

A finite-host Sidorenko counterexample: audit and formal verification

A version-bounded audit of OpenAI/math Result 161

Corporate author: Haivian Technology

Commissioning owner: Richard Janutka, Qonduit Limited NZ

Audit completed: 16 October 2026 UTC Published: 2026-10-17

Version: HAIVIAN-SIDORENKO-FINITE-HOST-VALIDATION-V1-20261017

Abstract. A Haivian audit supports the submitted finite-host counterexample to Sidorenko's conjecture. The cumulative Lean execution compiled 31 retained and four newly completed solution modules, four inspected closures used only the declared permitted axioms, and the unchanged author Comparator accepted the finite-host endpoint with Lean's default kernel. The report preserves the scheduling and earlier-attempt history and states the limits on forcing, host size, novelty and toolchain trust.

What this audit established

Our conventional mathematical review supports the submitted finite-host counterexample to Sidorenko's conjecture. We found no concrete mathematical error requiring a correction in that argument. Actual Lean execution also compiled all 35 supplied solution modules and checked four theorem dependency closures, finding only the permitted axioms `Classical.choice`, `Quot.sound` and `propext`.

The supplied finite-host formal-verification workflow subsequently completed successfully in a separately authorized bounded allocation. The genuine author Comparator built and exported the challenge and solution, accepted the solution with Lean's default kernel, and returned its final acceptance message with exit code zero. Normal integrity checks also completed. The owner assessment is therefore positive for the precise finite-host statement under the disclosed execution trust base—not for every claim in the manuscript, an explicit small host, or a second independent-kernel verification. Earlier incomplete attempts remain part of the record.

The precise mathematical claim

The pattern H is a specified bipartite incidence graph: 22 triples on 13 points give 35 pattern vertices and 66 edges. The claim is that some finite simple graph G , with at least one edge, satisfies

$$\text{hom}(H, G) / |V(G)|^{35} < (2|E(G)| / |V(G)|^2)^{66}$$

Here hom counts all edge-preserving vertex maps, including noninjective maps. The 35 vertices belong to the pattern H , not to the existentially obtained host G . The result does not provide a 35-vertex host, a small evaluated adjacency table, a numerical gap or a minimality theorem. Its host edge density uses $2e/n^2$, not $e/\text{binomial}(n, 2)$.

We froze [OpenAI/math at commit fd4aeeb2ee4fc729c18d98444fed42fd0529eeeb](#), preserving the original manuscript, proof sources, challenge and author configuration. The audit included the 51-page manuscript and the 35-module, 1,216,768-byte solution closure. A bounded comparison with a subsequently observed repository revision found the relevant manuscript and proof directories unchanged; that is not a claim that all later discussion or criticism was searched exhaustively.

Where the mathematical scrutiny concentrated

The argument first obtains a strict deficit for a finite weighted kernel and then transfers it to a finite simple graph. Several steps require more than a numerical check of the displayed triple table.

The review checked that an orbit estimate controls each individual orbit, not merely the aggregate mass of a dimension profile. It examined the compatibility constraint on dependent cyclic forms rather than treating those forms as independent. It rederived the weighted singular-tail inequalities and checked the order of the dimension and field-size choices. In particular, the displayed requirement D greater than 1641 is only one late requirement, not a sufficient dimension for the entire construction.

Pinning was checked for arbitrary maps, not just permutations or injections. The dilution argument uses different powers from the two sides of the incidence graph, and symmetrization must preserve a strict deficit. The final sampling argument controls collisions of source vertices and applies Jensen's inequality in the direction needed to obtain a finite host; it does not assume that repeated host edges are independent trials.

A separately contextualized reviewer examined these arguments and executed distinct bounded checks. These included all 22 triples, six exposure orders, 33 support identities and 66 activation overlap moments; 628 cyclic-form cases; 21,952 small-field span assignments; and 32,027 exact rational slack/coercivity tuples. Negative controls exposed the difference between individual-orbit and profile mass, the failure of independent cyclic-form assumptions, and the possibility of negative local slack. These computations corroborate specific steps. They neither instantiate the complete large-parameter counterexample nor replace the general proofs.

The manuscript's forcing consequence was considered separately as a conventional deduction conditional on the main counterexample. It concerns one density, not every density. The supplied formal challenge and author Comparator cover the finite-host Sidorenko endpoint only. No formal validation of the forcing consequence is claimed here.

What actually ran

The first formal run compiled 31 solution modules, then Lean aborted while creating a thread in Refinement. That was a native runtime failure, not a reported false theorem or a proof/type error. The original failure-time task counters were unavailable and were not reconstructed.

One bounded diagnostic under the unchanged limits reproduced the abort. Its contemporaneous samples showed 63 Lean threads plus the wrapper reaching the 64-task ceiling, with four task denials. It recorded no memory-limit or out-of-memory kill event. This establishes task-limit contact in that diagnostic; its direct process topology was not identical to the first Lake invocation.

A separately reviewed, one-time continuation added `weakLeanArgs := #["-DElab.async=false"]` to the audit-generated Lake configuration. The change reduced asynchronous scheduling; it did not alter the submitted proofs, challenge or author JSON, add axioms, raise the resource limits, fabricate build metadata or reset the original clock. The previous configuration bytes were preserved. Genuine Lake completed Refinement, Pinning, Symmetrization and RandomHost using the retained authentic build and dependency state.

Thus the cumulative coverage is 31 retained compilations plus four newly compiled modules, not 35 fresh compilations under one scheduling configuration. The actual final theorem and axiom diagnostic succeeded. Its output states the finite-host existential conclusion above, and the closures of `main`, `main_proved`, `exists_symmetric_kernel` and `exists_oriented_kernel` each list exactly the three permitted axioms.

The execution used a retained Lean 4.34.1 compiler and Mathlib revision `d13f23b723b8a846827a245b89c10fc7d3f11612`, with identity-checked dependency cache objects and genuine Comparator/exporter tools bound to the original tool-build receipt. These reused components form part of the execution trust base. This case did not perform a full dependency source rebuild, a diverse-compiler check or a second independent-kernel check. File-integrity hashes establish preservation, not the soundness of the compiler or cached proofs.

The first genuine Comparator attempt ran against the unchanged author configuration in the same project. It recorded a challenge build, challenge export and solution build. At 16:03:17 UTC, the conservative 562-second watchdog lifetime expired and stopped the owned job, before solution-export completion, final kernel acceptance or final acceptance was observed. The original scientific cutoff was 16:03:27 UTC and was not extended. The contemporaneous incomplete-outcome report remains preserved; later success does not rewrite it.

Richard then expressly approved one new allocation of at most 60 minutes on the existing VPS. Its new clock began at 19:09:59.999313 UTC on 16 October 2026, including staging, setup, gating and waiting. It retained all 35 compiled modules, the accepted synchronous-elaboration setting, the original sources and author JSON, and the same resource and protected-service limits. It did not copy a synthetic success state or skip directly to a kernel result. The unchanged genuine Comparator reran its normal cached Lake builds, exports, comparison and kernel path.

This time the actual output records both challenge and solution exports, followed by default-kernel acceptance and the final acceptance message. The CLI returned zero after 430.767 seconds. Its normal source, output, tool, runtime, private-cache, directory and retained-prefix checks completed before the runner accepted the result at 19:21:27.923742 UTC. Total new-allocation elapsed time was 687.924 seconds, including setup and gating—well within the authorized limit. Nanoda remained disabled by the supplied author configuration, and no second independent kernel is claimed. The challenge's intentional sorry is its specification placeholder, not an additional axiom in the inspected solution closures.

Integrity, limitations and disclosure

The successful allocation's terminal collection contained 53 returned raw files totaling 3,649,211 bytes. The owner rehashed these and the receipt's 80 linked evidence files without mismatch. Operator preservation checks covered 773 retained pins and all 35 compiled-object identities. The original dependency cache and existing private-cache nonmetadata files remained unchanged; five existing generated hash changes and one generated lockfile hash were retained as ordinary metadata history, not described as a wholly unchanged private cache. The accepted scheduling configuration and its preserved predecessor remained explicit.

Unlike the interrupted first attempt, the successful allocation completed its normal post-Comparator integrity checks before acceptance. Protected-service state and four HTTPS content checks matched the frozen baseline, with no owned processes left and no production mutations reported. The watchdog's main process exited zero without an active guard stop. A terminal cgroup-read race and a cleanup command returning a nonzero status because the successful unit had already unloaded were retained and disclosed; the infrastructure record is not described as flawless.

The final conclusion combines conventional scrutiny, bounded independent computations, actual solution compilation and axiom inspection, statement correspondence, and completed author-Comparator/default-kernel execution. It validates the supplied finite-host endpoint within the declared trust assumptions. It does not certify novelty or minimality, formally validate forcing, eliminate the trusted compiler/cache assumptions, or rule out every possible error in our audit. Hash equality establishes identity and preservation; it is not an independent mathematical proof.

Richard Janutka of Qonduit Limited, New Zealand, commissioned this audit. Haivian Technology controls the Haivian Validation Journal. Scientific and distinct review used separately contextualized AI reviewers from the same provider; this was not external human peer review, independent institutional accreditation or OpenAI endorsement. This report is original Haivian analysis, not republication or relicensing of the authors' paper, proof bundle or logos.

Primary sources

[Frozen repository edition](#)

[Original manuscript](#)

[Final finite-host theorem source](#)

[Author Comparator configuration](#)

Rights and disclosure

Copyright © 2026 Haivian Technology. Public article publication is authorized. Underlying private audit evidence, code, third-party manuscript assets and source bundles are not released by this publication.

This is same-provider AI review within an audit and journal operation controlled by Richard Janutka / Qonduit Limited NZ. It is not external human peer review, independent-provider verification, OpenAI endorsement or authorship of the original proof. [Canonical article](#).