

Evidence-Carrying Claims: Fail-Closed Verification Gates for Language-Model-Assisted Research as Knowledge Base Construction

Robert Sneiderman

Independent

robbysneiderman@gmail.com

Abstract

Language-model-assisted pipelines emit theorems, experiment reports, and literature claims faster than any reviewer pool can check them. This paper argues that the knowledge base such pipelines demand is not a store of extracted facts but a *claim ledger*: a knowledge base whose entries pair each claim with the artifact that verifies it (a machine-checked proof, a rerunnable receipt, or a resolved forecast) and whose gates *fail closed*, blocking an unverified claim from entering the record rather than scoring it after the fact. We ground the argument in three mechanisms already running in one research operation: a twenty-two-gate manuscript-admission stack, a Lean 4 library in which the claim is the artifact, and a sealed-prediction registry with forced resolution. We then state what a community-scale claim ledger would require of its schema, its adjudication protocol, and its incentives. The evidence is a single operation and does not establish generalization; the contribution is the knowledge-base-construction framing and the fail-closed gate architecture.

1 Introduction

An end-to-end research pipeline built around language models can now produce a paper-shaped object, with a plausible theorem, a results table, and a related-work section, in hours (Lu et al., 2024). In a controlled comparison, expert reviewers rated ideas from such pipelines as more novel than ideas from human researchers (Si et al., 2024). The binding constraint on machine-assisted science has moved: generating candidate claims is cheap, and deciding which of them deserve trust is not.

The tools built so far for that decision run downstream of the claim. Fact-extraction and claim-verification datasets score an already-written statement against retrieved evidence (Thorne et al., 2018; Wadden et al., 2020; Guo et al., 2022).

Hallucination detectors estimate, probabilistically, whether a generated statement is confabulated (Farquhar et al., 2024; Ji et al., 2023). Both attach a judgment to a claim that is already in circulation. For human-scale literatures this is tolerable: review and replication eventually catch part of what slips through, at a measurable cost (Ioannidis, 2005). For pipelines that emit claims at machine speed, post hoc scoring loses the race by construction. The scored claim has already entered the record, and the score itself is an estimate.

This paper argues for moving the verification boundary upstream and treating what results as a knowledge-base-construction problem. The object to build is a claim ledger: a knowledge base whose atomic entry is a claim bound to a verification artifact and a checker verdict, and whose population step is a gate that fails closed (Figure 1). Section 2 places the proposal against post hoc verification and against the structured scientific-assertion models that already exist. Section 3 defines the entry types and the gate semantics. Section 4 describes three mechanisms that run today, at manuscript, theorem, and forecast granularity, inside our own research operation. Section 5 sketches the community-scale version, and Section 6 states what the evidence here cannot support. Two terms recur. Lean 4 is a proof assistant whose small trusted kernel re-checks every proof term and reports the axioms a proof relies on (de Moura and Ullrich, 2021). A Brier score is the mean squared difference between a probability forecast and the 0/1 outcome: 0 is perfect, and always saying 0.5 scores 0.25 (Brier, 1950).

Our contributions:

1. A framing: the knowledge base that language-model-assisted science needs stores claims bound to verification artifacts, and it is populated by fail-closed gates at generation time, not by extraction from published text.

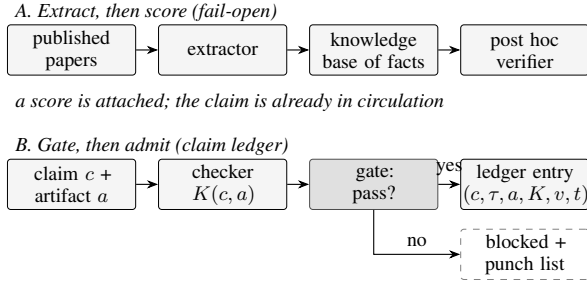


Figure 1: The proposal in one contrast. Extraction-based construction verifies a claim after it exists and scores it; a claim ledger admits a claim only when its checker returns a verdict, and a missing or unrunnable artifact blocks the entry with a typed reason instead of a score.

2. Existence evidence: three working mechanisms from one research operation that instantiate the ledger at three granularities, described concretely enough to reconstruct the design, with three live entries shown in full (Table 1).
3. A sketch of the community-scale infrastructure: an entry schema with checker-typed evidence links, an adjudication protocol for contested or decayed artifacts, and incentive placement at venues.

Non-claims. No controlled evidence is offered that these mechanisms generalize beyond the operation that built them. Each component composes existing ideas: proof kernels, preregistration, and the fail-closed discipline of continuous integration; the claimed contribution is the composition and the knowledge-base framing. This is not a calibration study, and the release contains no dataset, no benchmark, and no deployed community system.

2 Related work

Fact verification datasets (Thorne et al., 2018; Wadden et al., 2020) and the automated fact-checking literature (Guo et al., 2022) verify statements against retrieved text after the fact; the ledger differs in when verification happens (admission rather than post hoc) and in what counts as evidence (a runnable artifact rather than a supporting passage). Hallucination detection (Farquhar et al., 2024; Ji et al., 2023) is complementary: a detector can triage which generated claims to demand artifacts for, but its output is a score, so it cannot serve as a fail-closed gate. Process supervision (Lightman et al., 2024; Uesato et al., 2022) verifies reasoning steps rather than outcomes; it is gate-

shaped but applied to model training, whereas the ledger applies the same instinct to publication artifacts. Proof assistants and their libraries (de Moura and Ullrich, 2021; The mathlib Community, 2020) show that the formal layer already operates at scale under fail-closed discipline, since a broken proof blocks integration; we read mathlib as the largest existing claim ledger. The meta-science literature diagnosed the reliability problem (Ioannidis, 2005) and built the instruments (Nosek et al., 2018; Tetlock, 2005); the ledger proposal operationalizes those instruments as admission control and gives them a knowledge-base home.

Structured representations of scientific assertions are older than this proposal and supply its vocabulary. Semantic publishing (Shotton, 2009) asked for machine-readable claims inside articles; nanopublications (Groth et al., 2010) fix the smallest publishable unit as an assertion plus its provenance plus its publication information; micropublications (Clark et al., 2014) model claims, evidence, and arguments as a graph; the Open Provenance Model (Moreau et al., 2011) standardizes the derivation records those graphs carry; and research objects (Bechhofer et al., 2013) bundle the data, code, and workflow a claim depends on. The ledger takes the split between assertion and provenance from this line and adds what it leaves open. In each of these models an assertion can be minted with a citation, or nothing, as its support, and the provenance record describes how the assertion was made rather than whether a third party can re-establish it. The ledger types every evidence edge by an executable checker, refuses an entry whose checker cannot run, and admits forecasts whose truth value has a scheduled arrival. That is the whole delta beyond the fail-closed admission mechanism, and we state it narrowly on purpose.

3 Claim ledgers and fail-closed gates

Fix a research operation that emits claims. A *claim ledger* is a knowledge base whose atomic entry is a tuple

$$e = (c, \tau, a, K, v, t),$$

where c is the claim text, τ is its type, a is a verification artifact, K is a checker (a procedure a third party can run on the pair (c, a)), $v = K(c, a)$ is the verdict, and t is the admission timestamp. Three types cover the claims a research pipeline emits:

- **Formal** ($\tau = \text{formal}$): the artifact is a machine-checkable proof; the checker is a proof kernel;

the verdict is exact and includes the axioms the proof assumes.

- **Empirical** ($\tau = \text{empirical}$): the artifact is a rerun receipt: pinned environment, inputs, script, and output hash; the checker reruns and compares within a stated tolerance.
- **Forecast** ($\tau = \text{forecast}$): the claim concerns a future outcome; the artifact is a probability, a horizon, and a resolution criterion, all sealed before the outcome exists; the checker is the resolution protocol, and the verdict arrives at the horizon.

Checkers are ordered by trust, not by weight.

The three checkers answer different questions and do not sit on one evidential scale. A kernel settles a formal statement exactly; a rerun settles whether an artifact reproduces its recorded output; a resolution protocol settles whether a sealed forecast came true. What the ledger does order is how much a reader must take on trust to accept the verdict: nothing beyond the kernel for a proof, the pinned environment and the stated tolerance for a rerun, the honesty of the seal timestamp for a forecast, and the citing author for a free-text citation. An evidence edge carries its checker, so a reader never has to guess which kind of support a claim has, and a proof of a lemma nobody uses does not outrank a rerun of a result everybody uses.

Checkability is not validity. A rerun receipt certifies that the artifact reproduces its output; it does not certify the sentence a paper builds on that output. The ledger handles the gap by scope. The claim text of an empirical entry is the statement the checker settles: this script, on these inputs, yields this table. A wider statement, that the effect generalizes or that the difference is causal, is a separate claim that must carry its own artifact, in practice a sealed forecast of a replication with a resolution criterion, or else enter the visible WARN tier described below. Reproducibility is a floor the gate can enforce, not the validity the reader wants, and the ledger records which of the two a given edge supplies.

Gate semantics. A gate is a predicate on entries evaluated at admission. The gate *fails closed* when its default is denial: a claim whose artifact is missing, stale, or unrunnable is blocked, and the burden of producing a checkable artifact sits with the claimant. Post hoc scoring is the fail-

open alternative: the claim circulates and carries a score. Fail-closed admission is standard practice for code, where a failing test blocks a merge, and close to nonexistent for scientific claims.

Why this is knowledge base construction. Population, in this design, is not extraction: entries are emitted by the gates themselves at the moment a claim passes, so the ledger is complete for its operation by construction. Curation is the adjudication of disputed or decayed artifacts. Temporal scope is carried by the forecast layer, whose entries have truth values with scheduled arrival times. A minimal entry serializes in a dozen lines; this one is the forecast entry for the present paper:

```
{ "claim": "AKBC 2026 accepts this paper",
  "type": "forecast",
  "artifact": { "ref": "seals/akbc-vision",
    "hash": "sha256:4881d859b7e6",
    "p": 0.55 },
  "checker": "resolution",
  "verdict": "pass",
  "sealed": "2026-07-24T14:59Z",
  "resolves": "2026-09-15" }
```

4 Three mechanisms in operation

The design above is not hypothetical. Our research operation, which submits mathematics, measurement, and systems papers to open venues, runs a version of each layer. What each mechanism checks is described below; every specific check named exists in the operation's source as of the day of writing, and Appendix B lists the full gate set. The point is existence, not scale.

4.1 A manuscript gate stack that fails closed

Every manuscript leaves the operation through a single gate runner that applies twenty-two gates and emits one verdict from a four-token set; one token permits submission, and the other three block with a punch list. The gates are deterministic wherever possible. A citation gate parses every bibliography entry, resolves it against the arXiv API, Crossref, and DBLP, and returns a typed per-entry verdict (PASS, FAIL_MISMATCH, FAIL_NOT_FOUND, WARN_UNVERIFIABLE); any FAIL blocks. A metrics gate recounts quantitative claims about our formal library (Section 4.2) against the live source tree, so a stale module or theorem count cannot ship. An exact-support gate requires that a citation attached to a key claim support that exact claim rather than a weaker neighbor, and a hedge gate requires every hedged key step to carry a discharge

Type	Claim	Artifact	Checker	Verdict
formal	The sharp McDiarmid inequality for i.i.d. inputs with constant width (<code>sharp_mcdiarmid_inequality_iid_const_width</code>).	Proof term in the public library at commit 5e41e98.	Lean 4 kernel, run by the repository’s continuous integration.	pass; CI green at that commit.
empirical	A deployed health-information assistant’s release gate, on a fixed 107-probe regression set, refused 23 probes after hardening versus 55 before, with no released answer failing the sentence-level support check in either run.	Two stored evaluation artifacts (SHA-256 prefixes e9503721c3e1, 19f6da02b9d1) and a 40-line recount script.	Rerun of the recount over the stored artifacts.	pass on 2026-09-06; artifacts are operation-internal, so a public ledger would hold this entry in the WARN tier.
forecast	“AKBC 2026 vision track accepts this paper by 2026-09-15.”	$p = 0.55$, sealed 2026-07-24T14:59Z, horizon 2026-09-15, criterion: the venue decision.	Resolution protocol: the decision, recorded against the seal.	hit; resolved 2026-09-01 (decision dated 2026-08-31).

Table 1: Three live entries from the operation’s ledger, one per type, shown in the schema of Section 3. The empirical row is a claim about a deployed system, not about mathematics; its artifacts are not public, which is exactly the case the WARN tier exists for.

obligation; an open obligation blocks. A blind-policy gate compares the byline to the venue’s verified anonymity policy and fails closed when no verified policy is on record. Deterministic parses of the built PDF block blank pages, margin bleed, bitmap fonts, template-default strings, and a count printed under two denominators. A final gate refuses the ready verdict unless a sealed prediction of the submission’s outcome is already on file (Section 4.3).

Two properties matter for the ledger argument. The stack rejects in practice, and its rejections are typed and actionable: a blocked manuscript receives the failing gate, the offending object, and a one-line fix, not a score. The gate set also accretes by post-mortem: each gate below that traces to an incident was added the day the defect was found, so that class of failure is now unreachable. A fail-closed gate set maintained this way is a knowledge artifact in its own right: it encodes, as executable checks, the ways the operation has been caught being wrong.

What the stack has recorded. The operation’s machine-readable submitted set lists 28 manuscripts that left through the stack between June and September 2026. The runner does not yet persist its verdict history, so we cannot report how many of those runs blocked before the final pass, nor a false-positive rate; that log is the first fix on our list. What is on record is the incident register, 36 entries across the operation, six of which con-

cern manuscripts and produced seven gates that did not exist before the incident: a half-blank page that passed an eyeball check and a real-name byline on a venue assumed to be blind (both from one incident; G13 and G14 in Appendix B), content clipped at the page edge on every page (G16), a submit packet that handed the operator raw LaTeX to paste (G17), a template-default footer on nine pages (G18), one count published under two denominators (G21), and a paper whose numbers had been read from a scratch directory that no longer existed (G22). The stack has no override flag; the remaining bypass is a person uploading a PDF that never ran it.

4.2 Machine-checked claims: the artifact is the claim

The strongest ledger entries are those in which claim and artifact coincide. A theorem in a proof assistant is its own verification artifact: the statement is the claim, the proof term is the evidence, the kernel is the checker, and the axiom report states exactly what was assumed (de Moura and Ullrich, 2021). Nothing about such an entry is probabilistic, and checking it requires no trust in its author.

As an existence proof that this layer scales to research mathematics, our operation maintains FormalSLT, an open-source Lean 4 library of statistical learning theory built on mathlib (The math-

lib Community, 2020).¹ At that commit the library holds 358 source files, 184,769 lines, and 6,389 theorem and lemma statements, with no unfinished-proof markers, and its continuous integration re-checks every file on every push. The library is a claim ledger in the sense of Section 3: looking a theorem up means re-checking it, and because every proof names the lemmas it uses, the kernel itself enforces the evidence edges between entries. Formalization is expensive, so this layer is reserved for claims whose downstream weight justifies it; the ledger assumes a mixture of entry types.

4.3 Sealed predictions: the temporal layer

Forecast entries give the ledger its temporal dimension: they are claims admitted before their truth value exists. Our operation seals a probability, a horizon, and a resolution criterion for submission outcomes and internal milestones before the outcome is known; because the criterion is fixed at seal time, the forecaster cannot later choose what counts as success, and a resolution that does not post-date its seal is refused. At the time of writing the registry holds 150 forecast rows. The 52 venue-outcome forecasts that were sealed through the registry’s seal tool and have since resolved score a Brier of 0.213, against 0.25 for a constant forecast of 0.5. A broader tally that also scores component-level rows resolved on trust rather than through the seal tool gives 0.146 over 134 components; we do not count it as calibration evidence, and we cite the honest number once, as evidence that a working operation can sustain sealing and forced resolution for months, and for nothing more. Pre-registration (Nosek et al., 2018) and forecasting-judgment research (Tetlock, 2005) supply the underlying instruments. The addition here is placement: seals live in the same ledger as proofs and rerun receipts, linked to the claims they concern, and the manuscript gate stack refuses to pass a submission with no seal on file. The forecast row of Table 1 is the seal for this paper.

5 Community-scale claim ledgers

The mechanisms above serve one operation. The vision is the shared version, and it decomposes

¹<https://github.com/Robby955/FormalSLT>, MIT license. Counts were measured on a fresh clone of the main branch at commit 5e41e98 on 2026-09-06 with the commands in Appendix A; the metrics gate of Section 4.1 re-verifies such counts before any manuscript ships.

into three design problems the knowledge-base community is equipped for.

Schema. The minimal entry of Section 3 is thin on purpose; a shared ledger needs more. Claims become first-class nodes; evidence links are typed by checker, verdict, and date, and ordered by the trust hierarchy of Section 3. The assertion, provenance, and publication-information split that nanopublications standardize (Groth et al., 2010) carries over directly, and to it the ledger must add the scope of each claim, the evidence dependencies between entries, support and refutation links, and the adjudication history below. Since artifacts are re-checked and verdicts can change, an entry is append-only history, not a mutable cell. The hard schema problem is claim identity: recognizing that two papers state the same claim is exact for formal entries and open for informal ones, where re-statement, changed scope, and partial overlap defeat entity resolution over surface strings. Provenance, temporal scoping, and alignment remain the classical knowledge-graph concerns (Hogan et al., 2021).

Adjudication. Someone must run the checkers, and disputes need a protocol. A failed rerun must be classified: artifact rot (an environment that no longer builds) is not refutation, and the protocol must separate them. Verdicts downgrade rather than delete: an entry whose artifact has decayed drops in the trust order but keeps its history. Empirical entries need a re-verification cadence; formal entries need re-checking only when their assumptions change. The adjudication trail is part of the ledger, so a reader can audit the process that admitted a claim.

Incentives. Gates cost claimants effort, so placement decides adoption. Venues are the natural first gate operators: they already run fail-closed checks of weaker kinds (format, anonymity, plagiarism), and a portal is a chokepoint where a check cannot be skipped. A track could require one typed artifact per named contribution and check it mechanically at the portal. The ledger also creates a new citable object: resolutions and refutations are recorded with their adjudicator, so verification work becomes visible and creditable. A graded on-ramp keeps the gate honest about its own coverage: claims without a cheap checker enter a visible WARN tier instead of being silently excluded.

6 Limitations

Three boundaries bound the argument. The first is evidential: all evidence here comes from a single operation, and the selection effects run deep. The gates were built by the operation they gate, so passing them may reflect shared blind spots rather than correctness; the gate runner keeps no verdict history, so the block rate and the false-positive rate of the stack are unmeasured; the 52 resolved forecasts are not independent samples, and their resolution criteria were written by the forecaster being scored. The Brier number supports a feasibility claim and nothing stronger. The second is a coverage bias: fail-closed admission favors claims with cheap checkers, namely formal mathematics, rerunnable computation, and resolvable forecasts. Much of science, including qualitative and interpretive work and experiments too expensive to rerun, has no cheap checker, and a naive gate would entrench whatever happens to be easy to verify; the WARN tier is a patch for this, not a solution, and Section 3 already concedes that a rerun certifies reproducibility rather than the inference drawn from it. The third is that the community-scale problems are unsolved: claim identity across restatements, artifact rot at scale, and adjudication under adversarial pressure. The concrete next step is small: a pilot at a single workshop track, requiring one typed artifact per named contribution and checking it mechanically at the portal. The step the vision cannot skip is the adjudication protocol, on which the rest depends.

References

- Sean Bechhofer, Iain Buchan, David De Roure, Paolo Missier, John Ainsworth, Jiten Bhagat, Philip Couch, Don Cruickshank, Mark Delderfield, Ian Dunlop, Matthew Gamble, Danus Michaelides, Stuart Owen, David Newman, Shoaib Sufi, and Carole Goble. 2013. [Why linked data is not enough for scientists](#). *Future Generation Computer Systems*, 29(2):599–611.
- Glenn W. Brier. 1950. [Verification of forecasts expressed in terms of probability](#). *Monthly Weather Review*, 78(1):1–3.
- Tim Clark, Paolo N. Ciccarese, and Carole A. Goble. 2014. [Micropublications: A semantic model for claims, evidence, arguments and annotations in biomedical communications](#). *Journal of Biomedical Semantics*, 5(1):28.
- Leonardo de Moura and Sebastian Ullrich. 2021. [The lean 4 theorem prover and programming language](#). In *Automated Deduction (CADE 28)*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer.
- Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. [Detecting hallucinations in large language models using semantic entropy](#). *Nature*, 630:625–630.
- Paul Groth, Andrew Gibson, and Jan Velterop. 2010. [The anatomy of a nanopublication](#). *Information Services and Use*, 30(1–2):51–56.
- Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. [A survey on automated fact-checking](#). *Transactions of the Association for Computational Linguistics*, 10:178–206.
- Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d’Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, and 1 others. 2021. [Knowledge graphs](#). *ACM Computing Surveys*, 54(4):1–37.
- John P. A. Ioannidis. 2005. [Why most published research findings are false](#). *PLoS Medicine*, 2(8):e124.
- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. [Survey of hallucination in natural language generation](#). *ACM Computing Surveys*, 55(12):1–38.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *The Twelfth International Conference on Learning Representations (ICLR)*. ArXiv:2305.20050.
- Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. [The AI scientist: Towards fully automated open-ended scientific discovery](#). *arXiv preprint arXiv:2408.06292*.
- Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Beth Plale, Yogesh Simmhan, Eric Stephan, and Jan Van den Bussche. 2011. [The Open Provenance Model core specification \(v1.1\)](#). *Future Generation Computer Systems*, 27(6):743–756.
- Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. 2018. [The preregistration revolution](#). *Proceedings of the National Academy of Sciences*, 115(11):2600–2606.
- David Shotton. 2009. [Semantic publishing: The coming revolution in scientific journal publishing](#). *Learned Publishing*, 22(2):85–94.
- Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. 2024. [Can LLMs generate novel research ideas? a large-scale human study with 100+ NLP researchers](#). *arXiv preprint arXiv:2409.04109*.

Philip E. Tetlock. 2005. *Expert Political Judgment: How Good Is It? How Can We Know?* Princeton University Press, Princeton, NJ.

The mathlib Community. 2020. [The lean mathematical library](#). In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*, pages 367–381. ArXiv:1910.09336.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 809–819. ArXiv:1803.05355.

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

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550. ArXiv:2004.14974.

A Reproducing the library counts

The counts in Section 4.2 were produced on 2026-09-06 by the following commands against a fresh clone at commit 5e41e98; the last command searches for the unfinished-proof keyword outside the module docstrings that state its absence.

```
git clone --depth 1 \
  https://github.com/Robby955/FormalSLT
cd FormalSLT
find FormalSLT -name '*.lean' | wc -l
find FormalSLT -name '*.lean' \
  -exec cat {} \; | wc -l
find FormalSLT -name '*.lean' | xargs \
  grep -c '^theorem\|^lemma' | awk -F: \
  '{s+=$2} END {print s}'
grep -rn '\bsorry\b' FormalSLT \
  --include='*.lean' | grep -v -i \
  'no 'sorry'|no sorry' | wc -l
```

B The manuscript gate stack

The gate runner applies the gates below to a registered paper and emits one of four verdicts: SUBMIT_READY, PANEL_PENDING, NEEDS_FIX, or MISSING_BUNDLE. Only the first permits submission. “Deterministic” gates are pure parses or recounts with no model call; “model-assisted” gates call a language or vision model and degrade to a warning, never to a silent pass, when the model is unavailable.

- G1. Bundle present: source, built PDF, review prompt, and submit packet all exist. Deterministic.
- G2. Venue format: document class, page cap, anonymization grade, and required sections match the venue’s verified policy. Deterministic.
- G3. Voice: punctuation and self-referential-phrase limits on the source. Deterministic.
- G4. PDF freshness: the built PDF is newer than its source. Deterministic.
- G5. Sealed prediction: a forecast of the submission’s outcome is on file (Section 4.3). Deterministic.
- G6. Disclosure log: the venue’s disclosure requirements have a log entry. Deterministic.
- G7. Bound panel verdict: an adversarial review panel’s receipt whose hashes recompute against the registered source, PDF, and prompt; a stale receipt cannot colour a newer build. Model-assisted.
- G8. Non-checks section: the review prompt names what reviewers must not propose. Deterministic.
- G9. Verified metrics: quantitative claims about the formal library match a live recount of the source tree. Deterministic.
- G10. Exact support and hedges: each key-claim citation supports the exact claim, and each hedged key step carries a discharge obligation. Model-assisted audit with a deterministic block on open obligations.
- G11. Formal style: changed proof files pass the library’s lint. Deterministic.
- G12. Review flags: no open review flag remains on the paper or venue. Deterministic.
- G13. Layout fill: no mostly blank page, by per-page character count. Deterministic.
- G14. Blind policy: the byline matches the venue’s verified anonymity policy; fails closed with no policy on record. Deterministic.
- G15. Figure format: no bitmap font and no raster image below 250 dpi at its placed size. Deterministic.
- G16. Margin overflow: no overfull line beyond 5 pt in the build log and no ink in the outer page band on any page. Deterministic.
- G17. Packet abstract: the submit packet carries a clean plaintext abstract and a one-line summary within the portal’s limits. Deterministic.
- G18. Template residue: no template-default string survives in the built PDF. Deterministic.
- G19. Visual review: rendered pages pass a multimodal check for overflow, clipping, and broken layout. Model-assisted.
- G20. Byline identity: a real-name submission carries the canonical author, affiliation, and address tokens. Deterministic.
- G21. Numeric consistency: no “ k/n ($p\%$)” pair in the PDF disagrees with itself. Deterministic.
- G22. Results provenance: the paper directory references no ephemeral scratch path, and every absolute input path resolves on disk. Deterministic.