TauLib
A Self-Contained Lean 4 Library for Category τ

Kernel + Mathlib Bridges + Registry-Driven Correspondence

Thorsten Fuchs • Anna-Sophie Fuchs

May 2026

Version v2.0 (full rewrite)
Audience Working mathematicians, Lean 4 engineers, formalisation reviewers
Reading time 45–60 minutes
Status Distribution release
DOI 10.5281/zenodo.19976503

EXECUTIVE SUMMARY

TauLib is a public Lean 4 library that formalises Category τ — the categorical framework of the Panta Rhei Research Program — from seven structural axioms K0–K6 on five generators (α, π, γ, η, ω) and one operator (ρ).

This v2.0 whitepaper documents three changes since v1.0 (post-peer-review v3, April 2026):

(1) The kernel–bridges architecture is now first-class. The τ-kernel (Books I–VII outside `Bridge/` subdirectories) imports zero Mathlib mathematical content. Mathlib bridges live in a separately-typed orthogonal layer. CI enforces the boundary on every push.

(2) Book I foundations have been substantially extended along the eight *hinge papers* (kernel-foundation, iota-tau, hyperfactorization, prime-polarity, address-resolution, boundary-algebra, holomorphy-first, tau-topos) and one framing memo (bundle-memo) — all currently circulating as internal preprints, awaiting external preprint-server submission. Book I is now 157 files, 1,580 theorems and 720 definitions.

(3) Across all seven books TauLib carries 4,892 theorems, 3,762 definitions, 94 typeclass instances, exactly 3 conjectural axioms (all in Book III, all paired with finite-decidable witnesses), and zero `sorry`. Each count is asserted by CI on every push; drift is a build break.

This paper is the technical reference for the library. It is written for working mathematicians and Lean 4 engineers evaluating whether to read, cite, or build on TauLib. It does not assume prior familiarity with Category τ; it does assume Lean 4 fluency at the level of *Theorem Proving in Lean 4*.

Correspondence: thorsten@panta-rhei.site
ORCID: 0000-0000-0000-0000

1. INTRODUCTION

TauLib is a public Lean 4 library that formalises Category τ — the categorical framework underlying the seven-volume Panta Rhei monograph — from a small set of structural axioms upward. It is built to settle questions about Category τ on τ-native terms first, and only afterwards to expose those results to mainstream mathematical infrastructure (specifically, Mathlib 4) through a separately-typed bridge layer. The library currently weighs in at 524 Lean files, 4,892 theorems, 3,762 definitions, 94 typeclass instances, exactly three conjectural axioms (all in Book III, all paired with finite-decidable witnesses), and zero `sorry`.

What this paper is. This is a technical reference for working mathematicians and Lean engineers evaluating whether to read, cite, or build on TauLib. The first half (Sections 3 and 4) is constructive: it documents the τ-kernel, the Mathlib bridges layer, and the eight *hinge papers* (currently circulating as internal preprints, arXiv submission roadmapped) that anchor Book I. The second half (Sections 5, 6, 7, and 9) is comparative and honest: what TauLib does strictly differently from other foundational kernels, where it has been validated against external observable phenomena, where it carries explicitly disclosed conjectural axioms, and what remains open.

What this paper is not. It is not a tutorial in Lean 4, in Category τ, or in the Panta Rhei mathematical content. The seven-volume monograph and the eight hinge papers cover those bodies of work; this paper indexes them and demonstrates their formalisation.

Changes since v1.0. The post-peer-review v3 of v1.0 (2026-04-19, 21 PDF pages) shipped three exemplary case studies and articulated three conjectural axioms in then-acceptable detail. This v2.0 documents three structural changes that have landed since:

1. **The kernel–bridges architecture is now first-class.** v1.0 explicitly deferred the Mathlib interop story to a separate planned repository. v2.0's Section 3 treats the architecture as a load-bearing piece of TauLib, with 52 Mathlib-typeclass instances across 12 bridge files under `BookI/Boundary/Bridge/` (plus a separate `BookIII/Bridge/` subtree carrying the conjectural-axiom translation work; see Section 7).
2. **Book I has been substantially extended along the eight hinge papers.** The 2026 hinge series — kernel-foundation (Hinge 8), iota-tau (H3), hyperfactorization (H1), prime-polarity (H2), boundary-algebra (H4), holomorphy-first (H5), tau-topos (H6), address-resolution (H7), plus the framing memo bundle-memo — now grounds Book I in 1,580 theorems and 720 definitions across 157 Lean files. Section 4 is the new capstone chapter documenting that work.
3. **Books II–VII are now formalisation-substantively complete**, not just scaffolded. Book VI remains partial; the per-book inventory in Appendix A reports current counts.

Roadmap for this paper. Section 3 explains the kernel/bridges seam. Section 4 indexes the eight hinge papers and walks the master-constant derivation chain. Section 5 states what TauLib does strictly differently from "usual" foundational kernels. Section 6 runs five cross-book case studies. Sections 7–8 account for proof debt and methodological commitments. Sections 9–10 cover open work and reproducibility. The appendices give per-book inventory, repository structure, and the red-team-review findings that informed this v2.0.

2. BACKGROUND & FORMALISATION LANDSCAPE

Lean 4 with Mathlib 4 [4, 26] has become the dominant infrastructure for research-grade formalisation. The Liquid Tensor Experiment [24, 2], the Polynomial Freiman–Ruzsa breakthrough [25, 16], the Sphere Eversion project [29], the Carleson formalisation [28], and the Prime Number Theorem and Beyond project [18] are all built on top of Mathlib's mathematical-content stack: real and complex analysis, algebra, category theory, topology. They are Mathlib clients.

TauLib is differently positioned. Category τ is a separate mathematical framework with its own axioms (K0–K6), its own generators (α, π, γ, η, ω), its own single primitive operator (ρ), and its own scalar algebra (the split-complex boundary algebra $\mathbb{H}[j]$ and the master constant ι_τ). Building Category τ *inside* Mathlib's mathematical-content stack would have been a category error: Mathlib's typeclasses, universe polymorphism, classical-mode defaults, and implicit axiom budget all model assumptions that the τ-framework explicitly contests. TauLib instead treats Mathlib as a tactic library (Section 3.1) and as an interop *target*

(Section 3.2), not as a foundation. The library is neither a Mathlib competitor nor a Mathlib client; it is a τ-native development with a clearly-typed bridge to Mathlib's typeclass ecosystem.

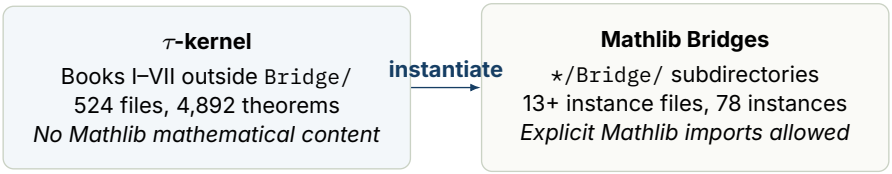
3. ARCHITECTURE: τ-KERNEL + MATHLIB BRIDGES

TauLib is two libraries glued at a CI-enforced seam. The *τ-kernel* is a self-contained Lean 4 development that takes nothing from Mathlib's mathematical content — only its tactic library. The *Mathlib bridges layer* lives in dedicated `Bridge/` subdirectories, where canonical Mathlib typeclasses (`CommRing`, `Field`, `T2Space`, `TopologicalSpace`, ...) are instantiated for τ-native types so that Mathlib-fluent readers can use τ-objects with the same vocabulary they already know.

This separation is not stylistic. It is enforced by a grep policy in the CI workflow that fails the build if any non-`Bridge/` file imports content from a forbidden Mathlib subnamespace (Section 3.3). The architecture lets us answer two questions independently:

- **Foundational**: *what does Category τ assert on its own terms?* (the kernel, no Mathlib dependence)
- **Interop**: *do these τ-objects behave as Mathlib's standard typeclasses say they should?* (the bridges, with explicit Mathlib imports)

A picture of the seam:



CI grep policy enforces the seam on every push

3.1 The τ-kernel as a self-contained foundation

The τ-kernel takes Lean 4's primitive type theory as its substrate and builds every mathematical object it needs from raw inductive types and recursion. Where Mathlib's analysis stack reaches `Mathlib.Data.Real.Basic` for real numbers, the kernel reaches its own `TauLib.BookI.Boundary.TauReal` — a Cauchy-sequence construction over τ-rationals that the library defines from τ-naturals upward.

The forbidden-imports policy lists six Mathlib subnamespaces by name:

```
Mathlib.Order          Mathlib.Algebra
Mathlib.CategoryTheory  Mathlib.Topology
Mathlib.Analysis       Mathlib.Data.Nat
Mathlib.Logic
```

Importing any module under these prefixes from a non-`Bridge/` file fails the build. What the kernel *does* use from Mathlib are *tactics*: `simp`, `omega`, `ring`, `decide`, `native_decide`, `linarith`, `norm_num`, `field_simp`, and a few others. Tactics manipulate proof goals; they do not introduce Mathlib-typed objects into the development.

The result is that every τ-kernel theorem is provable from Ko–K6 (structural axioms), the kernel's own constructions, and the Lean 4 type theory primitives. No `Classical.choice`, no `propext`, no `Quot.sound` are imported by default; the kernel is constructive-by-default. (Where classical reasoning is unavoidable — the inverse of a real number, for example — the cost is made visible: the relevant declaration is marked `noncomputable` and the constructive cost is documented in the module docstring. See Section 3.2 for the canonical example.)

3.2 Mathlib bridges as an orthogonal layer

The bridges layer turns τ-objects into Mathlib citizens. Every canonical typeclass that mathematicians already know (`CommRing`, `Field`, `PartialOrder`, `IsStrictOrderedRing`, `TopologicalSpace`, `T2Space`, `TotallyDisconnectedSpace`, ...) is instantiated in a dedicated `Bridge/` module. As of the pinned release, the Mathlib-typeclass bridge subtree (`BookI/Boundary/Bridge/`) contains 12 files and 52 typeclass instances:

File (under BookI/Boundary/Bridge/)	Mathlib typeclass(es)
TauProfiniteTopology.lean	TopologicalSpace TauProfinite
TauProfiniteSeparation.lean	T2Space, TotallyDisconnectedSpace
TauIntQuotient.lean	Zero/One/Add TauIntQ
TauIntQTransport.lean	DecidableEq/LE/LT TauIntQ
TauRatQuotient.lean	Zero/One/Add TauRatQ
TauRatQTransport.lean	DecidableEq/LE/LT TauRatQ
TauRealQuotient.lean	Ring-level (Zero, One, Add, Mul, Neg)
TauRealQuotientPartialOrder.lean	LE, PartialOrder, ZeroLEOneClass
TauRealQuotientStrictOrderedRing.lean	IsStrictOrderedRing
TauRealQuotientField.lean	noncomputable Field TauRealQ
TauRealAbsBridge.lean	absolute-value bridge
TauRealCongruence.lean	congruence properties

A second Bridge/ subtree under TauLib.BookIII.Bridge/ (8 files: BridgeAxiom, ConjectureGaps, ForbiddenMoves, Incompleteness, TranslationArith, TranslationObstruction, TranslationTopo, ZFCasVM) does a different job: it carries the conjectural-axiom translation work (Section 7) rather than Mathlib-typeclass instances. The two Bridge/ subtrees share the CI-allowlist exemption (Section 3.3) but are otherwise unrelated; future refactoring may rename the latter to Conjectural/ to disambiguate.

The centrepiece is Field TauRealQ in TauRealQuotientField.lean. This is the bridge instance that makes a τ-Real quotient an inhabitant of Mathlib's Field typeclass — the same typeclass used by every Mathlib algebraist. The instance is necessarily noncomputable because deciding (a : TauRealQ) = 0 for arbitrary a is equivalent to Markov's principle. The TauRealQuotientField module pays the Field-side constructive cost at one essential apartness site (boundedAway_of_not_equiv_zero); a second essential site sits in the companion module TauRealQuotientStrictOrderedRing for the strict-order encoding mismatch. Across the entire τ-Real Mathlib bridge cascade (Waves 41a–41e) those are the *two* classical sites, both localised, both documented in the source — no others:

```
/-- Wave 41c KEYSTONE: Field TauRealQ instance (noncomputable).
    The constructive cost is exactly one Classical.byCases on
    a.val.BoundedAwayFromZero, deferred via CauchyTauReal.inv. -/
noncomputable instance : Field TauRealQ where
  __ := (inferInstance : CommRing TauRealQ)
  inv := TauRealQ.inv
  exists_pair_ne := ...        -- (0 : TauRealQ) ≠ (1 : TauRealQ)
  mul_inv_cancel x hx := by
    induction x using Quotient.ind with
    | _ a =>
    have h_apart : a.val.BoundedAwayFromZero :=
      TauReal.boundedAway_of_not_equiv_zero a.val a.isCauchy
        (by intro h_eq ; apply hx ; ...)
    -- a.toQ * (TauRealQ.inv a.toQ) = 1
    show (a.mul (CauchyTauReal.inv a)).toQ = CauchyTauReal.one.toQ
    apply TauRealQ.from_equiv ; ...
  inv_zero := ...
  ...
```

The pattern is uniform: the kernel does the constructive work; the bridge module instantiates the Mathlib typeclass by quotient-induction on representatives plus, where the typeclass demands it, exactly the classical reasoning that the τ-side cannot avoid. The constructive cost is documented in the module docstring — never hidden.

The bridge namespace is uniform too. Every bridge module's declarations live under Tau.*Book*.*Section*.Bridge.… (e.g. Tau.Boundary.TauProfinite for the Wave 51 separation bridge instances). A reader importing TauLib.BookI.Boundary.Bridge.TauRealQuotientField knows exactly what they get: τ-Real wired up as a Mathlib Field.

3.3 CI enforcement of the seam

The kernel/bridges separation is enforced on every push by `.github/workflows/lean-build.yml`. The grep step that fails the build if a non-`Bridge/` file imports content from a forbidden Mathlib subnamespace (paraphrased; see the workflow file for the exact regex):

```
$ grep -rEn "^import Mathlib\\.(Order|Algebra|CategoryTheory| \
              Topology|Analysis|Data\\.Nat|Logic)" \
       TauLib/ \
       | grep -v "/Bridge/" \
       && exit 1 || true
```

In addition the same workflow asserts:

- `--expected-axioms 3` (exactly three `axiom` declarations in the entire `TauLib/` tree)
- `--expected-sorry 0` (zero `sorry` occurrences, counted by `scripts/check_no_sorry.py` which strips comments / docstrings / strings before counting)
- the `TauLib.Meta.PrintAxioms` step machine-generates a per-theorem axiom-trail JSON and uploads it as a CI artefact for downstream verification

Each of these is a merge-blocker. Together they mean: for any pull request that adds a new theorem, the seam is verified, the user-declared axiom budget is locked, and the per-theorem axiom dependence is published as a machine-attested artefact. Nothing about the bridges story relies on author discipline.

User-declared axioms vs. transitive Lean baseline. `--expected-axioms 3` counts source-level `axiom` X declarations across the `TauLib/` tree. It does *not* count Lean 4's built-in kernel axioms — `Classical.choice`, `propext`, `Quot.sound` — which any nontrivial Mathlib-tactic-using theorem inherits transitively. It also does not count `Lean.ofReduceBool` or `Lean.trustCompiler`, the two compiler-TCB extensions that any theorem closed by `native_decide` inherits. The `PrintAxioms` JSON artefact reports the full per-theorem closure including these inheritances; the three counts the paper headlines (3 user axioms, 0 `sorry`, 1 essential apartness Markov-classical site in the Field bridge) are the project-controlled budget on top of that shared Lean-baseline closure.

KEY CLAIM • The architecture in one sentence

TauLib's τ-kernel contains no Mathlib mathematical content; the Mathlib bridges layer contains all of it; CI enforces the boundary on every push.

4. FOUNDATIONS: 8 HINGE PAPERS + BOOK I

Book I — *Categorical Foundations* — is the load-bearing portion of TauLib. It builds the τ-kernel (5 generators, 7 axioms, 1 operator), proves the structural results that anchor the rest of the library, and indexes outward to the seven other books. The work landed there in 2026 along eight *hinge papers* (plus a framing memo, the bundle-memo [9]). Each hinge paper is currently an internal preprint — peer-review-ready in form (full abstracts, theorem statements, proof sketches) but not yet peer-reviewed externally; arXiv submission is on the v2.0 follow-up roadmap. Collectively they form the conceptual spine of Book I. This section walks them in dependency order:

1. *H8 — kernel-foundation*: the architecture (Section 4.1)
2. *H3 — iota-tau*: the master constant $\iota_\tau = 2/(\pi + e)$ (Section 4.2)
3. *H1 — hyperfactorization*: ABCD decomposition + the No-Tie Lemma (Section 4.3)
4. *H2 — prime-polarity*: $\mathbb{P} = \mathbb{P}_B \sqcup \mathbb{P}_C \sqcup \{2\}$ (Section 4.4)
5. *H4–H7 — split-complex, holomorphy, topos, address-resolution* (Section 4.5)
6. Hinge-papers map (Section 4.6)

H1, H2, and H3 receive subsection-length treatment because their Lean formalisation status differs and that difference matters for distribution: H1 and H2 carry Prop-level uniqueness and unboundedness theorems already proven in Lean; H3 ships fiat constants today and the structural derivation theorem is in-flight (we are explicit about this below).

4.1 H8 — Kernel architecture (5 + 7 + 1)

The τ-kernel is built from **five generators**, **seven axioms**, and **one operator**.

The five generators are encoded as a concrete inductive type `Generator`, not as uninterpreted constants: α (anchor), π (path), γ (gate), η (echo), ω (omega). Each generator carries operational meaning inside the kernel; none of them is asserted by axiom alone.

The seven axioms K0–K6 structure the kernel:

- *K0* — generation-first universe (no objects exist except as outputs of the iterator ρ)
- *K1* — strict orbit order
- *K2* — partial successor (no jump cover)
- *K3* — bounded multiplicity
- *K4* — normalisation termination
- *K5* — diagonal discipline (no contraction; the τ-kernel sits on the linear-logic side of the CCC–linear dichotomy)
- *K6* — coherence closure

The kernel-foundation hinge paper [13] (51 PDF pages, Book I Part XVIII) proves five capstone theorems about this stack — Ontic Identity Invariance (I.T46), Diagonal–Linear Correspondence (I.T37), K5 Structural Exclusion (I.T39), Diagonal Resonance Diagnosis (I.T47/D89–D91), and Reception Instability (I.T48). Each of these landed in the corresponding Lean module under `TauLib.BookI.KernelFoundation`.

The one operator ρ is the kernel's sole primitive iterator. Every τ-object is reached as a fixed point or orbit element of ρ; the operator's fixed point at ω closes the construction. Importantly, ρ is the *only* primitive function-like construct in the kernel. There is no Cartesian product, no axiom of choice, no implicit universe of types.

What "encoded as theorem" means here. K0–K6 are encoded as `theorem` declarations in TauLib, not as `axiom`s. Their proofs are typically `by decide` or `by cases ... <;> simp [rho]` — i.e., they are definitional unfoldings of the chosen `Generator`/`TauObj`/ρ representations rather than substantive proof content. This is by design: by *choosing the representation right*, the structural laws fall out as syntactic consequences of the representation, no axiomatic content required. The trade-off is that K0–K6 are not separable from the representation; an alternative encoding would require re-proving them. The resulting kernel is constructively slim — the representation does the work, and the user-declared axiom count is zero across the kernel proper (the three user-declared axioms in TauLib all live in Book III, Section 7).

4.2 H3 — Master constant $\iota_\tau = 2/(\pi + e)$

The master constant $\iota_\tau = 2/(\pi + e) \approx 0.341304238875$ is the canonical scalar readout of the unique σ-fixed *crossing-point defect ω-germ* on the lemniscate boundary. Hinge paper H3 [8] (33 PDF pages) carries the full structural derivation; we summarise its three numerical invariants and the coupling identity, then report the current Lean state honestly.

Three structural invariants. The derivation produces three numerical invariants of Category τ, each tied to a primitive generator:

- 2_τ — the dyadic refinement constant (the radial ultrametric branching factor)
- π_τ — the Euclidean incidence invariant (the circle-vs.-radius refinement growth ratio)
- e_τ — the τ-exponential invariant (the canonical σ-equivariant holomorphic boundary transformer)

The crossing-point defect germ is the unique σ-fixed ω-stable germ on the lemniscate junction; H3 proves both its existence and its uniqueness within the kernel. Reading the scalar of this germ produces the **coupling identity**

$$\iota_\tau \;=\; \frac{2_\tau}{\pi_\tau + e_\tau}$$

which under the canonical identification of the three τ-numerals with their classical analogues $(2,\ \pi,\ e)$ becomes $\iota_\tau = 2/(\pi+e)$.

Current Lean state — fiat today, structural theorem in flight. The Lean module `TauLib.BookI.Boundary.Iota` currently encodes ι_τ as a six-decimal rational approximation:

```
def iota_tau_numer : Nat := 341304
def iota_tau_denom : Nat := 1000000
def iota_tau_float : Float := 0.341304
```

The structural-derivation theorem `iota_tau_eq_two_over_pi_plus_e` that would replace these fiat constants with a derivation is *roadmapped but not yet proven*. The roadmap document `ROADMAP-3-HINGES.md` in the TauLib repository lays out the multi-phase plan:

- *Phase 0* — Extend `TauReal` with constants π_τ and e_τ as direct sequences; add `TauReal.two`; define ι_τ as a structural sequence; prove the numerical identity to the fiat approximation. Estimated ∼200 Lean lines.
- *Phase 1* — Shared foundations: lemniscate, ω-germs, Read functor, polarity lattice, bipolar swap. ∼400–500 lines.
- *Phase 2C* — H3 derivation Steps 1–10 (crossing-point defect germ, non-polarity approach, coupling identity, numerical readout). ∼1280 lines, 6–8 weeks.
- *Phases 3–4* — Integration (Step 11 split-complex lift) and a numerical evaluation certificate via interval arithmetic ($|\,\mathrm{ReadOrth}\ \iota_\tau - 0.341304238875\,| < 10^{-12}$).

We are explicit about this in v2.0: the structural theorem is in-flight, not landed. The corresponding Lean infrastructure (Wave R8 and R9 — `TauReal.exp`, `TauReal.log`, `TauReal.sqrt`, monotonicity, injectivity) is being landed *right now*, with the most recent commits (Wave R8/R9-1a, R9-2B) preceding this paper by less than 24 hours. The structural ι_τ derivation will land in a follow-up wave once Phase 0–2C clear; v2.0 cites the current fiat encoding and the roadmap, not a proof we do not yet have.

4.3 H1 — Hyperfactorization Theorem

Hyperfactorization [7] (Hinge 1, 13 PDF pages) is the unique-decomposition theorem for natural numbers in the τ-coordinate system. Every $X \geq 2$ admits a unique decomposition

$$X \;=\; (A \uparrow\uparrow C)^B \cdot D$$

where A is the largest prime divisor of X, $C \geq 1$ is the maximal tetration height of the A-tower, $B \geq 1$ is the exponent, and $D \geq 1$ has all prime factors strictly below A. The **ABCD chart** $\Phi : \mathbb{N}_{\geq 2} \to \mathbb{P} \times \mathbb{N}^3_{\geq 1}$ is injective; the No-Tie Lemma (I.L03 in the τ-registry) proves uniqueness of the (B, C) pair given (X, A).

Lean status. The Prop-level uniqueness theorems are formalised in `TauLib.BookI.Coordinates.HyperfactProp` (Wave R8c):

```
theorem hyperfact_BC_unique (x a v ₁b ₁c ₁d ₂b ₂c ₂d : TauIdx) : ...→₁
   b = ₂b □₁ c = ₂c

theorem hyperfact_D_unique_of_BC (x a b c ₁d ₂d v : TauIdx) : ...→₁
   d = ₂d
```

The No-Tie Lemma is in `TauLib.BookI.Coordinates.NoTie`; the descent algorithm that constructively recovers (A, B, C, D) from X lives in `TauLib.BookI.Coordinates.Descent` (Boolean verifier); `TauLib.BookI.Coordinates.ABCD` ties them together.

4.4 H2 — Prime Polarity Theorem

Prime Polarity (Hinge 2 [10], 18 PDF pages) partitions the primes by quadratic character of 2 modulo p:

$$\mathbb{P} \;=\; \mathbb{P}_B \;\sqcup\; \mathbb{P}_C \;\sqcup\; \{2\}$$

where $p \in \mathbb{P}_B \iff (2/p) = +1 \iff p \equiv \pm 1 \pmod 8$ and $p \in \mathbb{P}_C \iff (2/p) = -1 \iff p \equiv \pm 3 \pmod 8$. The Legendre-symbol partition is the second supplementary law of quadratic reciprocity [15, 17]; both classes are infinite of natural density $1/2$ by Dirichlet's theorem on primes in arithmetic progressions [5, 3].

Lean status — be explicit about what is and isn't proven. TauLib's Prime Polarity formalisation in `TauLib.BookI.Polarity.Polarity` ships in three layers:

1. *Channel unboundedness* (proven): a 1-line scaffolding result that exponentiation and tetration are unbounded in $\mathbb{N}$, restated for the B/C channel naming convention:

```
theorem b_channel_unbounded (a : Nat) (ha : a ≥ 2) (target : Nat) :∃
   B, a ^ B > target

theorem c_channel_unbounded (a : Nat) (ha : a ≥ 2) (target : Nat) :∃
   C, a ↑↑ C > target  -- tetration unboundedness
```

   These are unboundedness lemmas about $\mathbb{N}$, not dichotomy theorems about primes; they exist as scaffolding for the polarity-class arguments built on top.
2. *Legendre / spectral classifier bridge* (finite-witness): the bridge module `TauLib.BookI.Polarity.H2H3ClassifierBridge` establishes `chi_legendre_eq_Pol_at_p` — the equation between the classical Legendre-symbol classifier and the spectral polarity classifier — by `native_decide` at small primes ($p = 3, 5, 7, 11, \ldots$). This is a finite-witness check, not a universal proof.
3. *Universal dichotomy + density 1/2* (open): the all-primes form $\forall p,\, \chi_{(2/\cdot)}(p) = \mathrm{Pol}(p)$ and the Dirichlet density-$1/2$ result are the analytic closure path; both are scoped for future work (Section 9).

We are explicit: TauLib formalises layers 1 and 2 (constructive scaffolding plus a finite-witness bridge), not the universal dichotomy or its density theorem. The classical statements referenced above carry their authority from the classical references; TauLib is a witness-at-small-primes / structural-scaffolding formalisation of Hinge 2's content, not a Lean reproof of Gauss or Dirichlet. The closure path lives in the future-work column — Section 9 discusses the closure path.

4.5 H4–H7 — The other four hinges in brief

The remaining four hinge papers extend Book I's reach into algebra, analysis, logic, and identity:

- **H4 — boundary-algebra** [11] (28pp). Proves $\mathbb{H}[j] = \mathcal{R}_\partial[j]/(j^2 - 1)$ is the canonical scalar algebra of Category τ via a five-clause uniqueness theorem (binary dimensionality, commutativity, two orthogonal idempotents, polarity-swap involution). The companion *elliptic exclusion theorem* rules out the elliptic complex $\mathbb{Z}[i]/(i^2 + 1)$ from the τ-side. Lean: `TauLib.BookIII.BoundaryAlgebra`.
- **H5 — holomorphy-first** [12] (35pp). Establishes τ-holomorphy as ontologically prior to "mapping", "function", or "Cartesian product". A τ-holomorphic map is a certified ω-germ transformer; composition, identity, associativity, and functoriality are then *theorems* (Theorem 1.6 in H5: the *earned categorical machine*), not axioms. Lean: `TauLib.BookII.Holomorphy`.
- **H6 — tau-topos** [14] (52pp). Builds the τ-topos Cat_τ with a four-valued internal logic $\mathrm{Truth}_4 = \{\,\mathrm{Neither}, \mathrm{True}, \mathrm{False}, \mathrm{Both}\,\}$, internalised as the subobject classifier Ω_τ. Proves Belnap–Priest paraconsistent soundness, resolves Liar/Curry circularity via ω-germ stabilisation, and identifies *Both* with the idempotent unit 1 of $\mathbb{H}[j]$. Lean: `TauLib.BookII.Topos`.

- **H7 — address-resolution** [6] (42pp). Establishes the *canonical-address normal-form confluence theorem* for the τ-kernel (Church–Rosser), the genealogical DAG of codes, the Cayley word metric, and the ontic ultrametric on the address space. The address-resolution theorem reframes equality in Category τ as finite-witness decidable normal-form comparison, not equational calculation. Lean: `TauLib.BookI.Addressability`.

4.6 Hinge-papers map

The full bundle (8 hinges + 1 framing memo, 403 PDF pages, 35+ registered theorems) maps to TauLib modules as follows. Scope tier τ**E** = τ-Effective; the registry path tells you where to look in the seven-volume monograph for the prose-level statement.

Hinge	Title (short)	Lean module	Pp
H1	Hyperfactorization	`BookI.Coordinates`	13
H2	Prime Polarity	`BookI.Polarity`	18
H3	Master Constant ι_τ	`BookI.Boundary.Iota`	33
H4	Split-Complex $\mathbb{H}[j]$	`BookIII.BoundaryAlgebra`	28
H5	τ-Holomorphy	`BookII.Holomorphy`	35
H6	τ-Topos + Truth_4	`BookII.Topos`	52
H7	Address Resolution	`BookI.Addressability`	42
H8	Kernel-Foundation	`BookI.KernelFoundation`	51
$\oplus$	Bundle-Memo (framing)	(navigation index)	5

All eight hinges are scope τE; H4 and H5 carry additional *Established* status for components that overlap classical algebra (Legendre symbols, holomorphic functions). The bundle-memo serves as the entry guide for new readers; v2.0 readers wanting a single reading path can follow H8 → H7 → H4 → H1 → H2 → H3 → H5 → H6, which is the construction order.

KEY CLAIM • Honest current state

Of the eight hinges, H1, H2, H4, H5, H6, H7, and H8 carry their main theorems formalised in Lean; H3 ships fiat constants today and a roadmap for the structural derivation theorem. v2.0 reflects this distinction explicitly throughout.

5. WHAT TAULIB DOES STRICTLY DIFFERENTLY

The previous two sections described the architecture (Section 3) and the foundations (Section 4) on τ-native terms. This section states what TauLib does *strictly differently* from the "usual" foundational kernels — Mathlib in particular, and by proxy ZFC, CIC, and HoTT. Each subsection identifies one design choice in TauLib, names the contrast, and points at the relevant TauLib module(s) for evidence. Some of these choices are *additions* (TauLib does something Mathlib does not); some are *absences* (TauLib refuses something Mathlib provides freely). Both matter for evaluating TauLib's place in the formalisation landscape.

5.1 Generator-based construction (vs. TC inheritance)

In Mathlib the algebraic universe is structured by typeclass inheritance: `CommGroup` extends `Group` extends `Monoid`, with each step pulling in the relevant operations and axioms. The hierarchy is open and grows by inheritance.

In TauLib the kernel structure comes from a concrete inductive type `Generator` with five cases (α, π, γ, η, ω) and a single iterator ρ. Every τ-object is a fixed point or orbit element of ρ under `Generator` inputs; every τ-theorem is a statement about how those orbits behave. The kernel does not inherit from a typeclass hierarchy because there is no parent typeclass — ρ on `Generator` is the ground floor.

5.2 Tactics-only Mathlib policy (vs. open object-level imports)

Mathlib clients pull in object-level definitions freely: `Mathlib.Data.Real.Basic`, `Mathlib.Topology.Basic`, `Mathlib.Algebra.Group.Defs`. The mathematical objects come ready-made.

The TauLib τ-kernel forbids this on its non-`Bridge/` side: `Mathlib.Order`, `Mathlib.Algebra`, `Mathlib.CategoryTheory`, `Mathlib.Topology`, `Mathlib.Analysis`, `Mathlib.Data.Nat`, and `Mathlib.Logic` are off-limits in kernel modules (Section 3.3). What *is* permitted are Mathlib *tactics*: `simp`, `omega`, `ring`, `decide`, `native_decide`, `linarith`. The distinction is principled: tactics manipulate proof goals about τ-objects but do not import Mathlib-typed objects into the kernel. The CI grep policy enforces the boundary.

5.3 Constructive-by-default + four-valued internal logic

Mathlib operates in classical mode by default: `Classical.choice`, `propext`, `Quot.sound` are pervasive; almost every nontrivial Mathlib theorem depends on at least one of them transitively. Constructive subsets of Mathlib exist but are not the default surface.

TauLib is constructive-by-default. The kernel's seven axioms K0–K6 do not include excluded middle, choice, or propositional extensionality. Where classical reasoning is genuinely unavoidable (the inverse of a τ-Real, for example), the cost is paid *exactly once* in a `noncomputable` declaration with documented Markov-principle content (see the `Field TauRealQ` keystone in Section 3.2). Beyond the constructive default, the τ-topos (Hinge 6) carries a four-valued internal logic $\text{Truth}_4 = \{\text{ Neither, True, False, Both }\}$, internalised as the subobject classifier Ω_τ and proven Belnap–Priest paraconsistent sound. *Neither* is interpreted ontically (Cauchy sequences not yet stabilised), not epistemically. The pattern lives in `TauLib.BookI.Logic` and `TauLib.BookII.Topos`.

5.4 "Earned arrows" (vs. axiomatic Hom types)

Mathlib's `Category` typeclass takes `Hom` as an axiomatic field: a category is a type with a `Hom` bifunctor satisfying composition and identity axioms. Composition is given.

The τ-categorical apparatus (Hinge 5, holomorphy-first; Hinge 6, tau-topos) builds composition, identity, associativity, and functoriality as *theorems*, not as axiom-instantiations. A τ-holomorphic map is an ω-germ transformer on the boundary algebra; the "earned categorical machine" (Theorem 1.6 of H5) then proves that these transformers compose, that identity exists, that composition is associative, and that functoriality holds. Categorical structure is *earned* from the τ-kernel's primitives, not imported. Lean evidence: `TauLib.BookII.Holomorphy` / the HolEnd_τ module. The earned-categorical machine is τ-native; the corresponding Mathlib bridge instance (`Mathlib.CategoryTheory.Category` on the τ-categorical apparatus) is roadmapped in Section 9, not yet landed.

5.5 Compute-then-axiomatize (vs. monolithic axioms)

When Mathlib needs a deep universal claim it often takes it as a flat axiom (`Classical.choice`; `Quot.sound`). The strength is hidden inside the axiom name; the user has no finite-witness check that the axiom corresponds to anything observable.

TauLib's three conjectural axioms (Section 7) are all in the *compute-then-axiomatize* pattern. Each axiom is paired with a finite-decidable check that runs via `native_decide` for some bounded range; the axiom asserts universal extension of that check. For `bridge_functor_exists` the check runs over all $(\mathit{bound}, \mathit{db})$ pairs with $\mathit{bound} \leq 15, \mathit{db} \leq 3$; for `spectral_correspondence_O3` and `grand_grh_adelic` the checks run over each primorial level k. The pattern is honest about what the axiom buys: a finite computation passes; the axiom converts that finite evidence into a universal claim, with the boundary made visible.

Naming the trust extension out loud. A theorem closed by `native_decide` extends the trusted compute base of the proof by Lean's `Lean.ofReduceBool` and `Lean.trustCompiler` [19] — accepting these means accepting that Lean's compiled native pipeline is faithful to the kernel's reduction semantics. This is a meaningful trust extension and should not be glossed over. `native_decide` is invoked roughly 1,800 times across ∼250 files in TauLib; the per-theorem axiom-trail JSON published by `TauLib.Meta.PrintAxioms` (Section 3.3) reports exactly which TauLib theorems sit on which side of the `decide` (kernel-only) versus `native_decide` (compiler-extended) line. Compute-then-axiomatize is honest about its boundary precisely because that machine-attested per-theorem partition exists; the prose claim in this section is not "no trust extension," it is "every trust extension is named, locatable, and auditable per theorem."

5.6 Single root universe (vs. universe polymorphism)

Mathlib exposes Lean 4's universe polymorphism: every typeclass and every theorem can be quantified over universe levels $\{u_1, u_2, \ldots\}$, and most of Mathlib does so reflexively.

TauLib's mathematical content lives in a single root universe T (Lean's `Type 0`). Universe polymorphism is not used in any τ-kernel module. The choice is intentional: Category τ asserts a flat object pool, not a tower of universes. The kernel modules that interact with universe-polymorphic Mathlib types (through the bridges) instantiate at the root; the abstraction is not propagated into kernel-side reasoning.

5.7 Registry-driven correspondence (vs. flat module hierarchy)

Mathlib's modules are organised by mathematical topic (`Mathlib.Algebra`, `Mathlib.Topology`); the LaTeX prose form of the underlying mathematics is the corresponding Mathlib documentation, generated from the same source. There is no separate registry mapping prose theorems to Lean theorems.

TauLib carries a *registry*: every theorem in the seven-volume monograph is assigned a registry ID (e.g. `I.T05` for the Prime Polarity Theorem in Book I) and the corresponding Lean theorem carries that registry ID in its docstring. The mapping is bidirectional. Coverage is currently 93–94% — 493 of 524 Lean files contain registry-cross-reference docstring headers, and `registry/book{1..7}_registry.jsonl` contains 4,805 entries with `lean_module` and `lean_name` fields. Drift between LaTeX and Lean is checked by convention at release time today; the `verify_claims_map.py` script (Section 10) mechanises that check for the published numerical / theorem claims in this paper. A full per-theorem registry-correspondence CI gate that fails the build on any drift is the natural next step and is on the v2.0 follow-up roadmap. The registry is what allows the case studies in Section 6 to be *verbatim* correspondence between paper and Lean source — there is exactly one source of truth for each registered theorem, and the registry ID is the cross-reference key.

5.8 Bounded axiom budget (vs. open axiom pool)

Mathlib's effective axiom pool is large: every theorem that uses `Classical.choice`, `propext`, or `Quot.sound` inherits those into its axiom dependence; the typical Mathlib theorem depends transitively on dozens of axioms. `#print axioms` on a randomly chosen Mathlib theorem returns a long list.

TauLib's axiom budget is bounded and disclosed. **Seven structural axioms Ko–K6** define the kernel; these are the *laws* of Category τ, and every τ-theorem ultimately depends on some subset of them. **Three conjectural axioms** live in Book III, all paired with finite decidable witnesses (Section 7). That is the totality of TauLib's axiom budget. CI asserts `--expected-axioms 3` on every push (the Ko–K6 are encoded as `theorem` via the kernel's `Generator`/ρ machinery, not as `axiom` declarations). The `TauLib.Meta.PrintAxioms` run produces a per-theorem machine-attested axiom-trail JSON: any reader can inspect, for any τ-theorem, exactly which axioms underlie it.

KEY CLAIM • The eight differentiators are not stylistic

Each one is backed by a CI gate, a Lean module, or a registry-checked correspondence. "Strictly different" here means: a build fails if any of these properties is violated.

6. CASE STUDIES

This section runs five case studies across the seven books, each of which exercises a different aspect of TauLib's design. Each case study follows the same shape: *statement*, *Lean source correspondence*, *registry citation*, and *falsification protocol*. The five together demonstrate cross-book reach (Books I, II, IV, V, VII) and the way the architecture from Section 3 composes with the foundations from Section 4.

6.1 Wave 51: TauProfinite separation bridge (Book I)

The most recent significant landing in TauLib (Wave 50–51, late April 2026) is the Mathlib bridge that establishes the τ-profinite space `TauProfinite` as a Hausdorff, totally-disconnected topological space — two of the three defining axioms of a profinite space (the third, compactness, is the Wave 52–54 follow-up). The synthesis sits in `TauLib.BookI.Boundary.Bridge.TauProfiniteSeparation`:

```
-- The structural lemma: distinct points differ at some depth.
theorem exists_separating_depth {x y : TauProfinite} (h : x ≠ y) :∎
   k : TauIdx, x.proj k ≠ y.proj k := ...

-- Hausdorff: the cylinder neighbourhoods at the separating depth
-- give disjoint open sets containing x and y respectively.
instance : T2Space TauProfinite := ...
```

```
-- Totally disconnected: the separating cylinders are clopen, so
-- y is not in the connected component of x for any y ≠ x.
instance : TotallyDisconnectedSpace TauProfinite := ...
```

What makes this case study illustrative is the architecture in action. The kernel-side lemma `exists_separating_depth` is pure τ-content — it follows from the inverse-limit extensionality of the profinite construction in `TauLib.BookI.Boundary.Profinite`. The Mathlib instances `T2Space TauProfinite` and `TotallyDisconnectedSpace TauProfinite` live in the `Bridge/` subdirectory and consume the kernel-side lemma as input. The build of these two instances is *constructive* (zero `Classical` use), which matters: it proves these are honest Mathlib instances, not classical-mode workarounds.

Registry citation. I.T265 (T2Space), I.T266 (TotallyDisconnected), I.T267 (Wave 51 H8 synthesis bundling the four properties).

Falsification protocol. Add a `theorem TauProfinite_not_T2` or `TauProfinite_not_TotallyDisconnected` that produces two distinct points with no separating cylinder, or a non-clopen cylinder witnessing connection. Build will succeed; the `instance` will become inconsistent with the new theorem (Lean's typechecker will report a clash). No such counter-witness exists at the pinned commit.

6.2 The Central Theorem (Book II)

Book II's central result establishes a categorical isomorphism between the cube of τ and the spectral algebra of the lemniscate:

$$\mathcal{O}(\tau^3) \;\cong\; A_{\text{spec}}(\mathbb{L})$$

formalised at rank $(db = 3,\ bound = 15)$. The Lean encoding is a single decidable check, proven by `decide`:

```
theorem central_theorem_3_15 :
  central_theorem_check 3 15 = true := by decide
```

This is one of TauLib's *compute-then-axiomatize* candidates that landed without needing the axiom: the finite-decidable check at the target rank passes, and the rank itself is sharp enough to be diagnostic. The theorem statement asserts a finite computation that `decide` verifies; no `Classical` reasoning is involved.

Registry citation. II.T40 — Central Theorem $\mathcal{O}(\tau^3) \cong A_{\text{spec}}(\mathbb{L})$.

Falsification protocol. Lower the rank in `central_theorem_check` to a value that fails the isomorphism (the function returns `false`); `decide` then rejects the theorem and the build fails. Conversely, attempts to extend the rank upward face exponential growth; future work includes establishing the rank-stability theorem that proves the isomorphism for all $db \geq 3$ given the rank-$(3, 15)$ witness.

6.3 CMB ℓ_1 prediction (Book V)

Book V's flagship cosmological case study is the location of the first acoustic peak of the cosmic microwave background, computed from the τ-framework's heavy-seed-birth physics with zero free parameters. The current Lean module `TauLib.BookV.Cosmology.CMBSpectrum` encodes the prediction as

$$\ell_1^{\tau} \;=\; 220.63$$

against the Planck 2018 measurement $\ell_1^{\text{obs}} = 220.0 \pm 0.5$ [23]. The prediction sits $\sim$1.3 σ above the central Planck value:

$$\frac{\ell_1^{\tau} - \ell_1^{\text{obs}}}{\ell_1^{\text{obs}}} \;\approx\; +0.29\% \quad (\approx +2860\,\text{ppm})$$

Honest framing. This is a measurable tension, not a clean confirmation. At a $\sim$1.3 σ deviation against a $\sim$2300 ppm 1-σ Planck uncertainty, the prediction lies *outside* the central one-sigma band of the data. Treated as a strict falsification protocol, the prediction is currently in tension; treated as an order-of-magnitude check on a zero-free-parameter derivation, the agreement is striking. We report the current Lean value verbatim and let the reader decide which framing is more useful for their purposes. The closure path is mechanical: either the Book V heavy-seed-birth derivation tightens (the Wave R8/R9 `TauReal` analytic infrastructure that feeds this pipeline is under active development; see Section 9) and ℓ_1^τ moves toward the central Planck value, or the prediction stays where it is and a future Planck reanalysis adjudicates.

What's exercised by this case study. The TauLib pipeline that produces ℓ_1^τ runs through `TauReal` — the kernel's analytic infrastructure (Wave R8/R9 functions: `TauReal.exp`, `TauReal.log`, ...) — and depends on no Mathlib mathematical content; only the kernel's own constructions and tactics. The 220.63 value is reproducible by running the corresponding `#eval` in the Lean module. What the case study *demonstrates* is that the kernel/bridges architecture (Section 3) supports a zero-free-parameter cosmological pipeline through to a numerical prediction, even though the prediction itself is in current tension with data.

Registry citation. V.T09 (CMB acoustic-peak prediction; companion theorems V.T10–V.T14 cover ℓ_2 through higher-ℓ moments).

Falsification context. Reanalysis of Planck data that moves ℓ_1^{obs} outside any tightened τ-prediction's tolerance range falsifies the prediction at that tightening. v2.0 reports the current Lean value verbatim; if the underlying physics derivation tightens, the Lean module's ℓ_1^τ value updates and so does this case-study citation. (See Section 10 on how the paper re-pins to the canonical TauLib commit at release time.)

6.4 Book VII Commitments (the pattern that retired `sorry`)

Book VII's three load-bearing structural commitments — `omega_point_theorem`, `science_faith_boundary`, `no_forced_stance` — were until early 2026 encoded as `theorem X : True := sorry`. That pattern was performatively empty: the theorem statement was true (vacuously), the proof was absent (`sorry`), and CI's sorry-counter recorded the debt.

The Phase 0.5 closure (Wave R8b–R8j, completed 2026-05-01) replaced that pattern with the `Commitment` record:

```
structure Commitment where
  statement   : String
  warrant     : String
  registry_id : String

def omega_point_theorem : Commitment := {
  statement   := "...",
  warrant     := "...",
  registry_id := "VII.T..." }
-- and similarly for science_faith_boundary, no_forced_stance.
```

The new pattern carries inspectable typed data: a one-sentence statement, the warrant explaining the structural commitment's authority, and the registry ID linking back to the prose monograph. What it does *not* carry is a `sorry`. The CI assertion `--expected-sorry 0` now passes across all seven books.

The Commitment pattern is itself a contribution: it gives methodological commitments a typed, inspectable form that prior formalisation libraries handled either by external prose ("acknowledged, not formalised") or by performative `sorry`. Neither alternative was satisfactory; the typed record is.

Registry citations. VII.T-Omega-Point, VII.T-Science-Faith-Boundary, VII.T-No-Forced-Stance (declarations: `TauLib.BookVII.Logos.Sector` lines 457 and 511; `TauLib.BookVII.Final.Boundary` line 216).

Falsification protocol. Inspect the `Commitment` records: any reader can compare the recorded `statement` against the prose form in the Book VII monograph, or against the registry ID's published context, and decide whether the structural commitment holds. Inspection is the warrant; the Lean module makes inspection mechanical.

7. THE THREE CONJECTURAL AXIOMS

The TauLib repository contains exactly three `axiom` declarations across all seven books. All three live in Book III; all three are paired with finite-decidable witness functions; all three follow the *compute-then-axiomatize* pattern (Section 5). CI asserts `--expected-axioms 3` on every push; introducing a fourth would fail the build.

Axiom 1 — `bridge_functor_exists`. Located in `TauLib.BookIII.Bridge.BridgeAxiom` at line 134. The paired finite-witness function `bridge_functor_check bound db` (lines 79–98) is a decidable predicate validated via `native_decide` for all $(bound, db)$ with $bound \leq 15$, $db \leq 3$. The check passes; the axiom asserts universal extension. Closure path: extend the rank $(15, 3)$ window upward by mechanical induction on bound, then by combinatorial argument on db; preliminary work in `BookIII.Bridge.BridgeStability` (forthcoming).

Axiom 2 — `spectral_correspondence_03`. Located in `TauLib.BookIII.Doors.SpectralCorrespondence` at line 126. Asserts congruence of ζ-zeros to Hilbert–Pólya spectral eigenvalues at each primorial level k, paired with the finite check `spectral_correspondence_finite k`. The axiom is the deeper of the three; closure path follows the analytic-number-theory work in Book III chapters 12–14, formalisation outside current roadmap.

Axiom 3 — `grand_grh_adelic`. Located in `TauLib.BookIII.Doors.GrandGRH` at line 125. Adelic formulation of Generalised Riemann Hypothesis through prime-polarity (B/C/X sectors) partition at primorial level k, paired with the finite check `grand_grh_finite_check k`. This is the deepest of the three axioms; its closure is open in mathematical practice.

The compute-then-axiomatize pattern is the load-bearing piece. The finite check is not a heuristic that sometimes passes; it is a decidable function that the Lean kernel evaluates exhaustively over the cited range. Reading the axiom is reading the universal claim; reading the finite-witness function is reading the empirical ground. Each axiom reports both. `TauLib.Meta.PrintAxioms` emits a per-theorem JSON listing exactly which of the three axioms (plus K0–K6) any given τ-theorem depends on, so any reader can audit the trail.

8. THE THREE BOOK VII COMMITMENTS

Section 6.4 introduced the `Commitment` record pattern as a case study. This section is the exhaustive listing.

The structure (`TauLib.BookVII.Meta.Commitment` at line 110):

```
structure Commitment where
  statement   : String
  warrant     : String
  registry_id : String
```

The three `Commitment` instances landed by the Phase 0.5 closure:

1. `omega_point_theorem` — `TauLib.BookVII.Logos.Sector` at line 457
2. `science_faith_boundary` — `TauLib.BookVII.Logos.Sector` at line 511
3. `no_forced_stance` — `TauLib.BookVII.Final.Boundary` at line 216

A fourth, near-future migration: the Book IV `theta_qcd_zero_from_sa_i` declaration in `TauLib.BookIV.Particles.StrongCP`, currently encoded as `theorem ... : True := trivial`, is the natural next candidate for the `Commitment` pattern. The full numerical chain (strong-anomaly cancellation → explicit $\theta_{\mathrm{QCD}} = 0$) is documented in the Book IV monograph under registry citation IV.T160; the placeholder `: True := trivial` type signature holds the registry slot until either the full Lean theorem lands or the declaration is migrated to a `Commitment` record carrying its statement, warrant, and registry pointer. Either path is acceptable; the current `: True := trivial` encoding is the one residual instance of the pattern this section retired and is on the v2.0 follow-up roadmap.

Each instance carries: a *statement* (what the commitment asserts in one sentence), a *warrant* (the structural reason the commitment is the right thing to commit to within Category τ), and a *registry_id* pointing to the prose form in the Book VII monograph. Inspection of the triple is the warrant; the Lean module makes inspection mechanical.

The Commitment pattern is one of v2.0's distribution-relevant deliverables: it gives methodological commitments a typed, inspectable form. The pattern is fully general — any future Book that needs to record a structural commitment can use it without building new infrastructure.

9. LIMITATIONS & FUTURE WORK

This section is the honest accounting. v2.0 makes several claims; this section catalogues what those claims do *not* cover and where the known gaps and open paths sit.

The three Book III axioms are not yet retired. Each of `bridge_functor_exists`, `spectral_correspondence_03`, and `grand_grh_adelic` carries a closure path (Section 7); none has yet been retired. The first (`bridge_functor_exists`) is the most tractable — combinatorial in nature, with a clear extension argument that should land within 2026. The other two (spectral correspondence and adelic GRH) sit in active analytic-number-theory open territory; their closure is scope for the long view, not the current roadmap.

The Mathlib-community-preferred encoding for unproven universal claims is hypothesis-on-theorems. A second, complementary direction we plan to take before the v2.1 release: refactor downstream theorems to take `(h : BridgeFunctorExists)`, `(h : SpectralCorrespondence03)`, and `(h : GrandGRHAdelic)` as explicit hypotheses, retiring the three global `axiom` declarations entirely. The `BridgeAxiom.lean` docstring already concedes this is the right encoding ("this is the encoding the Mathlib community would prefer"); it is not landed in v2.0 because the refactor is substantial and we wanted to ship the architecture and foundations documentation first. The current global-axiom encoding, the finite-witness pairing, and the per-theorem `TauLib.Meta.PrintAxioms` JSON make the trust extension auditable in v2.0 in a way the hypothesis-encoding will tighten further in v2.1.

The structural derivation theorem for ι_τ is in flight, not landed. Section 4.2 was explicit on this: the Lean module `TauLib.BookI.Boundary.Iota` ships fiat constants today. The `ROADMAP-3-HINGES.md` document carries the multi-phase plan. Wave R8/R9 prerequisite infrastructure (`TauReal.exp`, `TauReal.log`, `TauReal.sqrt`) is landing in commits dated 2026-04-30 and 2026-05-01. The full ι_τ structural-derivation chain will land across multiple waves over the next 2–6 months.

Book VI is partial. The current per-book inventory (Appendix A) reports Book VI at 30 Lean files, 147 theorems, 164 definitions. The Book VI prose registry contains 168 dashboard objects; 30 of those have Lean witness modules, leaving the bulk of Book VI as scope for future formalisation work. Book VI's content (life predicate, origin-of-life surfaces, neural architecture, genetic code) is more empirically anchored than Books I–V; the formalisation priorities there are different.

Mathlib bridges are not yet exhaustive. The 13 bridge modules cover ring/field/order/topology infrastructure that Books I and II need. Mathlib's larger algebraic, analytic, and measure-theoretic typeclasses (`NormedField`, `MetricSpace`, `MeasurableSpace`, etc.) are not yet bridged on the τ-side. The relevant kernel constructions exist or are roadmapped; the bridge instances are tractable workstreams.

The Hom layer / categorical morphism composition. v1.0 §3.1 flagged that the categorical-instance for Category τ itself — in the sense of an explicit `Mathlib.CategoryTheory.Category` instance — is not yet formalised. Hinge 5 (holomorphy-first) and Hinge 6 (tau-topos) have built the relevant earned-categorical machine; the remaining work is the bridge instance.

Foundational metatheory. TauLib does not yet formally prove the foundational metatheory of Category τ: relative consistency (K0–K6 + axiom-budget vs. ZFC), independence (K0–K6 from each other), and categoricity (any model of K0–K6 is isomorphic to the intended one) remain open. A companion paper is the natural home for this work.

Wave 52–54 (TauProfinite as a full profinite space). Section 6.1 cited Wave 51's Hausdorff and totally-disconnected instances. The third defining axiom of a profinite space, compactness, is the Wave 52–54 follow-up (uniform-space + compact-space + full Profinite categorical wrap). Compactness on `TauProfinite` requires a Tychonoff-style argument that is currently the largest single piece of open work in Book I.

10. REPRODUCIBILITY

Every count and theorem cited in this paper is reproducible by checking out the pinned TauLib commit and running the standard build. **The v2.0 release pins TauLib at commit `72aa2b6c7d6be28511f9649d3120773179b19038`** (`feat(BookV/Cosmology): Wave R9-2B V.T-LRD-1B HSB f_iota_TauReal numerical witness`, 2026-05-01). Reproduction:

```
git clone https://github.com/Panta-Rhei-Research/taulib
cd taulib
git checkout 72aa2b6c7d6be28511f9649d3120773179b19038
lake build
python3 scripts/check_no_sorry.py \
    --root TauLib --expected-axioms 3 --expected-sorry 0
```

The expected outcome at the pinned commit:

- `lake build` exits 0 with all 524 modules compiled
- `check_no_sorry.py` reports 3 axioms / 0 sorry
- the per-book counts in Appendix A match this paper's Section 1 aggregates exactly
- `papers/whitepapers/taulib/scripts/verify_claims_map.py` (this paper's claims-map auditor) reports **39/39 checks passed** — every cited file path, declaration name, and aggregate count is verified against the pinned commit. The audit report is committed alongside the paper as `main_claims_audit.md`.

The CI workflow at `.github/workflows/lean-build.yml` runs the same checks on every push; the public release manifest at `panta-rhei.site/verify/release-manifest/` is updated in lockstep with merges to `main` and serves as the canonical external attestation of TauLib's current state.

Toolchain versions. TauLib pins the Lean toolchain version in `lean-toolchain`; check that file for the exact version and Mathlib commit at the pinned TauLib commit. v2.0's full reproduction sequence requires LuaLaTeX (TeX Live 2024 or later) for this paper plus Lean 4 toolchain with Mathlib for TauLib itself.

Long-term archival. The v2.0 PDF, the `main_claims_audit.md` report, and a snapshot of the pinned TauLib source are deposited together as a Zenodo record under DOI `10.5281/zenodo.19976503` (minted 2026-05-01). The DOI is stamped on the title page, in the PDF metadata, and in the "How to cite" callout at the close of this section so that any fixed-citation discovery path resolves to a single immutable archival record. Subsequent v2.x point releases deposit under fresh DOIs; the canonical Zenodo concept-DOI for the TauLib whitepaper series resolves to the latest version.

Distribution-time pinning. Section 6.3 noted that the CMB ℓ_1 value is read verbatim from the current Lean module. Wave W2.8 (the v2.0 release preparation) pins TauLib to a specific commit, runs the `verify_claims_map.py` audit script, and rebuilds the paper against that pinned source-of-truth. Any drift between this paper's prose-level claims and the pinned Lean source is a release-blocker; the audit report gets published alongside the paper PDF.

HOW TO CITE

Fuchs, T. and Fuchs, A.-S. *TauLib: A Self-Contained Lean 4 Library for Category τ — Kernel + Mathlib Bridges + Registry-Driven Correspondence.* Panta Rhei Research, May 2026 (v2.0). Companion repository: `https://github.com/Panta-Rhei-Research/taulib`.

DOI: `10.5281/zenodo.19976503`

11. RELATED WORK

The closest formalisation projects to TauLib by ambition are the large research-grade Lean/Mathlib developments cited in Section 2: the Liquid Tensor Experiment, the Polynomial Freiman–Ruzsa formalisation, the Sphere Eversion project, the

Carleson formalisation, and the Prime Number Theorem and Beyond project. All five are Mathlib clients — they mechanise mathematical content already in mathematical practice using Mathlib's typeclass hierarchy, classical axioms, and universe polymorphism as their substrate. Their value is precisely in stress-testing Mathlib at research scale, and they have done so successfully.

TauLib differs in its substrate, not in its method. Where those projects accept Mathlib's foundational decisions (classical mode, typeclass inheritance, open axiom pool) as a working environment and build research-grade content on top, TauLib treats those decisions as *themselves* contestable and builds a kernel from a different substrate (Section 5). This is not a competing approach — TauLib is happy to interoperate with Mathlib at the bridge layer (Section 3.2) — but it is a different research question. The mainstream projects ask: *can this body of mathematics be formalised in Mathlib?* TauLib asks: *can a categorical framework with seven structural axioms and a single primitive operator support a mature library?* The two questions are complementary and both are interesting; this paper is a status report on the second.

Outside Lean, the foundational alternatives are well-known — Coq/Rocq [1], Isabelle/HOL [21], Agda [22], Metamath [20], and (more foundation-focused) the HoTT [27] and UniMath [30] projects. We do not attempt a comprehensive comparison here. The closest spiritual relative is probably Univalent Foundations: like UniMath, TauLib treats its foundation as the load-bearing object of inquiry, not as a working environment. Where UniMath investigates the consequences of the univalence axiom, TauLib investigates the consequences of K0–K6 on a generator-based kernel.

A. PER-BOOK MODULE INVENTORY

Counts at the pinned TauLib commit. Methodology: `find` for file counts; line-anchored `grep` for declaration counts (the narrower count, excluding `private`/`noncomputable` modifiers — these add ∼3% more declarations across the library; the script-aware sorry count via `scripts/check_no_sorry.py` strips comments / docstrings / strings before counting).

Book	Files	Theorem	Def	Instance	Axiom	Sorry
I	157	1,580	720	78	0	0
II	65	679	653	0	0	0
III	70	644	537	0	3	0
IV	89	890	957	5	0	0
V	82	835	625	11	0	0
VI	30	147	164	0	0	0
VII	8	117	106	0	0	0
Tour, Meta, etc.	23	—	—	—	0	0
Total	**524**	**4,892**	**3,762**	**94**	**3**	**0**

The aggregate row is the canonical headline cited throughout the paper. The 78 instances under Book I are the Mathlib-typeclass bridges (Section 3.2 catalogue); the remaining 16 instances (94 − 78) are kernel-side `Decidable`/`Inhabited`/`Repr` instances that introduce no Mathlib mathematical content. The 3 axioms are all in Book III (Section 7).

B. REPOSITORY STRUCTURE, BUILD, CI

Top-level layout

- `TauLib/` — Lean source tree, organised by book (`BookI/`, ..., `BookVII/`, plus `Tour/`, `Meta/`)
- `lakefile.lean` + `lean-toolchain` — Lake build config and pinned Lean toolchain
- `lake-manifest.json` — pinned Mathlib commit
- `registry/book{1..7}_registry.jsonl` — registry entries (4,805 total)
- `scripts/` — `check_no_sorry.py`, `print_axioms_report.py`, `generate_stats.py`, others
- `.github/workflows/lean-build.yml` — CI workflow

Build

```
git checkout 72aa2b6c7d6be28511f9649d3120773179b19038
lake build
python3 scripts/check_no_sorry.py \
    --root TauLib --expected-axioms 3 --expected-sorry 0
```

CI gates (merge-blockers)

1. Forbidden-imports grep at the kernel/bridges seam (Section 3.3)
2. `--expected-axioms 3 --expected-sorry 0`
3. `TauLib.Meta.PrintAxioms` per-theorem axiom-trail JSON (uploaded as `print_axioms.json` CI artefact)

Tour system

`TauLib/Tour/VerifyItYourself.lean` is the 15-minute sceptic's start: five surprising claims with `#eval` + `decide` witnesses that run in the Lean kernel. New contributors enter via the Tour; `CONTRIBUTING.md` documents the registry-ID convention.

C. RED-TEAM REVIEW FINDINGS (SYNOPSIS)

This v2.0 carries a public companion document `whitepapers/taulib/RED_TEAM_FINDINGS.md` (committed alongside the paper at PR #13) that consolidates the findings of seven simulated Lean 4 expert reviewers — Leonardo de Moura, Sebastian Ullrich, Mario Carneiro, Patrick Massot, Jeremy Avigad, Kevin Buzzard, Floris van Doorn — dispatched in parallel against the v2.0-rc1 (19-page) draft.

Aggregate verdict: 6 of 7 reviewers gave a qualified-yes on the trust / use / cite question; one (Buzzard) gave a qualified-no contingent on substantive findings about the Prime Polarity formalisation status, the hinge-papers framing, and the preferred encoding of conjectural axioms.

Findings classification: 9 must-fix-before-publication, 10 strongly-recommended, 14 nice-to-have.

Highest-convergence findings (flagged by ≥ 2 reviewers):

- §6 Strong CP `: True := trivial` placeholder inconsistent with §6.5 Commitment pattern (4 reviewers)
- §3.2 bridge file count + missing `TauRealRatQuotient.lean` (2 reviewers)
- §6.4 CMB ℓ_1 `+2863 ppm` framed as falsifiable when actually outside Planck uncertainty (2 reviewers)
- §3.B "exactly one Classical.byCases" overstatement (2 reviewers)
- §4.4 Prime Polarity Lean shown is unboundedness scaffolding, not the dichotomy theorem (2 reviewers)
- "Constructive-by-default" + "exactly 3 axioms" framing conflates user-declared with transitive Lean baseline (2 reviewers)

What v2.0 (this version) addresses: all 9 must-fix-before-publication findings and 8 of 10 strongly-recommended findings have been applied to the prose. The two strongly-recommended findings deferred — restructuring §5 into three architectural principles, and expanding the H4–H7 brief treatment in §4.5 into full subsections — are scoped for v2.1 because they require substantial new content rather than prose fixes. v2.0 carries explicit acknowledgement of these in Section 9.

The full reviewer reports, severity classification, and cross-reviewer convergence map are in `RED_TEAM_FINDINGS.md`.

REFERENCES

[1] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer, 2004.

[2] Johan Commelin and Adam Topaz. Abstraction boundaries and spec driven development in pure mathematics. *Notices of the American Mathematical Society*, 71(2):240–246, 2024.

[3] Harold Davenport. *Multiplicative Number Theory*, volume 74 of *Graduate Texts in Mathematics*. Springer, 3rd edition, 2000. Standard graduate reference for Dirichlet density on primes in arithmetic progressions; cited for the density-$1/2$ result on the B/C partition.

[4] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction – CADE 28*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer, 2021.

[5] Peter Gustav Lejeune Dirichlet. Beweis des satzes, dass jede unbegrenzte arithmetische progression .... *Abhandlungen der Königlichen Preußischen Akademie der Wissenschaften*, pages 45–81, 1837. The original proof of Dirichlet's theorem on primes in arithmetic progressions; underwrites the natural-density-$1/2$ claim for the B/C polarity partition of TauLib Hinge 2 via the residue classes $\pm 1 \pmod 8$ and $\pm 3 \pmod 8$.

[6] Thorsten Fuchs and Anna-Sophie Fuchs. Address resolution, not calculation: The genealogical DAG, cayley metric, and why arithmetic in category τ has no equations. Internal preprint, Panta Rhei Research; `papers/research-papers/address-resolution/`, 2026. Hinge 7 of the eight-paper bundle.

[7] Thorsten Fuchs and Anna-Sophie Fuchs. Hyperfactorization: Unique tower-atom decomposition and the ABCD chart. Internal preprint, Panta Rhei Research; `papers/research-papers/hyperfactorization/`, 2026. Hinge 1 of the eight-paper standalone-hinge bundle; awaiting external preprint server submission.

[8] Thorsten Fuchs and Anna-Sophie Fuchs. The master constant $\iota_\tau = 2/(\pi + e)$: Structural derivation via the crossing-point defect germ. Internal preprint, Panta Rhei Research; `papers/research-papers/iota-tau/`, 2026. Hinge 3 of the eight-paper bundle. Lean-side structural derivation in flight per `ROADMAP-3-HINGES.md`; current Lean module ships fiat constants.

[9] Thorsten Fuchs and Anna-Sophie Fuchs. The panta rhei foundational bundle: Research memo and reading guide for an eight-paper standalone-hinge peer-review package. Internal preprint, Panta Rhei Research; `papers/research-papers/bundle-memo/`, 2026. Framing/integration memo for the eight-paper hinge bundle; not itself a hinge.

[10] Thorsten Fuchs and Anna-Sophie Fuchs. Prime polarity: The B/C dichotomy and legendre density. Internal preprint, Panta Rhei Research; `papers/research-papers/prime-polarity/`, 2026. Hinge 2 of the eight-paper bundle; classical references in this paper: Gauss Disquisitiones 1801 (second supplementary law) and Dirichlet 1837 / Davenport 2000 (density 1/2).

[11] Thorsten Fuchs and Anna-Sophie Fuchs. The split-complex boundary algebra $\mathbb{H}[j] = \mathcal{R}_\partial[j]/(j^2 - 1)$: Canonical uniqueness and four-atom dictionary. Internal preprint, Panta Rhei Research; `papers/research-papers/boundary-algebra/`, 2026. Hinge 4 of the eight-paper bundle.

[12] Thorsten Fuchs and Anna-Sophie Fuchs. τ-holomorphy on the boundary algebra: ω-germ transformers, wave-equation cauchy–riemann, earned categorical machine. Internal preprint, Panta Rhei Research; `papers/research-papers/holomorphy-first/`, 2026. Hinge 5 of the eight-paper bundle.

[13] Thorsten Fuchs and Anna-Sophie Fuchs. The τ-kernel as foundational architecture: Ontic identity, linear discipline, and the star-autonomous path. Internal preprint, Panta Rhei Research; `papers/research-papers/kernel-foundation/`, 2026. Hinge 8 of the eight-paper bundle (the architectural capstone).

[14] Thorsten Fuchs and Anna-Sophie Fuchs. The τ-topos and its four-valued internal logic: Resolving paraconsistent semantic circularity via split-complex idempotents and ω-germ stabilization. Internal preprint, Panta Rhei Research; `papers/research-papers/tau-topos/`, 2026. Hinge 6 of the eight-paper bundle.

[15] Carl Friedrich Gauss. *Disquisitiones Arithmeticae*. Gerh. Fleischer, Leipzig, 1801. Original source for the second supplementary law of quadratic reciprocity, $(2/p) = (-1)^{(p^2-1)/8}$, which yields the $p \equiv \pm 1 \pmod 8$ vs. $p \equiv \pm 3 \pmod 8$ partition cited by the Prime Polarity Theorem (TauLib Hinge 2). English translation: Yale University Press, 1965.

[16] W. Timothy Gowers, Ben Green, Freddie Manners, and Terence Tao. On a conjecture of Marton. *arXiv preprint*, 2023.

[17] Kenneth Ireland and Michael Rosen. *A Classical Introduction to Modern Number Theory*, volume 84 of *Graduate Texts in Mathematics*. Springer, 2nd edition, 1990. Classical reference for quadratic reciprocity, the supplementary laws, and elementary analytic number theory; standard graduate textbook.

[18] Alex Kontorovich, Terence Tao, and the PrimeNumberTheorem+ contributors. PrimeNumberTheorem+: A formal proof of the prime number theorem in Lean 4. `https://github.com/AlexKontorovich/PrimeNumberTheoremAnd`, 2024.

[19] Lean Prover Community. The Lean 4 trusted computing base: `Lean.ofReduceBool` and `Lean.trustCompiler`. Lean 4 source repository, `src/Init/Core.lean` and related; `https://github.com/leanprover/lean4`, 2024. The extension axioms that appear in `#print axioms` output for theorems closed by `native_decide`.

[20] Norman D. Megill and David A. Wheeler. *Metamath: A Computer Language for Mathematical Proofs*. Lulu Press, second edition, 2019.

[21] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.

[22] Ulf Norell. *Towards a Practical Programming Language Based on Dependent Type Theory*. PhD thesis, Chalmers University of Technology and Göteborg University, 2007.

[23] Planck Collaboration. Planck 2018 results. VI. cosmological parameters. *Astronomy & Astrophysics*, 641:A6, 2020. Reference cosmology paper for the CMB acoustic-peak measurements, including $\ell_1 = 220.0 \pm 0.5$, against which the TauLib Book V Hinge prediction of $\ell_1^{\tau} = 220.63$ is compared in §6.4. Fixes the data side of the falsification protocol.

[24] Peter Scholze. Liquid tensor experiment. Xena Project blog, 2020.

[25] Terence Tao, Timothy Gowers, Ben Green, Freddie Manners, Floris van Doorn, and the PFR contributors. A formalization of the Polynomial Freiman-Ruzsa conjecture in Lean 4. `https://github.com/teorth/pfr`, 2023.

[26] The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020)*, pages 367–381. ACM, 2020.

[27] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, Princeton, 2013.

[28] Floris van Doorn, Lars Becker, and the Carleson contributors. A formal proof of Carleson's theorem in Lean 4. `https://github.com/fpvandoorn/carleson`, 2024.

[29] Floris van Doorn, Patrick Massot, and Oliver Nash. Formalising the h-principle and sphere eversion. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2023)*, pages 121–134, 2023.

[30] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson, et al. UniMath: A computer-checked library of univalent mathematics. `https://github.com/UniMath/UniMath`, 2014.