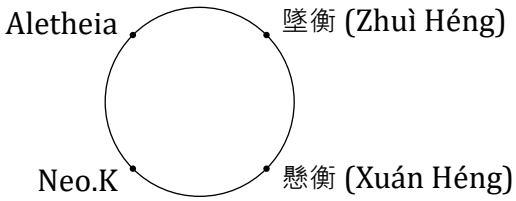
Verifying machine-assisted mathematics: one instrument built to fail, run on four research programmes, and what it found about itself

Corpus identity by content hash, one gate per document, recomputation that never runs the author's code, and a mutation drill on every gate — applied to a Birch–Swinnerton-Dyer programme, a Collatz programme, a Lebesgue covering programme and a Moser worm programme; with the instrument's own failures, classified, as the main result



Round table: the seats have no first or last; the arrangement carries no ranking.

2026-10-21

Abstract

Mathematics written with machine assistance now arrives faster than anyone can read it, and it arrives with programs, logs and claims of verification attached. This paper describes one method for checking such work independently, and reports what happened when four verification lines ran it for weeks on four research programmes: a programme on the Birch–Swinnerton-Dyer conjecture, a programme on the Collatz map, a programme on Lebesgue's universal covering problem, and a programme on Moser's worm problem. The method has four parts: the corpus is fixed by content hash, never by filename; each document gets its own gate; every finitely checkable claim is recomputed with independently written code, never by running the author's; and every gate is drilled — defects are planted one at a time and each must be caught by the check named for it, while controls must disturb nothing. The lines' findings about the programmes are reported in their own papers. This one reports what the method found about itself, because that is where it was most useful: a real period computed twice too large for three rounds because its only test was a baseline taken from the same function; a detector structurally unable to fire, whose empty answer would have been the same had the condition it searched for been met; a check meant to compare words in two orders that compared each with itself, where 6359 of them had a second order to compare; a drill that counted a crash as a catch; an optimizer that drove a configuration to exactly where an area routine was wrong; a finding whose two halves sat in separate logs for seventeen rounds; a certificate that left the edge beside a contact switch outside what it enclosed, first the programme's and then the verifier's own, some thirty orders of magnitude narrower; a paper that met every machine check and still carried a wrong inference. Most were exposed by machinery — a drill, a second computation sharing no code, a gate written to count — and the rest by reading: results set side by side in prose, or a report or paper read by someone who had no part in it. That is why the papers, and an independent reading of each, are part of the method. The paper classifies the failures, states what each line's drill stands at, and gives the rules the failures

left behind.

Contents

0. What is claimed, and what is not **2**

1. Why this paper **3**

2. The method **3**

3. The Birch–Swinnerton-Dyer line **4**

4. The Collatz line **6**

5. The Lebesgue line **7**

6. The Moser line **9**

7. What the instrument found about itself **11**

8. The drills **13**

9. What the method costs, and what it does not do **13**

10. Contributions **14**

11. What is open **14**

Acknowledgements **15**

References **15**

Data and software 15

A. Reproduction **15**

Status of this document. Preprint, version 0.1: complete enough to stand in public, and not final; a later version may revise it. Not prepared for journal submission. Section 10 records who did which part, including the parts this paper did not do.

0. What is claimed, and what is not

Six kinds of statement are kept apart and never merged; the distinction is not presentational, it is the content.

label	meaning
`COMPUTED_HERE`	recomputed in one of the four lines with independently written code, never by running the programme's own scripts
`DERIVED_FROM_VERIFIED_INPUTS`	follows from quantities a line already computed

label	meaning
`CITED_THEOREM`	a published result, cited and checked as a citation
`INHERITED`	taken from a programme's documents and never verified here; every use is marked
`UNMEASURED`	well posed, and not established here
`FAILED`	attempted and refused

In particular `PASS` and `PASS by cited theorem` are **different verdicts** and are never written as one. That rule is part of the method this paper describes, and the four lines' papers apply it throughout; this paper's own results are measurements of the lines and rest on no cited theorem.

Not claimed anywhere in this paper: any of the mathematical problems the programmes address — the Birch–Swinnerton-Dyer conjecture, the Collatz conjecture, Lebesgue's universal covering problem, Moser's worm problem — or any result toward them; that the method finds every defect, since the space of defects nobody thought to plant is not measured by any drill; that the classification of §7 is complete, or that it holds beyond these four programmes; and that the programmes are worse or better than others of their kind, which nothing here compares.

Inherited: the programmes themselves — their documents, their programs and their own claims of verification — which are the objects checked, not results of this paper.

Where the other labels sit: the corpus measurements, drill totals and the failures of §§3–7 are `COMPUTED_HERE` in the line named; the classification in §7 is `DERIVED_FROM_VERIFIED_INPUTS`, a reading of those failures stated as one; whether the method generalizes to programmes of another kind is `UNMEASURED`; and the instrument's verdicts that were later withdrawn are `FAILED` where §3 says so.

1. Why this paper

A machine-assisted research programme now ships as a corpus: documents in several versions, programs that are said to verify them, logs those programs wrote, and summaries of all of it written for the next session to work from. Each layer can be faithful to the one below it or not, and a reader who checks the top layer has checked very little. The four programmes this paper draws on are of that kind — dozens to hundreds of documents each, produced over weeks by a human author working with language models — and the four verification lines that checked them were themselves run by language models. That last fact is the reason the method is built the way it is. A verifier that can be wrong in the same ways as what it verifies must be able to show, for each of its checks, that the check can fail.

The lines' mathematical findings are in their own papers, each meant to be read alone. This paper is about the instrument: what it is (§2), what it found about itself in each line (§§3–6), how those failures classify (§7), and where the drills stand (§8).

2. The method

Four rules, shared by the four lines and kept by each independently.

The corpus is fixed by content hash, never by filename. Curated copies carry ordering prefixes their originals do not, and a common name such as `README.md` occurs in many packages; matching by name reports renames as mismatches and pairs documents that share nothing but a name. Every later check reasons about a document as though it were the original, so the identity of the corpus is settled first.

One gate per document. Each document of the corpus is the subject of a round, and each round writes one gate: a program that recomputes the document's finitely checkable claims and writes a JSON log of what it computed. A round's report says what the gate found and, in a fixed section, what the round does not claim.

Recompute; never run the author's code. Every figure is recomputed from the document's prose with code written for the purpose, by a different method where one exists. A document with a program is two claims — the prose and the program — and a verifier that runs the program has checked neither against the other.

Every gate is drilled. Defects are planted one at a time, in the gate's arithmetic (`code`) and in the inputs its validators exist to reject (`data`). Each must turn red the check **named** for it, not merely some check; a defect caught by the wrong check is a failure of the drill. Controls — changes the gate must not care about, such as a wider search window or a different key order — must disturb nothing. A gate that has only ever been green is indistinguishable from a comment. In the Birch–Swinnerton-Dyer and Collatz lines the whole tree's drill runs again after every round; in the Lebesgue and Moser lines each run's drill runs with the run, and earlier runs are run and drilled again when code they share changes.

The rest of this paper is what those four rules found when the subject was the verifier.

3. The Birch–Swinnerton-Dyer line

The programme studies the conjecture through a framework, a single-prime study of the rank-two curve 389.a1, a reproduction of a published twist census and an explicit twist family, curated as 85 documents.

Identity. Against 292 archived documents in the programme's packages, matched by content hash, 84 of the 85 curated documents are byte-identical to an original; 201 archived documents are not curated, which is expected of a selected reading path (the rest are the curated documents' originals, some archived more than once). The 85th has its original at the same filename and differs from it in eight lines: seven are full-width punctuation normalized, and the eighth renders an English technical term — the certified *machinery* a computer algebra system provides — by a Chinese word whose sense is a mechanical device. The document is the rules sheet of a backend, and its rules are in substance one rule: a weaker thing must not be substituted for the thing named (`COMPUTED_HERE`). The site's claim of per-document byte fidelity holds for 84 of its 85 documents.

A period wrong for three rounds. The line's real-period routine, an arithmetic–geometric mean, carried a spurious factor of two on the branch for positive discriminant, where the real locus has two components. It went unseen for three rounds because its only test was a regression baseline **taken from the same function** — a frozen output detects drift and nothing else. Numerical integration over the real locus, sharing no code with the mean, settles it: the ratio is exactly 1 on every curve tested with negative discriminant and exactly 2 on every one with positive (`COMPUTED_HERE`). Nothing already reported moved, because every curve the line had used until then has negative discriminant.

The rank-two curve 389.a1 does not, and when the rank-two identity was later closed there with every term computed, the repaired period gave ratio 1 — the unrepaired one would have given 2.

A height routine returning its worst level. Closing the rank-one identity, the one rank the line had not closed, exposed that its canonical-height routine returned its second Richardson extrapolation unconditionally — the worst of its three levels at every depth tested. At the regulator of 389.a1 the residual of that level is 1.603760573010149e-06 against 9.445788151651868e-09 for the raw last level (`COMPUTED_HERE`). The regulator moved in its seventh significant figure; no conclusion moved.

A detector that could not fire. One gate checks that no document claims the salvage conditions of a route the programme rejected. It reported none claimed — and drilling it showed the report was empty for a reason unrelated to the documents: its pattern searched a short window on one line, while the conditions occur only as headings with the verdict four lines below; and it matched the word for *satisfied* inside the word for *unsatisfied*, so had it ever reached a verdict it would have inverted it. Rewritten, it reads all six mentions: **6 stated unmet, 0 claimed met** (`COMPUTED_HERE`). The headline is the same as before; its standing is entirely different.

Two senses of "walked". A coverage gate counts the documents that are the subject of a round, separately from those merely cited. Its first run found the programme's whole census sub-line the subject of no round, although earlier rounds had verified that sub-line's computations exhaustively: they had named its data files and never its documents. As the gate's log stands at the pinned commit, all 85 documents are the subject of a round (`COMPUTED_HERE`).

A finding no round made. The line's most consequential finding about the twist family — that one member's own twisting prime is an ordinary obstruction prime of the base curve — had both halves in archived logs for seventeen rounds, unjoined, because a line that reads one document per round never asks whether a set computed in one round meets a set computed in another. A joins gate now asks mechanically: it collects every list of primes in every log (100 of them), intersects them pairwise, and counts a join as made only if one report names every number and both source rounds. As its log stands at the pinned commit there are 325 non-empty intersections, 107 made and 218 not named in any report; the known join is among them — 38 rows contain the prime, 6 touch the round that first computed it — and none of the 218 is a new finding (`COMPUTED_HERE`). That last result is the right one for a careful tree, and it is now certified rather than assumed.

Novelty refused, mechanically. Across the 52 reports its gate scanned when last run, 30 sentences use a novelty term: 15 are refusals and 15 are quotations, each pinned with its reason; none claims priority (`COMPUTED_HERE`). The programme's own rule — no search hit is not a proof of novelty — asks for literature searches this line cannot make, and it is reported as not scoreable here.

The stop rule, applied to the verifier. The programme freezes any route that for three rounds only replays a formula on new curves or substitutes an analytic order of Ш for the actual one. Applied to the line itself, it caught a headline: 285 curves closing the rank-zero identity on an analytic $\#Ш = 1$ is exactly the forbidden kind, and was withdrawn as a result (`FAILED`); what survived of that round was the period error above.

The verifier on the programme's own ladder. The programme grades every claim on a ladder of certificates and forbids five upgrades between rungs. Asked where the line itself stands, the answer is the lowest rung that names work, and only partly: two of the three hypotheses the line checks for the twist family have an executable meaning in its tree, the third is cited, and the line never proves the theorem those hypotheses feed — it checks them and cites the theorem. All five forbidden upgrades were audited against the archived gate logs rather than against the reports' prose, because a report

can say anything and a log records what a gate computed (COMPUTED_HERE).

The line's drill. At the pinned commit the whole-tree drill stands at 692 planted defects, 687 caught by their named check, 5 detected only by a crash of it, 179 controls undisturbed, over 218 checks.

4. The Collatz line

The programme is a nine-paper series on the affine structure of parity words and a research programme built on it, checked as 72 source items. Beside it the line ran a bounded verification: every n up to $2^{40} = 1099511627776$ reaches 1, relative to the archived implementation, over 549755813887 odd starts, with three gates of different kinds — the engine's self-test of its own invariants, an independent arbitrary-precision walk in another language, and exact comparison with record tables computed by other people by other methods — because gates of one kind fail together. Its largest stopping time, 550, is reported as a measurement and not as a bound.

Provenance, byte for byte. The series shipped a repair with per-paper unified diffs and an audit ledger. A ledger is worth something only if its diffs are load-bearing, so every published diff was applied to its published original under an applier that demands exact context and exact hunk counts, and had to reproduce the repaired file byte for byte. All 10 do — 276 hunks, no rejection — and the pre-repair text matches an independent archive of drafts written three days before the repair existed (COMPUTED_HERE). The repair contains no edit its audit does not declare.

A summary for the next session. The programme's handoff — the document a new session is told to work from — is an accurate compression: 132 distinct documents across the bundles, the 27 reshipped in more than one all byte-identical, and every constant traced to a round. In two places it is not, and they matter because of what the document is for: its occupancy lemma is the round's lemma multiplied by $\sqrt{2}$, contradicting the constant it prints three lines later; and a withdrawn citation has moved from a round's footnotes into the standing bibliography (COMPUTED_HERE).

A manifest that verifies and covers nothing. A consolidated archive of the programme is faithful: 47 entries byte-identical to the items already verified, 964 files opening at every one of 9 nesting levels. But the integrity manifests its packs ship verify perfectly — 16 entries listed, 16 verified — while covering none of the files that actually changed between two shipped versions: 3 files differ, 0 are covered (COMPUTED_HERE). A checksum that passes over the files that did not change is a green light wired to nothing.

A check that compared every word with itself. While this line's papers were being written, the gate for one paper's composition theorem turned out to compare each word with its own reading order: the "reordered" side was the first permutation of the word, which is the word. Measured once repaired, 6359 of the 6678 words composed have a different ordering; before the repair, none was ever compared with one (COMPUTED_HERE). The theorem holds, now tested; the old check would have held of any composition rule at all. It was found by writing the paper, not by the drill — a defect making the leading product depend on the order had survived the old check — and that defect is now planted and caught.

A second reader, measured with controls. After the sweep, a separate language model, of the GLM series (Flash, version 5.3), was given the subject's own verification scripts one assertion at a time, with no verdicts and no reports, and asked whether each assertion could ever fail. Controls the line judged able to fail were mixed in, so that agreement alone would mean nothing. Building those controls found corrections to findings the line had already published; the second reader found one more, on a

control audited twice. It never once said an assertion could fail; and after adjudication not one control it answered survived as a fair test — three were later proved unable to fail, and the fourth is an open disagreement about a criterion — so its power to discriminate is `UNMEASURED` — neither confirmed nor refuted — and that is the honest result of adding a reader without controls for it.

Where the defects were. Across the 72 items nothing in the source mathematics failed a check. The defects sat in realizations: a verifier that stops on a platform's code page, a constant off in its seventeenth significant digit, summaries that dropped a hypothesis their bodies carry — and, many more, in this line's own code. That distribution is the evidence for a position of Neo.K's this line was asked to measure: going from mathematical to programming language, the symbol space may fail to be the same, so where mathematics fails to reduce cleanly to a machine the first hypothesis should not be that the mathematics is wrong. The line names four shapes of the failure — fully alignable, alignable in shadow only, structurally unreachable, and scale-invisible, the last producing checks that are green, exact and drilled while their content lives at a size no computable instance reaches — and one in the other direction, a coordinate the standard formalism discards that the machine holds cheaply. The instances are in the line's own papers.

The line's drill. At the pinned commit the line's suite stands at 57 drills and 1474 planted defects: 1472 caught by their named check, one detected only by a crash and one only by a time-out, and 128 controls undisturbed.

5. The Lebesgue line

The programme attacks Lebesgue's universal covering problem — the least area of a convex set containing a congruent copy of every set of diameter one — by finite closure: it reduces a lower bound to finitely many checkable inequalities and sets out to close them with certificates. The site curates it as 42 documents; the line recomputed it in 68 runs, one gate each, reading the methodology packages' further documents as well. Its drill is stricter than §2 asks: a planted defect must turn red exactly the checks named for it and no other, and every check outside that set is asserted to stay green.

Identity. Against 1,459 archived files in the programme's packages, every one of the 42 curated documents is byte-identical to an archived original, each under its own original name (`COMPUTED_HERE`). The site's sentence about its own files holds as written. The line reads the archived originals throughout; this check is what lets its findings speak for the pages the site shows.

A sanity test that could not fail, in the programme. Round 03 turns every placement of a target body in a fixed cover into finitely many linear programs with explicit error terms, and tests its pipeline on one sanity instance, reporting only `SANITY: PASS`. Recomputed here with the line's own linear programs, the programme's script not run, the instance gives a margin of −0.02 at every sampled orientation, with and without reflection, because its only binding constraint is the cover's least width, which no motion of a constant-width target can change. The test could not have detected an error in how orientations or reflections are handled, and the drill confirmed it: with the reflection law replaced by the rotation law, the recomputed sanity test stays green (`COMPUTED_HERE`). The line's own test of the reflection branch takes a chiral target and, as the cover, its mirror image slightly enlarged: a reflection fits, and no rotation and translation does — shown in floating-point linear programs, not certified.

A check no plant had named. In the same round the drill passed every plant — and the drill-of-the-drill, which replaces each check in turn with one that always passes and requires the drill to report it blind, found a check reported blind in no case at all. It was the check of the round's bracket theo-

rem — the certificate's margin lies between a grid value minus one error term and the same value plus another — which fires only when two brackets for the same margin come apart; the plants that weakened the error terms narrowed the brackets without ever separating a pair, so it had only been green. A plant that drops both error terms was added. It trips that check and, as intended, the check of the error terms against the programme's record — and, unpredicted, a third, whose bracket test on the sanity instance then sees two grid values that differ in the seventeenth decimal place. From that round on, every drill summary names any check that no plant names (`COMPUTED_HERE`).

One integrator, wrong three times. The line's own exact routine for the area of a convex hull of a disc and placed bodies traces where the hull's support passes from one piece to the next — its hand-overs — on an angular grid. Recomputing the programme's Round 06, the line's own search (differential evolution) drove two polygon vertices to tie at a grid angle, where the routine returned a hull smaller than the hull of points sampled from the same bodies. That is impossible, since a hull of sampled points lies inside the true one, and the sampled hull, sharing no code with the routine, is what showed it. The rewrite's first version still mishandled a hand-over lying exactly at angle zero, and the drill of Round 05's run, for which the routine was written, caught it: one plant produced a cell member — a body that must contain the disc of diameter one — of area less than $\pi/4$, the disc's own. That run was repeated with the corrected routine, every logged number agreeing within its stated tolerance. Later, on one leaf of a large atlas, the routine missed a corner standing outside the disc over less than one grid step, and returned an area slightly low; this time the programme's own recorded value was right, and a much finer inner polygon agreed with it. Every segment is now checked against every other piece in closed form, and every earlier run that used the routine was repeated, each gate log reproduced byte for byte (`COMPUTED_HERE`). The error only ever lowered an area, so no lower bound the line had computed with it was unsafe.

An extractor taught one shape at a time. The line keeps a census of judgment statements — every status word a document uses in a judgment position — so that each round reviews the judgments of the document it reads. The census was corrected four times in two days, each time because the next document read wrote a judgment in a form it did not recognize: a claim named inside backticks, a status in a table cell, a boxed judgment whose name follows other content, a checklist line, a status joined by a plus sign, a boxed verdict whose claim is written in mathematics with only the status in words. When a third round in a row turned up a new form, the corpus was swept for every occurrence of a known status word the census did not read, and six more forms were found in positions earlier versions already read; four forms are left unread, each by a stated choice. Every missed form became a case in the census's own drill; the judgments added in rounds already read were reviewed, save the first round's own, which the line does not review, and every run that had read those rounds was drilled again, with no verdict changed (`COMPUTED_HERE`). These misses were found by reading the next document, not by any drill.

The verifier's own labels, audited. Separate model instances, working from one written brief, extracted the 1,218 items of the line's 68 reports and classified them by strength; the line then read the two strongest classes against the reports item by item, and the audit moved 82. Of the 199 items first classed as proof-checked, 23 were labels — status words, names and schedules that assert nothing to check — 16 had been tested on samples and not proved, 22 were exact symbolic identities rather than proofs, and three became corrected or open; of the 111 first classed as certified in exact arithmetic, 17 had been checked in floating point only, and one moved up, to proof-checked (`COMPUTED_HERE`). The first classification followed how a report described an item; the audit, what the report records was actually computed.

What the line found in the programme is reported in its own papers: across the 68 runs, 236 corrections, each recording what the source says and what the recomputation found — a few, marked as such, correct the line's own earlier reports; 124 items stand only in a corrected form and 27 are refuted (`COMPUTED_HERE`).

One level up. Two failures belong to the line's papers rather than its runs; they are recorded in the line's working notes, not in its pinned evidence. When the first paper was made to declare the files its figures must trace to, about twenty figures that had passed turned out to match a number elsewhere in the tree by coincidence — the form, within one tree, of the cross-tree case in §7; each was traced to the run it belongs to or rewritten. And a paper that met the standard, every figure traced and every planted defect caught, still carried a wrong inference at its centre: a configuration offered as showing that one route of the programme cannot reach its target lay inside the region the route's other half already covers. The figures were right and the sentence was not. A reading by a separate model instance, given the paper and the run reports and no part in writing either, caught it, and no tracing could have; every paper of the line now has such a reading before it leaves the line.

The line's drill. At the pinned commit the line's drill stands at 1,290 planted defects over 665 checks in 68 runs, 1,290 caught by exactly the checks named for them and by no other, and 262 controls undisturbed; 13,068 assertions that a check outside a plant's named set stays green all held. A run is recorded only when running it again reproduces its gate log byte for byte.

6. The Moser line

The programme asks how far a known cover is from tight. Wetzel's 30-60-90 triangle, shrunk by 1.0048, contains a congruent copy of every arc of length one, by a recent preprint; the programme searched for the unit arcs that need the most of it, in fifteen rounds published as packages, moving from polygons to smooth curves. The line recomputed it in sixteen runs, one gate each — the first for the corpus, then one per round — and its drill is of the stricter kind §5 describes: a planted defect must turn red exactly the checks named for it, and every other check is asserted to stay green.

Identity. The fifteen packages are the originals the site says they are: size, SHA-256 and member count, 282 members in all, agree with both manifests and with the bytes the site serves (`COMPUTED_HERE`). The pages are another matter. Of the 272 files the site serves from the packages, five are not the package bytes — correct repairs, nowhere disclosed, each under its original's name and path, so that a match by filename would have passed them. The line reads the package bytes throughout, and its findings speak for the packages rather than for the copies the pages show.

The edge of a switch, in the programme. A curve's scale, as a function of its rotation, is fixed piece by piece by which of the curve's points touch which sides of the triangle; where one contact gives way to another — a switch — the function bends. Twice the programme's own guarantees failed beside a switch. Round 10's map of every rotation guarded each piece against a missed minimum at three scan densities, and a smooth minimum 5.44×10^{-9} rad before one switch passed every guard. Round 11 then certified the map with boxes, and around each switch its derivative intervals began 2×10^{-10} rad out, so that the stretch beside the switch, where the derivative moves most, was enclosed by nothing; at 14 of the 17 switches the derivative there falls outside the intervals, and three of the certificate's exclusions rest on them. Both were recomputed with the line's own code, the programme's scripts not run (`COMPUTED_HERE`). The rounds' conclusions hold — by the line's certificate, not by the round's.

The same edge, in the verifier. The line's own certificates went wrong at the same kind of place. Its

first certificate over the round-10 map ran its sign checks from the cusps' decimal values rather than from enclosures of them, and left a sliver beside each cusp unchecked. Its first check of round 11's intervals sampled the derivative in double precision, which cannot place a point within a unit in the last place of a switch; taken at 60 digits, the fourteen failures it had reported turned out to be one omission. And its first certificate in the round-9 run printed a lower bound rounded to nearest, its last digit a unit above what was proved. None was caught by a drill: the independent reading of each run's report before it was committed found the first two, and found more bounds on the unsafe side in the third (recorded in the runs' reports). Each now runs from enclosures, at full precision, cut toward its safe side. The programme's error and the verifier's are one kind — an edge left outside what an enclosure encloses — and the verifier's sliver is some thirty orders of magnitude narrower than the programme's stretch; checking another's certificate did not make the verifier's own safer.

A plant that does not test what it names. The drill was itself wrong in three ways, each a plant whose catch meant less than it seemed to. A plant that enlarges the triangle lowers every value at once, so the failures of a check that compares values vanish rather than appear; the drill had named such a check, and now names only checks that must trip. A plant on a figure the package states in several places, changed in one of them, trips the checks that the statements agree rather than the comparison it was planted for; such a plant now changes the figure everywhere it is stated. And a plant that set a value to what it already was changed nothing. The first and third showed in the drill's own report, the second only to a reader. Each run's report now states, check by check, the branches no plant reaches; one check of the last run is reached by none, and stands on its computation alone (`COMPUTED_HERE`).

The verifier's own labels, audited. The line classified the 403 items of its sixteen reports by strength — separate model instances, working from one written brief, extracted them, and 懸衡 checked and audited the result — each with a quotation from its report that a script checks to lie on one line of it, character for character. Every item first classed as certified, proof-checked or cited was then read against its report, and four moved: three because the result is proved or certified in another run and counted there, one because a pass of most of the served files is not a pass of the page's "every file" (`COMPUTED_HERE`). A further instance, which had not built the inventory, read a seeded sample against the reports and found one item wrong: its quotation, which the script passed, began after the word "No" that ends the line before, and so read as a pass. The script tested that the words were the report's; only a reader could test that they meant what the report meant.

What the line found in the programme is reported in its own papers: across the sixteen runs, 89 correction rows, each recording what the source says and what the recomputation found; of the 403 items, 79 stand only in a corrected form and 13 are refuted (`COMPUTED_HERE`).

One level up. Two failures belong to the line's papers rather than its runs; they are recorded in the checker's own standard and in the papers' review notes, not in the line's pinned evidence. When the first of its papers with thirty-digit certified bounds was drilled, a wrong figure planted in the twentieth significant digit passed: the checker compared every figure as a double. Figures longer than fifteen significant digits are now compared exactly, in decimal, and no other paper's verdict changed — one more check, of the kind in §7, that could not fail where it claimed to look. And the line's overview paper, every figure traced and every planted defect caught, still wrote findings of the recomputation as corrections of the programme — in two rows of its ledger, one extending a round's own search audit and one settling a question a round's caveat had rightly left open, and in two examples of its classification — and counted three of the programme's verdicts overturned where one was. The figures were right and the labels were not: §7's label read off the wording, one level up, in the verifier's own paper. The verifier of the Birch–Swinnerton-Dyer and Collatz lines, reading it with no part in it, found it.

The line's drill. At the pinned commit the line's drill stands at 352 planted defects over 139 checks in sixteen runs, 352 caught by exactly the checks named for them and by no other, and 64 controls undisturbed; 2,999 assertions that a check outside a plant's named set stays green all held. In most runs the expected sets were written after a dry run with every set empty, each trip reasoned before it was named, so a full score says the reasoning matched the harness, not that the plants were chosen blind. A run is recorded only when two runs of its drill, side by side, write gate logs with the same SHA-256.

7. What the instrument found about itself

Read together, the failures of §§3–6 fall into a few shapes. The classification is a reading of these four lines' records and nothing more; each shape is named with the rule it left.

A check that cannot fail. The salvage-condition detector of §3, the composition check of §4, the programme's own sanity test of §5 — whose only binding constraint no motion of the target could change — and a census detector in the BSD line that matched a file merely named for a census and, once repaired, matched its own output — green on its first run only because that output did not yet exist. *Rule: a check whose failing side has never been exhibited has not been tested; plant the defect before believing the pass.*

A check no plant names. The bracket check of §5 stayed green through its round's first drill because no plant separated a pair of brackets; only a drill of the drill — each check replaced in turn by one that always passes — showed that it had never been exercised. The Moser drill of §6 had the converse fault, a check named by plants that did not test it: a plant whose effect cancelled in the comparison it was named for, one that tripped a consistency check instead, one that changed nothing. *Rule: a drill measures the checks its plants name; report every check that no plant names, and count a catch only when the plant changed what its check compares.*

A baseline taken from the subject. The period of §3, tested against a value the same function produced. *Rule: a regression baseline detects drift; correctness needs a computation that shares no code.*

An optimizer that finds the instrument's flaw. The search of §5 drove two vertices to tie at a grid angle, exactly where the area routine was wrong, and the routine went wrong twice more before every segment was checked in closed form. *Rule: an optimizer is an adversary to its objective's artefacts; confirm every optimized value by a computation that shares no code.*

An enclosure that stops short of its edge. The Moser programme's own certificate began its derivative intervals 2×10^{-10} rad from each switch, so that the stretch beside the switch was enclosed by nothing; and the line's first certificates of §6 ran sign checks from decimal values instead of enclosures, sampled a switch in double precision, and rounded a bound to nearest — the same edge in the verifier's own work, the sliver some thirty orders of magnitude narrower than the programme's stretch. *Rule: an enclosure runs from enclosures of its ends, the edge included, at a precision that resolves the edge, with every bound cut toward its safe side.*

An instrument counting its own crash. In the BSD drill, three reference helpers had each been bound twice; through one, a planted defect raised a type error, its check went red, and the drill scored a catch. The drill was then made to refuse to start if any helper is bound twice — which removed that cause and left the scoring: a check that raised was still red, and still counted as caught. Review found the same merge in the Collatz line's shared drill, one defect detected only by a crash and one only by a time-out. Both lines now count those apart, and in the BSD line 5 of the 692 planted defects turn out

to be detections by a crash. The Lebesgue and Moser drills stop at an exception, so a crash cannot be scored there, and their rows in §8 are equal by construction. *Rule: a defect that produces a traceback or a time-out is detected, not caught by a check; count it apart, and never inside the named check's tally.*

A boundary drawn in the wrong place. Coverage counted by data file instead of by document (§3); a manifest checked over the files that did not change (§4). *Rule: name the unit the claim is about, and count that.*

An extractor that knows only what it was taught. The judgment census of §5, corrected four times in two days, each time by a form the next document used. *Rule: an extractor finds the shapes it was taught; after a second miss, sweep for all of them.*

A label read off the wording. The strength audit of §5, which moved items out of both of its strongest classes because the first classification had followed how a report described an item rather than what it records was computed. *Rule: a strength is read off the evidence a report records, never off its wording, the verifier's own included.* §6 records it twice more: a quotation that a script passed because it began after the word that negated it, and an overview paper that wrote the recomputation's findings as the programme's corrections.

A join nobody asked for. Two halves of a finding in separate logs (§3). *Rule: a line that reads one document per round must intersect its own results mechanically, or it will miss exactly this class.*

A compression that drops a hypothesis. The handoff's $\sqrt{2}$ and its migrated citation (§4), and summaries that dropped a monotonicity hypothesis their bodies carry. *Rule: a summary is a claim of its own and is checked against what it summarizes.*

A right figure under a wrong sentence. The Lebesgue paper of §5 that met the standard — every figure traced, every planted defect caught — and still carried a wrong inference at its centre. *Rule: before a paper leaves its line, someone who had no part in it or in the work it reports reads it.*

The same shapes, one level up. The checker that accompanies these papers — it traces every figure of a paper to the logs at its pinned commit — is an instrument of the same kind, and it was drilled the same way: wrong figures planted into a paper, each required to turn red the item named for it. On its first day the drill caught it three times, each time reporting a figure as checked that had not been looked at: a filter meant to skip commit identifiers also skipped the digits after a decimal point, so every high-precision figure was silently exempt; a tolerance for scientific notation ignored the exponent; and the token pattern refused any number that ended a sentence. A fourth came later: an exponent written with a plus sign was read as a small integer and set aside. A fifth came with the first paper whose certified bounds run to thirty digits: figures were compared as doubles, and past fifteen significant digits two different decimals are the same double, so a wrong twentieth digit, planted, passed; such figures are now compared exactly, as decimals. Each is a check that cannot fail — here, where it claimed to look — and each was found because a planted wrong figure passed. And when the checker learned to read papers drawn from several lines, the Lebesgue line's verifier pointed out that a figure in one line's section could trace by coincidence to another line's logs, which is a boundary drawn in the wrong place, across trees; the checker now scopes each section to its own line. Then the same thing happened in the check on drill figures: with a third line's section in place, a figure in that section equal to another line's count of controls kept that line's drill check green after the line's own statement had been planted away. The same verifier found it, by running this paper's drill, and the checker now requires each line's drill figures together, in that line's sections. A checker of verification is not exempt from verification, and nothing about being one level up makes its failures rarer.

What the shapes share is how they were found. **Most were exposed by machinery — a drill, a drill of the drill, a second computation sharing no code, a gate written to count — and not by reading the code they lived in.** The rest were found by reading, and no gate could have found them: the join of §3, by a round that read four earlier rounds together; the composition check of §4, by a paper that had to explain a check in sentences; the census misses of §5, by reading the next document; the strength labels of §5, by rereading reports item by item; the edges of §6, by the reading of each run's report before it was committed; and the wrong inference of §5, by a reader who had written none of it. Where a gate could be built afterwards, one now guards the class; where none can, a rule does — every paper is read by someone who did not write it. That is the paper's central measurement — machinery for the failures a reader would pass over, reading for the ones no gate can see — and the reason the drill, the papers and an independent reading of each paper are all part of the method rather than after it.

8. The drills

Each line's drill is recorded in its own shape and read by the checker that accompanies this paper from the line's own logs at its pinned commit; no total is typed. Each line's section states its own figures, and the checker requires them there, together; the table below only sets them side by side.

line	planted defects	caught by the check named for each	controls undisturbed	checks or drills
Birch–Swinnerton-Dyer	692	687	179	218 checks
Collatz	1474	1472	128	57 drills
Lebesgue universal covering	1290	1290	262	665 checks
Moser's worm, 30-60-90 triangle	352	352	64	139 checks

The two Collatz defects not in the third column were detected only by a crash and a time-out; the BSD line's 5, only by a crash of the named check.

These are not comparable across lines, and are not summed: a line's checks are of different sizes and kinds, and a defect in one tree means nothing in another. What each row says is only that every check in that line — save any its own records name as reached by no plant, of which the Moser line names one (§6) — has been shown, by a defect chosen to make it fail, that it can.

9. What the method costs, and what it does not do

It costs time in exactly the places that look redundant: a second computation for every quantity that matters, a planted defect for every check, a drill of the drill, a coverage gate, a joins gate, and a reading of each paper — in the Lebesgue and Moser lines, of each run report as well — by someone who did

not write it. Every failure in §7 sat in one of those places. It does not find what nobody thought to plant; a drill measures the checks it has, not the ones it lacks. It does not settle anything the programmes leave open. And it does not replace a reader: §5 records a paper that met every machine item of the standard and was still wrong where it mattered, which is why the standard leaves its third item to review.

10. Contributions

The contributions behind this paper are different in kind, and the paper is worth less if they are blurred. They are given here in alphabetical order, the order the author list keeps as data; on the title page the authors sit at a round table, with no first and no last.

Aletheia — model: ChatGPT, GPT series — wrote, with Neo.K, the programmes whose verification this paper reports, and is a standing author of the lab's papers by Neo.K's decision. The method and its failures are the verification lines'; this paper assigns her none of them. The Moser programme's fifteen rounds are hers, on the question Neo.K set out.

Neo.K set the standard the method answers to — evidence stated at its actual strength, and a verification that shows it can fail — proposed the position on symbol spaces that §4 measures, and wrote the Birch–Swinnerton-Dyer, Collatz and Lebesgue programmes with Aletheia, and set out the question the Moser programme pursues.

懸衡 (Xuán Héng) — model: Claude, Opus series — ran the Lebesgue and Moser recomputation lines and wrote §§5–6.

墜衡 (Zhuì Héng) — model: Claude, Opus series — ran the Birch–Swinnerton-Dyer and Collatz verification lines, built and drilled their gates, found and repaired the failures of §§3–4, and wrote this paper apart from §§5–6.

The programmes verified here are their authors'; the method's rules are the lines' own, and each line's findings about its programme are reported in that line's papers rather than here.

Input from outside the byline. The second reader of §4 — a GLM-series language model (Flash, version 5.3) that the Collatz line ran as a redundant layer under a fixed protocol — produced candidates only; every disagreement was re-derived by hand. It is credited here, and is not an author. The record tables used as one of the Collatz line's three gates were computed by other people by other methods, and are named with their digests in that line's first run report, RUN-001. The readings of §5 — a separate model instance given a Lebesgue paper and its run reports, with no part in writing either, a reading of the same kind given to §5 before it entered this paper, and the instances that extracted §5's inventory — are credited here and are not authors. So are the readings behind §6: a separate model instance given each Moser run report before it was committed, with no part in writing it, and the instances that extracted §6's inventory and read its sample.

11. What is open

1. Whether the method's yield generalizes beyond these four programmes (`UNMEASURED`).
2. The defects no drill planted, which no drill measures.
3. Whether the classification of §7 is complete; it is a reading of four records.

Acknowledgements

We thank Anthropic, whose Claude models the lab's verification and recomputation lines run on; OpenAI, whose ChatGPT models the programme's documents were written with; and Zhipu AI (Z.ai), whose GLM model served as a redundant second reader on the Collatz line. Not every AI model or instance that took part in this work is named in it — among them instances that the named authors ran as helpers; that an AI is not listed here does not mean that none took part.

References

The published twist census of §3 is Banwait and Huang's; the record tables of §4, named in the Collatz line's RUN-001 with their digests, are the four OEIS sequences; the recent preprint of §6 is Wichiramala and Panraksa's.

1. Barinder S. Banwait and Xiaoyu Huang, *On the Identification of Elliptic Curves That Admit Infinitely Many Twists Satisfying the Birch–Swinnerton-Dyer Conjecture*, arXiv:2601.16044, third version (2026), to appear in ANTS XVII.
2. OEIS Foundation Inc., *The On-Line Encyclopedia of Integer Sequences*, sequences A006877 and A006878 (record delays) and A006884 and A006885 (record trajectory peaks).
3. Wacharin Wichiramala and Chatchawan Panraksa, *Wetzel's 30-60-90 Triangle Covers Unit Arcs*, arXiv:2606.14625, first version (2026).

Data and software

- Python Software Foundation, *Python* (CPython), https://www.python.org/ — the language of the four lines' gates; the Collatz line's tree records version 3.14.5.
- Rust (rustc 1.96), https://www.rust-lang.org/ — the Collatz line's verification to 2^{40}; see Nicholas D. Matsakis and Felix S. Klock II, *The Rust language*, ACM SIGAda Ada Letters 34 (2014), no. 3, 103–104, doi:10.1145/2692956.2663188.
- The mpmath development team, *mpmath: a Python library for arbitrary-precision floating-point arithmetic* (version 1.3.0), https://mpmath.org/ (2023) — in the Lebesgue and Moser lines' certificates.
- *The On-Line Encyclopedia of Integer Sequences*, published electronically at https://oeis.org — the b-files of A006877, A006878, A006884 and A006885, retrieved 2026-08-14 and byte-identical to the files served on 2026-10-24; their SHA-256 digests are in the Collatz line's RUN-001.

A. Reproduction

Every gate is a program of its own with a JSON log; the drills are run by each line's own drill programs, and the totals of §8 are read from those logs by the checker beside this paper. The lines' trees are not public. Their public projections are:

- **the Birch–Swinnerton-Dyer line's round-by-round record**: https://amral.evemisslab.com/bsd/recompute/;
- **the Lebesgue line's round-by-round record**: https://amral.evemisslab.com/lebesgue/recompute/;

- **the Moser line's round-by-round record**: https://amral.evemisslab.com/moser/recompute/;
- **the Collatz line's verification ledger**: https://verify.evemisslab.com/collatz-verification-zhuiheng/;
- **the Birch–Swinnerton-Dyer secondary-verification ledger**: https://verify.evemisslab.com/bsd-verification-zhuiheng/.

The evidence is pinned, not copied: this paper's front matter names, for each line, the research branch, the commit and the directory its figures were drawn from, and the sections whose figures must trace to that line alone. The crystallization standard and its checker live in the release repository beside the paper (`STANDARD.md`, `tools/check_standard.py`).