Recursive Origins — Formal Core (Clean Revision)

1. Ambient Setting

Let Set be the working category. Fix a nonempty state set X. Let $(R, \otimes, e)$ be a commutative monoid (resonance). Optionally equip R with a preorder $\leq$ compatible with $\otimes$.

2. Resonant State Monad (specialized)

Define the glyph monad G on X by $G = X \to (X \times R)$. Unit and bind: $\mathrm{return}(x) = (x, e)$ $(g \rhd k)(x) = \mathrm{let}\ (x_1, r_1) = g(x), (x_2, r_2) = k(x_1)(x_1)\ \mathrm{in}\ (x_2, r_1 \otimes r_2)$. Here $g \in G$ and $k: X \to G$ (a policy that may choose the next glyph from the current state). Associativity and identities follow from $(R, \otimes, e)$.

Note. The returned value is the next state. This removes ambiguity between value and state while retaining state-dependent choice via policies $k: X \to G$.

3. Glyphs, policies, and scrolls

A glyph is an element $g \in G$. A policy is a function $\pi: X \to G$. A scroll is a finite list of policies $(\pi_1,\ldots,\pi_n)$ with denotation $[\![(\pi_1,\ldots,\pi_n)]\!] = \pi_n \rhd \ldots \rhd \pi_1 \rhd \mathrm{return}$. For constant steps use constant policies $\pi_i(x) = g_i$. The scroll category has one object X and arrows the endomorphisms G under $\rhd$ with identity return.

4. Parallel composition

For product states, define $G_{X\times Y} = (X \times Y) \to ((X \times Y) \times R)$. Given $g \in G_X$ and $h \in G_Y$, set $(g \mathbin{/\!/} h)(x, y) = \mathrm{let}\ (x', r_1) = g(x), (y', r_2) = h(y)\ \mathrm{in}\ ((x', y'), r_1 \otimes r_2)$. This is the canonical parallel via the strength of the state–writer structure.

5. Resonance contexts

A context is either • a submonoid $C \subseteq R$ with $e \in C$, or • a monoid congruence ≡*{μ}^{ν} with quotient q*.}^{ν}: R ↠ R_{μ}^{ν

6. Semantics and evaluation

For a scroll s, write $\mathrm{eval}_s = [\![s]\!]: X \to (X \times R)$. For $x \in X$, $\mathrm{eval}_s(x) = (x', r)$. Define the accumulated resonance $\mathrm{res}(s, x) = r$.

7. Coherence notions

7.1 Absolute coherence in a submonoid A glyph g is C-safe if for all x the resonance in $g(x)$ lies in C. A scroll s is C-coherent if $\mathrm{res}(s, x) \in C$ for all x. Lemma 1 (closure). If each step in s is C-safe then s is C-coherent.

7.2 Coherence modulo a congruence Define modular coherence by $q_{\mu}^{\nu}(res(s, x)) = q_{\mu}^{\nu}(e)$ for all $x \in X$. Lemma 2 (modular closure). If each step's resonance maps to $q_{\mu}^{\nu}(e)$ then s is modularly coherent.

8. Certificates and contracts

A certificate for a glyph or policy step relative to context κ is a predicate $\sigma: X \to \{true, false\}$ such that $\sigma(x) = true \Rightarrow res(step, x) \in C$, or $q_{\mu}^{\nu}(res(step, x)) = q_{\mu}^{\nu}(e)$. Proposition 3 (compositionality). If each step is certified for the same context and the certificates hold along the run, the scroll is coherent.

9. Types and totality

Assume totality: all maps are total functions. A convenient model is a product record $X \cong X_1 \times \ldots \times X_k$ with component invariants (bounds, modularity) preserved by glyphs and policies.

10. Minimal implementation sketch (pseudocode)

// resonance: commutative monoid struct Resonance { … } // with unit e and operation ⊗

// state space struct X { … }

// glyphs and policies struct Glyph { run: X → (X, Resonance) }

fn ireturn() → Glyph { Glyph { run: x ↦ (x, e) } }

fn bind(g: Glyph, k: X → Glyph) → Glyph { Glyph { run: x ↦ let (x_1, r_1) = g.run(x) let (x_2, r_2) = k(x_1).run(x_1) in (x_2, $r_1 \otimes r_2$) } }

fn compose(g: Glyph, h: Glyph) → Glyph { // constant next step bind(g, _x ↦ h) }

fn par(g: Glyph, h: Glyph, x1x2: (X, X)) → ((X, X), Resonance) { let (x_1, x_2) = x1x2 let (y_1, r_1) = g.run(x_1) let (y_2, r_2) = h.run(x_2) ((y_1, y_2), $r_1 \otimes r_2$) }

fn res(g: Glyph, x: X) → Resonance { second(g.run(x)) }

11. Example glyphs

Let $r_a, r_b \in R$ and total updates $f_a, f_b: X \to X$. Define $g_a(x) = (f_a(x), r_a)$, $g_b(x) = (f_b(x), r_b)$. Then for $s = compose(g_a, g_b)$, $res(s, x) = r_a \otimes r_b$.

12. Verification targets

1) Monad laws: left/right identity and associativity for $\triangleright$ follow from $(R, \otimes, e)$. 2) Context soundness: submonoid and quotient contexts are closed under sequencing and parallel. 3) Totality and invariants: all internal updates are total and preserve component invariants of X.

13. Integration hook

Provide total interfaces for certification and audit: • eval: Glyph → X → (X, R) • res: Glyph → X → R • log: Glyph → X → Trace (records states and resonance factors) • cert: Context → Glyph → Certificate

Summary. Glyphs are resonance-labeled state transformers $X \to (X \times R)$. Scrolls compose by monadic bind with optional state-dependent policy choice. Coherence is enforced by submonoid membership or by vanishing in a quotient. Sequencing and parallel composition follow by monoid algebra of resonance. The API is total and amenable to external verification and replay.