MAZE.AI: A Claim-Bounded Answer Loop with Machine-Checked Provenance

An AI that can be trusted is not one whose every behaviour is proven — that is impossible for open language — but one whose every claim is proven. This paper describes a running system built on that criterion, the theorems that bound it, and the tests that tie the theorems to the code in production.

J. Konstapel, Leiden — with computational assistance from Claude (Anthropic) · 19 September 2026, revised 21 September 2026 · live: maze.swarp.nl/ai · code: maze/ai, maze/rai, maze/lean (Lean 4.34, no Mathlib)

What this paper is, and is not. This is a working report on a provenance layer for a knowledge net: an answer loop that can only say what stands at an address, with that guarantee machine-checked. It is not a general AI and not a chatbot. Its placement, measured on two procedure-generated blind sets of one hundred questions, is 73 and 78 of 100 (named things 94–96%, described things 50–62%, median 3.3 s); its first hand-picked test of 20 scored 15 and is superseded. Its use today is where invention is unacceptable and the question is "what is held?" — patent claims traceable to their source, a ledger of settlements, memoranda that must cite. We report the construction, the theorems, the bindings to running code, the measurements, and the defects, in that order.

Abstract. Large language models produce fluent assertions with no bound on what they assert. We argue that the only certifiable relation between an answer and the world is

provenance: that a piece of content was retrieved from a specific, certified store at a specific, computed address, along a derivation that is written down. MAZE.AI is an answer loop whose output type admits exactly four forms — derived, attested, vacancy, proposal — each carrying its address, and whose emission path is single. The three invariants of the loop (derived ⇒ the pair is in the store; vacancy ⇒ the address is empty; attested ⇒ a ledger entry settled this address for this canonical question, by a named person, on a date, and the content is in the store) are proven in Lean 4 against the running definitions, in both a list-store and a finite-map-store model where the first invariant becomes an equivalence. The address kernel (balanced-ternary bijection, range, uniqueness, prefix-stable truncation) is proven and differentially tested against the production JavaScript kernel (30,572 addresses, 0 differences). A separate verifier module re-reads store and ledger before any emission and refuses rather than rewrites. The ledger is append-only, hash-chained and signed; the canonicalisation of questions is a versioned module with a determinism test. Placement was calibrated in pre-registered steps from 18 to 73 of 100 on a procedure-generated blind set and confirmed at 78 on a second fresh set; every step is hashed before its run and booked, including the failed ones. Settlements are signed with the settler's own office key (roles per key), and an office interface for the notarial profession runs on the same loop. A reverse "who mentions whom" index (8.0 million pairs) was built and falsified as a placement mechanism. Of seven geometric translations of "resonance" from the vacuum.net theory, six measured at or below chance; the seventh — a quaternionic ring measure over sister cells of the address mesh — reached AUC 0.70 and stands in the gate as a tiebreaker. A simulated Kuramoto field with a phase per piece and a ledger-driven learning rule runs beside the loop as an advisory instrument. The language model is retained only as a proposer: it may phrase, it may not assert. We know of no other system that combines these elements in one output path with machine-checked invariants; we claim novelty only for the combination. The limits are stated explicitly: placement quality (73–78%), the thinness of the net's neighbourhoods (addresses from titles alone), the semantic gap between "found on an image of the address" and "found at the address", and an empty ledger until real users settle. The system is a proven building block with a measured calibration, not a finished AI.

1. The problem: assertion without bound

A language model is a function from text to text. Nothing in that function distinguishes a sentence that reports something from a sentence that merely continues something. Retrieval-augmented generation reduces the frequency of invention but does not change the type of the output: the model still emits an assertion, now decorated with citations that it also chose. The person reading it cannot tell, from the artefact alone, which parts are held by the sources and which are the model's continuation.

The obvious repair — "prove the model correct" — is blocked in principle. Correctness of an assertion about the world requires a truth relation between sentences and states of affairs; for an open language that relation is not a total, certifiable object (Tarski's undefinability theorem gives the sharp form). So "every behaviour proven" is not a research programme with a long horizon; it is not a programme at all.

2. The criterion: every claim proven

What can be certified is narrower and, we argue, sufficient: the relation between an output and a store. Define a certified store as a set of (address, content) pairs whose membership is decidable. Then an answer system satisfies criterion P if every output it can emit is of a type whose meaning is a decidable statement about the store, and each such statement is proven for the emitting function. Under P the system never says "X is true"; it says "X is what stands at address a, reached by derivation d" — and that sentence is either provable or the output cannot be constructed.

P moves the burden of meaning to where it can be borne: to the person who settles that a given address answers a given question, and to a ledger that records that settlement as an event. The machine proves provenance; the human settles meaning; neither pretends to do the other's work.

3. The construction

MAZE.AI runs as a context under the MAZE, a self-addressing ternary knowledge net of 8.19 million pieces (Wikipedia, Mathlib theorems, EPO patents, art, music, open questions) in which every piece has an address computed from its content. Five components, each small:

1Kernel (C1). Addresses are lists of trits (–1, 0, +1) in balanced ternary. The kernel is a bijection between addresses of depth n and integers in $[-(3^n-1)/2, (3^n-1)/2]$; truncation to a prefix is idempotent and stable under deepening. Proven in Lean (Trit.lean, Kern1.lean, Brug.lean).

2Store (C2). The production store is PostgreSQL (address, content, receipt of origin). Two Lean models: a list of pairs, and a finite map — a list with a proof that each address carries at most one content (Winkel.lean), matching the MAZE invariant that an address is never rewritten after admission.

3Loop (C3). One function, one output type with four constructors, each carrying an address: inductive Uitvoer

4 | afgeleid (addr : Int) (content : String) -- derived
5 | voorstel (addr : Int) (bron : String) -- proposal
6 | vacature (addr : Int) -- vacancy
7 | geattesteerd (addr : Int) (content : String) (who : String) (date : String)

The loop reads the ledger for the canonical question first; a settlement whose content is still in the store yields attested. Otherwise it reads the store at the computed address: content yields derived, nothing yields vacancy. A proposal is what a language model may supply — a phrasing with the addresses it cites; it enters the loop and leaves it only as derived or vacancy.

8Verifier (C4). A separate module through which every output passes before emission. It does not trust the loop: it re-reads the store and the ledger and checks the invariants on the output as it stands. If it refuses, nothing is emitted (an error with reason) — never a silent retyping.

9Ledger (C5). Append-only rows (canon version, canonical question, material id, address, who, via, date, previous hash, hash, signature, key fingerprint); hash = sha256 over previous hash and fields; signature = HMAC over the hash with the key that admitted the settlement. A chain check walks the whole list. Two settlements of the same question both remain; the latest is operative.

Two input layers feed the loop: exact names (a question that is the name of an inhabitant resolves by identity), and open language (embedding of the question through the same gate as material → 24-trit address → deepest inhabited prefix ≥ 6 → cosine ranking within that neighbourhood; a name bridge with a gate test rejects name hits that lie far from the question's meaning). The canonicalisation canon(q) — NFKC, lower case, letters/digits/spaces only, single spaces, trimmed, 400 characters — is its own versioned module with a determinism test (frozen examples, idempotence, 5,000 random inputs).

One output path. The MAZE's main conversational page also goes through the loop: the language model's answer is delivered as a proposal, the loop's output is shown above it, and both pass the verifier. There is no debug path, and no second path "for now".

4. The theorems

All statements below are checked by Lean 4.34 with no additional axioms beyond propext, Classical.choice and Quot.sound, and no sorry. Names are as in the source.

KERNEL

waarde_bereik: |value(a)| ≤ (3^|a|−1)/2. waarde_inj: equal depth and equal value ⇒ equal address. waarde_ofInt: value(ofInt n v) = v on the range. trunc_idem, trunc_prefix: truncation is idempotent and stable under deepening. waarde_eq_decode_reverse: the two kernel formulations (deepest-first, shallowest-first) compute the same integer.

Proof sketch: induction on the address list with Int.pow_succ; the range bound by generalising 3^n and case analysis on the head trit (omega); injectivity by contradiction from the bound on tails.

LOOP, LIST STORE (Lus.lean)

lus_afgeleid_store: lus s g q addr = afgeleid a c ⇒ (a, c) ∈ s.

lus_vacature_leeg: lus s g q addr = vacature a ⇒ a = addr ∧ ∀ c, (addr, c) ∉ s.

lus_geattesteerd_grootboek: lus s g q addr = geattesteerd a c who date ⇒ (∃ e ∈ g, e.canonQ = q ∧ e.addr = a ∧ e.who = who ∧ e.date = date) ∧ (a, c) ∈ s.

Proof sketch: unfold the loop; case analysis on List.find? over ledger and store; membership from List.mem_of_find?_eq_some, emptiness from List.find?_eq_none.

LOOP, FINITE-MAP STORE (Winkel.lean)

Winkel.vind_iff: lookup s a = some c ⇔ (a, c) ∈ s.

antwoordW_iff: answer s addr = derived addr c ⇔ (addr, c) ∈ s — the first invariant becomes an equivalence: the loop emits exactly what the store holds.

antwoordW_vacature_iff: vacancy ⇔ the address is empty.

Winkel.voeg_bewoond: inserting on an inhabited address changes nothing (write-once as a theorem); vind_voeg_leeg, vind_voeg_ander: insertion on an empty address is visible there and nowhere else.

The three loop invariants are re-proven over the map (lusW_*).

Proof sketch: the uniqueness field of the structure closes the converse direction; the write-once property follows from a dependent if on the lookup.

LEDGER (Keten.lean)

chain_deterministic, totality, chain_snoc (prefix property of the chain over a list), and chain_last_changes: altering the last entry changes the final hash — proven only under the explicit hypothesis that the step function is injective. Without collision-freeness it is false, and Lean does not let it be stated otherwise.

What is not proven, and not provable: that any derived content is a good answer to the question. That is the semantic boundary of §1, and the system does not cross it. It is measured instead (§6).

5. Binding the theorems to the running system

A proof about a model is worthless if the production code is a different function. Three bindings:

- •Differential kernel test. The Lean executable prints 30,572 addresses (−100,000…100,000 at depth 12; 2,000 pseudo-random integers up to 3^{42} at depth 42) in both formulations; the production JavaScript kernel prints the same series; `diff` reports 0 differences.
- •Production-path test. The real loop runs on the real store for a fixed question list; every output is passed through the verifier (C4), which re-reads store and ledger; the ledger chain is walked. Any refusal is red.
- •One control command. `node maze/ai/controle.mjs` rebuilds the Lean project, prints the axiom report and rejects anything beyond the three standard axioms, runs the differential test, the canon determinism test and — with a database — the production-path test. It stops at the first red step.

The residual gap is stated rather than hidden: production "derived" uses the deepest inhabited prefix of the computed address as an image and ranks within it by cosine; the Lean loop models exact lookup. The verifier therefore checks store membership and address provenance (the inhabitant's address, or the image as its prefix), and a `zwak` (weak) flag marks a top inhabitant with cosine < 0.32 to the question. The equivalence of §4 holds for exact addresses; on images it holds as implication.

6. Measurements

Placement, calibrated in the open. Two blind sets of one hundred questions were generated by a fixed procedure (a random sample of inhabitants with a description; a language model writes one natural question per inhabitant, half naming it, half describing it; Dutch and English alternating), hashed before the first run, and never used for tuning. Baseline 18/100. Five pre-registered steps, each with a prediction and a revert rule: word sequences of the question matched exactly against names (41); choice by cosine instead of length (44); parenthetical title variants and single capitalised words (50); the language model as title proposer, the net deciding on membership and cosine (69); a proposal that passes the gate takes precedence over the anchors named in the question (73). One step (three proposed titles) was reverted at 71. Confirmation on a second fresh set: 76, later 78. Named things 94–96%, described things 50–62%, median 3.3 s. A closure rule for proposals (the name, put back through the loop, must return to the same piece) and a weak-to-vacancy rule were added without loss. The first hand-picked twenty-question test (15/20) is superseded.

The reverse direction, built and falsified. "Who mentions whom" was extracted from 3.0 million descriptions into 8.0 million pairs. As a placement mechanism it gave nothing: with the proposer 73 → 71, without it 50 → 28 (in front) or 50 (behind). The net can decide, it cannot propose — a negative measurement in favour of the proposer/gate design.

Resonance, seven translations. The vacuum.net theory names resonance, not similarity, as the criterion. Each translation was pre-registered and scored by AUC (correct vs wrong answers, 196–197 derived answers over the 200 questions) against the cosine (0.87–0.89): trit sum of the address 0.41; closure of the pair 0.53; scalar path through the neighbours 0.55–0.60; deviation-axis variants 0.45 and 0.38; the quaternionic ring measure R_Q computed in the 1,536-dimensional carrier, centred, and in the 24 gate axes: identically 1.000 (flat mesh); R_Q with region frames (centre, principal direction with memory-fixed sign) over the sister cells of the address at depth 6, Markley-averaged: 0.704 [0.61, 0.79] — the pre-registered threshold. It now stands in the gate as a tiebreaker for candidates within 0.05 in cosine (73 / 78, no case lost). The combination with the cosine (0.84) remains below the cosine alone; the field coherence $r \cdot \cos(\theta_a - \theta_q)$ scores 0.52 as an answer measure and is kept for the field, not the gate.

The field. One oscillator per inhabited depth-6 region with an intrinsic region frame; the phase of every piece, including a fresh proposal, $\phi = \mathrm{atan2}(x \cdot e_2, x \cdot e_1)$; questions, intakes and settlements pull the region phase; an hourly Kuramoto step; couplings between regions initialised from the cosine of the centres, strengthened at each ledger settlement by $\eta = 0.1$ toward the measured resonance and relaxed daily by $\lambda = 0.01$. Two tests are fixed in advance: a shuffle test of the order parameter after 14 daily logs, and a 90-day test of whether learned couplings rank settlement resonance better than the frozen baseline. The field advises; the gate decides.

A separate, failed engineering track, reported for completeness. Before the claim-bounded loop, a byte-level language model with a computed address per byte, prefixrouted experts and native ternary weights was trained (NumPy, CPU) under two hashed plans. Against a conventional small model at 1.39 bits/byte, the computed-address model reached 5.54 (v0) and 2.96 (v2). Two of three pre-registered proofs passed to the letter but not in spirit. That track is not MAZE.AI; it is booked in maze/ai/uitkomst.md. The archive-level union, measured on the net itself, is clean: of 4.08 million distinct addresses, 0.06% are shared by two independently built sources.

7. What is claimed, and what is not

- Claimed: every output of the deployed loop is of one of four types, carries an address, and satisfies a proven statement about the store; a proposal never leaves the loop as an assertion; settlement is an auditable event; the proofs are bound to the code by differential and production-path tests under one command.
- Not claimed: that answers are true; that placement is reliable beyond the measured 73–78%; that the ring measure earns more than a tiebreak (open, needs data); that the ledger is unforgeable against a key holder (signatures are HMAC with the settler's office key or the server key; who is a name next to a key fingerprint); that sha256 is collision-free (assumed, stated).
- Novelty, carefully: proof assistants, retrieval with citations, transparency logs and human-in-the-loop review each exist. We know of no system in which they are combined into a single output path whose output type is claim-

complete by construction and whose invariants are machine-checked against the running definitions and tested against production. We claim the combination, not the parts.

8. Relation to existing work

Retrieval-augmented generation (Lewis et al. 2020) and citation-conditioned answering (e.g. Nakano et al. 2021) attach sources to generated text but keep the generation as the answer. Verified software (seL4, Klein et al. 2009; CompCert, Leroy 2009) proves programs against specifications and is the methodological model for §4–5, with one difference of scope: we prove a small loop and a kernel, not a compiler, and we bind them to an unverified production language by differential testing rather than by extraction. Transparency logs (Laurie 2014, Certificate Transparency) are the model for the ledger: append-only, hash-chained, publicly checkable, with the meaning of an entry left to the parties. The MAZE's address-from-content principle descends from content-addressable storage, but the balanced-ternary kernel gives addresses a metric structure (prefix = coarser class) that a hash lacks, and the truncation theorems are what make "an image of the address" a well-defined notion. On the impossibility side, Tarski (1936) gives the boundary that motivates P.

9. Reproduction

```
git clone … && cd swarp
node maze/ai/controle.mjs          # Lean rebuild + axiom report + differential test + canon test
```

```
DATABASE_URL=… node maze/ai/controle.mjs   # + production-path test on the live store
node maze/ai/honderd-toets.mjs "label" [-b]   # hundred-question sets A/B (hash must match); twintig-toets.mjs for the old twenty
node maze/ai/ringmaat-d-toets.mjs    # ring measure R_Q, region frames, Markley, bootstrap (plan-ringmaat-d.md)
curl -X POST https://maze.swarp.nl/api/ai/vraag -H 'content-type: application/json' -d '{"vraag":"balanced ternary"}'
```

Lean sources: maze/lean/MazeAI/{Trit,Kern1,Brug,Keten,Lus,Winkel}.lean (≈740 lines, no Mathlib). Production loop: maze/ai/{canon,lus,verifier,route,kantoor}.mjs; field: maze/rai/{veld,ringmaat}.mjs. Ledger table ai_attest; every plan, hash and outcome under maze/ai/plan-*.md, uitkomst-*.md.

Annotated references

- Tarski, A. (1936). Der Wahrheitsbegriff in den formalisierten Sprachen. Studia Philosophica 1. Why read: the theorem that a sufficiently rich language cannot define its own truth predicate — the formal reason why "prove the model correct about the world" is not a programme.
- Klein, G. et al. (2009). seL4: formal verification of an OS kernel. SOSP. Why read: the reference case for proving a running system against its specification, and for being explicit about what the proof does not cover.
- Leroy, X. (2009). Formal verification of a realistic compiler. CACM 52(7). Why read: the model for tying a proof to executable code; we substitute differential testing for extraction and say so.
- Laurie, B. (2014). Certificate Transparency. CACM 57(10). Why read: append-only, hash-chained logs whose entries are

facts about events, not judgements — the design our ledger follows.

•Lewis, P. et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. NeurIPS. Why read: the standard way to attach sources to a language model, and the point where MAZE.AI departs — the retrieved content, not the generation, is the output.

•Nakano, R. et al. (2021). WebGPT: browser-assisted question-answering with human feedback. arXiv:2112.09332. Why read: citations chosen by the model, which is exactly what leaves the assertion type unchanged.

•de Moura, L., Ullrich, S. (2021). The Lean 4 theorem prover and programming language. CADE. Why read: the tool in which the theorems of §4 are checked; statement and program are the same object.

•Knuth, D. E. (1997). The Art of Computer Programming, vol. 2, §4.1. Why read: balanced ternary, its range and uniqueness — the arithmetic of the address kernel.

•Konstapel, J. (2026). Clerk Maxwell and the Vacuum.Net Theory, 6th ed. Why read: the composition rule (ring measure), the phase of a piece and the ledger-driven learning rule measured and built in §6 above; the frame correction that made the measure bend.

•Konstapel, J. (2026). Why Twenty-Four Axes: The Mesh Is Not the Carrier. Why read: why the composition rule returns 1.000 in the embedding carrier and needs the mesh's own axes — the diagnosis behind the 0.70.

•Konstapel, J. (2026). Provable AI. and maze.ai/1 — Specification. Why read: the criterion P, the boundary argument and the work programme (a–j) this paper reports on; every item is now either built or listed in §7 as open.