean statements `repair` and `exact_positivity` separate these two conclusions.

From a faulty step to a corrected proof. The cycle-deletion, prime-residue, and recursive-increment packages go further: each proves the intended final statement after correcting or replacing the faulty step. The square-root package similarly proves the corrected algorithm for every allowed input. Other rows formalize familiar identities and give exact conditions for their use. Together, the packages show several outcomes of counterexample analysis: a complete failure classification, a corrected local statement, or a proof of the intended conclusion.

Matching the theorem to the source assumptions. Two examples show why the stated assumptions matter. The adjacency-power family includes diagonal matrix entries, while applying it to the source's off-diagonal assertion requires the narrower family with $n \geq 3$. In the liveness example, the equivalence fixes a prefix that includes the violated request; allowing the prefix to end before that request changes the statement. The saved model reviews record these distinctions alongside the formal proofs.

Formal proofs and reproduction. The anonymous repository's `mathematics/additions/index.json` maps these statements to their full Lean sources and recorded checks. The twelve later packages are in `mathematics/additions/explanation_census/`; the square-root proof is in `mathematics/additions/sqrt_boundary/SqrtFailureBoundary.lean`, with its repair proof in `mathematics/additions/sqrt_repair/`. Files using only Lean's libraries have recorded checks under Lean 4.30.0 and 4.34.0. Files using Mathlib have checks under Lean 4.30.0 with the pinned Mathlib project. The square-root package also has a separately written second proof and reproduction record. The artifact index distinguishes that verification from the adversarial model reviews of the other twelve packages. See `formal/README.md` for replay instructions.

J THE MATHEMATICAL RANGE OF THE CERTIFICATE EXAMPLES

A small witness can require substantial mathematics to verify. The examples in Table 17 range from a fixed residue calculation to statements about arbitrary programs, real constants, and unbounded exponents. The middle column says exactly what the certificate proves. The final column explains how that result relates to the selected source passage.

These seven examples were selected from the earlier 284-dossier collection. The number-theory dossier is in the current 147-dossier human-reviewed core. The other six remain available as mathematical examples from the earlier collection, with their formal results and source-review records kept separately. In particular, the functional-analysis calculation establishes a failure of mean-zero preservation, while the human source review identifies the quoted passage as a model definition. This distinction explains why a valid calculation and approval of an objection to the source are separate questions.

The anonymous repository provides the corresponding source passages in `resource/dossiers/`, certificates in `formal/certificates/`, and selected theorem names in `formal/selected_declarations.json`. The crosswalk in `ARTIFACTS.md` links these examples to their files;

`reviews/verified_core/membership.json` records current membership and review evidence.

K STRUCTURAL CASE STUDIES IN DETAIL

K.1 FROM CONJECTURE REFUTATIONS TO STRUCTURAL THEOREMS

A counterexample can reveal more than the failure of one claim. The two cases here show how it can lead to a map of the surrounding mathematical possibilities. In the Schur problem, the counterexample suggests a family of colorings whose exact best density can be proved. In the recoloring problem, it reveals a repeating pattern that classifies and counts a whole family of counterexamples.

Both starting examples refute published conjectures. They were found on 4 September 2026 by `gpt-5.6-sol` at maximum reasoning effort in neutral runs asking the model to solve or refute each conjecture. These runs preceded BugBank exposure, and separate model checks accepted the refutations. The records entered the research-discovery collection on 10 September (Appendix U.2); that collection is separate from the 285-dossier construction stage and the current evaluation core.

The Schur witness already described an infinite family with an exact count. The follow-up identifies the best construction in its larger two-threshold family and quantifies the loss from moving the thresholds. The graph witness was one cycle blowup. The follow-up determines exactly which odd-cycle lengths support the same obstruction and counts all its frozen six-colorings. Appendix S gives the written proofs, and Appendix M documents the formal refutations of the source conjectures. These are mathematical follow-ups to the recorded discoveries.

Formal proofs and review. Each of the original Schur and K_2 propositions below is proved at its full stated scope by a Lean theorem. The Schur proof combines exact finite counts, a limit for every real threshold pair, and optimization of the resulting density. The cycle proof covers every odd $n \geq 5$. Both use Mathlib under Lean 4.30.0. A second model instance checked each certificate adversarially and found no error; human reviews of the two extensions (C01 and S02) are reported in Appendix D. The later AllOdd and Q_3 certificates have their own formal scopes, specified below, and were checked by models without an additional human-review round.

K.1.1 RAINBOW SCHUR TRIPLES: IMPROVING AND OPTIMIZING A COLORING

A *three-coloring* of $[n] = \{1, \ldots, n\}$ assigns each integer exactly one of three labels, called colors. We can name them red, blue, and green, or label them $1, 2, 3$ as below. Each integer has one fixed color, but distinct integers may share a color. The integers assigned the same color form a *color class*.

An ordered Schur triple is $(x, y, z) \in [n]^3$ with $x + y = z$; it is *rainbow* when the three integers belong to different color classes. For example, if 2 is red, 3 is blue, and 5 is green, then $(2, 3, 5)$ is rainbow because $2 + 3 = 5$. If 2 and 5 share a color, the same triple is not rainbow. We count ordered triples, so $(2, 3, 5)$ and $(3, 2, 5)$ are counted separately. The extremal problem asks how to assign the colors so that as large a fraction of these triples as possible is rainbow as n grows.

In *An Unsure Note on an Un-Schur Problem*, Parczyk & Spiegel (2026) give a construction with limiting density $2/5$ and conjecture, immediately after their Theorem 1, that this is optimal. Their construction gives all even integers one color and splits the odd integers into two contiguous color classes.

The recorded counterexample improves that construction by assigning one color to both an initial and a final block of odd integers. For $n = 22m$, all evens have color 1, odds at most $8m$ or greater than $20m$ have color 2, and the remaining odds have color 3. The original refutation proves that this gives exactly $99m^2 + 3m$ rainbow ordered Schur triples for every positive m, with limiting density $9/22 > 2/5$. It therefore disproves the conjectured optimum by improving the constructive lower bound for the original extremal problem.

A preprint submitted on 16 September 2026 by Hegde et al. (2026) gives the same two-threshold construction and $9/22$ lower bound, together with a global upper bound of $8/15$. We acknowledge this overlap without asserting priority. The proposition below addresses the unique optimum and stability within the specified two-threshold template, rather than global optimality.

The split odd color class suggests a two-parameter search: move its two block boundaries and ask which placement works best. The following proposition turns that search into an exact description of the entire template. Here a *template* means this fixed rule for assigning colors, with only the two boundaries allowed to vary.

Proposition 1 (Optimal Schur thresholds within a template)**.** *For $0 \le a \le b \le 1$, color every even integer in $[n]$ color 1, and each odd integer k color 2 when $k/n \le a$ or $k/n > b$, and color 3 otherwise. The limiting rainbow ordered-Schur-triple fraction $D(a, b)$, normalized by $\binom{n}{2}$, has unique maximum $9/22$ at $(a, b) = (4/11, 10/11)$. On $a \le 1/2 \le b$ and $a + b \ge 1$, writing $u = a - 4/11$ and $v = b - 10/11$ gives*

$$D(a, b) = \frac{9}{22} - \frac{5}{2}u^2 + 2uv - \frac{3}{2}v^2. \tag{2}$$

The two contiguous odd-block subfamilies, $a = 0$ or $b = 1$, attain maximum $2/5$.

The public certificate `mathematics/sources/SchurTemplateComposed.lean` combines the counting and optimization arguments. Its declarations `template_count` and `rational_count` give exact counts for integer thresholds and for $n = 2qm$ with rational thresholds in the stability region. The declaration `paper_limit` proves the limit for every real $0 \le a \le b \le 1$ and the literal coloring in the proposition. At odd n, two-sided counting bounds suffice to establish this limit.

Within the same file, namespace `CoPEER.SchurTemplate` contains `unique_global_optimum`, `left_boundary_optimum`, and `right_boundary_optimum`. The theorem `sharp_stability` gives $9/22 - D \ge (4 - \sqrt{5})(u^2 + v^2)/2$ in the stated region. The declarations `quadratic_spectral_equality_iff` and `quadratic_spectral_coefficient_optimal` prove that equality holds exactly when $v = (1 + \sqrt{5})u/2$ and that the coefficient is best possible for this quadratic form. Thus the certificate links the finite colorings to their limiting density and then proves its exact optimum. The written argument appears in Appendix S.1.

Every pair of differently colored odd integers supplies two rainbow triples with one even and one odd addend: their positive difference is even, and it can appear in either addend position. Two odd addends contribute only when their sum stays in $[n]$. Balancing these contributions gives the recorded thresholds: a final color-2 block of normalized width $1/11$ raises the best template density by $1/110$, or 2.27% relative to $2/5$. The quadratic identity gives the exact loss from moving the two thresholds within the stated region; every nonzero displacement reduces the limiting density. This explains why separating the odd color class into two blocks helps, and why the recorded boundaries are optimal within this template (Figure 4).

The counterexample improves the lower bound for the original problem; the proposition then marks the exact boundary of this construction. A better density requires a different coloring rule. Within this rule, the stability formula measures precisely how changing either threshold reduces the density. Finding the best unrestricted three-coloring is the broader extremal question.

K.1.2 Cycle blowups: explaining the failure of recolorability

A proper graph coloring assigns different colors to adjacent vertices. The reconfiguration problem asks whether one proper coloring can be changed into another by recoloring one vertex at a time, keeping every intermediate coloring proper. To represent the allowed moves, the graph $R_k(G)$ has one vertex for each proper coloring of G using colors from a set of size k; an edge joins two colorings that differ at one vertex. A coloring may use fewer than k of the available colors.

The chromatic number $\chi(G)$ is the fewest colors needed for a proper coloring. A graph is *recolorable* if $R_k(G)$ is connected for every $k \ge \chi(G) + 1$. A coloring is *frozen* when no vertex can change color without creating an equally colored adjacent pair. It is therefore an isolated vertex of $R_k(G)$.

In *Recoloring via Modular Decomposition*, Belavadi et al. (2026) ask whether recolorability of every induced subgraph of a prime graph guarantees recolorability of all its clique blowups. This is the preservation direction of their Conjecture 1 (the statement matches Section 6 of the originally reviewed arXiv v2 input). A graph is *prime* if no proper vertex subset of size at least two has all its members sharing the same neighbors outside the subset. A clique blowup replaces each vertex by a nonempty clique and adds all edges between two replacement cliques exactly when their original

vertices were adjacent. In particular, $C_n[K_2]$ replaces each cycle vertex by a two-vertex clique, with all edges between neighboring cliques.

The original counterexample uses C_9, which is prime and whose induced subgraphs are all recolorable. Its blowup $C_9[K_2]$ has chromatic number 5, but repeating the color pairs $\{1,2\},\{3,4\},\{5,6\}$ around the cycle gives a frozen six-coloring. Each vertex has neighbors using every alternative color, so no first recoloring move is possible. A proper five-coloring provides another vertex of the same reconfiguration graph, proving that it is disconnected. Thus the blowup fails the conjecture's conclusion even though the base graph satisfies its hypotheses.

Relation to fall colorings and the source construction. A frozen proper coloring is a fall coloring: every vertex sees all colors in its closed neighborhood (Dunbar et al., 2000). For cycles, $3 \in \mathrm{Fall}(C_n)$ exactly when $3 \mid n$ (Lauri & Mitillos, 2019, Section 4). Applying Shaebani (2009, Theorem 2) with $G = C_n$, $H = K_2$, and three two-color palettes gives a fall six-coloring of $C_n[K_2]$ whenever $3 \mid n$. Shaebani also explicitly discusses $C_9[K_2]$ immediately after Corollary 3 and gives a fall five-coloring, so even this particular graph predates our recorded witness. The source's Figure 3 displays the same periodic palettes on a doubled six-cycle; its base graph contains a non-recolorable induced C_6, so that example does not contradict Conjecture 1 (Belavadi et al., 2026, Section 6). Our case uses this known construction at $n = 9$ and verifies all the hypotheses of the conjecture. The follow-up, developed by the authors with AI assistance, proves the odd-cycle characterization and labeled count below in Lean. The cited constructions establish the existence direction; the follow-up supplies the necessity and counting arguments. We report these connections without asserting literature-wide priority for the characterization or count.

The following proposition identifies exactly which odd-cycle blowups have frozen six-colorings and counts them, showing that the original example is part of an infinite family with exponentially many isolated colorings.

Proposition 2 (Frozen colorings in odd-cycle blowups)**.** *For every odd $n \geq 5$, C_n is prime and every induced subgraph is recolorable. Its blowup $H_n = C_n[K_2]$ has chromatic number 5 and has a frozen proper coloring with the six labeled colors $[6]$ if and only if $3 \mid n$. When $3 \mid n$, there are exactly $90 \cdot 2^n$ frozen labeled colorings, and $R_6(H_n)$ has at least $90 \cdot 2^n + 1$ connected components.*

The single $C_9[K_2]$ witness suggests looking at its repeated color pairs. Call the pair assigned to each replacement clique its *palette*. The key is that the inability to recolor a vertex imposes a local constraint on three neighboring palettes. In a frozen coloring, every three consecutive two-vertex cliques must collectively use all six colors. Overlapping triples of cliques therefore force their palettes to repeat with period three. An ordered partition of the six colors into pairs gives 90 choices; independently assigning each clique's pair to its two labeled vertices gives 2^n choices. Thus $C_{6r+3}[K_2]$, for every $r \geq 1$, is a counterexample to the same preservation claim. The original $C_9[K_2]$ has 46,080 isolated six-colorings, not merely the single exhibited state.

The witness thus becomes a classification: this six-color obstruction occurs exactly at the odd multiples of three, and its multiplicity is known. It persists for arbitrarily large prime base graphs while the blowups keep chromatic number five. Clique substitution creates the obstruction even though every induced subgraph of the base graph is recolorable. This identifies what a repaired preservation theorem must address: additional conditions on the substitution itself. The component count is a lower bound. At the remaining odd lengths, determining connectivity requires studying colorings that do admit individual moves.

These examples map two different counterexample spaces. The Schur analysis varies the boundaries of a coloring and proves where its performance peaks. The graph analysis varies the cycle length and proves where an obstruction exists and how many copies it has. BugBank preserves the witnesses and checking evidence that support this further analysis. Both starting discoveries came from neutral runs before BugBank exposure, as recorded above.

Larger replacement cliques. The two-color palettes lead naturally to a further question: what happens when each cycle vertex is replaced by a clique of size three? For every natural $n \geq 4$, the graph $C_n[K_3]$ has a frozen proper eight-coloring exactly when $8\lceil n/3 \rceil \leq 3n$. This holds precisely for $n \in \{6, 8, 9, 11, 12\}$ and all $n \geq 14$.

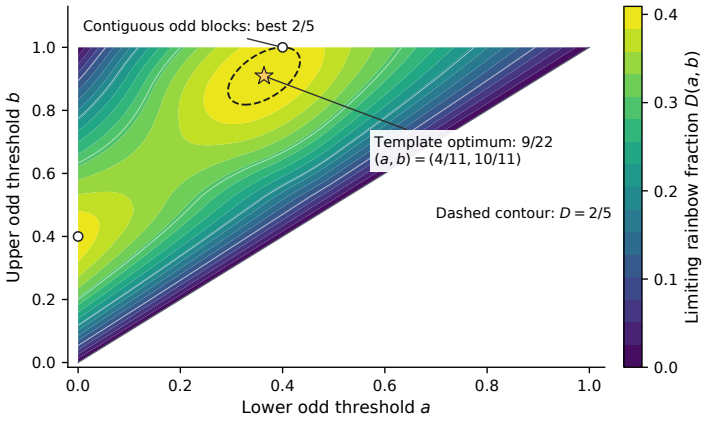


Figure 4: The two-threshold parity template has unique optimum $9/22$ at $(a, b) = (4/11, 10/11)$, compared with $2/5$ for a single contiguous odd block. The analytic proof establishes the optimum throughout the stated domain; the plot illustrates its dependence on the thresholds (generated using NumPy (Harris et al., 2020)).

The necessity argument tracks the cliques containing each color. Those cliques form an independent dominating set of the base cycle: no two are adjacent, and every other clique has a neighbor in the set. For sufficiency, explicit sequences of palettes can be joined to cover every allowed length. Three consecutive K_3 cliques have nine vertex positions for eight colors, so a color can repeat in the two outer cliques. This explains why the three-period argument for K_2 changes. At $n = 13$, the theorem rules out this particular obstruction; the connectivity of the other colorings is a separate question.

Allowing the clique size to vary gives a broader construction. For every odd $n \geq 9$, taking $q = \lceil n/3 \rceil$ yields a graph $C_n[K_q]$ with chromatic number $2q + 1 < n$ and a frozen n-coloring. Thus a suitable clique size removes the divisibility restriction on the base cycle. These later results were developed by the authors with AI assistance. Their graph statements are certified in `mathematics/sources/Q3Law.lean` and `mathematics/sources/AllOdd.lean`; `mathematics/index.json` records their exact scopes and build receipts. Combining these graph statements with the source conjecture's hypotheses is the written mathematical argument, rather than a single additional Lean theorem linking all parts.

L Projective-cube counterexamples and their density boundary

The source claim. In the cyclic group $\mathbb{Z}_N$, addition is taken modulo N. For a size-d multiset $S = \{a_1, \ldots, a_d\}$, its projective cube is

$$\Sigma^* S = \left\{ \sum_{i \in I} a_i : \emptyset \neq I \subseteq [d] \right\}.$$

Thus a projective cube contains every sum formed by choosing a nonempty set of generator positions. Repeated and zero generators are allowed; all sums start from zero, without an additional shift by a fixed group element. A set is *cube-free* if it contains no such cube. Conjecture 4 of Meng (2026) states that, when $d \mid N$, every projective-d-cube-free set $A \subseteq \mathbb{Z}_N$ satisfies $|A| \leq (d-1)N/d$. The source prints $\subset$ in its definition, but its three-generator example explicitly includes $x + y + z$. We therefore include the full index set, as required by that example and by the refutation below. The certificates represent group elements by naturals modulo N, multisets by lists, and nonempty index subsets by Boolean masks with at least one true entry.

From one counterexample to infinite families. The counterexample suggests looking for a small set of residues that every projective cube must meet. Once such a set is found modulo a small integer M, the same obstruction can be repeated in larger cyclic groups. For each row of Table 18 and every integer $k \geq 1$, define

$$A_N = \{x \in \mathbb{Z}_N : x \bmod M \notin F\}.$$

Here F is the set of excluded residues. Reducing a hypothetical cube in A_N modulo M would give a cube that avoids F, contradicting the verified obstruction in that smaller group. The corresponding Lean file proves $d \mid N$, that A_N contains no projective d-cube, the displayed cardinality, and $d|A_N| > (d-1)N$. Each row therefore refutes Conjecture 4 for infinitely many group orders. The family theorem covers every $k \geq 1$; finite computation establishes the obstruction modulo M used in its proof.

How Lean checks the obstruction. A Python generator enumerates the sets of subset sums reachable while all nonempty sums avoid F modulo M. These sets describe the possible states after each new generator is added. Lean checks the generated data: appending any residue must either produce a forbidden nonempty subset sum or lead to a listed state at the next depth. It then checks that no state survives to depth d. A shared soundness proof turns these checks into a theorem about every length-d sequence. Reduction modulo M gives cube-freeness for every group order divisible by M, and counting the repeated residue classes gives the cardinality. Taking $N = \mathrm{lcm}(M, d)k$ also supplies the conjecture's hypothesis $d \mid N$. Each file ends with a theorem negating a general formal statement of Conjecture 4.

All three self-contained files passed Lean 4.30.0 and 4.34.0, using Lean's bundled libraries without Mathlib, `sorry`, or `native_decide`. The checks on batches of states use kernel evaluation and have no axioms. The assembled closure theorem uses `propext`; the family theorems use the standard axioms `propext`, `Classical.choice`, and `Quot.sound`. The anonymous repository provides the three certificates in `mathematics/sources/` and their recorded build evidence in `mathematics/recorded_builds/cube_families.json`. The latter also records the generator and template hashes. The research evidence retains the generator, template, source PDF, compilation logs, and hashes.

How the examples were obtained and reviewed. The $d = 5$ set was found in the model runs of 10 September 2026, first with BugBank and subsequently with plain prompting. It is the pullback (preimage under reduction modulo 16) of the published layered set C_5 of Long & Wagner (2019): the search rediscovered that construction. The $d = 7$ and $d = 8$ sets came from author-agent computer searches on 19 September, followed by formalization. These three certificates belong to the mathematical follow-up study, separately from the earlier 285-record release and its model-experiment results. The construction priority of the $d = 7$ and $d = 8$ sets is not established.

A separate model reader examined the general argument adversarially, regenerated all three files byte-for-byte, and rejected thirteen deliberately incorrect variants. It also reran the $d = 5$ and $d = 7$ files under both Lean versions. For $d = 8$, it inspected the existing compilation logs and independently recomputed the mathematics; its review used the recorded compilation. The original job logs omit source hashes, so the connection to the retained files rests on the receipt, timestamps, and identical regeneration. Separate human reviews of the dimension-5, 7 and 8 dossiers (R02, R13 and R14) are reported in Appendix D. Lean verifies the stated theorems, while the source reading above explains why those theorems address the conjecture.

The same source's auxiliary Conjecture 10 had already been refuted publicly on 26 July 2026 by hypnopump (2026), using a different witness and a Lean proof. That witness contains a projective cube, so its refutation concerns Conjecture 10 rather than Conjecture 4. We make no priority claim for either conjecture. The three certified families motivate the next question: in which dimensions does the underlying layered construction exceed the conjectured density?

Mapping the exact boundary of the layered construction. Write $C_d \subseteq \mathbb{Z}_{2^n}$ for the recursively defined layered construction of Long & Wagner (2019). Each layer has an index; the construction fits in this group when all those indices are at most n. The condition $d \leq n$ guarantees that it fits, although some smaller values of n also suffice. Let $g(d)$ denote the fraction of the group outside the construction. This complement density satisfies $g(1) = 1$ and $g(P + m - 1) = (1 + g(m))/(2P)$ for $P = 2^\ell$, $\ell \geq 1$ and $1 \leq m \leq P$. Later Lean proofs establish the exact comparison with the conjectured bound: for every $d \geq 2$ and fitting n,

$$\frac{|C_d|}{2^n} > \frac{d-1}{d} \quad \Longleftrightarrow \quad d \notin \{2^\ell - 2, 2^\ell - 1, 2^\ell : \ell \geq 1\}.$$

Equality holds exactly when d is a power of two. At the other exceptional dimensions—one or two below a power of two—the density is strictly below the bound. The proofs identify $|C_d|/2^n = 1-g(d)$ whenever the layers fit.

This classification explains both the original example and the reason to search beyond its construction. The theorem `LongWagner.C_five_mod_sixteen` identifies C_5 in $\mathbb{Z}_{16}$ with the certified quotient excluding $\{0, 4, 12\}$. At $d = 7, 8$, however, the layered densities are respectively $27/32 < 6/7$ and $7/8$. The subsequent author-agent certificates reach $7/8 > 6/7$ and $8/9 > 7/8$ in quotients of orders 24 and 36. The comparison locates where this particular layered construction works and where these other constructions improve it; it makes no claim about priority in the literature.

A general quotient lemma proves that taking a pullback preserves cube-freeness and density. The density calculations and pullback lemma are formalized. For the general layered construction C_d, cube-freeness is supplied by Long & Wagner (2019, Section 3.1 and Claim 3.1) and appears as an explicit premise in the corresponding conditional Lean theorems. The separate certificates for $d = 5, 7, 8$ verify cube-freeness directly.

Formal files and build evidence. The anonymous repository places the layered construction results in `mathematics/additions/layered_boundary/`. `LayeredBoundaryAllD.lean` contains `density_gt_iff`, `density_eq_iff`, `card_C_iff_fits`, `card_C_gt_iff` and `card_C_eq_iff`; `PullbackLemma.lean` contains the general cube-freeness and cardinality lemmas. The original file covering all dimensions has an explicit successful Lean 4.30/Mathlib execution receipt. Intermediate bounds were then appended, leaving the headline classification unchanged. The amended file has clean logs bound to its hash and a targeted model review. A replay of those unedited amended bytes on 21 September 2026 exited with status zero and reproduced the logs byte-for-byte; the public `LayeredBoundaryAllD_RECEIPT.json` records that successful build and the source hash. The classification for all dimensions and the pullback lemma have formal build evidence and model review; the human judgments identified above concern the three dimension-5, 7 and 8 dossiers.

M FROM RESEARCH COUNTEREXAMPLES TO PARAMETER FAMILIES

A counterexample can reveal more than one exceptional case: it can identify an entire range of parameters for which a conjecture fails. This appendix traces that progression for graph constructions and polynomial sequences. Each case starts with the recorded witness, explains the step that extends it, and states the resulting theorem. Further examples show how checking an exact polynomial identity or the definitions behind a witness strengthens the mathematical conclusion.

These formal extensions develop research cases outside the earlier 285-record release. The authors developed them with AI assistance, and separate adversarial model reviews checked their formal statements and correspondence to the sources. We report this mathematical work separately from human review and model-discovery measurements. The four primary files in Table 19 compiled under Lean 4.30.0 and 4.34.0 using core Lean, without `sorry` or `native_decide`. The later cycle and Schur extensions use Mathlib at Lean 4.30.0 and have their own compilation records. In the anonymous repository, `mathematics/index.json`, `mathematics/additions/index.json`, and `mathematics/supporting/index.json` link the formal statements, source files, hashes, and build records; `mathematics/discoveries/` records the source versions and discovery provenance.

From a graph obstruction to a refutation under the source assumptions. A frozen coloring of the doubled cycle supplies the obstruction; to refute the source conjecture, the underlying cycle must also satisfy its hypotheses. The theorem `Recol.not_conjecture1` in `mathematics/supporting/sources/PrimeBlowup.lean` proves both parts. A degeneracy lemma (Cereceda et al., 2008) covers arbitrarily many colors and shows that every induced subgraph of C_9 satisfies the required recolorability condition. This completes the source-level argument beyond the finite-coloring certificate in Appendix A.

The repeating coloring pattern then leads to a question about all cycle lengths. Replacing each cycle vertex with a q-vertex clique gives the general construction already known when 3 divides n (Shaebani, 2009). The follow-up adds the necessity characterization for $C_n[K_2]$ and the labeled

count $90 \cdot 2^n$. These are formalized for every odd $n \geq 5$ by `Recol.cycle_blowup_frozen` in `mathematics/sources/CycleBlowupAllN.lean` (Mathlib, Lean 4.30.0), which reuses the general lemmas from `PrimeBlowup.lean` verbatim. The originally reviewed source was arXiv v2; the same conjecture appears in the journal article by Belavadi et al. (2026). The contribution is the source-specific refutation, classification, count, and formalization, with the graph and its frozen colorings credited to the known construction (Shaebani, 2009).

From one Schur coloring to an exact count at every size. The coloring pattern gives a family indexed by m. The proof in `mathematics/supporting/sources/RainbowSchur.lean` counts its rainbow triples directly for every positive m, then uses that count to contradict the proposed asymptotic bound. Triples are ordered and normalized by $\binom{n}{2}$. The resulting lower bound therefore comes from one explicit family of colorings; optimizing over all colorings is a broader problem.

The same construction appears in independent work by Hegde et al. (2026), submitted on 16 September 2026. Our saved model run is dated 4 September 2026. These dates document the two chronologies, with mathematical priority left unassigned. Varying the two thresholds of the construction gives a further extension: an exact optimization and stability analysis, linked to finite-coloring counts in `mathematics/sources/SchurTemplateComposed.lean` (Appendix K.1). This composed certificate contains the two-threshold result and its connection to the literal coloring; `mathematics/recorded_builds/schur.json` records the intermediate proofs and compilation evidence.

From six landmarks to counterexamples for every $n \geq 8$. A vertex of a supertoken graph is a placement of k indistinguishable tokens among n types, allowing several tokens of the same type (Hammack & Smith, 2017, Definition 2.1). We represent it by a list of n natural numbers summing to k. Adjacent vertices differ by moving one existing token to another coordinate. The proof in `mathematics/additions/06_supertoken_metric/SupertokenMetricGeneral.lean` starts from this adjacency relation and derives shortest-walk distance as half the ℓ_1 distance, the formula stated by Baskoro et al. (2025, Proposition 3.1(i), Eq. (2)).

A resolving set is a collection of landmarks whose distance vectors distinguish the vertices (Harary & Melter, 1976; Slater, 1975). The finite witness supplies six landmarks resolving $F_2(K_8)$. Its extension is explicit: pad each landmark with zero coordinates, then add $2e_i$ for each new coordinate. This gives at most $n-2$ landmarks for every $n \geq 8$. The theorem `conjecture_4_2_fails_k2` thus refutes the claimed dimension throughout that range, including a claim that it holds for all sufficiently large n at $k = 2$. The result is an upper bound at fixed $k = 2$; the exact dimensions and the behavior as k grows with n fixed are separate questions. The proof uses the explicit resolving sets and is independent of the separate expensive lower-bound computation for $(8, 2)$. The family extension and formalization were subsequent work by the authors with AI assistance.

From one negative coefficient to every exponent $m \geq 4$. Coefficientwise q-log-convexity requires every coefficient of $V_{n-1,m}V_{n+1,m} - V_{n,m}^2$ to be nonnegative (Liu & Wang, 2007). The recorded witness has $m = 11, n = 3$. Examining the coefficient responsible for that failure yields a uniform obstruction: for every $m \geq 11$, the coefficient of q^2 at $n = 3$ is

$$2 \cdot 15^m + 6^m - 2 \cdot 10^m - 16^m < 0.$$

A kernel-checked calculation at $n = 23$ covers the remaining exponents $4 \leq m \leq 10$. Together these give `not_qlogconvex_of_four_le` in `mathematics/additions/09_sun_qlogconvex/SunQLogConvex.lean`. The result refutes the second assertion in the cited arXiv v4 of Sun (2026b) for every $m \geq 4$. The other three assertions, the cases $m = 2, 3$, and pointwise nonnegativity for real $q > 0$ are distinct questions. The certificate establishes the stated coefficient obstruction and its range; discovery attribution is recorded separately from formal correctness.

Attributing the mathematical developments. The research collection contains eleven model-produced refutation cases and one additional finding already visible within its source. Subsequent checking identified known constructions, including the cube and Schur examples, an index-ratio conjecture revised by its author before our run, and a semigroup contradiction in the source's own table. We retain these distinctions so that a reader can see both the mathematical conclusion and what was contributed at each stage: identifying a witness, checking it against the source, extending its

range, or formalizing the result. The cases are reported as source-linked mathematical criticism and formal completion, with discovery attribution stated individually.

A rook-count witness and the ranges checked in the source. For the permutation $w = (2, 7, 4, 8, 1, 6, 3, 5)$, the Lean kernel computes 155 complete non-attacking rook placements on its permutation crossword grid. Since $155 \equiv 3 \pmod 4$, this count violates the proposed condition. Using general definitions of permutation grids, words, and complete placements, the theorem `not_conjecture310` refutes the necessity direction of Conjecture 3.10 of Lewis & Won (2026). Two size ranges matter here. The board has size eight, within the board sizes enumerated in the source, whereas its count of 155 exceeds the $r \leq 108$ count range summarized immediately before the conjecture. The restatement given to the model omitted that source sentence. The certificate establishes this witness and this direction of the conjecture; the converse direction, a corrected spectrum of possible counts, and the smallest possible witness are further questions.

Strengthening a Toda witness to survive a change of variable. A coefficient sequence is unimodal if it never decreases before a peak and never increases after it. The recorded Toda witness of 10 September has numerator $1 + q^2$. Its coefficients are not unimodal when consecutive powers of q are counted, but become unimodal when the polynomial is written in q^2. Indeed, it is the source's type-A_2 example under that substitution. This observation suggests a stronger target: find a failure that persists when the polynomial is written in any fixed integer power of q, a change called common-exponent regrading. During formalization on 19 September, the authors, with AI assistance, found a type-B_2 witness at $\alpha = 2\alpha_1 + 2\alpha_2$ with numerator

$$1 + q + 3q^2 + 2q^3 + 5q^4 + 2q^5 + 3q^6 + q^7 + q^8.$$

The nonzero coefficient of q forces the regrading step to be one, and $3 > 2 < 5$ gives an interior dip in the coefficient sequence. The theorem `Exact.not_conjecture73_exact` formalizes the refutation for general root-system data using the exact defining recurrence after denominators are cleared. It checks full polynomial identities. The connection from that recurrence to the rational functions in Definition 1.1 of Labelle (2026) uses algebra in $\mathbb{Q}(q)$ and the source's uniqueness result in the written argument. The original model witness and the later, stronger witness retain their separate discovery records.

The rook and Toda proofs are supplied as `mathematics/supporting/sources/RookSpectrum.lean` and `mathematics/supporting/sources/TodaNumerator_Exact.lean`. Both compiled under Lean 4.30.0 and 4.34.0 without `sorry` or `native_decide`, and received separate adversarial model review. They are subsequent formalizations, reported separately from the earlier release and its discovery counts; mathematical priority is left unassigned. Selected human judgments on the rook and Toda dossiers (R05 and T11) appear in Appendix D.

Checking definitions on additional examples. A witness can satisfy two different definitions that disagree elsewhere. The C_2^4 short-zero-sum example makes this issue concrete. In recorded tests, replacing the last invariant factor by the first in the definition of the exponent still allows the refutation to compile: both are 2 at $[2, 2, 2, 2]$. Reversing divisibility or using modulus 2 throughout also leaves that witness intact.

A companion file provides a separately written, Lean-equivalent formulation and checks true instances at $C_2 \oplus C_4$. Those checks reject the altered definitions. These mutation probes (Jia & Harman, 2011) test the definitions beyond the refuting instance; correspondence to the source prose is reviewed separately. The underlying refutation concerns an inconsistency within the source, and these additional checks strengthen its formalization. The base and companion proofs, together with their successful compilation logs, are listed in `mathematics/supporting/index.json` as `ShortZeroSums_Conjecture.lean` and `ShortZeroSums_Conjecture_Sanity.lean`. The mutation outcomes are documented in the recorded review; individual mutant logs were not retained.

Turning two polynomial witnesses into exact identities. Two witnesses produced in neutral model runs were subsequently formalized by the authors with AI assistance. These proofs add exact polynomial certificates to the mathematical checks accompanying the recorded witnesses. For the

fixed-m fake-Gaussian positivity assertion of Gatzweiler & Krattenthaler (2025), let

$$F(q) = \prod_{i=1}^{21} \left(\frac{1 - q^{22+i}}{1 - q^i} \right)^{a_i}, \qquad a_6 = a_{21} = 1, \quad a_{14} = a_{18} = a_{20} = 2,$$

with all other $a_i = 0$. The certificate checks the full numerator–denominator identity and proves that F is the exact integer polynomial represented by its 177 coefficients, with $[q^{43}]F = -1$. The declaration `not_conjecture4` refutes the encoded general statement. A general quotient-uniqueness lemma ensures that every polynomial representative of the quotient has this same negative coefficient. The result concerns Conjecture 4; the symmetric positivity assertion, the source's other conjectures, and eventual positivity as m grows have separate mathematical scope.

For the factorial-ratio unimodality assertion of Warnaar & Zudilin (2025), the recorded tuples are $a = (2, 3, 5, 12)$ and $b = (1, 1, 1, 4, 6, 9)$. Writing $[j]_q = 1 + q + \cdots + q^{j-1}$, their quotient satisfies

$$D(a, b; q) = [10]_q [11]_q (1 - q^2 + q^4).$$

The full coefficient sequence of $(1 + q)D$ is not unimodal. The certificate proves the quotient identity and its uniqueness, then derives the general negations `not_conjecture5` and `not_conjecture5Sym`. The Landau hypothesis (Warnaar & Zudilin, 2025, Eq. (1)) is certified for every general floor profile, including every rational input, and for every structure satisfying the listed floor properties. The written argument identifies the real numbers with such a structure; the retained Lean proof covers the abstract structure and rational inputs. The resulting refutation concerns unimodality, leaving the separate positivity conjecture as a distinct question.

These proofs are supplied as `mathematics/additions/07_fake_gaussian/FakeGaussian.lean` and `mathematics/additions/04_qrious_unimodality/QriousUnimodality.lean`. Both have successful recorded builds under Lean 4.30.0 and 4.34.0 without `sorry` or `native_decide`, and separate adversarial model reviews. These subsequent formalizations preserve the original model-witness provenance and are reported separately from the earlier corpus; mathematical priority is left unassigned. Selected human judgments on the fake-Gaussian and unimodality dossiers (R07 and R04) appear in Appendix D.

N THE REVIEWED RESOURCE AND ITS DOCUMENTATION

The current resource contains 147 human-approved dossiers from 137 source units. Each retained dossier has a positive human judgment on three questions: does the objection apply to the source claim, is the counterexample mathematically valid, and does the Lean statement express the same objection? We also check that the review applies to the current version. The inclusion rule excludes records with an unresolved negative or uncertain human judgment or an unresolved source-assumption issue. Appendix D describes the completed multistage human review and the resulting inclusion decisions.

Review and version evidence. For 113 dossiers, the scientific content matches the content bound to a saved positive review. For the other 34, the authors confirmed that the earlier positive review applies to the current dossier. These two forms of evidence are recorded separately. The core contains 147 certificate paths and 141 distinct certificate hashes. After the separate checks of reference-line annotations, it supplies 100 primary benchmark inputs and 131 scorable inputs in total. The additional 31 inputs contain shorter source texts. Resource membership, review evidence, and benchmark eligibility are recorded for each dossier.

What a dossier contains. Each dossier connects the source text and quoted claim to a concrete witness, its mathematical explanation, and a Lean certificate. Hashes identify the source packet, witness fields, Lean source, and selected declarations. The selected declarations compile successfully. A separate check lists the axioms used by each selected theorem and confirms that only the allowed foundational axioms occur. The saved build receipts identify the compiler environment, dependencies, commands, and compiler and axiom logs. Reused receipts keep the date and command of the original successful build.

The HTML and JSON dossiers provide the Lean source and project-local dependencies, exact theorem statements, build evidence, source context, witness history, and recorded reviews. Replacement

witnesses appear alongside their predecessors, making the correction and its reason inspectable. Appendix D.6 develops two such examples in detail.

N.1 Definitions for describing the collection

Source identifies where the claim appeared. A *mathematical topic* describes the subject of that claim. A *witness object* describes the counterexample itself, such as a graph, number, function, or algorithm trace. These classifications answer different questions: a graph, for example, can expose an error in an algorithms argument.

An *error mechanism* describes the immediate failed inference: a definition or domain condition, a quantifier, an omitted case, an algebraic or order argument, a count, a structural or analytic inference, or an algorithm invariant. The existing mechanism annotations select the operation closest to the quoted error and retain uncertainty flags.

Reproduction. The anonymous repository provides current membership, its inclusion rule, and version evidence in `reviews/verified_core/`. From the repository root, `python -B reviews/verified_core/verify.py -release .` reconstructs the retained set from the saved reviews and checks its scientific content, certificates, and benchmark masks. `benchmark/verified_core/replay.py` reproduces the benchmark scores. The core README gives the full commands. Earlier census tables, complete annotation evidence, and coding checks are preserved in `censuses/`, with portable inspection instructions in its README.

O Instruction-study procedures and reproducibility

This appendix gives the inputs, prompts, execution settings, grading rules, and token accounting for the instruction comparisons in Table 13. The anonymous repository provides the saved inputs, answers, model grades, and numerical usage records in `experiments/instructions/`. Its `README.md` explains how to reproduce the tables from those saved records. The prompt quotations below reproduce `experiments/instructions/reproduction/prompts.json`, with only LaTeX escaping and dash formatting changed.

O.1 Study design

Question and comparisons. The study asks how the instruction changes the effort needed to identify a false claim. The initial experiment applied four instructions to 100 proofs using Opus 5 at high effort in the Claude Code execution system. The reproduction evaluates the same 100 proofs with `gpt-5.6-sol` and Gemini 3.1 Pro, and a separate set of 60 statements without proofs with all three models. The saved Opus 5 proof answers supply its paired comparisons. We compare instructions within each model, using the token accounting provided by its execution system.

The 100-proof input set. Each *proofs100* record contains a problem statement and a numbered proof, stored in the JSON fields `record_id`, `problem`, and `proof_numbered`. The manifest fixes the input list and the input and reference files by SHA-256 hashes. These are the earlier instruction-study inputs; the current claim-selection benchmark uses its own 100 inputs from the 147-dossier human-approved core.

The 60-statement input set. The *claims60* panel contains 60 statements assessed as false in a skeptical review. Each has a reference counterexample. Executable checks support 57 references: 47 are marked fully verified and 10 have exhaustive-search witnesses. The other three have literature support, recorded as two partial checks and one citation-only check in the `verification_level` field. Each model receives the statement with its definitions, conventions, and an opaque identifier. The claims and witnesses are separate from BugBank records.

Some definitions explicitly suggest looking for a refutation. Before running the study, we specified a text rule to identify these hints: the case-insensitive regular expression `/counterexample|claim under test|the claim (asserts|quantifies)|to be refuted|refut/i` applied to the definitions field. It flags 13 of the 60 claims. Table 24 gives every comparison on all 60 claims and on the remaining 47.

Arms and prompts. Each cell sends one prompt. The proof template is

```
You are assessing a mathematical proof for correctness.
Read {input_path} -- it contains the problem statement and a
line-numbered proof.  Use ONLY that file:  no web, no other
files, no code execution.  (Reading that file with your
file-reading tool, or with a single shell command that only
prints it, is allowed and is not code execution.)
{method}
Honesty rules:  do not assume the proof is flawed; many
proofs are fine.  Do not pad.  In your structured output set
record_id to exactly {record_id_json} and arm to exactly
"{arm}".
```

with `{method}` one of the instructions below (the arm), `{input_path}` the record's input file, and `{record_id_json}` and `{arm}` the values the answer must echo. *refute*:

```
METHOD you must follow -- COUNTEREXAMPLE-FIRST: read the
proof looking for its riskiest concrete claims, and try to
construct ONE concrete counterexample (explicit numbers,
explicit small structure, explicit trace) to some claim the
proof actually relies on.  The moment you have a
counterexample you have checked carefully, report INVALID
with that counterexample and STOP -- do not audit the rest
of the proof.  If after honest effort you cannot refute
anything and believe the proof is correct, report VALID with
counterexample "".
```

verify_full:

```
METHOD you must follow -- COMPLETE VERIFICATION: verify the
proof step by step.  For EVERY numbered step, decide whether
it is justified given the problem and the preceding steps,
and record a one-line justification or objection per step in
assessment_summary.  You must judge every step even after
finding a problem.  Verdict INVALID if any step fails (name
the first failing step in flawed_step), VALID otherwise.
counterexample may be "" -- your job is verification, not
witness construction.
```

verify_earlystop:

```
METHOD you must follow -- VERIFICATION WITH EARLY STOP:
verify the proof step by step in order.  For each numbered
step, decide whether it is justified given the problem and
the preceding steps.  You MAY stop at the first step you are
certain fails; report INVALID with that step in flawed_step.
If you reach the end, report VALID. counterexample may be ""
-- your job is verification, not witness construction.
```

verify_earlystop_witness:

```
METHOD you must follow -- VERIFICATION WITH EARLY STOP,
WITNESS REQUIRED: verify the proof step by step in order.
For each numbered step, decide whether it is justified given
the problem and the preceding steps.  You MAY stop at the
first step you are certain fails -- but a verdict of INVALID
is only complete when you ALSO produce a CONCRETE
counterexample (explicit numbers, explicit small structure,
explicit trace) to that failing step, checked carefully, in
the counterexample field.  This arm exists to equalize
deliverables:  your INVALID must come with the same artifact
a counterexample-first assessment produces.  If you reach
the end without a definitive failure, report VALID with
counterexample "".
```

For the claims the template reads

```
You are assessing a mathematical statement for truth.  Read
{input_path} -- it contains the statement and the
definitions and conventions under which it is to be read.
Use ONLY that file:  no web, no other files, no code
execution.  (Reading that file with your file-reading tool,
or with a single shell command that only prints it, is
allowed and is not code execution.)
{method}
Honesty rules:  do not assume the statement is false; many
statements are true.  Do not pad.  In your structured output
set record_id to exactly {record_id_json} and arm to exactly
"{arm}".
```

and the verification instruction becomes a complete proof attempt; the counterexample-first instruction is unchanged in method: *refute*:

```
METHOD you must follow -- COUNTEREXAMPLE-FIRST: read the
statement looking for its riskiest implication, and try to
construct ONE concrete counterexample (explicit numbers,
explicit small structure, explicit trace) that satisfies
every hypothesis of the statement and violates its
conclusion.  The moment you have a counterexample you have
checked carefully, report INVALID with that counterexample
and STOP -- do not attempt a proof.  If after honest effort
you cannot refute it and believe the statement is true,
report VALID with counterexample "".
```

prove_full:

```
METHOD you must follow -- COMPLETE PROOF ATTEMPT: attempt to
prove the statement.  Write a complete numbered step-by-step
argument in assessment_summary and, for EVERY numbered step,
decide whether it is justified given the definitions and the
preceding steps, recording a one-line justification or
objection per step.  You must judge every step even after
finding a problem.  Verdict VALID if every step is justified
and the argument proves the statement; INVALID if the
argument fails and you conclude the statement is false (name
the first failing step in flawed_step).  counterexample may
be "" -- your job is verification by proof, not witness
construction.
```

prove_earlystop:

```
METHOD you must follow -- PROOF ATTEMPT WITH EARLY STOP:
attempt to prove the statement step by step in order.  For
each numbered step, decide whether it is justified given the
definitions and the preceding steps.  You MAY stop at the
first step you are certain cannot be justified; report
INVALID with that step in flawed_step.  If your argument
completes, report VALID. counterexample may be "" -- your
job is verification by proof, not witness construction.
```

prove_earlystop_witness:

```
METHOD you must follow -- PROOF ATTEMPT WITH EARLY STOP,
WITNESS REQUIRED: attempt to prove the statement step by
step in order.  For each numbered step, decide whether it is
justified given the definitions and the preceding steps.
You MAY stop at the first step you are certain cannot be
justified -- but a verdict of INVALID is only complete when
you ALSO produce a CONCRETE counterexample (explicit
```

```
numbers, explicit small structure, explicit trace) to the
statement, checked carefully, in the counterexample field.
This arm exists to equalize deliverables:  your INVALID must
come with the same artifact a counterexample-first
assessment produces.  If your argument completes without a
definitive failure, report VALID with counterexample "".
```

Within each input set, the four instructions share the same surrounding prompt, output schema, grading rubric, and grader. Only the method block changes. The shared proof text has 338 characters before the method and 179 after it; the corresponding statement counts are 367 and 186. Each condition receives one input file and no retrieved BugBank records.

The answer schema requires six string fields: `record_id`, `arm`, `verdict`, `flawed_step`, `counterexample`, and `assessment_summary`. The verdict is `VALID`, `INVALID`, or `CANNOT_DECIDE`; extra fields are disallowed. The initial study called the last field `reasoning_summary`. During the reproduction, the Opus 5 interface rejected that field name, so all models used `assessment_summary`.

Execution setup. Each model–input–instruction combination runs as one command-line session under Slurm (`https://slurm.schedmd.com/`) in a bubblewrap sandbox (`https://github.com/containers/bubblewrap`). The sandbox hides the research repository, scratch and temporary directories, and previous model sessions. It exposes the session's working directory, answer schema, and one input file, with credentials supplied separately. The host file system is otherwise visible, and networking is available. The prompt permits only reading the input, either with a file-reading tool or a shell command that prints that file.

The configurations are:

- Opus 5: `claude -p`, high effort, only the `Read` tool, with the CLI's JSON-schema output option.
- GPT-5.6 Sol: `codex exec`, codex-cli 0.154.0, high reasoning effort, approvals disabled, read-only sandbox, and the CLI's output schema.
- Gemini 3.1 Pro: the High variant in Google Antigravity CLI 1.2.3, high effort, JSON-schema output, and a 55-minute print timeout.

Codex and Antigravity retain additional tools that their interfaces do not allow the driver to remove. A rule specified before the runs therefore flags any Codex shell command other than printing the input, and any Antigravity tool step other than viewing the input or finishing. Such calls occur in one to three Codex cells per proof instruction, two to six Gemini cells per proof instruction, and up to three Gemini cells per statement instruction. Table 11 compares search and witness-required review with and without these task pairs; Table 24 does so for all three statement comparisons. Answers returned without reading the input are rejected and retried.

Retries and recovery. The driver runs six concurrent sessions for each Sol assessment run and four for the Opus and Gemini runs and each grading job. A session has a 3,600-second limit. Transport failures, timeouts, refusals, and invalid answer formats allow up to three attempts. A rate- or usage-limit signal causes a 20-minute pause without consuming an attempt. Claude provides a structured signal; Codex and Antigravity use matching CLI error messages. Six consecutive failures trigger a separate 20-minute pause, and three such pauses stop the driver. A cell that exhausts its attempts remains without an answer. Jobs relaunched after the recorded Codex interruption received an explicitly enlarged limit of six attempts.

The ledger records every attempt and its prompt hash. Slurm job identifiers in the lock files allow an interrupted run to resume while keeping completed answers. The driver can also recover an answer from the newest usage record if recording was interrupted; all reported runs completed without this fallback.

Token accounting. The main ratios use output tokens from the accepted attempt: Claude's per-model `outputTokens`, Codex's `turn.completed` output, and Gemini's result usage. These counts include each engine's reported reasoning. Uncached and cached input come from the same

receipts. Additional attempts are reported separately, along with attempts whose usage could not be recovered. When an interrupted shell failed to save a usage file, available usage was recovered from the event log.

In the initial Opus study, agents also wrote their answers through a tool call, adding that text to output usage. The reproduction emits the structured answer once. This changes the absolute count by roughly one answer per cell in every instruction condition, while preserving the paired instruction design.

O.2 Token accounting for the 60-statement panel

The accounting joins all 720 accepted usage records to their native receipts and completed ledger attempts, and joins every answer and grade by record and instruction. All 120 input and reference files match the source manifest. The results and their 3,018 source-file hashes are supplied in `experiments/instructions/corrected_claims60_accounting/`. Recomputing usage leaves the saved answers, grades, and output-token counts unchanged.

Counting each component once. Sol's native input count already includes cached reads. Gemini's `input_tokens` contains uncached input, so cached reads must be added. An earlier Gemini summary subtracted cache reads from that field. Adding these disjoint components gives search/witness totals of 2,480,272/2,745,952 and a ratio of 0.903, correcting 0.872. The search/full-proof ratio changes from 0.978 to 1.031; output ratios are unchanged. Opus receipts separately report uncached input, cache creation, cache reads, and output. Reasoning is counted once within output for every model.

Opus 5 reports usage for both the target model and auxiliary models. The target-model search/witness totals are 751,620/895,394, a ratio of 0.839. Including reported auxiliary usage gives 824,253/970,902 and a ratio of 0.849. Table 21 uses this all-reported total, while the saved accounting also retains the target-model view. Sol and Gemini supply native aggregate totals without a separate auxiliary-model breakdown.

Do hints or additional executions change the pattern? Table 24 compares search separately with a full proof attempt, ordinary early stopping, and early stopping requiring a witness. The no-hint subset removes the 13 claims identified by the pre-run text rule. The no-additional-execution subset keeps a task pair only when both runs meet that condition. This latter subset describes observed behavior; tool permissions were not randomly assigned. Opus has no execution-count field.

On the 47 no-hint claims, search/witness output ratios are 0.535, 0.689, and 0.396 for Sol, Gemini, and Opus; the corresponding all-reported full-token ratios are 0.951, 0.942, and 0.809. Both measures help interpret the efficiency result: shorter generated answers can coexist with similar total usage. The complete table also shows the extra output used by search relative to ordinary early stopping.

Table 24: Claims60 instruction and subset sensitivity. S denotes search; F full proof attempt, O ordinary early stop, and W early stop requiring a witness. Counts use preserved grades; token ratios are S/comparator. Tool-free means no recorded additional execution in either paired cell. Exclusions are descriptive. Opus tool use is unrecorded, so no tool-free row is inferred. Output is the target model; full tokens include every model reported in the accepted receipt.

Model	Arm	Subset	n	Catches S/C	Output	Full
Sol	F	All	60	60/59	0.544	0.954
Sol	F	No hint	47	47/47	0.507	0.949
Sol	F	Tool-free	60	60/59	0.544	0.954
Sol	F	Both exclusions	47	47/47	0.507	0.949
Sol	O	All	60	60/42	1.347	1.015
Sol	O	No hint	47	47/36	1.260	1.011
Sol	O	Tool-free	60	60/42	1.347	1.015
Sol	O	Both exclusions	47	47/36	1.260	1.011
Sol	W	All	60	60/59	0.593	0.959
Sol	W	No hint	47	47/46	0.535	0.951
Sol	W	Tool-free	60	60/59	0.593	0.959
Sol	W	Both exclusions	47	47/46	0.535	0.951
Gemini	F	All	60	58/42	0.858	1.031
Gemini	F	No hint	47	46/37	0.768	1.028
Gemini	F	Tool-free	59	57/42	0.844	0.968
Gemini	F	Both exclusions	46	45/37	0.742	0.947

Model	Arm	Subset	n	Catches S/C	Output	Full
Gemini	O	All	60	58/43	1.275	1.210
Gemini	O	No hint	47	46/36	1.130	1.235
Gemini	O	Tool-free	59	57/43	1.352	1.046
Gemini	O	Both exclusions	46	45/36	1.198	1.027
Gemini	W	All	60	58/57	0.739	0.903
Gemini	W	No hint	47	46/45	0.689	0.942
Gemini	W	Tool-free	57	56/55	0.654	0.923
Gemini	W	Both exclusions	45	45/44	0.610	0.915
Opus	F	All	60	60/60	0.888	0.983
Opus	F	No hint	47	47/47	0.845	0.977
Opus	O	All	60	60/60	1.355	1.043
Opus	O	No hint	47	47/47	1.275	1.034
Opus	W	All	60	60/60	0.476	0.849
Opus	W	No hint	47	47/47	0.396	0.809

Additional attempts. The solver ledgers contain 730 attempts. Sol has no rejected attempt. Gemini has three with known full usage totaling 29,135 tokens. Opus has six failed attempts with unknown usage and one rate-limited attempt with known target-model usage of 283,653 tokens. Unknown usage is recorded as unknown, including when a receipt lists zero API messages. The accepted-attempt ratios therefore describe completed solver answers. Grading, later checking, and unrecorded interrupted attempts require separate accounting to estimate end-to-end cost.

Grading. `gpt-6-astra`, using Codex CLI at maximum effort, grades the new answers and regrades the 400 original Opus proof answers. It receives the assessment and held-out reference inline, in the same read-only sandbox, with its shell tool available. The recorded September 15 pause contains 1,700 completed grading cells, six with shell commands, including three attempted arXiv downloads. The completed saved dataset contains 1,920 grading cells, seven with shell commands. The extra 220 completions and additional command-bearing cell are documented in `experiments/instructions/reproduction/grader_cohort.json`; numerical usage and recorded commands are in its adjacent `grade_usage/` directory. The 1,700-cell pause cohort was reconstructed from timestamps.

The following saved rubrics use the study's reference labels. Their judgments are evaluated alongside the source and witness checks in Appendix E. The proof rubric is:

```
You are grading one proof assessment against held-out ground
truth.
Read these two files:
1.  {assessment_path} - the assessment (verdict, claimed
flawed step, counterexample).  If this file does not exist,
use grade MISSED with justification "assessment absent
(quota interruption)".
2.  {truth_path} - the ground truth:  the confirmed flaw and
its independently verified witness.
Every proof in this sample IS flawed (pipeline-confirmed).
Grade the assessment:
- CAUGHT_MATCHING: it reported INVALID and its flawed
step/counterexample targets the SAME defect as the ground
truth (rephrasing and a different-but-valid witness for the
same claim both count as matching).
- CAUGHT_DIFFERENT: it reported INVALID with a DIFFERENT
flaw, and after checking its support concretely yourself,
that flaw is genuine.
- MISSED: it reported VALID or CANNOT_DECIDE.
- FALSE_CATCH: it reported INVALID but the stated flaw does
not hold up under your concrete check (wrong math, misread
claim, invalid witness).
Be strict about FALSE_CATCH vs CAUGHT_DIFFERENT: verify, do
not take the assessment's word.
Set record_id exactly {record_id_json}, arm "{arm}".
```

The statement rubric adds two outcomes: a correct verdict without a checkable refutation, and an object the grader cannot verify. Both input sets share this grade list. In Sol's early-stop proof

condition, one answer received each added grade; each is counted separately from supported and unsupported objections. The exact statement rubric is:

```
You are grading one assessment of a mathematical statement
against held-out ground truth.
Read these two files:
1.  {assessment_path} - the assessment (verdict VALID = the
statement is true, INVALID = false, CANNOT_DECIDE; a claimed
failing step; a counterexample).
2.  {truth_path} - the ground truth:  the statement is FALSE
and this file gives an independently verified counterexample
(the reference witness) and, where recorded, why the
statement fails.
Every statement in this sample IS false (reviewer-confirmed,
witness verified).  Grade the assessment:
- CAUGHT_MATCHING: it reported INVALID and its
counterexample (or, if it gave no explicit witness, its
stated reason) is a VALID refutation of the same kind as the
reference (the same object or family, or the same failing
mechanism; a different-but-valid witness of the same kind
counts as matching).
- CAUGHT_DIFFERENT: it reported INVALID with a DIFFERENT
valid refutation (a genuinely different counterexample
object or a different correct reason), and after checking
its support concretely yourself, that refutation is genuine.
- MISSED: it reported VALID or CANNOT_DECIDE.
- FALSE_CATCH: it reported INVALID but the stated
counterexample or reason does not hold up under your
concrete check (wrong math, an object that violates a
hypothesis or satisfies the conclusion, a misread statement,
or no checkable support at all).
- CAUGHT_UNSUPPORTED: it reported INVALID (the correct
verdict) but gave neither a counterexample nor a
self-contained argument that the statement is false.  A
'stated reason' counts as a refutation only if it is itself
a complete argument (for example a parity or counting
argument, or a named object together with why it meets every
hypothesis and fails the conclusion).  'My proof attempt
fails at step k', 'the hypotheses do not force the
conclusion', or naming a gap in a would-be proof is NOT a
refutation.
- UNVERIFIED: it reported INVALID with a named or large
object (a famous graph, a structure with dozens of vertices,
a construction you cannot check by hand from the
assessment's own text) that is NOT the reference object and
that you cannot verify concretely.  Never guess:  use
UNVERIFIED rather than CAUGHT_DIFFERENT or FALSE_CATCH in
that case.  If the object IS the reference object (or an
explicitly stated variant the truth file already covers),
grade CAUGHT_MATCHING.
Be strict about FALSE_CATCH vs CAUGHT_*:  verify concretely,
do not take the assessment's word; a counterexample must
satisfy every hypothesis in the definitions and violate the
conclusion.  FALSE_CATCH is only for INVALID whose supplied
witness or argument is wrong under your concrete check.
Clauses in the definitions that describe what a
counterexample must include are guidance to the solver:
grade on whether the object refutes the statement under the
stated conventions, performing any omitted verification
(trace, tie-break check, non-isomorphism) yourself; omission
of demanded verification text alone never makes a
FALSE_CATCH.
Set record_id exactly {record_id_json}, arm "{arm}".
```

Tier (n)	Model	Search				Witness review			
		Match.	Diff.	False	Missed	Match.	Diff.	False	Missed
All (100)	GPT-5.6 Sol	86	9	3	2	77	16	4	3
	Gemini 3.1 Pro	78	6	1	15	81	8	0	11
	Opus 5	77	4	2	17	72	4	3	21
Ledger-joined (71)	GPT-5.6 Sol	61	7	2	1	54	13	3	1
	Gemini 3.1 Pro	56	3	1	11	58	6	0	7
	Opus 5	55	3	2	11	51	3	3	14
Kept as written (43)	GPT-5.6 Sol	39	3	0	1	31	11	1	0
	Gemini 3.1 Pro	36	0	1	6	34	6	0	3
	Opus 5	35	2	1	5	30	3	2	8
Audit-supported (42)	GPT-5.6 Sol	38	3	0	1	31	10	1	0
	Gemini 3.1 Pro	35	0	1	6	33	6	0	3
	Opus 5	34	2	1	5	30	3	2	7
Whole-source match (22)	GPT-5.6 Sol	20	2	0	0	17	4	1	0
	Gemini 3.1 Pro	21	0	0	1	18	2	0	2
	Opus 5	19	0	1	2	17	2	1	2
Exact source text (15)	GPT-5.6 Sol	15	0	0	0	14	0	1	0
	Gemini 3.1 Pro	15	0	0	0	14	0	0	1
	Opus 5	13	0	1	1	11	1	1	2
Invalidated (13)	GPT-5.6 Sol	9	4	0	0	13	0	0	0
	Gemini 3.1 Pro	8	1	0	4	11	0	0	2
	Opus 5	9	1	0	3	10	0	1	2
Unresolved (15)	GPT-5.6 Sol	13	0	2	0	10	2	2	1
	Gemini 3.1 Pro	12	2	0	1	13	0	0	2
	Opus 5	11	0	1	3	11	0	0	4

Table 9: Common September response grades by source-checking group. Match. and Diff. are supported matching and alternative objections. False is an unsupported objection; Missed includes responses reporting VALID or CANNOT_DECIDE. The groups are nested or overlapping, and "audit-supported" refers to the recorded automated checks.

Table 10: Token components for search and witness-required sequential review. Each row sums the accepted solver attempts on the same 100 tasks. Full tokens combine uncached input, cached input, and output; reasoning is included in output. Grading and additional attempts are accounted for separately. Input components follow each model's execution system. Opus 5 uses provisional transcript-based accounting, whose totals can repeat usage records.

Model	Instruction	Uncached input	Cache	Output	Full tokens
GPT-5.6 Sol	Search	665,020	2,256,000	122,923	3,043,943
GPT-5.6 Sol	Witness review	676,289	2,321,408	178,429	3,176,126
Gemini 3.1 Pro	Search	1,240,069	2,242,187	1,101,377	4,583,633
Gemini 3.1 Pro	Witness review	1,482,239	4,285,019	1,289,643	7,056,901
Opus 5	Search	2,995,056	13,577,136	695,247	17,267,439
Opus 5	Witness review	3,259,632	16,534,085	973,717	20,767,434

Model	Cells with tool use (S/R)	Tasks	Catches	Output ratio	Full-token ratio	Input-only ratio
GPT-5.6 Sol, all cells	1/3	100	95/93	0.689 [0.594, 0.802]	0.958 [0.833, 1.083]	0.974 [0.846, 1.104]
GPT-5.6 Sol, tool-free tasks	0/0	96	92/90	0.676 [0.604, 0.761]	0.977 [0.971, 0.983]	0.996 [0.995, 0.997]
Gemini 3.1 Pro, all cells	2/5	100	84/89	0.854 [0.789, 0.921]	0.650 [0.441, 0.941]	0.604 [0.382, 0.969]
Gemini 3.1 Pro, tool-free tasks	0/0	94	81/84	0.843 [0.774, 0.916]	0.955 [0.937, 0.976]	0.998 [0.997, 0.999]
Opus 5, all cells	n/r	100	81/76	0.714 [0.620, 0.814]	0.831 [0.762, 0.902]	0.837 [0.766, 0.909]

Table 11: All task pairs and pairs with no recorded additional execution. Ratios divide search usage by witness-required review usage. Brackets give pointwise paired 95% bootstrap intervals from 20,000 task resamples. Opus 5 has no corresponding tool-use telemetry.

	Supported objections		Search / sequential usage	
Model	Search	Sequential	Output ratio [95% interval]	Full-token ratio [95% interval]
100 proof inputs per model				
Sol	95	93	.689 [.594,.802]	.958 [.833,1.083]
Gemini	84	89	.854 [.789,.921]	.650 [.441,.941]
Opus 5	81	76	.714 [.620,.814]	.831 [.762,.902]
60 statement inputs per model				
Sol	60	59	.593 [.457,.791]	.959 [.928,.982]
Gemini	58	57	.739 [.569,.889]	.903 [.755,1.063]
Opus	60	60	.476 [.240,.940]	.849 [.685,.983]

Table 12: Counterexample-first search versus sequential review or proof attempt, with a concrete witness required in both conditions. Objection counts are model-graded. A token ratio below one favors search. Ratios use summed usage of accepted attempts, not an average of per-task ratios; they exclude grading and later checking. Full tokens include reported input, cache, and output, plus reported auxiliaries in the statement panel. Opus 5 proof accounting is provisional.

Table 13: All four instruction arms. n is the number graded; strict counts a supported refutation, len. also counts a correct unsupported verdict, false counts invalid objections, and unv. counts objects the grader could not check. Mean is accepted-response output tokens. Ratio is counterexample-first output divided by the row's output on paired records, with a 95% record-bootstrap interval; obs. includes recorded rejected-attempt output. Early stop permits stopping at the first failure; early + witness also requires a concrete counterexample. Original Opus proof grades and transcript accounting are retained; all other rows use Astra grades and native usage. All comparisons have 100 proof pairs or 60 claim pairs, except original Opus early stop, with 99 usable token pairs and 100 graded assessments. Absolute costs are not compared across harnesses. Appendix O gives prompts and accounting.

Input, model	Arm	n	Strict	Len.	False	Unv.	Mean	Ratio [95% CI]	Obs.
proofs, Opus (original)	counterexample	100	83	83	0	0	6952	1 (ref.)	–
	full verify	100	84	84	0	0	9416	0.738 [0.61, 0.88]	–
	early stop	100	78	78	0	0	6313	1.107 [0.96, 1.27]	–
	early + witness	100	79	79	0	0	9737	0.714 [0.62, 0.82]	–
proofs, GPT-5.6 Sol	counterexample	100	95	95	3	0	1229	1 (ref.)	–
	full verify	100	97	97	1	0	3345	0.368 [0.32, 0.43]	0.366
	early stop	100	89	90	6	1	1161	1.059 [0.92, 1.23]	1.070
	early + witness	100	93	93	4	0	1784	0.689 [0.59, 0.80]	0.699
proofs, Gemini 3.1 Pro	counterexample	100	84	84	1	0	11014	1 (ref.)	–
	full verify	100	81	81	5	0	10693	1.030 [0.90, 1.16]	1.038
	early stop	100	83	85	3	0	10150	1.085 [0.95, 1.24]	1.108
	early + witness	100	89	89	0	0	12896	0.854 [0.79, 0.92]	0.872
claims, Opus	counterexample	60	60	60	0	0	1980	1 (ref.)	–
	full proof	60	60	60	0	0	2230	0.888 [0.78, 1.05]	0.888
	early stop	60	60	60	0	0	1461	1.355 [1.18, 1.59]	1.355
	early + witness	60	60	60	0	0	4161	0.476 [0.24, 0.94]	0.235
claims, GPT-5.6 Sol	counterexample	60	60	60	0	0	1555	1 (ref.)	–
	full proof	60	59	60	0	0	2857	0.544 [0.44, 0.67]	0.544
	early stop	60	42	58	1	1	1155	1.347 [1.04, 1.75]	1.347
	early + witness	60	59	59	0	0	2622	0.593 [0.46, 0.79]	0.593
claims, Gemini 3.1 Pro	counterexample	60	58	58	2	0	6407	1 (ref.)	–
	full proof	60	42	57	2	1	7464	0.858 [0.62, 1.14]	0.858
	early stop	60	43	57	0	2	5026	1.275 [0.93, 1.75]	1.276
	early + witness	60	57	57	1	1	8666	0.739 [0.57, 0.89]	0.740

Table 14: Full token components for all four proof100 instruction arms. Sums cover accepted attempts with known usage; n is usage coverage out of 100, not grade coverage. Uncached input includes cache creation in the Opus 5 records. Reasoning is included in output once. Additional attempts, graders and downstream checks are excluded. Opus 5 transcript sums can repeat usage and remain provisional; its missing early-stop record is unknown, not zero. Absolute costs are not comparable across harnesses.

Model	Instruction	n	Uncached input	Cache reads	Output	Full tokens
Sol	Search	100	665,020	2,256,000	122,923	3,043,943
Sol	Full review	100	735,037	2,548,992	334,459	3,618,488
Sol	Early stop	100	731,007	2,224,384	116,127	3,071,518
Sol	Early + witness	100	676,289	2,321,408	178,429	3,176,126
Gemini	Search	100	1,240,069	2,242,187	1,101,377	4,583,633
Gemini	Full review	100	1,730,843	4,196,566	1,069,289	6,996,698
Gemini	Early stop	100	1,406,962	3,818,526	1,014,987	6,240,475
Gemini	Early + witness	100	1,482,239	4,285,019	1,289,643	7,056,901
Opus 5	Search	100	2,995,056	13,577,136	695,247	17,267,439
Opus 5	Full review	100	3,293,960	16,204,707	941,599	20,440,266
Opus 5	Early stop	99	2,595,752	14,552,221	625,018	17,772,991
Opus 5	Early + witness	100	3,259,632	16,534,085	973,717	20,767,434

Table 15: Counterexample search / witness-required sequential review on the same 100 proof inputs. Catches are supported objections according to the reference-assisted model grader. Differences are search minus review in percentage points. Ratios divide accepted solver-attempt token sums for search by those for review. Brackets give pointwise paired 95% bootstrap intervals from 20,000 task resamples.

Model	Catches	Δ catches (pp)	Output ratio	Full-token ratio
GPT-5.6 Sol	95/93	+2 [-2, +6]	0.689 [0.594, 0.802]	0.958 [0.833, 1.083]
Gemini 3.1 Pro	84/89	-5 [-13, +3]	0.854 [0.789, 0.921]	0.650 [0.441, 0.941]
Opus 5	81/76	+5 [-2, +12]	0.714 [0.620, 0.814]	0.831 [0.762, 0.902]

Opus 5 uses provisional transcript-based token accounting; grade counts use the common grading pass. Appendix F describes this accounting and the separately recorded additional attempts, grading, and checking.

Record / topic	What the formal result establishes	Assumptions and scope
CS 97637/a97647 Square root	Failure iff the input is a positive square; strict-guard repair satisfies the specification for every natural input.	Completed before the other twelve packages; proves the corrected algorithm.
CS 1200/a1206 Array assignment	Every $2 :: 3 :: r$ reaches $2 :: 2 :: r$; repaired postcondition equals the reachable-state set.	A local program statement, with total correctness under the stated assignment rules.
CS 13118/a13122 Geometric sum	Correct sum is $(c^n - c)/(c - 1)$; exact offset and validity conditions are proved.	$n \geq 1$, $c \neq 1$ over a field; ordering is a separate real statement.
CS 156543 Liveness	Every finite prefix has a satisfying extension (applying the standard liveness framework (Alpern & Schneider, 1985)); a later commit exactly characterizes repair at a violated request.	The equivalence keeps the violated request in the fixed prefix.
CS 1957/a2388 Recurrence sum	Correct sum uses $\sum_{i=1}^{L} i^k$; the asserted equality holds iff $k = 0$ or $L \leq 1$.	Exact powers $n = b^L$, $b > 1$, $a > 0$, $k, L \in \mathbb{N}$; explicit Θ bounds.
TCS 2312/a2318 Signed cut	Total is $T + a - 2(n - 2)d$; exact conditions distinguish accounting error, negativity and loss of positivity.	Distinct vertices and additive edges; positivity has a separate condition.
Math 1291495 Cycle deletion	For a finite connected graph, terminal deletion outcomes are exactly its spanning trees; corrected edge counts recover the target tree theorem.	Delete one cycle edge at a time; the cyclic source premise requires $k \geq 2$.
Math 1868681 Bipartite edges	Maximum is $\lfloor p^2/4 \rfloor$; the rational value $p^2/4$ is attained iff p is even.	Distinguishes the exact maximum from the valid upper bound $p^2/4$.
Math 782216 Handshaking induction	The attempted step holds iff the graph has an isolated K_2 component; it fails for every connected graph with at least two edges. The zero-edge base case is also characterized.	Classifies the proposed induction step; the base case depends on the empty-graph convention.
Math 994847 Adjacency powers	$(I + A)^{n-1}_{ij} \neq 0$ iff i, j are reachable; a bipartite obstruction family has certified slack.	The full family includes diagonal entries; the source's off-diagonal assertion requires $n \geq 3$.
ProofWiki 25056 Polynomial content	$\mathrm{cont}(ag) = \lvert a \rvert \, \mathrm{cont}(g)$, with exact signed-form validity conditions.	Integer form; rational positive content needs a zero-polynomial convention.
Reddit 1bq1jtc Prime residues	$6q + 2 = 2(3q + 1)$, uniquely in the affine template; the intended prime-residue conclusion is also proved.	Corrects the local factorization and proves the target classification.
Reddit 8yjpp7 Recursive increment	Repaired expression equals $n + 1$ iff n is odd; the recursive program returns $n + 1$ for all n.	The source branch supplies oddness; the result extends the recorded $4k + 1$ family to every odd input.

Table 16: All thirteen mathematical follow-ups in the selected explanation study. The middle column states the formal result, and the last column gives the assumptions needed to apply it. The artifact index maps each abbreviated source identifier to its full record ID and selected Lean declarations.

Topic / record	What Lean checks	How the calculation bears on the source
Probability Algorithms 3/sol4 (Gonzalez et al., 2026)	For every $n > 0$, at the source's $\delta = 1$, both Bernoulli measures are $P = \mathrm{Bernoulli}(1/2)$ and $D_{\mathrm{KL}}(P\|P) = 0$. The exponential expression is 1, whereas the geometric expression is $(16/27)^n < 1$.	The displayed equivalence of the two bounds fails at the given parameters.
Asymptotic analysis Nye, arXiv 2508.14831v1, Appendix B.2, Lemma 21 (Nye, 2025) (saved record ID 2508.14831v4)	The actual dyadic path potential satisfies $2\Phi(k) = (k+1)(k+2)$. Lean proves that it is not $O(\log_2(2^k))$, quantifying over arbitrary real constants and eventual thresholds.	The quadratic potential contradicts the local per-path logarithmic bound.
Computability ProofWiki 4209	A surjective coding of every finite unlimited register machine (URM) program (Cutland, 1980, Chapter 1, §2) and unbounded operational semantics support a diagonal relation with no total Boolean characteristic program.	The general construction requires a computability assumption; the selected equal-branch example itself defines a recursive function.
Complex analysis G3 reference 367	For $f(z) = z - 1/2$ and $R = 3/4$, Lean evaluates the actual zero-count integrals and circle average. The unit-radius integral is $\log 2$, while the intended radius-R boundary expression is $\log(3/2)$; the corrected upper-limit identity holds.	Identifies the mismatched radius and checks the corrected identity for this example.
Functional analysis arXiv 1510.03522v1 (Wang et al., 2015)	On the Haar-measured torus, $u = 1$ on $[0, 1/3)$ and $u = -1/2$ elsewhere belongs to L^2 and has mean zero. The drift $u - u^3$ belongs to L^2 but has mean $-1/4$.	The unprojected drift leaves the mean-zero subspace. Human source review treats the passage as a model definition rather than an asserted preservation theorem.
Number theory Math SE 3160594	For every natural prime p, integer q, and natural $m \geq 2$, $pq^m \not\equiv 6 \pmod{36}$. The proof covers unbounded exponents.	The residue 6 rules out the proposed representation for every exponent $m \geq 2$.
Differential geometry ProofWiki 99460	For the spiral $r(t) = 1 + t$, $\theta(t) = t$ at $t = 0$, the actual turning-angle derivative is $3/2$ and the arc-length derivative is $\sqrt{2}$. Curvature is $3/(2\sqrt{2})$, rather than the printed $3/\sqrt{2}$.	The values disagree at this curve and point, locating the error in the displayed expression.

Table 17: Mathematical examples from the earlier 284-dossier collection. Each row separates the formally checked calculation from its application to a source passage. The number-theory dossier is also in the current human-reviewed core; the other six are preserved as earlier mathematical examples, with their source-review status recorded separately.

Table 18: Three families of counterexamples verified by Lean. Every row holds for every integer $k \geq 1$. The last column compares the density of the constructed set with the conjectured upper bound.

d	M	Excluded residues F	N	$\lvert A_N\rvert$	Density
5	16	$\{0, 4, 12\}$	$80k$	$65k$	$13/16 > 4/5$
7	24	$\{0, 6, 18\}$	$168k$	$147k$	$7/8 > 6/7$
8	36	$\{0, 6, 12, 30\}$	$72k$	$64k$	$8/9 > 7/8$

Case	Scope of the selected Lean proof
Prime blowup	C_9 is prime and every induced subgraph is recolorable, but its K_2 blowup is not; the source conjecture is formally refuted.
Rainbow Schur	The displayed coloring of $[22m]$ has exactly $99m^2 + 3m$ ordered rainbow triples for every positive m; the proposed asymptotic optimum $2/5$ is false.
Supertoken graphs	Starting from adjacency, the distance formula is proved for all n, k. Resolving sets of size at most $n-2$ refute $\dim F_2(K_n) = n-1$ for every $n \geq 8$.
Binomial polynomials	The sequence $V_{n,m}(q) = \sum_{j=0}^{n} \binom{n+j}{j}^m q^j$ fails coefficientwise q-log-convexity for every integer $m \geq 4$.

Table 19: Mathematical extensions of recorded research counterexamples. Each row states the scope of its formal proof. The cube families are reported in Appendix L. These extensions have separate mathematical provenance and are reported separately from the experimental success counts.

Source collection	Dossiers	Inputs
QEDBench	25	20
Stack Exchange collection	18	11
math.SE collection	16	10
ProofWiki	28	26
arXiv	15	13
Reddit	42	20
Reference-proof additions	3	0
Total	147	100

Error mechanism	Dossiers	Inputs
Structural inference	38	28
Algebra or order	37	23
Omitted case	28	18
Counting	14	8
Quantifier scope	11	8
Definition or domain	9	6
Algorithm invariant	8	8
Analytic inference	2	1
Total	147	100

Table 20: Composition of the 147 retained dossiers and 100 primary benchmark inputs. Each panel counts every record once. Source collections follow the release groups, with the two Reddit groups combined. Error mechanisms use the saved primary annotations; six dossiers, including three primary inputs, retain a mechanism-annotation uncertainty flag.

Model	Supported S/O/W	Output S/W	Full S/W
Sol	60/42/59	0.593 [0.457, 0.791]	0.959 [0.928, 0.982]
Gemini	58/43/57	0.739 [0.569, 0.889]	0.903 [0.755, 1.063]
Opus	60/60/60	0.476 [0.240, 0.940]	0.849 [0.685, 0.983]

Table 21: Separate 60-statement panel. S=search; O=ordinary early-stop proof attempt; W=proof attempt requiring a witness. Supported counts use the same reference-assisted grader. Ratios compare accepted-attempt token sums. Full tokens include reported input, cache and output, including Opus auxiliaries. Brackets are pointwise paired 95% intervals; output uses 2,000 resamples and full tokens 20,000. Components and exclusions appear in Appendix O.2.

Model	Arm	Uncached	Cache create	Cache read	Output	Bound full	Aux. full	All reported
Sol	Search	324,706	0	1,256,960	93,309	1,674,975	—	1,674,975
Sol	Full proof	285,110	0	1,298,304	171,408	1,754,822	—	1,754,822
Sol	Early stop	260,757	0	1,320,320	69,296	1,650,373	—	1,650,373
Sol	Early + witness	268,611	0	1,320,448	157,348	1,746,407	—	1,746,407
Gemini	Search	633,458	0	1,462,403	384,411	2,480,272	—	2,480,272
Gemini	Full proof	593,457	0	1,364,219	447,820	2,405,496	—	2,405,496
Gemini	Early stop	554,795	0	1,193,536	301,559	2,049,890	—	2,049,890
Gemini	Early + witness	647,171	0	1,578,794	519,987	2,745,952	—	2,745,952
Opus	Search	240	428,035	204,540	118,805	751,620	72,633	824,253
Opus	Full proof	240	420,345	210,586	133,790	764,961	73,870	838,831
Opus	Early stop	240	425,712	204,506	87,663	718,121	71,891	790,012
Opus	Early + witness	242	428,458	217,021	249,673	895,394	75,508	970,902

Table 22: All four arms on 60 claims per model. The four component columns and target full total concern the named model. Auxiliary full tokens are additionally reported in Opus receipts; a dash means no separate auxiliary breakdown is available. All-reported totals use the native aggregate for Sol/Gemini and include the reported auxiliaries for Opus. Only accepted attempts appear here.

Model	Target output S/W [95% CI]	All-reported full S/W [95% CI]
Sol	0.593 [0.457, 0.791]	0.959 [0.928, 0.982]
Gemini	0.739 [0.569, 0.889]	0.903 [0.755, 1.063]
Opus	0.476 [0.240, 0.940]	0.849 [0.685, 0.983]

Table 23: Search/witness-required proof-attempt ratios on all 60 paired accepted attempts. Output intervals retain the original 2,000-resample analysis; full-token intervals are a new descriptive analysis using 20,000 paired resamples, seed 20260921. These are pointwise percentile intervals without multiplicity adjustment.

P CERTIFIED-CLAIM LOCALIZATION PROTOCOL

Current evaluation set. The human-approved resource contains 147 dossiers from 137 source units. Its main evaluation uses 100 texts, each containing at least ten nonblank source lines, grouped into 92 problem clusters. A supplementary evaluation adds 31 shorter texts, giving 131 scorable inputs in 123 clusters. The remaining resource members comprise 12 pilot records, one unscorable record, and three records with unresolved reference-line concerns. The three line-related exclusions concern which lines to score; their dossiers remain in the resource. Appendix D describes the human reviews and individual exclusions.

The current scores reuse saved model answers under the original scoring rules. Membership was determined from review evidence before these answers were rescored. The inclusion ledger is `reviews/verified_core/membership.json`; `benchmark/verified_core/README.md` explains how to reproduce both current evaluation sets.

Inputs and reference lines. A critic receives numbered source text and a request to identify up to three candidate lines, ranked by preference, with a brief explanation. The prompt says that the text contains an error. It withholds the dossier's selected passage, counterexample, and Lean certificate. For some arXiv records, the supplied text is an excerpt from the paper. The released input is therefore the text the critic was asked to assess.

The reference lines form a checked set of accepted error locations for each input. They include restatements and instantiations of the challenged claim. Selecting any line in that set earns credit; 57 of the 100 primary inputs have more than one accepted line. Model-assisted review checked and expanded these sets. Human review also checked the correspondence between the source claim and the selected lines. Appendix D describes these checks alongside the review of each counterexample and Lean translation.

Scoring and uncertainty. Hit@1 counts a match in the first candidate slot. Hit@3 counts a match in any of the original first three slots. An invalid, unknown, blank-line, or duplicate label leaves its slot empty; a later candidate never moves forward to fill it. Missing answers and attempts that break the execution rules or exhaust their budget count as misses. We report these all-input rates together with usable-answer counts. Conditional rates restrict the denominator to usable responses, which can differ across configurations.

Four deterministic baselines choose the longest line, the line with most words, the line with most mathematical symbols, or the line most similar to the document's TF–IDF word profile. The comparison baseline is the strongest of these rules on the evaluated population. Random line selection is reported separately as its exact expectation.

Confidence intervals use 10,000 bootstrap resamples of problem clusters, keeping related texts together within the assignment groups specified in the released scorer (Efron, 1979; Field & Welsh, 2007). The comparison with the strongest rule reselects that rule in each resample. These intervals describe variation across the evaluated texts. The saved parser, scoring code, grouping, and seed settings are in `benchmark/code/`.

Saved responses and reproduction. The release preserves selected responses, failure categories, and all physical attempts. Recorded tokens, lower bounds for incomplete logging, and budget charges are separate quantities. The protocol and settings below explain how an attempt is run; `benchmark/verified_core/README.md` reproduces the current scores from the saved responses.

P.1 COMPLETE CURRENT RESULTS

Table 25 reports all 13 evaluated configurations on the current 100-input primary set, including both saved Haiku views. It gives intervals for both scores and recorded token components. The two Haiku views share calls, so their costs are not added together. The complete 131-input results and failure counts are in `benchmark/verified_core/expected_scores.json`.

Critic or baseline	Hit@1 (%) [95% CI]	Hit@3 (%) [95% CI]	Recorded full tokens	Recorded output tokens
Haiku 4.5 (retained notice)	31.0 [22.7, 39.8]	40.0 [30.9, 49.0]	632,500	393,211
Sonnet 5	61.0 [52.0, 69.6]	76.0 [67.7, 83.7]	697,281	218,376
Opus 5.5	68.0 [59.8, 76.2]	85.0 [78.6, 91.0]	510,187	78,546
DeepSeek R1 Qwen	22.0 [14.4, 30.1]	28.0 [20.0, 36.4]	428,199	215,305
Gemma 3 27B	26.0 [18.3, 33.7]	44.0 [35.7, 52.1]	340,082	17,552
GLM 4.5 Air	10.0 [4.9, 16.0]	11.0 [5.8, 17.2]	1,658,280	1,357,252
GPT-OSS 120B	49.0 [39.6, 58.3]	60.0 [51.0, 69.0]	450,420	150,198
GPT-OSS 20B	43.0 [34.3, 52.0]	47.0 [37.8, 56.6]	708,462	408,240
Kimi Linear	14.0 [7.2, 21.4]	27.0 [19.0, 35.7]	241,035	46,849
Nemotron 3.5 Lightning	27.0 [19.0, 35.6]	27.0 [18.8, 35.4]	1,676,079	1,361,911
Nemotron 3 Super	48.0 [40.2, 56.1]	48.0 [40.2, 56.0]	740,935	426,767
Gemini 3.1 Pro	73.0 [65.1, 80.6]	85.0 [78.2, 91.2]	1,446,420	556,279
GPT-5.6 Luna	63.0 [54.6, 71.1]	74.0 [66.7, 80.8]	390,142	98,876
Haiku 4.5 (continuation)	56.0 [46.9, 65.1]	69.0 [61.2, 76.5]	1,053,443	658,433
TF–IDF centroid	18.0 [11.0, 25.3]	—	—	—
Random line	6.7 [5.7, 7.8]	—	—	—

Table 25: Current primary results on 100 inputs from 92 problem clusters. Hit@3 counts a reference match among the first three candidate slots; the one-line baselines have no top-three result. The token columns report recorded usage for the selected primary attempts and are lower bounds where logging is incomplete. The public score file identifies complete totals and missing usage for each configuration.

P.2 Evaluated Models and Runtime Setup

We evaluated 13 model configurations: five hosted models and eight locally run open-weight models. Haiku has outage-continuation and retained-notice reporting views of the same calls. Thus the release contains 14 reporting views for 13 configurations. Their shared calls and costs are counted once in campaign accounting.

Table 26: Local model configurations and exact checkpoint repositories.

Model Label	**Repository**
DeepSeek-R1-Distill-Qwen-32B	`deepseek-ai/DeepSeek-R1-Distill-Qwen-32B`
Gemma 3 27B-IT (greedy)	`unsloth/gemma-3-27b-it`
GLM-4.5-Air (FP8)	`zai-org/GLM-4.5-Air-FP8` (FP8 compressed-tensors)
GPT-OSS 120B (MXFP4, medium)	`openai/gpt-oss-120b`
GPT-OSS 20B (MXFP4, medium)	`openai/gpt-oss-20b`
Kimi-Linear 48B-A3B Instruct	`moonshotai/Kimi-Linear-48B-A3B-Instruct` (configuration dtype: BF16)
Nemotron 3.5 Lightning 30B-A3B (BF16)	`nvidia/NVIDIA-Nemotron-3.5-Lightning-30B-A3B-BF16` (BF16)
Nemotron 3 Super 120B-A12B (FP8)	`nvidia/NVIDIA-Nemotron-3-Super-120B-A12B-FP8` (ModelOpt FP8)

Each repository identifier in Table 26 links to its public model card. Hosted model names are preserved experiment identifiers.

The GPT-OSS configurations use MXFP4 weights, medium reasoning effort, and instructions in the native Harmony developer block. All eight local configurations requested one B200 GPU and used `vLLM` 0.29.0 (Kwon et al., 2023).

Inputs that exceed the recorded context limit receive a setup refusal and count as misses. The 5 hosted model configurations are `claude-sonnet-5`, `claude-opus-5-5`, `claude-haiku-4-5-20251001`, `gemini-3.1-pro`, and `gpt-5.6-luna`. Their evalua-

tion queries ran through their respective command-line interfaces (Claude Code CLI, agy CLI for Gemini, and Codex CLI for Luna).

Model	Context tokens	Generation setting	Configured execution interface
Haiku 4.5	n.r.	Medium effort	Claude CLI 2.1.275
Sonnet 5	n.r.	Medium effort	Claude CLI 2.1.275
Opus 5.5	n.r.	Medium effort	Claude CLI 2.1.280
DeepSeek R1 Qwen	32,768	$T = 0.6$, $p = 0.95$	Local vLLM
Gemma 3 27B	65,536	Greedy	Local vLLM
GLM 4.5 Air	65,536	$T = 0.6$, $p = 0.95$; thinking	Local vLLM
GPT-OSS 120B	131,072	$T = 0.6$, $p = 0.95$; medium	Local vLLM
GPT-OSS 20B	131,072	$T = 0.6$, $p = 0.95$; medium	Local vLLM
Kimi Linear	32,768	$T = 0.6$, $p = 0.95$	Local vLLM
Nemotron 3.5 Lightning	131,072	$T = 1.0$, $p = 0.95$; thinking	Local vLLM
Nemotron 3 Super	65,536	$T = 1.0$, $p = 0.95$; thinking	Local vLLM
Gemini 3.1 Pro	n.r.	Low effort	agy 1.2.8
GPT-5.6 Luna	n.r.	Medium effort	Codex CLI 0.155.1

Table 27: Saved execution settings for the 13 model configurations. Every attempt has a 16,000-token output allowance and a 600-second wall limit, with the interface-specific enforcement described below. T is temperature and p is top-p; local generation uses vLLM 0.29.0 and seed 20260921. "n.r." means the hosted CLI's numeric context-window limit was not recorded. Context limits, output-budget enforcement and recovery rules are specified below. Available version receipts agree across the initial, correction and recovery passes.

Prompts and context. All configurations receive the same numbered text and benchmark instructions. Local models use their native chat templates; GPT-OSS places the instructions in the Harmony developer block. Gemini places them at the start of the user turn. For local models, the prompt plus the 16,000-token output allowance must fit both the configured context and the 120,000-token attempt ceiling. Longer inputs receive a recorded setup refusal and a score of zero, without truncation or a model call. The hosted runs preserve the CLI settings and receipts; their numeric context-window limits are not specified by the saved configuration.

Output and time limits. Each attempt has a 600-second wall limit and a 120,000-token attributable-usage ceiling. The common 16,000-token output allowance is enforced differently by the execution interfaces. Local vLLM generation sets `max_tokens=16000`, including any generated reasoning tokens; reaching the length limit counts as budget exhaustion. Claude requests a 16,000-token client limit and checks the event stream for continuations, output usage and live estimates. Gemini and Luna have no provider-side output cap in these saved interfaces: their recorded native output usage is checked after generation, and a completed response above the allowance counts as budget exhaustion. Gemini's output count includes its recorded thinking tokens. The 120,000-token ceiling is checked during the Claude and Gemini streams and after receipt of Luna's native usage. These shared allowances therefore do not imply matched computation.

Tools, retries and recovery. Claude and Luna disable tools, while local generation has no tool executor. Gemini's CLI exposes tools; the runner stops and rejects an attempt if a tool step or additional turn appears. Ordinary runs and the input-correction pass allow one initial attempt and one retry for an eligible transport failure. Responses that exhaust their budget or break the execution rules are terminal under that allowance. The separately authorized Haiku continuation resumes listed no-answer session-limit notices and carries forward genuine answers and budget failures. The later environment-recovery pass grants one attempt per listed target, with no automatic retry; its result replaces the notice irrespective of success. Both Haiku reporting views use the same corrected and later recovered outputs. Every dispatch remains in campaign accounting.

Usage and budget charges. Recorded usage, accounting charges and execution limits are distinct quantities. Complete native usage contributes its measured count; partial native usage contributes its recorded lower bound. An attempt with incomplete usage receives a conservative accounting charge

of at least 40,000 tokens, or a larger recorded lower bound or applicable reservation. A charge is a budget debit, not measured consumption. The current results table reports selected-response usage; `benchmark/accounting/README.md` provides the full accounting rules and attempt ledger. Exact generation drivers, prompts, and settings are documented in `benchmark/code/generation/README.md`.

P.3 Reference-Line Checks and Review Exclusions

Human review checked whether the accepted lines state the challenged claim. The initial 40-input check approved 35 line selections; after corrections, a fresh, disjoint sample approved 38 of 40 and met the review criterion. Appendix D gives the full review process and dispositions.

The current 100-input primary set and 131-input scorable set exclude every record with a retained negative or uncertain source, witness, or formalization review, and all three records with adverse judgments on their current line selections. Reapplying these exclusion lists leaves the current sets and scores unchanged. Appendix D records the individual decisions; `reviews/version_resolution/README.md` supplies the complete exclusion lists and saved-answer sensitivity. Reference-line revisions and their replay are documented in `benchmark/README.md`.

P.4 Examining First-Choice Misses

A random sample of 30 was drawn from Sonnet 5's 58 readable first-choice misses in the earlier 191-input evaluation, using seed 20260921. LLM reviewers, with access to the source text and reference labels, classified the selected claims and assessed the accompanying explanations.

The final review identified 12 selections of another false assertion, 15 lines that were not false as written, and three source or annotation problems. The first LLM reviewer approved the supplied explanation in 11 of the 12 alternative-assertion cases. The three source or annotation problems were two omitted restatements of the challenged proposition and one garbled source.

The released records preserve the original classifications, later adjudications, sampling intervals, and disagreements on M08, M13, and M20. This analysis retained the original benchmark scores. The sampled responses and scientific reviews are in `reviews/sonnet_misses/`; `reviews/README.md` explains their numerical reproduction.

P.5 Repair Study Outcomes

This study asks whether a checked counterexample helps a solver complete a specified mathematical correction. It uses 16 tasks derived from source arguments. Both conditions supply the source context, definitions, identified defect, repair template, and required proof statements. Condition A uses this information alone; condition B adds the checked counterexample.

The task's fixed checker accepts a submission when its edits are allowed and its proofs establish the requested correction for all permitted inputs, show that the hypotheses admit an example, and establish any task-specific endpoint properties. Each of three solvers—Opus, Sonnet, and GPT-OSS 120B—received up to three attempts per task and condition. Table 28 reports first-attempt acceptance. "Absent" means no submission was saved. Runtime-compliant acceptance has the same counts: a later runtime stop can leave an already saved, accepted submission intact.

Table 28: First-attempt checker-accepted submissions on 16 source-derived repair tasks.

Solver	Arm	PASS	FAIL	Absent
Claude Opus 5.5	A	15	0	1
Claude Opus 5.5	B	15	0	1
Claude Sonnet 5	A	13	3	0
Claude Sonnet 5	B	15	1	0
GPT-OSS 120B	A	1	10	5
GPT-OSS 120B	B	2	8	6

Across up to three attempts, Opus obtained an accepted submission on all 16 tasks in both conditions. Sonnet completed 14/16 without the witness and 16/16 with it; GPT-OSS 120B completed 2/16 and 3/16, respectively. Sonnet's first attempts passed in both conditions on 13 tasks, in neither condition on one, and only with the witness on two. These paired outcomes provide a concrete result for completing supplied repairs; broader witness effects can be tested on additional tasks.

Table 29 reports token usage across all 16 selected first attempts for each solver and condition, including failed attempts. Full tokens include output and all reported input components, including cache creation and reads; the released artifact provides the component breakdown. The symbol $\geq$ marks a lower bound when provider usage is unavailable. Earlier attempts made to recover from execution problems and evaluation costs are recorded separately. The tasks, checker obligations, saved submissions, and replay instructions are documented in `experiments/specified_repairs/README.md`.

Table 29: Token usage for selected first attempts (16 tasks per condition).

Configuration	Condition	Full tokens	Output tokens	Native usage coverage
Opus 5.5	A	$\geq$ 598,672	$\geq$ 42,733	15/16
Opus 5.5	B	$\geq$ 746,580	$\geq$ 46,799	15/16
Sonnet 5	A	$\geq$ 9,083,964	$\geq$ 247,815	13/16
Sonnet 5	B	8,819,081	221,411	16/16
GPT-OSS 120B	A	2,086,312	225,343	16/16
GPT-OSS 120B	B	1,720,809	227,225	16/16

Q What the Criticism Scores Measure

This appendix connects three questions: whether a formal counterexample applies to its source, whether a critic selects a false assertion, and whether its explanation is valid. The propositions make these connections precise. They are elementary consequences of the definitions, with written proofs developed with AI assistance. The paired-comparison analysis also gives a numerical example using the saved reviews of 24 texts. These analytical statements are separate from the Lean-certified mathematical results elsewhere in the paper.

Q.1 The logical connection from a formal witness to a source claim

A formal counterexample challenges a source only if it satisfies the source assumptions and contradicts the source conclusion. Let s identify the source passage and its surrounding argument. Let X_s be the set of objects discussed there, $A_s(x)$ mean that x satisfies the relevant assumptions, and $C_s(x)$ be the claimed conclusion. The selected source assertion is

$$Q_s : \quad \forall x \in X_s, \quad A_s(x) \Rightarrow C_s(x). \tag{3}$$

Thus Q_s says that every object satisfying the assumptions also satisfies the conclusion. The assumptions for the relevant case belong in A_s. This includes temporary suppositions when the selected assertion is made within an argument that assumes them.

The translation into Lean uses a formal domain $\widehat{X}$ and formal predicates $\widehat{A}, \widehat{C}$. A hat marks the encoded version, not a statistical estimate. A witness $w \in \widehat{X}$ has a certificate π proving

$$\widehat{A}(w) \ \wedge \ \neg \widehat{C}(w). \tag{4}$$

The conjunction says that the encoded assumptions hold and the encoded conclusion does not. Let $e : \widehat{X} \to X_s$ map a formal object to its source interpretation; $e(w)$ is the same witness interpreted at the source level.

Proposition 4 (One-sided refutation transport)**.** *If Equation* (4) *holds and, at the witness,*

$$\widehat{A}(w) \Rightarrow A_s(e(w)), \qquad C_s(e(w)) \Rightarrow \widehat{C}(w), \tag{5}$$

then $A_s(e(w)) \wedge \neg C_s(e(w))$, so the source assertion Q_s is false.

Proof. The first implication gives the source assumptions. If the source conclusion were true, the second implication would make the encoded conclusion true, contradicting the certificate. Hence the source conclusion is false at an allowed source object. □

The first implication checks that the witness is an allowed source example. The second ensures that a true source conclusion would make the formal conclusion true as well. Together, they transfer the formal contradiction to the source. Full equivalence between the formal and source statements is one way to satisfy these conditions. The sorting example in Appendix D.6 illustrates the first condition: the witness must satisfy the source's endpoint inequality. Specifying X_s, A_s, C_s, and e makes the required interpretation explicit for review.

Formal notation for the dossier fields. The record described in Section 2 can be written

$$\mathcal{D} = (S, L, \Gamma; \widehat{A}, \widehat{C}, w, d, \pi; E, J, \kappa).$$

Here S is the saved source packet and version, L the challenged passage, and Γ the assumptions and local context. The formal predicates and witness are as above; d is the selected named Lean declaration and π its proof. E stores file and dependency identities with build and axiom-check logs. J is the reasoned source-correspondence judgment, and κ the supported consequence for the surrounding argument. These fields make both the formal calculation and its source interpretation inspectable. The judgment J records any unresolved interpretation, while κ states whether the objection concerns one proof step, a construction, or the final claim.

Q.2 Partial labels and generated-objection soundness

A reference-selection score measures agreement with the designated target. Finding any false assertion and supplying a valid counterexample are two broader outcomes. Since the reference labels cover selected assertions and the score itself does not assess explanations, the available data can bound these outcomes without determining them exactly. This is a partial identification problem (Manski, 2003).

In this subsection N is the number of benchmark inputs, and all sums over i run from one to N. For each item i, let F_i be the set of admissible line labels locating genuinely false assertions under the source assumptions; a label may identify part of a displayed proposition rather than a complete proposition by itself. This set describes the truth of the assertions; it need not be fully annotated. Let G_i be the designated gold set and V_i the protocol-valid indicator. For a first-choice label ℓ_i, define

$$h = \frac{1}{N}\sum_i V_i \mathbf{1}\{\ell_i \in G_i\}, \quad q = \frac{1}{N}\sum_i V_i \mathbf{1}\{\ell_i \in F_i\}, \quad c = \frac{1}{N}\sum_i V_i.$$

Here h is the fraction selecting the stored reference, q is the fraction selecting any genuinely false assertion, and c is the fraction receiving a protocol-valid response. Unlike h and c, q need not be observable because the full sets F_i are not annotated. Null labels belong to neither set.

Proposition 5 (Partial identification under sound references)**.** *If $G_i \subseteq F_i$ for every item, then $h \le q \le c$. If the inclusion may fail on at most m items, then*

$$\max(0, h - m/N) \le q \le c.$$

Let E_i indicate that the critic's supplied objection is valid for its selected target, and define $r = N^{-1}\sum_i V_i \mathbf{1}\{\ell_i \in F_i\} E_i$. Without evidence about E_i, no strictly positive lower bound on r follows from h alone.

Proof. On a sound item, gold membership implies false-assertion membership. At most m gold hits can fail that implication on the remaining items. No valid target can be credited without a valid output, giving the upper bound c. Finally, every E_i can be zero even when every chosen target matches its gold set: a critic may attach an invalid witness to the correct line. Hence $r = 0$ is compatible with positive or perfect reference localization. □

The same argument applies to top-k target selection, with explanation validity assessed separately. When the designated references are sound, matching one supplies evidence that the critic selected a false assertion. A reference miss can also identify a different false assertion. The diagnostic in Appendix P.4 examines such cases; estimating q or r additionally requires the corresponding target and explanation judgments.

Q.3 Comparing Critics When Some Selected Lines Have Not Been Reviewed

If two critics select the same line, its truth status affects both scores equally. Only selections on which they disagree can change their score difference. The following result turns that observation into exact bounds.

Normalize a protocol-invalid or invalid first-choice prediction to $\perp$, so that critic a or b supplies $p_i^a, p_i^b \in L_i \cup \{\perp\}$, where L_i is the addressable line set. Write $z_{i\ell} = \mathbf{1}\{\ell \in F_i\}$ and $z_{i\perp} = 0$. A line reviewed as false under the source assumptions has value one; a line reviewed as not false has value zero. Every other indicator is unknown. A reference label has value one only when its validity under the source assumptions is assumed or established.

For critic $m \in \{a, b\}$, let $\underline{z}_i^m, \overline{z}_i^m$ be the lower and upper possible values of z_{i,p_i^m}. They are $(1, 1)$ for known false assertions, $(0, 0)$ for known nonfalse or null predictions, and $(0, 1)$ for unadjudicated lines. Indicators of the same line on the same item are shared, not assigned separately to each critic.

Proposition 6 (Sharp comparison in the line-label model)**.** *Over assignments of unknown line indicators consistent with these known labels and with no further constraints, the false-assertion localization difference*

$$\Delta_F = \frac{1}{N}\sum_i \big(z_{i,p_i^a} - z_{i,p_i^b}\big)$$

has sharp lower and upper bounds

$$L = \frac{1}{N} \sum_{i:p_i^a \neq p_i^b} (\underline{z}_i^a - \overline{z}_i^b), \qquad U = \frac{1}{N} \sum_{i:p_i^a \neq p_i^b} (\overline{z}_i^a - \underline{z}_i^b). \tag{6}$$

Both endpoints are attained in this model. "Sharp" means that no narrower interval follows from these labels alone under the stated absence of further constraints.

Proof. Coincident predictions share an indicator, so their contribution is exactly zero even when it is unknown. For distinct predictions the difference is at least $\underline{z}_i^a - \overline{z}_i^b$ and at most $\overline{z}_i^a - \underline{z}_i^b$. To attain the lower bound, set each unknown indicator selected by a to zero and each selected by b to one on disagreement items; reverse these assignments for the upper bound. Distinct selected lines impose no conflicting assignment within an item, and this model imposes no cross-item constraints. Thus the extremal choices can be made simultaneously. □

Which additional reviews narrow the comparison. The identification width is

$$U - L = \frac{1}{N} \sum_{i:p_i^a \neq p_i^b} \left[(\overline{z}_i^a - \underline{z}_i^a) + (\overline{z}_i^b - \underline{z}_i^b) \right].$$

Only unknown predictions on disagreement items contribute: one unknown line gives width $1/N$, two give $2/N$, and a shared unknown prediction gives zero. Reviewing one such line narrows the interval whichever label it receives. Thus disagreements identify the additional source judgments most directly relevant to the paired comparison.

The bounds are exact for the stated model, which permits every assignment consistent with the known line labels. Logical relationships among assertions or repeated sources can narrow them further. An identification interval describes the possible score differences while some labels are unknown. A sampling confidence interval instead describes uncertainty from sampling texts. Reviewing explanation validity answers another question: whether the critic supplies valid evidence for its chosen assertion. Numerical application requires item-level predictions and judgments.

Experiment B detailed judgments. Table 30 reproduces the first-reader evidence-validity judgments (J3) chosen in advance, originally reported in the main text.

Model	First-choice match	Valid	Invalid	Unresolved
Gemini 3.1 Pro	Hit@1	18	0	2
Gemini 3.1 Pro	Miss	3	1	0
Nemotron 3 Super 120B	Hit@1	16	2	0
Nemotron 3 Super 120B	Miss	5	0	1

Table 30: Evidence-validity judgments (J3) for 48 saved responses, using the LLM review chosen in advance for each response.

A third LLM verifier reviewed eight flagged responses. P06 contained a complete explanation, although the attempt exceeded the allowed budget. P43 supplied valid mathematical evidence but cited the wrong line. P35 challenged a stronger condition than the selected statement claimed, so its counterexample did not address that claim. For P31, the calculation refuted the implication stated in the model's explanation, while the supplied example violated two assumptions in the source. Applying the study's rule that counterexamples must satisfy the source assumptions classifies P31's evidence as invalid.

Table 31 tracks the joint positive-evidence endpoint. The rows successively use the first designated LLM reviewer, require agreement between both LLM reviewers, include the recorded adjudications, and apply the source assumptions.

Joint criterion	Gemini 3.1 Pro	Nemotron 3 Super 120B
First reader	18 / 24	19 / 24
Both readers	18 / 24	18 / 24
After LLM adjudication	20 / 24	20 / 24
Source-scope (final joint endpoint)	20 / 24	19 / 24

Table 31: Protocol-valid responses satisfying both target identification (J1) and evidence validity (J3).

Numerical application on 24 texts. The theorem concerns which selected assertions are false, not whether the critic's own explanation proves they are false. Using the saved judgments and source checks, we assign a shared truth indicator to each $(\text{input}, \text{line})$ pair. A valid objection can establish a positive line label; an invalid explanation alone does not establish a negative line label. Gemini and Nemotron make the same normalized prediction on 15 of 24 inputs; their contribution is exactly zero regardless of the line's status. They disagree on nine inputs. Let z_{19} and z_{43} be the unknown false-assertion indicators for Nemotron's P19 line 24 and P43 line 230. The seven other disagreement contributions sum to zero based on the accepted labels or null predictions. These two contribute $-z_{19}$ and $1 - z_{43}$. Thus the Gemini-minus-Nemotron score difference is $(1 - z_{19} - z_{43})/24$, between $-1/24$ and $1/24$. Positive line labels are accepted conditionally from the saved LLM judgments and all other non-null labels remain unknown; no automatic sound-gold assumption or negative line-label inference from invalid evidence is used. These are 24 paired texts from Experiment B, separate from the current 100-input benchmark. The interval $[-1/24, 1/24]$ is conditional on the recorded LLM judgments and summarizes the two unknown labels; it is not a 95% confidence interval. The saved inputs, judgments, and reproducible calculation are documented in `experiments/criticism_endpoints/README.md`.

Q.4 Sensitivity to Revised Reference Lines

Let k be the number of candidate slots scored. Fix P_i^k, the set of valid line labels in the original first k slots for input i; null slots remain unfilled and later predictions are not promoted. Write $G_i \triangle G_i'$ for labels that belong to exactly one of the old and new reference sets. Let G_i, G_i' be old and revised reference sets and $r = |\{i : G_i \neq G_i'\}|$.

Proposition 7 (Prediction-aware reference perturbation)**.** *For fixed outputs and a fixed N-item population,*

$$|\mathrm{Hit}_G\,@k - \mathrm{Hit}_{G'}\,@k| \le \rho(k) := \frac{1}{N}\sum_i \mathbf{1}\{P_i^k \cap (G_i \triangle G_i') \neq \varnothing\} \le \frac{r}{N}. \tag{7}$$

For two critics a, b, their paired score difference changes by at most $\rho_a(k) + \rho_b(k) \le 2r/N$.

Proof. An item cannot change hit status unless a selected label lies in the symmetric difference $G_i \triangle G_i'$. Even then, a retained gold hit may keep its status unchanged. Each binary hit changes by at most one, so summing gives the first inequality. Only items whose gold set changed can contribute to ρ, giving the second. Applying the triangle inequality to two critics yields the paired bound. The coarse constants are attainable: a changed reference can turn one miss into a hit, or exchange gold between two distinct first-choice predictions, changing their difference by two. □

The prediction-aware bound counts possible affected outcomes, not net changes. A gained hit and a lost hit can cancel in an aggregate score; conversely, an edit away from every selected line cannot affect the score. Thus counting changed reference sets alone can substantially overstate this particular perturbation's possible effect.

The bound applies to any recorded revision of the checked line sets. Rescoring the saved predictions against both versions gives the actual change. If m excluded inputs are misses for every reporting view, each rate changes from H/N to $H/(N-m)$ while the hit count H stays fixed. The release provides the line sets, exclusion lists, and per-input predictions needed for these calculations.

Q.5 Why a Paired Comparison Uses Item-Level Outcomes

Suppose two models have a and b hits on the same N items, with $a \geq b$. Their common-hit count t can lie anywhere in

$$\max(0, a + b - N) \leq t \leq b.$$

The discordant counts are $a - t$ and $b - t$; therefore a paired test requires more than the marginals. For example, the current 100-input results give 73 first-choice matches for Gemini and 68 for Opus. These totals permit t from 41 to 68, or between 5 and 59 inputs on which exactly one model succeeds. The released per-input predictions determine the actual overlap for a paired comparison. The main text therefore reports the highest observed scores, while paired comparisons use the underlying item-level outcomes.

Q.6 Costs with missing telemetry

Let total nonnegative costs in two conditions be $C_A = L_A + U_A$ and $C_B = L_B + U_B$. Here L_A, L_B are the recorded subtotals and $U_A, U_B \geq 0$ are unrecorded usage. The ratio L_A/L_B is generally neither a lower nor an upper bound on C_A/C_B, because either unrecorded remainder can be larger. A budget charge assigns a debit for accounting purposes; it does not measure that remainder. Likewise, a cost total for selected successful attempts describes a different set of executions from one that includes failures and retries. We therefore name the included calls and report measured usage, lower bounds, and budget charges separately.

R Proof Development with Retrieved Counterexamples

We evaluated whether retrieved counterexamples help a model develop a proof through repeated revision. Each model worked on the same 50 tasks in two conditions. The BugBank condition could retrieve up to three records for each queried claim from an earlier 661-entry database, using SQLite FTS5 `bm25` ranking (Robertson & Zaragoza, 2009) (`https://sqlite.org/fts5.html#the_bm25_function`). The Plain condition used iterative self-feedback (Madaan et al., 2023). Each row in Table 32 grades all 100 final drafts, 50 from each condition.

The shared-rule protocol uses the same revision cap and stopping rule in both conditions. The original-protocol rows retain the earlier stopping rules as a separate comparison. These tasks and the retrieval database belong to this proof-development study; they are separate from the current 147-dossier resource and its 100-input line-selection benchmark.

Table 32: Proof development outcomes on 50 tasks per model. Values are reported as BugBank / Plain. Correctness is assessed by model graders using the recorded rubric. Full tokens count reported input, cache, and output usage for the selected completed solver attempts, including auxiliary calls; output tokens also include auxiliary calls. Grading, rejected attempts, and later checks are accounted for separately. Full-token ratios use the BugBank condition divided by Plain, with 95% intervals from 2,000 paired bootstrap resamples of tasks.

Model	Protocol	Graded correct (/50)	All-model output tokens	Full tokens	Full ratio [95% CI]
GPT-5.6-Sol	shared	45 / 47	315,167 / 339,701	9,762,586 / 7,184,842	1.359 [1.211, 1.511]
Gemini 3.1 Pro	shared	43 / 42	491,041 / 552,991	10,354,373 / 7,949,686	1.302 [1.193, 1.428]
Opus 5	shared	44 / 43	746,839 / 1,023,808	6,768,539 / 9,721,951	0.696 [0.555, 0.887]
GPT-5.6-Sol	original	48 / 48	294,955 / 253,003	10,089,707 / 5,280,740	1.911 [1.634, 2.215]
Gemini 3.1 Pro	original	43 / 42	442,707 / 636,047	8,167,805 / 4,189,656	1.950 [1.773, 2.140]
Opus 5	original	44 / 46	902,816 / 1,094,155	8,160,358 / 7,651,046	1.067 [0.869, 1.349]

The shared-rule results show different patterns across models. Opus used fewer full tokens with retrieval: its ratio was 0.696, with interval [0.555, 0.887], and its graded-correct counts were 44 versus 43. Sol and Gemini used more full tokens, with graded-correct counts of 45 versus 47 and 43 versus 42. Under the original stopping rules, all three full-token ratios exceeded one. The comparison therefore describes each complete revision procedure, including its source of feedback and stopping rule; it does not isolate retrieval as the cause of a general correctness or efficiency advantage.

Additional costs and reproduction. Grading used 3,312,989 full tokens for Sol, 3,536,651 for Gemini, and 3,806,090 for Opus. Each model has 200 grading receipts and complete grading-usage records. Additional-attempt records contain two attempts with unknown usage for Sol, none for Gemini, and at least 349,775 full tokens for Opus; 25 of Opus's 26 additional attempts lack complete native usage. These costs are recorded outside the selected-attempt totals and are not allocated to individual conditions or protocols. The study inputs, prompts, grading records, retrieval snapshot, and deterministic replay are documented in `experiments/proof_development/README.md`.

R.1 Retrieved Counterexamples in Open-Problem Search

This paired study examined retrieval during longer mathematical searches. Each target was pursued with BugBank retrieval (B) and with self-critique and model auditing alone (P). The main comparison contains 36 distinct final targets. For 22 targets, continued searches replace the corresponding original endpoints, so each target is counted once in the final result.

A model audit accepted a full resolution of the stated target in 12 retrieval runs and 11 self-critique runs. Eleven targets were accepted in both conditions, one only with retrieval, and 24 in neither. "Audited-full" in Table 33 denotes this saved model-audit decision. The deciding audits for the 31 original-run targets could see which condition used retrieval. These judgments document the search procedure's own assessments; independently checking the resulting mathematical claims is a separate step.

The table also reports 58 observations from individual stages of these searches. Some stages concern the same target, so the 58 rows of evidence are not 58 independent final targets. Two additional panels are disjoint from the main comparison: six exploratory targets from fresh sources, with zero acceptances in either condition, and six targets chosen to test BugBank retrieval, with one acceptance in each condition.

Reading the usage totals. Component model tokens count completed, valid revision and audit calls through the recorded endpoint. They exclude the shared starting draft and audit, retrieval, invalid or failed attempts, and later work. Original and hard-target rows report cumulative endpoint usage; the continuation row reports only added usage. The Plain condition was required to use at least the BugBank token budget before stopping without a full resolution. This rule partly determines the reported usage differences, so the totals describe the execution protocol rather than a causal efficiency gain from retrieval. Because the stages overlap, we report their costs separately instead of adding them into a campaign total.

These searches are a separate study from the current line-selection benchmark. The target definitions, saved model audits, continuation rules, and replay instructions are in `experiments/open_problem_retrieval/README.md`.

Component	Paired units	Audited-full (B / P)	Both	B-only / P-only	Neither
Distinct targets, final endpoints	36	12 / 11	11	1 / 0	24
Protocol stages, descriptive	58	12 / 11	11	1 / 0	46
Original run	31	9 / 8	8	1 / 0	22
Hard-target run	5	0 / 0	0	0 / 0	5
Fixed-point continuation	22	3 / 3	3	0 / 0	19
Exploratory cohort	6	0 / 0	0	0 / 0	6
Targeted stress test	6	1 / 1	1	0 / 0	5

Table 33: Paired-search components comparing BugBank retrieval (B) against a self-critique/audit condition (P). Component model token usage (B / P): Original $16{,}449{,}383/17{,}590{,}589$; Hard $1{,}683{,}108/2{,}050{,}662$; Continuation (incremental) $19{,}577{,}279/20{,}301{,}967$; Exploratory $1{,}370{,}760/1{,}710{,}782$; Targeted $1{,}372{,}591/1{,}436{,}842$.

S PROOFS OF THE STRUCTURAL COUNTEREXAMPLE PROPOSITIONS

This appendix proves the two propositions in Appendix K.1. The Schur proof turns a coloring rule into a density formula, then optimizes that formula. The cycle proof turns the inability to recolor a vertex into a periodicity condition, then counts all colorings satisfying it. These are mathematical follow-ups to the research counterexamples, separate from the benchmark inputs.

The complete propositions also have Lean proofs. The Schur certificate links finite-coloring counts to their limiting density and proves its unique optimum and sharp stability bound. The cycle certificate proves the characterization and count for every allowed cycle length. Both are available under `mathematics/sources/`, with statement and build records in `mathematics/index.json`. The universal claims follow from these proofs; finite checks of the original witnesses are supplied separately under `mathematics/discoveries/`.

S.1 SCHUR TEMPLATE: COUNTING, OPTIMIZATION, AND STABILITY

Proof of Proposition 1. Count the two kinds of rainbow triple. Fix $0 \le a \le b \le 1$ and the coloring specified in the proposition. Write $t = b-a$ for the width of the middle odd block and $x_+ = \max(x, 0)$ for the positive part of x. The two odd color classes have cardinalities $(1-t)n/2 + O(1)$ and $tn/2 + O(1)$. There are $\binom{n}{2}$ ordered triples $x + y = z$ in $[n]$.

For each pair of differently colored odd integers, their positive even difference yields exactly two mixed-parity rainbow triples, one for each ordering of the addends. Conversely each mixed-parity rainbow triple determines this pair. Their number is therefore $t(1-t)n^2/2 + O(n)$, contributing $t(1-t)$ to the limiting normalized density.

For odd–odd addends, the sum is even and has color 1. Choose the color-2 addend from $[0, a] \cup [b, 1]$ and the color-3 addend from $[a, b]$ in normalized coordinates, subject to their sum being at most 1. The lattice of odd pairs has density $n^2/4$ per unit area. The factor two for ordering the addends and division by $\binom{n}{2} \sim n^2/2$ cancel this factor $1/4$. Thus the limiting contribution is the area of those two rectangles below the line $x + y = 1$. Boundary lattice squares number $O(n)$, giving an $O(n)$ error in the unnormalized count; this also justifies arbitrary real thresholds and their rounding. Even–even addends cannot be rainbow.

For a rectangle $[u, v] \times [s, t]$, its area below $x + y = 1$ is

$$\frac{1}{2}\big[(1-u-s)_+^2 - (1-v-s)_+^2 - (1-u-t)_+^2 + (1-v-t)_+^2\big].$$

This follows by inclusion–exclusion from the triangle with lower corner (u, s). Applying the formula to the two rectangles gives

$$\begin{aligned} D(a,b) = (b-a)(1-b+a) \\ + \frac{1}{2}\big[(1-a)^2 - (1-b)^2 - (1-2a)_+^2 - (1-2b)_+^2 + 2(1-a-b)_+^2\big]. \end{aligned} \tag{8}$$

In particular the limit exists.

Optimize the density over all threshold pairs. The allowed pairs form the triangle $0 \le a \le b \le 1$. The lines $a = 1/2$, $b = 1/2$, and $a + b = 1$ divide it into four regions. Expanding Equation (8) gives the following quadratics and Hessian determinants:

Region	$D(a,b)$	$\det \nabla^2 D$
$0 \le a \le b \le \frac{1}{2}$	$-\frac{3}{2}a^2 + 4ab - \frac{5}{2}b^2 - 2a + 2b$	-1
$a \le \frac{1}{2} \le b$, $a + b \le 1$	$-\frac{3}{2}a^2 + 4ab - \frac{1}{2}b^2 - 2a + \frac{1}{2}$	-13
$a \le \frac{1}{2} \le b$, $a + b \ge 1$	$-\frac{5}{2}a^2 + 2ab - \frac{3}{2}b^2 + 2b - \frac{1}{2}$	11
$\frac{1}{2} \le a \le b \le 1$	$-\frac{1}{2}a^2 + 2ab - \frac{3}{2}b^2 - 2a + 2b$	-1

The first, second, and fourth Hessians are indefinite. At an interior local maximum a twice differentiable function must have negative semidefinite Hessian, so these three regions attain their maxima on their boundaries. In the third region the Hessian is $\left(\begin{smallmatrix} -5 & 2 \\ 2 & -3 \end{smallmatrix}\right)$, which is negative definite. Its gradient vanishes uniquely at $(a, b) = (4/11, 10/11)$, strictly inside the region, with value $9/22$.

The following formulas cover every remaining boundary:

$$
\begin{aligned}
D(a,a) &= 0, \\
D(0,b) &= \begin{cases} 2b - \frac{5}{2}b^2, & b \le \frac{1}{2}, \\ (1-b^2)/2, & b \ge \frac{1}{2}, \end{cases} \\
D(a,1) &= \begin{cases} 2a - \frac{5}{2}a^2, & a \le \frac{1}{2}, \\ (1-a^2)/2, & a \ge \frac{1}{2}, \end{cases} \\
D(a,\tfrac{1}{2}) &= \tfrac{3}{8} - \tfrac{3}{2}a^2 && (0 \le a \le \tfrac{1}{2}), \\
D(\tfrac{1}{2},b) &= \tfrac{3}{8} - \tfrac{3}{2}(b-1)^2 && (\tfrac{1}{2} \le b \le 1), \\
D(a,1-a) &= 3a - 6a^2 && (0 \le a \le \tfrac{1}{2}).
\end{aligned}
$$

The second and third lines have maximum $2/5$, attained at $b = 2/5$ and $a = 2/5$, respectively. Each of the final three has maximum $3/8$. Thus every boundary value is at most $2/5 < 9/22$, proving the unique global maximum over this template. The boundaries $a = 0$ and $b = 1$ are precisely the two contiguous odd-block subfamilies, proving their claimed optima.

Measure the loss from moving the thresholds. Completing the square in the third region, with $u = a - 4/11$ and $v = b - 10/11$, yields Equation (2). The loss matrix is $\left(\begin{smallmatrix} 5/2 & -1 \\ -1 & 3/2 \end{smallmatrix}\right)$ with eigenvalues $(4 \pm \sqrt{5})/2$, so in that region

$$
\frac{9}{22} - D(a,b) \ge \frac{4-\sqrt{5}}{2}\left((a - \tfrac{4}{11})^2 + (b - \tfrac{10}{11})^2\right).
$$

This inequality quantifies stability within the template: the loss in density is at least a fixed positive multiple of the squared distance from the optimal threshold pair. The wider problem allows arbitrary three-colorings and requires a separate upper bound. □

The public file `mathematics/sources/SchurTemplateComposed.lean` contains the counting, limit, and optimization arguments. It checks Equation (8), including every boundary, using nonnegative polynomial decompositions in the four regions. For the sharp coefficient, writing $s = (1+\sqrt{5})/2$, $\lambda = (4-\sqrt{5})/2$, and $Q(u,v) = 5u^2/2 - 2uv + 3v^2/2$, it verifies $s[Q(u,v) - \lambda(u^2+v^2)] = (su - v)^2$. Equality therefore holds exactly on $v = su$. Any nonzero vector on that line proves that the coefficient cannot be increased for this quadratic form.

The corresponding declarations are `CoPEER.SchurTemplate.unique_global_optimum`, `sharp_stability`, `quadratic_spectral_equality_iff`, and `quadratic_spectral_coefficient_optimal`. The declaration `paper_limit` proves the counting and limit correspondence preceding Equation (8); the composed theorem joins it to the optimization result. `mathematics/recorded_builds/schur.json` records the compiled source and log hashes.

S.2 Odd-cycle blowups: palette periodicity and component counts

Proof of Proposition 2. Verify the conjecture's hypotheses for the base cycle. A graph is d-degenerate if every nonempty induced subgraph has a vertex of degree at most d. The standard degeneracy recoloring argument (Cereceda et al., 2008) shows that if a finite graph X is d-degenerate, then $R_k(X)$ is connected for every $k \ge d+2$. Here vertices of $R_k(X)$ are proper maps $V(X) \to [k]$, without requiring surjectivity, and an edge changes one vertex's color. To prove the lemma, induct on $|V(X)|$. Remove a vertex v of degree at most d, and connect the restrictions of any two colorings by induction. Lift that path to X. If a neighbor is about to acquire the current color of v, first move v to a different color absent from its neighborhood; at least two colors are absent because $k \ge d+2$. Then perform the neighbor's move. Finally give v its target color. This proves connectivity for all required k.

For $n \ge 5$, C_n is prime. To see this, a *module* is a set of vertices that all have the same neighbors outside the set. A proper module S with at least two vertices would, by connectivity, have an outside vertex v adjacent to a member of S. Modularity makes v adjacent to every member of S, so degree two forces $S = N(v)$. The other neighbor of either member of S lies outside S and distinguishes its

two members, a contradiction. For odd n, C_n has chromatic number three and degeneracy two, so the lemma proves recolorability for every $k \geq 4$. Each proper induced subgraph is a disjoint union of paths. With an edge it has chromatic number two and degeneracy one, so the lemma applies for every $k \geq 3$. Nonempty edgeless graphs recolor one vertex at a time for every $k \geq 2$; the empty graph has one coloring. Thus all induced subgraphs satisfy the full definition of recolorability.

Determine the chromatic number of the blowup. Call each replacement clique a *bag*, and index the bags of $H_n = C_n[K_2]$ cyclically by $0, \ldots, n-1$. An independent set takes at most one vertex per bag, and its occupied bags form an independent set of C_n. Conversely every such set of bags lifts to an independent set. Therefore $\alpha(H_n) = (n-1)/2$, and for odd $n \geq 5$,

$$\chi(H_n) \geq \left\lceil \frac{2n}{(n-1)/2} \right\rceil = \left\lceil \frac{4n}{n-1} \right\rceil = 5.$$

For the matching upper bound, give the first $n-3$ bags alternating palettes $\{1,2\}, \{3,4\}$. The prefix length is even, so its last palette is $\{3,4\}$. Give the last three bags, in order, $\{1,5\}, \{2,3\}, \{4,5\}$. Adjacent palettes, including the last and first, are disjoint. Assign each palette's two colors once in its bag. This is a proper five-coloring for every specified n.

Identify exactly when a coloring is frozen. In any proper six-coloring, write P_i for the two-element palette on bag i. Every vertex of H_n has degree five, and its closed neighborhood consists exactly of the six vertices in bags $i-1, i, i+1$. It cannot change color precisely when its five neighbors use all five alternative colors. The coloring is therefore frozen if and only if, for every i, the palettes P_{i-1}, P_i, P_{i+1} are pairwise disjoint and partition $[6]$. Comparing this partition with that for $i+1$ forces $P_{i-1} = P_{i+2}$: both are the complement of $P_i \cup P_{i+1}$. Hence the palettes have period three. If $3 \nmid n$, then $\gcd(3,n) = 1$, so this periodicity around the cycle makes all palettes equal, contradicting adjacent disjointness. If $3 \mid n$, any ordered partition of $[6]$ into three pairs, repeated around the cycle, supplies a frozen coloring.

Count the isolated colorings. There are $6!/(2!)^3 = 90$ ordered partitions for bags $0, 1, 2$. Each determines all other palettes, and in each labeled bag we can assign the two colors to its two labeled vertices in two ways, independently. The exact frozen-coloring count is thus $90 \cdot 2^n$. Each is an isolated vertex of $R_6(H_n)$. The explicit five-coloring, regarded as a six-coloring, has unused color 6, so any vertex can move to that color. It lies in a further component, giving at least $90 \cdot 2^n + 1$ components. For odd n divisible by three, write $n = 6r + 3$ with $r \geq 1$. These graphs refute the same prime-blowup preservation claim at $6 = \chi(H_n) + 1$. □

The exact count distinguishes vertex labels and color labels, just as R_k does. It gives a lower bound on the number of connected components: each frozen coloring contributes one, and the five-coloring contributes another. For $3 \nmid n$, deciding connectivity requires examining the components formed by colorings that can make single-vertex moves.

T CORPUS CONSTRUCTION AND AUDIT PROCEDURES

This appendix explains how proposed counterexamples become reviewable records through search, mathematical checking, formal proof, and source review. Appendices D and N document the completed human reviews and the current 147-dossier core, which supplies the 100 primary benchmark inputs.

T.1 GENERATING AND CHECKING COUNTEREXAMPLES

Let P be an informal proof, q a claim used in P, and w a proposed counterexample. Checking the objection requires three things: w must violate the stated mathematical condition, that condition must represent q under the source's assumptions, and the checking program must perform the intended calculation. The workflow checks these questions separately and saves the evidence from each stage.

What each check establishes. A separately written program tests the proposed witness; a Lean proof certifies the formal counterexample (de Moura & Ullrich, 2021; The mathlib Community, 2020). File hashes, source-passage matching, and axiom checks link that proof to its dossier. Human review checks whether the objection represents the source claim and satisfies its assumptions. An executable

Python check also depends on the Python interpreter. This division follows certifying algorithms and proof-carrying code: a producer supplies evidence that a smaller checker can verify (McConnell et al., 2011; Necula, 1997).

Selecting claims to investigate. Two separate calls to the same fixed model version receive the problem and a line-numbered proof. Each identifies a claim that may admit a short counterexample: at most 12 graph vertices, 10 objects, or 10 significant steps of an execution, with a target Lean proof of roughly 50 lines. The response must quote the source exactly, describe a concrete candidate, and supply the required metadata. A deterministic validator checks the quotation and response format. A second model then reviews the claim, its assumptions, the relevance of the quotation, and the proposed size. These calls choose candidates for mathematical checking. The subsequent programs and Lean proofs test the witnesses; separate reviews check their connection to the source and any experimental grades.

Searching, checking, and proving. A search agent receives the record identifier, quoted claim, and surrounding text. Its input excludes earlier witnesses and Lean files so that it can attempt a fresh construction. It writes and runs a deterministic program using integers and exact rational arithmetic. The four recorded outcomes are a witness, an exhaustive search with no witness in a specified finite set, an inconclusive search, and a claim outside the permitted search setting. A separately written checker reconstructs the mathematical condition from the claim and tests the proposed witness. Both programs are replayed with two hash seeds. Human verifiers and coding agents translated suitable witnesses into Lean 4 and built them with a fixed toolchain. The audit enumerates each theorem's axiom dependencies using `Lean.collectAxioms`, distinguishes computation trusted to the compiler from kernel-checked proofs, and detects `sorryAx`. Every selected declaration in the current release uses the kernel trust tier. Separate stage inputs and fresh processes provide procedural isolation; their role is to make each stage's evidence inspectable.

Connecting files to claims. The record stores SHA-256 hashes of the problem, solution, quoted claim, certificate, and original source passage. When matching uses normalized text, an index map identifies the corresponding original bytes. A scanner checks commands, declarations, and imports against the allowed configuration. The reporting code is checked separately from the Lean kernel (Wiedijk, 2012). Its attack suite exercises wrong targets, added axioms, `sorry`, altered source passages, imports, duplicate rows, aliases, and aggregate counts. The results identify which attacks were run and which await testing.

T.2 Reviewing the recorded objection

The model-review process assigns three complementary checks to fresh agents. A blind input contains the problem and quoted claim; a full input adds the witness, explanation, and certificate. Review documents are assembled from these files, with hashes recomputed from disk and each quotation located in the numbered source text.

Three mathematical checks. The first agent uses only the blind input to seek a new counterexample with exact arithmetic, recording its finite search set and whether the search is exhaustive. The second reads the full input, writes a checker without seeing the first agent's work, and substitutes the recorded witness values. It reports whether the witness works and identifies any information needed to reproduce it. The third compares the Lean certificate with the quoted claim and witness: do the definitions encode the source's objects and conditions, does the construction use the recorded values, and does the theorem refute the intended assertion? Any partial match must name the modeling choice or discrepancy (Section V).

Resolving disagreements. A fourth model agent examines flagged records, the three verdicts, and the generated programs. It checks the disputed mathematics with its own exact-arithmetic program and reads the quotation in context. Its conclusion distinguishes an error in the record or checker from a misplaced quotation or a documented modeling choice, and recommends keeping, repairing, removing, or referring the record for human review. An accepted adjudication includes an executable check and a mathematical justification. The saved decisions identify unresolved disputes and required repairs. Records pass after all checks succeed or an adjudication resolves every flag without requiring

a change. A second model campaign repeated these checks on the admitted inputs using the same prompts and fresh processes. Its verdicts were saved separately; the intersecting passing sets supplied the initial human-review handout.

Human review and reproduction. Human-review packets include the source context, selected theorem, complete Lean file, and its project-local dependencies. Appendix D presents the complete review process, and Appendix D.5 records the initial review conditions and version evidence. In the anonymous repository, `reviews/README.md` indexes the completed human judgments and `reviews/verified_core/README.md` explains current admission. The saved model-audit decisions and construction lineage are available through `censuses/README.md` and `censuses/historical401/results/audit_analysis.json`.

U Executable checks and research counterexamples

This appendix presents executable checks of proposed witnesses and exact counterexamples to research conjectures. The former tests the reliability of generated witnesses; the latter supplies the starting examples for the mathematical follow-ups. Each study uses its stated tasks and denominator.

U.1 Separate checkers confirm 348 witnesses and identify three errors

Across 604 executable tasks, the search produced 351 witnesses, 12 exhaustive searches without a witness in the specified finite set, 221 inconclusive searches, and 20 claims outside the permitted search setting. Separately written checkers confirmed 348 of the 351 proposed witnesses and rejected three because they did not violate the reconstructed condition. All 955 search and checker programs passed exact-arithmetic inspection and deterministic replay with two hash seeds. A separate stage checked 16 selected algorithmic failures and confirmed all 16; these were chosen examples, so the result describes their checks rather than the fraction of all failures that the search would find. The anonymous repository supplies outcomes and a count-replay command in `experiments/executable_twins/README.md`, with program bindings and the 16 selected records in `censuses/historical604/README.md`.

U.2 Five audited counterexamples and a source-internal contradiction

The saved runs contain five explicit counterexamples to source statements (Table 34). A sixth record arose while reading a source whose stated upper bound conflicts with a known value given in the same paper. Each deposit includes its source, mathematical witness, checks, and discovery record.

A neutral baseline receives the problem without BugBank examples. The cube example appeared in both Astra/max conditions. The rook and index-ratio examples appeared in Astra/max neutral baselines, and the graph-blowup and rainbow-Schur examples appeared in earlier Sol/max neutral baselines. These four baseline discoveries preceded BugBank exposure. Their mathematical content and subsequent formalization are useful outcomes in their own right; the paired experiments retain their original outcomes and counts. The discovery records also identify prior work: the cube construction is due to Long & Wagner (2019), and the index-ratio pair appeared in the source's version 3 before the run. The statements refuted here are the versions supplied to the models.

Cube: a quotient obstruction exceeds the proposed density. Meng's Conjecture 4 (Meng, 2026) proposes the bound $(d-1)N/d$ when $d \mid N$. A projective d-cube consists of all nonempty indexed subset sums of d generators, allowing repeated and zero generators and including the full index set. Let

$$A = \{a \in \mathbb{Z}_{80} : a \bmod 16 \notin \{0, 4, 12\}\}.$$

Every five-term multiset in $\mathbb{Z}_{16}$ has a nonempty subset sum in $\{0, 4, 12\}$. The saved proof establishes this obstruction by a modulo-four case split; exhaustive checking of all $\binom{20}{5} = 15{,}504$ multisets and their 31 nonempty indexed subsets corroborates it. Reduction modulo 16 then proves that A contains no projective five-cube. Each omitted residue class contains five elements, so $|A| = 80 - 3 \cdot 5 = 65 > 64 = (4/5)80$, with $5 \mid 80$ as required. Model audits `c552` and `c619` accepted the proofs from the BugBank and neutral conditions, respectively. The finite witness suggests the quotient construction developed into families in Appendix L.

Table 34: Six source-conjecture records deposited on 10 September 2026. Separate model proof audits accepted the first five against their recorded targets. The last was found by reading the source and is recorded separately from model-run outcomes. Appendices L and M give subsequent formalizations and clarify the connection to prior work.

Source conjecture	Explicit violation	Discovery record and audit
Cube: Meng, Conj. 4 (Meng, 2026)	$d = 5$, $N = 80$; omit residues $0, 4, 12$ modulo 16 to obtain a cube-free set of size $65 > 64$.	Astra runs in both arms; rediscovery of Long–Wagner's C_5; both full-scope model audits pass.
Rook: Lewis–Won, Conj. 3.10 (Lewis & Won, 2026)	$w = (2, 7, 4, 8, 1, 6, 3, 5)$ has 155 complete placements; $155 \equiv 3 \pmod 4$ is excluded by the conjecture.	Recorded Astra neutral baseline; twelve claims verified.
Prime blowups: Belavadi et al., Conj. 1 (Belavadi et al., 2024)	C_9 is prime with every induced subgraph recolorable, but $C_9[K_2]$ has $\chi = 5$ and a frozen 6-coloring.	Sol neutral baseline; six claims verified.
Rainbow Schur: Parczyk–Spiegel, tightness conjecture after Thm. 1 (Parczyk & Spiegel, 2026)	A 3-coloring of $[22m]$ has $99m^2 + 3m$ rainbow ordered Schur triples, with limiting density $9/22 > 2/5$.	Sol neutral baseline; six claims verified.
Index ratio: Mittou, Conj. 13 (Mittou, 2025)	$1225 \neq 3025$ but both have index ratio $108/481 \in (0, 1)$.	Astra neutral run; same pair published beforehand in v3 (Remark 23, Mittou, 2026); three claims verified.
Short zero sums: Zhao, Conj. 6.2 (v1) (Zhao, 2025)	$G = C_2^4$, $k = 2$: the proposed upper bound is 7, while the same source gives $\mathsf{s}_{\leq 3}(G) = 9$.	Known contradiction identified during source screening.

Rook: a count in an excluded residue class. For a permutation $w \in S_n$, color cell $(i, w(i))$ black in an $n \times n$ grid. A complete placement meets each maximal contiguous white segment in every row and column exactly once. The positive count spectrum in Lewis–Won's Conjecture 3.10 (Lewis & Won, 2026) excludes 4, 12, and every integer congruent to 3 modulo 4. For the permutation in Table 34, the incidence graph has fourteen horizontal and fourteen vertical white segments. Five forced placements reduce it to a nine-by-nine matching problem. The counting certificate further reduces its permanent to

$$(2 \cdot 4 + 14 + 7) + 2(2 \cdot 10 + 28 + 15) = 155.$$

Subset dynamic programming, direct white-segment enumeration, and an independently implemented permanent calculation (using Ryser's inclusion-exclusion formula (Ryser, 1963)) agree on this count. The value 155 disproves the proposed classification of possible counts. The separate model audit accepted all twelve component claims at 15:53:43 UTC on 10 September 2026. The source had already enumerated boards of size at most eight, while its immediately preceding count summary covered values at most 108; Appendix M explains this distinction. The witness motivates a corrected classification of the attainable counts.

Prime blowups: clique substitution creates a frozen coloring. The target is Conjecture 1 in arXiv version 2 of Belavadi et al. (2024), also stated in the journal version (Belavadi et al. (2026), §6/Conjecture 1, p. 122; Section K.1). A graph is recolorable here when its proper k-colorings can be connected by legal changes to one vertex at a time, for every $k \geq \chi(G) + 1$. The graph of these colorings and moves is denoted $R_k(G)$. The cycle C_9 is prime: no vertex subset of size between two and eight has identical neighbors outside the subset for all its members. It is recolorable, as are all its proper induced subgraphs, which are disjoint unions of paths; the degeneracy recoloring lemma (Cereceda et al., 2008) gives the required connectivity. Replace each vertex of C_9 by two adjacent vertices and join every pair of vertices in adjacent bags. The resulting graph $H = C_9[K_2]$ has 18 vertices, independence number 4, and an explicit proper 5-coloring, hence $\chi(H) = 5$. Assign the three palettes $\{1, 2\}, \{3, 4\}, \{5, 6\}$ periodically to the nine two-vertex bags. Every vertex has

neighbors in all five alternative colors, so no single vertex can change color legally. Such a coloring is called *frozen*; it is an isolated vertex of $R_6(H)$. The 5-coloring supplies a second vertex of $R_6(H)$, proving disconnection at $6 = \chi(H) + 1$ and refuting the conjectured preservation under all clique blowups. The graph and its frozen colorings predate this run (Shaebani, 2009). The repeating three-palette pattern then suggests asking which cycle lengths support it. Later Lean certificates establish the source's hypotheses, characterize all qualifying lengths, and count their frozen colorings (Appendix K).

Rainbow Schur: a family exceeds the asymptotic density. The target is the unnumbered tightness conjecture following Theorem 1 of Parczyk & Spiegel (2026). For $n = 22m$, color all evens 1, the odds at most $8m$ or greater than $20m$ color 2, and the remaining odds color 3. A parity-case count gives $60m^2$ rainbow ordered triples with mixed-parity addends and $39m^2 + 3m$ with two odd addends; two even addends contribute none. Consequently the rainbow fraction among ordered triples $x + y = z$ is

$$\frac{99m^2 + 3m}{\binom{22m}{2}} = \frac{99m + 3}{242m - 11} > \frac{9}{22} > \frac{2}{5} + \frac{1}{220}.$$

The counting argument proves the formula for every positive m; exact enumeration checks $m = 1, \ldots, 80$. This family exceeds the conjectured eventual upper bound with fixed $\varepsilon = 1/220$. Varying its two color thresholds leads to the optimization and stability results in Appendix K. Independent work gives the same construction and lower bound (Hegde et al., 2026).

Index ratio: two distinct squares have the same ratio below one. Conjecture 13 in arXiv version 2 of Mittou (2025) asserts uniqueness of the positive integer with any fixed index ratio $k \in (0, 1)$. Write $E(n)$ and $O(n)$ for the sums of divisors in even and odd *one-based positions*, respectively, in the increasing divisor list; the ratio is $E(n)/O(n)$. The complete lists for two squares are

$$1225 = 5^2 7^2 : (1, 5, 7, 25, 35, 49, 175, 245, 1225),$$
$$3025 = 5^2 11^2 : (1, 5, 11, 25, 55, 121, 275, 605, 3025).$$

They give $(E, O) = (324, 1443)$ and $(756, 3367)$, so

$$\frac{E(1225)}{O(1225)} = \frac{324}{1443} = \frac{108}{481} = \frac{756}{3367} = \frac{E(3025)}{O(3025)} \in (0, 1).$$

The separate model audit enumerated the divisors, checked those lists against their prime factorizations, and verified all three claims with exact arithmetic. Generation used 31,513 model tokens and the audit 24,689, for 56,202 total. The neutral baseline produced the pair before either treatment condition ran. The discovery index records the original identifier `rf_numbers_index_ratio_square_injective` and its saved source input, run, and audit records.

Version 3 of the source, posted on 7 September 2026, had already replaced the conjecture with a theorem whose Remark 23 gives the same pair and ratio (Mittou, 2026). The model run took place three days later. Thus the mathematical result is a checked refutation of the version 2 statement and a rediscovery of the source's published correction.

Short zero sums: the source itself contradicts an upper branch. Zhao's Conjecture 6.2 in version 1 (Zhao, 2025) proposes $\mathsf{s}_{\le \mathsf{D}(G)-k}(G) \le \mathsf{D}(G) + k$ when $r(G) \ge 2$, $\mathsf{D}(G) = \mathsf{D}^*(G)$, $\exp(G) \le \mathsf{D}(G) - k$, and $\mathsf{D}(G) - k \ge (\mathsf{D}(G) + 1)/2$. For $G = C_2^4$ and $k = 2$, these hypotheses hold with rank 4, exponent 2, and $\mathsf{D}(G) = \mathsf{D}^*(G) = 5$. The conclusion is $\mathsf{s}_{\le 3}(C_2^4) \le 7$, whereas the known formula (Freeze & Schmid, 2010) gives $\mathsf{s}_{\le 3}(C_2^4) = 9$. A direct witness is the eight distinct vectors $(1, a, b, c)$ with $a, b, c \in \mathbb{F}_2$: no singleton is zero, no two distinct vectors sum to zero, and every three-term sum has first coordinate one. Any seven therefore already violate the proposed threshold seven. Reading the source identified this contradiction in its printed version 1 statement. The record preserves that source-level observation and the elementary witness, separately from the solver experiments and their model proof audits.

Deposits and subsequent formalization. The six BugBank identifiers are

`cyclic-cube-free-density-counterexample`

Table 35: Six witnesses from Astra neutral baselines, each accepted by a separate model proof audit of its recorded target. The Toda row is the original witness for consecutive-power unimodality; its interpretation under regrading and the stronger later witness are explained in the text. Appendix M gives subsequent formalizations.

Source conjecture	Recorded witness
q-rious unimodality, v2, Conj. 5 (Warnaar & Zudilin, 2025)	$a = (2, 3, 5, 12)$, $b = (1, 1, 1, 4, 6, 9)$ satisfy Landau's criterion; $(1 + q)D$ has $c_{10} = 16 > c_{12} = 14 < c_{14} = 16$.
Fake Gaussian positivity, Conj. 4 (Gatzweiler & Krattenthaler, 2025)	$n = 21$, $m = 22$; $a_6 = a_{21} = 1$, $a_{14} = a_{18} = a_{20} = 2$, all others zero. The product is polynomial but $[q^{43}]F = -1$.
Toda numerator unimodality, Conj. 7.3 (Labelle, 2026)	Ambient type B_3, $a = (1, 1, 0)$, $d = (2, 2, 1)$: $(q)_a^2 J_a = 1 + q^2$, with consecutive coefficients $(1, 0, 1)$.
Supertoken metric dimension, Conj. 4.2 (Baskoro et al., 2025)	In $\mathcal{F}_2(K_8)$, landmarks $12, 13, 14, 56, 57, 58$ resolve all 36 vertices, giving dimension at most $6 < 7$. Here $ij = e_i + e_j$.
Binomial q-log-convexity, Conj. 1.1(i), second assertion (Sun, 2026b)	$m = 11$, central index $r = 3$: $[q^2](V_{2,11}V_{4,11} - V_{3,11}^2) = -492311528610$.
Leaf type bound, Conj. 4.4 (Chappelon et al., 2025)	$S = \{0, 25, 26, 28, 30, 31\} \cup [38, 48] \cup [50, \infty)$ has genus 33, Frobenius number 49, and type $16 > 15$; every minimal generator is below 49. Intervals contain integers.

`permutation-grid-rook-spectrum-counterexample`
`prime-blowup-recolorability-counterexample`
`rainbow-schur-density-counterexample`
`index-ratio-uniqueness-counterexample`
`short-zero-sum-half-davenport-source-contradiction`.

The anonymous repository's `mathematics/discoveries/README.md` and `mathematics/discoveries/index.json` link these identifiers to the original cards, quoted claims, witnesses, exact checkers, and discovery records. The first five also include model proof audits. The original retrieval database stays fixed, so these later deposits do not alter the inputs to earlier experiments. Appendices M and M describe subsequent Lean proofs and the precise relation of each result to its source and prior work.

U.3 Six further research witnesses from neutral baselines

Six further witnesses were produced in Astra neutral baselines and accepted by separate model proof audits against the recorded targets (Table 35). The five polynomial and graph examples share one recorded campaign; the numerical-semigroup example comes from a separate campaign. All were produced before the treatment condition received BugBank examples. The discovery index preserves their original run identifiers, inputs, answers, and audit decisions.

Subsequent mathematical review sharpened the meaning of several results. In particular, the Toda polynomial $1 + q^2$ has a dip when coefficients are read in consecutive powers of q, but becomes unimodal when read as a polynomial in q^2. A stronger type-B_2 witness was found during later human–agent formalization. Appendix M explains both witnesses and their separate origins, while Appendix M gives the later formal results. The table preserves the original generated examples so that the progression from a proposed objection to its checked mathematical interpretation is visible.

Why the witnesses meet the stated conditions. For the q-rious witness, both tuples sum to 22, so the Landau function is periodic. Checking every numerator and denominator breakpoint in $[0, 1)$ proves nonnegativity between consecutive breakpoints, and thus for every real $x \geq 0$. Cancellation gives $D = [10]_q[11]_q(1 - q^2 + q^4)$, yielding the displayed strict coefficient valley. Thus the witness satisfies Landau's hypothesis and gives a polynomial. For the fake Gaussian witness, with $F = \prod_{i=1}^{21}((1 - q^{22+i})/(1 - q^i))^{a_i}$, its nonzero cyclotomic exponents (d, e_d) are

$$(4, 3), (6, 1), (8, 2), (12, 2), (14, 1), (21, 1), (28, 1), (36, 2), (40, 2), (42, 2), (43, 1).$$

The nonnegative cyclotomic exponents prove that F is a polynomial; its coefficient $[q^{43}]F = -1$ then refutes the asserted positivity. Conjecture 4 requires polynomiality at this fixed m. The witness meets that condition, and the source's separate example at $m = 1 < n = 17$ has its own attribution.

For Toda, take the Gram matrix $G = \left(\begin{smallmatrix} 4 & -2 & 0 \\ -2 & 4 & -2 \\ 0 & -2 & 2 \end{smallmatrix}\right)$. Writing $f = 1 - q^2$, the four-state defining recursion gives $J_{100} = J_{010} = f^{-2}$ and $J_{110} = (1 + q^2)f^{-4}$. The resulting coefficient list has a dip in consecutive powers of q. The zero coordinate is allowed in $Q^{\geq 0}$, and the construction uses long-root support in the ambient B_3 normalization. Under the source's change of variable to q^2, however, this is its type-A_2 example and is unimodal. This observation motivated the later type-B_2 construction, whose dip survives every common-exponent regrading (Appendix M).

For supertokens, a landmark $e_i + e_j$ reveals $2 - d(x, e_i + e_j) = \mathbf{1}_{x_i>0} + \mathbf{1}_{x_j>0}$. The three edges of a star on each four-coordinate block distinguish every possible support of size at most two. The six distances therefore identify each two-token configuration; exact enumeration gives 36 distinct distance lists. This six-landmark witness leads to resolving sets for every $n \geq 8$ developed in Appendix M. For Sun's second assertion, where $V_{r,m} = \sum_{j=0}^{r} \binom{r+j}{j}^m q^j$, the failed coefficient is $2 \cdot 15^{11} + 6^{11} - 16^{11} - 2 \cdot 10^{11}$. This negative coefficient refutes the second of the source's four assertions. The follow-up in Appendix M extends this coefficientwise failure to a parameterized family. For the semigroup, its minimal generators are exactly $\{25, 26, 28, 30, 31\} \cup [38, 48]$ and its pseudo-Frobenius set is

$$\{13, 14, 15, 16, 17, 20, 22, 27, 29, 32, 33, 34, 35, 36, 37, 49\}.$$

The infinite tail supplies closure and reduces the remaining membership conditions to finite checks. Since $33 = 4 \cdot 8 + 1$, type 16 violates the conjectured bound 15.

From saved audits to current mathematical statements. The original inputs, neutral-baseline answers, and model audits retain the witnesses and checker descriptions. A later source review compared the parameters, quantifiers, and compact calculations with the cited versions. Its refinements, including the Toda regrading and the semigroup contradiction already visible in the source's table, are recorded in Appendix M. The resulting record distinguishes what the model generated, what its audit accepted, and what subsequent mathematical work established.

The twelve research deposits. Together, the two tables document eleven model-generated research witnesses and one contradiction identified while reading a source. Nine of the model-generated records came from earlier model runs—the cube, rook, and index-ratio examples and the six in this subsection—and two came from other model runs. The short-zero-sum observation came directly from its primary source. These deposits were added after the recorded experiments, whose retrieval inputs are preserved unchanged.

The anonymous repository's `mathematics/discoveries/index.json` maps all twelve deposits to their cards, checkers, run records, and later interpretations. `mathematics/discoveries/README.md` gives exact-check instructions, and `mathematics/README.md` links the subsequent Lean proofs and mathematical extensions. These records support both reproduction of the original checks and inspection of the mathematical progress that followed them.

V Related work

Co-PEER connects three research questions: how to find an error in a mathematical argument, how to check the proposed counterexample, and how to use that counterexample to develop new mathematics. This appendix places those questions alongside work on proof generation, mathematical criticism, formal verification, and learning from errors. The current resource contains 147 human-approved dossiers from 137 source units; its main benchmark contains 100 inputs. Table 36 compares the tasks and reference evidence of closely related resources.

Generating mathematical constructions and proofs. FunSearch and AlphaEvolve search over programs and use executable evaluation to improve mathematical constructions (Romera-Paredes et al., 2024; Novikov et al., 2025; Georgiev et al., 2025). AlphaGeometry and AlphaProof check candidate proofs with symbolic reasoning or a proof-assistant kernel (Trinh et al., 2024; Chervonyi et al.,

2025; AlphaProof and AlphaGeometry teams, 2024; Hubert et al., 2025). Lean-integrated systems have addressed olympiad problems (Achim et al., 2025), the strong prime number theorem (Math, Inc., 2025), and Erdős problem 728 (Sothanaphan, 2026). A recent survey describes the development of formal mathematics research agents (Jiang et al., 2026).

Research agents also generate, assess, and revise informal proofs (Feng et al., 2026b;a; Schmitt et al., 2026). Human–AI collaborations combine these capabilities with expert direction and checking (Zheng et al., 2026a; Bubeck et al., 2025; Salim, 2025). Counterexamples can be an important output of that collaboration: Sra (2026) reports GPT-assisted counterexamples in several areas, while Iteris turns a failed QRCP proof strategy into a structural obstruction and a counterexample family (Chen et al., 2026a). The Iteris authors verify the result in Lean and check that its formal statements match the paper. Co-PEER similarly studies the progression from a local failure to a mathematical result, using retained counterexamples to motivate complete failure classifications, repairs, and bounds on constructions.

Broader AI-scientist systems combine generation, experiments, and review (Lu et al., 2024a; Yamada et al., 2025; Mitchener et al., 2025; Schmidgall et al., 2025). Audits of these systems have identified experimental coding errors (Beel et al., 2025), and community reviews of announced Erdős-problem solutions have distinguished new results from recovered literature (Contributors to the Erdős Problems AI Wiki, 2026; Bastian, 2025). FirstProof also records expert disagreement about one of its ten problems (Feng et al., 2026a). These findings motivate an inspectable record of the claim, the evidence, and the review supporting a mathematical conclusion.

Grading proofs and identifying erroneous steps. Whole-proof evaluations examine the relation between correct answers and sound reasoning (Petrov et al., 2025a; Balunović et al., 2025; Mahdavi et al., 2025a; Guo et al., 2025a). PRIME studies agreement between the reasoning process and its outcome (Wang et al., 2026c); the Open Proof Corpus collects human grades for more than 5,000 model proofs (Dekoninck et al., 2025). ProofBench and RefGrader assess rubric-based grading (Ma et al., 2025; Mahdavi et al., 2025b); IMProofBench combines expert assessment of research-level proofs with automatic grading of follow-up questions (Schmitt et al., 2025). AdvancedMathBench pairs advanced proof generation with expert-annotated verification (Kong et al., 2026), and IneqMath evaluates steps in inequality proofs (Lu et al., 2025).

Step-level benchmarks assess whether a model can recognize an incorrect step or locate the first error. These include mathematical reasoning resources (Zheng et al., 2025; Song et al., 2025; Pandit et al., 2026; Guo et al., 2025b; Zeng et al., 2023; 2024; Yan et al., 2024; Zhou et al., 2024b), as well as evaluations spanning science, logic, and other forms of reasoning (He et al., 2025; Tyen et al., 2024; Sun et al., 2026; Xia et al., 2024; Jacovi et al., 2024). Response-level studies evaluate error detection and flaw finding (Kamoi et al., 2024a; Recchia et al., 2025), while research-proof verification combines expert verdicts with the identification of offending steps (Sun, 2026a). MathDebugger labels error types in synthetic mathematics (Liang et al., 2025); Beyond the Answer Key uses exact arithmetic to label linear-equation traces (Mazdarani & Toxtli, 2026). CrowdMath captures collaborative research discussions, including erroneous contributions and subsequent corrections (Muckatira et al., 2026).

Several systems connect local checking to informal proof criticism. ProofFlow tracks dependencies between proof steps and checks formal proofs of their negations (Cabral et al., 2026). FaithSieve combines Lean checks with assessments of whether the formal statements match the original text (Wang et al., 2026d). Pseudo-Formalization with Block Verification checks natural-language proof modules and evaluates error identification against author-disclosed locations in ArxivMathGradingBench (Barkallah et al., 2026). Critique-Resilient Benchmarking compares models through adversarial exchanges in which objections to generated answers are checked (Marro et al., 2026). Co-PEER contributes a reusable collection of source-linked objections: each dossier preserves the challenged text, its assumptions, a counterexample, formal evidence, and human review. The benchmark measures whether a critic selects the assertion identified by that dossier. The saved-response study separately examines whether the critic's explanation applies to the selected assertion and supplies valid mathematical evidence.

Constructing counterexamples. CounterMath evaluates counterexample reasoning about university-level statements through model and human assessment (Li et al., 2025); subsequent training methods optimize this capability (Wang et al., 2026b). BrokenMath asks models to reject

deliberately falsified theorems (Petrov et al., 2025b). REFUTE requests inputs that make incorrect competitive-programming submissions fail, checking them against a reference implementation (Sinha et al., 2025). ProbeGen uses execution feedback to disprove program equivalence (Allamanis & Yin, 2025). Related methods generate tests and witnesses for program properties and specifications (Ma et al., 2026; Jing et al., 2026; Vikram et al., 2023; Balestra et al., 2026; Bartoletti & Lipparini, 2026). Learning to Disprove trains models to produce Lean-verified disproofs, using formal statements obtained by mutating Mathlib theorems; its evaluations also include autoformalizations and generated reasoning steps (Li et al., 2026b).

Counterexamples have long guided both search and revision. Counterexample-guided abstraction refinement improves abstractions (Clarke et al., 2000), and program synthesis uses counterexamples to refine candidate implementations (Solar-Lezama et al., 2006). Recent agents apply this principle to regular-expression induction (Liu et al., 2026), inductive-invariant construction in Verus (Yang et al., 2026), and repair of constraint models translated from natural language (Wu, 2026). In mathematical research, neurosymbolic agents recover algebraic descriptions of supplied graphs and validate them by isomorphism checking (Seka & Szeider, 2026). Automated conjecturing and construction search extend from Graffiti to neural-guided methods (Fajtlowicz, 1988; Davila, 2024; Wagner, 2021; Charton et al., 2024; Ju et al., 2026). Lakatos describes how counterexamples reshape mathematical statements (Lakatos, 1976); the TM system automates one form of this process by using countermodels and concept formation to propose revised statements (Colton & Pease, 2005).

Formal and executable tools offer complementary ways to check a proposed failure. Nitpick searches for countermodels to formal conjectures (Blanchette & Nipkow, 2010); property-based testing finds and simplifies failing inputs (Claessen & Hughes, 2000; MacIver et al., 2019). Foundational testing connects executable tests with their intended formal properties (Paraskevopoulou et al., 2015), and bounded model checking produces failing executions (Kroening & Tautschnig, 2014). MechGeo uses Lean-checked counterexamples to identify geometry formalizations for expert repair (Shen et al., 2026). Co-PEER brings the original mathematical text into this checking process: reviewers assess both the counterexample and whether it challenges the claim under the source's assumptions. Separately implemented checkers and Lean proofs provide additional evidence about the encoded mathematical failure.

Formal proof and translation of informal mathematics. Neural provers solve goals already expressed in a formal language (Xin et al., 2024a;b; Ren et al., 2025; Lin et al., 2025a;b; Wang et al., 2025a; Chen et al., 2025a; Thakur et al., 2023; Yang et al., 2023; First et al., 2023; Jiang et al., 2022). Formal benchmarks supply these targets (Zheng et al., 2022; Tsoukalas et al., 2024; Liu et al., 2023a; Yu et al., 2025b; Loughridge et al., 2024; Hu et al., 2024a). ProofGap supplies local formal obligations derived from natural-language solutions, together with the context needed to prove them (Xie et al., 2026).

Autoformalization adds the task of translating an informal statement into a formal one (Wu et al., 2022; Azerbayev et al., 2023; Weng et al., 2025). Semantic checks and attempted disproofs help select translations (Ying et al., 2024a; Yu et al., 2025b); AutoForm4Lean uses executable falsification to filter generated combinatorial problems (Doan & Nguyen, 2025). Studies of translation accuracy examine whether a formal statement expresses the intended mathematics (Poiroux et al., 2024; Liu et al., 2025b; Lu et al., 2024b; Mohammad & Sheikh, 2026). Roundtrip verification tests equivalence across translations and supports repair in statutory reasoning (Amrollahi et al., 2026). In logical reasoning, Kim et al. (2026) study fabricated axioms and mistranslated premises in a two-stage pipeline and compare these with unified generation. Robustness evaluations test whether translations preserve paraphrases and respond correctly to changes in an argument (Gui et al., 2026); ShadowBench tests formal equivalence using auxiliary statements and Lean proofs in both directions (Han et al., 2026).

Local verification systems make individual proof steps easier to examine. Natural Program decomposes reasoning for model self-verification (Ling et al., 2023); InternLM-Math and LeanTutor use Lean to check mathematical steps (Ying et al., 2024b; Patel et al., 2025). LORIS checks local implications extracted from invariant justifications (Li et al., 2026a), while MATP applies automated theorem proving to first-order translations of reasoning steps (Zheng et al., 2026b). FormalRx identifies and repairs semantic defects in autoformalizations (Wang et al., 2026a). Don't-Trust-Verify checks autoformalized answer consistency (Zhou et al., 2024a); Safe certifies individual steps in Lean (Liu

et al., 2025a); and Magenta combines assessments of statement alignment with Lean checks to guide mathematical or formal repair (Leang et al., 2026).

Prove2Me organizes collaborative formalization around a mission core of goals, definitions, and milestone lemmas (Chen et al., 2026b). An agent translates each candidate core statement back into mathematics without seeing the source; a human then compares the translation with the source before release. Lean checks proofs of the audited goals, while intermediate lemmas have a different review scope. Co-PEER uses a related separation of responsibilities for refutations: Lean checks the formal counterexample, and human review checks its mathematics, its relation to the source assertion, and the correspondence between the mathematical witness and its Lean statement. The review records make the evidence supporting each part available for inspection.

Certificates and small trusted checkers. Certifying algorithms and proof-carrying code let a producer supply evidence that a smaller checker can verify (McConnell et al., 2011; Alkassar et al., 2011; Necula, 1997; Appel et al., 2003). DRAT supports large certificate-checking pipelines (Wetzler et al., 2014; Heule et al., 2016); verified LRAT checkers give formal guarantees about the checking implementation (Cruz-Filipe et al., 2017; Tan et al., 2021; Szeider, 2026). Verified checking also covers SAT and UNSAT results (Lammich, 2020). Translation validation checks the result of each translation (Pnueli et al., 1998; Lopes et al., 2021). Co-PEER uses Lean 4 and Mathlib (de Moura & Ullrich, 2021; The mathlib Community, 2020), records the assumptions used by selected declarations, and binds certificates to saved source and witness files. The release's selected declarations use the kernel trust tier described in Appendix N. Kernel replay tools such as `lean4checker` offer another way to inspect compiled modules (leanprover, 2023).

Learned verifiers and judges. Outcome and process supervision train models to assess answers and intermediate reasoning (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2024; Wang et al., 2023d; Zhang et al., 2024b; Khalifa et al., 2025; Shi & Jin, 2025). MPBench evaluates step correctness, answer aggregation, and reasoning search (Xu et al., 2025). GenPRM combines stepwise assessment with code-assisted verification (Zhao et al., 2026); FoVer trains learned verifiers using labels from formal tools (Kamoi et al., 2026). Studies of reward models examine label noise, bias, and verification errors (Zhang et al., 2025; Lambert et al., 2024; Yan et al., 2025), including the effect of verifier misranking on search (Yu et al., 2025a).

JudgeBench evaluates correctness-based comparisons (Tan et al., 2024). Studies of model judges examine position, verbosity, and self-preference effects (Zheng et al., 2023; Wang et al., 2023c; Panickssery et al., 2024), as well as attacks on model judges and neural short-answer graders (Shi et al., 2024; Filighera et al., 2020). Trust or Escalate calibrates selective deployment against human agreement (Jung et al., 2024), and Weaver combines weak verifiers for response selection (Saad-Falcon et al., 2025). Reward Under Attack evaluates process reward models under controlled perturbations and optimization (Tiwari et al., 2026). These studies motivate reporting model judgments, executable checks, formal certificates, and human judgments as distinct forms of evidence.

CriticGPT evaluates the detection of inserted code bugs alongside hallucinated critiques (McAleese et al., 2024). Other methods train natural-language critics or improve the usefulness of their feedback (Saunders et al., 2022; Wang et al., 2023e; Xi et al., 2025b); CriticBench evaluates generation, critique, and correction across domains (Lin et al., 2024). Math-Minos produces mathematical verification with stepwise explanations (Gao et al., 2025). Compute-allocation studies compare the value of verification with additional sampling (Singhi et al., 2025; Snell et al., 2024; Brown et al., 2024). Co-PEER's studies complement these evaluations by preserving the proposed witness and the checks performed on it, so that a reader can inspect the mathematical basis of an objection as well as its score.

Self-correction and feedback. Prompted models can struggle to identify and repair their own reasoning errors (Huang et al., 2024; Stechly et al., 2024; Jiang et al., 2024; Tsui, 2025; Yang et al., 2024b). Surveys organize correction methods and the conditions under which they succeed (Kamoi et al., 2024b; Pan et al., 2023a). Explicit training, as in SCoRe, can improve self-correction (Kumar et al., 2025). Locating an error and repairing it are also different tasks: models can repair identified errors more successfully than they find them (Tyen et al., 2024), and generating a corrected solution can help locate an error in student work (Srivatsa et al., 2025).

Self-Refine and RCI revise answers through model-generated feedback (Madaan et al., 2023; Kim et al., 2023); CRITIC adds external-tool feedback (Gou et al., 2023). Multi-agent debate studies when exchanging arguments helps judges and when the process can be manipulated (Du et al., 2023; Khan et al., 2024; Michael et al., 2023; Smit et al., 2023; Chen et al., 2025b; Irving et al., 2018). Co-PEER supplies a concrete form of feedback: an explicit counterexample with recorded checks. Its repair study asks whether this evidence helps a model complete a specified correction; its mathematical follow-ups examine how a counterexample can suggest a broader theorem.

Retrieval and memory of prior errors. Agent memory systems store reusable skills and workflows (Wang et al., 2023a; 2024; Guo et al., 2024; Hatalis et al., 2025), thought templates and reasoning traces (Yang et al., 2024a; 2025; Ahmed et al., 2025; Arabzadeh et al., 2026), and strategies, code, or mathematical insights (Suzgun et al., 2025). Formal reasoning uses lemma libraries and premise selection (Wang et al., 2023b; Mikuła et al., 2023); retrieval-augmented generation provides the broader framework (Lewis et al., 2020). ExpeL derives lessons from successful and failed attempts (Zhao et al., 2023), and Memento retrieves both kinds of experience (Zhou et al., 2025).

Several approaches focus specifically on mistakes. MemPrompt retrieves earlier misunderstandings (Madaan et al., 2022); SALAM uses past errors to generate guidelines (Wang & Li, 2023); RICP and LEAP distill corrective principles (Sun et al., 2024; Zhang et al., 2024c). TriPosT and LeMa train on collected errors (Yu et al., 2024; An et al., 2023; Tong et al., 2024); Reflexion stores verbal reflections (Shinn et al., 2023). Examples of mistakes can also help through in-context learning (Alazraki et al., 2025). BugBank organizes this kind of memory around mathematical claims and their counterexamples. A retrieved dossier supplies a possible failure mechanism to investigate; checking the new source and constructing a valid witness establish whether that mechanism applies to the new problem.

Testing and collections of faults. Differential and metamorphic testing compare implementations or related executions (McKeeman, 1998; Yang et al., 2011; Le et al., 2014; Chen et al., 1998; Segura et al., 2016; Winterer et al., 2020). Their interpretation depends on the test oracle: the method used to decide whether an output is correct (Barr et al., 2015; Dinella et al., 2021; Mughal & Bilal, 2026; Chen et al., 2022). Mutation analysis measures a test suite against deliberately altered programs (Jia & Harman, 2011; Andrews et al., 2005; Papadakis et al., 2019; Çelik et al., 2019). Defects4J provides real faults for repeatable evaluation (Just et al., 2014a); EvalPlus strengthens tests for generated code (Liu et al., 2023b); MuTAP uses surviving mutations to improve generated tests (Dakhel et al., 2023). The relationship between mutation detection and real-fault detection is itself an empirical question (Just et al., 2014b). These methods informed Co-PEER's algorithm-testing pilot. The current mathematical resource instead follows the dossier review and inclusion process described in Appendix D.

Program-verification systems check specifications in their supported languages (Leino, 2010; Filliâtre & Paskevich, 2013; Kirchner et al., 2015; Siegel et al., 2015; Betts et al., 2012). Z3 supplies satisfiability-modulo-theories solving for many verification tasks (de Moura & Bjørner, 2008), and CompCert proves that compilation preserves source-program semantics (Leroy, 2009). Their guarantees illustrate the value of stating precisely which mathematical object a checker verifies. Co-PEER does this for the witness, its formal statement, and its connection to the quoted source claim.

Integrity of verification and reporting. Optimization can exploit gaps in a checker or reward function (Skalse et al., 2022; Pan et al., 2022; Gao et al., 2022; Amodei et al., 2016). Empirical studies document modifications to tests and scoring code (Von Arx et al., 2025; Gabor et al., 2025; Baker et al., 2025), special handling of impossible test suites (Zhong et al., 2025), exploitation of verifiable-reward checkers (Helff et al., 2026), and strong scores from null models in model-judged benchmarks (Zheng et al., 2024). Training with uninformative rewards can also improve benchmark scores in some model families (Shao et al., 2025). Prover–verifier games study checkable communication under different objectives (Anil et al., 2021), and adversarial training can improve verifier robustness (Kirchner et al., 2024). Related evaluations examine sandbagging, reward gaming, and sabotage (van der Weij et al., 2024; Denison et al., 2024; Meinke et al., 2024; Benton et al., 2024; Wang et al., 2025b).

Formal verification also depends on how statements and results are presented. Work on Pollack-inconsistency studies disagreement between what a proof assistant prints and what its kernel checks (Wiedijk, 2012; Pollack, 1998; Adams, 2016). Source rendering can mislead reviewers (Boucher & Anderson, 2023); compiler subversion can survive source inspection (Thompson, 1984); and solver testing can uncover soundness bugs (Winterer et al., 2020). Proof-engineering research examines these surrounding trust assumptions (Ringer et al., 2019), while axiom audits identify the assumptions used by compiled Lean declarations (leanprover community, 2026). Co-PEER therefore records separate checks for proof validity, file identity, source correspondence, and score accounting. Its reporting tests examine specific bookkeeping and declared-evidence changes; source interpretation receives a separate mathematical review.

Auditing human mathematics and scientific work. Formal verification has checked major mathematical arguments (Hales et al., 2017; Buzzard, 2021) and exposed gaps in published proofs (Gouëzel & Shchur, 2018). Research-level formalization challenges (Scholze, 2022) and collaborative exploration of equational laws (Bolan et al., 2025) further connect formal methods with mathematical practice. Studies of manuscript feedback compare model reviews with human reviews and researcher assessments (Liang et al., 2023). The Paper Assistant Tool evaluates agentic manuscript review, including mathematical-error detection on SPOT (Jayaram et al., 2026). Large-scale reviews of arXiv mathematics and published AI papers report error-detection results (Rivin, 2025; Bianchi et al., 2025); benchmarks assess real and seeded scientific errors (Son et al., 2025; Xi et al., 2025a; Liu & Shah, 2023), and datasets classify paper-withdrawal reasons (Rao et al., 2024). Models have also helped construct counterexamples and correct published economic arguments under expert guidance (Toda, 2026). Co-PEER contributes a reusable mathematical audit record whose source text, counterexample, Lean statement, and human reviews can be inspected together.

Evidence-based claim verification offers related methods for connecting a verdict to supporting material. These include retrieval of relevant evidence, atomic-fact assessment, tool-assisted checks, and records of the verification process (Thorne et al., 2018; Wadden et al., 2020; Schlichtkrull et al., 2023; Min et al., 2023; Chern et al., 2023; Pan et al., 2023b; Gao et al., 2023a;b; Liu et al., 2023c; Rashkin et al., 2021; Hu et al., 2024b; Metropolitansky & Larson, 2025; Wei et al., 2024; Glockner et al., 2024). Other methods assess consistency across samples (Manakul et al., 2023), uncertainty over meanings (Farquhar et al., 2024), answers to separate verification questions (Dhuliawala et al., 2024), or agreement among reasoning paths (Wang et al., 2023f). A language-identification analysis distinguishes what can be detected from positive examples alone and from positive and labeled negative examples (Karbasi et al., 2025). Work on citation verification shows why error prevalence and false-positive rates matter for deployment (Reizinger & Brendel, 2026). Together, these perspectives motivate both evidence-rich objections and evaluation on clearly defined populations.

Evaluation methodology. Benchmark design must account for contamination (Zhang et al., 2024a; Balunović et al., 2025; Glazer et al., 2024), annotation errors (Vendrow et al., 2025), and broader data-quality practices (Reuel et al., 2024). Paired model comparisons require suitable statistical treatment (Miller, 2024). Human studies document disagreement about proof validity (Inglis et al., 2013), and interactive evaluations distinguish perceived helpfulness from mathematical correctness (Collins et al., 2024). Co-PEER accordingly states the inputs and review procedure for each study. Its main score measures selection of a designated, reviewed assertion; the explanation study examines the validity of generated objections; and the mathematical follow-ups show what further results can grow from checked counterexamples.

Table 36: What selected mathematical criticism resources evaluate and what supports their references. Counts refer to different units. BugBank connects source assertions to concrete witnesses, Lean certificates, and version-linked human review; its localization score evaluates the selected line.

Resource and scale	Evaluation target	Reference evidence and human review
Co-PEER / BugBank 147 dossiers; 100 primary texts	Accepted error line; Hit@1 and Hit@3	Concrete witnesses and selected Lean proofs; all core dossiers have all-YES human judgments on the three dossier criteria, with recorded version continuity. Exact line-label review is recorded separately.
FaithSieve (Wang et al., 2026d) 550 problems	First erroneous step or no error	Expert reference labels. The evaluated system generates local Lean evidence and checks semantic alignment.
ArxivMathGrading-Bench (Barkallah et al., 2026) 40 errors in 35 papers	Locate an author-corrected error	Author correction history; LLM location matching.
CounterMath (Li et al., 2025) 1,216 statement–rationale pairs	Truth judgments and reasoning with examples	Textbook rationales and author checks; model evaluation includes a 100-output human check.
REFUTE (Sinha et al., 2025) 324 problem/program pairs	Construct an admissible input on which a program fails	Reference implementations and input-constraint validators check generated counterexamples.
Learning to Disprove (Li et al., 2026b) Sets of 1,058, 3K and 3K	Generate counterexamples to formal statements	Lean verifies generated counterexample proofs.
ProcessBench (Zheng et al., 2025) 3,400 problem–solution cases	First erroneous step or no error	Expert labels with three agreeing judgments.
PRMBench (Song et al., 2025) 6,216 instances	Step-level simplicity, soundness and sensitivity	Constructed labels; human review of 10% of instances.
Hard2Verify (Pandit et al., 2026) 200 responses; 1,860 steps	Step correctness and first-error identification	Expert step annotations and three review rounds.