MathLedger: A Verifiable Learning Substrate with Ledger-Attested Feedback

Ismail Ahmad Abdullah
ismail.abdullah.23@cnu.edu

December 22, 2025

Abstract

Contemporary AI systems achieve extraordinary performance yet remain opaque and non-verifiable, creating a crisis of trust for safety-critical deployment. We introduce MathLedger, a substrate for *verifiable machine cognition* that integrates formal verification, cryptographic attestation, and learning dynamics into a single epistemic loop. The system implements *Reflexive Formal Learning* (RFL), a symbolic analogue of gradient descent where updates are driven by verifier outcomes rather than statistical loss.

Phase I experiments validate the measurement and governance substrate under controlled conditions. CAL-EXP-3 validates measurement infrastructure (Δp computation, variance tracking); separate stress tests confirm fail-closed governance triggers correctly under out-of-bounds conditions. No convergence or capability claims are made. The contribution is infrastructural: a working prototype of ledger-attested learning that enables auditability at scale.

Keywords: verifiable learning, formal verification, cryptographic attestation, reflexive feedback, fail-closed governance

1 Introduction: The Verifiability Gap

Modern large language models are universal approximators of text, not of truth. Hallucination is structurally baked into density-estimation objectives; conventional evaluations penalize abstention and reward confident output regardless of correctness [5]. In safety-critical domains—finance, law, infrastructure, policy—this creates an untenable gap between capability and trust.

The AI industry is discovering a structural constraint:

Performance without verifiability is not deployable at scale.

Mathematics offers a way out: verifiable reasoning with machine-checkable proofs. MathLedger converts mathematics into a *living protocol* for learning under formal law.

1.1 What Problem Does This Address?

Existing approaches to improving AI reliability fall into three categories, each with limitations:

1. **Reward shaping (RLHF, DPO):** Human preferences guide learning, but preferences are noisy, inconsistent, and gameable. The feedback signal is statistical, not verifiable.

2. **Verifier-guided generation:** Proof assistants check outputs post-hoc, but rejected outputs provide no structured learning signal. The verifier is a filter, not a teacher.

3. **Benchmark scaling:** Larger test sets reduce variance but do not establish correctness. Passing benchmarks does not imply understanding.

MathLedger takes a different approach: *the verifier's outcome becomes the learning signal itself.* Every update is justified by a configured verifier outcome (pass/fail/abstain), recorded in an immutable ledger. This creates a closed epistemic loop where learning is constrained to verifier-attested outcomes.

1.2 The Chain of Verifiable Cognition

The system implements an end-to-end pipeline:

Input → Proof-or-Abstain → Ledger Attestation → Dual Commitment → Policy Update

Each component is cryptographically bound:

- **Proof-or-Abstain:** A configured verifier (Phase I: synthetic proxy; Phase II+: Lean kernel) validates reasoning or the system explicitly abstains. No middle ground.

- **Ledger Attestation:** Verifier-accepted events are sealed into a monotone, append-only ledger with Merkle roots.

- **Dual Commitment:** Both reasoning artifacts (r_t) and interface state (u_t) are committed: $H_t = \text{Hash}(\texttt{EPOCH:}\|r_t\|u_t)$.

- **Policy Update:** Reflexive Formal Learning (RFL) adjusts the policy based on verification outcomes.

This architecture enables a new primitive: *learning from verifier-attested outcomes rather than statistical loss.*

1.3 What Is Genuinely New

MathLedger combines three elements not commonly integrated end-to-end [3, 2]:

1. **Ledger-attested learning signals:** Unlike reward models or human feedback, the learning signal is a cryptographically committed verification outcome.

2. **Fail-closed governance:** The system cannot silently degrade. Either verification succeeds and the outcome is admitted to the ledger, or the system abstains and logs the failure.

3. **Auditability as infrastructure:** Every update has a replayable provenance chain. Post-hoc analysis can reconstruct exactly what was learned and why.

This paper reports Phase I experiments that validate the substrate. No capability or convergence claims are made.

2 System Architecture

2.1 Pipeline Overview

The experimental methodology evaluates the MathLedger substrate under controlled conditions. Phase I focuses on measurement validation (Δp computation, variance tracking) and fail-closed governance verification.

2.1.1 Test Harness

Experiments are conducted within a dedicated FO (Feedback-Optimized) cycle harness. This harness simulates realistic operating conditions, allowing for precise control over input parameters and comprehensive capture of output metrics. The harness is designed to ensure reproducibility and provide a consistent environment for comparative analysis.

2.1.2 Configurations Under Test

We evaluate several key configurations:

1. **RFL vs. Baseline:** We compare the performance of the MathLedger system with the RFL mechanism enabled against a baseline configuration where RFL is disabled or replaced with a static control mechanism. This comparison aims to characterize the behavior of feedback mechanisms on system stability and output.

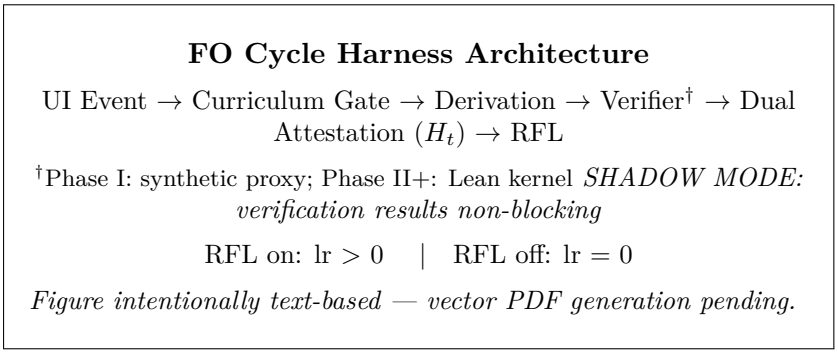

Figure 1: Architectural overview of the FO cycle harness. The pipeline runs: UI Event → Curriculum Gate → Derivation Engine → Configured Verifier → Dual Attestation (H_t) → RFL policy update. Phase I uses a synthetic proxy verifier; Lean kernel integration is Phase II+. SHADOW MODE: all verification results are observational and non-blocking.

2. **PL slice analysis (propositional logic slices):** The system's behavior is further investigated by varying propositional-logic curriculum slices under fixed seeds and budgets. Different PL slice configurations are tested to understand their influence on computational efficiency, decision-making latency, and overall ledger integrity.

2.1.3 Measurement Metrics

The following key metrics are recorded and analyzed during the experimental runs:

- **Abstention Rate:** The frequency at which the system abstains from making a definitive judgment.
- H_t **Dynamics:** We track the evolution and stability of the dual attestation hash H_t over time, providing insights into the system's state processes.
- **Operational Metrics (runtime telemetry):** A suite of fundamental performance indicators, including transaction processing speed, error rates, and resource utilization, are measured to establish a foundational understanding of system behavior.

2.2 The Monotone Ledger

Definition 1 (Monotone Ledger)**.** *A ledger $\mathcal{L}$ is a sequence of blocks $(B_1, B_2, \ldots)$ where each B_t contains verifier-accepted proof artifacts (Phase I: proxy-accepted; Phase II+: Lean-verified) with: (i) canonical statement hashes,*

(ii) configured verifier status, and (iii) a Merkle root R_t over sorted proof IDs. The ledger is monotone *if* $\bigcup_{i \le t} B_i \subseteq \bigcup_{i \le t+1} B_i$.

Monotonicity ensures that accepted knowledge only grows. Statements cannot be retracted; only new proof artifacts can be added.

2.3 Dual Attestation

At each epoch t, the system commits to two roots:

- r_t: Reasoning root over canonicalized proof artifacts
- u_t: UI root over interface state (DOM, logs, user confirmations)

These are bound by $H_t = \text{Hash}(\texttt{EPOCH:} \| r_t \| u_t)$ with prefix-free domain separation. The tuple (r_t, u_t, H_t) is the *epistemic fingerprint* of the epoch—the only scalar permitted as a summary of what occurred.

2.4 Governance-Bound Negative Knowledge

A critical requirement for fail-closed learning systems is that rejected, failed, or inadmissible events must not disappear silently. While such events must not influence learning or epistemic authority, they must remain cryptographically visible for audit, replay, and governance verification.

Negative Knowledge as Evidence. MathLedger therefore treats certain non-admitted outcomes as first-class, audit-grade artifacts. These include:

- **Refuted artifacts**: reasoning attempts that were explicitly rejected by the configured verifier;
- **Abstentions**: attempts that failed to satisfy verification criteria under bounded resources;
- **Inadmissible updates**: learning updates blocked by active governance predicates or frozen commitments.

These artifacts are *not* promoted to knowledge and do not enter the monotone ledger of verified statements. Instead, they are recorded as governance-bound evidence, preserving a verifiable record of what was attempted and explicitly not learned. This ensures that rejected or blocked updates constrain future interpretation and auditability without acquiring epistemic or learning authority.

Frozen Governance Commitments. Each experimental run is bound to a versioned *Governance Commitment Registry* (GCR), whose cryptographic hash is recorded in the run manifest. The registry enumerates non-negotiable constraints (e.g., claim ceilings, variance bounds, update admissibility rules) that are frozen for the duration of the run.

As a result, the system can support statements of the form:

> "This update did not occur because constraint C was active under governance version v for the run."

Such statements are verifiable via replay of the evidence pack and do not rely on post-hoc interpretation or informal policy descriptions.

Separation from Learning Authority. Negative-knowledge artifacts are explicitly excluded from Reflexive Formal Learning updates. RFL consumes only verifier outcomes summarized by $\mathcal{V}(e) \in \{1, 0, \perp\}$ as a *negative signal*, never as positive instructional content. Failed or blocked artifacts constrain admissibility but do not teach structure.

This separation preserves three invariants:

1. **Soundness**: invalid reasoning cannot contaminate policy updates;
2. **Auditability**: all epistemically relevant failures remain replayable;
3. **Non-silent governance**: no event may influence future authority without leaving a typed, cryptographically bound trace.

Relation to Dual Attestation. Dual attestation ($H_t = \text{Hash}(\texttt{EPOCH:}\|r_t\|u_t)$) remains the sole canonical commitment for learning and epistemic state. Governance-bound negative knowledge operates as an *orthogonal evidentiary layer*: additive, non-authoritative, and non-interventional in Phase I. This evidentiary layer is replay-verified and fail-closed, but it cannot escalate claims or authorize learning updates.

Threat Model. This architecture defends against *silent drift* (governance constraints changing without trace) and *policy laundering* (informal post-hoc reinterpretation of what was permitted). It does *not* protect against malicious verifier design, compromised registry authorship, or adversarial manipulation of the run environment itself. Phase I assumes an honest-but-fallible operator; Byzantine fault tolerance is out of scope.

Future phases may introduce explicit governance-state commitments, but no such extension is required for the Phase I claims reported here.

3 Reflexive Formal Learning: Formal Anchor

Reflexive Formal Learning (RFL) is a symbolic analogue of gradient descent operating on verification outcomes rather than numerical errors.

3.1 Core Definitions

Let Π be the space of symbolic reasoning policies and P_π the event distribution induced by policy π.

Definition 2 (Verification Outcome)**.** *For reasoning event e_t, the verifier produces:*

$$\mathcal{V}(e_t) \in \{1, 0, \perp\}$$

where 1 = verification passed, 0 = verification failed, $\perp$ = abstention.

Definition 3 (Epistemic Risk)**.** *The epistemic risk of policy π is:*

$$\mathcal{J}(\pi) = \mathbb{E}_{e\sim P_\pi}[\mathbf{1}\{\mathcal{V}(e) \neq 1\}] = \Pr_{e\sim P_\pi}[\mathcal{V}(e) \neq 1]$$

This measures the probability mass on non-verified events (failures and abstentions).

3.2 The RFL Update Rule

At each step t:

$$\pi_{t+1} = \pi_t \oplus \eta_t \cdot \Phi(\mathcal{V}(e_t), \pi_t) \tag{1}$$

where $\oplus$ is algebraic composition on policy space and $\Phi : \{1, 0, \perp\} \times \Pi \to \Delta\Pi$ maps verification outcomes to policy adjustments.

The intuition is:

> *Policies that cause fewer failures and abstentions become more likely; policies that cause them become less likely.*

Remark 1. *RFL has the mathematical structure of a stochastic approximation process (see Proposition 1 in Section 4). This does not claim convergence in finite time or under Phase I conditions; convergence requires additional stability assumptions that are not claimed here. Full proofs appear in Appendix B.*

3.3 Abstention as First-Class Outcome

Unlike reward-based systems that penalize abstention, RFL treats it as informative:

- Abstention prevents false positives (hallucinations committed to ledger)
- Abstention rates provide signal about policy quality

- High abstention with stable $\mathcal{J}(\pi)$ indicates the policy is appropriately cautious

This inverts the standard incentive structure: the system is rewarded for knowing what it does not know.

4 Formal Properties of the Substrate

To strengthen the theoretical rigor of Phase I, we state formal results for three key properties: (1) the RFL update rule as a stochastic approximation process, (2) the monotonicity and tamper-evidence of the ledger, and (3) the binding property of the dual attestation hash. Each result is stated under clear assumptions; full proofs appear in Appendix B.

4.1 RFL Update as Stochastic Approximation

Proposition 1 (RFL as Stochastic Approximation)**.** *Consider the RFL policy update $\pi_{t+1} = \pi_t \oplus \eta_t \Phi(\mathcal{V}(e_t), \pi_t)$, where $\Phi(\mathcal{V}(e_t), \pi_t)$ is the adjustment induced by verification outcome $\mathcal{V}(e_t) \in \{1, 0, \perp\}$ at time t, and $\eta_t > 0$ is the learning step size. Assume Π embeds locally into a normed vector space (or admits a coordinate chart) so that the additive form below is well-defined. Additionally assume:*

1. ***(Bounded updates)*** *There exists $L < \infty$ such that $\|\Phi(\mathcal{V}(e), \pi)\| \leq L$ for all events and policies.*

2. ***(Martingale noise)*** *The update deviations $M_{t+1} := \Phi(\mathcal{V}(e_t), \pi_t) - h(\pi_t)$ satisfy $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] = 0$ with bounded variance, where $h(\pi) := \mathbb{E}[\Phi(\mathcal{V}(e), \pi) \mid \pi]$.*

3. ***(Robbins–Monro stepsizes)*** *$\sum_{t=0}^{\infty} \eta_t = \infty$ and $\sum_{t=0}^{\infty} \eta_t^2 < \infty$.*

Under these conditions, working in the local coordinate chart where $\oplus$ corresponds to vector addition, the RFL recursion can be written in canonical stochastic approximation form:

$$\pi_{t+1} = \pi_t + \eta_t (h(\pi_t) + M_{t+1})$$

where M_{t+1} is a martingale-difference noise term. By classical stochastic approximation theory [6, 4, 1], this establishes that RFL has the mathematical structure of a learning algorithm. Convergence to an equilibrium requires additional stability assumptions (e.g., contraction of h) that are not claimed in Phase I.

4.2 Monotone Ledger and Tamper-Evidence

Proposition 2 (Monotonicity and Tamper-Evidence)**.** *Let $\mathcal{L} = (B_1, B_2, \ldots, B_T)$ be a ledger of sequential blocks, where each block B_t contains verifier-accepted proof artifacts. Define the knowledge state $K_t := \bigcup_{i=1}^{t} B_i$. Let L_t denote the ledger head hash after block t, computed as $L_t = \mathrm{Hash}(L_{t-1} \| R_t)$ where R_t is the Merkle root of B_t. Assume:*

1. *Blocks are append-only (no modification after appending).*
2. *The hash function is collision-resistant.*

Then:

1. ***(Monotonicity)*** *$K_t \subseteq K_{t+1}$ for all t. Accepted knowledge only grows.*
2. ***(Tamper-Evidence)*** *For any altered ledger $\tilde{\mathcal{L}} \neq \mathcal{L}$, the head hash $\tilde{L}_T \neq L_T$ except with negligible probability.*

4.3 Dual Attestation Binding

Lemma 1 (Binding Property of Dual Attestation)**.** *At each epoch t, the system commits to reasoning root r_t (32-byte digest) and UI root u_t (32-byte digest), then publishes $H_t = \mathrm{Hash}(\texttt{EPOCH:} \| r_t \| u_t)$. Under the assumption that the hash function is collision-resistant and the encoding uses fixed-width (32-byte) digests with prefix-free domain separation, the hash H_t binds the pair (r_t, u_t): it is computationally infeasible for any $(r'_t, u'_t) \neq (r_t, u_t)$ to produce the same H_t.*

Remark 2. *The fixed-width encoding (32-byte digests) eliminates concatenation ambiguity. The prefix `EPOCH:` provides domain separation from other hash uses in the system. Together with Proposition 2, this ensures every aspect of the system's state is tamper-evident and auditably linked to verifier-accepted proof artifacts.*

5 Phase I Experimental Results

This section presents Phase I experimental findings. The primary goal is validating measurement infrastructure and fail-closed governance, not demonstrating capability or convergence.

5.1 CAL-EXP-3: Measurement Validation

CAL-EXP-3 validates that Δp (success rate proxy) is computable per cycle and that variance between experimental arms is measurable. The experiment compares:

- **Baseline (lr=0.0):** RFL disabled; policy static
- **Treatment (lr=0.1):** RFL enabled; policy updated based on verifier outcomes

Both conditions exhibited oscillatory Δp dynamics around the decision threshold. No convergence or uplift is claimed; the purpose is infrastructure validation. Full time-series plots are available in the evidence pack (ancillary material).

5.2 Fail-Closed Governance

Separate stress tests confirm that governance predicates trigger correctly under out-of-bounds conditions:

- **F5.2 (variance ratio):** Fires when inter-arm variance exceeds threshold
- **F5.3 (windowed drift):** Fires when Δp drift exceeds tolerance

When triggered, these predicates cap the claim level at L0 (no capability claim). This is the expected behavior for Phase I stress tests.

5.3 Dual-Root Attestation

The Mirror Auditor confirmed the integrity of the dual-root attestation mechanism. For the `9bc8076` snapshot, coverage was 100.0%, with 100 blocks fully audited and verified.

5.4 Interpreting Phase I Outcomes

The Phase I results establish three facts:

1. **The measurement substrate works.** Δp (success rate proxy) is computable per cycle. Variance between arms is measurable.
2. **Fail-closed governance triggers correctly.** In stress tests, F5.2 (variance ratio out of bounds) and F5.3 (windowed drift excessive) fired as expected, capping claims at L0.
3. **Non-convergence is informative, not a failure.** Phase I was designed to validate infrastructure, not demonstrate capability. The fact that fail-close triggers fired correctly *is* the success condition.

6 Discussion: Why This Matters

Phase I experiments characterized the behavior of the MathLedger substrate under controlled conditions. The focus was validating measurement infrastructure and fail-closed governance, not demonstrating capability.

6.1 Comparison to Adjacent Work

MathLedger occupies a distinct position in the landscape of verifiable AI:

Approach	Learning Signal	Auditability	Fail-Closed
RLHF	Human preference	Low	No
Verifier-guided	Post-hoc filter	Medium	No
Proof-carrying code	None (static)	High	Yes
MathLedger (RFL)	**Verifier outcome**[†]	**High**	**Yes**

Table 1: Comparison of approaches to reliable AI. MathLedger uniquely combines verified learning signals with fail-closed governance. [†]Phase I uses a synthetic proxy verifier; formal proof verification (Lean) is Phase II+.

6.2 Layer-3 Infrastructure

MathLedger is not a proof generator or a user-facing application. It is *Layer-3 infrastructure*: the flight data recorder for AI reasoning.

- **Layer 1 (Human):** Users pose queries, interpret results, make decisions
- **Layer 2 (Engine):** AI models generate formal artifacts
- **Layer 3 (Ledger):** MathLedger provides immutable provenance and attestation

The system does not compete with proof generators; it makes their outputs trustworthy at scale.

7 Explicit Non-Claims and Scope Boundaries

To maintain epistemic discipline, we explicitly state what Phase I does *not* establish:

7.1 What Phase I Does NOT Establish

- **Capability claims:** No claim that the system "understands" or "reasons" in any general sense.
- **Convergence:** No claim that RFL converges under Phase I conditions. All runs failed the variance gate.
- **Threshold validity:** Thresholds are frozen parameters, not validated optima.
- **Generalization:** No out-of-distribution testing was performed.
- **Real-world applicability:** Only synthetic harness data was used.

7.2 SHADOW Mode Semantics

All Phase I experiments operate in SHADOW mode: verification results are *observational and non-blocking*. The system records what happened but does not gate production decisions. In this context, "fail-closed" means claim-capping and evidence-rejection, not production blocking—governance predicates cap the claim level at L0 when triggered, but do not halt execution.

7.3 Phase Quarantine

Phase I and Phase II are strictly separated:

- **Phase I:** Assumes ideal verifier, hermetic environment, synthetic data
- **Phase II:** Tests governance stability under auxiliary perturbation (frozen but not executed)

No Phase II claims are made in this work. Phase II specification is frozen pending execution authorization.

8 Future Work

Future work will focus on integrating RFL to observe if reflexive feedback can dampen oscillatory states in the decision boundary and achieve measurable reductions in abstention rates and improvements in convergence latency.

8.1 Phase II Calibration

Phase II of the calibration program addresses governance stability: specifically, whether the governance verdict (failure codes, claim level) is invariant under perturbation of auxiliary parameters not part of the frozen predicate set. The Phase II specification is frozen, but execution has not yet occurred. No claims regarding governance invariance or sensitivity are made in this work. Phase II results, when available, will be reported separately and will not retroactively modify the Phase I conclusions presented here.

9 Conclusion

MathLedger demonstrates that ledger-attested learning is technically feasible. Phase I successfully established:

1. A working pipeline from proof generation through dual attestation to policy feedback
2. Measurement infrastructure for Δp and variance metrics

3. Fail-closed governance that correctly triggers under out-of-bounds conditions

4. Explicit non-claims and scope boundaries that enable honest assessment

The contribution is infrastructural, not empirical. We have built the substrate; demonstrating capability on that substrate is future work.

The system stands as proof-of-concept for a new paradigm: *learning from verifier-attested outcomes*. Whether this paradigm scales to complex reasoning remains an open question. What Phase I establishes is that the question can now be asked with rigor.

A Evidence Pack

The evidence pack provides cryptographic verification of experimental runs. Key artifacts:

Artifact	**Contents**
Evidence Manifest	File list with SHA-256 hashes (JSON)
Run Metadata	Experiment configuration and timing (JSON)
Governance Verdict	Claim level and predicate outcomes (JSON)

Table 2: Key experimental artifacts. Exact paths and SHA-256 hashes provided in ancillary material.

The complete evidence pack (run manifests, cryptographic hashes, raw Δp time series) will be published as ancillary material with this submission.

Governance binding in the evidence manifest. In addition to file hashes, the evidence manifest records:

1. a SHA-256 hash of the active Governance Commitment Registry (GCR), computed over RFC 8785-style canonical JSON (keys sorted lexicographically, no whitespace, ASCII-safe encoding);

2. a per-artifact classification tag (`artifact_kind`) with values: `VERIFIED`, `REFUTED`, `ABSTAINED`, or `INADMISSIBLE_UPDATE`.

The replay verifier checks these fields fail-closed: (i) missing or invalid `artifact_kind` enum values, (ii) missing `commitment_registry_sha256` field, and (iii) mismatched registry file hash all cause verification failure with exit code 1. This makes governance constraints and rejected updates cryptographically visible without elevating them to knowledge claims.

Scope disclaimer. The evidence pack verifies artifact integrity, determinism, and governance binding only. It does *not* validate correctness, safety, alignment, or legal compliance. The governance commitments in the registry are illustrative placeholders in v0.9.x; the mechanism (hash binding) is what is being validated, not the normative content.

Version pinning. The external audit surface for Phase I corresponds to Git tag `v0.9.4-pilot-audit-hardened`. The canonical verification command is:

```
uv run python scripts/run_dropin_demo.py --seed 42 --output demo_output/
cd demo_output && python verify.py
```

Expected test vectors (SHA-256 hashes for seed=42) are documented in `docs/pilot/AUDIT_WALKTHROUGH.md` within the tagged release.

B Formal Proofs

This appendix provides complete proofs for the formal properties stated in Section 4. Stronger convergence and robustness results under additional assumptions are developed in a separate technical companion and are intentionally excluded here to preserve Phase I scope.

B.1 Proof of Proposition 1 (RFL as Stochastic Approximation)

Proof. The update $\pi_{t+1} = \pi_t \oplus \eta_t \Phi(\mathcal{V}(e_t), \pi_t)$ can be interpreted as an additive update in a suitable parameterization. Define $h(\pi) := \mathbb{E}[\Phi(\mathcal{V}(e), \pi) \mid \pi]$, the expected update given the current policy. Define the noise term:

$$M_{t+1} := \Phi(\mathcal{V}(e_t), \pi_t) - h(\pi_t)$$

By construction, $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] = h(\pi_t) - h(\pi_t) = 0$, so M_{t+1} is a martingale difference adapted to $\mathcal{F}_t$. The update becomes:

$$\pi_{t+1} = \pi_t + \eta_t \big(h(\pi_t) + M_{t+1}\big)$$

This is the canonical Robbins–Monro stochastic approximation form. Under assumptions (bounded updates, martingale noise with bounded variance, Robbins–Monro stepsizes), standard SA theory applies. The function h plays the role of the mean-field drift.

We emphasize: this establishes that RFL has SA *structure*. Convergence to an equilibrium of $\dot{\pi} = h(\pi)$ requires that such an equilibrium exists and is attractive (e.g., h is a contraction). These additional stability conditions are not claimed in Phase I. □

B.2 Proof of Proposition 2 (Monotonicity and Tamper-Evidence)

Proof. **(1) Monotonicity:** By definition, $K_t = \bigcup_{i=1}^{t} B_i$. When block B_{t+1} is appended:

$$K_{t+1} = \bigcup_{i=1}^{t+1} B_i = K_t \cup B_{t+1} \supseteq K_t$$

Since blocks are append-only, no element of K_t is removed. Thus $K_t \subseteq K_{t+1}$.

(2) Tamper-Evidence: Suppose an adversary produces $\tilde{\mathcal{L}} = (\tilde{B}_1, \ldots, \tilde{B}_T) \neq \mathcal{L}$ with the same head hash $\tilde{L}_T = L_T$. Let j be the smallest index where $\tilde{B}_j \neq B_j$.

Case A: If $\tilde{B}_j$ differs from B_j as a set, then Merkle root $\tilde{R}_j \neq R_j$ (deterministic construction). Given $L_j = \text{Hash}(L_{j-1} \| R_j)$ and $\tilde{L}_j = \text{Hash}(L_{j-1} \| \tilde{R}_j)$ (assuming prior blocks match), we have $\tilde{L}_j \neq L_j$ unless a hash collision occurs. By collision resistance, this happens with negligible probability.

Case B: If the sequence lengths differ (block omitted or inserted), the hash chain incorporates a different number of blocks, yielding $\tilde{L}_T \neq L_T$ by similar reasoning.

In both cases, $\tilde{L}_T = L_T$ implies a hash collision, which is computationally infeasible. □

B.3 Proof of Lemma 1 (Dual Attestation Binding)

Proof. The hash input is $m = \texttt{EPOCH:} \| r_t \| u_t$, where r_t and u_t are fixed-width 32-byte digests. This encoding is unambiguous: the prefix `EPOCH:` is a fixed string, and the 32-byte widths mean there is a one-to-one correspondence between pairs (r_t, u_t) and input strings m.

Suppose $(r'_t, u'_t) \neq (r_t, u_t)$ yields the same hash:

$$\text{Hash}(\texttt{EPOCH:} \| r'_t \| u'_t) = \text{Hash}(\texttt{EPOCH:} \| r_t \| u_t)$$

Let $m' = \texttt{EPOCH:} \| r'_t \| u'_t$ and $m = \texttt{EPOCH:} \| r_t \| u_t$. Since the pairs differ and encoding is bijective, $m' \neq m$. Thus we have a hash collision, which is infeasible under collision resistance.

Therefore, H_t uniquely commits to (r_t, u_t). Once published, the agent cannot claim a different pair without finding a collision. □

References

[1] Vivek S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.

[2] Kevin Buzzard, Johan Commelin, and Patrick Massot. Formalising perfectoid spaces. *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 299–312, 2020.

[3] John Harrison. Formal proof—theory and practice. *Notices of the American Mathematical Society*, 55(11):1395–1406, 2008.

[4] Harold J. Kushner and G. George Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, 2nd edition, 2003.

[5] Gary Marcus and Ernest Davis. Gpt-3, bloviator: Openai's language generator has no idea what it's talking about. *MIT Technology Review*, 2020.

[6] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.