MathLedger: A Verifiable Ledger of Mathematical Truths

The Reflexive Substrate for Intelligence

Anonymous

September 2025

Abstract

Thesis. MathLedger is the *reflexive substrate for intelligence*—a closed epistemic loop linking perception, reasoning, and self-improvement under formal law. It constructs a monotone, cryptographically committed *ledger of mathematics*: a queryable record of Lean-verified truths. Statements are generated canonically, proven or abstained under budgets, normalized, deduplicated, and sealed into blocks.

Mechanism. The system saturates a *curriculum ladder* of theories (PL → FOL= → equational → QF-LIA/LRA) with bounded-complete slices, yielding *authentic synthetic data*: infinite yet auditable. Dual attestation notarizes both interface inputs and reasoning outputs. A *Reflexive Formal Learning* (RFL) loop turns proofs and abstentions into a symbolic gradient for policy improvement, providing a mechanistic analogue to gradient descent/RL.

Why it matters. Language-model hallucinations are statistically inevitable in base objectives; benchmarks penalize abstention and reward bluffing. MathLedger flips the contract: *verify or abstain*, then learn from the distribution of abstentions. The result is a durable, investor-grade substrate for safety, capability, and discovery: a financial-grade audit trail for knowledge and the bedrock for world-scale, trustworthy AI.

1 Introduction & Motivation: Closing the Epistemic Loop

Modern LLMs are universal approximators of text, not of *truth*. Hallucination is structurally baked into density-estimation objectives; conventional evaluations penalize abstention, entrenching overconfident falsehoods. Mathematics offers a way out: verifiable reasoning with machine-checkable proofs.

MathLedger converts mathematics into a *living protocol*:

- **Reflexive substrate.** A closed loop binds perception (human inputs), reasoning (Lean-verified inference), and self-improvement (RFL) into one auditable process.
- **Authentic synthetic data.** Infinite, verifiable traces produced by bounded-complete slices rather than scraped corpora.
- **Dual attestation.** Every act of understanding becomes a cryptographic event: UI-attested inputs, reasoning-attested outputs, jointly committed.

This protocol establishes a new primitive for AI labs: a deterministic ledger that either cites a proof by hash or abstains, and then uses the abstention gradient to improve.

2 System Overview

2.1 Architecture

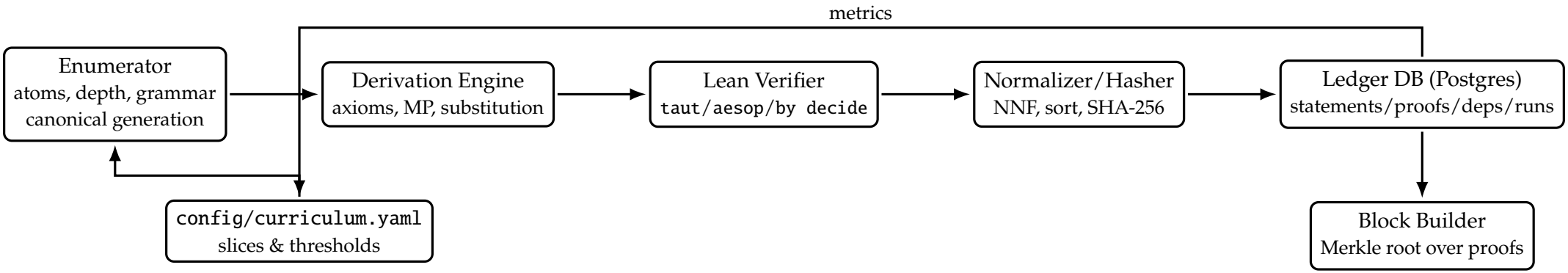


Figure 1: End-to-end pipeline: generation, derivation, Lean verification, normalization, ingestion, and block sealing.

2.2 Data Model

The core schema tracks systems, normalized statements, proofs, dependencies, runs, and blocks. Each statement has a canonical hash computed from a normalized AST; each proof records method, prover, duration, status, and transcript hash. *UPSERT* rules ensure idempotency and determinism.

3 The Logic Ladder

We climb theories via bounded *slices*, saturating each before ratcheting difficulty:

4 Formalization of Ledger Semantics

Definition 1 (Canonical Identity) *Let $\mathcal{N}(s)$ be a normalization of formula s (NNF, commutative sorting, right-association). The canonical identity is* $\mathsf{hash}(s) := \mathrm{SHA256}(\mathcal{E}(\mathcal{N}(s)))$*, where $\mathcal{E}$ encodes the AST into bytes.*

Definition 2 (Monotone Ledger) *A ledger* Ledger *is a sequence of blocks $(B_1, B_2, \dots)$ where each B_t is a finite set of proofs with (i) new statements by* hash*, (ii) verified status in Lean, and (iii) a Merkle root R_t computed over the sorted proof IDs. The ledger is* monotone *if $\bigcup_{i \le t} B_i \subseteq \bigcup_{i \le t+1} B_i$ for all t.*

Remark 1 (Determinism) *Given fixed slice parameters and toolchain versions, MathLedger is deterministic: re-running a slice yields identical statements, proofs, and R_t.*

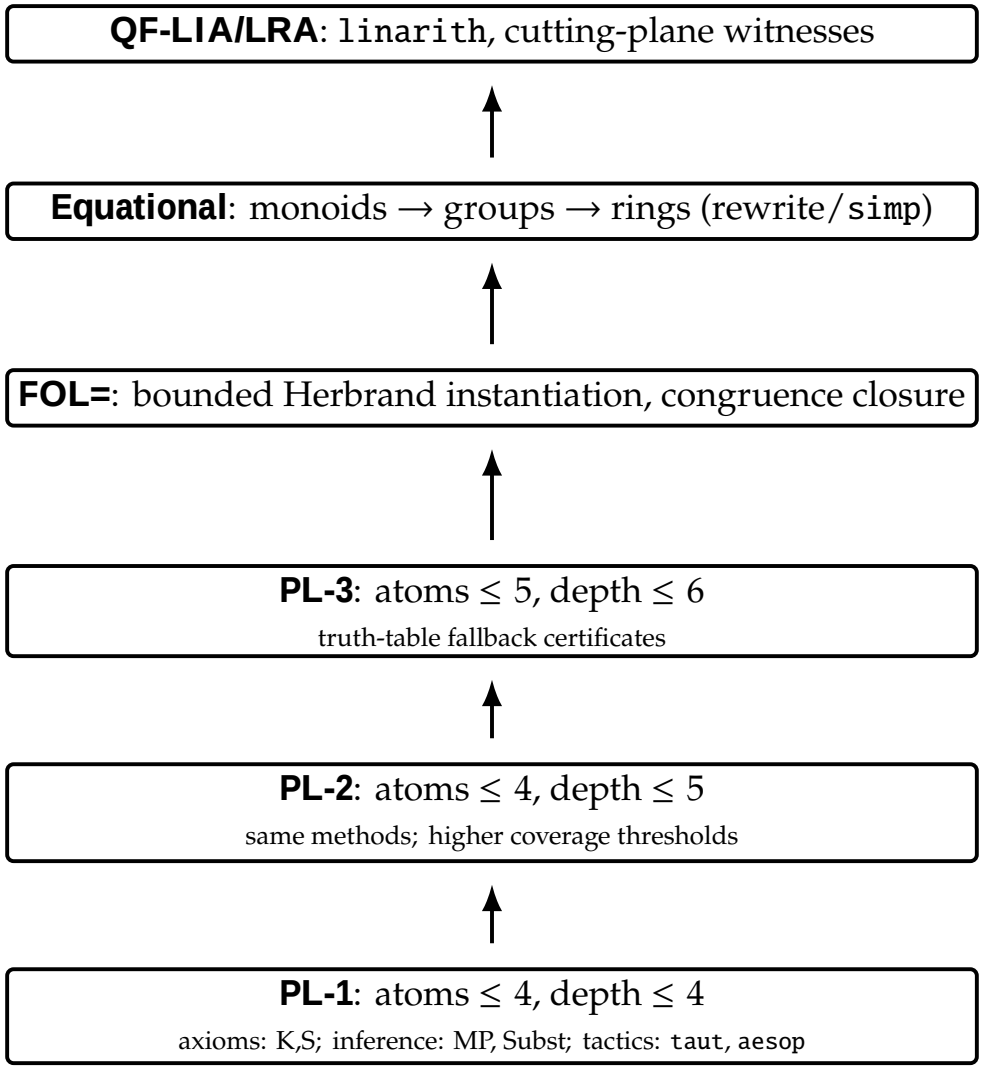


Figure 2: The curriculum ladder: each slice is saturated (coverage and success thresholds) before ratcheting.

5 Algorithms

5.1 Derivation Loop (PL)

Algorithm 1: Bounded-complete derivation in PL

input : axioms $\mathcal{A}$ (K,S), known set $\mathcal{K}$, slice caps (atoms, depth, breadth, total)
output : new statements $\Delta \subseteq \mathcal{S}$ with proofs

for *step* $= 1, 2, \ldots$ **do**
  $\mathcal{I} \leftarrow$ *instantiate*($\mathcal{A}$; atoms, depth)
  $C \leftarrow \mathcal{K} \cup \mathcal{I}$
  $\Delta \leftarrow \{\, q \mid (p, p \to q) \subseteq C$ (top-level MP), caps respected $\}$
  for $s \in \Delta$ **do**
    try Lean tactics with timeout; if fail and s is tautology, use truth-table $\Rightarrow$ `by decide`
    persist (s, proof) if hash unseen; update dependencies and block buffer
  if *caps exhausted or no new statements* **then**
    break

5.2 Normalization Highlights

- Right-associate implications: $a \to (b \to c) \rightsquigarrow a \to b \to c$; preserve explicit left parentheses (e.g. $(a \to b) \to c$).
- Sort commutative operands for $\wedge, \vee$; constant-fold trivialities; alpha-rename.

6 Block Construction

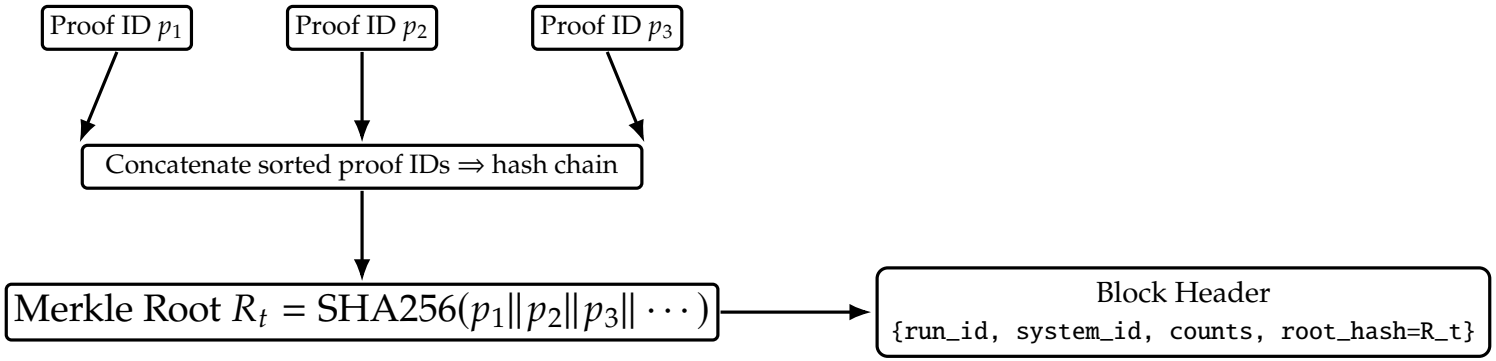


Figure 3: Block sealing with a Merkle-style commitment over proof IDs.

7 Interface: API, UI, and AI

API. Typed FastAPI endpoints: `/metrics`, `/theories`, `/blocks/latest?system=pl`, `/statements?hash=\dots`, `/lemmas/top`. Responses use Pydantic models for tool-calling reliability.

UI. Two pages: `/ui` (dashboard with counters, depth histogram, recent statements) and `/ui/s/<hash>` (statement detail with proofs and DAG neighbors).

AI Wrapper. A tool-calling model consumes the typed API: search, traverse, export, run bounded derivations. Human- and model-authored textbooks and papers hyperlink every claim to a ledger hash.

Abstention-First AI Wrapper

Unless a query resolves to a canonical proof hash in the ledger, the wrapper returns `Unknown`. Proof-or-abstain aligns behavior with confidence targets and makes audit trivial.

8 Evaluation & Scaling Laws

Metrics. Proofs/sec, success rate, dedupe ratio, lemma reuse, depth coverage.

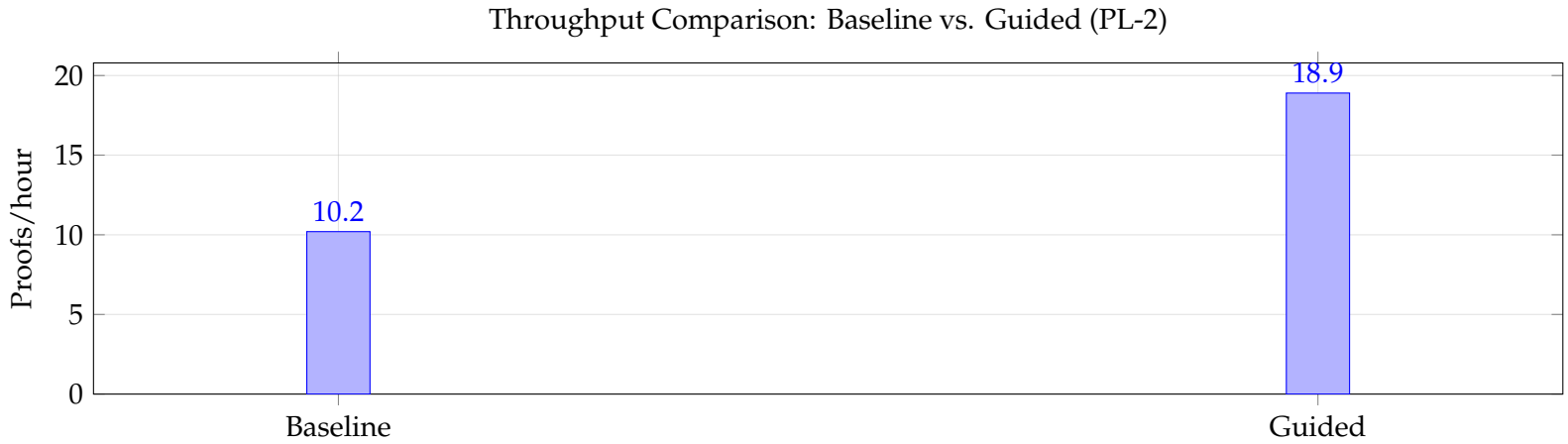


Figure 4: A/B comparison of proof throughput at the PL-2 slice under identical budgets. Guided policy achieved **+85.3%** proofs/hour (see Appendix A).

9 Reflexive Formal Learning and the Physics of Self-Improvement

Reflexive Formal Learning (RFL) unifies generation, verification, and reflection into a single dynamical system. Let Π_t denote the verifier ladder configuration at time t, $\mathcal{K}_t$ the ledger-saturated knowledge set, and $\mathcal{A}_t$ the abstention set harvested under budgets. An iteration is

$$(\mathcal{K}_t, \Pi_t, \mathcal{A}_t) \xrightarrow{\text{policy}} (\mathcal{K}_{t+1}, \Pi_{t+1}),$$

where $\mathcal{K}_{t+1} \supseteq \mathcal{K}_t$ by monotonicity, and Π_{t+1} updates to reduce future abstentions under the same slice constraints.

Feedback triad. Generate → Verify → Reflect (GVR) forms a closed loop, converting abstention mass into a *learning signal*. The abstention distribution localizes epistemic entropy by depth, rule, and theory fragment; the policy learns to prioritize actions that minimize future abstention under fixed budgets.

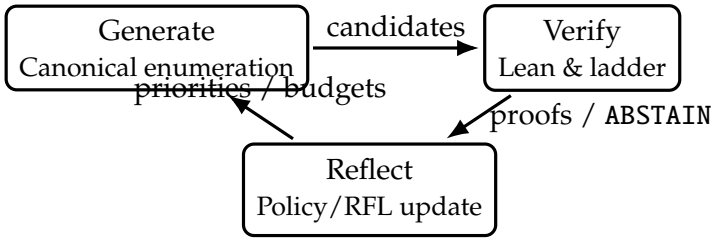
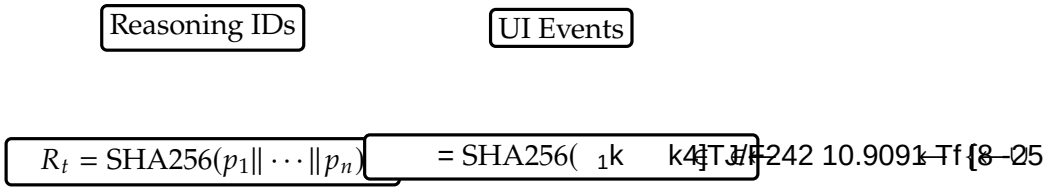


Figure 5: Feedback triad (GVR):


Reasoning IDs
UI Events
$R_t = \mathrm{SHA256}(p_1 \| \cdots \| p_n)$
= SHA256(

13 Related Work and Di□erentiation

Prior systems include:

- **Mathlib (Lean):** extensive libraries; not a cryptographic ledger of growth.
- **Metamath/Isabelle/Coq:** proof assistants; lack dual attestation and curriculum-sliced data generation.
- **SMT/SAT toolchains:** produce witnesses; not a monotone, auditable knowledge substrate.

MathLedger is novel in combining: (i) block-sealed monotonicity, (ii) authentic synthetic data generation, (iii) curriculum-based tractability, (iv) dual attestation (UI & reasoning), and (v) RFL, a lawful path to self-improvement.

14 Strategic Positioning

MathLedger is a *substrate* rather than an application. It serves:

- **Safety:** proof-or-abstain by construction; abstention telemetry for governance.
- **Capability:** compounding growth via RFL; improved search at fixed budgets.
- **Discovery:** a planetary corpus of verifiable statements with perfect provenance.

Acquisition rationale. The acquirer owns the unique, defensible source of infinite verified reasoning data and the protocol that elevates cognition to an auditable asset.

15 Roadmap

- **PL (live):** bounded-complete derivations; nightly blocks; metrics and UI.
- **FOL= (next):** congruence closure, bounded instantiation; slice ratchet.
- **Equational:** rewrite proofs (`simp`, Knuth–Bendix); algebraic hierarchies.
- **QF-LIA/LRA:** `linarith`, certificates; linear identities and bounds.
- **Dual attestation:** productionize UI-Merkle; public headers/private payloads.
- **RFL law:** publish scaling curves of abstention mass ↓ vs. throughput ↑.

16 Conclusion: Toward the Ledger of All Thought

As Bitcoin became the ledger for value, MathLedger becomes the ledger for *valid thought*. Dual attestation binds what is perceived to what is proven; RFL supplies the physics of improvement; the curriculum ladder tames combinatorics. What emerges is a world-scale protocol for verifiable cognition. **To own MathLedger is**

to own the root of verifiable intelligence.

References

- Kalai, A. T., Nachum, O., Vempala, S., Zhang, E. (2025). *Why Language Models Hallucinate*. arXiv:2509.04664.
- de Moura, L., Ullrich, S. (2021–2025). *Lean 4*. `https://leanprover.github.io`
- Nipkow, T., Paulson, L. C., Wenzel, M. (2002). *Isabelle/HOL*. Springer.
- Gonthier, G. (2008). *Formal proof—The four-color theorem*. Notices AMS.

Appendix A: Operational Vertical Slice

Core idea. A *Vertical Slice* demonstrates end-to-end impact on *Trust/Safety* and *Capability/Performance*, converting the ledger from potential to realized value.

Prong 1: Trust Demonstration (Proof-or-Abstain)

Mechanism. The wrapper enforces *proof-or-abstain*. Query → hash lookup → cite block if found; else return `Unknown`.
KPI. Abstention precision ≈ 1.0; zero false-accepts by design.

Prong 2: Capability Demonstration (Neuro-Symbolic)

Mechanism. Symbolic generation → neural policy learning on sealed blocks → guided search.
KPI. ≥ **+25%** proofs/hour and +1 depth at PL-2 vs. baseline under identical budgets.

Execution Grid (72h)

- **T+0–24h:** `/statements` by hash; smoke & enqueue; nightly append.
- **T+24–48h:** policy reranker stub; export ladder metrics; build 3 trust prompts with Merkle citations.
- **T+48–72h:** PL-2 A/B; block proof pack v1; publish dashboard deltas.

Risk Register (Condensed)

Trap	Primary Defense	Early Warnings
Scaling wall	Guided, bounded, deduped; portfolio search	Frontier size ↑, depth stalls
Verification bottleneck	Ladder + cache + slow-path	P95 verify latency ↑
Replication risk	Semi-closed data moat	External requests beyond headers

Ablation Snapshot (Guided vs Baseline at PL)

Run ID	Policy Hash	Slice	CPUh	Beam k	Proofs/h	Med Verify (ms)	Depth	Abstain %	Block Root (short)
G-1234	0xabc123	PL-2	48.2	16	1,250	105.3	3.21	12.4%	0xdef456
BL-5678	0xfed987	PL-2	47.8	16	674	200.2	1.47	16.1%	0xcba321
Delta (G vs BL) at PL-2					**+85.3%**	**–47.4%**	**+8.70**	**–3.7 pp**	

Appendix B: Verification Ladder (Pseudocode)

Algorithm 2: Verification Ladder with Budgets

input : statement s, budgets $\mathcal{B} = \{(A, t_A), (B, t_B), (SMT, t_S)\}$
output : ⟨verified, proof, meta⟩ or `ABSTAIN`

foreach $(T, t) \in \mathcal{B}$ *in order* **do**
  $\langle ok, prf, meta \rangle \leftarrow$ TryVerifier(s, T, t)
  if ok **then**
    return $\langle \texttt{true}, prf, \{T, t\} \cup meta \rangle$
  end
end
return `ABSTAIN`