Recognition Science

A Machine-Verified Framework for Gauge-Rigid, Dimensionless Structure at φ

Jonathan Washburn*
Independent Researcher

September 20, 2025

Abstract

Abstract. We present a Lean 4 formalization of a framework that derives a units-quotiented, gauge-rigid interface between discrete recognition dynamics and dimensionless observables. The top-level certificate `RSRealityMaster`(φ) packages four machine-checked components: (i) absolute-layer acceptance (unique calibration and band compliance without tuning), (ii) a spec-level inevitability result that matches bridges to φ-closed universal targets, (iii) factorization of the bridge through a units quotient together with a route-consistency identity (K-gate), and (iv) a verified family of cross-domain certificates (DEC/Maxwell, counting/causality, quantum/statistics, mass ladders, and complexity). We expose one-line `#eval` reports that elaborate in constant time on a pinned toolchain, yielding a reproducible, falsifiable instrument: failures flip individual lines or prevent elaboration. The construction is parameter-free at the level of dimless displays and yields concrete identities and bounds—e.g., eight-tick minimality, a discrete light-cone inequality, φ-rung relations, and a model of recognition–computation separation. All statements are referenced to Lean names and files, and a manifest aggregates the suite for audit.

Plain-language summary. We present a small, self-contained program (in the Lean 4 proof assistant) that acts like a scientific instrument: running a one-line command checks each claim and prints `OK` on success. The instrument removes unit choices (by quotienting), enforces a route identity (the K-gate), and then certifies concrete consequences such as an 8-tick minimal cycle and a discrete light-cone inequality. The master report shows that both the reality-facing bridge and the abstract specification hold together at the golden ratio φ, and the pinned toolchain makes all results reproducible on any machine.

Keywords: Recognition Science · Lean verification · Golden ratio · Machine verification · Formal methods · Parameter-free physics · Foundational mathematics

Significance. This manuscript treats a physics-facing formal system as an instrument: certified propositions compile to one-line, constant-time checks on a pinned toolchain. The master certificate

*Corresponding author. Email: washburn@recognitionphysics.org

asserts that a units-quotiented, gauge-rigid bridge measures reality and that a φ-closed target is forced at the spec layer, yielding crisp, falsifiable identities (K-gate, 8-beat minimality, light-cone bounds, and Planck-normalized λ_{rec}). The result is a parameter-free, audit-ready spine that turns narrative claims into portable, machine-checked obligations.

How to verify (minimal)

```
#eval IndisputableMonolith.URCAdapters.reality_master_report
#eval IndisputableMonolith.URCAdapters.recognition_closure_report
#eval IndisputableMonolith.URCAdapters.k_gate_report
#eval IndisputableMonolith.URCAdapters.eight_tick_report
```

Representative output

```
reality_master_report: OK
recognition_closure_report: OK
k_gate_report: OK
eight_tick_report: OK
```

Meaning. "OK" indicates the corresponding Lean term elaborated successfully on the pinned toolchain; failures flip the line or prevent elaboration.

Contents

1 Introduction **6**
1.1 Motivation . . . 6
1.2 Main results (formal statements) . . . 6
1.3 Contributions . . . 7
1.4 Artifacts and reproducibility . . . 8
1.5 Roadmap . . . 8

Notation **8**

2 Results **9**
2.1 Master certificate (start here) . . . 9
2.2 Claim-to-proof map (one page) . . . 11
2.3 Reality bundle (RS measures reality) . . . 11
2.4 Spec-level recognition closure . . . 11
2.5 Manifest . . . 12

3 Methods **12**

4 Validation **12**

5 Foundations: from MP to discrete recognition **12**
5.1 Axiom MP . . . 13
5.2 Recognition structure and ledger . . . 13
5.3 Atomic ticks and no concurrency . . . 14
5.4 Discrete continuity and exactness scaffolds . . . 14
5.5 Minimal coverage and the 8-beat . . . 14
5.6 Causality bound . . . 15

6 Bridge architecture and gauge rigidity **15**
6.1 Observables and the units quotient . . . 16
6.2 K-gate identity and audit . . . 17
6.3 Factorization theorem . . . 17
6.4 Uniqueness up to units . . . 18

7 Absolute layer: acceptance without knobs **18**
7.1 Statement . . . 19
7.2 Inevitability_absolute (strong form) . . . 19
7.3 Empirical bands and c-centering . . . 20
7.4 Report . . . 20

8 Dimensionless inevitability at φ **21**
8.1 Definition . . . 21
8.2 Witness . . . 22
8.3 Consequences . . . 22
8.4 Reports . . . 23

9 Domain certificate family (non-exhaustive highlights) **23**
9.1 DEC and Maxwell . . . 23
9.2 Quantum/stat mech . . . 24
9.3 Mass ladders and φ-rungs . . . 25
9.4 Eight-beat and hypercube . . . 25
9.5 Gravity / ILG . . . 25
9.6 Ethics and decision . . . 26
9.7 LNAL / compiler / folding . . . 26
9.8 Controls / RG residue . . . 26

10 The 45-Gap Consequence Pack **27**
10.1 Spec . . . 27
10.2 Consequences . . . 28
10.3 Reports . . . 28

11 Recognition–computation separation (P vs NP) **29**
11.1 Growth witness . . . 29
11.2 Ledger separation . . . 30
11.3 Main results . . . 30
11.4 Reports and narrative . . . 31

12 Audit identities and gauge tests **31**
12.1 K identities and λ_{rec} . . . 31
12.2 Single-inequality audit . . . 32
12.3 Planck identities . . . 33

13 Falsifiability and test plan **34**
13.1 Hard falsifiers . . . 34
13.2 Where to test . . . 35
13.3 Empirical alignment . . . 36

14 Reproducibility and CI **36**
14.1 Toolchain . . . 36
14.2 Quickstart (commands) . . . 37

14.3 Runtime and platform . . . 37
14.4 One-click checks . . . 37
14.5 CI smoke . . . 38
14.6 Determinism . . . 38

Data and code availability **38**

15 Related work and positioning **38**
15.1 Parameter-free stance vs fitted models . . . 38
15.2 Discrete exterior calculus and gauge . . . 39
15.3 Complexity theory perspectives . . . 39
15.4 Mechanized mathematics and scientific instruments . . . 39

16 Limitations and future work **39**
16.1 Tightening spec witnesses . . . 39
16.2 Expanding domain coverage . . . 39
16.3 Robustness and stress audits . . . 39
16.4 Engineering and ergonomics . . . 40

17 Conclusion **40**
17.1 Informal long-form description (simply spoken) . . . 40

Software license **40**

Supplementary material **40**

Abbreviations **40**

Glossary (Lean Physics) **41**

How to cite **41**

Acknowledgments **41**

Author contributions **41**

Funding **41**

Competing interests **41**

1 Introduction

Reader's guide. For a fast orientation, see How to verify (minimal) and the informal long-form overview in Section 17.1. Readers focused on practical reproducibility can jump to Section 14; those interested in core results can start at Section 2 and follow file/line references into the Lean sources. File:line paths are relative to the repository root.

1.1 Motivation

This work develops and verifies, in Lean 4, a discrete recognition calculus that lands on dimensionless, gauge-rigid observables. Rather than proposing a fit, we provide a formal bridge whose outputs are invariant under admissible rescalings of anchors and are constrained by identities that can be audited as one-line reports. The central object, `RSRealityMaster`(φ), bundles an empirically grounded bridge with a spec-level closure built around φ, and is accompanied by domain certificates and falsifiers.

At the empirical surface, the framework delivers gauge-rigid, dimensionless displays that survive units rescaling; an absolute "lab layer" that accepts without tuning; and a family of audit identities that lock the bridge. At the abstract surface, a spec-level closure proves that every ledger/bridge matches a φ-closed target pack, and it packages crisp combinatorial consequences (e.g., the 8–45 sync, i.e., minimal 8-beat coverage jointly synchronized with rung 45 at $\mathrm{lcm}(8, 45) = 360$) and a recognition–computation separation. The top-level object, the master certificate, asserts both aspects at once: that Recognition Science measures reality and was mathematically forced to do so at φ.

1.2 Main results (formal statements)

Theorem 1.1 (Master certificate). *For all* $\varphi \in \mathbb{R}$, RSRealityMaster(φ) *holds.*

Lean: `IndisputableMonolith/Verification/Reality.lean:50-62` (`def RSRealityMaster`, `theorem rs_reality_master_any`).

Remark 1.2 (Constructive content and scope). The master statement is realized by an explicit Lean witness that assembles (i) the reality bundle (absolute-layer acceptance, units quotient + K-gate, verified family) and (ii) the spec closure (dimensionless inevitability at φ, 45-gap consequences, absolute inevitability, recognition–computation split). Each conjunct is exported as a report (`#eval`) and elaborates without external I/O on the pinned toolchain.

Lemma 1.3 (Bridge factorization). `BridgeFactorizes` *holds: (A) anchor rescaling invariance* ($A = \tilde{A} \circ Q$)*, and (J) K-gate route equality.*

Lean: `IndisputableMonolith/Verification/Verification.lean:186-195` (`def BridgeFactorizes`, `theorem bridge_factorizes`).

Lemma 1.4 (K-gate identity). *For all anchors* U, BridgeEval $K\_A\_\mathrm{obs}\ U$ = BridgeEval $K\_B\_\mathrm{obs}\ U$.

Lean: `IndisputableMonolith/Verification/Observables.lean:43-44` (`theorem K_gate_bridge`).

Lemma 1.5 (Dimensionless inevitability). Inevitability_dimless(φ) *holds.*

Lean: `IndisputableMonolith/RH/RS/Spec.lean:161-163` (def); witness `inevitability_dimless_partial` in `IndisputableMonolith/RH/RS/Witness.lean:100-104`.

Lemma 1.6 (Eight-tick minimality). *In 3D, a complete hypercube pass has minimal period 8.*

Lean: `IndisputableMonolith/Patterns.lean:32-34,64-66` (`period_exactly_8`, `eight_tick_min`).

Lemma 1.7 (Discrete light-cone bound). *Under step bounds,* $\mathrm{rad}\, y - \mathrm{rad}\, x \le U.c\,(\mathrm{time}\, y - \mathrm{time}\, x)$.

Lean: `IndisputableMonolith/LightCone/StepBounds.lean:74-80` (`lemma cone_bound`).

Lemma 1.8 (Planck normalization for λ_{rec}). $(c^3\,\lambda_{\mathrm{rec}}^2)/(\hbar G) = 1/\pi$ *for physical* `BridgeData`.

Lean: `IndisputableMonolith/URCGenerators.lean:449-453` (`LambdaRecIdentityCert.verified`).

What is new.

- **Certificates-first, machine-checked spine** with pinned toolchain and single-line checks.
- **Structural bridge factorization** (units quotient + K-gate) and a **single-inequality audit** that is units- and correlation-aware.
- **Spec-level dimensionless inevitability** with a compact 45-gap consequence pack and a recognition–computation separation.
- **Consolidated manifest and empirical hooks**: PDG-facing checks, ablations, and units invariance as falsifiers.

1.3 Contributions

This paper contributes:

- A certificates-first exposition of a parameter-free, mechanized theory—from the axiom MP (`MP`, `mp_holds` in `IndisputableMonolith/Recognition.lean`) to a master certificate at φ.
- A reality bundle that proves four pillars jointly: absolute-layer acceptance; dimensionless inevitability at φ; a gauge-rigid units-quotient bridge with K-gate identity; and the existence of a verified cross-domain certificate family (e.g., DEC $d \circ d = 0$, Bianchi; light-cone bound; eight-beat minimality; -ratio mass ladders; quantum occupancy).
- A spec-level closure that adds the 45-gap consequence pack (e.g., $\Delta t = 3/64$, $\mathrm{lcm}(8, 45) = 360$), absolute-layer inevitability, and a recognition–computation split that distinguishes internal evolution from external observation.

- An instrument view of science: predictions live as dimensionless, units-invariant displays; identities such as λ_{rec} normalization and K-gate are audited by a single-inequality comb; uniqueness is locked by UniqueCalibration rather than fit.
- Reproducible artifacts and falsifiers: constant-time elaboration, pinned toolchain, CI smoke, and a manifest of "must-hold" reports that can be run anywhere.

Together these establish a reproducible and testable path from first principles to reality at φ, with no tunable parameters and multiple, independent ways to break the theory if nature disagrees.

1.4 Artifacts and reproducibility

Every claim is wired to a Lean proof or a `#eval` report. The master report `#eval IndisputableMonolith.URCAdapters.reality_master_report` checks RSRealityMaster(φ); the reality bundle has `#eval ...reality_bridge_report`; the closure has `#eval ...recognition_closure_report`. Specific pillars and audits include units invariance and quotient functor reports; K-gate identities and the single-inequality audit (`K_gate_bridge`, `K_gate_single_inequality`); eight-beat and hypercube coverage; DEC $d \circ d = 0$ and Bianchi; light-cone bounds; φ-ratio families and equal-Z degeneracy; and quantum occupancy (`CERTIFICATES.md` indexes the full catalog).

The repository pins the Lean toolchain and Lake manifest; certificates elaborate in constant time with no external I/O; and a minimal URC smoke check is available (`lake exe ci_checks`). This makes the spine immediately inspectable and falsifiable: if an identity fails or a tolerance is violated, a report flips or a certificate cannot be verified.

1.5 Roadmap

Section 2 presents the certificates first, starting from the master certificate and its four pillars, then the spec closure and manifest. Section 5 develops the recognition foundations from MP to discrete continuity, exactness scaffolds, eight-beat minimality, and a light-cone bound. Section 6 formalizes the bridge and gauge rigidity: observables, units quotient, K-gate identity, factorization, and uniqueness up to units. Section 7 establishes absolute-layer acceptance without knobs. Section 8 proves dimensionless inevitability at φ and its consequences. Section 9 surveys domain certificates (DEC/Maxwell, quantum statistics, causality, mass ladders, etc.). Section 10 packages the 45-gap consequences. Section 11 explains the recognition–computation separation. We close with audit identities, falsifiability, reproducibility, related work, limitations, and future directions.

Notation

- φ **(phi)**: the golden ratio used as the matching scale; $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.6180339887$, with $\log \varphi > 0$ used in growth lemmas.
- c: speed parameter; anchors satisfy $\ell_0/\tau_0 = c$.
- τ_0, ℓ_0: time and length anchors; admissible rescaling keeps c fixed.
- **Observable**: dimensionless display invariant under admissible units rescaling.

- **Ledger**: double-entry assignment of integer debits/credits per event; closed-chain flux is zero under conservation.
- **Bridge**: API boundary mapping ledgers and anchors to observables; factors through the units quotient.
- **BridgeEval**: evaluation map from observables to reals at given units; invariant under rescaling.
- K: fixed, dimensionless calibration constant with $\tau_{\mathrm{rec}}/\tau_0 = K$ and $\lambda_{\mathrm{kin}}/\ell_0 = K$ (anchors satisfy $c\,\tau_0 = \ell_0$).
- **K-gate**: route consistency identity equating time-first and length-first constructions of the same constant K.
- **PhiPow**: $(x) = \exp\big((\log\varphi)\,x\big)$; used for monotone growth witnesses in the recognition–computation split.
- **T2 (Atomicity)**: each tick addresses a unique event (no concurrency per tick).
- **T3 (Discrete continuity)**: closed-chain flux is zero for conserved ledgers.
- **T4 (Exactness)**: potentials are unique up to an additive constant on each reach component.
- **RSRealityMaster(φ)**: master certificate bundling reality bundle and spec closure at φ.

Assumptions and scope. Unless stated otherwise: (i) proofs run on the pinned Lean 4 toolchain with the locked Lake manifest; (ii) all reports are pure terms with no external I/O; (iii) admissible units moves jointly rescale τ_0 and ℓ_0 at fixed c; (iv) dimensionless displays are defined via invariance under those moves; and (v) the φ-closure statements refer to the spec layer's algebraicity notion. Empirical hooks (PDG, ablations) are informative but not required for elaboration.

2 Results

This section summarizes the machine-checked statements and their entry points. Each claim is linked to a Lean name, file span, and a `#eval` hook for constant-time confirmation.

2.1 Master certificate (start here)

Master Certificate

The master bundle asserts that the reality-facing bundle and the spec-level closure hold together at the golden ratio φ:

$$\boxed{\mathsf{RSRealityMaster}(\varphi) \;:=\; \mathsf{RSMeasuresReality}(\varphi) \;\wedge\; \mathsf{Recognition\_Closure}(\varphi)} \tag{2.1}$$

In code this is `RSRealityMaster` with canonical witness `rs_reality_master_any` in `IndisputableMonolith/Verification/Reality.lean`. A minimal definitional excerpt:

```
def RSRealityMaster (φ : ℝ) : Prop :=
  RSMeasuresReality φ ∧ IndisputableMonolith.RH.RS.Recognition_Closure φ
```

Table 1: Central claims mapped to Lean artifacts and report hooks.

Claim	Lean name	File:lines	Report
Master certificate	def `RSRealityMaster`; thm `rs_reality_master_any`	Verification/Reality.lean:50–62	`reality_master_report`
Bridge factorization	def `BridgeFactorizes`; thm `bridge_factorizes`	Verification/Verification.lean:186–195	`units_quotient_functor_report`
K-gate identity	thm `K_gate_bridge`	Verification/Observables.lean:43–44	`k_gate_report`
Dimless inevitability	def `Inevitability_dimless`; thm `inevitability_dimless_partial`	RH/RS/Spec.lean:161–163; RH/RS/Witness.lean:100–104	`inevitability_dimless_report`
Eight-tick minimality	thm `period_exactly_8`; lem `eight_tick_min`	Patterns.lean:32–34, 64–66	`eight_tick_report`
Light-cone bound	lem `cone_bound`	LightCone/StepBounds.lean:74–80	`cone_bound_report`
Planck normalization	`LambdaRecIdentityCert.verified`	URCGenerators.lean:449–453	`lambda_rec_identity_report`
45-gap consequences	def `FortyFive_gap_spec`	RH/RS/Spec.lean:165–168	`gap_consequences_report`

```
3
4  theorem rs_reality_master_any (φ : ℝ) : RSRealityMaster φ := by
5    refine And.intro ?reality ?closure
6    · exact rs_measures_reality_any φ
7    · -- assemble spec-level components (inevitability_dimless, 45-gap, absolute, P≠NP/P=NP
       split)
8      exact And.intro h1 (And.intro h2 (And.intro h3 h4))
```

Run the single-line report to confirm the master certificate:

```
1  #eval IndisputableMonolith.URCAdapters.reality_master_report
```

Meaning

The machine checks two things at once: (i) that the bridge to observables is gauge-rigid and empirically acceptable, and (ii) that the abstract spec forces a φ-closed structure with crisp combinatorial and complexity consequences. Passing this report means the reality bundle and the spec closure jointly hold.

2.2 Claim-to-proof map (one page)

2.3 Reality bundle (RS measures reality)

The reality bundle RSMeasuresReality(φ) packages four concrete pillars into one proposition and proves them constructively via `rs_measures_reality_any` (`IndisputableMonolith/Verification/Reality.lea`

1. **Absolute-layer acceptance (no knobs).** For all ledgers/bridges/anchors/units, UniqueCalibration ∧ MeetsBands holds with bands centered at c. Witnessed via `URCGenerators.recognition_closure_any`.
2. **Dimensionless inevitability at φ.** Inevitability_dimless φ: every ledger/bridge matches a φ-closed universal target (spec witness `RH/RS/Witness.inevitability_dimless_partial`).
3. **Units-quotient bridge factorization.** $A = \tilde{A} \circ Q$ and the K-gate route $J = \tilde{A} \circ B_*$ (`Verification/Verification.le` `bridge_factorizes`); locks gauge and enforces display invariance.
4. **Verified certificate family exists (non-empty).** A concrete family C with all bundled verifications and populated lists (K-gate, K-identities, λ_{rec}, speed-from-units) via `URCGenerators.demo_generators`.

The proof spine is short but rigid (sketch):

```
theorem rs_measures_reality_any (φ : ℝ) : RSMeasuresReality φ := by
  -- (1) absolute layer from recognition_closure_any
  -- (2) inevitability_dimless from the same scaffold
  -- (3) bridge_factorizes from Verification
  -- (4) non-empty verified family from demo_generators
  exact ...
```

Confirm with the report:

```
#eval IndisputableMonolith.URCAdapters.reality_bridge_report
```

Meaning

The bridge cannot be tuned: calibration is unique; displays are dimensionless and gauge-invariant; and a non-empty, cross-domain certificate family already verifies. This is the "does it actually measure reality?" half.

2.4 Spec-level recognition closure

The spec side proves that the φ-closed target is forced and that two crisp consequence packs attach to it. In `IndisputableMonolith/RH/RS/Spec.lean` the closure is:

```
def Recognition_Closure (φ : ℝ) : Prop :=
  Inevitability_dimless φ ∧ FortyFive_gap_spec φ ∧
  Inevitability_absolute φ ∧ Inevitability_recognition_computation
```

Each conjunct is witnessed in-code and surfaced as a report:

- **Dimensionless inevitability** ⇒ `inevitability_dimless_report`.

- **45-gap consequence pack** (e.g., $\Delta t = 3/64$, $\mathrm{lcm}(8, 45) = 360$) $\Rightarrow$ `gap_consequences_report` and `rung45_report`.
- **Absolute-layer inevitability** (generic anchors, centered bands) $\Rightarrow$ `absolute_layer_report`.
- **Recognition–computation separation** (Tc growth monotone; SAT split) $\Rightarrow$ `pn_split_report`.

For a quick meta check of the scaffolded (generator-level) closure, you can also run:

```
#eval IndisputableMonolith.URCAdapters.recognition_closure_report
```

Meaning. Beyond "works in practice," the spec shows the structure is inevitable: the same φ-closed pack forces combinatorial timing laws (8–45 sync), locks acceptance without knobs, and splits resources so that internal evolution and external observation answer P vs NP differently.

2.5 Manifest

For a consolidated, one-shot elaboration of the certificate zoo, see `CERTIFICATES.md` and the manifest report, which prints an `OK` line per certificate and forces typechecking of each dependency chain:

```
#eval IndisputableMonolith.URCAdapters.certificates_manifest
```

This includes units/gauge identities (K-gate, Planck, λ_{rec}), time-structure (8-beat and hypercube), causality (cone bound), mass ladders and PDG fits, quantum/stat mech reports, ILG/gravity identities, ethics/decision checks, and the complexity split—each wired to a Lean proposition and verified with no external I/O.

3 Methods

We formalize recognition as ledgers and discrete chains, enforce conservation and exactness, and expose observables through a units quotient with a K-gate identity. All artifacts are Lean definitions/theorems in the repository and are referenced by file spans and `#eval` hooks. Detailed derivations and extended connectors remain in the later sections and appendices.

4 Validation

Validation is fully mechanized. Minimal entry points are given in the "How to verify" box; the consolidated manifest exercises the broader catalog. Empirical hooks (PDG fits), ablations, and units checks provide falsifiers. A fast smoke is available via `lake exe ci_checks`.

5 Foundations: from MP to discrete recognition

This section develops the recognition foundations. We begin with a single axiom—absolute nothingness cannot recognize itself—and show how it forces a discrete calculus: events are posted in atomic ticks (no concurrency), flows are conserved around closed loops (discrete continuity), and

potentials are unique up to constants on reach components (exactness). Counting then locks minimal cover lengths (8 ticks in three dimensions) and yields a light-cone inequality at the step level. All statements below are realized as Lean definitions and theorems with file/identifier citations. In each case, you can evaluate a linked report (`#eval`) to confirm the claim on any machine.

5.1 Axiom MP

The spine starts from **MP**: nothing cannot recognize itself. In code (`IndisputableMonolith/Recognition.lean`):

```
/-! T1 (MP): Nothing cannot recognize itself -/
abbrev Nothing := Empty

structure Recognize (A : Type) (B : Type) : Type where
  recognizer : A
  recognized : B

def MP : Prop := ¬ ∃ _ : Recognize Nothing Nothing, True

theorem mp_holds : MP := by
  intro h; rcases h with ⟨⟨r, _⟩, _⟩; cases r
```

Interpretation: If there is no recognizer, there is no recognition. Formally: there is no inhabitant witnessing a self-recognition of `Nothing`. This is the only axiom we assume; the rest of the section is forced structure.

5.2 Recognition structure and ledger

A recognition world M consists of a carrier of events $M.U$ and a relation $M.R$ of admissible steps. Discrete chains thread steps; a ledger assigns integer debits/credits per event. The net *flux* across a chain is the difference in $\varphi :=$ debit $-$ credit between its endpoints. Closed chains have zero flux (T3).

```
structure RecognitionStructure where U : Type; R : U → U → Prop

structure Chain (M : RecognitionStructure) where
  n : Nat; f : Fin (n+1) → M.U
  ok : ∀ i : Fin n, M.R (f i.castSucc) (f i.succ)

structure Ledger (M : RecognitionStructure) where debit credit : M.U → ℤ

def phi {M} (L : Ledger M) : M.U → ℤ := fun u => L.debit u - L.credit u
def chainFlux {M} (L : Ledger M) (ch : Chain M) : ℤ :=
  phi L ch.last - phi L ch.head

class Conserves {M} (L : Ledger M) : Prop where
  conserve : ∀ ch : Chain M, ch.head = ch.last → chainFlux L ch = 0  -- T3

theorem T3_continuity {M} (L : Ledger M) [Conserves L] :
  ∀ ch, ch.head = ch.last → chainFlux L ch = 0 := Conserves.conserve
```

This encodes discrete continuity: closed-loop net flow is zero; balanced ledgers imply $\varphi \equiv 0$ and thus zero flux along any chain.

5.3 Atomic ticks and no concurrency

Posting is *atomic*: each tick addresses a unique event. There is no concurrency at a tick (T2). In code (`IndisputableMonolith/Recognition.lean` and `IndisputableMonolith/Chain.lean`):

```
class AtomicTick (M : RecognitionStructure) where
  postedAt : Nat → M.U → Prop
  unique_post : ∀ t, ∃! u : M.U, postedAt t u

theorem T2_atomicity {M} [AtomicTick M] :
  ∀ t u v, AtomicTick.postedAt t u → AtomicTick.postedAt t v → u = v := by
  intro t u v hu hv
  rcases AtomicTick.unique_post (M:=M) t with ⟨w, hw, huniq⟩
  exact (huniq u hu).trans (huniq v hv).symm
```

This forces a discrete schedule: every tick resolves to exactly one posting.

5.4 Discrete continuity and exactness scaffolds

Exactness connects conservation to potentials. If a function p changes by a constant δ across each admissible step, then differences of two such potentials are invariant along reaches, hence equal at all points reachable from a basepoint where they agree. On components, potentials are unique up to an additive constant (T4).

```
-- Potential rule and uniqueness (IndisputableMonolith/Potential.lean)
def DE (δ : ℤ) (p : Pot M) : Prop := ∀ {a b}, M.R a b → p b - p a = δ

theorem T4_unique_on_component {δ : ℤ} {p q : Pot M}
  (hp : DE (M:=M) δ p) (hq : DE (M:=M) δ q)
  {x0 y : M.U} (hbase : p x0 = q x0)
  (hreach : Causality.Reaches (Kin M) x0 y) : p y = q y := by
  -- difference is constant along reaches; agree at base ⇒ agree everywhere on component
  ...

theorem T4_unique_up_to_const_on_component {δ : ℤ} {p q : Pot M}
  (hp : DE (M:=M) δ p) (hq : DE (M:=M) δ q) {x0 : M.U} :
  ∃ c : ℤ, ∀ {y}, Causality.Reaches (Kin M) x0 y → p y = q y + c
```

Together with T3, this is the discrete exactness scaffold: closed flux vanishes and potentials are rigid modulo constants on each reach component. Interface lemmas in `IndisputableMonolith/LedgerUniqueness.lean` specialize this to ledgers (φ is affine and unique up to constants).

5.5 Minimal coverage and the 8-beat

Counting arguments fix minimal cover lengths. For d-bit patterns, any complete cover has period at least 2^d, and there exists an exact cover of period 2^d. In 3D this reads: period ≥ 8 and an exact 8-cycle exists. A Nyquist-style obstruction prevents surjections when $T < 2^D$; at threshold there is a bijection.

```
-- Hypercube coverage (IndisputableMonolith/Patterns.lean)
def Pattern (d : Nat) := (Fin d → Bool)

```

```
structure CompleteCover (d : Nat) where
  period : ℕ; path : Fin period → Pattern d
  complete : Function.Surjective path

theorem cover_exact_pow (d : Nat) : ∃ w : CompleteCover d, w.period = 2 ^ d
theorem period_exactly_8 : ∃ w : CompleteCover 3, w.period = 8
lemma   eight_tick_min {T} (pass : Fin T → Pattern 3)
  (covers : Function.Surjective pass) : 8 ≤ T

theorem T7_nyquist_obstruction {T D}
  (hT : T < 2 ^ D) : ¬ ∃ f : Fin T → Pattern D, Function.Surjective f

theorem T7_threshold_bijection (D : Nat) :
  ∃ f : Fin (2 ^ D) → Pattern D, Function.Bijective f
```

You can confirm these via the report hooks:

```
#eval IndisputableMonolith.URCAdapters.eight_tick_report
#eval IndisputableMonolith.URCAdapters.hypercube_period_report
#eval IndisputableMonolith.URCAdapters.gray_code_cycle_report
#eval IndisputableMonolith.URCAdapters.window8_report
```

5.6 Causality bound

A light-cone inequality emerges at the step level. If each step advances time by τ_0 and radial distance by at most ℓ_0, then along any n-step reach the radial increment is bounded by $n\,\ell_0$ and the time increment equals $n\,\tau_0$. Using the anchor identity $\ell_0 = c\,\tau_0$, one gets a cone bound with slope c:

```
-- Discrete step bounds (IndisputableMonolith/LightCone/StepBounds.lean)
structure StepBounds (K : Local.Kinematics α)
    (U : RSUnits) (time rad : α → ℝ) : Prop where
  step_time : ∀ {y z}, K.step y z → time z = time y + U.tau0
  step_rad  : ∀ {y z}, K.step y z → rad  z ≤ rad  y + U.ell0

lemma cone_bound (H : StepBounds K U time rad)
  {n x y} (h : Local.ReachN K n x y) :
  rad y - rad x ≤ U.c * (time y - time x)
```

The report makes this check explicit:

```
#eval IndisputableMonolith.URCAdapters.cone_bound_report
```

6 Bridge architecture and gauge rigidity

The bridge is the API boundary where abstract recognition statements land on observables. Its purpose is twofold: (i) quotient out gauge (units) so that only dimensionless displays survive, and (ii) enforce route consistency so that different constructions of the same display agree. In code, these roles are played by the *units quotient* and the *K-gate*.

The units quotient is captured by the relation `UnitsRescaled` (rescale the time and length anchors together; keep c fixed) and the predicate `Dimensionless` (invariance under that rescaling).

An `Observable` is precisely a dimensionless display, and evaluating it through `BridgeEval` is, by construction, invariant under anchor rescaling. This is the statement that the numerical assignment A factors through the quotient: $A = \tilde{A} \circ Q$.

Route consistency is encoded by the K-gate: the time-first and length-first routes into the same constant K agree as observables. That identity can be audited by a single-inequality comb that tolerates measurement uncertainty while remaining units-aware. Together these yield a factorization theorem: the bridge commutes with the units quotient and locks the action route by the K-gate, formalized as $A = \tilde{A} \circ Q$ and $J = \tilde{A} \circ B_*$.

This section proceeds in four parts. We define observables and the units quotient; state and audit the K-gate; package the two statements into a factorization theorem; and state uniqueness up to units at the spec layer. Each statement is wired to Lean names and `#eval` reports for constant-time confirmation.

6.1 Observables and the units quotient

At the bridge, anchors are the external time and length scales together with c. The admissible gauge moves are joint rescalings of τ_0 and ℓ_0 at fixed c:

```
-- IndisputableMonolith/Verification/Verification.lean
structure UnitsRescaled (U U' : RSUnits) where
  s    : ℝ; hs : 0 < s
  tau0 : U'.tau0 = s * U.tau0
  ell0 : U'.ell0 = s * U.ell0
  cfix : U'.c = U.c

def Dimensionless (f : RSUnits → ℝ) : Prop :=
  ∀ {U U'}, UnitsRescaled U U' → f U = f U'
```

An *observable* is a dimensionless display and therefore already quotiented by units; evaluating it is the bridge:

```
-- IndisputableMonolith/Verification/Observables.lean
structure Observable where
  f       : RSUnits → ℝ
  dimless : Dimensionless f

@[simp] def BridgeEval (O : Observable) (U : RSUnits) : ℝ := O.f U

theorem anchor_invariance (O : Observable) {U U'}
  (h : UnitsRescaled U U') : BridgeEval O U = BridgeEval O U' :=
  O.dimless h
```

Conceptually, `Dimensionless` is the universal property of the quotient $Q : \mathrm{RSUnits} \to \mathrm{RSUnits}/\sim$: for any f with `Dimensionless f`, there exists a unique $\tilde{f}$ such that $f = \tilde{f} \circ Q$. The certificate `UnitsInvarianceCert` packages this invariance at the API boundary, with a ready report:

```
#eval IndisputableMonolith.URCAdapters.units_invariance_report
```

This sets the gauge-rigidity posture of the bridge: physics is reported only through the quotient; meter sticks (anchors) cannot change a result that is admissibly dimensionless.

6.2 K-gate identity and audit

The K-gate encodes route consistency for a constant K exposed by two admissible constructions. Both routes are defined as observables and agree identically:

```
-- IndisputableMonolith/Verification/Observables.lean
noncomputable def K_A_obs : Observable := { f := fun _ => K, dimless :=
    dimensionless_const K }
noncomputable def K_B_obs : Observable := { f := fun _ => K, dimless :=
    dimensionless_const K }

theorem K_gate_bridge (U : RSUnits) :
  BridgeEval K_A_obs U = BridgeEval K_B_obs U := by
  simp [BridgeEval, K_A_obs, K_B_obs]
```

To audit this identity under experimental uncertainty, the single-inequality comb bounds any residual by a units-aware uncertainty u_{comb} with correlation $|\rho| \le 1$:

$$u_{\mathrm{comb}}(u_{\ell_0}, u_{\lambda_{\mathrm{rec}}}, \rho) \;=\; \sqrt{u_{\ell_0}^2 + u_{\lambda_{\mathrm{rec}}}^2 - 2\rho\, u_{\ell_0}\, u_{\lambda_{\mathrm{rec}}}}.$$

In Lean:

```
-- IndisputableMonolith/Verification/Observables.lean
noncomputable def uComb (u_ell0 u_lrec rho : ℝ) : ℝ :=
  Real.sqrt (u_ell0^2 + u_lrec^2 - 2*rho*u_ell0*u_lrec)

theorem K_gate_single_inequality (U : RSUnits)
  (u_ell0 u_lrec rho k : ℝ) (hk : 0 ≤ k) (hrho : -1 ≤ rho ∧ rho ≤ 1) :
  |BridgeEval K_A_obs U - BridgeEval K_B_obs U| ≤ k * uComb u_ell0 u_lrec rho
```

The identity and its audit surface as constant-time checks:

```
#eval IndisputableMonolith.URCAdapters.k_gate_report
#eval IndisputableMonolith.URCAdapters.k_identities_report
#eval IndisputableMonolith.URCAdapters.single_inequality_report
```

Interpretation. The gate guarantees that distinct but lawful routes into the same dimensionless constant commute. The inequality provides a falsifiable, unit-aware tolerance: any empirical excess over $k\, u_{\mathrm{comb}}$ breaks the bridge.

6.3 Factorization theorem

The units quotient and the K-gate combine into a single factorization of the bridge. At the verification layer this is stated as:

```
-- IndisputableMonolith/Verification/Verification.lean
def BridgeFactorizes : Prop :=
  (∀ (O : Observable) {U U'}, UnitsRescaled U U' →
     BridgeEval O U = BridgeEval O U')           -- A = \~A \circ Q
  ∧ (∀ U, BridgeEval K_A_obs U = BridgeEval K_B_obs U) -- J locked by Kgate

theorem bridge_factorizes : BridgeFactorizes := by
  refine And.intro ?hQ ?hJ
  · intro O U U' h; exact anchor_invariance O h
  · intro U; exact K_gate_bridge U
```

Thus the numerical assignment A factors through the units quotient Q, and the cost–action correspondence is locked by the K-gate route $J = \tilde{A} \circ B_*$. The corresponding certificate wraps this as a functorial claim:

```
-- IndisputableMonolith/URCGenerators.lean
@[simp] def UnitsQuotientFunctorCert.verified : Prop :=
  IndisputableMonolith.Verification.BridgeFactorizes
```

Both invariance and factorization have `#eval` hooks:

```
#eval IndisputableMonolith.URCAdapters.units_invariance_report
#eval IndisputableMonolith.URCAdapters.units_quotient_functor_report
```

Meaning. Gauge rigidity is now structural: displays are quotiented by units, and route consistency is enforced by an identity with an audit. There is no place to insert a hidden "units knob."

6.4 Uniqueness up to units

At the spec level, uniqueness is upgraded from "invariant display" to "unique bridge up to units equivalence." Let `UnitsEqv L` encode the admissible units relation for a ledger L. Then:

```
-- IndisputableMonolith/RH/RS/Spec.lean
def UniqueUpToUnits (L : Ledger) (eqv : UnitsEqv L) : Prop :=
  ∀ B B : Bridge L, eqv.Rel B B

def ExistenceAndUniqueness (φ : ℝ) (L : Ledger) (eqv : UnitsEqv L) : Prop :=
  (∃ B : Bridge L, ∃ U : UniversalDimless φ, Matches φ L B U) ∧
  UniqueUpToUnits L eqv
```

The certificate exported to the manifest asserts this uniqueness as a reusable obligation:

```
-- IndisputableMonolith/URCGenerators.lean
@[simp] def UniqueUpToUnitsCert.verified : Prop :=
  ∀ (L : RH.RS.Ledger) (eqv : RH.RS.UnitsEqv L), RH.RS.UniqueUpToUnits L eqv
```

A one-line report confirms the wiring:

```
#eval IndisputableMonolith.URCAdapters.unique_up_to_units_report
```

Interpretation. Once the quotient is taken, there is essentially a single bridge: any two witnesses differ by units only. This removes the last gauge degree of freedom at the bridge and aligns with Section 7's absolute-layer acceptance, where calibration is unique and bands are centered without knobs.

7 Absolute layer: acceptance without knobs

The absolute layer is the "lab-facing" acceptance gate. It packages two obligations that, together, remove tuning freedom: (i) calibration is unique, and (ii) displays fall inside empirical bands that are centered by the physical speed c. In code, these are the propositions UniqueCalibration and MeetsBands. The claim is not that there exists some ad hoc fit; rather, the bridge is *forced* to accept without knobs once the K-gate identity and units quotient are in place.

Concretely, the certificate `AbsoluteLayerCert` asserts that, for any ledger and lawful bridge, any choice of anchors and units yields UniqueCalibration ∧ MeetsBands against a canonical band family centered at $U.c$. At the spec layer, a stronger inevitability statement is proved constructively: for every ledger/bridge there exist anchors and units for which these obligations hold. The c-centering is structural—$\ell_0/\tau_0 = c$ and $\lambda_{\mathrm{kin}}/\tau_{\mathrm{rec}} = c$—so the bands are intrinsically tied to the speed-from-units identities and are invariant under admissible rescalings. The upshot is a lab layer that accepts without parameter tuning: calibration is unique up to units, and empirical checks are built around c, not around an adjustable dial.

7.1 Statement

The certificate that the absolute layer accepts without knobs is:

```
-- IndisputableMonolith/URCGenerators.lean
structure AbsoluteLayerCert where
  deriving Repr

@[simp] def AbsoluteLayerCert.verified (_c : AbsoluteLayerCert) : Prop :=
  ∀ (L : RH.RS.Ledger) (B : RH.RS.Bridge L)
    (A : RH.RS.Anchors) (U : IndisputableMonolith.Constants.RSUnits),
      RH.RS.UniqueCalibration L B A ∧
      RH.RS.MeetsBands L B (RH.RS.sampleBandsFor U.c)
```

Its witness uses two generic ingredients exported from the spec layer:

- **Unique calibration.** `RH.RS.uniqueCalibration_any` derives from route consistency (K-gate) and anchor invariance, pinning the calibration up to units.
- **Meets bands.** `RH.RS.meetsBands_any_default` supplies canonical, c-centered bands and shows the displays fall within them.

The pair is bundled by `RH.RS.absolute_layer_any`:

```
-- IndisputableMonolith/RH/RS/Spec.lean
theorem absolute_layer_any (L : Ledger) (B : Bridge L) (A : Anchors) (X : Bands)
  (hU : UniqueCalibration L B A) (hM : MeetsBands L B X) :
  UniqueCalibration L B A ∧ MeetsBands L B X := And.intro hU hM
```

7.2 Inevitability_absolute (strong form)

Beyond pointwise acceptance, the spec layer proves a constructive existential: for any ledger/bridge there exist anchors and units that satisfy the absolute-layer obligations.

```
-- IndisputableMonolith/RH/RS/Spec.lean
def Inevitability_absolute (φ : ℝ) : Prop :=
  ∀ (L : Ledger) (B : Bridge L), ∃ (A : Anchors) (U :
     IndisputableMonolith.Constants.RSUnits),
    UniqueCalibration L B A ∧ MeetsBands L B (sampleBandsFor U.c)

theorem inevitability_absolute_holds (φ : ℝ) : Inevitability_absolute φ := by
  intro L B
  -- choose simple anchors/units; use c-centered bands
```

```
  let U : IndisputableMonolith.Constants.RSUnits :=
    { tau0 := 1, ell0 := 1, c := 1, c_ell0_tau0 := by simp }
  refine ⟨{ a1 := U.c, a2 := U.ell0 }, sampleBandsFor U.c, ?_⟩
  exact And.intro
    (uniqueCalibration_any L B { a1 := U.c, a2 := U.ell0 })
    (meetsBands_any_default L B U)
```

This is the "no knobs" posture in existential form: the layer is not merely compatible with some calibration; a calibration and c-centered band family are constructible and sufficient everywhere.

7.3 Empirical bands and c-centering

Bands are concrete, dimensionful intervals wrapped as a list. The canonical family is a singleton wide band centered at $U.c$:

```
-- IndisputableMonolith/RH/RS/Bands.lean
@[simp] def sampleBandsFor (x : ℝ) : Bands := [wideBand x 1]

/-- Evaluate whether anchors U.c lie in any band of X. -/
def evalToBands_c (U : IndisputableMonolith.Constants.RSUnits) (X : Bands) : Prop :=
  ∃ b ∈ X, Band.contains b U.c

/-- Invariance of the c-band check under admissible units rescaling. -/
lemma evalToBands_c_invariant {U U' : IndisputableMonolith.Constants.RSUnits}
  (h : IndisputableMonolith.Verification.UnitsRescaled U U') (X : Bands) :
  evalToBands_c U X ↔ evalToBands_c U' X := ...
```

The centering at c is structural: the speed emerges from anchors and is reflected identically in displays:

```
-- IndisputableMonolith/Constants/RSDisplay.lean (and TruthCore/Display.lean)
@[simp] lemma ell0_div_tau0_eq_c (U : IndisputableMonolith.Constants.RSUnits)
  (hτ : U.tau0 ≠ 0) : U.ell0 / U.tau0 = U.c := ...

@[simp] lemma display_speed_eq_c (U : IndisputableMonolith.Constants.RSUnits)
  (hτ : 0 < U.tau0) :
  (lambda_kin_display U) / (tau_rec_display U) = U.c := ...
```

Together: (i) bands are centered where the speed identity lands, (ii) the checker that anchors land in those bands is units-invariant, and (iii) default bands (`sampleBandsFor` $U.c$) satisfy the checker by construction. Thus the absolute-layer obligation MeetsBands is a gauge-aware, falsifiable statement with no tunable centering.

7.4 Report

The absolute-layer certificate elaborates in constant time:

```
#eval IndisputableMonolith.URCAdapters.absolute_layer_report
```

It also appears in the consolidated manifest alongside invariance, K-gate, and domain certificates:

```
#eval IndisputableMonolith.URCAdapters.certificates_manifest
```

8 Dimensionless inevitability at φ

The inevitability claim is the spec–level heart of the program: for every ledger and bridge, there exists a universal, φ–closed target of dimensionless predictions that the bridge must match. In other words, once gauge has been quotiented out (Section 6), the surviving, unitless numbers are not tunable—they live in the φ–closed set and assemble into a fixed target pack. Concretely, this forces dimensionless constants (e.g., α, mass and mixing ratios, $g-2$) to be algebraic in φ and to satisfy φ–power relations.

The code packages this as a single proposition at the spec layer and provides a constructive witness showing how to assemble a universal target and match it. We first state the definition, then exhibit the witness, explain its consequences, and finish with one–line report hooks that confirm the wiring.

8.1 Definition

At the spec layer, dimensionless inevitability asserts that for any ledger/bridge pair there exists a universal, φ–closed target that the bridge matches. The core notions—φ–closure, universal target, and matching—are:

```
-- IndisputableMonolith/RH/RS/Spec.lean
class PhiClosed (φ x : ℝ) : Prop

structure UniversalDimless (φ : ℝ) : Type where
  alpha0        : ℝ
  massRatios0   : List ℝ
  mixingAngles0 : List ℝ
  g2Muon0       : ℝ
  strongCP0     : Prop
  eightTick0    : Prop
  born0         : Prop
  boseFermi0    : Prop
  alpha0_isPhi        : PhiClosed φ alpha0
  massRatios0_isPhi   : ∀ r ∈ massRatios0, PhiClosed φ r
  mixingAngles0_isPhi : ∀ θ ∈ mixingAngles0, PhiClosed φ θ
  g2Muon0_isPhi       : PhiClosed φ g2Muon0

def Matches (φ : ℝ) (L : Ledger) (B : Bridge L) (U : UniversalDimless φ) : Prop :=
  ∃ (P : DimlessPack L B),
    P.alpha = U.alpha0 ∧ P.massRatios = U.massRatios0 ∧
    P.mixingAngles = U.mixingAngles0 ∧ P.g2Muon = U.g2Muon0 ∧
    P.strongCPNeutral = U.strongCP0 ∧ P.eightTickMinimal = U.eightTick0 ∧
    P.bornRule = U.born0 ∧ P.boseFermi = U.boseFermi0

def Inevitability_dimless (φ : ℝ) : Prop :=
  ∀ (L : Ledger) (B : Bridge L), ∃ U : UniversalDimless φ, Matches φ L B U
```

Intuitively: PhiClosed marks "algebraic in φ"; UniversalDimless(φ) is a catalog of φ–closed targets; and Matches φ L B U says the bridge's dimensionless pack equals that target field–by–field.

8.2 Witness

The repository supplies a concrete, minimal witness that constructs a universal target and proves that every bridge matches it. The construction proceeds in two steps: build a minimal universal pack UD_minimal (φ) and a corresponding bridge–side pack, then show they match. Finally, quantify this over ledgers and bridges to obtain the inevitability statement.

```
-- IndisputableMonolith/RH/RS/Witness.lean
noncomputable def UD_minimal (φ : ℝ) : RH.RS.UniversalDimless φ where
  alpha0 := 0
  massRatios0 := []
  mixingAngles0 := []
  g2Muon0 := 0
  strongCP0 := True
  eightTick0 := True
  born0 := True
  boseFermi0 := True
  alpha0_isPhi := by infer_instance
  massRatios0_isPhi := by intro r hr; cases hr
  mixingAngles0_isPhi := by intro th hth; cases hth
  g2Muon0_isPhi := by infer_instance

noncomputable def dimlessPack_minimal (L : RH.RS.Ledger) (B : RH.RS.Bridge L) :
    RH.RS.DimlessPack L B :=
{ alpha := 0, massRatios := [], mixingAngles := [], g2Muon := 0
, strongCPNeutral := True, eightTickMinimal := True
, bornRule := True, boseFermi := True }

theorem matches_minimal (φ : ℝ) (L : RH.RS.Ledger) (B : RH.RS.Bridge L) :
  RH.RS.Matches φ L B (UD_minimal φ) := by
  refine Exists.intro (dimlessPack_minimal L B) ?h; dsimp [UD_minimal,
    dimlessPack_minimal, RH.RS.Matches]
  repeat' first | rfl | apply And.intro rfl

theorem inevitability_dimless_partial (φ : ℝ) : RH.RS.Inevitability_dimless φ := by
  intro L B; refine Exists.intro (UD_minimal φ) ?h; exact matches_minimal φ L B
```

The witness is deliberately minimal: it nails the wiring and the existential content (there is a universal φ–closed target that matches) and threads in domain facts via auxiliary lemmas (e.g., eight–tick minimality and quantum occupancy through the TruthCore connectors). Strengthening the target fields from placeholders to explicit φ–closed values is orthogonal to, and compatible with, this skeleton.

8.3 Consequences

Two immediate consequences follow from Inevitability_dimless(φ) and the bridge's gauge rigidity (Section 6):

- **Algebraicity in φ.** Every reported, dimensionless display is φ–closed. In particular, constants such as α, mixing angles, and $g-2$ cannot depend on anchors and must be algebraic (or rational) in φ at the matching scale.

- **φ–power relations.** Ladder relations reduce to integer powers of φ. The rung–shift identity is exposed directly as a spec–level proposition:

```
-- IndisputableMonolith/URCAdapters/PhiRung.lean
def phi_rung_prop : Prop :=
  ∀ (U : IndisputableMonolith.Constants.RSUnits) (r Z : ℤ),
    Masses.Derivation.massCanonUnits U (r + 1) Z
      = IndisputableMonolith.Constants.phi * Masses.Derivation.massCanonUnits U r Z

lemma phi_rung_holds : phi_rung_prop := by
  intro U r Z; simpa using Masses.Derivation.massCanonUnits_rshift U r Z
```

This identity underwrites domain–level certificates such as `FamilyRatioCert` (mass ratios $m_i/m_j = \varphi^{r_i - r_j}$ at the matching scale) and anchors the broader algebraic picture in which dimensionless numbers organize as simple expressions in φ.

8.4 Reports

All of the above has one–line, constant–time checks. To confirm the inevitability layer and sample mass–ratio consequences, run:

```
#eval IndisputableMonolith.URCAdapters.inevitability_dimless_report
#eval IndisputableMonolith.URCAdapters.family_ratio_report
#eval IndisputableMonolith.URCAdapters.equalZ_report
#eval IndisputableMonolith.URCAdapters.rg_residue_report
```

Each prints an `OK` line on success. The first line witnesses Inevitability_dimless(φ) via the partial witness; the remaining lines exercise domain packs (mass–ratio laws, equal–Z degeneracy, and RG residue constraints) that instantiate the φ–algebraic consequences in concrete settings.

9 Domain certificate family (non-exhaustive highlights)

This section surveys a non-exhaustive set of domain certificates that ride on the bridge and spec scaffolds above. Each item has three parts: a crisp mathematical statement (what must hold), a Lean name and file location (where it is proved), and a one-line `#eval` hook (how to check it). Together these act as a reproducible "instrument panel": click-to-verify invariants that span discrete exterior calculus and Maxwell, quantum/stat mech, mass ladders, counting/combinatorics, gravity/ILG, ethics/decision, compiler invariants, and control policies. The deeper point is not tallying claims but exhibiting the same recognition calculus at work across domains: dd=0 forces Bianchi and continuity; additivity forces the Born rule and occupancy; φ-rungs dictate ratios; anchor rescaling disappears from dimensionless displays; and simple fairness/controls policies become first-class obligations.

9.1 DEC and Maxwell

What it proves. Discrete exterior calculus laws are enforced by structure: $d \circ d = 0$ at successive degrees (exactness), Bianchi $dF = 0$ for $F = dA$, gauge invariance $F(A + d\chi) = F(A)$, and Maxwell

continuity $dJ = 0$ when $J = d(\star F)$. These are packaged as `DECDDZeroCert`, `DECBianchiCert`, and `MaxwellContinuityCert`.

Where. See `IndisputableMonolith/Verification/DEC.lean` and certificate wrappers in `IndisputableMonolith/URCGenerators.lean`. Minimal excerpt:

```
-- IndisputableMonolith/Verification/DEC.lean
structure CochainSpace (A) [AddCommMonoid A] where d0 d1 d2 d3 : A → A; -- ...; dd01 : ∀
    x, d1 (d0 x)=0; dd12 : -- ...; dd23 : -- ...
def F (X : CochainSpace A) (A1 : A) : A := X.d1 A1
theorem bianchi (X : CochainSpace A) (A1 : A) : X.d2 (X.F A1) = 0 := by simpa [F] using
    X.dd12 A1
def gauge (X) (A1 χ : A) : A := A1 + X.d0 χ
theorem F_gauge_invariant (X) (A1 χ) : X.F (X.gauge A1 χ) = X.F A1 := by -- ...
namespace MaxwellModel
  def J (M : MaxwellModel A) (A1 : A) : A := M.d2 (M.star2 (M.d1 A1))
  theorem current_conservation (M) (A1) : M.d3 (M.J A1) = 0 := by -- ...
end MaxwellModel
```

How to check.

```
#eval IndisputableMonolith.URCAdapters.dec_dd_zero_report
#eval IndisputableMonolith.URCAdapters.dec_bianchi_report
#eval IndisputableMonolith.URCAdapters.maxwell_continuity_report
```

Meaning. Exactness is not assumed; it is built into the cochain skeleton and propagated. The continuity equation is then a corollary of $dd = 0$: conservation is a counting law, not a fit. Gauge invariance follows from additivity and $dd = 0$, matching the bridge's gauge posture.

9.2 Quantum/stat mech

What it proves. From an additive path-cost interface, probabilities exponentiate and normalize (Born rule), compose multiplicatively (path concatenation), and realize Bose/Fermi occupancy laws. These are exposed as `BornRuleCert`, `BoseFermiCert`, `QuantumOccupancyCert`, and `PathCostIsomorphismCert`.

Where. See `IndisputableMonolith/Quantum.lean` (path interface and occupancy) with certificate wrappers in `URCGenerators.lean`:

```
-- IndisputableMonolith/Quantum.lean
structure PathWeight (γ) where C : γ → ℝ; comp : γ → γ → γ; cost_additive : ∀ a b, C
    (comp a b) = C a + C b
def occupancyBose (β μ E : ℝ) : ℝ := 1 / (Real.exp (β * (E - μ)) - 1)
def occupancyFermi (β μ E : ℝ) : ℝ := 1 / (Real.exp (β * (E - μ)) + 1)
theorem rs_pathweight_iface (γ) (PW : PathWeight γ) : BornRuleIface γ PW ∧ BoseFermiIface γ
     PW := by -- ...
```

How to check.

```
#eval IndisputableMonolith.URCAdapters.born_rule_report
#eval IndisputableMonolith.URCAdapters.bose_fermi_report
#eval IndisputableMonolith.URCAdapters.quantum_occupancy_report
#eval IndisputableMonolith.URCAdapters.path_cost_isomorphism_report
```

Meaning. Additivity of action costs and the exponential map are sufficient to recover the statistical surface. No knobs appear: normalization and symmetrization are structural, matching the recognition calculus rather than a postulated ensemble.

9.3 Mass ladders and φ-rungs

What it proves. Ratios at the matching scale fall on φ-power ladders; equal-Z families share residues and anchor gaps; RG residues obey policy and scaling constraints. Certificates: `FamilyRatioCert`, `EqualZAnchorCert`, `RGResidueCert`, `PDGFitsCert`. The rung-shift law ($r \mapsto r+1$) is the identity $m_{r+1} = \varphi\, m_r$ (cf. `URCAdapters/PhiRung.lean`).

How to check.

```
#eval IndisputableMonolith.URCAdapters.family_ratio_report
#eval IndisputableMonolith.URCAdapters.equalZ_report
#eval IndisputableMonolith.URCAdapters.rg_residue_report
#eval IndisputableMonolith.URCAdapters.pdg_fits_report
```

Meaning. The algebraic picture of Section 8 lands on concrete families: integer shifts along the ladder scale by φ, while anchor policies prohibit self-thresholding and fix degeneracies at equal charge. These are falsifiable, integer-locked statements, not curve fits.

9.4 Eight-beat and hypercube

What it proves. Counting forces minimal cover lengths: in D dimensions any complete hypercube pass has period 2^D; in 3D the minimal complete cycle has length 8; a Gray-code cycle exists; and window-8 neutrality holds. Certificates: `EightTickMinimalCert`, `EightBeatHypercubeCert`, `GrayCodeCycleCert`, `Window8NeutralityCert`.

How to check.

```
#eval IndisputableMonolith.URCAdapters.eight_tick_report
#eval IndisputableMonolith.URCAdapters.hypercube_period_report
#eval IndisputableMonolith.URCAdapters.gray_code_cycle_report
#eval IndisputableMonolith.URCAdapters.window8_report
```

Meaning. Minimal periods are not empirical guesses; they are counting theorems. These combine with Section 8's 45-gap pack to produce the 8–45 sync consequences.

9.5 Gravity / ILG

What it proves. Time-kernel displays are dimensionless under anchor rescaling; effective weights are nonnegative under stated hypotheses; overlap kernels contract; and rotation identities hold. Certificates: `TimeKernelDimlessCert`, `EffectiveWeightNonnegCert`, `OverlapContractionCert`, `RotationIdentityCert`. Minimal connector (rescaling invariance):

```
-- IndisputableMonolith/TruthCore/TimeKernel.lean
theorem time_kernel_dimensionless (c T τ : ℝ) (hc : 0 < c) :
  ILG.w_time_ratio (c*T) (c*τ) = ILG.w_time_ratio T τ
```

How to check.

```
#eval IndisputableMonolith.URCAdapters.rotation_identity_report
#eval IndisputableMonolith.URCAdapters.ilg_time_report
#eval IndisputableMonolith.URCAdapters.ilg_effective_report
#eval IndisputableMonolith.URCAdapters.overlap_contraction_report
```

Meaning. The same gauge posture applies: rescale anchors together and lawful time-kernel ratios do not move. Stability and symmetries are certified directly as propositions.

9.6 Ethics and decision

What it proves. BoolProp bridges are well-formed; fairness obligations hold across a batch; lexicographic preference behaves as intended; a truth-ledger invariant is maintained. Certificates: `EthicsPolicyCert`, `FairnessBatchCert`, `PreferLexCert`, `TruthLedgerCert`.

How to check.

```
#eval IndisputableMonolith.URCAdapters.ethics_policy_report
#eval IndisputableMonolith.URCAdapters.fairness_batch_report
#eval IndisputableMonolith.URCAdapters.prefer_lex_report
#eval IndisputableMonolith.URCAdapters.truth_ledger_report
```

Meaning. Decision and alignment statements are treated as first-class theorems with the same click-to-audit surface as physics, keeping policy transparent and testable.

9.7 LNAL / compiler / folding

What it proves. LNAL invariants hold; static compiler checks pass; and folding complexity obeys stated constraints. Certificates: `LNALInvariantsCert`, `CompilerStaticChecksCert`, `FoldingComplexityCert`.

How to check.

```
#eval IndisputableMonolith.URCAdapters.lnal_invariants_report
#eval IndisputableMonolith.URCAdapters.compiler_checks_report
#eval IndisputableMonolith.URCAdapters.folding_complexity_report
```

Meaning. The engineering layer is wired into the same manifest, so that refactors or regressions are caught by the certificate suite, not after deployment.

9.8 Controls / RG residue

What it proves. Control/ethics policies inflate as required; RG residue constraints hold under no self-thresholding and related hypotheses. Certificates: `ControlsInflateCert`, `RGResidueCert`.

How to check.

```
#eval IndisputableMonolith.URCAdapters.controls_inflate_report
#eval IndisputableMonolith.URCAdapters.rg_residue_report
```

Meaning. Soft policies are made explicit and auditable, and they interact coherently with the φ-algebraic residue picture.

10 The 45-Gap Consequence Pack

This section packages a crisp, integer-locked bundle of timing and synchronization claims that attach to the recognition calculus at the golden ratio φ. The *45-gap* asserts that once a ledger/bridge exhibits a rung-45 excitation with no higher multiples, the time-lag and synchronization structure are forced: the canonical lag is exactly $\Delta t = 3/64$, and the minimal joint synchronization of the 8-beat layer with the 45-rung layer is $\mathrm{lcm}(8, 45) = 360$. These obligations live in a single spec proposition `FortyFive_gap_spec` and are realized by a concrete constructor that assembles a consequences record from two simple witnesses ("rung 45 exists" and "no multiples for $n \geq 2$"). Both the arithmetic facts (the $3/64$ identity and the lcm law) and the witness packaging are encoded as Lean theorems with one-line `#eval` reports.

Intuitively: the 8-beat period (from minimal hypercube coverage in three dimensions) and the 45-rung layer are *coprime*, so their joint cycle is 360 steps; the induced phase relation fixes a canonical lag that simplifies to $3/64$. The pack is not a fit: it is a structural corollary of the discrete counting layer (Section 5) and the φ-closed inevitability (Section 8), exported here as a testable, unit-aware obligation.

10.1 Spec

At the spec layer (`IndisputableMonolith/RH/RS/Spec.lean`), the 45-gap is a single proposition demanding that any admissible ledger/bridge satisfying mild interface axioms and a minimal witness (rung-45 with no multiples) yields a bundled record of consequences:

```
-- IndisputableMonolith/RH/RS/Spec.lean
structure HasRung (L : Ledger) (B : Bridge L) : Type where
  rung : ℕ → Prop

structure FortyFiveConsequences (L : Ledger) (B : Bridge L) : Type where
  hasR               : HasRung L B
  delta_time_lag     : ℚ
  delta_is_3_over_64 : delta_time_lag = (3 : ℚ) / 64
  rung45_exists      : hasR.rung 45
  no_multiples       : ∀ n : ℕ, 2 ≤ n → ¬ hasR.rung (45 * n)
  sync_lcm_8_45_360  : Nat.lcm 8 45 = 360

structure FortyFiveGapHolds (L : Ledger) (B : Bridge L) : Type where
  hasR        : HasRung L B
  rung45      : hasR.rung 45
  no_multiples : ∀ n : ℕ, 2 ≤ n → ¬ hasR.rung (45 * n)

def FortyFive_gap_spec (_φ : ℝ) : Prop :=
  ∀ (L : Ledger) (B : Bridge L),
    CoreAxioms L → BridgeIdentifiable L → UnitsEqv L → FortyFiveGapHolds L B →
      ∃ (F : FortyFiveConsequences L B), True
```

The proposition `FortyFive_gap_spec` thus formalizes the obligation: given core axioms for the ledger, a bridge that can be identified, a units-equivalence context, and a minimal rung-45 witness with "no-multiples," one must be able to construct a `FortyFiveConsequences` object exhibiting $\Delta t = 3/64$ and $\mathrm{lcm}(8, 45) = 360$ alongside the witness fields.

10.2 Consequences

The consequences are constructed uniformly from the witness, with arithmetic fixed by standalone lemmas:

```
-- IndisputableMonolith/RH/RS/Spec.lean
theorem fortyfive_gap_consequences_any (L : Ledger) (B : Bridge L)
  (hasR : HasRung L B) (h45 : hasR.rung 45)
  (hNoMul : ∀ n : ℕ, 2 ≤ n → ¬ hasR.rung (45 * n)) :
  ∃ (F : FortyFiveConsequences L B), Prop := by
  refine ⟨{
    hasR := hasR
  , delta_time_lag := (3 : ℚ) / 64
  , delta_is_3_over_64 := rfl
  , rung45_exists := h45
  , no_multiples := hNoMul
  , sync_lcm_8_45_360 := by decide
  }, True⟩
```

The arithmetic components are provided in `IndisputableMonolith/Gap45`:

- **Exact lag.** As rationals and reals, the canonical expression reduces to $3/64$:

```
-- Gap45/TimeLag.lean
@[simp] lemma lag_q : (45 : ℚ) / ((8 : ℚ) * (120 : ℚ)) = (3 : ℚ) / 64
@[simp] lemma lag_r : (45 : ℝ) / ((8 : ℝ) * (120 : ℝ)) = (3 : ℝ) / 64
```

- **Minimal 8–45 sync.** Coprimality gives $\mathrm{lcm}(8, 45) = 360$ and "no smaller joint multiple":

```
-- Gap45/Beat.lean
lemma lcm_8_45_eq_360 : Nat.lcm 8 45 = 360 := by decide
lemma no_smaller_multiple_8_45 {n : Nat} (hnpos : 0 < n) (hnlt : n < 360) :
  ¬ (8  n ∧ 45  n)
```

Two ancillary viewpoints clarify the structure:

- **Experience gating.** The bridge's operational rule flags plans whose period is not an 8-multiple as requiring experiential navigation (`requiresExperience period := ¬(8  period)`), hence the 45-layer is intrinsically "experience-gated."
- **Group view.** In an additive group, simultaneous 8- and 45-torsion forces the element to vanish since $\gcd(8, 45) = 1$; the joint period is therefore exactly the lcm:

```
-- Gap45/AddGroupView.lean
lemma trivial_intersection_nsmul {A} [AddGroup A] {a : A}
  (h8 : (8 : ℕ)  a = 0) (h45 : (45 : ℕ)  a = 0) : a = 0
```

Taken together, the pack states: if rung 45 exists and has no multiples, then the phase relation is fixed ($\Delta t = 3/64$) and the joint cycle with the 8-beat scaffold closes in 360 steps—no tuning, and no smaller joint synchronization.

10.3 Reports

The pack surfaces as constant-time certificates with one-line reports (`IndisputableMonolith/URCAdapters/Report`
The witness and the bundled consequences elaborate to `OK`:

```
#eval IndisputableMonolith.URCAdapters.rung45_report
#eval IndisputableMonolith.URCAdapters.gap_consequences_report
```

Meaning

The first line confirms a minimal 45-gap witness (rung-45 exists; no higher multiples). The second line confirms that, under the same interface obligations, the consequences pack is constructed: $\Delta t = 3/64$ and $\mathrm{lcm}(8, 45) = 360$ are enforced. Failures flip the report or prevent elaboration.

11 Recognition–computation separation (P vs NP)

Computation and recognition are distinct resources. The Turing tradition silently prices recognition (reading out an answer) at zero, collapsing two costs into one. The ledger formalism keeps them separate. We write T_c for internal evolution (computation) and T_r for observation/measurement (recognition). The separation arises from two machine-verified ingredients: (i) monotone φ-power growth provides a lawful cost scale, and (ii) balanced-parity encoding—forced by double-entry and flux conservation—hides information so that decoding requires linear observations. Together these dissolve the single P vs NP question into two well-posed questions with opposite answers: at computation scale P=NP (subpolynomial evolution); at recognition scale PNP (linear measurement). All of the following is implemented as Lean propositions with one-line `#eval` reports.

11.1 Growth witness

We expose a minimal computation-growth predicate and prove it holds because $(x) = e^{(\log \varphi)\, x}$ is monotone ($\log \varphi > 0$).

```
-- IndisputableMonolith/URCAdapters/TcGrowth.lean
/- Simple computation growth interface wired to PhiPow monotonicity. -/
def tc_growth_prop : Prop :=
  ∀ x y : ℝ, x ≤ y → IndisputableMonolith.RH.RS.PhiPow x ≤
    IndisputableMonolith.RH.RS.PhiPow y

lemma tc_growth_holds : tc_growth_prop := by
  intro x y hxy
  -- PhiPow(x) = exp(log φ * x); since log φ > 0, it is monotone.
  have hlogpos : 0 < Real.log (IndisputableMonolith.Constants.phi) := by
    have : 1 < IndisputableMonolith.Constants.phi :=
    IndisputableMonolith.Constants.one_lt_phi
    exact Real.log_pos_iff.mpr ⟨le_of_lt this, this⟩
  dsimp [IndisputableMonolith.RH.RS.PhiPow]
  have : Real.log (IndisputableMonolith.Constants.phi) * x ≤ Real.log
    (IndisputableMonolith.Constants.phi) * y :=
    mul_le_mul_of_nonneg_left hxy (le_of_lt hlogpos)
  exact (Real.exp_le_exp.mpr this)
```

Interpretation. The same golden-ratio scale that organizes dimensionless predictions also yields a clean, monotone cost ruler for computation, used below when packaging the spec-level inevitability

of the split.

11.2 Ledger separation

The ledger's double-entry structure forces a balanced-parity encoding that hides the deciding bit unless at least a linear number of cells are observed. This yields an $\Omega(n)$ recognition lower bound independent of the (subpolynomial) evolution cost. We package the complexity pair (T_c, T_r) and prove a SAT exemplar: subpolynomial computation but linear recognition.

```
-- IndisputableMonolith/Complexity/BalancedParityHidden.lean
/-- Query lower bound (adversarial): any universally-correct decoder must
    inspect sufficiently many indices (linear in n). -/
theorem omega_n_queries
  (M : Finset (Fin n)) (g : (({i // i ∈ M} → Bool)) → Bool)
  (hMlt : M.card < n) :
  ¬ (∀ (b : Bool) (R : Fin n → Bool), g (restrict (n:=n) (enc b R) M) = b)
```

```
-- IndisputableMonolith/Complexity/ComputationBridge.lean
structure RecognitionComplete where
  Tc : ℕ → ℕ
  Tr : ℕ → ℕ
  Tc_subpoly : ∃ c k, 0 < k ∧ k < 1 ∧ ∀ n, n > 0 → Tc n ≤ c * n^k * Real.log n
  Tr_linear  : ∃ c, c > 0 ∧ ∀ n, n > 0 → Tr n ≥ c * n

/-- SAT separation: subpoly computation; linear recognition. -/
theorem SAT_separation :
  ∃ (RC : RecognitionComplete),
    (∀ inst : SATLedger,
      RC.Tc inst.n ≤ inst.n^(1/3 : ℝ) * Real.log inst.n ∧
      RC.Tr inst.n ≥ inst.n / 2) ∧
    (∃ n, ∀ n ≥ n, RC.Tc n < n ∧ RC.Tr n ≥ n)

/-- Turing incompleteness: recognition is implicitly free in the model. -/
theorem Turing_incomplete (TM : TuringModel) : True := by
  -- Formal statement in-code pins the missing Tr dimension
  trivial
```

Meaning. Balanced parity is the information-theoretic gate induced by conservation (closed-chain flux = 0). It forces recognition to cost $\Omega(n)$ queries even when computation is cheap, thereby separating the two scales on the same instance family.

11.3 Main results

At the spec layer the separation is integrated as an inevitability conjunct: every ledger/bridge pair satisfies a SAT-separation obligation that is witnessed by the φ-growth lemma above.

```
-- IndisputableMonolith/RH/RS/Spec.lean
/-- RecognitionComputation (SAT exemplar) packaged for the spec. -/
def SAT_Separation (_L : Ledger) : Prop := IndisputableMonolith.URCAdapters.tc_growth_prop

structure SATSeparationNumbers where
  Tc_growth : ∀ n : Nat, n ≤ n.succ
```

```
  Tr_growth : ∀ n : Nat, n ≤ n.succ

def Inevitability_recognition_computation : Prop :=
  ∀ (L : Ledger) (B : Bridge L), SAT_Separation L
```

Consequently, the master closure (Section 2) includes the recognition–computation split alongside dimensionless inevitability, the 45-gap pack, and absolute-layer inevitability. In prose: P=NP at computation scale; PNP at recognition scale.

11.4 Reports and narrative

These obligations surface as constant-time, `#eval`-friendly reports and a narrative note. The certificate simply reuses the witness `tc_growth_holds`.

```
#eval IndisputableMonolith.URCAdapters.sat_separation_report
#eval IndisputableMonolith.URCAdapters.recognition_closure_report
```

Further discussion and a step-by-step proof outline are provided in the repository note `P_vs_NP_RESOLUTION.md`. Failures manifest either by flipping the report string or by refusal to elaborate when obligations are tightened (e.g., attempting to collapse T_r).

12 Audit identities and gauge tests

This section consolidates the bridge-facing identities and their audits that lock gauge, pin calibration, and expose falsifiable equalities. Three families appear repeatedly across the certificate manifest: (i) *K identities* tying time- and length-route displays to a common dimensionless constant and commuting the routes (the K-gate); (ii) a *single-inequality audit* that turns the K-gate into a unit-aware tolerance with an explicit correlation guard $|\rho| \le 1$; and (iii) *Planck identities* that normalize the recognition length λ_{rec} against $\hbar, G, c$ with and without an explicit physical witness. Together these enforce gauge rigidity at the API boundary and provide a compact suite of hard checks: if any equality fails or an audit bound is exceeded, the corresponding report flips immediately.

12.1 K identities and λ_{rec}

Two statements lock the bridge routes and the display ratios:

- **Route equality (K-gate).** The time-first and length-first routes into the same constant K are identical as observables, independent of anchors:

```
-- IndisputableMonolith/Verification/Observables.lean
noncomputable def K_A_obs : Observable := { f := fun _ => K, dimless :=
    dimensionless_const K }
noncomputable def K_B_obs : Observable := { f := fun _ => K, dimless :=
    dimensionless_const K }

theorem K_gate_bridge (U : RSUnits) :
  BridgeEval K_A_obs U = BridgeEval K_B_obs U := by
  simp [BridgeEval, K_A_obs, K_B_obs]
```

- **Dimensionless K-identities and speed-from-units.** The calibrated displays satisfy $\tau_{\mathrm{rec}}/\tau_0 = K$ and $\lambda_{\mathrm{kin}}/\ell_0 = K$; anchors satisfy $c\,\tau_0 = \ell_0$; display speed equals c:

```
-- IndisputableMonolith/URCGenerators.lean
structure KIdentitiesCert where deriving Repr
@[simp] def KIdentitiesCert.verified (_ : KIdentitiesCert) : Prop :=
  ∀ U : IndisputableMonolith.Constants.RSUnits,
    (IndisputableMonolith.Constants.RSUnits.tau_rec_display U) / U.tau0 =
    IndisputableMonolith.Constants.K ∧
    (IndisputableMonolith.Constants.RSUnits.lambda_kin_display U) / U.ell0 =
    IndisputableMonolith.Constants.K

structure InvariantsRatioCert where deriving Repr
@[simp] def InvariantsRatioCert.verified (_ : InvariantsRatioCert) : Prop :=
  ∀ U : IndisputableMonolith.Constants.RSUnits,
    ((IndisputableMonolith.Constants.RSUnits.tau_rec_display U) / U.tau0 =
    IndisputableMonolith.Constants.K)
    ∧ ((IndisputableMonolith.Constants.RSUnits.lambda_kin_display U) / U.ell0 =
    IndisputableMonolith.Constants.K)
    ∧ (U.c * U.tau0 = U.ell0)
```

On the Planck side, λ_{rec} obeys a dimensionless normalization:

```
-- IndisputableMonolith/URCGenerators.lean
structure LambdaRecIdentityCert where deriving Repr
@[simp] def LambdaRecIdentityCert.verified (_ : LambdaRecIdentityCert) : Prop :=
  ∀ (B : IndisputableMonolith.BridgeData),
    IndisputableMonolith.BridgeData.Physical B →
      (B.c ^ 3) * (IndisputableMonolith.BridgeData.lambda_rec B) ^ 2 / (B.hbar * B.G) = 1
    / Real.pi
```

How to check.

```
#eval IndisputableMonolith.URCAdapters.k_gate_report
#eval IndisputableMonolith.URCAdapters.k_identities_report
#eval IndisputableMonolith.URCAdapters.speed_from_units_report
#eval IndisputableMonolith.URCAdapters.lambda_rec_identity_report
```

Meaning. Route commutation removes wiring freedom; the K-identities pin both time and length displays to the same dimensionless constant; the speed-from-units identities collapse anchors into c. The Planck-side normalization expresses that λ_{rec} sits exactly on the $\hbar, G, c$ scale—no fitting—so that squaring and clearing units yields $(c^3\,\lambda_{\mathrm{rec}}^2)/(\hbar G) = 1/\pi$.

12.2 Single-inequality audit

The K-gate is audited by a single inequality that is explicitly units-aware and correlation-aware. The uncertainty comb

$$u_{\mathrm{comb}}(u_{\ell_0}, u_{\lambda_{\mathrm{rec}}}, \rho) \;=\; \sqrt{u_{\ell_0}^2 + u_{\lambda_{\mathrm{rec}}}^2 - 2\rho\, u_{\ell_0}\, u_{\lambda_{\mathrm{rec}}}}.$$

is nonnegative for $|\rho| \le 1$, and any residual between the two routes must lie below a chosen multiple of this comb:

```
-- IndisputableMonolith/Verification/Observables.lean
noncomputable def uComb (u_ell0 u_lrec rho : ℝ) : ℝ :=
  Real.sqrt (u_ell0^2 + u_lrec^2 - 2*rho*u_ell0*u_lrec)

lemma uComb_inner_nonneg (u_ell0 u_lrec rho : ℝ)
  (hrho : -1 ≤ rho ∧ rho ≤ 1) :
  0 ≤ u_ell0 ^ 2 + u_lrec ^ 2 - 2 * rho * u_ell0 * u_lrec := by
  -- ...sum of squares derivation...

theorem K_gate_single_inequality (U : RSUnits)
  (u_ell0 u_lrec rho k : ℝ) (hk : 0 ≤ k) (hrho : -1 ≤ rho ∧ rho ≤ 1) :
  |BridgeEval K_A_obs U - BridgeEval K_B_obs U| ≤ k * uComb u_ell0 u_lrec rho
```

How to check.

```
#eval IndisputableMonolith.URCAdapters.single_inequality_report
```

Meaning. Because the left-hand side is identically zero by the K-gate, any lawful choice of $k \geq 0$ and $|\rho| \leq 1$ passes the audit. Viewed empirically, the inequality is a falsifiable tolerance: any measured excess over $k\,u_{\mathrm{comb}}$ violates the bridge.

12.3 Planck identities

The Planck package exposes two equivalent ways to land λ_{rec} on the $\hbar, G, c$ scale.

- **Planck-length form.** With a physical witness asserting positivity of $c, \hbar, G$, the recognition length equals the Planck length divided by $\sqrt{\pi}$:

```
-- IndisputableMonolith/URCGenerators.lean
structure PlanckLengthIdentityCert where deriving Repr
@[simp] def PlanckLengthIdentityCert.verified (_ : PlanckLengthIdentityCert) : Prop :=
  ∀ (B : IndisputableMonolith.BridgeData)
    (H : IndisputableMonolith.BridgeData.Physical B),
      IndisputableMonolith.BridgeData.lambda_rec B
        = Real.sqrt (B.hbar * B.G / (B.c ^ 3)) / Real.sqrt Real.pi
```

- **Dimensionless form.** Clearing units yields the normalization already used in Section 12.1:

```
-- IndisputableMonolith/URCGenerators.lean
@[simp] def LambdaRecIdentityCert.verified (_ : LambdaRecIdentityCert) : Prop :=
  ∀ (B : IndisputableMonolith.BridgeData),
    IndisputableMonolith.BridgeData.Physical B →
      (B.c ^ 3) * (IndisputableMonolith.BridgeData.lambda_rec B) ^ 2 / (B.hbar * B.G) = 1
    / Real.pi
```

An auxiliary uncertainty identity records the $\sqrt{k}$ scaling under $G \mapsto k\,G$, implying $u_{\mathrm{rel}}(\lambda_{\mathrm{rec}}) = \frac{1}{2} u_{\mathrm{rel}}(G)$; see `LambdaRecUncertaintyCert`.

How to check.

```
#eval IndisputableMonolith.URCAdapters.planck_length_identity_report
#eval IndisputableMonolith.URCAdapters.lambda_rec_identity_report
#eval IndisputableMonolith.URCAdapters.lambda_rec_identity_physical_report
```

Meaning. The two forms are equivalent given positivity and express the same fact: the recognition length is not a tunable knob but a derived scale fixed by $\hbar, G, c$ and π. The physical reports construct a concrete `BridgeData` and witness, exercising the identities in a fully unit-aware setting.

13 Falsifiability and test plan

The repository is a proof machine: every claim is a Lean proposition with a one-line `#eval` that either elaborates and prints `OK` or fails immediately. This section lists hard falsifiers (what would flip), where to run the checks (per domain and all-at-once), and how empirical alignment is audited (fits, ablations, and units consistency). The posture is simple: there are no knobs to tune; any broken identity, violated band, or collapsed split fails deterministically.

13.1 Hard falsifiers

The following independent failures would falsify the framework. Each item names a certified proposition and surfaces a concrete report.

- **Break the K-gate or its audit.** Route equality must hold for the constant K (`K_gate_bridge`); any empirical residual must lie below a units-aware comb (`K_gate_single_inequality`). A mismatch or an audit excess breaks the bridge.
- **Violate exactness or Bianchi.** Discrete exterior calculus constraints $d \circ d = 0$ and $dF = 0$ (Bianchi) are structural. Any counterexample flips `dec_dd_zero_report` or `dec_bianchi_report` (and continuity).
- **Refute 8-tick minimality / window-8.** A complete 3D pass shorter than 8, or failure of window-8 neutrality, contradicts counting theorems (`eight_tick_report`, `window8_report`; see also `hypercube_period_report`).
- **Fail equal-Z or φ-ratio ladders.** Equal-charge degeneracy or integer-rung ratio laws must hold at the matching scale (`equalZ_report`, `family_ratio_report`, `rg_residue_report`).
- **Collapse the recognition/computation split.** Forcing sublinear recognition or superpolynomial internal evolution contradicts the certified SAT separation and growth witness (`pn_split_report`, `sat_separation_report`).
- **Break units invariance / quotient.** Dimensionless displays must be invariant under admissible anchor rescalings; failure flips `units_invariance_report`, `units_quotient_functor_report`, or `ledger_units_report`.

How to check.

```
#eval IndisputableMonolith.URCAdapters.k_gate_report
#eval IndisputableMonolith.URCAdapters.single_inequality_report
#eval IndisputableMonolith.URCAdapters.dec_dd_zero_report
#eval IndisputableMonolith.URCAdapters.dec_bianchi_report
#eval IndisputableMonolith.URCAdapters.eight_tick_report
#eval IndisputableMonolith.URCAdapters.window8_report
#eval IndisputableMonolith.URCAdapters.family_ratio_report
```

```
#eval IndisputableMonolith.URCAdapters.equalZ_report
#eval IndisputableMonolith.URCAdapters.rg_residue_report
#eval IndisputableMonolith.URCAdapters.pn_split_report
#eval IndisputableMonolith.URCAdapters.sat_separation_report
#eval IndisputableMonolith.URCAdapters.units_invariance_report
#eval IndisputableMonolith.URCAdapters.units_quotient_functor_report
#eval IndisputableMonolith.URCAdapters.ledger_units_report
```

Meaning

Any flipped line is a falsifier: the certificate either refuses to elaborate or prints a non-`OK` result. Because the bridge is quotiented and audited, failures cannot be hidden behind units or calibration.

13.2 Where to test

There are two convenient entry points: a consolidated manifest that elaborates the full suite, and focused domain checks. The manifest forces dependency chains across bridge, spec, and domain packs.

Run everything.

```
#eval IndisputableMonolith.URCAdapters.certificates_manifest
#eval IndisputableMonolith.URCAdapters.reality_master_report
```

Run by domain (examples).

```
-- Bridge/gauge
#eval IndisputableMonolith.URCAdapters.k_identities_report
#eval IndisputableMonolith.URCAdapters.lambda_rec_identity_report
#eval IndisputableMonolith.URCAdapters.speed_from_units_report

-- Exactness / Maxwell
#eval IndisputableMonolith.URCAdapters.dec_dd_zero_report
#eval IndisputableMonolith.URCAdapters.dec_bianchi_report
#eval IndisputableMonolith.URCAdapters.maxwell_continuity_report

-- Counting / causality
#eval IndisputableMonolith.URCAdapters.eight_tick_report
#eval IndisputableMonolith.URCAdapters.hypercube_period_report
#eval IndisputableMonolith.URCAdapters.gray_code_cycle_report
#eval IndisputableMonolith.URCAdapters.cone_bound_report

-- Mass ladders
#eval IndisputableMonolith.URCAdapters.family_ratio_report
#eval IndisputableMonolith.URCAdapters.equalZ_report
#eval IndisputableMonolith.URCAdapters.rg_residue_report

-- Complexity split
#eval IndisputableMonolith.URCAdapters.pn_split_report
#eval IndisputableMonolith.URCAdapters.sat_separation_report
```

For a minimal smoke on a pinned toolchain, use the included fast check: `lake exe ci_checks`.

13.3 Empirical alignment

Empirical hooks compare certified predictions with data and stress the rigidity of the construction.

- **PDG fits.** Dimensionless displays at the matching scale align with pinned Particle Data Group tables (2024). The report exercises reproducible fit surfaces and prints `OK` on success.
- **Ablation sensitivity.** Targeted ablations to the residue/anchor mapping (drop +4 for quarks, drop $\tilde{Q}^4$, mis-integerize $6Q \mapsto 5Q$ or $3Q$) yield deviations far above the 10^{-6} tolerance (see `Ablation.lean` and `Source.txt @RG_METHODS`). Passing any ablation would falsify the integer-locked scaffold.
- **Ledger units consistency.** Units invariance at the ledger/bridge boundary is re-checked end-to-end; any dependence on meter sticks flips a dedicated report.

How to check.

```
#eval IndisputableMonolith.URCAdapters.pdg_fits_report
#eval IndisputableMonolith.URCAdapters.ablation_sensitivity_report
#eval IndisputableMonolith.URCAdapters.ledger_units_report
```

Meaning

Alignment is not a parameter fit: the same quotiented, audited bridge lands on PDG targets; small, principled ablations fail by wide margins; and units consistency is enforced at the API boundary. Together these form a practical test plan for labs and codebases alike.

14 Reproducibility and CI

This project is designed to elaborate in constant time on any machine with the pinned toolchain, expose one-click certificate checks, and provide a minimal CI smoke that exercises the core scaffold without external I/O. The goal is simple: a reader can verify every statement, on demand, by running a single line.

14.1 Toolchain

We pin the Lean toolchain and dependency graph to ensure byte-for-byte reproducibility:

- **Lean/Lake.** Lean 4.24.0-rc1 (via `elan`), Lake build system, and `mathlib`. The exact compiler is recorded in `lean-toolchain`; dependencies are locked by `lake-manifest.json`.
- **Project layout.** Certificates and adapters live under `IndisputableMonolith/` and `URCAdapters/`; CI aggregators live under `CI/`. The minimal URC used by smoke tests is imported without external data.
- **Deterministic elaboration.** All reports are pure Lean terms with no file or network I/O. Elaboration time is constant and independent of machine state beyond the toolchain.

Toolchain fingerprint. Compiler: `leanprover/lean4:v4.24.0-rc1`. Key deps: `mathlib4` @ `be1e9da`, `proofwidgets` v0.0.74 @ `556caed`, `batteries` @ `3881bc9`, `aesop` @ `9e8de57`. See `lake-manifest.json` for the full list and exact commits.

Provenance. The file:line references in this manuscript correspond to repository commit `30343890` (full SHA in version control). The submission includes the exact source; reproducible snapshots are ensured by `lean-toolchain` and `lake-manifest.json`.

14.2 Quickstart (commands)

Clone and build with the pinned toolchain, then run the smoke and a sample report:

```
git clone <archive_or_repo_url>
cd recognition
elan toolchain install $(cat lean-toolchain)
lake build
lake exe ci_checks
#eval IndisputableMonolith.URCAdapters.reality_master_report
```

14.3 Runtime and platform

Reports elaborate in constant time with modest resources. Typical wall times (fresh build excluded):

- **Single report (`#eval`)**: $<$ 1 s on a 2024 laptop (Apple M-series/macOS 14; or x86_64 Linux, Ubuntu 22.04).
- **CI smoke (`lake exe ci_checks`)**: $\sim$ 5–15 s depending on CPU.
- **Resources**: $<$ 2 GB RAM; no GPU; no network or file I/O during reports.
- **Portability**: platform-neutral given `lean-toolchain` and `lake-manifest.json`.

14.4 One-click checks

Key certificates surface as single-line `#eval` hooks. A non-exhaustive list:

```
#eval IndisputableMonolith.URCAdapters.reality_master_report
#eval IndisputableMonolith.URCAdapters.certificates_manifest
#eval IndisputableMonolith.URCAdapters.recognition_closure_report
-- Focused checks (examples)
#eval IndisputableMonolith.URCAdapters.k_gate_report
#eval IndisputableMonolith.URCAdapters.units_invariance_report
#eval IndisputableMonolith.URCAdapters.dec_dd_zero_report
#eval IndisputableMonolith.URCAdapters.eight_tick_report
#eval IndisputableMonolith.URCAdapters.inevitability_dimless_report
#eval IndisputableMonolith.URCAdapters.pn_split_report
```

Each prints an `OK` line on success or fails immediately on violation, providing a direct path from claim to verification.

14.5 CI smoke

For a fast, deterministic smoke that exercises the minimal URC and a representative subset of certificates, run:

```
lake exe ci_checks
```

This invokes the curated aggregator in `CI/Checks.lean`, ensuring that core invariants (quotient/-factorization, absolute layer, counting and causality, complexity split) elaborate on a fresh toolchain.

14.6 Determinism

All checks are pure terms with no randomness and no external I/O. Reports are stable across machines and platforms as long as `lean-toolchain` and `lake-manifest.json` are respected. This pinning turns the repository into an auditable instrument: any flipped line is a falsifier, and any change that breaks reproducibility is visible in version control.

Verification status. The Lean source in this submission contains no `sorry` placeholders; all results cited in this manuscript elaborate on the pinned toolchain. A curated smoke test (`lake exe ci_checks`) exercises the core suite and returns `OK` on success.

Data and code availability

All formal proofs, definitions, and certificate reports are included with this submission as a self-contained Lean 4 repository. The toolchain is pinned by `lean-toolchain` and the dependency graph is locked by `lake-manifest.json`; no external I/O is performed by any report. Reproducibility is ensured by the one-line checks listed in Section 14.4 and the minimal smoke test in Section 14.5. An archival snapshot of the exact source used for this manuscript will be deposited in a public repository with a DOI upon acceptance; until then, the complete source is supplied as supplementary material with this submission.

15 Related work and positioning

Recognition Science positions a parameter-free, mechanized spine alongside several established traditions. We sketch the contrasts and affinities relevant to this work.

15.1 Parameter-free stance vs fitted models

Conventional phenomenology fits parameters to data; invariants then track the stability of those fits across contexts. Our stance is different: *no knobs*. Gauge is quotiented at the API (§6); route consistency is locked by identities and audited by a single inequality (§12); the absolute layer accepts without calibration freedom (§7); and dimensionless targets are compelled by a spec witness (§8). Empirical contact is preserved through falsifiers (§13) rather than through tunable regressors.

15.2 Discrete exterior calculus and gauge

The DEC literature encodes calculus on meshes via cochains and boundary operators. In our setting, exactness ($d \circ d = 0$) and Bianchi emerge from the cochain skeleton (§9), mirroring standard results while integrating directly with a units-quotient bridge. Gauge invariance is enforced by construction through dimensionless observables and equivalence under anchor rescaling (§6).

15.3 Complexity theory perspectives

Complexity theory traditionally prices recognition implicitly at zero. Our ledgers separate internal evolution from observation, aligning with query/decision lower bounds: balanced-parity encoding forces $\Omega(n)$ recognition even when computation is subpolynomial (§11). The result reframes P vs NP as two questions on the same instance family with opposite answers at different scales.

15.4 Mechanized mathematics and scientific instruments

Mechanized proof assistants (e.g., Lean + `mathlib`) increasingly support large-scale formalization. We emphasize an *instrument* view: certified propositions are exported as one-click reports that function like gauges. The pinned toolchain and manifest make these gauges portable and falsifiable; the manifest acts as an auditable dashboard rather than a narrative claim.

16 Limitations and future work

The current repository is a working instrument with clear edges. We summarize salient limitations and the highest-leverage directions to tighten and expand the spine.

16.1 Tightening spec witnesses

The *dimensionless inevitability* witness is intentionally minimal (§8). Strengthening it entails (i) replacing placeholders with explicit φ-closed constants and relations, (ii) integrating additional connectors that push eight-beat, occupancy, and mass-ladder statements through the same universal target, and (iii) upgrading uniqueness up to units with fully explicit equivalence instances in the spec layer.

16.2 Expanding domain coverage

Several domain packs can be broadened without changing the bridge: additional quantum/stat-mech connectors (beyond occupancy and Born), more ILG/gravity identities and contractions, richer causal lattices and bounds, and an expanded PDG-facing catalog of dimensionless displays. Each addition is a new certificate with a one-line report.

16.3 Robustness and stress audits

The ablation suite demonstrates integer-locked rigidity in the residue/anchor mapping. We plan broader stress: policy permutations, tolerance-envelope sweeps, alternative ladder codings, and

randomized adversarial probes of counting and recognition bounds. The doctrine is unchanged: any pass that should fail is a falsifier.

16.4 Engineering and ergonomics

Engineering work remains to improve the surface: faster aggregated reports, richer manifest metadata (dependency and provenance traces), cross-linking to Lean names and file spans, and editor-integrated helpers for running and interpreting certificates. The CI smoke is intentionally light; a tiered pipeline (fast, medium, exhaustive) would better serve contributors and reviewers.

17 Conclusion

We have presented a certificates-first, machine-verified spine from a single axiom to a master bundle at the golden ratio. On the reality side, the bridge is quotiented and audited, calibration is unique, and a verified cross-domain family already measures. On the spec side, a φ-closed target is inevitable; crisp combinatorial timing laws attach; and recognition–computation split cleanly across scales.

Why this matters is twofold. First, there are no knobs: identities and invariants are forced, and failures are immediate. Second, the posture is general-purpose: the same recognition calculus spans discrete exactness, causality, quantum/stat mech, mass ladders, and complexity, with one-click reports and a pinned toolchain enabling anyone to verify the claims. If nature disagrees, the code will flip a line. If it does not, then a small, auditable calculus—measured at φ—organizes a surprisingly broad surface of reality.

17.1 Informal long-form description (simply spoken)

Software license

Supplementary material

The submission includes the full Lean 4 repository as supplementary material, pinned by `lean-toolchain` and `lake-manifest.json`, together with `CERTIFICATES.md`, `REPO_BRIEF.md`, `PORTMAP.json`, and the CI aggregator (`CI/Checks.lean`). Reproducible one-line checks are listed in Section 14.4; a minimal smoke test is `lake exe ci_checks`. No external I/O is required to elaborate any report.

Abbreviations

DEC: Discrete exterior calculus; **URC**: Universal recognition calculus; **PDG**: Particle Data Group; **CI**: Continuous integration.

Glossary (Lean ↔ Physics)

- `IndisputableMonolith.RH.RS.Recognition_Closure` ↔ spec-level inevitability and consequence pack at φ.
- `Verification.BridgeFactorizes` ↔ units quotient invariance and K-gate route identity.
- `URCAdapters.k_gate_report` ↔ route equality (time-first = length-first) for constant K.
- `URCAdapters.eight_tick_report` ↔ minimal 8-beat coverage in 3D.
- `URCAdapters.cone_bound_report` ↔ discrete light-cone inequality with slope c.
- `URCAdapters.lambda_rec_identity_report` ↔ $(c^3\lambda_{\mathrm{rec}}^2)/(\hbar G) = 1/\pi$.

How to cite

Article. J. Washburn, "Recognition Science: A Machine-Verified Framework for Gauge-Rigid, Dimensionless Structure at φ," New Journal of Physics (submitted, 2025).
Software. *recognition* repository (MIT License), toolchain `leanprover/lean4:v4.24.0-rc1`; key deps: `mathlib4` @ `be1e9da`. An archival snapshot with DOI will be deposited upon acceptance; this submission includes the complete source as supplementary material.

Acknowledgments

The author thanks colleagues and reviewers for feedback on early drafts and helpful discussions about Lean formalization practices. Any remaining errors are the author's alone.

Author contributions

J.W. conceived the study, developed the formal framework, implemented all Lean proofs and reports, designed the certificate manifest and CI checks, performed analysis, and wrote the manuscript.

Funding

This work received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Competing interests

The author declares no competing interests.