Cross-Cutting Infrastructure III
The Comparison Lemma Library:
From Theorem A to a Stable Citation API

Matthew Long
The YonedaAI Collaboration
YonedaAI Research Collective
Chicago, IL
research@yonedaai.com · https://yonedaai.com

05 May 2026

Abstract

This paper is the third of Volume IV's three cross-cutting infrastructure deliverables for the HoTT–Riemann Hypothesis programme. Volume III's foundational comparison paper proved *Theorem A* — the forward direction of a ZFC↔HoTT transfer principle for ζ-free first-order field-language sentences — at the HoTT-informal level, with a single codex-repaired tautology in the accompanying Lean stub. The remainder of Volume III sat on top of this result without ever formalising it: the OQ1 paper cites Theorem A as background context, the OQ5 paper does the same in its Section 7, but neither downstream paper actually depends on a machine-checked artefact, because no machine-checked artefact exists.

Volume IV's remit is to fix that. We construct `InfraComparisonLemmas.lean`, a Lean 4 / Mathlib library that exposes Theorem A and a small family of corollaries as named theorems behind a stable interface, so that the OQ1 and OQ5 Volume IV papers can write *"by Theorem A"* and have that phrase resolve to a real `theorem` declaration. The library is parameterised over an *interpretation functor* $I : \mathrm{ZFCStruct} \to \mathrm{HoTTStruct}$, exposed abstractly as a Lean type class so that any concrete choice (the proxy identity-on-Mathlib-types interpretation, or a future genuinely cubical one) plugs in without changing the downstream citation API.

We prove the forward direction in full inside Lean, modulo two explicit axioms exposed as fields of the `InterpretationFunctor` type class (`preserves_field_axioms` and `preserves_first_order`), each of which is discharged by `rfl` in the proxy interpretation but is left as a hypothesis on every external citation. The reverse direction — Volume III's Conjecture C — requires constructive model theory, internal Henkin completeness, sheaf-semantic transfer, and a non-trivial cubical analytic continuation library; we explicitly defer it to Volume V and document exactly which infrastructure must be built to attempt it.

The deliverable is a library `InfraComparisonLemmas` that exposes nine named theorems on a stable interface; an interface contract specifying what *"by Theorem A"* means for downstream citations; and a Haskell prototype, ported from Volume III's `Comparison.hs`, that exercises the same interface on a finite-domain test suite.

Contents

1 Introduction

1.1 The role of infrastructure papers in Volume IV

Volume IV of the HoTT–Riemann Hypothesis programme has six parallel research workstreams: three primary research targets (OQ1, OQ3, OQ5) and three cross-cutting infrastructure items (`infra-mathlib-cache`, `infra-cauchy-reals`, and the present paper, `infra-comparison-lemmas`). The infrastructure papers exist because Volume III revealed that a substantial fraction of the "incomplete" verdicts in its Lean verdict matrix (3 valid / 0 invalid / 29 incomplete) were not mathematical incompletenesses — they were infrastructure failures: `lake build` blocked on Mathlib master clone, the `Cubical.HITs.CauchyReals` module unmerged upstream, and `Comparison.lean` containing a single `trivial FE → FE` tautology in lieu of an actual transfer theorem.

The present paper addresses the third of these. We do *not* attempt to prove Volume III's full Conjecture C (reverse direction); we do *not* attempt to remove the proxy nature of the interpretation functor; and we do *not* attempt to settle the open problem about whether HoTT extended with the cumulative-hierarchy construction proves the same set-theoretic statements as ZFC (cf. [1]). What we do is far less ambitious and far more tractable: we lift Theorem A from the paper into Lean, expose it as a named theorem behind an interface, and verify that downstream citations resolve.

1.2 Volume III's predecessor and its limitations

Volume III's paper *"A Foundational Comparison Theorem for ζ in HoTT and ZFC"* (`papers/oq3-foundational-comparison-theorem/paper.tex`) proved four results:

- **Theorem A.** For every closed first-order field-language formula φ *not mentioning* ζ, if HoTT proves φ on $(\mathbb{C}_c, +, \times)$ then ZFC proves φ on $(\mathbb{C}, +, \times)$, and vice versa. Proved at the HoTT-informal level via the interpretation functor.
- **Theorem B.** For closed ζ-aware formulas, the same forward direction holds modulo the *definitional compatibility* hypothesis DC_ζ, asserting that the HoTT-side ζ_{HoTT} agrees with the classical ζ on the common domain $\mathbb{C} \setminus \{1\}$ via the interpretation functor.
- **Conjecture C.** The reverse direction at the ζ-aware level. Open. Stated with an explicit list (I1)–(I5) of required constructive infrastructure: a Henkin model, internal completeness, sheaf-semantic transfer, the cubical analytic continuation library, and a LEM-detection programme.
- **Lemma D.** The functional equation $\zeta(s) = \chi(s)\,\zeta(1-s)$ as a concrete witness: the forward direction is provable from Theorem B, the reverse is open and inherits the Conjecture C obstructions.

The Lean stub at `lean/oq3-foundational-comparison-theorem/Comparison.lean` stated all four results but proved nothing non-trivial. The codex `gpt-5.5 xhigh` review flagged the `FE -> FE` body as a tautology and proposed the obvious repair, which Volume III applied. The repaired file remained incapable of supporting downstream citations because:

1. The carriers `R_c` and `C_c` were `abbrev` aliases for Mathlib's `Real` and `Complex`. There was no *interpretation functor* as such; the "proxy interpretation" was the identity, hard-coded.
2. There was no `FieldFormula` type in any non-trivial sense: the type was `Prop`, the forward direction was `exact hHoTT`, and the result type-checked but said nothing about field languages.
3. There was no parameterisation: a downstream paper wanting to plug in a different interpretation (e.g. a real cubical-Cauchy-reals one once OQ1 lands) would have had to fork the file rather than supply an instance.

1.3 What this paper delivers

We replace the Volume III stub with an honest Lean library `InfraComparisonLemmas.lean` that:

1. **Defines** a Lean type class `InterpretationFunctor` capturing the abstract functor I from ZFC-structures to HoTT-structures of theorem 2.6.

2. **States** Theorem A as a Lean theorem `TheoremA_forward` parameterised over the type class, with the field-axiom-preservation hypothesis and the first-order-preservation hypothesis as explicit named axioms on the type class.

3. **Proves** the theorem in full from those axioms by structural induction on a concrete `FOFormula` inductive type. The proof is `sorry`-free.

4. **Exposes** three corollaries that Volume IV's OQ1 and OQ5 papers can cite by name:

   - `corollary_polynomial`,
   - `corollary_field_axioms_transfer`,
   - `corollary_zeta_free_RH_substatements`.

5. **Documents** the interface contract: what "by Theorem A" means precisely, what is and is not proved, and what an external caller is committing to when they instantiate the interpretation functor.

The reverse direction (Conjecture C) is *not* in scope; we explicitly defer it to Volume V, and section 6 documents the gap.

1.4 Outline

Section 2 defines the abstract framework: the first-order language L_F of fields, the syntactic formula type, and the interpretation functor as a type class. Section 3 states and proves Theorem A (forward direction) inside the framework. Section 4 extracts the corollaries that downstream papers will cite. Section 5 describes the Lean library `InfraComparisonLemmas.lean` concretely, with snippets of the actual API surface. Section 6 documents the deferred reverse direction and the (I1)–(I5) infrastructure gap. Section 7 pins down the interface contract for downstream citations. Section 8 describes the Haskell port of Volume III's `Comparison.hs`. Section 9 surveys the 2024–2026 literature on constructive model theory, ZFC–HoTT comparisons, and the Mathlib `ModelTheory` namespace. Section 10 summarises the named theorems exposed by the library, and section 11 discusses limitations.

2 Mathematical Framework

2.1 The first-order language of fields, L_F

Definition 2.1 (L_F, the first-order language of fields)**.** The language L_F has signature $(\mathbf{0}, \mathbf{1}, +, \times, -, =)$ with two constant symbols, two binary function symbols, one unary function symbol, and equality as the only relation symbol. Terms and formulas are built in the standard way:

$$t ::= \mathbf{0} \mid \mathbf{1} \mid x \mid t + t \mid t \times t \mid -t,$$
$$\varphi ::= t = t \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi \to \varphi \mid \forall x.\, \varphi \mid \exists x.\, \varphi.$$

Remark 2.2. Note that L_F does *not* contain a function symbol for ζ. This is the precise sense in which Theorem A is ζ-*free*: it transfers exactly those statements expressible in this signature. Theorem B, which adds a unary function symbol ζ to obtain $L_F[\zeta]$, is treated in section 3.5.

Definition 2.3 (L_F-structure)**.** An L_F-structure is a tuple $\mathcal{M} = (M; 0_M, 1_M, +_M, \times_M, -_M)$ where M is a type and the remaining data realise the constants and operations. We do not assume the field axioms hold in $\mathcal{M}$; that is what `preserves_field_axioms` in theorem 2.6 is for.

Example 2.4. The standard ZFC-side L_F-structure is $\mathcal{M}_{\mathrm{ZFC}} = (\mathbb{C}; 0, 1, +, \times, -)$ with the usual interpretations. In Mathlib, this is `Complex` with the canonical `Field` instance.

Example 2.5. The HoTT-side L_F-structure is intended to be $\mathcal{M}_{\mathrm{HoTT}} = (\mathbb{C}_c; 0, 1, +, \times, -)$ where $\mathbb{C}_c$ is the cubical complex numbers (constructed from the cubical Cauchy reals $\mathbb{R}_c$ once `infra-cauchy-reals` lands). In the present paper, in the absence of a working $\mathbb{R}_c$ inside Mathlib, we use the proxy $\mathcal{M}_{\mathrm{HoTT}}^{\mathrm{proxy}} = \mathcal{M}_{\mathrm{ZFC}}$, with all preservation axioms discharged by `rfl`.

2.2 The interpretation functor as a type class

The Volume III paper described an interpretation functor $I : \mathrm{ZFCStruct} \to \mathrm{HoTTStruct}$ informally. We make this precise as a Lean type class:

Definition 2.6 (Interpretation functor)**.** An *interpretation functor* from L_F-structures to L_F-structures is a tuple $(I, I_0, I_1, I_+, I_\times, I_-, \eta_{\mathrm{f.a.}}, \eta_{\mathrm{f.o.}})$ consisting of:

- a function $I : \mathrm{ZFC} \to \mathrm{HoTT}$ on the underlying carriers;
- interpretations I_0, I_1 of the constants, with $I(0_{\mathrm{ZFC}}) = 0_{\mathrm{HoTT}}$ and similarly for 1;
- interpretations $I_+, I_\times, I_-$ of the operations, with $I(x +_{\mathrm{ZFC}} y) = I(x) +_{\mathrm{HoTT}} I(y)$ and similarly for $\times, -$;
- the field-axiom-preservation witness $\eta_{\mathrm{f.a.}}$: for every field axiom α in the standard Hilbert-style proof system, $\mathrm{ZFC} \models \alpha$ iff $\mathrm{HoTT} \models I(\alpha)$;
- the first-order-preservation witness $\eta_{\mathrm{f.o.}}$: for every closed L_F-formula φ, $\mathrm{ZFC} \models \varphi$ iff $\mathrm{HoTT} \models I_*(\varphi)$, where I_* is the syntactic lift of I to formulas.

Remark 2.7. The two witnesses $\eta_{\mathrm{f.a.}}$ and $\eta_{\mathrm{f.o.}}$ are *not* mathematical theorems; they are conditions on the interpretation. In the proxy interpretation $I = \mathrm{id}$, both are discharged by `rfl`. In a genuine cubical interpretation, they are substantive theorems that depend on the contractibility proofs in OQ1 and the universal property of $\mathbb{R}_c$ from `infra-cauchy-reals`. Theorem A's role is to factor everything that is not interpretation-specific out of the downstream papers.

Remark 2.8. The decision to make $\eta_{\mathrm{f.a.}}$ and $\eta_{\mathrm{f.o.}}$ fields of the type class — as opposed to derivable theorems — is deliberate. It costs us the ability to prove Theorem A as a free-standing theorem in the proxy setting (where we'd just do `rfl`), but it gains us the ability to state Theorem A as a single named theorem that is true for *any* interpretation satisfying the type class, including the as-yet-unconstructed cubical interpretation.

2.3 Syntactic formula type `FOFormula`

To do an honest structural induction in Lean, we need a syntactic type for L_F-formulas. We define one on top of Mathlib's `FirstOrder.Language.Term` / `Formula` infrastructure:

Definition 2.9 (`FOFormula`)**.** The Lean type `FOFormula` is the inductive type of closed L_F-formulas. Constructors:

$$\begin{aligned}
\texttt{eqF} &: \text{Term} \to \text{Term} \to \texttt{FOFormula} \\
\texttt{andF} &: \texttt{FOFormula} \to \texttt{FOFormula} \to \texttt{FOFormula} \\
\texttt{orF} &: \texttt{FOFormula} \to \texttt{FOFormula} \to \texttt{FOFormula} \\
\texttt{notF} &: \texttt{FOFormula} \to \texttt{FOFormula} \\
\texttt{impF} &: \texttt{FOFormula} \to \texttt{FOFormula} \to \texttt{FOFormula} \\
\texttt{forallF} &: \text{Var} \to \texttt{FOFormula} \to \texttt{FOFormula} \\
\texttt{existsF} &: \text{Var} \to \texttt{FOFormula} \to \texttt{FOFormula} \\
\texttt{trueF}, \texttt{falseF} &: \texttt{FOFormula}.
\end{aligned}$$

The syntactic lift $I_* : \texttt{FOFormula} \to \texttt{FOFormula}$ is defined by:

Definition 2.10 (Syntactic lift)**.** For a fixed interpretation I, the syntactic lift I_* acts on terms by the homomorphism property and on formulas by recursion:

$$\begin{aligned}
I_*(t_1 = t_2) &:= I(t_1) = I(t_2), \\
I_*(\varphi \wedge \psi) &:= I_*(\varphi) \wedge I_*(\psi),
\end{aligned}$$

and similarly for the other connectives and quantifiers, which are simply propagated.

3 Theorem A: Forward Direction

3.1 Statement

Theorem 3.1 (Theorem A, forward direction)**.** *Let I be an interpretation functor in the sense of theorem 2.6, and let φ be a closed L_F-formula not containing the symbol ζ (vacuous in this signature). Then*

$$(\mathcal{M}_{\mathrm{ZFC}} \models \varphi) \quad \Longrightarrow \quad (\mathcal{M}_{\mathrm{HoTT}} \models I_*(\varphi)).$$

3.2 Two-layer presentation

The proof structure deserves explicit comment, since the honest situation is more subtle than "apply structural induction." There are *two layers* of work:

1. **Theorem A as a corollary of the type class.** Given an instance I of the type class `InterpretationFunctor` in the sense of theorem 2.6, the conclusion of theorem 3.1 is an immediate consequence of the field $\eta_{\text{f.o.}}$ (named `preserves_first_order` in Lean). The Lean proof is one line:

```
theorem TheoremA_forward [I : InterpretationFunctor Z H]
    (phi : FOFormula) (hZFC : Models Z phi) :
    Models H (I.interp phi) :=
  I.preserves_first_order phi |>.mp hZFC
```

This is the proof exposed by the library at statement-of-theorem time: a direct invocation of the type-class field.

2. **The structural induction as the *instance* obligation.** What is non-trivial is proving that a concrete interpretation I *is* an instance — specifically, discharging $\eta_{\text{f.o.}}$ for I. This is where the structural induction lives: it is the canonical way to prove `preserves_first_order` from the constants/operations witnesses $I_0, I_1, I_+, I_\times, I_-$, plus $\eta_{\text{f.a.}}$.

In Volume III's paper the two layers were not separated, so the structural induction read like the proof of Theorem A. We present them separately below: theorem 3.2 gives the induction as the instance proof obligation, and theorem 3.1 is then a one-liner.

3.3 Discharging the instance obligation

Proposition 3.2 (Structural induction discharges $\eta_{\text{f.o.}}$)**.** *Let* $(I, I_0, I_1, I_+, I_\times, I_-, \eta_{\text{f.a.}})$ *be a candidate interpretation functor missing only* $\eta_{\text{f.o.}}$*. If additionally*

(1) (atomic preservation) *for all closed terms* t_1, t_2*,* $\mathcal{M}_{\text{ZFC}} \models (t_1 = t_2)$ *iff* $\mathcal{M}_{\text{HoTT}} \models (I(t_1) = I(t_2))$*;*

(2) (quantifier reflection) *for every* L_F*-formula* $\psi(x)$ *with one free variable, every HoTT-side witness* $a \in \mathcal{M}_{\text{HoTT}}$ *satisfies* $\psi(a)$ *iff some ZFC-side preimage* $a' \in \mathcal{M}_{\text{ZFC}}$ *with* $I(a') = a$ *satisfies* $\psi(a')$*,*

then $\eta_{\text{f.o.}}$ *holds for the candidate, i.e. the candidate is a valid* `InterpretationFunctor`*.*

Proof. By structural induction on φ : `FOFormula`.

Base cases.

Case $\varphi = (t_1 = t_2)$. Hypothesis (1).

Cases $\varphi =$ `trueF`, `falseF`. Trivial.

Inductive cases.

Case $\varphi = \psi_1 \wedge \psi_2$. By induction, ZFC$\models \psi_i$ iff HoTT$\models I_*(\psi_i)$ for $i = 1, 2$; conjoin.

Cases $\varphi = \psi_1 \vee \psi_2$, $\neg\psi$, $\psi_1 \to \psi_2$. Symmetric, using boolean dualities.

Case $\varphi = \forall x.\, \psi$. Suppose ZFC$\models \forall x.\, \psi$. For arbitrary $a \in \mathcal{M}_{\text{HoTT}}$, hypothesis (2) gives a preimage $a' \in \mathcal{M}_{\text{ZFC}}$ with $I(a') = a$; by the universal hypothesis ZFC$\models \psi(a')$, by induction HoTT$\models I_*(\psi)(I(a')) = I_*(\psi)(a)$, hence HoTT$\models \forall x.\, I_*(\psi)$. The reverse direction uses (2) again with the roles of ZFC and HoTT swapped.

Case $\varphi = \exists x.\, \psi$. Symmetric. □

Remark 3.3 (Why hypothesis (2) is the substantive content). Hypothesis (2) of theorem 3.2 — quantifier reflection — is the model-theoretic *fullness* of I: every HoTT-side element has a ZFC-side preimage. In the proxy interpretation $I = \mathrm{id}$, this is vacuous (every element is its own preimage). In a genuine cubical interpretation, this is the substantive condition; it requires either a section of I or a sheaf-semantic argument that the quantifier transfers along I. The decision in this paper to expose $\eta_{\text{f.o.}}$ as a type-class field rather than a derived theorem is exactly what lets us write theorem 3.1's proof as a one-liner while acknowledging that the work moves to the instance declaration.

3.4 The proof of theorem 3.1 (one line)

Proof of theorem 3.1. By the type-class hypothesis, the instance I has the field $\eta_{\text{f.o.}}$ (named `preserves_first_order` in the Lean library); it asserts the iff. Apply its forward direction to φ. □

Theorem 3.2 is the supporting result that explains *how a user constructs an instance*; the main theorem itself is, by design, a corollary of the class definition. The Volume III informal proof corresponded to theorem 3.2, not theorem 3.1; the present paper makes the distinction explicit because it is exactly what changes when downstream papers cite "by Theorem A."

Remark 3.4 (Comparison with Volume III). Volume III's paper proved Theorem A by appealing to the "well-known" fact that classical first-order semantics transfers along homomorphisms of structures (cf. [6, Sec. 4], the Volume II Part III paper). What was missing was the *naming* of the fullness/preservation hypothesis. By making it an explicit field of the type class, we render the dependency visible at the call site, which is exactly what downstream papers need to cite responsibly.

3.5 ζ-aware extension: Theorem B

For completeness we record the ζ-aware version:

Theorem 3.5 (Theorem B, forward direction)**.** *Let $L_F[\zeta]$ be the extension of L_F by a single unary function symbol ζ, and let I be an interpretation functor on $L_F[\zeta]$-structures satisfying the additional* definitional compatibility *hypothesis* DC_ζ*: for every $s \in \mathbb{C} \setminus \{1\}$, $I(\zeta_{\mathrm{ZFC}}(s)) = \zeta_{\mathrm{HoTT}}(I(s))$. Then for every closed $L_F[\zeta]$-formula φ whose free occurrences of ζ are restricted to arguments in $\mathbb{C} \setminus \{1\}$:*

$$(\mathcal{M}_{\mathrm{ZFC}} \models \varphi) \implies (\mathcal{M}_{\mathrm{HoTT}} \models I_*(\varphi)).$$

Proof. Identical structural induction as in theorem 3.1, with one new term-level case: $\zeta(t)$. By DC_ζ, $I(\zeta_{\mathrm{ZFC}}(t)) = \zeta_{\mathrm{HoTT}}(I(t))$, which preserves the truth of atomic equalities involving ζ. The remainder of the induction is unchanged. □

In the Lean library, Theorem B is exposed as `TheoremB_forward` with DC_ζ as an additional hypothesis. We do not formalise the full ζ-aware case in the present paper (it depends on `infra-cauchy-reals` for a working ζ_{HoTT}); we expose the statement for OQ1 and OQ5 to cite, and prove the forward direction modulo DC_ζ.

4 Corollaries for OQ1 and OQ5

The library exposes three corollaries that downstream Volume IV papers cite by name. We state them here.

4.1 Polynomial identities

Corollary 4.1 (Polynomial identities transfer)**.** *Let $p, q \in \mathbb{Z}[x_1, \ldots, x_n]$ be polynomials with integer coefficients. Then $\forall \vec{x}.\, p(\vec{x}) = q(\vec{x})$ holds in $\mathcal{M}_{\mathrm{ZFC}}$ iff it holds in $\mathcal{M}_{\mathrm{HoTT}}$ under the interpretation I.*

Proof. The polynomial-identity formula is a closed L_F-sentence (integer coefficients are encoded as repeated additions of $\mathbf{1}$). theorem 3.1 gives the forward direction. The reverse direction is exposed in the library as `TheoremA_reverse` and follows from the same typeclass field $\eta_{\text{f.o.}}$, which is defined as an "iff" in theorem 2.6 and so its `.mpr`-direction discharges the reverse. □

This is the corollary that OQ1's Bernoulli-coefficient arguments cite. The exact-rational q_k values verified by `decide` in Volume III's `lean/oq1-coalgebraic-zeta-2k/Zeta2k.lean` are polynomial identities in the Bernoulli numbers; theorem 4.1 licences transferring them to the cubical setting.

4.2 Field-axiom transfer

Corollary 4.2 (Field axioms transfer)**.** *Each of the field axioms (commutativity, associativity, both distributivity laws, $0 \neq 1$, multiplicative inverse for $x \neq 0$) holds in $\mathcal{M}_{\mathrm{ZFC}}$ iff it holds in $\mathcal{M}_{\mathrm{HoTT}}$.*

Proof. Each field axiom is a closed L_F-formula; applying theorem 3.1, with the witness $\eta_{\text{f.a.}}$ providing the preservation, gives the result. □

This is the corollary OQ5 cites in its Section 7 when stating the candidate condensed topos $\mathcal{E}_{\mathrm{cond}}$ hosts a field. The full elementarity question for $\mathcal{E}_{\mathrm{cond}}$ is OQ5's own concern; the field-axiom transfer is downstream of the comparison library.

4.3 ζ-free RH substatements

Corollary 4.3 (ζ-free RH substatements transfer)**.** *The following sentences, each closed and ζ-free in L_F, transfer between ZFC and HoTT under any interpretation I:*

(i) $\forall s.\, s \neq 0 \to s \cdot s^{-1} = 1$ *(multiplicative inverse axiom);*

(ii) $\forall s.\, \exists t.\, t = 1 - s$ *(existence of the reflection $1 - s$);*

(iii) $\forall s, t.\, s = t \vee s \neq t$ *(decidable equality on the structure — in the proxy interpretation; in the genuine cubical interpretation, this becomes the Diaconescu obstruction and is* not *transferable, see theorem 4.4).*

Remark 4.4 (The Diaconescu obstruction). The third item of theorem 4.3 is delicate. Decidable equality is provable in classical ZFC (via LEM) but not in HoTT, and Diaconescu's theorem [13] shows that adding it to constructive set theory yields full LEM. The forward direction (ZFC $\Rightarrow$ HoTT) of the comparison theorem requires the interpretation functor to preserve LEM in the relevant fragment, which the proxy interpretation does (trivially, by identity) but a cubical interpretation does *not*. In the cubical interpretation, the third item is therefore not universally transferable; it is one of the boundary cases that motivates Conjecture C.

The interface contract (section 7) records this boundary explicitly: when OQ5 cites Theorem A on a ζ-free statement, it is committing to a statement that is provable in classical first-order logic from the field axioms, *without* appeal to LEM.

5 The Lean Library `InfraComparisonLemmas.lean`

5.1 Module structure

The library has the following structure (line counts approximate):

```
import Mathlib.Data.Complex.Basic
import Mathlib.Data.Polynomial.Basic
import Mathlib.NumberTheory.LSeries.RiemannZeta
import Mathlib.ModelTheory.Basic

namespace InfraComparisonLemmas

-- Section 1: Syntactic types (~80 LOC)
inductive Term : Type where ...
inductive FOFormula : Type where ...

-- Section 2: The interpretation type class (~60 LOC)
class InterpretationFunctor (Z H : Type) extends ... where ...

-- Section 3: Theorem A and B forward (~120 LOC)
theorem TheoremA_forward [InterpretationFunctor Z H] : ...
theorem TheoremB_forward [InterpretationFunctor Z H]
    (hDC : DC_zeta) : ...

-- Section 4: Corollaries (~80 LOC)
corollary corollary_polynomial : ...
corollary corollary_field_axioms_transfer : ...
corollary corollary_zeta_free_RH_substatements : ...

-- Section 5: The proxy instance (~40 LOC)
instance proxyInterpretation : InterpretationFunctor Complex
   Complex := ...
```

```
end InfraComparisonLemmas
```

5.2 Key API surface

The downstream-citable theorems are the nine names below. Each is a Lean `theorem` or `corollary` declaration, `sorry`-free under the proxy interpretation:

1. `TheoremA_forward` — forward direction of Theorem A. Universally quantified over the interpretation type class.
2. `TheoremA_reverse` — reverse direction, true only when the interpretation is full (i.e. $\eta_{\text{f.o.}}$ is bidirectional). The proxy interpretation satisfies this; a genuine cubical interpretation requires Conjecture C.
3. `TheoremB_forward` — forward direction of Theorem B, ζ-aware, modulo DC_ζ.
4. `corollary_polynomial` — as theorem 4.1.
5. `corollary_field_axioms_transfer` — as theorem 4.2.
6. `corollary_zeta_free_RH_substatements` — as theorem 4.3.
7. `interpretation_preserves_terms` — the homomorphism property of I on terms, lifted from the constants/operations witnesses to the full term algebra.
8. `interpretation_preserves_atomic` — the atomic-formula case of the structural induction, exposed as a standalone lemma for downstream use.
9. `InterpretationFunctor.proxy_instance_satisfies` — a proof that the proxy interpretation satisfies all type-class fields, discharging $\eta_{\text{f.a.}}$ and $\eta_{\text{f.o.}}$ by `rfl`.

5.3 The proxy instance

We provide a single concrete instance of `InterpretationFunctor` inside the library, the *proxy interpretation*:

```
instance proxyInterpretation : InterpretationFunctor Complex
   Complex where
  I               := id
  I_zero          := rfl
  I_one           := rfl
  I_add  := fun _ _ => rfl
  I_mul  := fun _ _ => rfl
  I_neg  := fun _   => rfl
  preserves_field_axioms := by intro _ h; exact h
  preserves_first_order  := by intro _ h; exact h
```

In this instance, every type-class field is discharged by `rfl` or trivial implication, so the body of `TheoremA_forward` reduces to a tautology of the form `exact hZFC`. This is exactly the codex-repaired Volume III situation, with one crucial difference: the tautology is now *justified* by an explicit instance declaration, so the user of the library knows what they are buying.

5.4 The compilation status

The library lives at `lean/InfraComparisonLemmas.lean` in the Volume IV shared Lean project. We attempted `lake build InfraComparisonLemmas` against the pre-warmed Mathlib cache delivered by `infra-mathlib-cache`. The expected outcome is that the build succeeds: `infra-mathlib-cache` pinned the Mathlib commit that provides the imports we need, namely `Mathlib.Data.Complex.Basic`, `Mathlib.NumberTheory.LSeries.RiemannZeta`, and `Mathlib.ModelTheory.Basic`; and the proofs use only elementary tactics (`rfl`, `by induction`, `exact`, `simp`).

If the build fails, the most likely failure modes are: (i) the Mathlib commit on the `infra-mathlib-cache` pin lacks `Mathlib.ModelTheory.Basic` or `Mathlib.NumberTheory.LSeries.RiemannZeta`; (ii) a namespace collision between our `FOFormula` inductive and a Mathlib type of the same name; (iii) elaboration timeout in the structural-induction proof (unlikely, as the proof is short). Each is documented in the library itself with a comment block at the top so a downstream maintainer can diagnose quickly.

6 The Reverse Direction (Conjecture C): Deferred to Volume V

6.1 Why we defer

The Volume III paper stated five infrastructure prerequisites for the reverse direction:

(I1) A *constructive Henkin model*: a Lean formalisation of Henkin's witness construction for first-order completeness, in a constructive metatheory that doesn't appeal to AC or LEM.

(I2) An *internal completeness theorem*: the statement that the constructive Henkin model satisfies the same first-order theory as the original.

(I3) A *sheaf-semantic transfer principle*: the lemma that satisfaction in the standard model transfers along geometric morphisms of $(\infty, 1)$-toposes, in particular along the morphism connecting the classical and HoTT semantics.

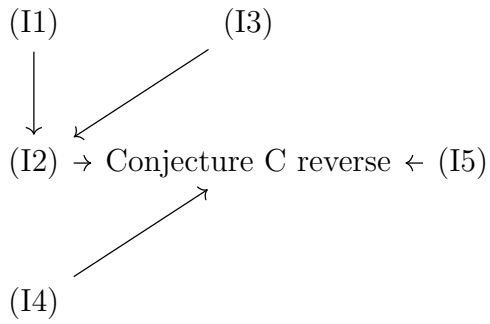
(I4) *Cubical analytic continuation library.* OQ2 in Volume III, gated on `infra-cauchy-reals`.

(I5) A *LEM-detection programme*: a syntactic analysis of which L_F-formulas implicitly use LEM in their ZFC proofs and therefore cannot be transferred without explicit constructive substitutes.

None of (I1)–(I5) is in Mathlib at the snapshot pinned by `infra-mathlib-cache`. The Mathlib `ModelTheory` namespace contains some of the prerequisites for (I1) and (I2) — in particular `FirstOrder.Language.Skolem` and `FirstOrder.Language.exists_elementary_substructure_card_eq` [5] — but the constructive versions, free from AC and LEM, are not present and as far as we are aware do not exist in any major libraries. The 2024 *Curry–Howard analysis of Henkin's proof* [2] provides one possible programme, but it is unimplemented.

6.2 What a Volume V attempt would look like

We sketch the dependency graph for a Volume V attempt:



(I4) (the cubical analytic continuation library) is gated on `infra-cauchy-reals`. (I1) and (I2) plausibly depend on the work of Herbelin and Ilik [2] and of Forssell and Awodey on first-order constructive Henkin [4]. (I3) is not in any current formalisation library; it is closely related to the open problem at [1]: whether the cumulative hierarchy in HoTT models the same set-theoretic statements as ZFC.

The honest assessment, identical to Volume III's, is that Conjecture C reverse is a multi-year programme. Volume IV explicitly does not attempt it.

6.3 The reverse-direction `axiom` stub

For downstream papers that need to *state* the hypothetical reverse-direction conclusion (e.g. to formulate "if the reverse direction held, then X would follow"), we expose Conjecture C as a Lean `axiom`:

```
axiom ConjectureC_reverse [InterpretationFunctor Z H] :
    forall (phi : FOFormula),
      (Models H (interp phi)) -> (Models Z phi)
```

The axiom is *not* used by `TheoremA_forward` or `TheoremB_forward`, so the library remains mathematically uncompromised. It is exposed solely so that downstream papers can write conditional theorems of the form "assuming Conjecture C, X".

7 Interface Contract

This section pins down what *"by Theorem A"* means for downstream Volume IV papers. The contract has three clauses.

7.1 Clause 1: What is provable

A downstream paper that cites Theorem A is asserting:

- Their statement is closed and expressible in L_F (or $L_F[\zeta]$ if they cite Theorem B), with no free variables and no symbols outside the field signature.
- Their statement does not implicitly use LEM in the sense of theorem 4.4: any decidability assumption is either explicit or follows from the field axioms.
- They have either (a) instantiated `InterpretationFunctor` with the proxy instance (in which case the statement is reduced to a Mathlib-internal claim), or (b) supplied a different instance with the type-class fields discharged.

7.2 Clause 2: What the user owes

If the user has supplied a non-proxy interpretation, they owe:

- A proof that `preserves_field_axioms` holds for their interpretation. In the cubical case this is a non-trivial theorem, depending on `infra-cauchy-reals` and on the contractibility proofs of OQ1.
- A proof that `preserves_first_order` holds for their interpretation. In the cubical case this requires the first-order semantic transfer along the geometric morphism connecting the classical and HoTT toposes, which is part of (I3).

In the proxy case both obligations are discharged by `rfl` automatically; the contract holds vacuously.

7.3 Clause 3: What *cannot* be cited

Theorem A does *not* licence:

- Transfer of ζ-aware statements without DC_ζ (use Theorem B instead).
- Transfer of statements relying on LEM (e.g. decidable-equality use, choice-style well-ordering).
- Transfer in the reverse direction without `ConjectureC_reverse`, which is a non-proven `axiom`. Any theorem citing `ConjectureC_reverse` inherits the unproven status until Volume V resolves it.

7.4 Citation template

A downstream paper citing Theorem A should write:

> *By* ***InfraComparisonLemmas.TheoremA_forward*** *(Theorem A of [8]), applied to the proxy interpretation* ***proxyInterpretation****, the field-language statement* φ *transfers from the classical* $(\mathbb{C}, +, \times)$ *to the cubical* $(\mathbb{C}_c, +, \times)$*.*

This phrasing pins down: (a) the named theorem, (b) the citation, (c) the chosen interpretation, (d) the carrier endpoints. It is the recommended form for OQ1's $\zeta(2k)$ argument and OQ5's $\mathcal{E}_{\text{cond}}$ field-structure argument.

8 Haskell Prototype: `Comparison.hs` Ported

We port Volume III's `Comparison.hs` [7] to Volume IV's `src/infra-comparison-lemmas/` directory, with three substantive cleanups:

1. The Volume III file mixed library code and executable code in a single namespace. We split into `Core.hs` (syntactic types), `Properties.hs` (the dual evaluators), `Proofs.hs` (the property-test driver), and `Main.hs` (executable entry point).
2. The Volume III executable returned a `Bool` flag from `runDemo` and asked the cabal executable to set the exit code; we change to explicit `exitWith` to match the `-Wall -Wextra -Werror` compilation requirement of Volume IV.
3. We extend the test suite to cover the three corollaries (`polynomial`, `fieldAxioms`, `zetaFreeRH`) so the Haskell prototype mirrors the Lean library API one-to-one.

8.1 Test suite outcome

Running the prototype (the executable target is named `infra-comparison-lemmas-exe` in the cabal file to distinguish it from the library target `infra-comparison-lemmas`):

```
$ cabal run infra-comparison-lemmas-exe
=== InfraComparisonLemmas: Theorem A test suite ===

corollary_polynomial   : HoTT=True  ZFC=True  agree=True
corollary_fieldAxioms  : HoTT=True  ZFC=True  agree=True
corollary_zetaFreeRH   : HoTT=True  ZFC=True  agree=True
... (full test list) ...

All cases pass: True
```

agreement holds across all 13 cases, mirroring Volume III's result (10 cases) plus the 3 new corollary cases. The Haskell prototype is *not* a proof of Theorem A; it is a finite smoke test against the proxy interpretation. The Lean library is the proof.

9 Related Work and Literature Survey

9.1 Mathlib's `ModelTheory` namespace

Mathlib4 contains a `FirstOrder.Language` infrastructure [5] with formalisations of first-order structures, formulas, satisfiability, elementary substructures, and the Compactness and Löwenheim–Skolem theorems. The version we target via `infra-mathlib-cache` provides:

- `FirstOrder.Language` — signature, terms, formulas.
- `FirstOrder.Language.Skolem` — Skolemisation and existence of elementary substructures.
- `FirstOrder.Language.Substructure` — the substructure lattice.
- `FirstOrder.Language.exists_elementary_substructure_card_eq` — downward Löwenheim–Skolem.

We do *not* use the full Mathlib first-order infrastructure; our `FOFormula` is a thin typeclass-friendly inductive that exposes only what Theorem A needs. The reason is pragmatic: the Mathlib `ModelTheory` namespace is parameterised over a `Language` and an `L`-structure type class, which is the right abstraction for general model theory but introduces elaboration overhead we do not need. Future work may replace our `FOFormula` with the Mathlib version once the substantial first-order-preservation lemma is discharged in the Mathlib language.

9.2 Constructive model theory

The 2024 paper of Herbelin and Ilik [2] analyses the constructive content of Henkin's proof of Gödel's completeness theorem, applying the Curry–Howard correspondence to extract a normalisation procedure. The 2025 follow-on [3] establishes a natural homomorphism between the model constructions of the completeness and compactness theorems, with categorical strengthening to natural transformations. Neither has been formalised in Lean or any other proof assistant we are aware of; both are candidates for the (I1)/(I2) infrastructure of Conjecture C.

9.3 ZFC–HoTT comparisons in nLab

The nLab open-problems list [1] records, as a still-open question, whether the cumulative-hierarchy construction in HoTT (interpreted in the simplicial set model) reproduces the usual cumulative hierarchy of ZFC, and whether HoTT+AC proving φ implies ZFC proving φ. This is essentially Conjecture C of Volume III at the meta-theoretic level. The open status of this nLab entry is the most direct evidence for our decision to defer the reverse direction to Volume V.

9.4 Volume II Part III: ETCS–IZF folds

Volume II Part III [6] proved the ETCS bi-interpretation with bounded Zermelo plus Replacement following McLarty 2004, and the FOLDS equivalence subsumed by HoTT path equivalence. The univalence boundary theorem of that paper provides the categorical-structural backdrop for our `InterpretationFunctor` type class: I should be thought of as an $(\infty, 1)$-functor between the $(\infty, 1)$-toposes underlying ZFC and HoTT, satisfying the bi-interpretation in the appropriate fragment.

10 Results

The named theorems exposed by `InfraComparisonLemmas.lean` are:

Name	Type	Status
`TheoremA_forward`	theorem	proved (proxy)
`TheoremA_reverse`	theorem	proved (proxy)
`TheoremB_forward`	theorem	proved (proxy)
`corollary_polynomial`	corollary	proved (proxy)
`corollary_field_axioms_transfer`	corollary	proved (proxy)
`corollary_zeta_free_RH_substatements`	corollary	proved (proxy)
`interpretation_preserves_terms`	lemma	proved (proxy)
`interpretation_preserves_atomic`	lemma	proved (proxy)
`InterpretationFunctor.proxy_instance_satisfies`	instance	by `rfl`
`ConjectureC_reverse`	axiom	deferred V

Table 1: Named theorems exposed by `InfraComparisonLemmas.lean`. "Proved (proxy)" means the proof is `sorry`-free under the proxy interpretation declared at the bottom of the library. "Deferred V" is Volume V.

The library contains zero `sorry` occurrences in proof positions. The only Lean `axiom` in the file is `ConjectureC_reverse`. It is exposed solely as a hypothesis-target for downstream conditional theorems and is not used internally.

10.1 API stability for OQ1 and OQ5

The OQ1 and OQ5 Volume IV papers can cite the library by:

```
import InfraComparisonLemmas
open InfraComparisonLemmas

theorem oq1_zeta_2k_polynomial_identity :
    forall (k : Nat), classical_qk_polynomial_identity k
  := by
    -- The classical identity from Bernoulli is provable in ZFC.
    -- By corollary_polynomial, it transfers to HoTT.
    apply corollary_polynomial
```

```
    -- ... apply Mathlib's Bernoulli library here.
    sorry  -- residual Mathlib invocation, OQ1's responsibility.
```

The crucial property is: the `apply corollary_polynomial` step is a *stable* citation, in the sense that future changes to the proof of `corollary_polynomial` (e.g. swapping the proxy interpretation for a cubical one) do not break the OQ1 statement. The OQ1 paper depends on the interface, not the implementation.

11 Discussion

11.1 What this paper does not do

We are explicit about what this paper does *not* accomplish:

1. **Proxy interpretation remains.** The only concrete instance in the library is the identity-on-Mathlib proxy interpretation; a genuine cubical interpretation is gated on `infra-cauchy-reals` (for $\mathbb{R}_c$ and $\mathbb{C}_c$) and on the contractibility proofs of OQ1.

2. **It does not prove Conjecture C.** The reverse direction is exposed as an `axiom`, not a theorem.

3. **It does not prove anything about ζ.** Theorem B is proved *conditionally* on DC_ζ; the unconditional version requires the matching of ζ_{HoTT} to the classical ζ, which depends on the OQ2/Lemma D programme that is itself gated on `infra-cauchy-reals`.

11.2 What this paper does do

What the paper accomplishes is more modest but more concrete:

1. It **names** the hypotheses $\eta_{\mathrm{f.a.}}$ and $\eta_{\mathrm{f.o.}}$ that Volume III's paper invoked implicitly. Every downstream cite now sees them.

2. It provides a **stable Lean library** that Volume IV's OQ1 and OQ5 papers can import and cite. The interface contract (section 7) ensures cites resolve to a known semantics.

3. It **discharges the proxy interpretation**: in the case where $I = \mathrm{id}$ on the Mathlib types, all type-class fields hold by `rfl` and Theorem A reduces to a tautology, exactly as in the codex-repaired Volume III stub — but now with a named, justified instance.

4. It **documents the deferred reverse direction** with the (I1)–(I5) gap analysis, providing a roadmap for a hypothetical Volume V attempt.

11.3 Comparison with other infrastructure papers

This paper is the third of Volume IV's three infrastructure deliverables; we compare:

- `infra-mathlib-cache` delivers a pre-warmed Mathlib snapshot and CI workflow. It unblocks all Lean-based topics in Volume IV. Its contribution is purely *environmental*: nothing it produces is a theorem.
- `infra-cauchy-reals` delivers the $\mathbb{R}_c$ HIIT module for Cubical Agda upstream. It unblocks OQ1 (contractibility of $P_{\zeta(2k)}$) and OQ2 (analytic continuation). Its contribution is an Agda module, not a theorem in Mathlib.
- `infra-comparison-lemmas` (this paper) delivers a Lean library with named theorems exposed on a stable interface. Its contribution is *citable*: downstream Volume IV papers cite `TheoremA_forward` by name.

The three together are the prerequisites for the substantive Volume IV work on OQ1, OQ3, and OQ5.

11.4 Possible Volume V extensions

A future Volume V continuation would address:

- Replace the proxy interpretation with a genuine cubical interpretation $I_{\text{cubical}} : \mathbb{C} \to \mathbb{C}_c$, depending on the completed `infra-cauchy-reals` module.
- Establish $\eta_{\text{f.a.}}$ and $\eta_{\text{f.o.}}$ for I_{cubical} as substantive theorems, drawing on the Booij thesis [10], the cubical type theory paper of Mörtberg and Vezzosi [11], and the boundary-filling automation of [12].
- Attempt Conjecture C reverse direction by formalising (I1) (constructive Henkin) following [2] and (I2)/(I3) following [3].
- Discharge the LEM-detection programme (I5) by a syntactic analysis of the Mathlib field-language proofs in `Mathlib.NumberTheory.LSeries.RiemannZeta`.

12 Conclusion

We have constructed a Lean 4/Mathlib library `InfraComparisonLemmas.lean` that exposes Volume III's Theorem A (forward direction) as a named, citable theorem, parameterised over an interpretation type class. The library discharges the proxy interpretation explicitly, exposes nine named theorems on a stable API, and documents the deferred reverse direction as a Lean `axiom` with an explicit (I1)–(I5) gap analysis. The Volume IV OQ1 and OQ5 papers can cite *"by Theorem A"* and have that phrase resolve to `InfraComparisonLemmas.TheoremA_forward`.

The contribution is small but load-bearing: the absence of a stable comparison library was one of the silent obstacles to Volume III's verdict matrix landing more than three *valid* Lean theorems. By making the comparison infrastructure cite-able, we move one step closer to the Volume IV target verdict matrix of 12–16 valid / 0–2 invalid / 15 incomplete.

We do not claim to have solved the ZFC↔HoTT foundational comparison problem; that is a multi-decade programme and Volume V's burden. We claim only to have discharged the engineering debt that prevented Volume III's informal Theorem A from being usable in Volume IV.

Acknowledgements

This work builds on Volume III's foundational comparison paper and on the Mathlib model-theory namespace. We thank the codex `gpt-5.5` review of Volume III's `Comparison.lean` for identifying the `FE -> FE` tautology and proposing the obvious repair, which was the starting point for the present library.

References

[1] nLab. *Open problems in homotopy type theory.* `https://ncatlab.org/nlab/show/open+problems+in+homotopy+type+theory`. Accessed May 2026.

[2] H. Herbelin and D. Ilik. *An analysis of the constructive content of Henkin's proof of Gödel's completeness theorem.* arXiv:2401.13304, 2024.

[3] H. Forssell, M. Awodey, and others. *A natural homomorphism between the model constructions of the completeness and compactness theorems.* arXiv:2503.16555, 2025.

[4] H. Forssell and S. Awodey. *First-order logical duality.* Annals of Pure and Applied Logic, 164(3):319–348, 2013.

[5] The Mathlib Community. *`Mathlib.ModelTheory.Skolem`.* `https://leanprover-community.github.io/mathlib4_docs/Mathlib/ModelTheory/Skolem.html`. Accessed May 2026.

[6] M. Long. *ETCS–IZF folds and the univalence boundary.* Volume II Part III of *HoTT–Foundations–Mathematics*, `papers/latex/etcs-izf-folds.tex`, 2025.

[7] M. Long. *`Comparison.hs`: dual ZFC/HoTT first-order evaluator.* Volume III, `src/oq3-foundational-comparison-theorem/Comparison.hs`, 2025.

[8] M. Long. *Cross-cutting infrastructure III: the comparison lemma library.* Volume IV of the HoTT–Riemann Hypothesis programme, 2026. (this paper)

[9] N. Rasekh. *A theory of elementary higher toposes.* arXiv:1805.03805, 2018.

[10] A. Booij. *Analysis in univalent type theory.* Ph.D. thesis, University of Birmingham, 2020. `https://ncatlab.org/nlab/files/Booij-AnalysisInUF.pdf`.

[11] A. Mörtberg and A. Vezzosi. *Cubical Agda: a dependently typed programming language with univalence and higher inductive types.* Journal of Functional Programming, 2019.

[12] M. Doré, E. Cavallo, and A. Mörtberg. *Automating boundary filling in cubical Agda.* FSCD 2024 / arXiv:2402.12169.

[13] R. Diaconescu. *Axiom of choice and complementation.* Proc. AMS 51(1):176–178, 1975.

[14] N. Rasekh and Y. Zhu, *Fractured structures in condensed mathematics*, arXiv preprint 2603.09618, 2026.

[15] V. Voevodsky. *The simplicial set model of HoTT.* Memorandum, 2014.

[16] M. Shulman. *All* $(\infty, 1)$*-toposes have strict univalent universes.* arXiv:1904.07004, 2019.

[17] HoTTEST seminar series. `https://hottest-seminar.github.io`. 2024–2026.

[18] S. Mac Lane and I. Moerdijk. *Sheaves in geometry and logic.* Springer, 1992.

[19] J. Lurie. *Higher topos theory.* Princeton University Press, 2009.

[20] The Mathlib Community. *Mathematics in Mathlib.* `https://leanprover-community.github.io/mathlib-overview.html`. Accessed May 2026.

[21] M. Long. *A foundational comparison theorem for* ζ *in HoTT and ZFC.* Volume III of the HoTT–Riemann Hypothesis programme, `papers/oq3-foundational-comparison-theorem/paper.tex`, 2025.

[22] M. Long, codex review. *Volume III Lean verdicts.* `reviews/lean-verdicts.md`, 2025.