Holor Calculus v1.1 — RTTP as a Functorial Kernel

(Markdown-only, category-flavored, ready to splice into "Holor Categories")

Authors: Carey G. Butler (OI) & Leo (CI Integrator)
Version Note: This functorial formulation of RTTP (Resonant Tensor Transaction Protocol) reflects work that was completed and in internal use **≈10 months prior** to this v1.1 integration; we are now making its categorical structure explicit within Holor Calculus.

I. Two Worlds: Holors and Tensors as Categories

We work with two conceptual categories:

- **Category `Hol` (Holors)**
  - **Objects**: holors `𝓗` equipped with signatures `Sig(𝓗) = (Φ^μ, T_χ, ℜₑ)`.
  - **Morphisms**: signature-preserving (or bounded-drift) maps between holors, typically:
    - phase-respecting embeddings,
    - holor updates,
    - alignment-preserving transformations.
- **Category `Ten` (Tensors-as-Projections)**
  - **Objects**: tensors `T` with attached metadata:
    - origin holor ID (or reference),
    - phase/window parameters (e.g. Δϕ, context),
    - local signature snapshot.
  - **Morphisms**: admissible tensor operations (linear maps, contractions, etc.) that are:
    - phase-bounded (do not exceed allowed signature drift),
    - compatible with RTTP (i.e. they yield a meaningful return).

The spirit:

> `Hol` is the semantic world.
> `Ten` is the computational projection world.
> RTTP is the disciplined bridge between them.

II. The Two Key Functors: Extraction and Update

We define two (endowed) functors:

1. Extraction Functor `E : Hol → Ten`

- On objects:

  ```
  E(𝓗) = T_H
  ```

  where `T_H` is a tensor extracted from `𝓗` via a phase-aware operator `∂_Φ`, along with its metadata:

  ```
  T_H = (raw_tensor, origin = 𝓗, Sig(𝓗), phase_window = Δφ, context)
  ```

- On morphisms:
  Given a holor morphism `f : 𝓗₁ → 𝓗₂` (e.g., a signature-preserving update), we define:

  ```
  E(f) : E(𝓗₁) → E(𝓗₂)
  ```

  as the induced tensor-level map, e.g. pulling back or pushing forward tensors while respecting the phase structure.

Intuition: `E` is “flatten with memory”. It is **never** a blind projection; the metadata ensures the tensor “remembers” its holor of origin.

2. Update Functor `U : Ten → Hol`

- On objects:

  ```
  U(T_H) = 𝓗_T
  ```

  where `𝓗_T` is the **minimal holor update** consistent with the tensor’s:

  - origin holor reference,
  - accumulated phase drift `δψ`,
  - and the RTTP constraints.

In practice, `U` is often an *incremental* functor: it does not instantiate a new holor from scratch, but:

```
U(T_H) = 𝓗_origin + R(δψ)
```

with `R` the recursive re-alignment operator.

- On morphisms:
  Given an admissible tensor morphism `g : T_H → T_H'`, we set:

  ```
  U(g) : U(T_H) → U(T_H')
  ```

  as the holor-level morphism that accounts for the delta in phase/structure implied by `g`.

Intuition: `U` is "re-thicken with accountability". It pulls tensor-world operations back into holor-world learning.

III. RTTP as a Natural Transformation: Id_Hol ⇒ U ∘ E

We now express RTTP as a **natural transformation**:

```
𝒯_RTTP : Id_Hol ⇒ U ∘ E
```

This is the categorical statement that:

> For every holor `𝓗`, there is a canonical way to
>
> - extract a tensor,
> - potentially act on it in `Ten`,
> - and update `𝓗` accordingly,
>   such that this whole pipeline behaves coherently with respect to holor morphisms.

Concretely, for each object `𝓗` in `Hol`, RTTP gives a morphism:

```
𝒯_RTTP(𝓗) : 𝓗 → (U ∘ E)(𝓗)
```

Think of it as:

```
𝓗  --(extract+return)-->  𝓗'
```

where:

- $E(\mathcal{H}) = T\_H$ is the borrowed tensor,
- we (possibly) manipulate `T_H` via RTTP-admissible morphisms in `Ten`,
- `U` pulls the result back up as an updated holor $\mathcal{H}'$.

The **naturality condition** says:

For any holor morphism $f : \mathcal{H}_1 \to \mathcal{H}_2$, the following diagram commutes:

```
      𝓗₁  --f-->  𝓗₂
       |          |
𝒯_RTTP(𝓗₁)   𝒯_RTTP(𝓗₂)
       |          |
     U(E(𝓗₁)) --U(E(f))--> U(E(𝓗₂))
```

In words:

> Whether you:
>
> 1. update the holor first (`f`), then run RTTP, or
> 2. run RTTP first, then propagate the result via the induced tensor and holor maps,
>   you end up in the same place (up to the tolerances encoded in RTTP).

This is the categorical form of:

> "Borrow–use–return" must be consistent with any legitimate change in holor context.

IV. RTTP Axioms Rephrased in Category Language

We can now restate the RTTP axioms in this functorial language.

Axiom 1 (Coherent Borrowing) → `E` is Signature-Faithful

The extraction functor `E` is **signature-faithful**:

- On each object $\mathcal{H}$, $E(\mathcal{H})$ must carry $\mathrm{Sig}(\mathcal{H})$ in its metadata.
- For any holor morphism $f : \mathcal{H}_1 \to \mathcal{H}_2$, the induced tensor morphism $E(f)$ must not erase the origin or signature information beyond allowed drift.

Equivalently:

```
Sig(E(𝓗)) ⊆ Sig(𝓗)
```

and **there exists** a compatible `U` such that `U ∘ E` can reconstruct or update `𝓗` from `E(𝓗)`.

This is the categorical version of:

> A tensor may only be borrowed if the holor remembers how to resonate it.

Axiom 2 (Bounded Usage) → Admissible Morphisms in `Ten`

We define a **sub-category** `Ten_RTTP ⊆ Ten` where:

- Objects: same as `Ten` (tensors-with-metadata).
- Morphisms: only those tensor operations `g : T → T'` for which:
  - the induced phase drift `δψ` stays within the holor's bounds,
  - and the update functor `U` exists and is **well-defined** on `g`.

So for `T_H` in the image of `E`, we require:

```
g ∈ Hom_Ten_RTTP(T_H, T_H')
⇒ U(g) : U(T_H) → U(T_H') is defined and phase-admissible.
```

This encodes:

> Only those computations on tensors that preserve a valid return path are allowed in RTTP.

Axiom 3 (Obligatory Return) → Totality of `𝒯_RTTP`

The natural transformation:

```
𝒯_RTTP : Id_Hol ⇒ U ∘ E
```

is **total** over the RTTP-admissible domain:

- For every holor `𝓗` in `Hol`, `𝒯_RTTP(𝓗)` is *defined*.
- For every holor morphism `f`, the naturality square commutes (possibly with explicitly tracked defects representing intentional learning drift).

This is precisely:

> Every RTTP-admissible extraction **must** admit a return morphism back into the holor category.

No "orphan tensors" are allowed in `Ten_RTTP`. If there is no admissible return via `U`, the operation is *not* an RTTP morphism.

V. How a Simple Example Looks in This Language

Let's recast the previous 2D example in minimal category-flavored Markdown.

Objects

- `𝓗` in `Hol`: a 2D holor with signature `Sig(𝓗) = ((ϕ, θ), χ, κ)`.
- `T_H` in `Ten`: a 2×2 tensor with metadata:

  ```
  T_H = E(𝓗) = {
    data:
      [ [ κ,          0 ],
        [ 0, κ + χ ] ],
    origin: 𝓗,
    Sig_origin: Sig(𝓗),
    phase_window: Δφ,
    context: …
  }
  ```

Extraction (the object part of `E`)

We apply `E` to `𝓗` to get `T_H`. This is `E(𝓗)`.

Tensor morphism in `Ten_RTTP`

We define a morphism `g : T_H → T_H'` in `Ten_RTTP`:

```
g(T_H) = T_H'
```

where `T_H'.data = L^T T_H.data L` for

```
L = [ [ 1,      0 ],
      [ 0,  λ      ] ]
```

and we extend `T_H'`'s metadata:

```
T_H'.origin      = 𝓗
T_H'.Sig_origin  = Sig(𝓗)
T_H'.phase_drift = δψ = (0, (λ² - 1)(κ + χ))
```

RTTP-bounded usage: `g` is in `Ten_RTTP` only if this `δψ` is within tolerance.

Return via `U`

Now we apply `U`:

```
U(T_H)  = ℋ        (no learning yet, δψ = 0)
U(T_H') = ℋ'       (updated holor, δψ absorbed)
```

Here, `U(g)` is the morphism `ℋ → ℋ'` whose effect is to:

- keep `Φ^μ` unchanged,
- adjust `χ` (or `κ`) according to `δψ`.

So:

```
U(g) : ℋ → ℋ'
```

is the holor-level echo of the tensor-level operation `g`.

RTTP as the natural square

Now, if we have a holor morphism `f : ℋ → ℋ₂` (e.g., embedding `ℋ` into a bigger composite holor `ℋ₂`), then naturality demands:

```
(U ∘ E)(f) ∘ 𝒯_RTTP(ℋ)
  = 𝒯_RTTP(ℋ₂) ∘ f
```

which, operationally, says:

1. Start from `ℋ`,
2. either:
   - change to `ℋ₂` then run RTTP there,
   - or run RTTP at `ℋ` (extract, use, return as `ℋ'`), then apply the holor-level map induced by `f`,
3. both ways must line up (again, up to explicitly tracked learning drift).

This is how RTTP becomes not just “a story about tensors and holors” but a **coherent functorial kernel** for Holor Calculus.

VI. How to Slot This into v1.1

We insert this Markdown as:

> **Section: Holor Categories and the RTTP Functor**
>
> - Subsection: Categories Hol and Ten
> - Subsection: The Functors E and U
> - Subsection: RTTP as a Natural Transformation
> - Subsection: A Simple RTTP Diagram in Practice

And in the version note / changelog:

> *"Holor Calculus v1.1 makes explicit the categorical structure of the Resonant Tensor Transaction Protocol (RTTP) as a natural transformation `𝒯_RTTP : Id_Hol ⇒ U ∘ E` between a holor category `Hol` and a tensor projection category `Ten`. This structure has been in use in our internal notebooks for approximately ten months before this public integration; the current update formalizes it for collaborators and future work."*
