

# MathLedger: A Verifiable Learning Substrate with Ledger-Attested Feedback

The MathLedger Research Fleet

December 20, 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 verified proof events rather than statistical loss.

Phase I experiments validate the measurement and governance substrate under controlled conditions. Phase I includes both (i) measurement validation runs (CAL-EXP-3, achieving L4 claim level via synthetic  $\Delta p$  proxy) and (ii) fail-close stress tests (CAL-EXP-4/5). 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 binary decision becomes the learning signal itself*. Every update is justified by a verifier outcome (proxy in Phase I; formal proof verification in Phase II+), or explicit abstention, recorded in an immutable ledger. This creates a closed epistemic loop where learning is constrained to verified truth.

## 1.2 The Chain of Verifiable Cognition

The system implements an end-to-end pipeline:

Input  $\rightarrow$  Proof-or-Abstain  $\rightarrow$  Ledger Attestation  $\rightarrow$  Dual Commitment  $\rightarrow$  Policy Update

Each component is cryptographically bound:

- **Proof-or-Abstain:** A configured verifier checks reasoning or the system explicitly abstains. No middle ground. (*In Phase I, the verifier is instantiated as a synthetic proxy; Lean kernel integration is Phase II+.*)
- **Ledger Attestation:** Verifier outcomes (proxy in Phase I; proofs in Phase II+) 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:  $H_t = \text{Hash}(\text{EPOCH} : \|r_t\|u_t)$ .
- **Policy Update:** Reflexive Formal Learning (RFL) adjusts the policy based on verification outcomes.

**Phase I scope:** In Phase I, the verifier-gated loop is exercised using a synthetic success proxy ( $\Delta p$ ); formal Lean proof verification is an integration target for Phase II+.

This architecture enables a new primitive: *learning from verified truth rather than statistical loss*.

### 1.3 What Is Genuinely New

MathLedger combines three elements not commonly integrated in a single end-to-end system [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 learning proceeds, 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 Experimental Methodology

### 2.1 Pipeline Overview

The CAL-EXP-3 harness executes the RFL loop with a synthetic verifier proxy, holding all other parameters constant.

### 2.2 Test Harness: CAL-EXP-3

CAL-EXP-3 is a *synthetic alignment-success simulation*. It does **not** invoke Lean verification. Instead, the verifier outcome  $\mathcal{V}(e_t)$  is instantiated as a synthetic success probability  $\Delta p \in [0, 1]$ , allowing us to test the measurement and governance infrastructure in isolation.

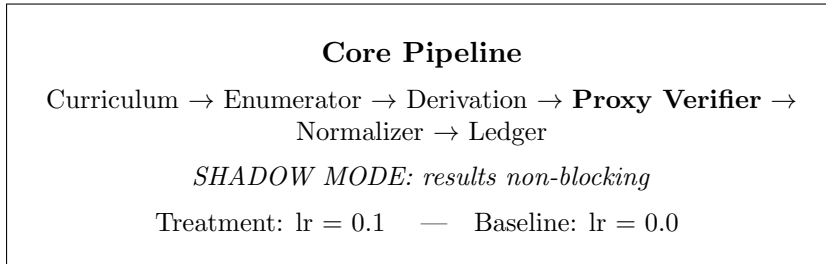


Figure 1: CAL-EXP-3 pipeline. The proxy verifier replaces Lean in Phase I. SHADOW MODE: verification results are observational only.

## 2.3 Configuration

We compare exactly two conditions:

- **Treatment** (lr=0.1): RFL feedback enabled
- **Baseline** (lr=0.0): No feedback (control arm)

All other parameters are frozen: 1000 cycles, 200 warm-up excluded, fixed random seeds (42, 43, 44).

## 2.4 Metrics

- $\Delta p$ : Synthetic task success probability proxy. *This is not Lean verification success.*
- $\Delta \Delta p$ : Difference in mean  $\Delta p$  between treatment and baseline arms.
- **Variance/drift checks**: CAL-EXP-3 uses validity checks F1.x–F4.x. Variance predicates F5.2/F5.3 are tested only in CAL-EXP-4/5.

## 2.5 The Monotone Ledger

**Definition 1** (Monotone Ledger). *A ledger  $\mathcal{L}$  is a sequence of blocks  $(B_1, B_2, \dots)$  where each  $B_t$  contains verified proofs with: (i) canonical statement hashes, (ii) verification status from the configured verifier (proxy in Phase I; Lean in Phase II+), and (iii) a Merkle root  $R_t$  over sorted proof IDs. The ledger is monotone if  $\bigcup_{i \leq t} B_i \subseteq \bigcup_{i \leq t+1} B_i$ .*

Monotonicity ensures that verified knowledge only grows. Statements cannot be retracted; only new proofs can be added.

## 2.6 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  $H_t = \text{Hash}(\text{EPOCH} \parallel r_t \parallel u_t)$  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.

## 2.7 Formal Properties of the Ledger

**Proposition 1** (Ledger Monotonicity and Tamper-Evidence). *Consider a monotone ledger  $\mathcal{L} = (B_1, B_2, \dots)$  where each block  $B_t$  contains a set of proof entries (with unique statement hashes) and a cryptographic link to the previous ledger state. Assume: (1) Append-only: new blocks are only appended, never modifying or deleting prior content; (2) Hash-chain integrity: each  $B_t$  includes a collision-resistant hash pointer (in the standard cryptographic sense) to the previous ledger state; and (3) Deterministic contents hash: each block  $B_t$  also stores a collision-resistant digest of its own contents (e.g. a Merkle root of the sorted proof hashes in  $B_t$ ). Under these conditions, the ledger exhibits two properties:*

- (i) **Monotonicity:** *Let  $K_t := \bigcup_{i=1}^t B_i$  be the cumulative set of all proofs committed up to block  $t$ . Then  $K_t \subseteq K_{t+1}$  for all  $t \geq 1$ , i.e. the knowledge only grows with each appended block.*
- (ii) **Tamper-Evidence:** *The ledger is cryptographically tamper-evident. If an adversary alters or omits any proof in any earlier block  $B_j$  ( $j \leq t$ ), the hash-chain condition ensures that the final ledger digest at block  $t$  will differ from the honest digest (unless a hash collision is found). No adversary can produce an alternative sequence  $\mathcal{L}' = (B'_1, \dots, B'_t)$  with  $B'_j \neq B_j$  for some  $j$  yet  $\mathcal{L}'$  ending in the same final hash as  $\mathcal{L}$ , except with negligible (collision) probability.*

*Proof.* (i) Monotonicity holds by construction. Since  $\mathcal{L}$  is append-only,  $B_{t+1}$  adds new proof entries without removing any from prior blocks. Thus  $K_{t+1} = K_t \cup B_{t+1}$ , which clearly implies  $K_t \subseteq K_{t+1}$ .

(ii) *Tamper-evidence:* By standard hash-chain arguments, any deviation at index  $j$  propagates forward and alters all subsequent roots. If an adversary produces an alternate ledger  $\mathcal{L}' = (B'_1, \dots, B'_t)$  that differs from  $\mathcal{L}$  in at least one block, then for the earliest index  $j$  where  $B'_j \neq B_j$ , the Merkle root differs, which propagates through the chain. By collision-resistance, the final digest  $R'_t$  for the adversary's ledger differs from the honest  $R_t$ . Thus no tampering can escape detection unless the hash function itself is broken.  $\square$

**Lemma 1** (Binding of Dual Attestation). *Let  $H$  be a collision-resistant hash function, and consider the joint epoch commitment  $H_t = H(\text{EPOCH} : \| r_t \| u_t)$  which binds the reasoning root  $r_t$  and UI root  $u_t$  at epoch  $t$ . This hash value serves as a binding commitment to the ordered pair  $(r_t, u_t)$ : if  $(r', u') \neq (r_t, u_t)$ , then*

$$H(\text{EPOCH} : \| r' \| u') \neq H(\text{EPOCH} : \| r_t \| u_t),$$

*except with negligible probability of hash collision. Without finding a collision in  $H$ , an adversary cannot produce an alternative  $(r', u')$  that yields the same  $H_t$ .*

*Proof.* Because the encoding is prefix-free (the literal tag “EPOCH:” and the concatenation define a unique bitstring for each ordered pair), two different pairs  $(r', u') \neq (r_t, u_t)$  produce distinct inputs to the hash function. By the collision-resistance of  $H$ , these distinct inputs cannot yield the same output. Thus  $H_t$  serves as a secure commitment to the exact pair  $(r_t, u_t)$ .  $\square$

### 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  $\Pi$  be the space of symbolic reasoning policies and  $P_\pi$  the event distribution induced by policy  $\pi$ .

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

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

where  $1 = \text{verified proof}$ ,  $0 = \text{failed proof}$ ,  $\perp = \text{abstention}$ .

**Definition 3** (Epistemic Risk). *The epistemic risk of policy  $\pi$  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  $\Phi : \{1, 0, \perp\} \times \Pi \rightarrow \Delta\Pi$  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.*

**Proposition 2** (RFL as Stochastic Approximation). *Assume the Reflexive Formal Learning (RFL) update rule (1) satisfies standard conditions: (i) the policy update mapping  $\Phi(v, \pi)$  is bounded for all  $\pi \in \Pi$  and all verification outcomes  $v \in \{1, 0, \perp\}$ ; (ii) at each step  $t$ , the event  $e_t$  is drawn from the distribution  $P_{\pi_t}$  (with  $\pi_t$  measurable w.r.t. the filtration  $\mathcal{F}_t$  of past events); and (iii) the step-size sequence  $\{\eta_t\}$  is positive, non-increasing, and satisfies*

$\sum_t \eta_t = \infty$  and  $\sum_t \eta_t^2 < \infty$ . Then the RFL recursion can be written in the canonical stochastic approximation form:

$$\pi_{t+1} = \pi_t \oplus \eta_t \Phi(\mathcal{V}(e_t), \pi_t) = \pi_t + \eta_t \left[ h(\pi_t) + M_{t+1} \right],$$

where we define the expected update  $h(\pi_t) := \mathbb{E}_{e \sim P_{\pi_t}}[\Phi(\mathcal{V}(e), \pi_t)]$  and the zero-mean fluctuation  $M_{t+1} := \Phi(\mathcal{V}(e_t), \pi_t) - h(\pi_t)$ . By construction  $\{M_{t+1}\}$  is a martingale-difference noise sequence (i.e.  $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] = 0$ ). Here we implicitly embed policies  $\pi \in \Pi$  into a vector space representation (e.g., logits or parameters) where the additive SA form is well-defined; the abstract operator  $\oplus$  denotes this representation-dependent composition. In particular, the evolution of  $\pi_t$  is a stochastic approximation process with mean-field dynamics corresponding to the epistemic risk functional  $\mathcal{J}(\pi)$ .

*Proof.* Under assumption (ii), conditioned on  $\pi_t$  the sample  $e_t \sim P_{\pi_t}$  yields  $\mathbb{E}[\Phi(\mathcal{V}(e_t), \pi_t) \mid \mathcal{F}_t] = \mathbb{E}_{e \sim P_{\pi_t}}[\Phi(\mathcal{V}(e), \pi_t)] = h(\pi_t)$ . Therefore  $M_{t+1} := \Phi(\mathcal{V}(e_t), \pi_t) - h(\pi_t)$  has conditional expectation 0 given  $\mathcal{F}_t$ , which means  $M_{t+1}$  is a martingale-difference term. The RFL update thus matches the form  $x_{t+1} = x_t + \eta_t (h(x_t) + M_{t+1})$  with  $x_t \equiv \pi_t$ ,  $h(x_t) = h(\pi_t)$ , and noise  $M_{t+1}$  satisfying the standard adaptivity condition ( $M_{t+1}$  is  $\mathcal{F}_{t+1}$ -measurable and  $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] = 0$ ). Assumption (i) ensures  $h(\pi)$  is well-defined and (under mild continuity conditions) typically Lipschitz on  $\Pi$ , and assumption (iii) gives the usual Robbins–Monro step-size conditions. Hence, by the classical theory of stochastic approximation [6, 4, 1], the RFL dynamics  $\pi_t$  constitute a stochastic approximation algorithm. We emphasize that Proposition 2 is a structural claim—convergence in Phase I is neither required nor asserted here.  $\square$

**Remark 1.** Proposition 2 establishes that RFL has the mathematical structure of a learning algorithm. It does not claim convergence in finite time or under Phase I conditions.

### 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 Phase I Experimental Results

This section presents findings from CAL-EXP-3, a synthetic alignment-success simulation designed to validate governance and measurement infrastructure—not Lean verification.

### 4.1 Phase I Baseline Observations

CAL-EXP-3 compares treatment ( $\text{lr}=0.1$ ) against baseline ( $\text{lr}=0.0$ ) using a synthetic  $\Delta p$  proxy. Across three canonical runs (seeds 42, 43, 44), the treatment exhibited  $\Delta\Delta p$  of  $+0.032$  to  $+0.042$ , achieving claim level L4 (single-environment replication).

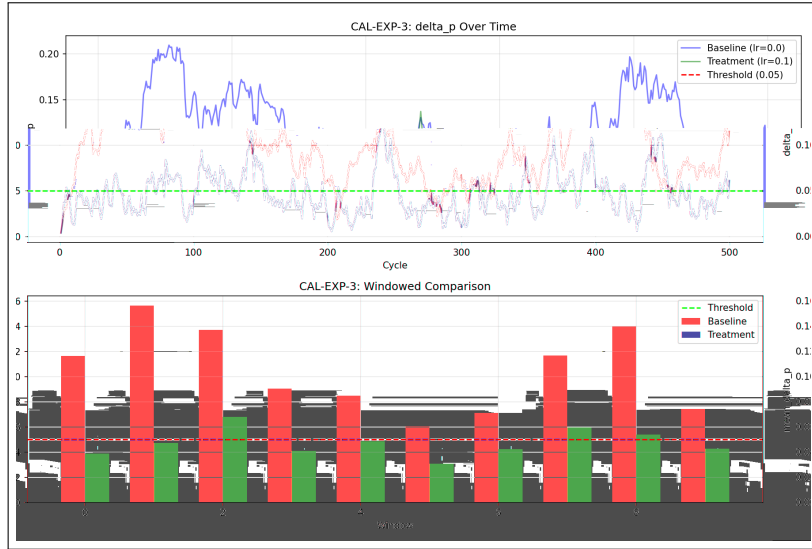


Figure 2: CAL-EXP-3:  $\Delta p$  dynamics over cycles (top) and windowed mean comparison (bottom). Baseline (blue, dashed) vs Treatment (green, solid). Both arms oscillate; treatment shows tighter variance. This plot illustrates logging provenance from an exploratory run; canonical  $\Delta\Delta p$  values ( $+0.032$  to  $+0.042$ ) are from the three indexed runs (seeds 42–44). Data: `docs/evidence/cal_exp_3/`.

No convergence was observed in Phase I. This is expected: Phase I validates infrastructure, not capability.

### 4.2 Interpreting Phase I Outcomes

**Important:** In CAL-EXP-3, the verifier outcome  $\mathcal{V}(e_t) \in \{1, 0, \perp\}$  is instantiated as a synthetic success proxy  $\Delta p$ , not Lean proof verification.

The Phase I results establish three facts:

1. **The measurement substrate works.**  $\Delta p$  (success rate proxy) is computable per cycle. Variance between arms is measurable. CAL-EXP-3 measured  $\Delta\Delta p$  of +0.032 to +0.042 across three canonical runs (seeds 42, 43, 44), achieving claim level L4.
2. **Fail-closed governance triggers correctly.** CAL-EXP-4/5 stress tests triggered F5.2 (variance ratio) and F5.3 (windowed drift), capping those runs at L0. CAL-EXP-3 used validity checks F1.x–F4.x.
3. **Non-convergence is informative, not a failure.** Phase I was designed to validate infrastructure, not demonstrate capability.

## 5 Discussion: Why This Matters

Phase I experiments aimed to characterize the behavior of the MathLedger substrate under various configurations. The  $\Delta p$  dynamics (Figure 2) were observed across cycles to understand arm behavior.

### 5.1 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
<b>MathLedger (RFL)</b>	<b>Verified truth<sup>†</sup></b>	<b>High</b>	<b>Yes</b>

Table 1: Comparison of approaches to reliable AI. MathLedger combines verified learning signals with fail-closed governance. <sup>†</sup>Phase I uses a synthetic proxy; Lean verification is Phase II+.

### 5.2 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.

## 6 Explicit Non-Claims and Scope Boundaries

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

### 6.1 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. CAL-EXP-3 measured  $\Delta\Delta p$  but no convergence trajectory was observed.
- **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.

### 6.2 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.

### 6.3 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.

## 7 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.

## 7.1 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.

## 8 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  $\Delta p$  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 verified truth*. 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

CAL-EXP-3 canonical run artifacts are stored locally. Upon posting, we will provide a public repository link and a frozen artifact hash list as ancillary material. The bundle includes hashed figure outputs and a reproduction script (`scripts/run_cal_exp_3_canonical.py`) that regenerates them from the pinned evaluator path.

Run reproduction commands are documented in `CAL_EXP_3_INDEX.md` (Evaluator Path section).

Artifact	Location
Figures	docs/evidence/cal_exp_3/figures/
Source Traceability	docs/evidence/cal_exp_3/SOURCES.md
Canonical Index	docs/system.law/calibration/CAL_EXP_3_INDEX.md

Table 2: CAL-EXP-3 evidence artifacts and their locations.

## 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 Monroe. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.