

RESEARCH LIBRARY

Stratified Universality for Resolved Representations

Profiled lane transport, semantic quotients, carrier change, and certified descent

Lean and evidence guide

Komorebi P.

Standardized publication edition / 8 October 2026

REALIZATION CALCULUS

Reading scope

An exact source map for finite module transport, recorded categorical and holonomy formalization, and the remaining semantic correspondence work.

Contents

1 What this guide certifies 2

2 Three evidence levels 2

3 Source identity and reconstruction 2

4 The module normalization 3
4.1 Why the coordinate type matters . . . 3

5 L1: exact analysis and synthesis 3

6 L2: exact transfer blocks 3

7 L3: composition and path contraction 4

8 L4: refinement 4

9 L5: selectors and exact residuals 4

10 L6: propagation through composition 4

11 L7 and L8: realization and fold loss 5

12 Complete resolution change 5

13 Recorded holonomy and categorical coverage 5

14 Recorded cyclotomic coverage and remaining identifications 6

15 The v2 semantic extension: a precise translation plan 6
15.1 Behavioral kernels . . . 6
15.2 Task combination and bridge loss . . . 7
15.3 Carrier gates and admissibility . . . 7

16 Reading the manuscript registers 7

17 Reproducing the recorded Lean checks 7

18 Verification boundaries for this edition 8

A Clause-level concordance 9

B Inspected declaration inventory 10

C Files accompanying the guide 12

D Evidence sources 12

1 What this guide certifies

This companion connects the finite transport and descent statements of *Stratified Universality for Resolved Representations* to the actual formalization evidence recovered for this edition. It gives a declaration-level map for the inspected module normalization and a separately labeled map for the recorded categorical, holonomy, and cyclotomic release.

The complete written monograph is not claimed to be Lean-complete. The word "formal" in its title refers to explicit mathematical structures and proofs. Lean evidence applies to the precisely named models and declarations, with their hypotheses. Mathematical completion of a written theorem and completion of its Lean translation are separate statements.

Edition status. One complete standalone Lean source was inspected. The v0.7 release README and audit were read. The two matching ZIP transfers failed with HTTP 502. Lean and Lake were absent from PATH, so no fresh Lean compilation or axiom inspection was executed for this final edition.

2 Three evidence levels

1. **Inspected source.** The complete 602-line `StratifiedFoundationsCore.lean` contains 30 theorem declarations, 12 definitions, two structures, and 18 `#check` commands. These are textual source counts. The declaration bodies and hypotheses were inspected.
2. **Recorded release checks.** The v0.7 audit records seven production sources, 183 theorem/lemma declarations, warning-as-error passes, zero `sorry`, `admit`, or custom axiom declarations, and representative standard-foundation axiom reports. Those are historical recorded checks. Their complete source archive and compiler logs were not recovered here.
3. **Unestablished correspondence.** The v2 semantic and doctrine chapters, the broader atlas claims, and all other components without an inspected declaration map remain unmapped in this guide. This does not assert that no formalization exists elsewhere.

The source inventory, declaration concordance, and component status register are included as CSV and JSON files. A row with no Lean declaration is a coverage boundary, not a proposed certificate.

3 Source identity and reconstruction

The main monograph source archive contains TeX, exact Python validators, figures, audit records, and publication material. Its inspected file tree contains no `.lean`, `lean-toolchain`, `lakefile`, or `.olean` file. The Lean normalization is a separate release.

The individually indexed standalone Core was available through a complete text read. Its attempted byte transfer also returned HTTP 502. The included copy was reconstructed from that complete read; a terminal line feed was restored to match the authoritative size of 22,747 bytes. Its local digest is recorded in the provenance JSON. Matching a published byte count is not an independent upstream hash authentication.

This standalone Core is *not* silently identified with any of the seven production modules in the v0.7 audit. The latter names `AbelianCore`, `CategoricalAtlasCore`, `CategoryCore`, `CyclotomicGaloisCore`, `GroupoidLocalSystemCore`, `HolonomyFiniteCore`, and `ModulePolynomialCore`.

4 The module normalization

The inspected namespace is `Komorebi.StratifiedFoundations`. Its basic structure is `ProjectorFamily RMI`, over a commutative ring R, an additive commutative group M, an R-module structure, and a finite index type. It supplies linear endomorphisms P_i satisfying

$$P_i^2 = P_i, \qquad P_i P_j = 0 \ (i \neq j), \qquad \sum_i P_i = 1_M.$$

These identities are fields of the declared structure. The subsequent theorems quantify over every such family; they do not discover a projector family for an arbitrary represented object.

The monograph's finite transport theorem is stated in an additive idempotent-complete category with chosen splittings. The inspected Core specializes that theorem to modules and uses fixed submodules of projectors as the coordinate objects. This is a meaningful normalization, but it is not a declaration-level proof at the full category-theoretic generality.

4.1 Why the coordinate type matters

The exact coordinate target is

$$\mathrm{Lane}(P) = \{c : I \to M \mid P_i(c_i) = c_i \text{ for every } i\}.$$

The unrestricted type $I \to M$ would contain coordinates outside their lanes. Synthesis and analysis are inverse only on the resolved target. The definitions `laneSpace`, `analysisLinear`, `synthesisLinear`, and `laneEquiv` implement this restriction.

5 L1: exact analysis and synthesis

Analysis sends x to $(P_i x)_i$; synthesis adds the coordinates. The source proves both inverse laws:

$$\mathsf{SA}(x) = x, \qquad \mathsf{AS}(c) = c \quad (c \in \mathrm{Lane}(P)).$$

The first law is completeness. For the second, apply P_j to the finite sum; orthogonality removes all terms except j, whose lane condition returns c_j. The resulting `laneEquiv` is a linear equivalence.

`synthesize_analyze` is a convenient earlier function-level lemma. The stronger normalized inverse pair is `synthesisLinear_analysisLinear` and `analysisLinear_synthesisLinear`.

6 L2: exact transfer blocks

For projector families P on M, Q on N, and a linear map $T : M \to N$, the source defines

$$T_{ji} = Q_j T P_i.$$

`transferBlock_reconstruction` proves

$$T = \sum_{j,i} Q_j T P_i.$$

`transferBlock_ext` says two maps are equal if every ambient corner block agrees. These statements retain their full finite index types and module hypotheses.

The printed category theorem uses typed split-lane blocks $\pi_j T \iota_i : M_i \to N_j$. An ambient corner is the same block inserted back into the ambient source and target. Showing equality of the two formulations requires the chosen splitting maps and their identities; they must not be erased in a categorical correspondence.

7 L3: composition and path contraction

For $U : N \to L$, a third projector family S, and source/target addresses i, k, the source proves

$$S_k U T P_i = \sum_j (S_k U Q_j)(Q_j T P_i).$$

The declaration `transferBlock_comp_matrix` is the morphism-level two-step contraction. The finite path law for a longer chain follows by repeated contraction in the written theorem. A future fully explicit Lean map for arbitrary chain length should supply the induction and the address product type; this guide does not invent that declaration.

Cancellation can make a composite block zero even when individual paths are supported. Thus support propagation gives a containment result, not equality of support sets.

8 L4: refinement

The structure `Refines` supplies a declared address map and the identity that every coarse projector is the finite sum of its fine descendants. `transferBlock_refinement` proves

$$Q_a T P_i = \sum_{\rho_Y(b)=a} \sum_{\rho_X(j)=i} Q'_b T P'_j.$$

This is one exact refinement step. The full finite resolution system also needs identity maps and compatibility of successive address maps. Those are additional data, not consequences of cardinality comparisons.

9 L5: selectors and exact residuals

A Boolean source-target mask E keeps the ambient blocks whose addresses belong to E:

$$\mathrm{Sel}_E(T) = \sum_{(j,i)\in E} Q_j T P_i, \qquad R_E(T) = T - \mathrm{Sel}_E(T).$$

The inspected source proves that a retained block is unchanged, a discarded block is zero, selectors are idempotent, and two selectors compose by intersection. The exact reconstruction law is

$$\mathrm{Sel}_E(T) + R_E(T) = T.$$

The residual is retained mathematical data. Discarding blocks does not become lossless merely because the retained blocks have convenient labels.

10 L6: propagation through composition

The support relation `composeSupport` is existential composition:

$$(F \circ E)(k, i) \iff \exists j,\ F(k, j) \wedge E(j, i).$$

`select_comp_of_supported` proves that a composite lies in this support relation when both factors satisfy their selected-support contracts.

For retained maps T_0, U_0, the exact residual identity is

$$UT - U_0T_0 = (U - U_0)T_0 + U_0(T - T_0) + (U - U_0)(T - T_0).$$

The declaration `composition_residual_identity` proves the general linear-map identity, and `selected_composition_residual_identity` specializes it to selectors. The residual-residual term is required.

No norm, tolerance, probability, or convergence claim follows from this algebraic identity without further hypotheses.

11 L7 and L8: realization and fold loss

The v0.7 report records an additive-category layer, including split-lane Hom/block equivalence and additive-functor realization. Its declaration bodies were not inspected here, so L7 has report-level coverage only.

The standalone Core gives two exact fold statements for a linear map $\Phi : A \to B$:

$$\Phi(x) = \Phi(y) \iff x - y \in \ker \Phi,$$

and

$$L\Phi = 1_A \Longrightarrow \ker \Phi = 0.$$

The declarations are `fold_eq_iff_sub_mem_ker` and `ker_eq_bot_of_leftInverse`. A zero kernel proves injectivity; it does not, over an arbitrary ring, automatically supply a chosen linear left inverse. The monograph's sufficient split-lossless criterion uses globally typed selectors and supplied retractions. Constructing its injectivity or left inverse from those hypotheses is not declaration-mapped here.

12 Complete resolution change

The inspected `changeResolution` is $\mathsf{A}_Q\mathsf{S}_P$, a linear equivalence between two complete resolved coordinate spaces on the same ambient module. Its inverse is the reversed change, and successive changes compose strictly. These are the declarations `changeResolution_symm` and `changeResolution_trans`.

This is an object-level prerequisite or analogue of the printed Hom/block coordinate coarsening. The latter uses both source and target splittings and intertwines matrix multiplication; that full map is not declaration-mapped here.

This construction retains all ambient information in each complete resolution. A noninjective fold or a selector deletion is governed by its kernel or residual instead.

13 Recorded holonomy and categorical coverage

The v0.7 README records the following finite normalization layers:

- multi-chart additive local systems and compatible sections;
- base-fiber loop-fixed normal form;
- categorical realization of H2–H5 under ambient Hom coherence;

- finite holonomy orbit descent, pair-orbit coordinate expansion, and exact orbital path-count multiplication;
- finite permutation-module equivalence between pair-orbit coordinates and equivariant homomorphisms.

The report names `CoherentHomPair` as an explicit coherence package. Deriving it from a complete category-valued pseudofunctor/descent-object layer remains a reported open formalization obligation.

Object-address orbits and source-target pair orbits answer different questions. The former classify globally labeled projector-sum coarsenings. The latter classify invariant matrix positions and equivariant support selectors. Transitive object action can still have several pair orbits.

14 Recorded cyclotomic coverage and remaining identifications

The v0.7 report records exact additive-order classification of cyclic unit orbits, actual primitive Fourier idempotents in the group algebra, complete orthogonal orbit sums, coefficientwise rationality, literal descent to $\mathbb{Q}[C_n]$, and integral Ramanujan character sums. The audit's representative axiom examples include `primitiveFourierIdempotent_sum`, `galoisOrbitProjector_coeff_mem_rational_range`, `rationalOrbitProjector_sum_divisors`, `exists_int_ramanujanOrbitSum`, and `rationalOrbitProjector_eq_inv_smul_integralRamanujanElement`. These names are recorded audit examples, not declaration bodies inspected in this edition.

The same release explicitly leaves the following identifications open:

1. the certified integral orbit sum with $c_q(\ell) = \sum_{d \mid \gcd(q,\ell)} d\,\mu(q/d)$;
2. the associated classical Ramanujan normal form;
3. the factor identification $\mathbb{Q}[C_n]\widetilde{E}_q \cong \mathbb{Q}(\zeta_q)$ and full cyclotomic product decomposition in the printed formulation.

The existence of rational orthogonal projectors is already a strong finite formal result. It does not supply these further identifications by naming the coefficient a Ramanujan sum.

15 The v2 semantic extension: a precise translation plan

The behavioral quotient chapters contain written results and exact finite validators. No matching Lean declaration body was established for those chapters during this edition. The following is a formalization plan, not executable checked Lean code.

15.1 Behavioral kernels

In a module slice, represent observations as linear maps and continuations as a monoid action by module endomorphisms. Define

$$N = \bigcap_{m,j} \ker(o_j \rho(m)).$$

The module intersection exists independently of a possibly infinite product of observations. Prove invariance using the action multiplication law, then use the quotient universal property. The direction of the coarsest-quotient factorization is from any exact surjective representation Z to X/N.

15.2 Task combination and bridge loss

For two task kernels, the joint task uses intersection. Their common coarsening uses the sum only in the linear setting with the stated task interpretation. A task-compatible bridge satisfies $f(N_X) \subseteq N_Y$, enabling the quotient map $\overline{f} : X/N_X \to Y/N_Y$. Its exact loss and failure of target coverage are separate objects:

$$\ker \overline{f} = f^{-1}(N_Y)/N_X, \qquad \operatorname{coker} \overline{f} = Y/(N_Y + f(X)).$$

The formal translation must prove the quotient identifications and maps, not merely define these expressions and assume exactness.

15.3 Carrier gates and admissibility

For set-valued carrier semantics, the factorization gate uses kernel pairs: $\mathrm{KerPair}(s) \subseteq \mathrm{KerPair}(t)$. Equality of residual functions and linear dependence of residual functions must remain distinct relations. The v2 relation-module construction is the explicit bridge between them.

Admissibility monotonicity is an infimum theorem. Formalizing nonattainment requires the declared topology and compactness hypotheses. An arbitrary Hausdorff space need not turn compactness into sequential compactness; the final traditional edition instead uses an attained positive minimum on each compact conditioned sublevel. No placeholder proof is supplied here for these analytic results.

16 Reading the manuscript registers

The final-edition component concordance retains 75 stable rows: 52 internally proved, 12 classical imports or reinterpretations, two conditionally proved, six conjectural, and three framework/definition components. This register uses the amended final source's hypotheses, including finite branch/history aggregation, empty conventions, and corrected support and metric scope. The separate inherited v0.4 table preserves the original hypotheses as historical evidence. It is not the canonical final-edition scope. These registers track mathematical status and source selection, not a count of Lean-certified statements or a completion percentage.

The separate manuscript result inventory records theorem, lemma, proposition, corollary, and conjecture environments in the frozen original v2.0.0 input graph. Its 242 rows preserve original labels and source coordinates, not amended final-edition statement text. Environment counts differ from component counts because one component can gather several results and one theorem can contain multiple clauses.

The later September 24 Blocks 17–26 checksum manifest lists Markdown, Python, and JSON material, with no Lean entries. Its ZIP bytes were not recovered. Consequently it contributes no additional Lean coverage claim in this guide and is not silently substituted for the controlling v2 paper.

17 Reproducing the recorded Lean checks

The recorded v0.7 pins are Lean 4.34.0-rc2, Lean commit `6a10ac8c22beadecabdbb0919c2b50214762f91d`, and Mathlib commit `aeb72f1e9981b6d35b426b16dca7759be3e7ba06`. Use the actual recovered v0.7 project and its dependency files when available; do not manufacture a project lock from these prose pins alone.

In its pinned portable environment, the release's recorded check command is:

```
lake env lean -DwarningAsError=true <source>
```

Here `<source>` must be replaced by one actual source path from the release audit. An explicit example, once that module has been recovered, is:

```
lake env lean -DwarningAsError=true src/CyclotomicGaloisCore.lean
```

The standalone inspected Core can separately be checked in a compatible project using:

```
lake env lean -DwarningAsError=true StratifiedFoundationsCore.lean
```

Its compatibility with the v0.7 pin has not been freshly tested here.

Retain compiler exit codes, complete logs, source digests, the toolchain and Mathlib revision, and declaration-level `#printaxioms` output. An audit name without a recovered statement is report-level evidence. A successful `#check` establishes elaboration only in the executed environment; an unexecuted `#check` line in a source is not a pass.

The recorded standard axiom union is `propext`, `Classical.choice`, and `Quot.sound` for the representative examples. No universal axiom coverage is inferred from representative checks. Placeholder scanning is supplementary to compilation and the declaration-level axiom audit.

18 Verification boundaries for this edition

The final edition separately replays the principal exact Python validators. That execution checks finite arithmetic formulas and implementation cases. It is separate from Lean proof checking and does not upgrade unmapped semantic or analytic results to a formal certificate.

No dependencies were installed and no compiler result was invented for this guide. The attempted transfers failed with HTTP 502. The source read and the historical audit read succeeded. The included reconstructed source preserves that distinction in its provenance. The guide supplies a usable theorem map and a reproduction route while preserving the written paper's actual mathematical statements.

A Clause-level concordance

The table uses stable manuscript labels. “Inspected” always means an inspected declaration body with no fresh compiler run. “Recorded” refers to the separate historical v0.7 evidence.

Clause	Manuscript target	Inspected declarations or evidence boundary

L1	Exact analysis and synthesis
	`thm:block15-universal-lane-transport`
		Inspected.
		`ProjectorFamily.synthesisLinear_analysisLinear`
		`ProjectorFamily.analysisLinear_synthesisLinear`
		`ProjectorFamily.laneEquiv`
		Finite module normalization; resolved laneSpace, not unrestricted function space.

L2	Unique transfer blocks
	`thm:block15-universal-lane-transport`
		Inspected.
		`transferBlock_reconstruction`
		`transferBlock_ext`
		Ambient corner blocks Q_j T P_i; categorical Hom-to-split-block bijection requires additional category layer.

L3	Composition and finite path contraction
	`thm:block15-universal-lane-transport`
		Inspected.
		`transferBlock_comp`
		`transferBlock_comp_matrix`
		Two-step finite contraction is inspected; iterated address-path expansion needs induction.

L4	Projector-sum refinement `thm:block15-universal-lane-transport`	**Inspected.** `transferBlock_refinement` One declared source and target refinement step, with fibre-sum hypotheses.

L5	Boolean selectors and named residuals
	`thm:block15-universal-lane-transport`
		Inspected.
		`select_idempotent`
		`select_inter`
		`select_add_residual`
		Selector endomorphisms on module morphisms; masks retain exact ambient blocks.

L6	Support and residual propagation
	`thm:block15-universal-lane-transport`
		Inspected.
		`select_comp_of_supported`
		`selected_composition_residual_identity`
		Support containment and exact three-term residual identity; no equality-of-support claim.

L7	Additive profile-preserving realization `thm:block15-universal-lane-transport`	**Recorded only.** Historical v0.7 report describes CategoryCore; no matching declaration body inspected here.

Clause	Manuscript target	Inspected declarations or evidence boundary

L8	Fold loss
	`thm:block15-universal-lane-transport`
		Inspected.
		`fold_eq_iff_sub_mem_ker`
		`ker_eq_bot_of_leftInverse`
		Linear fold kernel and supplied-left-inverse criterion inspected; construction of injectivity/left inverse from globally typed Q_e and r_e hypotheses is not mapped.

RES	Complete resolution change
	`prop:block15-certified-coarsening`
		Inspected.
		`ProjectorFamily.changeResolution_symm`
		`ProjectorFamily.changeResolution_trans`
		Object-level prerequisite/analogue inspected; the Hom/block coarsening map and its multiplication compatibility are not declaration-mapped here.

H1-H5	Holonomy-twisted compatible sections `thm:block16-holonomy-twisted-lane-transport`	**Recorded only.** Reported v0.7 GroupoidLocalSystemCore/CategoricalAtlasCore coverage; ambient Hom coherence explicitly assumed.
BASE	Base-fiber invariant normal form `cor:block16-base-fiber-invariant-normal-form`	**Recorded only.** Reported v0.7 local-system coverage; no source body inspected in this edition.
ORBIT	Finest globally labeled coarsening `thm:block16-orbit-descent`	**Recorded only.** Reported v0.7 HolonomyFiniteCore coverage; object and pair orbits must remain distinct.
PAIR	Permutation orbital calculus `thm:block16-permutation-orbital-calculus`	**Recorded only.** Reported v0.7 orbit basis and path-count coverage; no inspected declaration map.
CYC	Rational cyclotomic Fourier lanes `prop:block16-rational-cyclotomic-orbit-lanes`	**Recorded only.** Partial recorded v0.7 coverage. Classical Möbius formula and cyclotomic factor/CRT identification remain explicit formalization obligations.

B Inspected declaration inventory

All entries below occur in the included standalone Core. The source location is a reading coordinate, not a compiler certificate. The namespace prefix is `Komorebi.StratifiedFoundations`; declarations in the first projector namespace additionally use `ProjectorFamily`.

Line	Kind	Declaration
23	structure	`ProjectorFamily`
37	def	`ProjectorFamily.analyze`
41	def	`ProjectorFamily.synthesize`
44	theorem	`ProjectorFamily.synthesize_analyze`

Line	Kind	Declaration
49	theorem	`ProjectorFamily.eq_of_projectors_eq`
63	def	`ProjectorFamily.laneSpace`
74	def	`ProjectorFamily.analysisLinear`
80	def	`ProjectorFamily.synthesisLinear`
86	theorem	`ProjectorFamily.synthesisLinear_analysisLinear`
93	theorem	`ProjectorFamily.analysisLinear_synthesisLinear`
109	def	`ProjectorFamily.laneEquiv`
117	def	`ProjectorFamily.changeResolution`
125	theorem	`ProjectorFamily.changeResolution_symm`
134	theorem	`ProjectorFamily.changeResolution_trans`
156	theorem	`linearMap_fintype_sum_apply`
164	def	`transferBlock`
172	theorem	`transferBlock_reconstruction_apply`
188	theorem	`transferBlock_reconstruction`
198	theorem	`transferBlock_ext`
217	theorem	`transferBlock_comp`
235	theorem	`transferBlock_comp_matrix`
256	def	`maskBlocks`
263	theorem	`maskBlocks_idempotent`
273	theorem	`maskBlocks_inter`
283	theorem	`transferBlock_self`
294	theorem	`transferBlock_ne_target`
304	theorem	`transferBlock_ne_source`
314	def	`select`
323	theorem	`transferBlock_select_of_mem`
355	theorem	`transferBlock_select_of_not_mem`
384	theorem	`select_idempotent`
399	theorem	`select_inter`
422	def	`composeSupport`
427	theorem	`select_comp_of_supported`
462	def	`residual`
470	theorem	`select_add_residual`

Line	Kind	Declaration
479	theorem	`select_all`
487	theorem	`residual_all`
496	theorem	`composition_residual_identity`
509	theorem	`selected_composition_residual_identity`
524	structure	`Refines`
533	theorem	`transferBlock_refinement`
571	theorem	`fold_eq_iff_sub_mem_ker`
576	theorem	`ker_eq_bot_of_leftInverse`

C Files accompanying the guide

Lean_Concordance.csv / .json
Clause-level correspondences, evidence classes, exact scope boundaries, and execution status.

Inspected_Declaration_Inventory.csv
Every declaration found in the included standalone Core, with namespace and source location.

Manuscript_Result_Inventory.csv
The 242 statement environments in the frozen original v2.0.0 source graph, with original labels and reading coordinates.

Component_Status_Concordance.csv
The canonical 75-row final-edition component table, with amended hypotheses and publication-amendment notes.

Component_Status_Concordance_Inherited_v0.4.csv
The original 75-row table with inherited hypotheses, explicitly historical rather than canonical.

Recorded_v0.7_Audit_Evidence.json
The complete readable historical audit record, with source identity and evidence level.

inspected_source/
The reconstructed complete Core and provenance record. No compiled artifact or fabricated project lock is included.

D Evidence sources

1. Komorebi P., *Stratified Universality for Resolved Representations: Profiled Lane Transport, Semantic Quotients, Carrier Change, and Certified Descent*, Formal Foundations v2.0.0, September 23, 2026. Controlling TeX, theorem sections, exact validators, and component concordance.
2. Komorebi P., `StratifiedFoundationsCore.lean`, indexed standalone source, September 9, 2026. Complete 602-line read; reconstructed local copy.

3. Komorebi P., *Stratified Foundations Lean Normalization v0.7.0*, `README.md` and `AUDIT.json`, September 9, 2026. Readable historical verification and scope records; archive bytes unavailable here.

4. Komorebi P., `Komorebi_Stratified_Blocks17_26_SHA256SUMS.txt`, September 24, 2026. Later companion manifest only; archive not inspected.