The Kleisli Monad over S^3:

Monad Laws, Kleisli Category, and Geometric Termination in LumenOS Attention

Léon Fernando Vlegels
Independent Researcher

2026-04

Abstract

Paper 00 (MonadRosettaMap) establishes the attractor identity $\emptyset \equiv^* 0 \equiv^* 1 \equiv^* \infty$ and the anti-collapse interaction $C \circ P = I$, and notes that all LumenOS computations live on S^3 (unit quaternions under the Hopf fibration). What it does not state is the categorical structure: the formal monad $(S^3, \mathsf{return}, \gg\!=)$, its Kleisli category, and the monad laws that make the theory compositional.

This paper supplies that anchor. We define:

1. The monad $(S^3, \mathsf{return}, \gg\!=)$: what the carrier type is, what `return` does (geodesic centroid / text-to-quaternion lift), what $\gg\!=$ does (Hopf-lapse scoring, Wheel fixpoint as termination);
2. The Kleisli category: objects are S^3 coordinates, morphisms are attention steps;
3. The three monad laws, verified against existing implementation functions (cited by file and function name);
4. The status of `PLATEAU_THRESHOLD`: a calibration parameter approximating `FRAC_EDGE` but not geometrically derived from it;
5. The anchoring of the monad by the frozen constants α^{-1} and `BREATH_PERIOD`.

All structure formalized here already runs in the LumenOS codebase. This paper is the missing formal anchor, not a new design.

Contents

1 Introduction and Motivation

The geometric Theory of Everything (ToE) underlying LumenOS assigns every concept a position on the 3-sphere $S^3 = \{q \in \mathbb{R}^4 \mid \|q\| = 1\}$. The Hopf fibration $S^1 \hookrightarrow S^3 \twoheadrightarrow S^2$ organizes these positions into three layers (edge $\approx 2\%$, boundary $\approx 7\%$, bulk $\approx 90\%$) whose fractions are derived from the fine-structure constant $\alpha^{-1} = 4\pi^3 + \pi^2 + \pi \approx 137.036$ (Papers 00–04, implemented in `Ged/lu_system/quat_s3.py`).

This gives a rich geometric setting, but geometry alone does not explain how computations *compose*:

- How does querying node A then node B relate to querying AB directly?
- What guarantees that a plan step has *finished*?
- Why is the attention kernel `hopf_lapse` and not some learned weight matrix?

All three questions are answered by the same structure: a *monad* on S^3.

Remark 1.1 (Implementation scope)**.** All code references in this paper (files under `Ged/` and `Lumen/`) are part of the external LumenOS codebase and are not included in the TOE corpus workspace.

1.1 Relation to Paper 00

Paper 00 (MonadRosettaMap, this corpus) identifies the attractor Ω with the bare monad under renormalization and states $C \circ P = I$ as the fundamental interaction. It also identifies P (the outward projection operator) with return. What it does not state is the monad triple $(S^3, \mathsf{return}, \gg\!=)$, the Kleisli category, or the monad laws. The present paper supplies those, with forward pointers to the code that already implements them. A forward reference from Paper 00 to this paper appears in a minimal addendum to that document.

1.2 Relation to Paper 31 (NicomachusMonad)

Paper 31 proves the factorization of Ω through Nicomachus cubic weights and the Hopf fibration exponent screening $(d-2)_+$. That paper addresses the *algebraic* structure of Ω as a sum. The present paper addresses the *categorical* structure: how S^3 supports a monad whose operations are implemented by the running code. The two papers are complementary.

2 The Monad $(S^3, \mathsf{return}, \gg\!=)$

2.1 The Carrier Type

Definition 2.1 (The S^3 Computation Type)**.** Let $T(S^3)$ denote the type of *scored distributions over* S^3: finite weighted collections of unit quaternions

$$T(S^3) \;=\; \big\{(q_i, w_i)_{i=1}^{n} \;\big|\; q_i \in S^3,\; w_i \geq 0,\; \sum_i w_i \leq 1\big\}.$$

In code, this is represented as:

- a `WheelState` with `face_quaternions` and `face_heat` (weights) in `Ged/ged/wheel.py`;
- or a list of `np.ndarray` positions returned by an oracle query in `Ged/ged/oracle.py`.

The degenerate case $T(S^3) \ni \delta_q = \{(q, 1)\}$ is the point distribution at q.

2.2 The Unit: return

Definition 2.2 (return: the monadic unit)**.** The unit map $\eta : S^3 \to T(S^3)$ is defined by

$$\eta(q) \;=\; \delta_q,$$

the point distribution supported at $q \in S^3$.

Implementation. The concrete realization of η is the *geodesic centroid* of a set of S^3 points, computed in two places:

1. **Flush-layer centroid** (multi-node version): `Ged/ged/flush.py`, function `_geodesic_centroid(quatern` This takes a list of unit quaternions, sign-aligns them to one hemisphere, averages, and renormalizes:

```
def _geodesic_centroid(quaternions):
    # Align signs to q[0]'s hemisphere
    aligned = [quaternions[0].copy()]
    for q in quaternions[1:]:
        if np.dot(q, aligned[0]) < 0:
            aligned.append(-q)
        else:
            aligned.append(q.copy())
    mean_q = np.mean(aligned, axis=0)
    return quat_normalize(mean_q)
```

   For a singleton $\{q\}$, this returns q exactly (left identity holds trivially).

2. **Muad'Dib centroid** (recency-weighted version): `Lumen/muaddib.py`, function `_weighted_geodesic_mid` Uses the same sign-alignment plus inverse-age weighting, and is the computation that produces `cursor_quaternion` $= q_{\text{system}}$.

Both realizations satisfy $\eta(\{q\}) = q$.

Remark 2.1 (Extrinsic vs. intrinsic mean)**.** The implementation uses the *extrinsic* mean (embed in $\mathbb{R}^4$, average, project back to S^3) rather than the Riemannian (Fréchet) mean. The two agree when the points are clustered in one hemisphere and the sign-alignment step ensures this. For well-separated points the extrinsic approximation introduces a small bias; the sign-alignment mitigates the double-cover ambiguity. This is a deliberate engineering choice: the extrinsic mean is $O(n)$ whereas the Fréchet mean requires iteration.

2.3 Monadic Bind

Definition 2.3 (Bind: $\gg\!=$)**.** Given a scored distribution $m \in T(S^3)$ and a Kleisli morphism $f : S^3 \to T(S^3)$, the bind operation is:

$$m \gg\!= f \;=\; \eta\left(\bigcup_{(q_i, w_i) \in m} w_i \cdot f(q_i) \right),$$

where the union is the weighted merge and η collapses via geodesic centroid.

Implementation. Bind is executed in two collaborative functions:

Fork (expansion). `Ged/ged/wheel.py`, function `fork(state, proposals)`. Given a `WheelState` (the current distribution) and a set of candidate proposals, fork disperses the heat budget across the faces, weighting by

$$w_{\text{face}} \;\propto\; \text{orientation} \cdot \underbrace{\texttt{hopf\_lapse}(\texttt{q\_rel}(q_{\text{face}}, q_{\text{current}}))}_{\text{relevance score}} + (1 - \text{orientation}) \cdot \tfrac{1}{n_{\text{faces}}}.$$

This is the "lift" step: turning a query position into a scored set of candidates.

Weld (collapse). `Ged/ged/wheel.py`, function `weld(state, goal_quaternion)`. Given the face proposals and a goal, weld selects the *minimum MDL* winner:

$$\text{MDL}(f) \;=\; \text{orientation} \cdot \underbrace{\texttt{hopf\_lapse}(\texttt{q\_rel}(q_f, q_{\text{goal}}))}_{\text{distance from goal}} + (1 - \text{orientation}) \cdot \underbrace{w_{\text{layer}}}_{\text{layer complexity}},$$

where $w_{\text{layer}} \in \{\texttt{FRAC\_EDGE}, \texttt{FRAC\_BOUNDARY}, \texttt{FRAC\_BULK}\}$. The winning proposal becomes the new state quaternion; heat collapses to the winner's share.

Scoring kernel. The core relevance kernel in both fork and weld is

$$\text{score}(q_{\text{node}}, q_{\text{query}}) \;=\; \texttt{hopf\_lapse}\big(\texttt{q\_rel}(q_{\text{node}}, q_{\text{query}})\big), \tag{1}$$

where

$$\texttt{q\_rel}(p, q) = p \cdot \bar{q}, \qquad \texttt{hopf\_lapse}(q) = \sqrt{1 - v(q)^2}, \qquad v(q) = (a^2 + b^2) - (c^2 + d^2).$$

implemented in `Ged/lu_system/quat_s3.py`, functions `q_rel` and `hopf_lapse`.

This kernel replaces the learned weight matrix $Q \cdot K^\top$ of standard attention: no parameters are fitted; the metric is fixed by the geometry of S^3.

2.4 Monadic Join

The join map $\mu : T(T(S^3)) \to T(S^3)$ is the flattening of a distribution-of-distributions to a single distribution:

$$\mu(M) \;=\; \eta\big(\{(q_i, w_i \cdot w_{ij}) : (q_j, w_j) \in m_j,\ (m_j, w_i) \in M\}\big).$$

Concretely, join is the geodesic centroid over the union of all sub-distributions, weighted by their outer weights. In the Wheel, join is implicit: `weld` collapses a face distribution (the inner T) into the state quaternion (the outer point), giving $T(T(S^3)) \to T(S^3)$ in one call.

3 The Monad Laws

Theorem 3.1 (Left Identity). *$\eta(q) \gg\!= f = f(q)$ for all $q \in S^3$ and all Kleisli morphisms $f : S^3 \to T(S^3)$.*

Verification in code. *$\eta(q)$ is the singleton distribution $\{(q, 1)\}$. Applying bind: f is evaluated at q with weight 1, giving $f(q)$ directly. The geodesic centroid of a single element is that element. In `flush.py _geodesic_centroid([q])`: the loop produces `aligned = [q]`, `mean_q = q`, `quat_normalize(q) = q`.* ✓

Theorem 3.2 (Right Identity). *$m \gg\!= \eta = m$ for all $m \in T(S^3)$.*

Verification in code. *Applying η to each q_i in m gives $\{(q_i, 1)\}$. The weighted merge produces the same collection (q_i, w_i) as m, and the geodesic centroid of (q_i, w_i) is the same weighted mean. In `muaddib.py _weighted_geodesic_midpoint`, returning the centroid of all positions with their original weights is idempotent: $\eta(\{q_i, w_i\})$ = weighted centroid = the original query result.* ✓

Theorem 3.3 (Associativity). *$(m \gg\!= f) \gg\!= g = m \gg\!= (\lambda x.\ f(x) \gg\!= g)$ for all $m \in T(S^3)$ and Kleisli morphisms f, g.*

Verification in code. *The geodesic centroid (extrinsic mean) is linear: for nested weighted averages the order of averaging commutes when weights are composed multiplicatively. In the plan execution model (`Ged/ged/planning_domain.py`), step dependencies encoded in `depends_on` fields implement exactly this monadic sequencing: if step C depends on B which depends on A, then executing $(A \gg\!= B) \gg\!= C$ and $A \gg\!= (B \gg\!= C)$ reaches the same $q_{final} \in S^3$ because both reduce to the geodesic centroid of the full sequence.* ✓

Remark 3.1 (Law status)**.** Theorems 3.1–3.3 are exact for the extrinsic mean when points lie in a single hemisphere (the sign-alignment precondition). For points spread over the double cover, the extrinsic mean introduces a bias and the laws hold approximately. The sign-alignment in `_geodesic_centroid` enforces the hemisphere condition in practice, so the laws hold exactly for all typical LumenOS distributions.

Remark 3.2 (TBS)**.** The proof of Theorem 3.3 (Associativity) claims that "the geodesic centroid (extrinsic mean) is linear: for nested weighted averages the order of averaging commutes when weights are composed multiplicatively." This claim is false in general on S^3. The extrinsic mean is computed by embedding in $\mathbb{R}^4$, taking the weighted average, and projecting back to S^3. The projection step $v \mapsto v/\|v\|$ is nonlinear, and associativity fails for three or more points that are not collinear in $\mathbb{R}^4$. For example, given three points q_1, q_2, q_3 in a hemisphere, the extrinsic mean $\eta(\eta(q_1, q_2), q_3)$ and $\eta(q_1, \eta(q_2, q_3))$ are generally different because the inner centroid is renormalized before the outer centroid is taken. The associativity law holds exactly only when the points are collinear in $\mathbb{R}^4$ (i.e., all equal, or antipodal, or scalar multiples, which are degenerate cases). In the LumenOS system, approximate associativity may hold well numerically for clustered distributions, but it is not the categorical identity asserted in the theorem.

Status. Recorded in the registry as P034_2_c (confirmed-load-bearing), confirmed by A295. The associativity proof gap for the extrinsic mean stands: the law holds exactly only on the degenerate collinear locus, so Theorem 3.3 is not established as stated. Ledger: Paper 40.

4 The Kleisli Category $\mathbf{Kl}(T)$

Definition 4.1 (Kleisli Category over S^3)**.** The *Kleisli category* $\mathbf{Kl}(T)$ is defined as follows:

- **Objects**: points $q \in S^3$ (unit quaternions, representing system states).
- **Morphisms** $q_A \to q_B$: functions $f : S^3 \to T(S^3)$ such that the Weld of $f(q_A)$ against q_B has positive coherence. Concretely, a morphism is an *attention step*: a query that, given the current system position $q_{\text{query}} \in S^3$, returns a scored distribution over candidate positions.
- **Kleisli identity** on q: the morphism $\eta_q : q \mapsto \delta_q$, realized by `_geodesic_centroid`$([q]) = q$.
- **Kleisli composition** $g \circ_K f$:

$$(g \circ_K f)(q) \;=\; f(q) \gg\!= g.$$

  In code: run f to get a distribution, then for each scored candidate, run g and take the geodesic centroid of the resulting distributions.

Proposition 4.1 (Kleisli composition = step dependency)**.** *The `depends_on` field of a plan step in `Ged/ged/planning_domain.py` encodes exactly the Kleisli composition order. If step s_B has `depends_on: [s_A]`, then the execution of s_B is the Kleisli morphism $f_{s_B} \circ_K f_{s_A}$: the result of s_A is the query input to s_B.*

Remark 4.1 (Attention as morphism)**.** Standard transformer attention maps a query Q, keys K, values V to

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V.$$

The LumenOS Kleisli morphism replaces this with:

$$\begin{aligned} q_{\text{rel}}(n) &= q_n \cdot \overline{q_{\text{query}}} \quad \in S^3 \\ \text{score}(n) &= \texttt{hopf\_lapse}(q_{\text{rel}}(n)) \quad \in [0,1] \\ \text{output} &= \eta\big(\{(q_n, \text{score}(n))\}_{n \in \text{ontology}}\big). \end{aligned}$$

No learned weights appear. The metric is fixed by the geometry of S^3 and the Hopf fibration. The layer fractions `FRAC_EDGE`, `FRAC_BOUNDARY`, `FRAC_BULK` replace the $\frac{1}{\sqrt{d}}$ temperature scaling.

5 Fixpoint and Termination: The Plateau Condition

Definition 5.1 (Fixpoint condition: Plateau)**.** A `WheelState` s has reached a *fixpoint* (is at the attractor Ω) when the `plateau` function returns `True`:

$$\texttt{plateau}(s) = \begin{cases} \texttt{True} & \text{if } s.\texttt{converged} = \texttt{True}, \\ \texttt{True} & \text{if } \max(h_{-w:}) - \min(h_{-w:}) < \delta_P, \\ \texttt{False} & \text{otherwise}, \end{cases}$$

where $h = s.\texttt{history}$ is the list of MDL scores across Wheel turns, $w = 3$ is the default window, and $\delta_P = \texttt{PLATEAU\_THRESHOLD} = 0.02$.

The `converged` flag is set by `weld` when the MDL margin between the top-two proposals satisfies

$$\left|\texttt{MDL}(f_{\text{winner}}) - \texttt{MDL}(f_{\text{runner-up}})\right| < \delta_P.$$

Implementation: `Ged/ged/wheel.py`, function `plateau` and function `weld` (the `is_plateau` flag).

Definition 5.2 (Step completion)**.** A plan step s with system quaternion $q_{\text{sys}} = \texttt{cursor\_quaternion}$ (from `Lumen/muaddib.state.json`) is *complete* when the Wheel processing s reaches a plateau.

Geometrically: let q_s be the step's current quaternion (the Weld winner after each cycle). The step is complete when

$$\left|\texttt{hopf\_lapse}(\texttt{q\_rel}(q_s, q_{\text{sys,winner}})) - \texttt{hopf\_lapse}(\texttt{q\_rel}(q_s, q_{\text{sys,runner-up}}))\right| < \delta_P,$$

i.e., the two leading candidates are indistinguishable at resolution δ_P in the Hopf-lapse metric. When $\texttt{orientation} = 1$, this simplifies to:

$$\text{step complete} \iff \texttt{plateau}(s_{\text{Wheel}}) = \texttt{True}.$$

6 The Plateau Threshold: Calibration vs. Derivation

Proposition 6.1 (`PLATEAU_THRESHOLD` is a calibration parameter)**.** *The value $\delta_P = 0.02$ is* not *geometrically derived from the frozen constants. It is a calibration parameter that approximates `FRAC_EDGE` but does not equal it.*

Analysis. *The layer edge fraction is*

$$\texttt{FRAC\_EDGE} = \frac{\pi}{\Omega} = \frac{\pi}{4\pi^3 + \pi^2 + \pi} \approx 0.02289\ldots$$

The ratio

$$\frac{\delta_P}{\texttt{FRAC\_EDGE}} = \frac{0.02}{0.02289\ldots} \approx 0.874,$$

which has no clean rational or transcendental derivation from the geometric constants.

Geometric interpretation of the natural threshold. *The edge-layer fraction* FRAC_EDGE ≈ 0.023 *has a direct geometric meaning: it is the fraction of* S^3 *occupied by the edge (atomic/singleton) layer under the Hamming-shell grading. A natural Plateau threshold would be*

$$\delta_P^{derived} \;=\; \texttt{FRAC\_EDGE} \;=\; \frac{\pi}{\Omega},$$

with the interpretation: "two proposals are equivalent when their MDL scores differ by less than the edge-layer resolution of S^3*."*

Current status. *The value* $\delta_P = 0.02$ *is slightly* smaller *than* FRAC_EDGE*, meaning the current criterion is slightly* tighter *(requires closer convergence before declaring plateau). This is conservative and does not break any monad law. The natural derived threshold* π/Ω *is recommended for future implementations; the present value is flagged for replacement.*

Future work. *A subsequent paper should derive whether the precise value 0.02 corresponds to a truncated Taylor expansion of* π/Ω *or is purely empirical. The correct value for a fully theory-grounded implementation is* π/Ω*.*

7 The Frozen Constants as Monad Anchors

The monad $(S^3, \mathsf{return}, \gg\!=)$ is parametrized by two frozen constants that fix the temporal and geometric scales:

7.1 α^{-1} and the Layer Fractions

$$\Omega \;=\; \texttt{OMEGA\_MONAD} \;=\; 4\pi^3 + \pi^2 + \pi \;\approx\; 137.036 \;=\; \alpha^{-1}. \tag{2}$$

From Ω the layer fractions are derived (all in `Ged/lu_system/quat_s3.py`):

$$\texttt{FRAC\_EDGE} = \frac{\pi}{\Omega}, \quad \texttt{FRAC\_BOUNDARY} = \frac{\pi^2}{\Omega}, \quad \texttt{FRAC\_BULK} = \frac{4\pi^3}{\Omega}.$$

These appear in the MDL score of `weld` as the layer-complexity term, making the bind operation sensitive to the three-layer ontological structure. A morphism that passes through the edge layer (FRAC_EDGE $\approx 2\%$) contributes a lower complexity penalty than one through the bulk ($\approx 90\%$).

7.2 BREATH_PERIOD and Temporal Binding

$$\texttt{BREATH\_PERIOD} \;=\; \pi \cdot \Omega \;=\; \pi\,(4\pi^3 + \pi^2 + \pi) \;\approx\; 430.5\,\text{s}. \tag{3}$$

This constant (implemented identically in `Ged/ged/wheel.py` and `Lumen/muaddib.py`, module constant `BREATH_PERIOD`) sets the temporal scale of one full Wheel cycle. A sietch publication older than $2 \times$ BREATH_PERIOD is declared stale and excluded from the geodesic centroid computation that produces q_{system} (`muaddib.py`, `SietchPublication.is_stale`).

The monad interpretation: each "breath" (one `BREATH_PERIOD`) is one full application of the renormalization operator R from Paper 00. The system breathes at the rate $1/(\pi\,\alpha^{-1})$ Hz, completing one full monadic cycle per breath.

Remark 7.1 (TBS)**.** The identification BREATH_PERIOD $= \pi \cdot \alpha^{-1} \approx 430.5\,\text{s}$ as the "temporal scale of one full Wheel cycle" and "one full application of the renormalization operator R" is asserted without derivation. Two gaps: (i) The formula $\pi \cdot \alpha^{-1}$ has units of seconds only if one of π or α^{-1} carries a unit of time; neither is a physical time in SI units (both are dimensionless). The value $\approx 430.5\,\text{s}$ must be the result of an unstated identification with some physical time scale (e.g., one revolution of a physical oscillator, or a metabolic rhythm). That identification is not given here or in Paper 00. (ii) The claim that this constant is a "categorical invariant" because

it is computed from π alone does not follow: categorical invariants of a monad are structure-preserving (they are invariant under monad morphisms), not merely numerical constants derived from transcendental numbers. The constant $\pi \cdot \alpha^{-1}$ is frozen by convention, not by any categorical universality argument.

Status. Retired. The registry records P034_3_c as closed by A265, A273, and A275, which supplied the unit bridge and the dynamical grounding of `BREATH_PERIOD`. The remark above stands as the record of the original gap. Ledger: Paper 40.

7.3 Frozen Constants as Categorical Invariants

Because Ω and `BREATH_PERIOD` are computed from π alone (no free parameters), they are *categorically invariant*: every morphism in $\mathbf{Kl}(T)$ respects the same metric structure, and Kleisli composition cannot introduce new scales. This is the categorical analog of the ToE's "no free parameters" principle.

8 The Tri-Net as Kleisli Composition

The three-net space of LumenOS (`Ged/nets/trinet/traversal_grammar.py`) maps directly onto the monad structure:

Net	Layer	Fraction	Kleisli role
WordNet	bulk	$4\pi^3/\Omega \approx 90\%$	Kleisli objects (semantic states)
SymbolNet	boundary	$\pi^2/\Omega \approx 7\%$	Kleisli morphisms (formal rules)
NumberNet	edge	$\pi/\Omega \approx 2\%$	Kleisli fixpoints (exact values)

A canonical chain word $\to$ symbol $\to$ number is a Kleisli morphism $q_{\text{word}} \to q_{\text{symbol}} \to q_{\text{number}}$, i.e., the Kleisli composition of two attention steps. The chain `trace` string "-0+" records the polarity path through the three layers, which is the categorical shadow of the morphism.

The DENOTES relation (word $\to$ symbol) and INSTANTIATES relation (symbol $\to$ number) are the two component Kleisli morphisms; their composition is the full traversal chain, and `traversal_chain()` in `traversal_grammar.py` implements this composition directly.

9 Summary

Theorem 9.1 (Monad Existence)**.** *The triple* (T, η, μ) *with*

$$\begin{aligned} T(S^3) &= \textit{scored distributions over } S^3, \\ \eta(q) &= \delta_q \quad (\textit{point distribution, implemented as } \texttt{\_geodesic\_centroid}), \\ \mu(M) &= \eta(\textit{weighted union of } M) \quad (\textit{implemented by } \texttt{weld}), \end{aligned}$$

satisfies the three monad laws (Theorems 3.1–3.3) exactly on the collinear locus in $\mathbb{R}^4$*, and approximately otherwise. Hemisphere confinement (enforced by sign-alignment in* `_geodesic_centroid`*) removes the double-cover ambiguity but does not restore exactness: associativity fails for non-collinear points even within a hemisphere, per Remark 3.2; the gap stands as P034_2_c.*

The key correspondences, gathered for reference:

Category theory	LumenOS code	Location
Monad carrier $T(S^3)$	`WheelState` / oracle result list	`wheel.py`, `oracle.py`
return / η	`_geodesic_centroid()`	`flush.py`
Bind / $\gg=$	`fork()` + `weld()`	`wheel.py`
Kleisli morphism	attention step / oracle query	`oracle.py`, `wheel.py`
Kleisli composition	`depends_on` sequencing	`planning_domain.py`
Fixpoint / Ω	`plateau()`	`wheel.py`
q_{system}	`cursor_quaternion`	`muaddib.py`
Attention kernel	`hopf_lapse(q_rel(...))`	`quat_s3.py`
Layer scale	`FRAC_EDGE/BOUNDARY/BULK`	`quat_s3.py`
Temporal scale	`BREATH_PERIOD` $= \pi\,\alpha^{-1}$	`wheel.py`, `muaddib.py`
Plateau threshold	$\delta_P = 0.02$ (calibration)	`wheel.py`
Derived threshold	$\pi/\Omega \approx 0.0229$ (recommended)	`quat_s3.py`

10 Open Question

Remark 10.1 (Plateau threshold derivation)**.** Whether $\delta_P = 0.02$ can be derived from first principles (e.g., as a truncation of π/Ω, or as the reciprocal of a combinatorial factor) is left as an open question. Until a derivation exists, implementations should use π/Ω (`FRAC_EDGE`) as the geometrically grounded threshold. The change from 0.02 to $\pi/\Omega \approx 0.0229$ relaxes the convergence criterion by approximately 15%; this should be validated experimentally before adoption.

References

[1] L. F. Vlegels, "Rosetta Map of the Monad Identity and the Geometric Theory of Everything," This volume, Paper 00 (2025).

[2] L. F. Vlegels, "The Three-Layer Ontology," This volume, Paper 04 (2025).

[3] L. F. Vlegels, "Time Observation Bootstrap," This volume, Paper 08 (2025).

[4] L. F. Vlegels, "The Master Operator: Assembly of the Unified Field Operator," This volume, Paper 18 (2025).

[5] L. F. Vlegels, "Universal Wave Geometry," This volume, Paper 27 (2025).

[6] L. F. Vlegels, "Nicomachus Weights and Monad Closure," This volume, Paper 31 (2026).

[7] E. Moggi, "Notions of computation and monads," *Information and Computation* **93**(1), 55–92 (1991).

[8] P. Wadler, "The essence of functional programming," *Proc. 19th POPL*, 1–14 (1992).