

# The Reproducibility Criterion for Machine-Generated Evidence: Sufficient Conditions and Certification Limits for Independently Verifiable AI Decisions

Raza Sharif  
CyberSecAI Ltd, London, United Kingdom  
`contact@cybersecai.co.uk`

September 2026

Working paper v0.5 — machine-checked margin certificate added

## Abstract

An integrity-protected record of an AI decision and an independently reproducible decision establish different properties. This paper proposes an evidence-oriented framework for the latter. We give sufficient conditions for exact output replay under explicit assumptions about authentic execution binding, artifact availability, deterministic computation and complete inputs. For greedy decoding, we prove a conservative interval-margin certificate with a tie-breaking-aware boundary condition. Failure of this certificate means that this procedure has not certified stability; it need not imply actual divergence. We also give examples in which omitted execution information leaves multiple outputs compatible with the recorded configuration, without claiming that every omitted field defeats every possible verifier. Preliminary author-reported experiments illustrate the risk of an unrecorded decoding penalty. A revised artifact harness captures requests, responses, output bytes and available metadata, rejects API failures and applies strict benchmark scoring. The original accuracy figures are not retained as validated measurements because the earlier scorer admitted false positives. No new model-inference results or hardware attestations are claimed in this revision. The logical core of the greedy margin certificate is additionally formalised and machine-checked in the Lean 4 proof assistant, reported with its complete axiom basis and no unproved steps, so that the selection-preservation implication is verified rather than asserted. The contribution is a proposed evidentiary framework and an inspectable replication protocol, rather than a universal necessity theorem or a determination of legal admissibility.

## 1 Introduction and scope

An authenticated, hash-linked log can support the integrity and attribution of recorded events, subject to the security of its capture and signing mechanisms. It does not by itself establish that the logged input, model or runtime actually produced the logged output. Nor does repeated output equality establish scientific correctness: an implementation may reproduce the same error.

The central distinction is between *record verification*, *observed replay agreement*, and *certified replay stability*. The first concerns the evidence record; the second compares particular executions; the third establishes output invariance over a stated class of compatible executions. Independent reproduction also requires access to the artifacts and inputs, not merely their digests.

We restrict exact replay to a finite inference decision with a specified output encoding, stopping rule and execution boundary. An agentic workflow can be treated similarly only if its external observations, retrieval results, tool outputs and state transitions are captured or supplied by a

controlled replay mechanism. Re-running an external action is neither required nor necessarily appropriate.

The framework is related to the author’s Agent Audit Trail proposal [1]. This paper stands on its own definitions; it does not rely on a particular Internet-Draft section being implemented or accepted as an IETF standard. Deterministic inference and byte-equality verification are prior work [4]; the proposed contribution is their evidentiary framing, the explicit capture/replay assumptions, and the distinction between conservative certification and observed agreement.

## 2 Definitions and trust assumptions

**Definition 1** (Decision and execution record). *A decision is  $D = (x, y, R)$ , where  $x$  is the complete replay input,  $y$  is the output byte string, and  $R$  is the captured evidence record. A fixed serialization specifies exactly which bytes are compared. Hashes of generated text do not include transport timestamps or timing metadata unless those are explicitly part of  $y$ .*

**Definition 2** (Execution closure). *Write  $y = F(x; M, K, T, \theta, E, s, \omega)$ , where  $M$  is the loaded model including adapters and quantization,  $K$  the tokenizer,  $T$  the prompt template and system instructions,  $\theta$  the decoding and stopping configuration,  $E$  the numerical implementation,  $s$  relevant state, and  $\omega$  any randomness or external observations. The closure contains every dependency that can affect the specified output within the declared execution boundary. A closed replay record identifies or constrains these dependencies sufficiently to reproduce their relevant behavior. Digests must resolve to available artifacts; a digest alone cannot reconstruct missing bytes.*

Closure is a semantic requirement, not a promise that a short metadata checklist captures every dependency. A recorded seed must be accompanied by the generator, state initialization, draw order and numerical behavior of the sampling procedure. Fixed hardware and thread counts alone do not establish determinism. Drivers, kernels, reduction schedules, caches, concurrent requests, defaults and postprocessing may matter.

**Definition 3** (Configuration-compatible executions). *Let  $C(R)$  contain executions satisfying the independently verified input, artifact and runtime constraints in  $R$ . Membership is defined before the final output-digest comparison: otherwise defining compatibility by matching  $H(y)$  would make the stability claim circular. The actual captured execution must belong to  $C(R)$ .*

**Definition 4** (Certified exact replay). *A record supports certified exact replay when (i) its provenance and binding to the claimed original execution are valid under a stated trust model; (ii) an independent verifier can obtain the required inputs and artifacts and execute a compatible replay; and (iii) every execution in  $C(R)$  yields the same output bytes as the recorded execution. Observed equality of two runs establishes only equality for those runs.*

The provenance assumption requires an explicit trusted capture mechanism or a verifiable execution proof, with binding to the input, artifacts, configuration and output. An operator-written manifest or a signature on an arbitrary assertion is not, by itself, proof of correct execution. Cryptographic checks depend on trusted keys, valid measurements and the relevant unforgeability and hash-security assumptions. This paper does not prove a particular attestation implementation secure. Integrity and authenticity of the verifier and its comparison procedure are also assumed.

### 3 Greedy decoding and a conservative margin certificate

At a fixed generation step and prefix, let  $z \in \mathbb{R}^m$  be a reference vector of finite logits. Fix a total tie-priority order  $\tau$  on tokens; smaller rank means higher priority. Let  $r$  be the token selected by applying argmax with  $\tau$  to  $z$ .

Let  $\mathcal{A}$  be the possible logit vectors for compatible executions at that prefix. Assume a sound enclosure

$$\mathcal{A} \subseteq \mathcal{N} = \prod_{i=1}^m [z_i - e_i, z_i + e_i], \quad e_i \geq 0.$$

The box is a conservative relaxation; its points need not all be executable. For each competitor  $j \neq r$ , define

$$g_j = z_r - z_j, \quad \delta_j = e_r + e_j, \quad \varepsilon = \max_{j \neq r} \delta_j.$$

Here  $\delta_j$  is the maximum decrease of the  $r$ - $j$  gap over the box. All three quantities are specific to the input prefix and numerical model. In particular, the bound is not determined by a hardware label alone.

For a simple floating-point sum, under the usual assumptions including  $nu < 1$  and exclusion or explicit handling of overflow and exceptional values, a standard bound is

$$\left| \text{fl} \left( \sum_k v_k \right) - \sum_k v_k \right| \leq \gamma_n \sum_k |v_k|, \quad \gamma_n = \frac{nu}{1 - nu}.$$

This illustrates one source of enclosures [3]. Applying it to an entire transformer requires propagation through the full computation, including nonlinear operations, quantization, matrix kernels and previous layers. The summands and intermediate bounds are data-dependent. A valid full-model enclosure is an assumption of the theorem below, not something furnished automatically by an engine version or by the experimental harness.

**Lemma 1** (Tie-aware margin certificate). *Suppose the enclosure is sound at every step along the reference trajectory. If, for every competitor  $j \neq r$ ,*

$$g_j > \delta_j \quad \text{or} \quad (g_j = \delta_j \text{ and } \tau(r) < \tau(j)),$$

*then every compatible greedy execution selects the reference token at every step. With the same stopping rule and output serialization, all such executions produce identical output bytes.*

*Proof.* For every  $\tilde{z} \in \mathcal{N}$ ,  $\tilde{z}_r - \tilde{z}_j \geq g_j - \delta_j$ . A strict positive lower bound makes  $r$  beat  $j$ . At a zero lower bound the stipulated priority makes  $r$  win any resulting tie. Hence  $r$  is selected against all competitors. Identical prefixes then permit induction over subsequent steps, using the enclosure for that shared prefix. Identical stopping and serialization complete the byte-level result.  $\square$

The simpler condition  $\min_{j \neq r} g_j > \varepsilon$  is sufficient independently of tie priority, but is not necessary. A deterministic engine may reproduce exactly even when a conservative enclosure fails this test.

**Proposition 1** (Meaning of acceptance and rejection). *The procedure that verifies the enclosure and applies Lemma 1 is sound for greedy output stability. It can be incomplete for  $\mathcal{A}$ . Rejection establishes only that this enclosure-based procedure did not certify stability. It does not establish an actual divergent execution or the impossibility of another certificate.*

*Proof.* Soundness follows from Lemma 1. For incompleteness, take reference logits  $(3, 2)$ , admissible vectors  $\{(3, 2), (3, 0)\}$  and enclosure  $\{3\} \times [0, 4]$ . The first token wins in both admissible vectors, but  $g_2 = 1 < \delta_2 = 2$ , so the procedure rejects. The enclosure includes a reversing perturbation which the actual implementation never realizes.  $\square$

**Remark** (Why a fixed tie rule alone is insufficient). *With reference  $(3, 2)$  and enclosure  $\{3\} \times [1, 3]$ ,  $g_2 = \delta_2 = 1$ . If token 2 has priority at ties,  $(3, 3)$  changes the selected token. The corrected certificate rejects this case. It accepts equality only if token 1 has priority.*

These results concern greedy decoding. Preserving the argmax does not preserve a sampled token: a small probability change can move a fixed random draw across a sampling threshold. Sampled replay therefore requires deterministic numerical sampling with a fixed random stream, or a separate certificate for the sampling decisions. Lemma 1 does not provide that certificate. Certificate arithmetic itself must be exact or conservatively rounded.

### 3.1 Machine-checked formalisation of the margin certificate

The logical core of Lemma 1—that a strict margin over a sound enclosure forces selection of the reference token—has been formalised and checked in the Lean 4 proof assistant [2]. Writing  $w$  for the reference token  $r$ , and taking  $\delta_j = e_r + e_j$  as in Section 3, the checked single-step statement is:

```
theorem margin_certificate_sound
  {k : Nat} (z z' : Fin k -> Int) (delta : Fin k -> Int) (w : Fin k)
  (hadm : forall j, j != w -> z' w - z' j >= (z w - z j) - delta j)
  (hmargin : forall j, j != w -> z w - z j > delta j) :
  forall j, j != w -> z' w > z' j := by
  intro j hj
  have h1 := hadm j hj
  have h2 := hmargin j hj
  omega
```

Here  $\mathbf{z}$  is the reference logit vector,  $\mathbf{z}'$  the logit vector of any compatible execution at the same prefix,  $\mathbf{delta}$  the per-competitor enclosure width, and  $\mathbf{w}$  the reference token. The hypothesis `hadm` is exactly enclosure soundness in gap form: no compatible execution reduces the  $w$ - $j$  gap by more than  $\delta_j$ , i.e.  $z'_w - z'_j \geq (z_w - z_j) - \delta_j$ . The hypothesis `hmargin` is the strict margin  $g_j > \delta_j$ . The conclusion is that  $w$  strictly outscores every competitor, so greedy selection returns  $w$ . An integer model is used so that the discrete comparison is exact and the goal is discharged by Lean core's `omega` decision procedure with no external library; a real-valued model checked against Mathlib yields the same statement. This is the greedy-certificate route to condition **L4** of Theorem 1.

The value of the formalisation is the accompanying axiom report. Lean's `#print axioms` lists the complete foundational basis a proof depends on:

```
'margin_certificate_sound' depends on axioms:
  [propext, Classical.choice, Quot.sound]
```

These are the three standard axioms underlying Lean and Mathlib, and the report contains no `sorryAx`. The token `sorry` is Lean's placeholder for an unproved step; its absence, over only the standard axioms, is an objective and independently reproducible certificate that the statement is proved rather than assumed. A reader can regenerate it by pasting the source into <https://live.lean-lang.org> or building it with `lake build`.

We are precise about scope. What is machine-checked is the discrete, single-step implication: a strict margin together with a sound enclosure implies preservation of the reference token, which is the strict branch of the boxed condition in Lemma 1. The equality-boundary refinement (the case  $g_j = \delta_j$  resolved by tie priority, in the remark following Lemma 1) and the step-wise induction to identical output bytes are argued in the text and are not part of the checked core. The proof assistant certifies neither that a given implementation’s enclosure is sound—an  $E$ -dependent analytic assumption of Section 3—nor any floating-point bound; it certifies that, granted a sound enclosure and a strict margin, the selection conclusion follows with no hidden assumption. In the terms of Proposition 1, the formalisation strengthens the soundness of the accept rule; it does not address its incompleteness for  $\mathcal{A}$ .

## 4 Sufficient replay conditions and certification limits

**Theorem 1** (Sufficient conditions for certified exact replay). *Assume the execution-record provenance and binding conditions of Section 2 hold. Suppose:*

- L1. The decoding, tie-breaking, stopping and output-encoding rules are fixed. Any stochastic choices are fixed and replayable, including the relevant generator and draw semantics.*
- L2. The actual loaded model, tokenizer, template and other model artifacts are verifiably identified and available to the verifier.*
- L3. The complete inputs and relevant initial state, including captured external observations within scope, are recoverable as the same bytes or under an explicitly fixed semantics.*
- L4. The execution closure is constrained so that its output is invariant over  $C(R)$ . This can be established by fully deterministic execution semantics, or for greedy decoding by a verified accepting certificate from Lemma 1 together with deterministic stopping and serialization.*

*Then  $D$  supports certified exact replay.*

*Proof.* The authenticated execution binding places the original execution in  $C(R)$  and binds its output to  $y$ . Availability allows the verifier to execute a member of  $C(R)$ . L4 makes the relevant output invariant across that class; L1–L3 fix its interpretation and dependencies. Thus the replay yields the same bytes. Comparing full byte strings directly establishes equality for those strings. A full cryptographic digest provides a computational equality check under the stated hash-security assumptions; it is not an unconditional mathematical identity proof or an independent proof of capture authenticity.  $\square$

This theorem is intentionally conditional. It is an engineering assurance framework, not a claim that metadata pinning constructs a determinism proof. A fixed reduction schedule addresses one source of variation; all other output-affecting behavior must also satisfy L4. The theorem does not establish a universal necessary implementation checklist: equivalent artifact identifiers, semantic constraints and execution proofs may provide alternative evidence.

**Proposition 2** (Ambiguity from omitted execution information). *There exist configurations in which omitting a randomness choice, model choice, input or numerical evaluation rule permits two compatible executions with different outputs. Their remaining configuration fields alone cannot establish a unique output for that class.*

*Proof.* The following finite examples witness the respective omissions.

- (i) A decoder selects between tokens A and B using an unrecorded random bit. Both outputs are compatible with the same recorded model and input.
- (ii) Two permitted model artifacts have output biases (1, 0) and (0, 1), respectively. An unidentified model choice changes the greedy output under identical other rules.
- (iii) A fixed byte-sensitive function emits A for input byte 0 and B for input byte 1. Omitting the input leaves both executions compatible.
- (iv) Under binary64 round-to-nearest arithmetic, evaluating  $(10^{16} + (-10^{16})) + 1$  gives 1, whereas  $10^{16} + ((-10^{16}) + 1)$  gives 0. Use this sum as token A’s logit and the fixed value 0.5 as token B’s logit. Omitting the permitted reduction order leaves different greedy outputs compatible.

In each example the specified retained configuration information is compatible with two outputs, so it does not identify a unique one.  $\square$

A sound verifier can reject such ambiguous records. The examples do not imply that every verifier omitting a field is unsound, that every decision needs that exact field, or that all differing inputs necessarily produce differing outputs. If a trusted proof supplies the missing constraint, the evidence is no longer limited to the retained configuration fields in the examples.

An unrecorded penalty is a practical instance of configuration ambiguity. Calling this a *cryptographic forgery* would require a defined cryptographic verification scheme and a demonstrated break. Two different outputs from incompletely specified runs do not constitute such a break. A securely committed output hash can detect a mismatching replay, although it cannot make an underspecified computation uniquely replayable.

## 5 Preliminary observations and corrected replication protocol

### 5.1 Status of the historical observations

Earlier drafts reported local Ollama runs on Apple Silicon/Metal using `llama3.2:1b` and `qwen2.5-coder:0.5b`. The supplied materials contain shortened output hashes, but not the historical raw response files, complete original environment manifest, or a trusted execution attestation. Table 1 preserves those reports for transparency; they are not new measurements or independently verified replay results.

| Author-reported configuration          | Run 1 prefix | Run 2 prefix |
|----------------------------------------|--------------|--------------|
| Llama, temperature 0, penalty 1.1      | 809dd715     | 809dd715     |
| Llama, temperature 0, penalty 1.3      | 7e8e9bec     | 7e8e9bec     |
| Llama, temperature 0.9                 | e1f8afb1     | 0d62d43a     |
| Llama, reported thread/batch variation | 809dd715     | 809dd715     |
| Qwen, temperature 0, penalty 1.1       | 37f6875f     | 37f6875f     |
| Qwen, temperature 0, penalty 1.3       | faa56511     | faa56511     |
| Qwen, temperature 0.9                  | e808ea28     | a3a4c085     |

Table 1: Historical digest prefixes transcribed from the author-supplied draft. Prefixes cannot substitute for archived full digests and output bytes. The reported thread/batch equality does not establish that a margin bound held.

The historical 30-item battery reported 21/30 on two greedy passes and 17/30, 21/30, 19/30 and later 22/30 on sampled passes. Inspection of the original scorer found substring matching

and acceptance of any matching number in a response. For example, it accepted “not spam” as “spam” and a response explicitly rejecting 43 as numerically correct for 43. These scores are therefore **withdrawn as reliable accuracy estimates pending rescoring of preserved outputs or a new run**. No corrected accuracy figures are asserted. The first three sampled reports span 13.3 percentage points; including the fourth spans 16.7 points, but neither range repairs the underlying scoring problem.

The original shell harness could hash empty output after an API/JSON failure and report equality as reproduction. That failure mode does not establish that a historical run failed, but prevents treating unchecked harness summaries as sufficient evidence of successful inference.

## 5.2 Corrected artifact bundle

The revised `table1_experiments.sh` invokes a standard-library Python runner. For each model, it runs two repetitions of: a greedy baseline; a penalty-only change; unseeded sampling; a thread-only change; a batch-only change; the combined thread/batch change; and a fixed-seed sampling control. The baseline explicitly requests one thread and batch 64; the changed values are eight threads and batch 512. Runtime options are requests, not proof that the backend honors them. The fixed-seed sampling control is a separate condition and is not described as unseeded sampling.

The revised mini-evaluation retains the supplied 30 questions and uses prespecified whole-answer matching. It allows only documented case, whitespace, presentation-Unicode and terminal-full-stop normalization for strings, and whole-string decimal equivalence for numeric answers. This is a strict answer-format benchmark, not comprehensive semantic grading. Extra explanatory text is scored incorrect even if it contains a correct answer. Two greedy passes and four sampled passes with recorded distinct seeds are the default. These choices define a corrected experiment; they do not recreate undocumented historical defaults.

Both runners stop with a nonzero status on API errors, malformed or incomplete responses, missing output or empty output. They preserve the exact request, raw JSON response, extracted UTF-8 output, full SHA-256 digests and per-item or per-condition records. Token-limit termination is recorded separately. Repeat agreement is compared on actual output bytes, not shortened hashes. Available engine and model metadata are captured before and after the run, and detected changes invalidate completion. A local integrity checker verifies the archive without contacting the inference server.

Metadata includes available model details and template/parameter information, explicit request options and client-host information. An operator may additionally hash specified engine binaries or artifact files. This does *not* automatically establish computational closure, server-side configuration, kernel determinism, artifact retention, or hardware-backed attestation. API-reported model identifiers are recorded as such; the harness does not relabel them as independently measured weights-blob hashes [11].

## 5.3 Validation and limits

Nine targeted automated tests passed during preparation of this revision. They cover false-positive scoring, permitted normalization, incomplete/error/empty responses, full mock table execution, full mock benchmark execution, failure propagation, metadata drift, preservation of existing run directories, archive tampering, and the tie-priority counterexample (some tests cover several checks). The mock table exercises 28 generation records and the mock benchmark 180; these are test fixtures, not LLM measurements.

No model inference was rerun during this correction: the revision environment was Linux/x86\_64

without Ollama or the author’s Apple Silicon/Metal setup. Future reports must distinguish (a) archive-integrity checks, (b) repetition on a new environment, (c) exact historical replay and (d) a trusted stability/attestation certificate. A digest difference across environments neither automatically falsifies a within-environment claim nor proves that an environment change caused it. Likewise a changed penalty need not change every answer, and two unseeded sampled runs can coincide. The runner reports observed relationships without forcing an expected outcome.

## 6 Evidence grades and legal relevance

The following labels are proposed reporting categories, not independently recognized certifications:

- **Grade A: certified exact replay.** The provenance, availability and stability assumptions of Theorem 1 are supported by explicit evidence.
- **Grade B: traceable record without certified exact replay.** Available evidence supports specified integrity/provenance checks, but the replay or stability requirements have not been established. Even this label requires actual record protection; it does not apply automatically to every API response.
- **Distributional verification (Q when discussing quantum inference).** A separately defined statistical test assesses output distributions with stated error bounds. This is a different target, not an exact-replay grade or a consequence of two repeated runs.

A hosted API that does not expose the relevant artifacts or execution evidence cannot support the external replay procedure described here through that interface alone. This is an access and evidence limitation, not a universal impossibility result for all hosted services. Alternative access arrangements, controlled replay or cryptographic execution proofs may change the available assurance.

No-cloning concerns arbitrary unknown quantum states [8]; it does not prohibit preparation of a known state or deterministic outcomes for every circuit. Quantum inference is therefore not assigned an unconditional distribution-only ceiling. Statistical reproduction is appropriate where the task genuinely depends on stochastic measurement; specifying and validating such a test is outside this paper’s experiments.

Proposed U.S. Federal Rule of Evidence 707 concerns applying expert-evidence reliability requirements to certain machine-generated evidence offered without an expert witness [9]. This framework addresses a technical replay question within that broader reliability inquiry. Reproducing an output does not establish the sufficiency of the underlying data, scientific validity, error rate, fitness for purpose, or admissibility. No independent legal “re-derivation limb” is assumed.

EU AI Act Article 12 concerns automatic logging and traceability for high-risk AI systems; Article 14 concerns effective human oversight [10]. Captured inputs and verifiable records may support these objectives, but neither exact replay nor either grade alone establishes compliance. Article 12 is not represented here as mandating byte-identical re-execution.

## 7 Related work, contribution and remaining work

EigenAI studies deterministic inference and equality-based verification [4]. DiFR addresses inference verification despite nondeterminism [5]. Zero-knowledge inference can prove a relationship between committed computations and outputs while protecting model confidentiality [6]; that

relationship proof is distinct from giving an external verifier the artifacts required for direct replay. Provenance work connects evidence requirements to generated content [7]. These are adjacent approaches, not excluded alternatives.

The proposed contribution is the separation of record integrity, observed agreement and certified stability; the explicit treatment of capture binding and execution closure; a tie-aware conservative certificate whose strict-margin core is machine-checked in Lean 4; and a practical protocol for exposing configuration ambiguity. No priority claim is made for deterministic decoding, cryptographic hashing or logging. Establishing broader novelty requires a fuller literature review.

The next empirical step is a fresh run of the corrected harness on a documented environment, with full outputs, artifacts and manifest retained. Establishing Grade A additionally requires an implemented trusted binding mechanism and deterministic execution proof or sound full-model enclosure. A wider preregistered benchmark, independent replication and statistical analysis would be needed for claims about accuracy or generality. Computational and operational cost of closure and certification remains to be measured; this revision does not claim that it is universally cheap.

## 8 Conclusion

A record can be intact while the decision it describes remains underspecified. Under explicit provenance, availability and deterministic-behavior assumptions, exact replay is achievable. A conservative certificate can establish greedy stability but its rejection is not proof of divergence. The corrected artifact bundle makes the experimental protocol inspectable while preserving the distinction between code tests, reported historical observations and new inference evidence. These scoped claims form the basis for further independent evaluation.

**Revision note.** Version 0.4 corrects the tie boundary, removes the universal necessity claim, restricts the greedy certificate to its applicable decoding regime, separates trusted attestation from operator metadata, repairs the experiment harness, withdraws unsupported accuracy estimates, and repairs the page layout. The revision incorporates external review feedback; it is not represented as independent human peer review. Historical observations remain author-reported.

**Revision note (v0.5).** Version 0.5 adds a machine-checked Lean 4 formalisation of the strict-margin core of Lemma 1, together with its standard-axioms report, and updates the references accordingly. No existing claim is altered and no new inference or hardware-attestation result is introduced; the formalisation certifies the discrete selection-preservation implication only, under the scope stated in Section 3.

## References

- [1] R. Sharif. *Agent Audit Trail: A Standard Logging Format for Autonomous AI Systems*. Internet-Draft, draft-sharif-agent-audit-trail. <https://datatracker.ietf.org/doc/draft-sharif-agent-audit-trail/>. Author proposal; not an IETF standard.
- [2] L. de Moura and S. Ullrich. “The Lean 4 Theorem Prover and Programming Language.” In *Automated Deduction — CADE 28*, LNCS 12699, Springer, 2021. Formalisation sources accompany this paper’s artifact bundle; the statement and its axiom report can be reproduced at <https://live.lean-lang.org> or via `lake build`.
- [3] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*, 2nd ed. SIAM, 2002, Chapter 4.

- [4] D. R. Alves et al. *EigenAI: Deterministic Inference, Verifiable Results*. arXiv:2602.00182, 2026. <https://arxiv.org/abs/2602.00182>.
- [5] A. Karvonen et al. *DiFR: Inference Verification Despite Nondeterminism*. arXiv:2511.20621, 2025. <https://arxiv.org/abs/2511.20621>.
- [6] H. Sun, J. Li and H. Zhang. *zkLLM: Zero Knowledge Proofs for Large Language Models*. arXiv:2404.16109, 2024. <https://arxiv.org/abs/2404.16109>.
- [7] *Verifiable Provenance and Watermarking for Generative AI: An Evidentiary Framework for International Operational Law and Domestic Courts*. arXiv:2605.21002, 2026. <https://arxiv.org/abs/2605.21002>.
- [8] W. K. Wootters and W. H. Zurek. “A single quantum cannot be cloned.” *Nature* 299, 802–803, 1982.
- [9] Advisory Committee on Evidence Rules. Proposed Federal Rule of Evidence 707, circulated for public comment in 2025–2026. See the U.S. Courts rulemaking materials: <https://www.uscourts.gov/forms-rules/proposed-amendments-published-public-comment>. Discussed as a proposal, not an enacted requirement.
- [10] Regulation (EU) 2024/1689, Articles 12 and 14. <https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng>.
- [11] Ollama. API and Modelfile documentation. <https://docs.ollama.com/api/generate>; <https://docs.ollama.com/api/tags>; <https://docs.ollama.com/modelfile>. Request metadata is distinct from an execution attestation.