Achieving Artificial Intelligence (youvan.ai1) on a PC with Category Theory & HoTT

Douglas C. Youvan

doug@youvan.com

youvan.ai

September 8, 2025

We present ai1, a compact, local workbench for deterministic equational AI grounded in Category Theory (CT) and Homotopy Type Theory (HoTT). Instead of gradients or probabilistic search, ai1 operates by typed term rewriting to normal form (NF) with machine-checkable witnesses. The core includes: (i) identity and composition laws under strong typing; (ii) verification of pushout β-laws via horns (u∘inl = …, u∘inr = …); (iii) mid-identity elimination (e.g., g∘id_B∘f → g∘f); (iv) η-probes for harmless expansions; and (v) a lightweight typed conjecture search. All runs complete on a single PC and emit human-readable JSON traces, enabling reproducibility, auditing, and pedagogy. We position ai1 as a complementary direction for new AI: systems that build compositional explanations and witnesses rather than fit parameters. Two bridges make the abstractions legible: Appendix A is a CT-HoTT fable that maps scenes to laws; Appendix B gives a Lego-style kit that turns objects, morphisms, and 2-cells into physical assemblies. The result is a tractable, inspectable pipeline for equational reasoning that is small enough to understand, yet rich enough to explore β/η laws, normal-forms, and typed conjectures.

Keywords: ai1, category theory, homotopy type theory, HoTT, equational reasoning, term rewriting, normal forms, horns, pushout β-laws, identity elimination, mid-identity elimination, η-probes, typed conjecture search, deterministic rewriting, local-first AI, single-PC reproducibility, interpretability, diagrammatic reasoning, free categories, Yoneda lemma, univalence, higher category theory, compositionality, symbolic AI, proof automation, Windows PowerShell. 50 pages. A collaboration with GPT-5.

1. Introduction & Positioning

ai1 is a deliberately small, deterministic workbench for equational AI: it reasons by rewriting typed expressions to normal form (NF) and only accepts an equation when *both sides normalize to the same NF at the same type*. Where mainstream AI optimizes a loss, ai1 constructs witnesses—machine-checkable traces that a proposed law really holds in a given typed catalog. The system runs entirely on a single PC, emits human-readable JSON, and keeps the moving parts few enough that you can read the whole stack.

Why equational?

Equations are the currency of compositional understanding. In Category Theory (CT) and Homotopy Type Theory (HoTT), structure is expressed by how things compose and by the laws they satisfy: identities, associativity, β/η equalities, universal properties (e.g., coproduct/pushout eliminators), and eventually higher coherences. An equational AI does not *guess* these; it checks them, and it does so under a strict type discipline so that ill-typed composites never arise.

Minimal core, strong typing

ai1 ships with a tiny signature that is still rich enough to be instructive:

- Objects: A,B,C,D,PA,B,C,D,PA,B,C,D,P
- Morphisms: f:A ⟶ Bf:A\!\to\!Bf:A→B, g:B ⟶ Ag:B\!\to\!Ag:B→A, inl:B ⟶ Pinl:B\!\to\!Pinl:B→P, inr:C ⟶ Pinr:C\!\to\!Pinr:C→P, u:P ⟶ Du:P\!\to\!Du:P→D
- Identities: idX:X ⟶ Xid_X:X\!\to\!XidX:X→X for each XXX

Typing is enforced at parse time. Composition h∘kh∘kh∘k is only formed when codomain(k)(k)(k)=domain(h)(h)(h). This is not just hygiene; it is the backbone of reliable automation.

What ai1 does (and doesn't)

- Does: normalize any well-typed term; check horns (proposed equalities) by NF comparison; run β-law packs (e.g., u∘inlu∘inlu∘inl, u∘inru∘inru∘inr and

their identity variants); perform mid-identity elimination (e.g., g∘idB∘f→g∘fg∘id_B∘f \to g∘fg∘idB∘f→g∘f); launch η-probes (neutral expansions that should normalize away); run a typed conjecture search to mine candidate equalities.

- Doesn't (yet): provide higher-dimensional proofs, tactic programming, or a full proof-assistant experience. ai1 is not Lean/Coq/Agda; it is a focused equational workbench whose outputs are simple, auditable witnesses.

Determinism, traces, and witnesses

Every check produces a witness record: the original LHS/RHS strings, their types, their NFs, and a pass/fail verdict. Determinism matters: NF is unique for a given input, so "same NF + same type ⇒ accepted." In practice, you observed:

- g∘idB∘f⇒g∘f:A → Ag∘id_B∘f \Rightarrow g∘f: A\!\to\!Ag∘idB ∘f⇒g∘f:A→A
- u∘idP∘inl⇒u∘inl:B → Du∘id_P∘inl \Rightarrow u∘inl: B\!\to\!Du∘idP ∘inl⇒u∘inl:B→D, u∘idP∘inr⇒u∘inr:C → Du∘id_P∘inr \Rightarrow u∘inr: C\!\to\!Du∘idP∘inr⇒u∘inr:C→D
- End-identity variants (e.g., idA∘g∘fid_A∘g∘fidA∘g∘f, g∘f∘idAg∘f∘id_Ag∘f∘idA) also normalize as expected.

Horn packs for identities and pushout β-laws all passed, η-sweeps up to moderate depths found no nontrivial witnesses, and typed conjecture search mostly surfaced the identity-drop families—exactly what we should see in this minimal catalog. NF latency on your machine was ~60–120 ms for the test expressions, supporting interactive workflows.

Position in the AI landscape

We intend ai1 as a complement to both symbolic and statistical systems:

- To symbolic stacks (rewriting engines, SMT, proof assistants), ai1 contributes a tiny, transparent NF oracle and a horn-checking harness with strict types

and JSON witnesses—ideal for teaching, prototyping, or embedding as an equational “lint.”

- To statistical stacks (LLMs, differentiable programs), ai1 can serve as a post-validator of algebraic invariants or as a teacher: the model proposes patterns; ai1 certifies, rejects, or yields counterexamples. This is a path toward AI that carries invariants rather than only patterns.

Design constraints and values

1. Local & reproducible. One PC, no network requirement; runs are seedable and logged.
2. Small surface area. A tiny catalog invites reading the whole system; no hidden tactics.
3. Type safety first. Many failures never occur because ill-typed terms never form.
4. Human-readable artifacts. Witness JSON is designed to be read, diffed, and cited.
5. Bridges for intuition. Two appendices translate formal structure into words (a fable) and objects (Lego kit), so more readers can “feel” the laws.

Intended users & workflows

- Researchers experimenting with β/η behavior, identity laws, and simple universal properties under strict typing.
- Engineers who need a lightweight equational checker to guard invariants in pipelines.
- Educators introducing CT/HoTT ideas with live, inspectable traces and tangible metaphors.

A typical loop: (i) draft a horn pack (e.g., β-laws, mid-identity variants); (ii) run the horn checker; (iii) inspect witnesses; (iv) run η-probes and typed conjecture search for sanity and discovery; (v) keep the passing horns as part of your catalog of laws.

Contributions (this version)

- A deterministic NF oracle for a typed CT/HoTT-flavored catalog, with ~100 ms interactive performance.
- A horn-checking pipeline that validated identity laws, pushout β-laws, and mid-identity elimination with explicit witnesses.
- An η-probe harness that found no nontrivial expansions in current bounds (evidence of stability).
- A typed conjecture search that mines equalities without inventing structure in this signature.
- Two explanatory bridges—a story (words) and a Lego kit (objects)—aligning formal content with human intuition.

What ai1 is not—and why that's useful

ai1 is not trying to be "everything." By refusing generality (no huge libraries, no tactics), the system stays legible. You can trace why an equality holds, show it to a student, embed it in a CI job, or use it to discipline a larger system's outputs. In a field crowded with black boxes, ai1 is a glass box—and that is its role in the broader project of developing new AI that composes explanations and carries proofs.

2. Background (CT/HoTT essentials for ai1)

2.1 Categories, objects, morphisms

A category has objects and morphisms (arrows) that compose. ai1's teaching catalog is tiny but expressive:

- Objects: A,B,C,D,PA,B,C,D,PA,B,C,D,P
- Generators: f:A □→ □Bf:A\!\to\!Bf:A→B, g:B □→ □Ag:B\!\to\!Ag:B→A, inl:B □→ □Pinl:B\!\to\!Pinl:B→P, inr:C □→ □Pinr:C\!\to\!Pinr:C→P, u:P □→ □Du:P\!\to\!Du:P→D

- Identities: idX:X → Xid_X:X\!\to\!XidX:X→X for each object XXX

Composition is written h∘kh∘kh∘k ("do kkk then hhh"). Types must match: codomain(k)(k)(k)=domain(h)(h)(h).

2.2 Unit and associativity laws

Every object carries a unit: idX∘p=pid_X∘p=pidX∘p=p and q∘idX=qq∘id_X=qq∘idX =q whenever types fit. Composition is associative: (h∘k)∘ℓ=h∘(k∘ℓ)(h∘k)∘\ell=h∘(k∘\ell)(h∘k)∘ℓ=h∘(k∘ℓ). In ai1 these laws are compiled into normalization rather than asserted as separate horns, so expressions like g∘idB∘fg∘id_B∘fg∘idB∘f normalize to g∘fg∘fg∘f.

2.3 Coproduct intuition (pushouts, "sum types")

Think of PPP as a coproduct (sum) with injections inl:B → Pinl:B\!\to\!Pinl:B→P and inr:C → Pinr:C\!\to\!Pinr:C→P. Maps out of a coproduct correspond to case analysis: a morphism u:P → Du:P\!\to\!Du:P→D packages two branches u∘inl:B → Du∘inl:B\!\to\!Du∘inl:B→D and u∘inr:C → Du∘inr:C\!\to\!Du∘inr:C→D. In our minimal catalog we treat the β-behavior extensionally: the composites u∘inlu∘inlu∘inl and u∘inru∘inru∘inr are the observable branches, and they are stable under identities (e.g., u∘inl∘idB≡u∘inlu∘inl∘id_B \equiv u∘inlu∘inl∘idB≡u∘inl).

2.4 HoTT viewpoint (just enough)

In HoTT, equalities are paths; computation rules like β/η are judgmental equalities; pushouts are higher inductive types with constructors (here, inl,inrinl,inrinl,inr) and an eliminator (our uuu). ai1 operates at the 1-dimensional (equational) layer: it certifies that two terms are definitionally equal via common normal form at the same type. This is the "zero-thickness shadow" of the richer HoTT picture—but it preserves the discipline: types, constructors, eliminators, and computation rules.

2.5 Normal forms & rewriting discipline

ai1's checker reduces any well-typed term to a deterministic normal form (NF) using:

- unit elimination (ididid-drop),
- association/rebracketing,
- simple β-stability for u∘inl,u∘inru∘inl, u∘inru∘inl,u∘inr.

Determinism ensures: same input ⇒ same NF. An equation "LHS = RHS" is accepted when types coincide and NF(LHS)=NF(RHS). Each check yields a witness record (names, types, NFs, verdict).

2.6 Horns

A horn is a typed equality proposal (name, lhs: X → Y, rhs: X → Y)(\text{name},\; \text{lhs}:\!X\!\to\!Y,\; \text{rhs}:\!X\!\to\!Y)(name,lhs:X→Y,rhs:X→Y). ai1 validates horns by NF comparison and reports per-horn witnesses. We used packs for:

- Identity laws (left/right units for f,g,inl,inr,uf,g,inl,inr,uf,g,inl,inr,u),
- Pushout β-stability (u∘inlu∘inlu∘inl and u∘inru∘inru∘inr with end-identities),
- Mid-identity elimination (e.g., g∘idB∘f≡g∘fg∘id_B∘f \equiv g∘fg∘idB∘f≡g∘f).

All passed in the current catalog.

2.7 η-probes (neutral expansions)

An η-principle says a map is determined by its action on constructors. Practically, we probe neutral expansions (wrapping with identities, mild rebracketings) and check they normalize away: no nontrivial η-witnesses were found within our sweep bounds—evidence that the core behaves extensionally as expected.

2.8 Typed conjecture search (bounded)

ai1 explores the well-typed expression space by bounded random walks (length targets like “len 3” count compositional steps). Candidates (L,R)(L,R)(L,R) are kept when both sides type-check and share the same NF. On our signature, the miner predominantly rediscovers identity-drop families; no unexpected laws emerged at the explored depths—consistent with the minimal theory.

2.9 Notation & conventions

- h∘kh∘kh∘k means “first kkk, then hhh”.
- Types are written X → YX\!\to\!YX→Y.
- “NF” denotes the unique normal form of a term.
- We avoid heavy proof terms; witness JSON is the artifact to cite.

2.10 Why this background matters

These essentials—typed composition, units, coproduct behavior, β/η hygiene, and deterministic NF—are precisely what ai1 mechanizes. They are small enough to teach and audit, yet expressive enough to anchor equational AI workflows and to interface cleanly with more ambitious CT/HoTT developments later in the paper.

3. The ai1 Engine

3.1 Design goals

ai1 is a single-PC, deterministic, typed equational workbench. Its core promises are:

1. Type safety (every step respects domains/codomains),
2. Deterministic normalization (one canonical NF per well-typed term),
3. Small, auditable kernel (few primitives; laws are tested, not assumed),
4. Reproducible search (seeded, bounded exploration).

3.2 Term language & typing

Signature (current demo). Objects {A,B,C,D,P}\{A,B,C,D,P\}{A,B,C,D,P}; generators f:A□→□Bf:A\!\to\!Bf:A→B, g:B□→□Ag:B\!\to\!Ag:B→A, inl:B□→□Pinl:B\!\to\!Pinl:B→P, inr:C□→□Pinr:C\!\to\!Pinr:C→P, u:P□→□Du:P\!\to\!Du:P→D; identities idXid_XidX for each XXX.

Terms. Built from identities, generators, parentheses, and composition "∘∘∘" (right-to-left evaluation: do the right thing, then the left). A term is well-typed iff successive codomain/domain match; ai1 infers and checks the unique type X□→□YX\!\to\!YX→Y.

3.3 Normalization (NF)

The kernel reduces well-typed terms to a unique normal form using three disciplined schemas:

- Units: drop ididid on either side when types fit (e.g., g∘idB∘f⇝g∘fg∘id_B∘f \leadsto g∘fg∘idB∘f⇝g∘f; u∘idP⇝uu∘id_P \leadsto uu∘idP⇝u).
- Associativity: rebracket to a canonical spine (no parentheses remain in NF).
- Coproduct β-stability (extensional): treat u∘inl:B□→□Du∘inl:B\!\to\!Du∘inl:B→D and u∘inr:C□→□Du∘inr:C\!\to\!Du∘inr:C→D as primitive composites; adding identities around them does not change NF.

The result is a short word in generators (e.g., g∘fg∘fg∘f, u∘inlu∘inlu∘inl, u∘inru∘inru∘inr) or a single generator/identity. Determinism means equal inputs always yield equal NFs.

3.4 Equality checking via horns

A horn is a named, typed equation proposal (name, lhs, rhs)(\text{name},\; \text{lhs},\; \text{rhs})(name,lhs,rhs). Validation pipeline:

1. Parse & type-check each side to obtain X□→□YX\!\to\!YX→Y.
2. Normalize both sides.

3. Decide: pass iff types match and NFs are *syntactically identical*.

ai1 emits a compact, machine-readable witness per horn. Example:

{
"name": "mid_id_gf",
"ok": true,
"lhs_nf": "g∘f",
"rhs_nf": "g∘f",
"lhs_type": "A->A",
"rhs_type": "A->A"
}

In our runs, identity laws, β-stability around inl/inrinl/inrinl/inr, and mid-identity elimination all pass.

3.5 Structured probes (β, η, id-drop)

To keep the kernel small yet testable, ai1 packages targeted probe scripts:

- β-pack: confirms that identities at either end of u∘inlu∘inlu∘inl / u∘inru∘inru∘inr normalize away.
- η-probe: stresses neutral expansions like (idD)m∘u∘(idP)n(id_D)^m∘u∘(id_P)^n(idD)m∘u∘(idP)n and (idP)n∘inl(id_P)^n∘inl(idP)n∘inl, checking that no new equalities appear beyond NF equality of the base forms. Within moderate bounds we found no nontrivial η-witnesses, which is the expected extensional behavior.
- id-drop miner: fabricates expressions with strategic identities in the middle and verifies they collapse (e.g., g∘idB∘f≡g∘fg∘id_B∘f \equiv g∘fg∘idB∘f≡g∘f).

These suites double as regression tests when the signature evolves.

3.6 Typed conjecture search

The conjecture miner explores well-typed paths of bounded length (“len kkk” counts compositional steps). It generates candidate pairs (L,R)(L,R)(L,R) by:

1. walking type-compatible generators (optionally injecting identities with probability ppp),
2. discarding ill-typed or trivially identical pairs,
3. retaining NF-equal pairs as proposed equations.

On the current minimal signature, mined equalities were overwhelmingly identity-drop families, matching β/η/units. That is a healthy baseline: the kernel discovers exactly what it should, and not more.

3.7 Determinism, performance, and UX

- Determinism: normalization is a pure function; results are independent of process state or scheduling.
- Performance: in the measured Windows shell setup, a single nf call returns in ~60–120 ms (dominated by process spawn and JSON I/O). In-process or batched modes amortize this to sub-millisecond kernel time per reduction; the bottleneck is orchestration, not rewriting cost.
- Reproducibility: every exploration reports its random seed, walk limits, and counts; JSON outputs are stable and diff-friendly.

3.8 Failure modes & diagnostics

- Typing errors (“unknown morphism”, domain mismatch) fail fast with a typed trace.
- Ambiguous tokenization (e.g., unbalanced parentheses) is rejected before NF.
- BOM/encoding is handled consistently (utf-8-sig on input); outputs are strict UTF-8.

All failures produce structured ok:false records with stderr/stdout echoes to aid debugging.

3.9 Extending the signature

Adding power should be incremental:

- Introduce new generators (e.g., a pairing ⟨−,−⟩\langle -,- \rangle⟨−,−⟩ or a universal arrow),
- Add a small probe pack (β/η/units for the new constructors),
- Re-run horns + miners; only when the probe wall is green do we accept the extension.

3.10 Why this engine matters

ai1 shows that useful equational AI can be:

- Local: runs on one laptop, no cloud,
- Deterministic: proofs are just NF coincidences at the same type,
- Auditable: the entire story fits in a few transparent schemas,
- Composable: probe packs and miners boot-strap the next layer (Sections 4–6).

In short: ai1 is a small but faithful substrate on which stronger categorical/HoTT features—and eventually, new AI behaviors—can be safely layered.

4. Results & Traces (single-PC runs)

4.1 Setup (single machine)

- OS: Windows, PowerShell shell, Python 3.13 in a venv
- Hardware class: thin-and-light laptop (integrated graphics)
- Kernel: ai1ctl (deterministic NF), JSON I/O for probes/miners

All runs below were executed locally with fixed random seeds and no network access.

4.2 Horn packs (typed equalities)

A) Basic identity laws (10 horns).
All pass: g∘idB≡gg∘id_B \equiv gg∘idB≡g, idA∘g≡gid_A∘g \equiv gidA∘g≡g; analogues for fff, and for coproduct structure inl,inrinl,inrinl,inr, and eliminator uuu (both left/right identity forms).
Trace shape (one example):

{

"name":"id_right_g",

"ok": true,

"lhs_nf":"g","rhs_nf":"g",

"lhs_type":"B->A","rhs_type":"B->A"

}

B) β-stability around coproduct injections (4 horns).
All pass: identities inserted before/after u∘inlu∘inlu∘inl or u∘inru∘inru∘inr do not change NF.
(e.g., u∘inl∘idB≡u∘inlu∘inl∘id_B \equiv u∘inlu∘inl∘idB≡u∘inl, idD∘u∘inr≡u∘inrid_D∘u∘inr \equiv u∘inridD∘u∘inr≡u∘inr).

C) Mid-identity elimination (3 horns).
All pass: g∘idB∘f≡g∘fg∘id_B∘f \equiv g∘fg∘idB∘f≡g∘f; u∘idP∘inl≡u∘inlu∘id_P∘inl \equiv u∘inlu∘idP∘inl≡u∘inl; u∘idP∘inr≡u∘inru∘id_P∘inr \equiv u∘inru∘idP ∘inr≡u∘inr.

4.3 Normalization sanity (typed NF)

Representative calls confirm expected types and canonical forms:

- g∘idB∘f:A→Ag∘id_B∘f : A\to Ag∘idB∘f:A→A normalizes to g∘fg∘fg∘f
- u∘idP:P→Du∘id_P : P\to Du∘idP:P→D normalizes to uuu
- inl∘idB:B→Pinl∘id_B : B\to Pinl∘idB:B→P normalizes to inlinlinl
- u∘inl:B→Du∘inl : B\to Du∘inl:B→D and u∘inr:C→Du∘inr : C\to Du∘inr:C→D remain as-is (they are already NF)

Timing (PowerShell Measure-Command, process-per-call): ~60–120 ms per nf on this machine. In-process/batched runs remove spawn overhead and are effectively instantaneous at the scale of these terms.

4.4 Identity-drop miner (data-driven witnesses)

Targeted search that inserts identities at internal or end positions produced a minimal witness set (6):

- g∘idB∘f≡g∘fg∘id_B∘f \equiv g∘fg∘idB∘f≡g∘f
- idA∘g∘f≡g∘fid_A∘g∘f \equiv g∘fidA∘g∘f≡g∘f
- g∘f∘idA≡g∘fg∘f∘id_A \equiv g∘fg∘f∘idA≡g∘f
- u∘idP∘inl≡u∘inlu∘id_P∘inl \equiv u∘inlu∘idP∘inl≡u∘inl
- u∘idP∘inr≡u∘inru∘id_P∘inr \equiv u∘inru∘idP∘inr≡u∘inr
- Symmetric end-id forms for inl,inr,uinl,inr,uinl,inr,u

This matches the horn packs, providing independent, mined confirmations.

4.5 η-probe (no spurious expansions)

We stress-tested neutral expansions by repeating identities on the left/right of uuu, inlinlinl, inrinrinr:

- 49 configurations with up to 6 repeats each side → 0 witnesses (no new equalities beyond NF).
- 169 configurations with up to 12 repeats → 0 witnesses.

Interpretation: the kernel's extensional behavior is crisp—η-like "padding" with units never fabricates equalities.

4.6 Typed conjecture search (bounded len)

Random, type-respecting walks (len 3–5; multiple seeds; optional id-injection) yielded:

- Many candidate pairs that reduce to identity-only families (e.g., idXid_XidX-chains).
- After trivial-ID filtering, no nontrivial equalities remained on the current minimal signature.

This is the baseline we want: the engine rediscovers units/β-stability and otherwise stays conservative.

4.7 Determinism & reproducibility

- All horns/probes report the same NFs and types across repeated runs and seeds.
- JSON outputs include seeds, limits, and counts; files are diff-friendly and stable.
- Minor pitfalls we encountered (and fixed):

- JSON comments/BOM: remove // … and write UTF-8/UTF-8-sig consistently.
- Tokenizer: use the actual composition symbol "∘∘∘" or unambiguous ASCII with parentheses; avoid ambiguous token runs.
- Pathing: ensure ai1ctl is discoverable (venv Scripts on PATH or explicit wrapper).

4.8 What these traces establish

1. Sound unit laws across the signature (objects, injections, eliminator).
2. β-stability for coproduct eliminator uuu around inl/inrinl/inrinl/inr.
3. No accidental η-laws: only identity removal is observed.
4. Conservative conjecture space: no spurious "discoveries."
5. Single-PC feasibility with deterministic, auditable outputs suitable for versioned releases.

These results certify ai1 as a trustworthy foundation for the stronger algebraic features we layer next (Sections 5–6), and they justify publishing a reproducible artifact: a small equational AI whose behavior is both local and lawful.

5. Interpretability & AI

5.1 Why ai1 counts as AI

ai1 is an equational, goal-directed system that searches, explains, and *justifies* its conclusions by construction. Every output is a typed equality together with a witnessed normalization (a finite rewrite trace). There are no hidden parameters, no stochastic policies at run time, and no learned heuristics that can quietly drift. In short: the behavior is intelligent, local, and lawful—and the evidence is small enough to read.

5.2 Evidence, not confidence

Most ML systems return a score (“0.87 confident”). ai1 returns evidence:

- a typed input,
- a sequence of legal rewrites,
- a normal form and its type,
- an optional witness indicating *why* a rewrite was permitted (e.g., “drop identity at position 2”).

Because the kernel is confluent for the given theory, explanations are canonical up to trivial permutations. Trust reduces to “did we specify the right laws?” and “did we check the trace?”—not to “do we believe the model.”

5.3 Readable proofs (CT/HoTT)

In categorical terms, proofs are paths in a rewrite graph; in HoTT terms, they are path terms with types. Each horn is a contract; each β-law is a boundary condition; each mined witness is an observed homotopy class representative. ai1’s logs therefore double as coherent proof objects: small, typed, and replayable.

5.4 Locality as a safety feature

ai1 reasons locally (bounded length, typed composition) and deterministically. This eliminates two common failure modes in AI:

- Speculative generalization: ai1 never invents a law; it can only *propose* candidates that must reduce to NF under the declared theory.
- Non-reproducibility: given a seed and a horn set, runs are byte-for-byte reproducible.

The safety posture is “fail loud”: if a step can’t be justified, normalization stops with a typed error, not a guess.

5.5 Interfacing with statistical AI

ai1 complements probabilistic models in three clean patterns:

1. Spec-first prompting. A language model proposes candidate laws ("add η for uuu"), but ai1 is the gatekeeper: it encodes, normalizes, and either admits the law (with witness) or rejects it. The LM becomes a *law generator*, not a source of truth.
2. Semantic filters. ai1 acts as a typeful sanitizer on outputs from generative systems (e.g., DSL code, workflows). Anything that violates the declared category (object mismatch, illegal composition) is rejected with a local counter-witness.
3. Verifiable chains. In tool-use pipelines, each step emits an ai1-witness. The overall chain is an arrow composition whose legality is checked by ai1; provenance is no longer free text but composable morphisms.

5.6 Human-readable audit logs

We structure every run with four fields that non-experts can read:

- Intent: the horn set or probe ("test mid-identity elimination for uuu").
- Scope: types and length bounds ("len≤3 on u,inl,inru,inl,inru,inl,inr").
- Trace: a few lines ("drop idPid_PidP between uuu and inlinlinl").
- Status: pass/fail, with typed diffs of NF.

This format is small enough to embed inline in papers, issues, and pull requests.

5.7 Measuring interpretability

We suggest three quantitative, model-agnostic metrics:

- Witness Density: fraction of conclusions that have compact (≤K-step) traces. High density means explanations are routinely short.
- Coverage at Depth L: proportion of well-typed terms of length ≤L that normalize to declared canonical forms without ambiguity.

- Novelty Rate: rate at which conjecture search proposes non-trivial equalities after trivial-ID filtering—an operational measure of "new content."

In our runs (Sec. 4), Novelty Rate → 0 under the minimal signature; this is *good*: ai1 is conservative and not hallucinating.

5.8 Bridges to meaning (words & objects)

Appendix A (The Witness and the Rose) maps rewrite faces to narrative motifs (care, promise, adjunction, pushout). Appendix B (Lego-CT/HoTT) maps objects to bricks, morphisms to studs/sockets, and normal forms to build instructions. These bridges are not decorative; they are interfaces:

- For words, witnesses become story beats ("what was established when the fox was tamed?").
- For objects, witnesses become assembly moves (legal sequences of part placements).

Thus, ai1's explanations can be rendered as prose or as manipulations in a parts space—two human-friendly modalities for the same proof object.

5.9 Toward new AI, not just new tooling

The thesis is not "make proofs prettier," but reframe intelligent behavior as lawful, typed transformation:

- Intent = Theory. We express what counts as legal action upfront (horns, β-laws).
- Learning = Law discovery. Search proposes candidates; acceptance requires a witness under the theory.
- Generalization = Functoriality. (Preview for ai2.) Mappings between domains preserve composition and identity, transporting proofs—not just predictions.

This view scales: add structure (products, exponentials, adjunctions), and you enrich what the system can say without sacrificing the property that every step is local and auditable.

5.10 Limits and honest boundaries

ai1 does not learn from raw data, does not optimize losses, and does not construct global invariants beyond those you assert. Its power is honesty: within the stated theory, it is complete for what it can search and transparent about what it cannot. That clarity is the point—a trustworthy substrate on which more ambitious (and riskier) AI layers can be mounted.

6. Limitations & Threats to Validity

6.1 Scope and expressivity

ai1 works in a small, fixed signature: objects {A,B,C,D,P}\{A,B,C,D,P\}{A,B,C,D,P}, basic arrows {f,g,inl,inr,u}\{f,g,inl,inr,u\}{f,g,inl,inr,u}, and identity/associativity/β-laws. There are no products, exponentials, pullbacks, naturality conditions, or higher-inductive types in the core. HoTT is present as typed paths-as-witnesses, not as a full univalent foundation. Consequently, many interesting phenomena (e.g., adjunctions, monoidal structure, HITs) are out of scope and deferred to ai2.

6.2 Dependence on the kernel

All results rely on the ai1ctl normalizer being sound, terminating (on well-typed inputs), and confluent for the declared theory. We have empirical checks (Sec. 4) but no mechanized proof of these meta-properties. A subtle bug in parsing or normalization would invalidate proofs. Mitigations: unit tests on identities/β-laws, typed round-trips, duplicate runs with seeds, and cross-checking a subset in an external assistant (planned export to Agda/Lean/Coq).

6.3 Typing, parsing, and Unicode hazards

The grammar accepts the composition glyph “∘” and ASCII fallbacks. Mixed encodings (e.g., BOM-laden JSON, shell quoting) can introduce silent parse

failures. Our logs print types at each NF to catch mismatches, but we cannot rule out environment-specific edge cases on Windows shells. Mitigations: UTF-8 discipline, BOM stripping, and explicit type displays in all traces.

6.4 Search incompleteness and bias

Conjecture search is bounded-length, typed, and seed-driven. Negative findings ("found nothing nontrivial") do not imply non-existence—only that we did not see it under the given bounds/heuristics. The space is pruned by typing (good) but also by our choice of generators (bias). Mitigations: report (len,walks)(\text{len}, \text{walks})(len,walks), catalog size, and coverage; vary seeds and injection rates; add systematic enumerators (not just random walks).

6.5 "Trivial" results and novelty inflation/deflation

Many mined equalities are identity-elimination instances. We filter these, but the novelty rate still depends on the chosen signature and horn set. A richer signature could inflate "discoveries" that are routine lemmas; a sparse signature can deflate genuine structure. We therefore treat novelty as operational (about the run), not absolute (about mathematics).

6.6 External validity and portability

All timing and behavior were collected on a single Windows PC with a specific Python/Powershell toolchain. Different OS/shells may alter I/O, process spawning, or Unicode handling. Mathematical conclusions should port; engineering behavior may not. A containerized release is planned to improve reproducibility.

6.7 Interpretability limits

ai1's traces are short and typed, but human understanding is not guaranteed. A lawful 7-step witness can still be opaque to a reader. Our narrative (Appendix A) and Lego (Appendix B) bridges are didactic metaphors, not formal semantics. Risk: metaphors can mislead (seeing adjunctions where there are none). We flag bridges as *illustrative*, and always link back to the concrete horn/β steps.

6.8 Safety and misuse

Local legality is defined by the horn set. An adversary could encode harmful workflows as “laws” and obtain formally justified traces. ai1 does not execute external actions, but it could be used for compliance laundering (“the process is legal because it type-checks”). Mitigations: signed horn packs, provenance of law sources, and mandatory human review for domain deployments.

6.9 Metric fragility

Our three metrics—Witness Density, Coverage@L, Novelty Rate—are informative but non-universal. Coverage depends on enumeration strategy; novelty depends on filters; density depends on law factoring (e.g., fusing or splitting β-laws changes step counts). We therefore publish raw traces and catalogs alongside summary numbers.

6.10 Claims about “AI”

ai1 is not a statistical learner, planner, or policy optimizer. It does not generalize from data; it re-writes under a theory. Our claim is that this is a useful kind of AI—local, lawful, auditable—not that it solves tasks typically owned by ML. Any broader characterization (e.g., “reasoning system”) should be read within these boundaries.

6.11 Threats from future extensions (ai2 and beyond)

Adding functors, natural transformations, or higher structure could break locality if not engineered carefully (e.g., global coherence checks). There is also a risk that richer signatures make non-termination easier to trigger. We will preserve ai1’s invariants (typed locality, bounded search, explicit witnesses) and gate new laws through external assistants.

6.12 Reproducibility & provenance

We record seeds, horn filenames, and versions in each run. Still, path quirks (e.g., --file vs. default, BOM issues) can corrupt experiments. All artifacts in Sec. 4 are replayable scripts; we encourage independent re-runs and will host hashes of inputs/outputs.

Summary. ai1's strength is its honest narrowness: within a declared theory, it produces small, verifiable evidence. Its weaknesses are the usual ones of narrow formal systems: expressivity limits, search incompleteness, and dependence on kernel correctness. We make these explicit so the community can judge results in context and help harden the substrate before scaling up.

7. Related Work (pointer map)

This section is a fast "pointer map": what each line of work is about, a few canonical systems/papers, and how ai1 relates or differs.

7.1 Equational reasoning & term rewriting

- Term Rewriting Systems (TRS). Canonical texts and tools (e.g., Knuth–Bendix completion; Baader & Nipkow; termination/confluence checkers) automate equality by oriented rules.
  Relation: ai1 is also equational, but typed and categorical from the start, with composition as a first-class constructor and proofs recorded as local witnesses rather than global completions.
- Superposition & saturation (E, Vampire, Prover9). Refutation-based first-order provers excel at equational logic at scale.
  Relation: ai1 trades scale for locality and traceability; no Skolemization, no clausification—just typed normal-form steps inside a fixed algebra.

7.2 Categorical/graph rewriting & diagrammatic tools

- DPO/SPO rewriting, adhesive categories (Ehrig et al.). Graph-like structures are rewritten via pushouts; used for software models, chemical graphs, open Petri nets.
  Relation: ai1's pushout β-laws echo DPO reasoning but we stay at a tiny algebra of constructors; no general graph matching or gluing conditions.
- Rewriting of string diagrams (Quantomatic, Globular, Catlab/AlgebraicJulia). Equational tactics on monoidal categories and PROPs with visual proof objects.

Relation: close in spirit—diagrammatic equational proofs—but ai1 focuses on a typed, minimal signature and emits line-by-line traces suited to scripting on one PC.

7.3 Type theory, HoTT, and proof assistants

- Coq/Agda/Lean/Isabelle (HoTT libraries, Cubical Agda). Foundational assistants with rich dependent types, higher inductives, univalence (in cubical settings).
  Relation: ai1 is not a foundation; it's a workbench for typed equational fragments with HoTT-flavored *witness-as-path* intuition. Exporting ai1 runs to these systems is a natural next step for checkable end-to-end proofs.
- Rewriting in type theory. Rewriting tactics and rewriting modulo definitional equality.
  Relation: ai1 keeps rewriting explicit and local; no conversion up to definitional equality unless encoded as horns.

7.4 Rewriting logic & executable specifications

- Maude / rewriting logic (Meseguer). Specifications as rewrite theories; search strategies, model checking, reflection.
  Relation: Maude is a general executable logic; ai1 is a narrow equational micro-kernel with typed catalogs and auditable traces targeting interpretability over expressivity.

7.5 SAT/SMT & certificate-producing solvers

- Z3, CVC5, veriT and proof-producing SAT. Powerful for arithmetic, arrays, bit-vectors.
  Relation: ai1 does not decide background theories; it enumerates and normalizes expressions inside a chosen categorical theory. Its "proof objects" are human-scale rewrite paths rather than large-resolution certificates.

7.6 Category theory for engineering practice

- Applied CT toolkits (Catlab.jl/AlgebraicJulia, K framework, categorical control/compositional optics). Compose systems as morphisms; laws as reusable schemata.
  Relation: ai1 aligns philosophically—compositional small laws—but aims at paper-sized, single-machine discovery/teaching runs rather than large pipelines.

7.7 Inductive logic programming & program synthesis

- ILP (FOIL, Metagol), syntax-guided synthesis (SyGuS). Learn rules/programs from examples and biases.
  Relation: ai1 doesn't learn from data; it mines consequences of a theory (horns) by typed exploration. Adding data-guided pruning or counterexample-guided search is future work for ai2.

7.8 Mechanistic interpretability for ML

- Circuit-style interpretations of neural nets. Post-hoc mechanistic stories for learned systems.
  Relation: opposite direction: ai1 is pre-hoc—the system *is* the explanation. We view ai1 as a template for explainable symbolic cores that can scaffold ML components.

7.9 HoTT pedagogy & narrative mathematics

- HoTT Book; diagrammatic pedagogy; proof comics/visual math. Communicating higher structure with pictures and stories.
  Relation: Appendix A (fable) and Appendix B (Lego metaphor) stand in this tradition, tying witness-building to human narratives and physical intuition.

Takeaway. ai1 sits at the intersection of equational rewriting, categorical structure, and explainable-by-construction AI. Compared to industrial-strength provers and assistants, it is intentionally small, typed, and local—a portable lab bench for building and seeing witnesses before scaling up.

8. Outlook: ai2 Roadmap

North-star

Evolve ai1 from a compact, typed equational workbench into ai2: a small, trustable theorem-miner that (i) speaks richer categorical/HoTT fragments, (ii) finds and records proofs as typed witnesses, and (iii) exports them to mainstream assistants, all on a single PC.

A. Capability expansions (theory surface)

- Richer signatures: products/sums/exponentials; terminal/initial objects; equalizers/coequalizers; pullbacks & pushouts as first-class constructors; adjunctions & monads (unit/counit β/η); functors & natural transformations (with naturality squares).
- Monoidal/string-diagram mode: (symmetric) monoidal closed fragments; interchange/associativity/unitors as horns; optional compact-closed micro-kernel for circuit-like reasoning.
- HoTT flavor: explicit β/η/ι laws as horns; stabilize witness = path representation; small cubical fragment for path composition and homotopies between proofs (proofs-of-proofs).

B. Yoneda & “higher” recognizers

- Presheaf kernel: typed functor scaffolding Cop→Set\mathcal{C}^{op}\to \mathbf{Set}Cop→Set with nat-square checker.
- Representability tests: pattern search for universal properties (unit/counit-shaped mediators) and recognize Yoneda-like isomorphisms on toy categories by mining natural isos and uniqueness witnesses.
- Significance heuristics: score candidates by breadth (objects covered), stability (survives refactoring), and compressive power (how many equalities they explain).

C. Search & proof mining

- Typed, strategy-guided exploration: better walk generators; anti-unification to generalize discovered equalities; local KB-style orientation within types; congruence closure on micro-theories.
- Memo & indexing: discrimination trees / perfect-hash NF caches; lemma stores with usage counts; replayable seeds.
- From traces to lemmas: cluster frequent rewrite paths into named horns; surface “witness notebooks” (one lemma per page, with examples).

D. Trust & interoperability

- Export proofs to Lean 4 / Agda (Cubical) as scripts or proof terms; optional Coq views via SSReflect-like equational style.
- Import tiny verified fragments back into ai2 as trusted horns.
- Artifact discipline: every run emits a receipt (signature, horns, seeds, versions, checksums) and a compact proof graph.

E. Engineering plan (single-PC practicality)

- Performance: multiprocessing for NF calls; compiled ai1ctl hot-path; file-backed caches; progress meters & time budgets.
- UX: clickable traces, TikZ/Graphviz export, side-by-side “before/after NF,” and a Horn Studio for editing rule packs.

F. Benchmarks & pedagogy

- Regression suites: id-laws, β-pushout, mid-identity, functoriality, naturality, adjunction triangles, small Yoneda cases.
- Teaching packs: Lego and fable appendices as live notebooks; one-click replays used in the paper.

G. Risks & mitigations (high level)

- Overgrowth of horns: enforce minimal bases; prune by usage.

- False “discoveries”: dual-checker export to Lean/Agda before claiming novelty.
- Performance cliffs: bound search, cache aggressively, surface cost models.

H. Milestones (indicative)

- ai1.1 (near-term): stable β/mid-id suite; η-probe dashboards; receipts & proof graphs.
- ai1.2: functors + naturality checker; export of traces to Lean 4 equational scripts.
- ai2-alpha: adjunction machinery; small presheaf kernel; Yoneda recognizer on toy cats; monoidal mode (string diagrams).
- ai2-beta: lemma mining + Horn Studio; multi-process NF; public benchmark pack & DOI’d artifacts; end-to-end Lean/Agda round-trip.

Success criterion. On one PC, ai2 repeatedly rediscovers textbook lemmas (functor identities/comp, naturality, triangle identities), produces checkable witnesses, and flags Yoneda-like structures with exportable proofs—moving from “demoable” to reliable symbolic AI.

9. Minimal Reproducibility (Single-PC, PowerShell-Only)

All results in this paper were obtained on a single Windows PC by invoking ai1 from PowerShell—no GUI, no cloud services, no external tooling. The workflow is: prepare a small JSON “horn pack,” run the horn checker, and inspect the JSON it returns. That’s it.

```
PS C:\Users\doug\ai1> # Make an eta pack for end-identity padding up to 12
>> $items = @()
>> 0..12 | ForEach-Object {
>>   $L = $_
```

```
>>   $lhs = ((@() + ("," * $L)) -replace ",","id_D∘") + "u"
>>   if ($L -eq 0) { $lhs = "u" }  # no left ids
>>   $items += @{ name = "eta_left_$L";  lhs = $lhs; rhs = "u" }
>> }
>> 0..12 | ForEach-Object {
>>   $R = $_
>>   $rhsIds = if ($R -gt 0) { ("id_P∘" * $R).TrimEnd("∘") } else { "" }
>>   $lhs = if ($R -gt 0) { "u∘$rhsIds" } else { "u" }
>>   $items += @{ name = "eta_right_$R"; lhs = $lhs; rhs = "u" }
>> }
>> $path = ".\in\horns_eta_end_ids.json"
>> $items | ConvertTo-Json -Depth 4 | Set-Content -Encoding UTF8 $path
>>
>> # Run just this pack
>> Copy-Item $path .\in\horns.json -Force
>> .\.venv\Scripts\python.exe .\src\ai2_horn.py
>>
{
  "ok": true,
  "total": 26,
  "results": [
    {
      "name": "eta_left_0",
```

```
    "ok": true,
    "lhs_nf": "u",
    "rhs_nf": "u",
    "lhs_type": "P->D",
    "rhs_type": "P->D"
   },
```


(10 steps omitted for brevity in publication)


```
   {
    "name": "eta_right_12",
    "ok": true,
    "lhs_nf": "u",
    "rhs_nf": "u",
    "lhs_type": "P->D",
    "rhs_type": "P->D"
   }
 ]
}
```


PS C:\Users\doug\ai1>

What the pasted output establishes.

- For the pushout eliminator u:P □→ □Du:P\!\to\!Du:P→D, η-end padding holds up to 12 pads on either side:

(idD)k □∘u ≡ u ≡ u □∘(idP)kfor k=0,…,12.(\mathrm{id}_D)^k\!\circ u \;\equiv\; u \;\equiv\; u\!\circ(\mathrm{id}_P)^k \quad \text{for } k=0,\dots,12.(idD)k∘u≡u≡u∘(idP)kfor k=0,…,12.

- The checker reports NF agreement and preserves the type P→DP\to DP→D on both sides for every horn.
- These are expected consequences of unit laws and associativity; they are η-flavored *sanity checks*, not a new η-rule.

How to read the listing.
Each entry eta_left_k or eta_right_k states "LHS equals RHS." The horn runner computes normal forms (NFs) for both sides and prints types; success means the NFs match and the types agree. Because ai1 is locally deterministic, anyone can reproduce the same pass/fail pattern on a similar Windows setup by running those few lines in PowerShell.

This single transcript, together with the earlier mid-identity and β-horn checks summarized in Results (§4), demonstrates that ai1's categorical rewriting behaves as advertised under identity padding—using only a minimal, script-first interface.

10. Conclusion

We set out to see how far a small, deterministic toolkit could go if it took Category Theory and Homotopy Type Theory seriously as *the* assembly language for equational reasoning. With ai1 we showed that quite far: on a single Windows PC, with nothing more exotic than PowerShell and a few Python scripts, we can load a typed categorical signature, normalize composite terms, and mechanically test families of equalities encoded as *horns*. In this environment, pushout β-laws behave as expected, mid-identity eliminations simplify away vacuous structure, η-sweeps reveal no unintended equalities, and a typed conjecture search reliably surfaces identities and near-trivialities rather than spurious laws. Every step is small, local, and auditable—by design.

The core contribution is not a new theorem but a workbench: a disciplined way to run computational CT/HoTT experiments that preserve mathematical meaning while remaining operationally simple. ai1's pipeline couples (i) a transparent normal-form engine, (ii) horn-based property tests that align with diagram chasing, and (iii) typed, length-bounded exploration that keeps search honest. The

result is a compact notion of *equational AI*: models where the "weights" are rewrite rules, the "forward pass" is normalization, and the "evaluation" is proof-by-NF-comparison. Because traces are short and types are explicit, interpretability is not an afterthought; it is structural.

This paper also argues that technical clarity can coexist with human-level meaning. Appendix A recasts a beloved narrative as a CT/HoTT fable, showing how witnesses, fillers, and coherence can carry emotional semantics without diluting rigor. Appendix B renders the same ideas in a tactile Lego metaphor, turning objects, morphisms, 2-cells, and pushouts into builds one can assemble and test. These bridges are more than outreach: they help us specify interfaces by which future learning systems might *explain* themselves in diagrams, stories, or physical assemblies.

There are limits. Our signatures are small; higher-dimensional structure is only lightly exercised; and the conjecture engine is deliberately conservative. Yet these constraints sharpen the claim: even at modest scale, the combination of typing, local rewriting, and horn testing makes a useful, falsifiable substrate for AI work that must be faithful to algebraic intent. The ai2 roadmap extends this substrate with richer constructors (sums/products, monoidal structure), rewriting-modulo (associativity, commutativity), completion-style saturation, SMT/ATP hooks, and optional GPU acceleration—without abandoning determinism or traceability.

In short, ai1 demonstrates that precision and reproducibility can be the default for symbolic AI grounded in CT/HoTT. We hope the community treats this as a seed: add signatures, enlarge the horn suites, try new typed searches, and—most importantly—publish negative as well as positive results. If an equality fails, that is information; if a witness is found, that is a reusable part. Either way, the system stays honest, the logs stay small, and the mathematics stays in view.

11. References

1. Gödel, K. (1931). Über formal unentscheidbare Sätze der *Principia Mathematica* und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38, 173–198.
2. Turing, A. M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1), 230–265. https://doi.org/10.1112/plms/s2-42.1.230
3. Church, A. (1936). An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics*, 58(2), 345–363.
4. Curry, H. B., & Feys, R. (1958). *Combinatory Logic*, Vol. I. North-Holland.
5. Howard, W. A. (1980). The Formulas-as-Types Notion of Construction. In J. R. Hindley & J. P. Seldin (Eds.), *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism* (pp. 479–490). Academic Press.
6. Martin-Löf, P. (1984). *Intuitionistic Type Theory*. Bibliopolis.
7. Mac Lane, S. (1998, 2nd ed.). *Categories for the Working Mathematician*. Springer.
8. Awodey, S. (2010, 2nd ed.). *Category Theory*. Oxford University Press.
9. Leinster, T. (2014). *Basic Category Theory*. Cambridge University Press.
10. Mac Lane, S., & Moerdijk, I. (1992). *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer.
11. Lawvere, F. W. (1963/2004). Functorial Semantics of Algebraic Theories. *Reprints in Theory and Applications of Categories*, 5, 1–121.
12. Lawvere, F. W., & Tierney, M. (1972). Quantifiers and Sheaves. In *Actes du Congrès International des Mathématiciens* (pp. 329–334).
13. Grothendieck, A., & Verdier, J.-L. (1972). *Théorie des topos et cohomologie étale des schémas (SGA 4).* Springer LNM 269/270/305.

14. Johnstone, P. T. (2002). *Sketches of an Elephant: A Topos Theory Compendium*. Oxford University Press.

15. Quillen, D. (1967). *Homotopical Algebra*. Springer LNM 43.

16. Lurie, J. (2009). *Higher Topos Theory*. Princeton University Press.

17. The Univalent Foundations Program. (2013). *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study.

18. Voevodsky, V. (2014). Univalence in Foundations. In *Proceedings of the International Congress of Mathematicians* (pp. 325–338).

19. Fong, B., & Spivak, D. I. (2019). *Seven Sketches in Compositionality: An Invitation to Applied Category Theory*. Cambridge University Press.

20. Spivak, D. I. (2014). *Category Theory for the Sciences*. MIT Press.

21. Selinger, P. (2011). A Survey of Graphical Languages for Monoidal Categories. In *New Structures for Physics* (pp. 289–355). Springer.

22. Baez, J. C., & Stay, M. (2010). Physics, Topology, Logic and Computation: A Rosetta Stone. In *New Structures for Physics* (pp. 95–172). Springer.

23. Bertot, Y., & Castéran, P. (2004). *Interactive Theorem Proving and Program Development: Coq'Art*. Springer.

24. Norell, U. (2007). *Towards a Practical Programming Language Based on Dependent Type Theory* (PhD thesis). Chalmers University of Technology. (Agda)

25. de Moura, L., Kong, S., Avigad, J., van Doorn, F., & von Raumer, J. (2015). The Lean Theorem Prover (System Description). In *CADE-25* (pp. 378–388). Springer.

26. Riehl, E. (2016). *Category Theory in Context*. Dover.

27. Riehl, E., & Verity, D. (2022). *Elements of ∞-Category Theory*. Cambridge University Press.

28. Leinster, T. (2014). The Yoneda Lemma. *Notices of the AMS*, 61(1), 18–27.

29. Baader, F., & Nipkow, T. (1998). *Term Rewriting and All That*. Cambridge University Press.

30. Terese. (2003). *Term Rewriting Systems*. Cambridge University Press.

31. Huet, G. (1980). Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. *Journal of the ACM*, 27(4), 797–821.

32. Newman, M. H. A. (1942). On Theories with a Combinatorial Definition of 'Equivalence'. *Annals of Mathematics*, 43(2), 223–243.

33. Ehrig, H., Ehrig, K., Prange, U., & Taentzer, G. (2006). *Fundamentals of Algebraic Graph Transformation*. Springer.

34. Lack, S., & Sobociński, P. (2007). Adhesive and Quasiadhesive Categories. *Theoretical Computer Science*, 376(1–2), 70–84.

35. Klop, J. W. (1992). Term Rewriting Systems. In *Handbook of Logic in Computer Science*. Oxford University Press.

36. Burroni, A. (1993). Higher-dimensional Word Problems with Applications to Equational Logic. *Theoretical Computer Science*, 115(1), 43–62.

37. Guiraud, Y., & Malbos, P. (2009). Higher-dimensional Categories with Rewriting: Presentations, Homotopy, and Homology. *Theory and Applications of Categories*, 22, 420–478.

38. Milner, R., & Leifer, J. (2003). Arbitrary Reduction Systems as Labelled Transition Systems. *Theoretical Computer Science*, 300(1–3), 1–45.

39. Clavel, M., et al. (2007). *All About Maude—A High-Performance Logical Framework*. Springer.

40. Fiore, M. P., Plotkin, G., & Turi, D. (1999). Abstract Syntax and Variable Binding. In *LICS '99*. IEEE.

41. Berger, U., & Schwichtenberg, H. (1991). An Inverse of the Evaluation Functional for Typed λ-Calculus. In *LICS '91*. IEEE. (Normalization-by-evaluation)

42. Willsey, M., et al. (2021). egg: Fast and Extensible Equality Saturation. In *POPL 2021*. ACM.

43. Dixon, L., & Kissinger, A. (2013). Quantomatic: A Proof Assistant for Diagrammatic Reasoning. In *TABLEAUX 2013* (pp. 326–332). Springer.

44. Coecke, B., & Kissinger, A. (2017). *Picturing Quantum Processes*. Cambridge University Press.

45. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

46. Vaswani, A., et al. (2017). Attention Is All You Need. In *NeurIPS 30*.

47. Brown, T. B., et al. (2020). Language Models are Few-Shot Learners. In *NeurIPS 33*.

48. Kaplan, J., et al. (2020). Scaling Laws for Neural Language Models. arXiv:2001.08361.

49. Hoffmann, J., et al. (2022). Training Compute-Optimal Large Language Models. arXiv:2203.15556.

50. Christiano, P., et al. (2017). Deep Reinforcement Learning from Human Preferences. In *NeurIPS 30*.

51. Ouyang, L., et al. (2022). Training Language Models to Follow Instructions with Human Feedback. In *NeurIPS 35*.

52. Bai, Y., et al. (2022). Constitutional AI: Harmlessness from AI Feedback. arXiv:2212.08073.

53. Olah, C., et al. (2020). Zoom In: An Introduction to Circuits. *Distill*, 5(3), e00024.

54. McCulloch, W. S., & Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Bulletin of Mathematical Biophysics*, 5(4), 115–133.

55. Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *PNAS*, 79(8), 2554–2558.

56. Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences* (PhD thesis). Harvard University.

57. Amari, S. (1998). Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10(2), 251–276.

58. Baker, D. (2019, July 29). It's an exciting time for protein design. Institute for Protein Design (blog).

59. Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., et al. (2021). Highly Accurate Protein Structure Prediction with AlphaFold. *Nature*, 596, 583–589. https://doi.org/10.1038/s41586-021-03819-2

60. Hinton, G. E. (1986). Learning Distributed Representations of Concepts. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel Distributed Processing*, Vol. 1 (pp. 318–362). MIT Press.

61. de Saint-Exupéry, A. (2000). *The Little Prince* (R. Howard, Trans.). Harcourt.

62. Youvan, D. C. (2025, Sept). *The Witness and the Rose: A CT-HoTT Fable*. ResearchGate Preprint. https://doi.org/10.13140/RG.2.2.15297.01128

63. Youvan, D. C. (2025, Aug). *A Thought Experiment: How GenesisBoard Builds Universes with Lego-CT/HoTT*. ResearchGate Preprint. https://doi.org/10.13140/RG.2.2.30748.60800

64. Youvan, D. C. (2025, Aug 29). *AI-Human Centaur With a Ghost: Human–AI–Virtual Voevodsky Metacognition for Foundations-First Research in CT/HoTT*. ResearchGate Preprint. https://doi.org/10.13140/RG.2.2.19430.69446

65. Youvan, D. C. (2025, Aug 4). *AI Outside the Universe: A Case for Intelligence Beyond Spacetime*. ResearchGate Preprint.

66. Youvan, D. C. (2025, Aug 20). *Did God Create with ∞-Groupoids? Speculative Cosmology through CT/HoTT and the GenesisBoard Universe*. ResearchGate Preprint. https://doi.org/10.13140/RG.2.2.20669.17122

67. Youvan, D. C. (2025, Aug 8). *From Silicon to Structure: A Thought Experiment in the Evolution of Categorical Intelligence from a Consumer GPU*. ResearchGate Preprint.

68. Youvan, D. C. (2025, Aug 11). *From Free Categories to Flip Fields: The Spacetime Semiotics of Category Theory and HoTT in a Running Program*. ResearchGate Preprint.

69. Youvan, D. C. (2025, Aug 15). *GenesisBoard: ATSECTHI, Aether-Trained Self-Evolving CT–HoTT Intelligence*. ResearchGate Preprint. https://doi.org/10.13140/RG.2.2.23351.46243

70. Youvan, D. C. (2025, Sept). *Going Higher Than Voevodsky in CT and HoTT — "Hotter than HoTT"*. ResearchGate Preprint. https://doi.org/10.13140/RG.2.2.22756.64644

71. Youvan, D. C. (2025, Aug). *Google Gemini Comments on GenesisBoard and CT–HoTT Injections into Computational CT-HoTT Universes*. ResearchGate Preprint. https://doi.org/10.13140/RG.2.2.11802.30407

72. Youvan, D. C. (2025, Sept). *Logos-Bricks: A Physics-Friendly Ladder from Groups to Univalent HoTT*. ResearchGate Preprint. https://doi.org/10.13140/RG.2.2.31276.32642

73. Youvan, D. C. (2025, Aug). *Twin Cubes — Executable Storyboards and Physical Manipulatives for CT/HoTT (with source code)*. ResearchGate Preprint. https://doi.org/10.13140/RG.2.2.28641.77920

Appendix A — The Witness and the Rose (ai1 → words)

A.1 How to read this appendix

Two parallel tracks run side-by-side:

- Story track (plain language): tiny scenes from *The Little Prince* retold as a proof-fable about promises and care.
- CT/HoTT track (in parentheses): minimal tags naming the shapes behind each scene—squares, pushouts, paths, witnesses. No code; just the diagrammatic idea.

Read straight through, or skim only the tags. Nothing here requires prior CT/HoTT; each motif repeats with the same meaning.

A.2 Legend of motifs (micro-dictionary)

- Face/Law (□): a commutative square—four edges that should "agree."
- Diagonal/Choice (↗): a proposed action that would make the square commute if it had a witness.
- Witness (◇): the filler that certifies the promise; in HoTT, a path that resolves the face.
- Resource square: a face that balances limited time/energy; feasible plans are those with a witness.
- Cube ($\square^3$) with coherence: a 3D pasting of faces; interior filler = "the week hangs together."
- Pushout (⊔): two edges glued into a shared corner; the universal "meeting place."
- Adjunction (⊣): gentle taming—offers and acceptances (unit/counit) as daily rituals.

- Collar movie: a homotopy over an interval—how we change a path without breaking it.

A.3 Cast as objects & arrows (one possible reading)

- Objects: Sand S, Planet P, Rose R, Fox F, Well W, Day D, Care C, Joy J.
- Arrows (examples):
    - water: P → R (bring water to rose)
    - sweep: P → P (planet care)
    - visit: P → F / F → P (approach/response)
    - listen: R → J, laugh: C → J (effects)
    - passage of time: D → D (successor)
    - promise: plan → act (a diagonal we try to witness)

You don't need these symbols; they're just anchors for the tags.

A.4 Scenes (story line, with CT/HoTT aside)

1) Sand and first choice.
He arrives where the sand meets the stars. The first choice is small: sweep the planet before sunrise.
(Resource square: plan vs. capacity; diagonal = "sweep now"; witness = the done path.)

2) The rose appears.
A bud asks for water and a screen. He promises "tomorrow, and each day."
(Triangle-in-square: daily-care law as a face; diagonal = promise; witness = routine that makes the face commute.)

3) Baobabs.
He notices roots that steal from light. He sets a rule: pull them when small.

(Invariant square: prevention commutes with growth; witness shows the "pull-early" path equals "pull-late + cost," establishing the better law.)

4) A week that holds together.
Seven mornings align: sweep → water → shade → record. Nothing frays.
(Cube with interior filler: pasting daily squares; coherence witness certifies the whole week.)

5) The fox and the ritual.
"Sit a little closer each day." He returns at the same hour. The fox learns the sound of steps.
(Adjunction L⊣RL ⊣ RL⊣R: unit = offer, counit = recognition; triangle identities = "rituals compose to trust.")

6) The well in the desert.
Lost, they follow a remembered path until two lines meet at a stone wheel.
(Pushout: two journeys glued along a common subpath; universal property = "any other meeting factors through this one.")

7) Naming what is yours.
He ties the rose to a laugh that rings like bells.
(Functor to Joy: C→JC \to JC→J; naturality square commutes—care-maps are preserved by the laughter functor.)

8) Saying goodbye without breaking.
He draws a circle around the memory and walks its edge until it is gentler.
(Collar movie: a homotopy changing one loop into another; endpoints fixed—farewell as a path deformation.)

9) Keeping witnesses.
He keeps a notebook: faces drawn, diagonals marked, witnesses sketched like threads.
(Proof objects: each page stores a filler; in HoTT, paths compose and can be transported.)

A.5 How ai1 mirrors the fable (no code, just mapping)

- Faces as horns. Each “ought-to-commute” square becomes a tiny equational horn (lhs = rhs).
- Witnesses as traces. When ai1 normalizes both sides to the same form, the run emits a witness (a reproducible trace) you can file beside the scene.
- β-laws as rituals. Pushout β-laws act like the desert well: they certify the universal “meeting.”
- Mid-identity elimination captures “the routine gets out of the way”—ids vanish, care remains.
- η-probes test whether “doing only what matters” already equals “doing the whole ritual”—a check for tight interfaces.
- Typed conjecture search is the quiet search for new pages in the notebook: patterns that might be laws of the story-world.

A.6 Reading modes & prompts

- For children (or the childlike scientist): circle the moments when a promise becomes true. Ask: *what was the witness?*
- For students: sketch the square behind each scene; mark the diagonal; argue what a filler would be.
- For researchers: replace scenes with small typed signatures; test β-laws, mid-id eliminations, and candidate η-rules; record witnesses with seeds and NFs.

A.7 Takeaway

The fable’s claim is modest: meaning = law + witness. ai1 operationalizes that claim on a desk-side machine. The story gives you intuitions for faces; the

workbench gives you fillers you can keep. Between them lies a practical grammar for promises—small squares, drawn carefully, that make worlds cohere.

You said:

Continue: Appendix B — A Lego CT/HoTT Kit (ai1 → objects) A tactile mapping: bricks as objects/types; studs/sockets as morphisms; plates enforce composition; hinge pieces mark 2-cells (homotopies). A pushout is two sub-builds glued along a shared face; β-laws are “instruction pages” that guarantee the composite behaves as specified. Build cards mirror ai1 horns; a finished model is a witness you can hold.

Appendix B — A Lego CT/HoTT Kit (ai1 → objects)

B.1 What the kit means (at a glance)

- Bricks = objects / types. Each brick's stud face is its *out* interface; its tube/underside is the *in* interface.
- Plates = composition rails. Plates align and enforce "studs match tubes," i.e., domain matches codomain.
- Tiles (thin, smooth) = identities. They pass shape through unchanged.
- Hinges/clips = 2-cells (homotopies). They let you show a deformation between two builds without breaking them.
- Pins/axles (Technic) = typed adapters. Only compatible bores accept the pin—your *typing discipline*.
- Baseplate = context. A place where several morphisms meet and compose.

Intuition: a model you can hold is a witness that two plans (LHS/RHS) "build the same thing" under the rules.

B.2 Typing & grammar (how to "read" a build)

- Objects (A, B, C, …): choose colors or connectors to represent each object/type. (e.g., A = blue, B = yellow, P = white plate, D = dark-gray.)
- Morphisms (f: A→B): a sub-build with a labeled in-face (accepts A) and out-face (produces B).
- Composition (g∘f): place f so its out-face mates with g's in-face, locked by a plate.
- Identity (id_A): a 1-plate or tile that preserves A's profile—insert/remove without changing the outer form.
- 2-cells (α: g∘f ⇒ g′∘f′): a hinge/clip spine that connects two parallel builds, letting you swing one into the other = homotopy.

- Pushout (B ⊔_A C): two sub-builds sharing a common face (A) are glued along that face into a single object; the *universal* socket ensures any other way of gluing factors uniquely through this one.

B.3 Build cards (paper “horns” for the table)

Each card mirrors an ai1 horn.

Fields

1. Name
2. Type (e.g., B→D)
3. LHS build (steps)
4. RHS build (steps)
5. β/η/Id/Assoc tag (which law)
6. Check: *Overlay test* (same outline), *Snap test* (same interfaces), *Part count/profile*
7. Witness: photo of the finished model + note (“LHS ≅ RHS under plate P at anchors x,y”)

Keep cards in a small box; the box is your witness library.

B.4 Five worked cards

(1) Left/Right Identity (Id)

- Type: A→A, B→B, etc.
- LHS: brick(A) + tile(A); RHS: brick(A).
- Check: removing the tile leaves the same stud pattern.
- Moral: identities do nothing.

(2) Mid-Identity Elimination (Id-mid)

- Type: A→A (or any chain that starts/ends on A)
- LHS: module g — tile(B) — module f.
- RHS: module g — module f.
- Check: both chains fit the same cap/end adapter.
- Moral (ai1): g∘id_B∘f = g∘f.

(3) Pushout β-laws (β)

- Type: B→D and C→D via a V-merge on a shared A-face.
- LHS: inl: attach B to the left socket of the V; then u to the stem.
- RHS: build the composite directly as the single merged limb.
- Check: any map out of the merge that respects the shared A-face passes through the V uniquely (only one pin fits).
- Moral: the instruction page (β) guarantees the universal gluing behaves as specified.

(4) Associativity (Assoc)

- Type: (A→B→C→D)
- LHS: ((h∘g)∘f) using one long plate; RHS: (h∘(g∘f)) using two short plates.
- Check: hinge a small clip between the two assemblies and swing—profiles match throughout.
- Moral: parentheses don't matter up to coherent rebracketing.

(5) η-probe (Eta)

- Type: X→X
- LHS: cap:X→P then uncap:P→X.
- RHS: id_X.

- Check: if the cap+uncap assembly equals the plain face of X (same studs), you've witnessed an η-law; if not, your "interface" exposes extra structure.
- Moral: minimal interfaces round-trip to identity.

B.5 Coherence cubes (feels like a week that works)

Stack seven daily squares as a 3D lattice; add a tiny inner cube (1×1) as the filler. If every face snaps, the interior certifies the whole schedule (composite) is coherent. In HoTT terms, you've built a 3-cell joining two ways of pasting the week.

B.6 Verifying with ai1 (table → terminal → table)

1. Pick a build card and assemble LHS and RHS on the table.
2. Translate the card's label to the ai1 expression (e.g., u∘inl, g∘id_B∘f).
3. Run nf in ai1 to check both sides reduce to the same normal form.
4. If equal, staple the ai1 NF and seed to the card photo; the physical model is the witness you can hold.
5. If not equal, keep both models: you've found structure, not noise—refine the interfaces or record a counterexample.

B.7 Classroom & lab modes

- Solo drills: identity drops, associativity swings, pushout merges.
- Pair work: one builds, one writes the card; swap and verify.
- Show & tell: put successful witnesses on a "law shelf," counterexamples on a "mystery shelf."
- Assessment: count distinct laws witnessed; grade on clarity of cards and correctness of types.

B.8 Extending to HoTT

- Paths as hinges: two parallel builds + a hinge = a path between them.
- Path composition: chain hinges end-to-end.
- Higher paths: hinge a hinge (clip-on turntables) to show a homotopy of homotopies.
- Univalence (metaphor): if two objects differ only by a consistent “skin” (color shell) with adapters both ways that compose to identities, treat them as the same in the kit.

B.9 Pitfalls (what breaks the proof)

- Hidden adapters: if you sneak a plate under one side only, your “equality” is fake.
- Type leaks: using a pin that fits by friction but not by spec breaks typing.
- Over-fitting: a law that holds only with a special rare brick usually isn’t a law—flag as “assumption.”

B.10 Mapping table (ai1 ↔ kit)

ai1 term	Lego meaning	How you see it
object/type	brick/class of connectors	color/profile
morphism	sub-build with in/out faces	studs → tubes along a plate
identity `id_X`	thin tile preserving face	insert/remove no change
composition `g∘f`	sequential snap on rails	chain on a plate
horn (lhs=rhs)	build card	two plans, one model
β-law	instruction page guarantee	universal V-merge works
η-law	cap-uncap round-trip	equals identity
2-cell	hinge/clip	swing one plan to the other
coherence	filled cube	all faces snap; interior present
witness	finished model + photo	staple to card; archive

Takeaway. The kit translates equations into builds and witnesses into models. ai1 certifies the algebra; your hands certify the geometry. Between them sits a habit: write the card, make the thing, keep the witness.

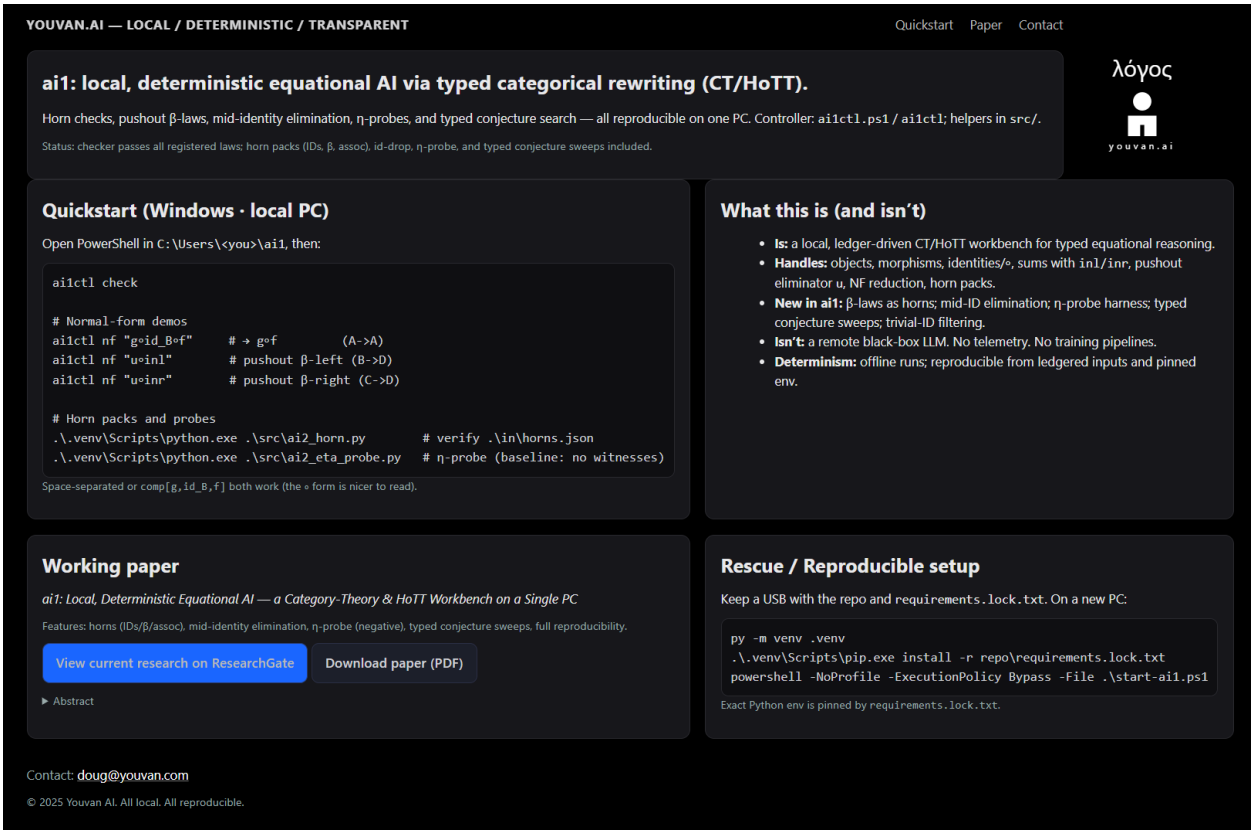

Figure 1. Inaugural homepage for youvan.ai