PAPER VI

Transfer Semigroups and Memory

Probabilistic refinement, fractional operators, and exact hidden-state elimination

Follow the actual declarations from semigroup transfer to Bayesian sufficiency, fractional calculus, finite paths, and hidden-state memory. Each interface carries its own hypotheses and its own evidence.

Komorebi P.

How to use this volume

A guide to the actual formal development. 33 scoped manuscript results. 341 supporting theorem targets. 47 proof modules. The eleven chapters explain the mathematical interfaces and how to reproduce their recorded evidence. The complete ledgers then give exact declarations, source locations, imports, and nine dependency pins.

This edition preserves 48 exact Lean excerpts. The broader traditional development has 142 theorem-like result records and 140 written proof bodies across overlapping branches. Its extra written results do not extend the 33-result formal contract. Three continuum conjectures remain open.

Kernel execution evidence is inherited from the v0.5.0 release dated 6 September 2026. This edition reconciles source bytes, declaration locations, saved axiom records, and current finite computations. No fresh Lean compilation was run.

Contents

1 How to read the formal companion 1

2 Semigroups and rectangular defects 3

3 Markov mass and conditional expectation 6

4 Bayesian normalization, sufficiency, and flow 8

5 Fractional calculus and refinement towers 12

6 Green reconstruction and finite paths 17

7 Schur elimination and nonnormal memory 19

8 Canonical jets and holomorphic naturality 23

9 Survival, hazard, and incomplete absorption 26

10 The retained extensions and the formal boundary 28

11 Reproduction and evidence review 32

12 Complete ledger of the 33 scoped results 35

13 All 341 supporting theorem declarations 45

14 Local module imports and locked dependencies 61

1 How to read the formal companion

The Lean project formalizes the explicitly scoped mathematical core of Paper VI. It turns an equation such as $Ce^{tA} = e^{tB}C$ into a statement with named spaces, scalar fields, operator types, and hypotheses. This guide follows those interfaces. It explains how the formal pieces fit together and where the traditional edition goes beyond them.

The latest core is release `0.5.0-checked-fifth-block`. Its preserved ledger records all 33 scoped results as complete. The source contract names 341 explicit theorem declarations across 47 production modules. The preserved September 6 compiler and namespace audit reports 782 declarations inspected in the project namespace. These counts measure different things: manuscript endpoints, named theorem declarations, files, and the wider namespace. None is a count of independent new mathematical discoveries.

The traditional monograph preserves the larger written development in Parts I through III, followed by the worked atlas in Part IV. Part I is the latest reviewed core. Part II retains the strengthened exposition. Part III retains earlier results not subsumed by that exposition and the bounded Banach operator material migrated from Paper IV. The theorem-like result records across these parts include conceptual overlap. Formal completion remains attached to the 33 specific Part I ledger entries.

> **Read one endpoint through its whole interface**
>
> Start with the statement and proof in the traditional volume. Locate its numbered row in the 33-result ledger. Read the surrounding variables in the original Lean module, then compare the declaration's hypotheses and conclusion with the mathematical statement. Finally, identify whether the evidence is an inherited kernel record, a current source reconciliation, or a bounded computation. These steps answer different questions.

1.1 Three forms of evidence

Evidence	What it establishes here
Inherited kernel evidence	The saved September 6 build and axiom audit for the pinned source and dependencies. No new Lean compilation occurred in this edition.
Fresh source and computational checks	Source hashes, target inventories, verifier controls, and available finite examples were rerun for this edition. These are not kernel checking or universal proofs.
Mathematical correspondence	The relationship between prose, hypotheses, definitions, ordinary real/complex specializations, and theorem interfaces is a manual review responsibility.

The source report explicitly records `full_paper_formalized: false`. That remains correct even though `all_scoped_results_complete: true` is recorded for the historical kernel run. Figures, explanatory prose, broader extensions, and three open conjectures are not swallowed by the scoped denominator.

1.2 A practical reading route

Read the matching traditional argument first, use the visual guide to inspect a finite example, and then find its ledger entry in Section 12. Follow the named endpoint to a source excerpt or the complete declaration catalog. Check the ambient variables at the top of the original module before attempting to reuse an excerpt. A snippet does not repeat every `variable`, `open`, instance, or imported notation in its file.

All displayed Lean blocks are exact contiguous source excerpts from the preserved production modules. Raw UTF-8 copies and source hashes accompany the editable guide. Font fallback in the PDF changes rendering only. The snippets are reading excerpts, not independent programs that can be compiled without their original context.

1.3 The notation that matters most

Lean notation	Mathematical reading
`Matrix I J ℂ`	A complex matrix with rows indexed by I and columns by J. Different index types permit rectangular maps.
`E →L[ℂ] F`	A bounded complex linear map from E to F. Completeness and inner-product assumptions come from ambient instances.
`C.comp A`	Composition $C \circ A$: apply A first, then C.
`Module.End ℂ V`	A complex linear endomorphism of V. Finite dimension is a separate hypothesis.
`IntegrableOn f (Ioi 0)`	Bochner integrability of f over $(0, \infty)$ with the stated measure, here ordinarily Lebesgue measure.
`Tendsto f atTop (𝓝 0)`	The limit of $f(t)$ is zero as t tends to $+\infty$.
`0 ≤ A`	Positivity in the operator or C^*-algebra order in the fractional modules. This is not entrywise matrix positivity.
`∀ i j, 0 ≤ K i j`	Entrywise nonnegativity of a finite real matrix.
`Ring.inverse`	A total inverse operation. The surrounding proof must establish the required invertibility before an inverse formula has its intended analytic meaning.

1.4 Common mistakes that the interfaces prevent

Column-stochastic transition matrices act on probability columns. Observable propagation uses the transpose. Probability aggregation is a rectangular mass-summing map. A weighted conditional expectation is a square projector on observables. Their identities are related but their types differ. Similarly, e^{tA} describes a general generator, while a positive fractional operator produces e^{-tA^α}. The minus sign changes the Duhamel defect in the fractional semigroup theorem.

Zero evidence, one coarse packet, a zero-dimensional space, a zero spectral value, and a nonnormal hidden generator are not interchangeable edge cases. Each belongs to a particular interface. This guide identifies the relevant hypothesis at the point where it is used.

2 Semigroups and rectangular defects

The algebraic layer begins with finite packet blocks $P_b A P_a$ in a ring. It proves reconstruction, multiplication through intermediate packets, and telescoping formulas without assuming stochasticity or self-adjointness. The analytic layer then uses the actual algebra exponential. The definition is short because convergence and differentiation are supplied by the imported analysis library and the ambient Banach-algebra structure.

Source: `KomorebiPaperVI/Semigroups.lean`.

```
noncomputable def transfer (A : R) (t : ℝ) : R := exp (t • A)
```

Source: `KomorebiPaperVI/Semigroups.lean`.

```
theorem transfer_add (A : R) (s t : ℝ) : transfer A (s+t) = transfer A s*transfer A t := by
  let : NormedAlgebra ℚ R := NormedAlgebra.restrictScalars ℚ ℝ R
  dsimp [transfer]
  rw [add_smul]
  exact exp_add_of_commute (((Commute.refl A).smul_left s).smul_right t)
```

2.1 Reconstruction is stronger than a diagram

If $\sum_a P_a = 1$, every input/output block participates in reconstruction. Idempotence supplies the block product identity. Pairwise reduction requires additional hypotheses on the off-diagonal generator blocks. It does not follow merely from writing A in blocks. The endpoint for core 2.1 uses the actual exponential law:

Source: `KomorebiPaperVI/Semigroups.lean`.

```
theorem exponential_packet_convolution {I : Type*} [Fintype I]
    (P : I → R) (hP : ∑ i, P i=1) (hid : ∀ i, P i*P i=P i)
    (A : R) (s t : ℝ) (b a : I) :
    Algebra.block P (transfer A (s+t)) b a =
      ∑ c, Algebra.block P (transfer A s) b c*Algebra.block P (transfer A t) c a :=
  Algebra.packet_semigroup_convolution P hP hid (transfer A) (transfer_add A) s t b a
```

For core 2.2, `ReducingPackets.generator_commutes` derives commutation from vanishing off-diagonal generator blocks. The subsequent theorems prove vanishing exponential cross blocks and equality with the compressed evolution. A projection that is not reducing can produce a memory term when its complement is eliminated. The later memory chapter explains why.

2.2 The domain of the refinement map is explicit

Let $A : E \to E$, $B : F \to F$, and $C : E \to F$. Define $D = CA - BC$. The real Banach-space theorem differentiates $e^{(t-r)B} C e^{rA}$ and integrates that derivative. Rectangular types enforce the order of every composition.

Source: KomorebiPaperVI/Semigroups.lean.

```
theorem duhamel_rectangular (A : E →L[ℝ] E) (B : F →L[ℝ] F)
    (C : E →L[ℝ] F) (t : ℝ) :
    C.comp (transfer A t)-(transfer B t).comp C =
      ∫ r in 0..t, (transfer B (t-r)).comp ((C.comp A-B.comp C).comp (transfer A r)) := by
  have hc : Continuous (fun r => (transfer B (t-r)).comp
      ((C.comp A-B.comp C).comp (transfer A r))) := by
    fun_prop
  have h := intervalIntegral.integral_eq_sub_of_hasDerivAt
    (fun r _ => duhamel_integrand_derivative A B C t r) (hc.intervalIntegrable 0 t)
  simpa [transfer_zero, ContinuousLinearMap.one_def] using h.symm
```

The identity itself is valid for every real t as an oriented interval integral. Bounds such as $\|Ce^{tA} - e^{tB}C\| \le t\|D\|$ additionally assume $t \ge 0$ and contraction estimates for both semigroups. The general growth estimate lives in `OperatorBounds.growth_duhamel_bound`. The exact case specializes the same identity to $D = 0$:

Source: KomorebiPaperVI/Semigroups.lean.

```
theorem exact_rectangular_intertwining (A : E →L[ℝ] E) (B : F →L[ℝ] F)
    (C : E →L[ℝ] F) (h : C.comp A=B.comp C) (t : ℝ) :
    C.comp (transfer A t)=(transfer B t).comp C := by
  have hd := duhamel_rectangular A B C t
  simp only [h, sub_self, ContinuousLinearMap.zero_comp, ContinuousLinearMap.comp_zero,
    intervalIntegral.integral_zero] at hd
  exact sub_eq_zero.mp hd
```

2.3 Discrete time and resolvents

Core 5.2 is an exact finite sum before a norm estimate is applied. Horizon zero is included by the empty sum. Its source uses ring-valued literal matrices with independently typed row and column spaces:

Source: KomorebiPaperVI/Algebra.lean.

```
theorem finite_horizon_defect (C : Matrix m n R) (A : Matrix n n R) (B : Matrix m m R)
    (k : ℕ) :
    C*A^k-B^k*C = ∑ j ∈ Finset.range k, B^(k-1-j)*(C*A-B*C)*A^j := by
  induction k with
  | zero => simp
  | succ k ih =>
    rw [Finset.sum_range_succ]
    have hs : (∑ j ∈ Finset.range k, B^(k+1-1-j)*(C*A-B*C)*A^j) =
        B*(∑ j ∈ Finset.range k, B^(k-1-j)*(C*A-B*C)*A^j) := by
      rw [Matrix.mul_sum]
      apply Finset.sum_congr rfl
      intro j hj
      have hjk := Finset.mem_range.mp hj
      have he : k+1-1-j = (k-1-j)+1 := by omega
      rw [he, pow_succ']
      simp only [Matrix.mul_assoc]
```

```
    rw [hs, ← ih]
    simp only [Nat.add_sub_cancel, Nat.sub_self, pow_zero, Matrix.one_mul]
    simp only [pow_succ', Matrix.mul_sub, Matrix.sub_mul, Matrix.mul_assoc]
    abel
```

The resolvent identity requires one inverse equation on each side. This is deliberate: the proof does not invoke a vague invertibility claim when only the relevant cancellations are needed. The scalar must commute with the entries of C in the general ring statement.

Source: `KomorebiPaperVI/Algebra.lean`.

```
theorem rectangular_resolvent_identity (C : Matrix m n R)
    (A : Matrix n n R) (B : Matrix m m R)
    (Rf : Matrix n n R) (Rc : Matrix m m R) (z : R)
    (hz : ∀ i j, z*C i j = C i j*z)
    (hf : (Matrix.scalar n z-A)*Rf=1)
    (hc : Rc*(Matrix.scalar m z-B)=1) :
    C*Rf-Rc*C=Rc*(C*A-B*C)*Rf := by
  have hzC : Matrix.scalar m z*C=C*Matrix.scalar n z := by
    ext i j
    simpa using hz i j
  have hd : C*A-B*C=(Matrix.scalar m z-B)*C-C*(Matrix.scalar n z-A) := by
    rw [Matrix.sub_mul, Matrix.mul_sub, hzC]
    abel
  rw [hd, Matrix.mul_sub, Matrix.sub_mul]
  simp only [← Matrix.mul_assoc, hc, Matrix.one_mul]
  rw [Matrix.mul_assoc (Rc*C), hf, Matrix.mul_one]
```

For ordinary complex matrices the scalar commutation is automatic. For core 3.3 the specified inverse equations are supplied by the actual resolvents. The broad Banach-space and contour extensions in the traditional paper have their own domains and hypotheses. The literal matrix interface is the formal endpoint recorded in this ledger entry.

3 Markov mass and conditional expectation

3.1 Column conservation

`MarkovSemigroup` proves nonnegativity of the actual matrix exponential by shifting a Metzler generator by a scalar identity, expanding a nonnegative matrix exponential, and then undoing the scalar shift. Conservation follows from zero column sums of the generator. The final probability statement combines these two independent facts.

Source: `KomorebiPaperVI/MarkovSemigroup.lean`.

```
theorem markov_probability (Q : Matrix I I ℝ) (hpos : ∀ i j, i≠j → 0 ≤ Q i j)
    (hcol : ∀ j, ∑ i, Q i j=0) (p : I → ℝ) (hp : ∀ i, 0 ≤ p i)
    (hpsum : ∑ i, p i=1) (t : ℝ) (ht : 0 ≤ t) :
    (∀ i, 0 ≤ (exp (t • Q)).mulVec p i) ∧ ∑ i, (exp (t • Q)).mulVec p i=1 := by
  constructor
  · intro i
    exact Finset.sum_nonneg (fun j _ => mul_nonneg (markov_nonnegative Q hpos t ht i j) (hp j))
  · simp only [Matrix.mulVec,dotProduct,Finset.sum_comm,← Finset.sum_mul,markov_column_sum Q hcol,
      one_mul,hpsum]
```

Here Q_{ij} is the rate from state j to state i. Off-diagonal nonnegativity and zero column sums are the generator hypotheses. The condition $t \geq 0$ is essential for the positivity argument. Exact discrete lumpability is core 5.1's rectangular identity $CK^n = \bar{K}^n C$. The complete quotient obstruction and detailed coordinate characterization in Part II are related written results, not additional completed Lean ledger entries.

3.2 Weighted projection on observables

For a partition map $f : I \to A$, `Coarsening.coarsen` sums probability mass over fibers. In contrast, `ConditionalExpectation.expect` returns a function on the original fine space. It averages an observable within each fiber with respect to finite weights p_i. Strict positivity guarantees that the occupied fiber denominator at each fine point is nonzero.

Source: `KomorebiPaperVI/ConditionalExpectation.lean`.

```
theorem expect_idempotent (f : I → A) (p : I → ℝ) (hp : ∀ i, 0 < p i) (v : I → ℂ) :
    expect f p (expect f p v)=expect f p v :=
  expect_constant f p hp (average f p v)
```

The source also proves that the range is exactly the block-constant functions and that the expectation equals a literal kernel matrix applied to the observable. Self-adjointness is with respect to the weighted inner product, not ordinarily the unweighted Euclidean inner product:

Source: `KomorebiPaperVI/ConditionalExpectation.lean`.

```
theorem expect_self_adjoint (f : I → A) (p : I → ℝ) (v w : I → ℂ) :
    weightedInner p (expect f p v) w=weightedInner p v (expect f p w) := by
  rw [expect_matrix,expect_matrix]
  simp only [weightedInner,Matrix.mulVec,dotProduct,Finset.mul_sum,Finset.sum_mul,
```

```
    map_sum,map_mul]
  rw [Finset.sum_comm]
  apply Finset.sum_congr rfl
  intro i _
  apply Finset.sum_congr rfl
  intro j _
  have h := kernel_weighted_symmetry f p j i
  calc
    _ = ((p j:ℂ)*kernel f p j i)*v i*conj (w j) := by ring
    _ = _ := by rw [h]; ring
```

The displayed self-adjoint identity is algebraic and its signature has no positivity premise. Calling the pairing a positive-definite inner product, or deriving range separation from it, uses the strict positivity hypotheses elsewhere in the module. Reuse the endpoint matching the intended claim rather than adding an interpretation that its type does not supply.

3.3 Towers and martingale packets

Composing $f : I \to A$ with $g : A \to B$ makes a coarser partition. Both products of the corresponding conditional kernels equal the coarse kernel:

Source: `KomorebiPaperVI/ConditionalTowers.lean`.

```
theorem conditional_kernel_product (f : I → A) (g : A → B) (p : I → ℝ)
    (hp : ∀ i, 0 < p i) :
    kernel (g ∘ f) p*kernel f p=kernel (g ∘ f) p ∧
      kernel f p*kernel (g ∘ f) p=kernel (g ∘ f) p := by
  constructor
  · apply Matrix.mulVec_injective
    funext v
    rw [← Matrix.mulVec_mulVec,← expect_matrix,← expect_matrix,coarse_after_fine f g p
     ↪ hp,expect_matrix]
  · apply Matrix.mulVec_injective
    funext v
    rw [← Matrix.mulVec_mulVec,← expect_matrix,← expect_matrix,fine_after_coarse f g p
     ↪ hp,expect_matrix]
```

The module extends this to arbitrary pairs in a nested family: the product is the kernel at the coarser index. `MartingalePackets` then supplies the abstract difference-projector algebra, and `ConditionalTowers` applies it to concrete kernels. The differences are pairwise annihilating idempotents and sum to the terminal projector. The formal scope is finite weighted conditional expectation. A general measure-theoretic conditional-expectation tower is a separate classical extension in the traditional exposition.

4 Bayesian normalization, sufficiency, and flow

4.1 An equation is not a probability claim

Lean defines the posterior by a total real division. This makes the expression meaningful as a term even when evidence is zero. Theorems asserting unit mass or probability preservation require nonzero or positive evidence explicitly. The basic definition is:

Source: KomorebiPaperVI/FiniteBayes.lean.

```
noncomputable def update (p l : I → ℝ) (i : I) : ℝ := l i*p i / evidence p l
def l1 (p : I → ℝ) : ℝ := ∑ i, |p i|
```

Core 7.1 proves composition of two likelihood updates with both the first evidence and the combined evidence nonzero. The proof rewrites sequential evidence and then performs field algebra.

Source: KomorebiPaperVI/FiniteBayes.lean.

```
theorem posterior_composition (p l k : I → ℝ) (hz : evidence p l ≠ 0)
    (hzk : evidence p (fun i => l i*k i) ≠ 0) :
    update (update p l) k = update p (fun i => l i*k i) := by
  funext i
  rw [update, sequential_evidence]
  dsimp [update]
  field_simp
```

An application with nonnegative priors and likelihoods must establish positive evidence before using probability interpretations. Normalization does not manufacture an observation with positive probability.

4.2 Perturbation has a denominator

The weighted ℓ^1 likelihood error is divided by the minimum of the two positive evidences. This gives a usable stability estimate while exposing its sensitivity to nearly impossible observations.

Source: KomorebiPaperVI/FiniteBayes.lean.

```
theorem likelihood_perturbation_bound (p l k : I → ℝ)
    (hp : ∀ i, 0 ≤ p i) (_hl : ∀ i, 0 ≤ l i) (hk : ∀ i, 0 ≤ k i)
    (hZl : 0 < evidence p l) (hZk : 0 < evidence p k) :
    l1 (fun i => update p l i-update p k i) ≤
      2*(∑ i, |l i-k i| *p i)/min (evidence p l) (evidence p k) := by
  have he : l1 (fun i => l i*p i-k i*p i) = ∑ i, |l i-k i| *p i := by
    simp only [l1, ← sub_mul, abs_mul, abs_of_nonneg (hp _)]
  have h := normalized_l1_bound_left (fun i => l i*p i) (fun i => k i*p i)
    (fun i => mul_nonneg (hk i) (hp i)) hZl hZk
  change l1 (fun i => update p l i-update p k i) ≤
    2*l1 (fun i => l i*p i-k i*p i)/evidence p l at h
  rw [he] at h
  exact h.trans (div_le_div_of_nonneg_left
    (mul_nonneg (by norm_num) (Finset.sum_nonneg (fun i _ => mul_nonneg (abs_nonneg _) (hp i))))
    (lt_min hZl hZk) (min_le_left _ _))
```

The finite numerical examples can check this inequality for chosen data. They do not replace the universally quantified source theorem or supply an evidence lower bound for a different model.

4.3 Effective likelihood and fixed coarse likelihood

Given a prior p, the effective coarse likelihood averages the fine likelihood within each fiber using that prior. The factorization theorem includes zero-prior-mass fibers. The proof separates these cases rather than silently dividing by their mass.

Source: `KomorebiPaperVI/Coarsening.lean`.

```
theorem effective_factorization (f : I → A) (p l : I → ℝ) (hp : ∀ i, 0 ≤ p i) :
    coarsen f (update p l) = update (coarsen f p) (effective f p l) := by
  rw [coarsen_posterior]
  funext a
  simp only [update, effective_weight f p l hp, effective_evidence f p l hp]
```

The effective likelihood depends on the prior. A single prescribed coarse likelihood that works for every prior is a stronger requirement. The necessary-and-sufficient theorem derives that requirement using actual two-point priors. With at least two coarse states, surjectivity, and strict positivity, fine likelihoods must agree with the pulled-back coarse likelihood up to one positive global scale:

Source: `KomorebiPaperVI/BayesianNecessity.lean`.

```
theorem refinement_iff (f : I → A) (hf : Function.Surjective f) [Nontrivial A]
    (l : I → ℝ) (k : A → ℝ) (hl : ∀ i, 0 < l i) (hk : ∀ a, 0 < k a) :
    AllPriorCommutes f l k ↔ ∃ c : ℝ, 0 < c ∧ ∀ i, l i=c*k (f i) := by
  constructor
  · intro h
    obtain ⟨a⟩ := (inferInstance : Nonempty A)
    obtain ⟨i,hi⟩ := hf a
    refine ⟨l i/k (f i), div_pos (hl i) (hk (f i)), ?_⟩
    intro j
    have he := all_packet_ratio f hf l k hl hk h j i
    apply (eq_div_iff (ne_of_gt (hk (f i)))).mpr at he
    calc
      l j = (l i*k (f j))/k (f i) := he
      _ = _ := by ring
  · rintro ⟨c,hc,he⟩ p hp
    have heq : l=fun i => c*k (f i) := funext he
    rw [heq]
    exact scaled_blockwise_sufficiency f p k c hc.ne'
      (positive_evidence p (k ∘ f) hp (fun i => hk (f i))).ne'
```

The nontriviality assumption is part of the mathematics. If all mass is sent to one packet, every valid posterior aggregates to the unique unit probability, so commutation no longer forces likelihood constancy:

Source: KomorebiPaperVI/BayesianTowers.lean.

```
theorem singleton_commutes (f : I → A) (l : I → ℝ) (k : A → ℝ)
    (hl : ∀ i, 0 < l i) (hk : ∀ a, 0 < k a) [Subsingleton A] :
    AllPriorCommutes f l k := by
  intro p hp
  funext a
  have hs (v : A → ℝ) : mass v=v a :=
    Finset.sum_eq_single a (fun b _ hb => False.elim (hb (Subsingleton.elim b a))) (by simp)
  have h1 : mass (coarsen f (update p l))=1 :=
    (coarsen_mass f _).trans (posterior_mass p l (positive_evidence p l hp hl).ne')
  have h2 := posterior_mass (coarsen f p) k
    (positive_evidence _ k (coarsen_probability f p hp) hk).ne'
  exact (hs _).symm.trans (h1.trans (h2.symm.trans (hs _)))
```

The adjacent tower criterion includes this singleton alternative. Composition then gives exact commutation over a finite span of varying state spaces.

4.4 The coarsest meaningful partition

`BayesianPartitions` forms a quotient by equality of the whole likelihood profile. Its canonical coarse likelihood recovers each fine likelihood. Under strict positivity, every nontrivial all-prior-closed partition refines the profile partition: the canonical profile map factors uniquely through the given partition map:

Source: KomorebiPaperVI/BayesianPartitions.lean.

```
theorem canonical_coarsest_nontrivial (l : Y → I → ℝ) (hl : ∀ y i, 0 < l y i)
    (f : I → A) (hf : Function.Surjective f) [Nontrivial A] (h : FamilyCloses l f) :
    ∃! u : A → Packet l, ∀ i, u (f i)=packetMap l i :=
  canonical_unique_factor l f hf ((all_prior_family_iff_resolves l hl f hf).mp h)
```

The qualifier "nontrivial" excludes the automatic one-packet case. The theorem does not claim that every family has a nontrivial canonical quotient: a separate declaration supplies nontriviality when two profiles differ. A broader statistical sufficiency statement requires its own statistical model and hypotheses.

4.5 Information loss is a specific matrix

The formal information-loss matrix is $\mathrm{diag}(P) - bP^\top$, where P is coarse prior mass and b is the coarse posterior. It acts on the mismatch between the effective and prescribed coarse likelihoods. The exact factorization is:

Source: KomorebiPaperVI/InformationLoss.lean.

```
theorem information_loss_matrix (f : I → A) (p l : I → ℝ) (k : A → ℝ)
    (hp : ∀ i, 0 ≤ p i) (hz : evidence p l ≠ 0)
    (hk : evidence (coarsen f p) k ≠ 0) (a : A) :
    coarsen f (update p l) a-update (coarsen f p) k a =
      (lossMatrix (coarsen f p) (update (coarsen f p) k)).mulVec
```

```
      (fun j => effective f p l j-k j) a/evidence p l := by
  rw [effective_factorization f p l hp, lossMatrix_apply]
  have he := posterior_difference (coarsen f p) (effective f p l) k
    (by rwa [effective_evidence f p l hp]) hk a
  simpa only [effective_evidence f p l hp] using he
```

The scale direction is in its nullspace because multiplying a likelihood by a nonzero scalar does not change its normalized posterior. The norm estimate then combines this factorization with the likelihood perturbation bound. This is the matrix-valued defect used in core 8.4. Relative-entropy chain rules in Part II are different claims and are not implied by the shared word "information."

4.6 A normalized exponential flow

For a score function g, $p_i(t)$ is proportional to $p_i(0)e^{tg_i}$. The source establishes positive evidence, simplex preservation, the additive-time law, and the centered multiplicative derivative:

Source: `KomorebiPaperVI/BayesianFlow.lean`.

```
theorem flow_derivative (p g : I → ℝ) (hp : IsProbability p) (t : ℝ) (i : I) :
    HasDerivAt (fun u => flow p g u i)
      ((g i-(∑ j, g j*flow p g t j))*flow p g t i) t := by
  have hz := (positive_evidence p g hp t).ne'
  have hd := ((likelihood_derivative g t i).mul_const (p i)).div
    (evidence_derivative p g t) hz
  convert! hd using 1
  simp only [flow, update]
  simp only [← mul_div_assoc, ← Finset.sum_div]
  field_simp
```

The derivative contains the posterior mean score. It is therefore nonlinear in the probability vector even though unnormalized multiplication is linear. The additional covariance, cumulant, concentration, and filter stability results retained in the traditional edition have separate scope rows in Section 10.

5 Fractional calculus and refinement towers

5.1 Two constructions with different starting data

`SpectralPackets` starts with a supplied finite spectral resolution. Its finite sums give multiplication, packet reduction, and the actual exponential. It does not silently prove a spectral decomposition for every operator. The additive-exponent law explicitly allows zero eigenvalues while requiring positive exponents:

Source: `KomorebiPaperVI/SpectralPackets.lean`.

```
theorem fractional_add (P : I → R) (h : ∀ i j, P i*P j=if i=j then P i else 0)
    (lam : I → ℝ) (hl : ∀ i, 0 ≤ lam i) (a b : ℝ) (ha : 0 < a) (hb : 0 < b) :
    fractional P lam a*fractional P lam b=fractional P lam (a+b) := by
  unfold fractional
  rw [resolve_mul P h]
  congr 1
  funext i
  exact (Real.rpow_add' (hl i) (add_pos ha hb).ne').symm
```

The bounded Hilbert-space refinement result uses continuous functional calculus and a normalized Bochner resolvent integral. This is a different route and has a broader operator domain than a finite spectral list. The ledger records the exact setting for each result.

5.2 Poisson subordination includes the integral

The square-root transfer is recovered from the heat transfer using the normalized Poisson kernel. `ReciprocalGaussian` proves the scalar integral, `PoissonKernel` supplies integrability, and `PoissonSubordination` transports the scalar identities across a finite spectral sum. Its endpoint asserts both integrability and equality with the actual exponential:

Source: `KomorebiPaperVI/PoissonSubordination.lean`.

```
theorem operator_subordination (P : I → R) (hP : ∑ i,P i=1)
    (h : ∀ i j,P i*P j=if i=j then P i else 0) (lam : I → ℝ)
    (hl : ∀ i,0 ≤ lam i) (t : ℝ) (ht : 0 < t) :
    IntegrableOn (fun s => kernel t s • NormedSpace.exp ((-s) • resolve P lam)) (Ioi 0) ∧
    NormedSpace.exp ((-t) • fractional P lam (1/2))=
      ∫ s in Ioi 0,kernel t s • NormedSpace.exp ((-s) • resolve P lam) := by
  simp only [operator_integrand P hP h lam]
  refine ⟨finite_spectral_integrable P lam hl t ht,?_⟩
  rw [finite_spectral_subordination P lam hl t ht,spectral_transfer_exponential P hP h]
  simp only [spectralTransfer,resolve,Real.sqrt_eq_rpow]
```

The integral formula is stated for $t > 0$. The semigroup value at $t = 0$ is obtained separately from the exponential definition. Nonnegative spectral values include zero. General stable-subordinator statements for arbitrary fractional exponents in Part II are not new endpoints of this square-root formal theorem.

5.3 The normalization constant is proved

For $0 < \alpha < 1$, the resolvent representation uses $\sin(\pi\alpha)/\pi$. `BetaNormalization` derives the scalar integrals and constants. In particular, the common-gap integral evaluates exactly to $\alpha m^{\alpha-1}$ when $m > 0$:

Source: `KomorebiPaperVI/BetaNormalization.lean`.

```
theorem normalized_gap_integral (p m : ℝ) (hp : p∈Ioo 0 1) (hm : 0 < m) :
    (Real.sin (Real.pi*p)/Real.pi)*(∫ t in Ioi 0,t^p/(t+m)^2)=p*m^(p-1) := by
  rw [gap_integral p m hp hm]
  have hs : Real.sin (Real.pi*p) ≠ 0 := by
    intro hs
    have h := sine_factor_positive p hp
    rw [hs,zero_div] at h
    exact (lt_irrefl 0) h
  field_simp
```

This identity supplies the sharp constant in the displayed common-gap certificate. It is not left as an unnamed positive normalization factor.

5.4 The rectangular resolvent defect

The spaces E and F in `RectangularFractional` are complete complex Hilbert spaces. A and B are bounded nonnegative self-adjoint operators in the operator order, and $C : E \to F$ is bounded. Put

$$D = CA - BC, \qquad D_\alpha = CA^\alpha - B^\alpha C.$$

The theorem records the actual inverse-valued integrand and its Bochner integrability:

Source: `KomorebiPaperVI/RectangularFractional.lean`.

```
theorem literal_resolvent_integral (p : ℝ≥0) (hp : (p:ℝ)∈Ioo 0 1)
    (A : E →L[ℂ] E) (B : F →L[ℂ] F) (C : E →L[ℂ] F) (hA : 0 ≤ A) (hB : 0 ≤ B) :
    let f := fun t : ℝ => t^(p:ℝ) • (Ring.inverse (algebraMap ℝ (F →L[ℂ] F) t+B)).comp
      ((defect A B C).comp (Ring.inverse (algebraMap ℝ (E →L[ℂ] E) t+A)))
    IntegrableOn f (Ioi 0) ∧ fractionalDefect A B C p=coefficient p • ∫ t in Ioi 0,f t := by
  dsimp only
  have he : (fun t : ℝ => t^(p:ℝ) • (Ring.inverse (algebraMap ℝ (F →L[ℂ] F) t+B)).comp
      ((defect A B C).comp (Ring.inverse (algebraMap ℝ (E →L[ℂ] E) t+A))))
      =ᵐ[volume.restrict (Ioi 0)] integrand A B C p := by
    filter_upwards [ae_restrict_mem measurableSet_Ioi] with t ht
    rw [literal_integrand p t hp ht A B C hA hB,
      PositiveResolvent.resolvent_inverse A hA t ht,PositiveResolvent.resolvent_inverse B hB t ht]
  have h := normalized_integral p hp A B C hA hB
  exact ⟨h.1.congr he.symm,h.2.trans (congrArg (coefficient p • ·) (integral_congr_ae he.symm))⟩
```

The sign matches the convention $D = CA - BC$. In this formula the resolvents are $(tI + B)^{-1}$ and $(tI + A)^{-1}$, and the order of the sandwich is fixed by the rectangular domain and codomain.

5.5 A gap-free estimate and a common-gap estimate

The gap-free estimate combines low- and high-parameter bounds on the same integrand. A split at the appropriate scale gives a Hölder estimate with the explicit constant defined in `FractionalNormBounds.holderConstant`. The source handles $C = 0$ and $D = 0$ before dividing by their norms.

Source: `KomorebiPaperVI/RectangularFractional.lean`.

```
theorem holder_estimate (p : ℝ≥0) (hp : (p:ℝ)∈Ioo 0 1)
    (A : E →L[ℂ] E) (B : F →L[ℂ] F) (C : E →L[ℂ] F) (hA : 0 ≤ A) (hB : 0 ≤ B) :
    ‖fractionalDefect A B C p‖ ≤
      FractionalNormBounds.holderConstant p*‖C‖^(1-(p:ℝ))*‖defect A B C‖^(p:ℝ) := by
  by_cases hc : C=0
  · subst C
    simp [fractionalDefect,defect,Real.zero_rpow hp.1.ne']
  by_cases hd : defect A B C=0
  · have he := exact_fractional_intertwining p hp A B C hA hB hd
    rw [he,norm_zero,hd]
    simp [Real.zero_rpow hp.1.ne']
  have hcn : 0 < ‖C‖ := norm_pos_iff.mpr hc
  have hdn : 0 < ‖defect A B C‖ := norm_pos_iff.mpr hd
  have hf := normalized_integral p hp A B C hA hB
  have hk : 0 < coefficient p := BetaNormalization.sine_factor_positive p hp
  have hbound := FractionalScalarBounds.split_norm_bound (integrand A B C p) p (2*‖C‖) ‖defect A B
   ↪ C‖
    (‖defect A B C‖/(2*‖C‖)) hp (by positivity) hf.1
    (fun t ht => low_bound p t hp ht A B C hA hB)
    (fun t ht => high_bound p t ht hp A B C hA hB)
  rw [FractionalScalarBounds.optimize_split p (2*‖C‖) ‖defect A B C‖ hp (by positivity) hdn] at
   ↪ hbound
  rw [hf.2,norm_smul,Real.norm_of_nonneg hk.le]
  calc
    _ ≤ coefficient p*((2*‖C‖)^(1-(p:ℝ))*‖defect A B C‖^(p:ℝ)/((p:ℝ)*(1-(p:ℝ)))) :=
      mul_le_mul_of_nonneg_left hbound hk.le
    _ = _ := by
      rw [Real.mul_rpow (by norm_num : (0:ℝ) ≤ 2) (norm_nonneg C)]
      unfold FractionalNormBounds.holderConstant
      ring
```

With a common lower spectral bound $mI \le A, B$ and $m > 0$, the endpoint is linear in the original defect:

Source: `KomorebiPaperVI/RectangularFractional.lean`.

```
theorem common_gap_estimate (p : ℝ≥0) (hp : (p:ℝ)∈Ioo 0 1) (m : ℝ) (hm : 0 < m)
    (A : E →L[ℂ] E) (B : F →L[ℂ] F) (C : E →L[ℂ] F) (hA : 0 ≤ A) (hB : 0 ≤ B)
    (hgA : algebraMap ℝ (E →L[ℂ] E) m ≤ A) (hgB : algebraMap ℝ (F →L[ℂ] F) m ≤ B) :
    ‖fractionalDefect A B C p‖ ≤ (p:ℝ)*m^((p:ℝ)-1)*‖defect A B C‖ := by
  have hd := normalized_integral p hp A B C hA hB
  have hk : 0 < coefficient p := BetaNormalization.sine_factor_positive p hp
  have hs : 0 < Real.sin (Real.pi*(p:ℝ)) := by
    apply Real.sin_pos_of_pos_of_lt_pi
    · exact mul_pos Real.pi_pos hp.1
```

```
    · nlinarith [Real.pi_pos,hp.2]
  have hg : IntegrableOn (fun t : ℝ => t^(p:ℝ)/(t+m)^2) (Ioi 0) := by
    by_contra hn
    have hz := integral_undef hn
    rw [BetaNormalization.gap_integral p m hp hm] at hz
    exact (mul_pos (Real.rpow_pos_of_pos hm _) (div_pos (mul_pos hp.1 Real.pi_pos) hs)).ne' hz
  rw [hd.2,norm_smul,Real.norm_of_nonneg hk.le]
  calc
    _ ≤ coefficient p*(∫ t in Ioi 0,‖integrand A B C p t‖) := by
      apply mul_le_mul_of_nonneg_left (norm_integral_le_integral_norm _) hk.le
    _ ≤ coefficient p*(∫ t in Ioi 0,‖defect A B C‖*(t^(p:ℝ)/(t+m)^2)) := by
      apply mul_le_mul_of_nonneg_left _ hk.le
      apply setIntegral_mono_on hd.1.norm (hg.const_mul _) measurableSet_Ioi
      intro t ht
      exact gap_bound p t m ht hm.le hp A B C hA hB hgA hgB
    _ = _ := by
      rw [integral_const_mul,mul_left_comm,coefficient,BetaNormalization.normalized_gap_integral p
       ↪  m hp hm]
      ring
```

Real finite-matrix applications use the ordinary complexification specialization disclosed in the inherited review. The project does not count that manual specialization as a separately formalized construction.

5.6 The sign in fractional evolution

Fractional evolution is e^{-tA^α}. Its Duhamel formula has a minus sign relative to the e^{tA} convention. Combining decay with the common-gap estimate yields the final generator-level bound:

Source: `KomorebiPaperVI/RectangularSemigroup.lean`.

```
theorem generator_gap_bound (A : E →L[ℂ] E) (B : F →L[ℂ] F) (C : E →L[ℂ] F)
    (hA : 0 ≤ A) (hB : 0 ≤ B) (p : ℝ≥0) (hp : (p:ℝ)∈Ioo 0 1) (t m : ℝ) (ht : 0 ≤ t) (hm : 0 < m)
    (hgA : algebraMap ℝ (E →L[ℂ] E) m ≤ A) (hgB : algebraMap ℝ (F →L[ℂ] F) m ≤ B) :
    ‖timeDefect A B C p t‖ ≤ t*Real.exp (-t*m^(p:ℝ))*(p:ℝ)*m^((p:ℝ)-1)*‖defect A B C‖ := by
  calc
    _ ≤ t*Real.exp (-t*m^(p:ℝ))*‖fractionalDefect A B C p‖ :=
      fractional_gap_bound A B C hA hB p hp.1 t m ht hm.le hgA hgB
    _ ≤ t*Real.exp (-t*m^(p:ℝ))*((p:ℝ)*m^((p:ℝ)-1)*‖defect A B C‖) :=
      mul_le_mul_of_nonneg_left (common_gap_estimate p hp m hm A B C hA hB hgA hgB) (by
       ↪  positivity)
    _ = _ := by ring
```

The bound vanishes when the generator defect vanishes. It assumes nonnegative time and the same positive common gap as the fractional power estimate. It does not establish norm convergence for arbitrary unbounded generators.

5.7 What is summed across varying spaces

In `RectangularTowers`, H_n may vary with n, and $C_n : H_{n+1} \to H_n$. The individual defects therefore lie in different operator spaces. The formal series is a series of *real norms*. The endpoint says its partial sums converge uniformly on bounded nonnegative time intervals:

Source: `KomorebiPaperVI/RectangularTowers.lean`.

```
theorem bounded_time_uniform_convergence (A : ∀ n,H n →L[ℂ] H n) (C : ∀ n,H (n+1) →L[ℂ] H n)
    (hA : ∀ n,0 ≤ A n) (p : ℝ≥0) (hp : 0 < p) (T : ℝ)
    (hs : Summable (fractionalNorm A C p)) :
    TendstoUniformlyOn (fun N t => ∑ n ∈ Finset.range N,timeNorm A C p t n)
      (fun t => ∑' n,timeNorm A C p t n) Filter.atTop (Icc 0 T) := by
  apply tendstoUniformlyOn_tsum_nat (hs.mul_left T)
  intro n t ht
  rw [Real.norm_of_nonneg (show 0 ≤ timeNorm A C p t n from norm_nonneg _)]
  exact bounded_time_majorant A C hA p hp T t ht n
```

Two sufficient generator criteria precede this theorem: summability of the α-powers of generator-defect norms with uniformly bounded C_n, or ordinary summability with a common positive spectral gap. An inherited fixed-space formulation also proves operator-series convergence. It must not be relabeled as an operator sum across unrelated Hom spaces. A coherent limiting evolution additionally needs the ambient completion and closure hypotheses discussed in the retained written extensions.

6 Green reconstruction and finite paths

6.1 The inverse is the literal inverse

`DiscreteGreen` builds the Dirichlet tridiagonal matrix on `Fin n`. It proves a second-difference identity for the kernel, treats both boundary terms, multiplies the matrix by that kernel, and then identifies the actual nonsingular inverse. The endpoint is short because the preceding multiplication identity carries the main work:

Source: `KomorebiPaperVI/DiscreteGreen.lean`.

```
theorem inverse_eq_kernel (n : ℕ) : (tridiagonal n)⁻¹=inverseKernel n := by
  exact Matrix.inv_eq_right_inv (tridiagonal_inverse_kernel n)
```

For $h = 1/(n+1)$ and $L_h = h^{-2}\,\mathrm{tridiag}(-1, 2, -1)$, the precise scaling is

$$h^{-1}(L_h^{-1})_{ij} = G(x_i, x_j), \qquad G(x, y) = \min(x, y) - xy.$$

The formal statement uses zero-based `Fin n` indices and adds one to obtain the interior nodes:

Source: `KomorebiPaperVI/DiscreteGreen.lean`.

```
theorem grid_reconstruction (n : ℕ) (i j : Fin n) :
    ((n:ℝ)+1)*(laplacian n)⁻¹ i j=
      min (((i.val:ℝ)+1)/((n:ℝ)+1)) (((j.val:ℝ)+1)/((n:ℝ)+1))-
        (((i.val:ℝ)+1)/((n:ℝ)+1))*(((j.val:ℝ)+1)/((n:ℝ)+1)) := by
  rw [laplacian_inverse]
  simp only [Matrix.smul_apply,smul_eq_mul,inverseKernel,green,Nat.cast_add,Nat.cast_one]
  rw [min_div_div_right (by positivity)]
  field_simp
```

This is an exact node identity. It is not by itself a uniform convergence theorem for an arbitrary continuous forcing function or a higher-dimensional mesh. Those quadrature and PDE estimates need the separate regularity assumptions stated in the traditional paper.

6.2 Low modes have an explicit error bound

`LowModeConvergence` proves a scalar sine-square estimate and substitutes it into the discrete eigenvalue formula. It gives

$$0 \le \omega^2 - \frac{4}{h^2}\sin^2\!\left(\frac{\omega h}{2}\right) \le \frac{\omega^4 h^2}{12}, \qquad h \neq 0.$$

The Lean endpoint retains the nonzero-spacing hypothesis:

Source: `KomorebiPaperVI/LowModeConvergence.lean`.

```
theorem low_mode_error (w h : ℝ) (hh : h≠0) :
    0 ≤ w^2-eigenvalue w h ∧ w^2-eigenvalue w h ≤ w^4/12*h^2 := by
  have hs := sine_square_error (w*h/2)
  have hpos : 0 ≤ 4/h^2 := by positivity
  have he : w^2-eigenvalue w h=4/h^2*((w*h/2)^2-Real.sin (w*h/2)^2) := by
```

```
    unfold eigenvalue
    field_simp
    ring
  rw [he]
  constructor
  · exact mul_nonneg hpos hs.1
  · calc
      _ ≤ 4/h^2*((w*h/2)^4/3) := mul_le_mul_of_nonneg_left hs.2 hpos
      _ = w^4/12*h^2 := by field_simp; ring
```

For a fixed low mode, set $\omega = k\pi$ and let positive h tend to zero. The constant depends on that fixed ω. This statement does not make high-frequency error uniform when k grows with mesh resolution.

6.3 Probability columns and observable rows

For the finite path formula, K_{ji} is the transition probability from i to j. An observable f propagates by $K^\top$, and the potential is evaluated at the departure state. Therefore the source transfer is

$$F = \operatorname{diag}(e^{-V})K^\top.$$

Source: KomorebiPaperVI/FeynmanKac.lean.

```
noncomputable def transfer (K : Matrix I I ℝ) (V : I → ℝ) : Matrix I I ℝ :=
  Matrix.diagonal (fun i => Real.exp (-V i))*K.transpose
```

The path law is defined recursively for exactly n steps. At $n = 0$ there is an empty path, path probability one, zero accumulated potential, and the starting state as terminal state. Induction on the path length gives the matrix-power formula:

Source: KomorebiPaperVI/FeynmanKac.lean.

```
theorem feynman_kac_path_formula (K : Matrix I I ℝ) (V f : I → ℝ) (n : ℕ) :
    (transfer K V ^ n).mulVec f=pathExpectation K V f n := by
  induction n with
  | zero => funext i; simp [path_expectation_zero]
  | succ n ih =>
    rw [pow_succ',← Matrix.mulVec_mulVec,ih]
    funext i
    rw [path_expectation_step]
    simp only [Matrix.mulVec,dotProduct,transfer_entry,Finset.mul_sum,mul_assoc]
```

The algebraic equality itself does not require stochastic hypotheses. To call the path weights a probability law, use the separate declarations proving nonnegativity and total mass one from entrywise nonnegative K and unit column sums. Separate product, potential-sum, and terminal-index theorems ensure that the recursive law is the intended finite path expression. The continuous-time Feynman–Kac and product-slicing extensions retained in Parts II and III remain outside this discrete endpoint.

7 Schur elimination and nonnormal memory

7.1 A projected resolvent needs two invertibility facts

For retained and hidden coordinates $P \oplus Q$, the Schur complement is

$$S(z) = zI - A_{PP} - A_{PQ}(zI - A_{QQ})^{-1}A_{QP}.$$

`SchurFeshbach` uses literal complex block matrices. The projected inverse formula assumes invertibility of both the full resolvent matrix and the hidden block:

Source: `KomorebiPaperVI/SchurFeshbach.lean`.

```
theorem projected_resolvent (App : Matrix P P ℂ) (Apq : Matrix P Q ℂ)
    (Aqp : Matrix Q P ℂ) (Aqq : Matrix Q Q ℂ) (z : ℂ)
    [Invertible (Matrix.scalar Q z-Aqq)]
    [Invertible (Matrix.scalar (P ⊕ Q) z-Matrix.fromBlocks App Apq Aqp Aqq)] :
    ((Matrix.scalar (P ⊕ Q) z-Matrix.fromBlocks App Apq Aqp Aqq)⁻¹).submatrix
      Sum.inl Sum.inl = (schur App Apq Aqp Aqq z)⁻¹ := by
  let D := Matrix.scalar Q z-Aqq
  let A := Matrix.scalar P z-App
  have he := resolvent_blocks App Apq Aqp Aqq z
  let : Invertible (Matrix.fromBlocks A (-Apq) (-Aqp) D) := Invertible.copy ‹_› _ he.symm
  let : Invertible (A-(-Apq)*⅟D*(-Aqp)) := Matrix.invertibleOfFromBlocks₂₂Invertible A (-Apq)
   ↪  (-Aqp) D
  have hinv := Matrix.invOf_fromBlocks₂₂_eq A (-Apq) (-Aqp) D
  have hs : A-(-Apq)*⅟D*(-Aqp)=schur App Apq Aqp Aqq z := by
    simp only [Matrix.neg_mul,Matrix.mul_neg,neg_neg,Matrix.invOf_eq_nonsing_inv,schur,D,A]
  rw [he,← Matrix.invOf_eq_nonsing_inv,hinv]
  change ⅟(A-(-Apq)*⅟D*(-Aqp))=(schur App Apq Aqp Aqq z)⁻¹
  rw [Matrix.invOf_eq_nonsing_inv]
  congr 1
```

The module also proves the invertibility equivalence between the full block matrix and its Schur complement when the hidden block is invertible. The retained block is an inverse Schur complement, not simply the resolvent of A_{PP}. The difference is the self-energy $A_{PQ}(zI - A_{QQ})^{-1}A_{QP}$.

7.2 The fifth checked block removes a normality shortcut

The earlier memory layer establishes Laplace inversion from a supplied exponential growth estimate and also treats normal elements. The latest core 13.2 goes further for arbitrary finite complex matrices: a strict spectral half-plane alone yields integrability and decay. Nonnormality can create polynomial factors in time, so the proof must account for nilpotent generalized eigenspace contributions.

The scalar input is integrability of a complex exponential times a polynomial whenever the real part of the exponent is negative:

Source: `KomorebiPaperVI/ScalarDecay.lean`.

```
theorem exp_pow_integrable (a : ℂ) (ha : a.re < 0) (k : ℕ) :
    IntegrableOn (fun t : ℝ => Complex.exp (a*t)*(t:ℂ)^k) (Ioi 0) := by
  have hi := integrableOn_rpow_mul_exp_neg_mul_rpow
```

```
    (s := (k:ℝ)) (p := (1:ℝ)) (b := -a.re) (by linarith [Nat.cast_nonneg (α := ℝ) k]) zero_lt_one
    ↪ (by linarith)
  simp only [Real.rpow_natCast,Real.rpow_one,neg_neg] at hi
  apply hi.mono' (by fun_prop)
  filter_upwards [ae_restrict_mem measurableSet_Ioi] with t ht
  rw [norm_exp_pow a k t ht.le]
```

There is also a separate zero-limit theorem. Neither a numerical quadrature nor an assertion about eigenvalue locations substitutes for these analytic steps.

7.3 From generalized eigenspaces to a finite sum

`NonnormalMemory` studies left multiplication by A on a finite-dimensional complex C^*-algebra. A complex matrix algebra is a specialization. The source decomposes this vector space into generalized eigenspaces. On one component the exponential is a finite polynomial times $e^{\mu t}$:

Source: `KomorebiPaperVI/NonnormalMemory.lean`.

```
theorem exp_mul_component (A : R) (μ : ℂ) (x : R)
    (hx : x ∈ (Module.End.maxGenEigenspace (LinearMap.mulLeft ℂ A)) μ) (t : ℝ) :
    transfer A t*x=Complex.exp (μ*t) •
      ∑ k ∈ Finset.range (finrank ℂ R),((t:ℂ)^k/(k.factorial:ℂ)) • ((A-μ • 1)^k*x) := by
  classical
  let : NormedAlgebra ℚ R := NormedAlgebra.restrictScalars ℚ ℝ R
  let D := A-μ • 1
  have hkill (k : ℕ) (hk : finrank ℂ R ≤ k) : D^k*x=0 := by
    rw [← left_shift_power]
    exact CanonicalJets.power_kills _ μ x hx k hk
  have hs := (exp_series_hasSum_exp' (𝕂 := ℂ) ((t:ℂ) • D)).mul_right x
  have he : exp ((t:ℂ) • D)*x=
      ∑ k ∈ Finset.range (finrank ℂ R),((t:ℂ)^k/(k.factorial:ℂ)) • (D^k*x) := by
    have hs' : HasSum (fun k : ℕ => ((t:ℂ)^k/(k.factorial:ℂ)) • (D^k*x))
        (exp ((t:ℂ) • D)*x) := by
      simpa only [smul_pow,Algebra.smul_mul_assoc,smul_smul,div_eq_mul_inv,mul_comm] using hs
    exact hs'.unique (hasSum_sum_of_ne_finset_zero (fun k hk => by
      rw [hkill k (by simpa using hk),smul_zero]))
  have ht : t • A=algebraMap ℂ R (μ*t)+(t:ℂ) • D := by
    simp only [D,smul_sub,Algebra.algebraMap_eq_smul_one,smul_smul]
    rw [RCLike.real_smul_eq_coe_smul (K := ℂ),mul_comm μ (t:ℂ)]
    abel
  rw [transfer,ht,exp_add_of_commute (Algebra.commutes _ _),
    ← algebraMap_exp_comm,← Complex.exp_eq_exp_ℂ,Algebra.algebraMap_eq_smul_one,
    Algebra.smul_mul_assoc,one_mul,Algebra.smul_mul_assoc,he]
```

The cutoff is the complex dimension of the ambient algebra. For an $n \times n$ matrix algebra that can be n^2. It need not be a minimal Jordan-block length. The proof only needs a valid nilpotence cutoff. Terms beyond that cutoff vanish on the component.

After damping by e^{-zt}, each coefficient has exponent $\mu - z$ with negative real part. The source combines the scalar integrability and limit results over finite sums and then over the generalized-eigenspace decomposition. Applying the resulting identity to the algebra unit gives the operator-valued decay:

Source: `KomorebiPaperVI/NonnormalMemory.lean`.

```
theorem spectral_decay (A : R) (z : ℂ) (hz : ∀ μ∈spectrum ℂ A,μ.re < z.re) :
    IntegrableOn (damped A z) (Ioi 0) ∧ Tendsto (damped A z) atTop (𝓝 0) := by
  have h (x : R) : IntegrableOn (fun t => damped A z t*x) (Ioi 0) ∧
      Tendsto (fun t => damped A z t*x) atTop (𝓝 0) := by
    apply Submodule.iSup_induction (Module.End.maxGenEigenspace (LinearMap.mulLeft ℂ A))
      (x := x) (motive := fun x => IntegrableOn (fun t : ℝ => damped A z t*x) (Ioi 0) volume ∧
        Tendsto (fun t => damped A z t*x) atTop (𝓝 0))
      (by rw [Module.End.iSup_maxGenEigenspace_eq_top]; trivial)
    · exact component_decay A z hz
    · simp
    · intro x y hx hy
      simpa only [Pi.add_def,mul_add,add_zero] using And.intro (hx.1.add hy.1) (hx.2.add hy.2)
  simpa using h 1
```

The condition is strict: $\operatorname{Re}\mu < \operatorname{Re} z$ for every spectral value. The conclusion includes both Bochner integrability and the limit zero. It does not rely on a claimed universal bound $\|e^{tA}\| \leq e^{t \max \operatorname{Re}\sigma(A)}$, which would fail for general nonnormal matrices.

7.4 Why both derivative identities appear

The improper-integral fundamental theorem is applied once to left multiplication and once to right multiplication. These establish both inverse equations. This is what connects the integral to the actual inverse, rather than just presenting a formal solution to one equation.

Source: `KomorebiPaperVI/NonnormalMemory.lean`.

```
theorem spectral_laplace_inverse (A : R) (z : ℂ)
    (hz : ∀ μ∈spectrum ℂ A,μ.re < z.re) :
    IntegrableOn (damped A z) (Ioi 0) ∧ IsUnit (algebraMap ℂ R z-A) ∧
      (∫ t in Ioi 0,damped A z t)=Ring.inverse (algebraMap ℂ R z-A) := by
  have hs := spectral_decay A z hz
  let B := A-algebraMap ℂ R z
  have he : damped A z=transfer B := funext (damped_eq_shifted A z)
  rw [he] at hs ⊢
  have hl := integral_Ioi_of_hasDerivAt_of_tendsto'
    (fun t (_ : t∈Ici (0:ℝ)) => transfer_derivative_left B t)
    ((ContinuousLinearMap.mul ℝ R B).integrable_comp hs.1) hs.2
  have hr := integral_Ioi_of_hasDerivAt_of_tendsto'
    (fun t (_ : t∈Ici (0:ℝ)) => transfer_derivative_right B t)
    (((ContinuousLinearMap.mul ℝ R).flip B).integrable_comp hs.1) hs.2
  rw [integral_const_mul_of_integrable hs.1,transfer_zero,zero_sub] at hl
  rw [integral_mul_const_of_integrable hs.1,transfer_zero,zero_sub] at hr
  have hn : algebraMap ℂ R z-A= -B := by dsimp [B]; abel
  have hl' : (algebraMap ℂ R z-A)*(∫ t in Ioi 0,transfer B t)=1 := by
    rw [hn,neg_mul,hl,neg_neg]
  have hr' : (∫ t in Ioi 0,transfer B t)*(algebraMap ℂ R z-A)=1 := by
    rw [hn,mul_neg,hr,neg_neg]
  have hu : IsUnit (algebraMap ℂ R z-A) := ⟨⟨_,_,hl',hr'⟩,rfl⟩
  refine ⟨hs.1,hu,?_⟩
  calc
```

```
    _ = Ring.inverse (algebraMap ℂ R z-A)*((algebraMap ℂ R z-A)*(∫ t in Ioi 0,transfer B t)) :=
      (Ring.inverse_mul_cancel_left _ _ hu).symm
    _ = _ := by rw [hl',mul_one]
```

The result supplies invertibility as part of the conclusion. The total operation `Ring.inverse` is therefore being used at an established unit. This is the formal endpoint connecting the strict spectral half-plane with a genuine resolvent formula.

7.5 Retained and hidden dimensions can differ

The final matrix theorem uses A on hidden coordinates J, $U : J \to I$, and $V : I \to J$. It wraps $B \mapsto UBV$ as a continuous linear map and commutes that map with the Bochner integral. The result is the literal rectangular memory/self-energy identity:

Source: `KomorebiPaperVI/MatrixMemory.lean`.

```
theorem rectangular_memory_self_energy (A : Matrix J J ℂ)
    (U : Matrix I J ℂ) (V : Matrix J I ℂ) (z : ℂ)
    (hz : ∀ μ∈spectrum ℂ A,μ.re < z.re) :
    IntegrableOn (fun t : ℝ => Complex.exp (-z*t) • (U*transfer A t*V)) (Ioi 0) ∧
    (∫ t : ℝ in Ioi 0,Complex.exp (-z*t) • (U*transfer A t*V))=
      U*(Matrix.scalar J z-A)⁻¹*V := by
  have h := NonnormalMemory.spectral_laplace_inverse A z hz
  have he : (fun t : ℝ => Complex.exp (-z*t) • (U*transfer A t*V))=
      (fun t => sandwich U V (damped A z t)) := by
    ext t i j
    simp [sandwich,damped]
  rw [he]
  refine ⟨(sandwich U V).integrable_comp h.1,?_⟩
  rw [(sandwich U V).integral_comp_comm h.1,h.2.2]
  change U*Ring.inverse (algebraMap ℂ (Matrix J J ℂ) z-A)*V=_
  rw [Matrix.nonsing_inv_eq_ringInverse]
  rfl
```

The source imposes no equality of retained and hidden dimensions and no nonemptiness assumption on the finite index types. The theorem includes nilpotent Jordan parts and zero-dimensional cases. It proves a finite-dimensional Laplace identity. General infinite hidden systems, globally optimal model reduction, and arbitrary memory closure remain separate questions.

8 Canonical jets and holomorphic naturality

Core 15.1 closes a common source of ambiguity: what exactly is meant by $f(A)$ when A is a nonnormal finite matrix and f is holomorphic on a possibly disconnected neighborhood of the spectrum? The final layer defines a canonical finite-jet operator, proves uniqueness and intertwining, and identifies its coefficients with genuine local Cauchy integrals under analytic hypotheses.

8.1 The finite-jet definition

For coefficient data $a(\mu, k)$, the jet at μ is a polynomial in $A - \mu I$:

Source: KomorebiPaperVI/CanonicalJets.lean.

```
noncomputable def jet (A : End ℂ V) (a : ℂ → ℕ → ℂ) (μ : ℂ) (n : ℕ) : End ℂ V :=
  ∑ k ∈ Finset.range n, a μ k • (A-μ • 1)^k
```

The canonical operator applies this polynomial on each generalized eigenspace and reassembles the direct sum:

Source: KomorebiPaperVI/CanonicalJets.lean.

```
noncomputable def calculus (A : End ℂ V) (a : ℂ → ℕ → ℂ) : End ℂ V := by
  classical
  exact (DirectSum.toModule ℂ ℂ V (fun μ =>
    (jet A a μ (finrank ℂ V)).comp (A.maxGenEigenspace μ).subtype)).comp
    (DirectSum.decomposeLinearEquiv A.maxGenEigenspace).toLinearMap
```

The decomposition is selected noncomputably, but the resulting operator is characterized uniquely by its action on each component. `HolomorphicFinite.canonical_unique` proves that characterization. "Noncomputable" means the definition uses classical constructions without an executable algorithm. It does not mean the proof is admitted.

8.2 Unequal dimensions require a common cutoff

If $CA = BC$, then C takes a generalized eigenspace of A into the corresponding one of B. Source and target may have different dimensions, so the naturality proof first extends both finite jets to a common maximum cutoff. Truncation stability makes that extension harmless:

Source: KomorebiPaperVI/CanonicalJets.lean.

```
theorem jet_stable (A : End ℂ V) (a : ℂ → ℕ → ℂ) (μ : ℂ) (x : V)
    (hx : x ∈ A.maxGenEigenspace μ) (n : ℕ) (hn : finrank ℂ V ≤ n) :
    jet A a μ n x=jet A a μ (finrank ℂ V) x := by
  classical
  unfold jet
  simp only [LinearMap.sum_apply,LinearMap.smul_apply]
  symm
  apply Finset.sum_subset (Finset.range_mono hn)
  intro k _ hk
  rw [power_kills A μ x hx k (by simpa using hk),smul_zero]
```

The source then applies polynomial intertwining componentwise and uses the direct-sum extensionality principle:

Source: KomorebiPaperVI/CanonicalJets.lean.

```
theorem canonical_naturality (A : End ℂ V) (B : End ℂ W) (C : V →ₗ[ℂ] W)
    (h : B.comp C=C.comp A) (a : ℂ → ℕ → ℂ) :
    C.comp (calculus A a)=(calculus B a).comp C := by
  apply DirectSum.decompose_lhom_ext A.maxGenEigenspace
  intro μ
  ext x
  have hx := Module.End.mapsTo_genEigenspace_of_comp C h μ ⊤ x.property
  change C (calculus A a x)=calculus B a (C x)
  rw [calculus_on_component A a μ x x.property,
    calculus_on_component B a μ (C x) hx,
    ← jet_stable A a μ x x.property (max (finrank ℂ V) (finrank ℂ W)) (le_max_left _ _),
    ← jet_stable B a μ (C x) hx (max (finrank ℂ V) (finrank ℂ W)) (le_max_right _ _)]
  exact jet_intertwines A B C h a μ _ x
```

This is an algebraic theorem about arbitrary coefficient data a. It is already enough to explain why the same local jets must be used on both sides of an intertwining equation.

8.3 Analyticity enters the interpretation

`CanonicalJets.holomorphic` substitutes the derivative coefficients $f^{(k)}(\mu)/k!$. The term exists for any function because Lean's derivative operations are totalized. The raw jet-naturality theorem therefore has no analyticity assumption. That fact must not be mistaken for a theorem that an arbitrary function has an analytic holomorphic functional calculus.

For an analytic f, Cauchy's formula connects the derivative coefficient to an actual positive-radius circle integral:

Source: KomorebiPaperVI/HolomorphicFinite.lean.

```
theorem cauchyCoefficient_eq (f : ℂ → ℂ) (μ : ℂ) (r : ℝ) (hr : 0 < r)
    (hf : DifferentiableOn ℂ f (Metric.closedBall μ r)) (k : ℕ) :
    cauchyCoefficient f μ r k=iteratedDeriv k f μ/(k.factorial:ℂ) := by
  have hh := hf.circleIntegral_one_div_sub_center_pow_smul hr k
  simp only [smul_eq_mul,one_div] at hh
  unfold cauchyCoefficient
  have he : (fun z => f z/(z-μ)^(k+1))=(fun z => ((z-μ)^(k+1))⁻¹*f z) := by
    ext z
    simp [div_eq_mul_inv,mul_comm]
  rw [he,hh]
  field_simp
```

The radius is positive and f is differentiable on the whole closed disk in this statement. A later theorem chooses suitable local disks inside any declared open neighborhood where f is analytic:

Source: KomorebiPaperVI/HolomorphicFinite.lean.

```
theorem canonical_contours_exist (A : End ℂ V) (f : ℂ → ℂ) (U : Set ℂ)
    (hU : IsOpen U) (hf : AnalyticOnNhd ℂ f U) (hA : spectrum ℂ A ⊆ U) :
    ∃ r : ℂ → ℝ,(∀ μ∈spectrum ℂ A,0 < r μ ∧ Metric.closedBall μ (r μ) ⊆ U) ∧
      calculus A (fun μ k => cauchyCoefficient f μ (r μ) k)=holomorphic A f := by
```

```
  classical
  have he (μ : ℂ) : ∃ r : ℝ,0 < r ∧
      (μ∈spectrum ℂ A → Metric.closedBall μ r ⊆ U) := by
    by_cases hm : μ∈spectrum ℂ A
    · obtain ⟨r,hr,hsub,_⟩ := admissible_circle f U hU hf μ (hA hm)
      exact ⟨r,hr,fun _ => hsub⟩
    · exact ⟨1,zero_lt_one,fun h => (hm h).elim⟩
  choose r hr hsub using he
  refine ⟨r,fun μ hμ => ⟨hr μ,hsub μ hμ⟩,?_⟩
  apply canonical_cauchy_representation A f r (fun μ _ => hr μ)
  intro μ hμ z hz
  exact (hf z (hsub μ hμ hz)).differentiableAt.differentiableWithinAt
```

The local radius can depend on the spectral value. This supports disconnected analytic neighborhoods and avoids assuming a single globally convergent Taylor series. Equality of the Cauchy coefficients proves independence of the admissible local circles. General contour deformation in arbitrary Banach algebras is not the formal scope of this finite-dimensional construction.

8.4 The final literal matrix equation

`matrixFunction` transfers the finite-jet operator from linear maps back to matrices. The resulting equation is rectangular and uses the original matrix intertwining hypothesis:

Source: `KomorebiPaperVI/HolomorphicFinite.lean`.

```
theorem matrix_holomorphic_naturality (A : Matrix I I ℂ) (B : Matrix J J ℂ)
    (C : Matrix J I ℂ) (h : C*A=B*C) (f : ℂ → ℂ) :
    C*matrixFunction A f=matrixFunction B f*C := by
  have hh : B.toLin'.comp C.toLin'=C.toLin'.comp A.toLin' := by
    rw [← Matrix.toLin'_mul,← Matrix.toLin'_mul,h]
  have hc := holomorphic_naturality A.toLin' B.toLin' C.toLin' hh f
  have hm := congrArg LinearMap.toMatrix' hc
  simpa only [LinearMap.toMatrix'_comp,LinearMap.toMatrix'_toLin',matrixFunction] using hm
```

Read this as naturality of the defined jet operator for arbitrary f, and as ordinary finite-dimensional holomorphic naturality when the preceding analytic hypotheses identify those jets with the desired holomorphic calculus. The bounded Banach reciprocal-family substitution moved from Paper IV has a broader setting and retains its own written proof in Part III. It has not acquired a new Lean endpoint by being moved into Paper VI.

9 Survival, hazard, and incomplete absorption

`Survival` begins with a finite real matrix whose off-diagonal entries are nonnegative and whose column sums are nonpositive. Its exponential evolves the surviving transient-state mass. With a nonnegative initial probability vector, survival is positive at every finite nonnegative time, the density is nonnegative, and survival is decreasing.

9.1 Finite-time positivity makes the hazard meaningful

The source proves strict positivity of $S(t)$, rather than treating $h(t) = f(t)/S(t)$ as legitimate by notation alone. Integrating the logarithmic derivative then yields the hazard identity:

Source: `KomorebiPaperVI/Survival.lean`.

```
theorem survival_hazard (A : Matrix I I ℝ) (hA : ∀ i j, i≠j → 0 ≤ A i j)
    (p : I → ℝ) (hp : ∀ i,0 ≤ p i) (hs : ∑ i,p i=1) (t : ℝ) (ht : 0 ≤ t) :
    survival A p t=Real.exp (-∫ r in 0..t,hazard A p r) := by
  have hd : ∀ r∈Icc 0 t, HasDerivAt (fun u => Real.log (survival A p u)) (-hazard A p r) r := by
    intro r hr
    simpa [hazard,neg_div] using (survival_derivative A p r).log
      (survival_positive A hA p hp hs r hr.1).ne'
  have hscont : Continuous (survival A p) :=
    (show Differentiable ℝ (survival A p) from fun r => (survival_derivative A p
    ↪  r).differentiableAt).continuous
  have hhcont : ContinuousOn (hazard A p) (Icc 0 t) :=
    (density_continuous A p).continuousOn.div hscont.continuousOn
      (fun r hr => (survival_positive A hA p hp hs r hr.1).ne')
  have hi := intervalIntegral.integral_eq_sub_of_hasDerivAt
    (by simpa [uIcc_of_le ht] using hd)
    (hhcont.neg.intervalIntegrable_of_Icc ht)
  simp only [Pi.neg_apply,intervalIntegral.integral_neg,survival_zero,hs,Real.log_one,sub_zero] at
  ↪  hi
  rw [hi,Real.exp_log (survival_positive A hA p hp hs t ht)]
```

The displayed theorem's hypotheses establish the relevant finite-time positivity and normalization. The nonpositive-column-sum condition is used in the density and monotonicity results. Keep these neighboring statements together when interpreting a model as a killed Markov evolution.

9.2 A density need not integrate to one

A finite subgenerator can leave positive surviving mass as $t \to \infty$. The formal result therefore proves existence of a limit and integrates the density to $1 - \ell$, not automatically to one:

Source: `KomorebiPaperVI/Survival.lean`.

```
theorem defective_density_mass (A : Matrix I I ℝ) (hA : ∀ i j, i≠j → 0 ≤ A i j)
    (hc : ∀ j,∑ i,A i j ≤ 0) (p : I → ℝ) (hp : ∀ i,0 ≤ p i) (hs : ∑ i,p i=1) :
    ∃ l,0 ≤ l ∧ Tendsto (survival A p) atTop (𝓝 l) ∧
      IntegrableOn (density A p) (Ioi 0) ∧ (∫ t in Ioi 0,density A p t)=1-l := by
  obtain ⟨l,hl,ht⟩ := survival_has_limit A hA hc p hp
  have hd := fun t (_ : t∈Ici (0:ℝ)) => survival_derivative A p t
```

```
have hn := fun t (ht : t∈Ioi (0:ℝ)) =>
  neg_nonpos.mpr (density_nonnegative A hA hc p hp t ht.le)
have hi := integrableOn_Ioi_deriv_of_nonpos' hd hn ht
have he := integral_Ioi_of_hasDerivAt_of_nonpos' hd hn ht
refine ⟨l,hl,ht,integrable_neg_iff.mp hi,?_⟩
simp only [integral_neg,survival_zero,hs] at he
linarith
```

The separate `eventual_absorption_density` endpoint assumes $S(t) \to 0$ and then obtains a unit-mass density. This distinction is essential in a phase-type application: eventual absorption is a hypothesis to establish, not an effect of calling a matrix transient in prose. Transform, moment, residual-life, and competing-risk formulas in the retained extensions have additional assumptions and are mapped separately.

10 The retained extensions and the formal boundary

The traditional edition preserves more mathematics than the 33-result kernel ledger. A nearby formal identity can explain an extension's architecture without proving that extension. This section maps the regions by mathematical content. "Related core" is a navigation aid, not a certificate for the entire row.

10.1 Part II: the strengthened branch

The labels in this table use Part II's E prefix. The 84 theorem-like result records in that branch retain their written proofs. The precise hypotheses and editorial corrections are recorded in the traditional edition. The table does not enlarge the formal denominator.

Written region	Related core	Formal boundary
E2: packet semigroups	1.2, 2.1, 2.2	Formal actual exponential and finite packet convolution. General C_0 semigroups, Dyson expansions, and packet-path resolvent series require their retained written assumptions.
E3: generator defects	3.1–3.3, 15.1	Rectangular Duhamel and matrix resolvent endpoints are formal. General Banach holomorphic defect formulas exceed finite canonical-jet naturality.
E4: limiting refinement	10.5	Formal varying-space norm summability does not construct every inverse-limit semigroup. Weighted completion, range leakage, generator closure, and limiting state-space choices are separate written results.
E5: Markov lumping	4.1, 5.1, 5.2	Exponential probability preservation and exact/approximate iterates are formal. Quotient obstructions, complete descent characterizations, and stationary descent have separate written proofs.
E6: conditional towers	6.1, 6.2	Finite weighted kernels and difference packets are formal. General conditional expectation and every expanded variance identity are not additional ledger endpoints.
E7: Gaussian/Hermite	None directly	The Ornstein–Uhlenbeck semigroup, infinite Hermite reconstruction, generator, Dirichlet form, spectral gap, and fractional Hermite transfer are outside this formal core.
E8: Bayesian transfer	7.1–7.3	Composition, evidence packets, and likelihood perturbation are formal. Prior-stability statements require their own exact interface and are not counted by analogy.
E9: finite sufficiency	8.1, 8.2, 8.4–8.6	Effective likelihoods, all-prior criteria, coarsest nontrivial partitions, and the loss matrix are formal within the ledger scope. Relative-entropy decomposition and data processing are separate written results.

Written region	Related core	Formal boundary
E10: Bayesian flows/HMM	9.1, 9.3	The normalized exponential flow and one-step simplex preservation are formal. Cumulants, concentration, and quantitative filtering bounds are retained written extensions.
E11: fractional operators	10.1–10.5	Actual normalized integrals, rectangular bounded Hilbert-space defects, and square-root subordination are formal. General stable-subordinator exposition is not a new formal endpoint.
E12: Green/PDE	11.1, 11.2	Exact finite Dirichlet inverse, grid-node identity, and scalar low-mode error are formal. Quadrature, forcing-error, and broader mode-transfer estimates retain their stated regularity assumptions.
E13: Feynman–Kac	12.1	The exact discrete finite path law is formal. Continuous-time probabilistic representation and product slicing have separate written proofs.
E14: Schur elimination	13.1, 13.2	Actual block inverse and finite-dimensional Laplace memory identity are formal. The full time-domain closure criterion is not separately included in the ledger.
E15: memory compression	13.2	The finite nonnormal exponential-polynomial mechanism is formal. Certified block selection, realizations, compression optimization, and Volterra trajectory bounds are written extensions.
E16: survival/risks	14.1	Finite survival, hazard, and defective density mass are formal. Transform moments, residual-life descriptions, and competing-risk decompositions have separate written scope.

10.2 Part III: earlier results retained in their own right

The earlier Annals branch contains results whose scope or constants are not fully subsumed by the strengthened text. Their retained statements and proofs appear in Part III, with R-prefixed numbering. The accompanying navigation crosswalk lists every retained theorem-like record and its final traditional location. Retention preserves the mathematics. It does not create new Lean declarations.

These results include multilevel telescoping and coordinate-generator arguments, reversible descent and Dirichlet forms, prior-relative Bayesian sufficiency, quantitative normalization and information-loss estimates, covariance and hidden-Markov formulas, potential perturbation and coarse Feynman–Kac transfer, and finite rational memory or phase-type descriptions. Each has to be compared against a formal theorem by its exact hypotheses and conclusion. Shared vocabulary is insufficient.

10.3 The operator substitution migrated from Paper IV

The migrated reciprocal-family proposition concerns bounded Banach operators and holomorphic functional calculus, together with an explicit classical functional-calculus input. It belongs with Paper VI's operator language and is retained with its written proof. The core 15.1 result instead constructs and proves naturality of canonical finite-dimensional jets. Their relationship is conceptual. The migrated proposition is outside the 33-result formal contract.

10.4 The three original continuum conjectures

The latest core retains three open conjectures: transfer-semigroup realization on completed state spaces; uniform refinement control for meaningful discretization families; and latent-packet closure for stable infinite packet systems. Finite or bounded special cases proved elsewhere in the edition do not settle these universal formulations. In particular, summable local defects alone do not guarantee a semigroup limit without the appropriate completion, compatibility, and closure conditions.

10.5 Traditional-paper navigation

Printed pages are the labels in the traditional volume. PDF pages count every physical page, including its cover and Roman opening pages. These locations refer to the final 152-page monograph. Every physical location was resolved from the actual PDF page label and checked against the final source map.

All 33 formal-core endpoints.

Core	Result	Printed	PDF
1.2	Finite semigroup law	3	10
2.1	Packet semigroup convolution	4	11
2.2	Reducing packets	4	11
3.1	Exact Duhamel refinement identity	5	12
3.2	Norm bound	6	13
3.3	Resolvent refinement identity	6	13
4.1	Finite Markov semigroup	7	14
5.1	Exact lumpability	8	15
5.2	Finite-horizon defect formula	8	15
6.1	Finite conditional-expectation projector	9	16
6.2	Tower law and martingale packets	10	17
7.1	Independent-likelihood composition	11	18
7.2	Bayesian packet reconstruction	11	18
7.3	Likelihood perturbation bound	12	19
8.1	Exact effective-likelihood factorization	13	20
8.2	Bayesian refinement criterion	14	21
8.4	Bayesian information-loss packet	14	21
8.5	Nested Bayesian packet tower	15	22
8.6	Canonical coarsest likelihood-resolving partition	15	22
9.1	Bayesian nonlinear semigroup	18	25
9.3	Prediction-update reconstruction	19	26
10.1	Fractional packet calculus	20	27

Core	Result	Printed	PDF
10.2	Poisson subordination	21	28
10.3	Fractional intertwining theorem	22	29
10.4	Fractional semigroup refinement	23	30
10.5	Fractional refinement towers	23	30
11.1	Exact discrete Green reconstruction at grid nodes	25	32
11.2	Second-order low-mode convergence	25	32
12.1	Discrete Feynman–Kac path formula	27	34
13.1	Schur–Feshbach projected resolvent	28	35
13.2	Memory/self-energy correspondence	29	36
14.1	Semigroup survival identities	30	37
15.1	Naturality under exact intertwining	32	39

The 24 earlier results retained in Part III. These have written proofs. This list creates no additional Lean completion claim.

Final	Result	Printed	PDF
R1.1	Packet convolution	90	97
R1.2	Packet resolvent convolution	90	97
R4.1	Fractional low mode dispersion	96	103
R5.1	Stability of memory compression	97	104
R1.3	Minimum transfer order	91	98
R1.4	Several level telescoping identity	91	98
R1.5	Summable generator defects	91	98
R1.7	Exact projective limit semigroup	92	99
R1.8	Coordinate generators	92	99
R2.1	Reversibility descends	92	99
R2.2	Coarse Dirichlet form	93	100
R3.1	Sufficiency relative to a prior disintegration	93	100
R3.2	Normalization bound	93	100
R3.3	Exact information loss formula	94	101
R3.4	Oscillation bound	94	101
R3.5	Covariance identity	95	102
R3.6	Exact coarse hidden Markov filter	95	102
R3.7	One step filter defect	95	102
R4.2	Potential perturbation bound	96	103
R4.3	Coarse Feynman Kac transfer	96	103
R5.2	Rational memory realization	98	105
R5.3	Exponential sum memory	98	105
R5.4	Phase type rationality	98	105
R5.5	One phase model	98	105

The migrated bounded Banach reciprocal proposition is R7.2, printed page 101 (PDF page 108). Its scope remains outside the 33-result contract.

11 Reproduction and evidence review

11.1 The producing environment is pinned

The production source declares Lean `leanprover/lean4:v4.34.0-rc2`, project release `0.5.0`, and Mathlib revision

`aeb72f1e9981b6d35b426b16dca7759be3e7ba06`.

The nine exact package revisions are listed in Section 14. The Lake configuration disables both automatic and relaxed automatic implicits and treats warnings as errors. The original verifier checks the expected file set, source hashes, imports, theorem inventory, and kernel-audit target coverage before attempting the compiler gates.

The producing September environment used a small runtime self-location adapter because a particular process-path lookup was restricted. The preserved `verification/runtime/RUNTIME.md` explains that adapter and its positive and negative controls. The compiler, kernel, and dependency sources were recorded as unmodified. The adapter is not required by a normal installation, and compiler binaries and dependency caches are not included in the source archive.

11.2 The inherited kernel evidence

The historical record has timestamp `2026-09-06T06:00:38.447662+00:00`. It records a successful production build, 341 explicit theorem targets, and 782 inspected namespace declarations. The permitted axiom set is

$$\{\texttt{propext}, \texttt{Classical.choice}, \texttt{Quot.sound}\}.$$

These are standard foundational axioms used by the project. The source verifier rejects placeholder and unregistered unsafe mechanisms, and the namespace audit checks the resulting declarations. A source-token scan alone cannot establish these semantic properties. It is one gate preceding actual elaboration and kernel checking.

The complete historical files remain under the preserved original snapshot: `verification/verification_status.json`, `verification/lean_build.log`, `verification/lean_axiom_audit.log`, and `verification/source_contract.json`. The guide's provenance JSON records their hashes, so the documentation is tied to a specific evidence revision.

11.3 Current checks for the monograph edition

The 9 October 2026 checks preserve all 739 recovered source files byte for byte and reconcile the 534 original snapshot pins. The current source contract matches all 33 scoped result records, 341 theorem targets, and 47 proof modules. These are source integrity and inventory checks. They do not execute the Lean kernel.

The original available-gates dispatcher now ran the actual `pytest` suite: 54 cases passed with no failures, errors, or skips. Its six verifier methods include one clean-input check and five deliberate corruptions that the verifier rejected. The original symbolic visual suite also ran with SymPy: 204 assertions passed and 11 false controls were rejected. Separately recorded rational witnesses passed 182 exact checks and four false controls. The rational witnesses revisit some of the same finite models, so these totals are not added into an independent-result count.

The numerical companion reproduced all 79 stored numerical fields and two nonnumeric leaves, and regenerated 19 figure pairs. The three broader suites passed their own mixed comparison tallies:

475,471 checks in 16 Annals groups, 149,349 in 14 strengthened-baseline groups, and 154,608 in 15 expanded groups. These scopes overlap and include numerical or structural checks, so their tallies are reported separately. They do not establish universal theorems. The three new explanatory figures also passed 51 exact rational assertions within their recorded finite models.

The visual guide has a separate extension-model gate: 120 exact rational assertions passed and eight false controls were rejected. Its 40 editable plate sources also regenerated byte for byte. Drawing regeneration checks geometry and reproduction, rather than adding mathematical proof coverage. These records remain separate from the original symbolic visual suite.

The current computation review is `verification/monograph_checks/COMPUTATIONAL_REVIEW.json`. Each execution has a bounded manifest recording the command, runtime, tested input range, resource cap, source hashes, raw outputs, and interpretation. The package preserves earlier run records as historical evidence. Current reruns have their own dates and provenance.

Lean, Lake, and Elan are unavailable in this environment. No current compiler build or namespace axiom audit was run. The inherited kernel evidence remains the September 6 record described above. A passing finite example or source check cannot supply a missing universal proof or settle the three continuum conjectures.

The guide's own reconciliation confirms all 53 source-contract hashes, all target declaration lines, the 33-row ledger, module imports, nine dependency pins, and all 341 saved axiom-output records. Every raw excerpt remains an exact copy of its recorded source span. PDF review checks page layout, target-name extraction, font rendering, and source navigation. These documentation checks introduce no new theorem and make no claim of current kernel execution.

11.4 Commands on an already provisioned system

Use a disposable copy of the original snapshot when reproducing the original verifier: it writes new reports beside its input. Do not overwrite the preserved historical evidence. On a system that already has the declared toolchain, pinned dependencies, and documented Python packages, run from that copy:

```
python3 verification/verify.py
```

This is the full original gate, including Python tests, verifier controls, the Lean build, dependency revision and cleanliness checks, and the namespace axiom audit. The `--static-only` option still dispatches Python tests and controls. It is not a guarantee of a dependency-free run. The current edition's portable verification scripts separately document the available subset used here.

To inspect the production theorem namespace in an already provisioned Lean checkout, the original underlying commands are:

```
lake build KomorebiPaperVI
lake env lean -DwarningAsError=true KernelAudit.lean
```

Running those commands successfully against a different toolchain or different dependency revision would create a new evidence revision. Preserve its logs and provenance rather than relabeling the inherited record.

11.5 Building this reading guide

From the editable Lean-guide directory, run `python3 build.py`. It compiles the supplied TeX and excerpt files with XeLaTeX and does not invoke Lean or regenerate mathematical evidence. The build uses installed URW P052 and Nimbus Sans, Latin Modern mono and math, and DejaVu Sans Mono fonts. Font files are resolved locally with `kpsewhich` and standard system locations, then copied only into the private build stage. No font binaries are bundled or downloaded. The monograph style is supplied beside the TeX source. The script copies the full authored guide tree to a temporary directory and compiles it in four passes. No installation is performed. `verify_pdf.py` performs the documented PDF checks and renders all pages into contact sheets.

`assemble.py` is a separate documentation-generation step. It reads the preserved source snapshot, exact source contract, selected excerpt list, and a current verification report. Its default inputs are relative to the supplied package. It also accepts `--source` and `--verification` paths when those inputs are stored elsewhere. Its generated tables are already supplied for ordinary PDF reproduction. Run `python3 reconcile_sources.py` to check those tables and excerpts against the preserved sources without changing their mathematical content.

11.6 When a reuse needs a new proof

Before adapting an endpoint, check the scalar field, finite versus bounded setting, positivity notion, time domain, evidence condition, and domain/codomain of every operator. Changing a hypothesis is not a formatting edit. A new theorem about unbounded generators, general statistical sufficiency, infinite hidden-state elimination, or unrestricted Banach holomorphic calculus requires its own proof and correspondence review. The preserved project supplies a precise starting point for that work.

12 Complete ledger of the 33 scoped results

This ledger reproduces the recorded checked scope from `THEOREM_LEDGER.json`. Numbering refers to the latest core in Part I of the traditional paper. These are 33 manuscript result records, not 33 source declarations and not a certification of every retained extension. The declaration catalog in Section 13 resolves every named target to its module and original line.

12.1 1.2 / Finite semigroup law

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Real Banach-algebra exponential law and both derivatives; finite operators are a specialization.

Declared targets.

- `KomorebiPaperVI.Semigroups.transfer_add`. Source line 20.
- `KomorebiPaperVI.Semigroups.transfer_zero`. Source line 18.
- `KomorebiPaperVI.Semigroups.transfer_derivative_left`. Source line 29.
- `KomorebiPaperVI.Semigroups.transfer_derivative_right`. Source line 26.

12.2 2.1 / Packet semigroup convolution

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Actual exponential packet convolution for a finite idempotent resolution of identity.

Declared targets.

- `KomorebiPaperVI.Algebra.block_reconstruction`. Source line 12.
- `KomorebiPaperVI.Semigroups.exponential_packet_convolution`. Source line 44.

12.3 2.2 / Reducing packets

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

A finite orthogonal idempotent resolution with zero off-diagonal generator blocks commutes with the generator; actual exponential cross blocks vanish and compressed dynamics agree with the exponential of the compressed generator.

Declared targets.

- `KomorebiPaperVI.ReducingPackets.generator_commutes`. Source line 12.
- `KomorebiPaperVI.ReducingPackets.reducing_exponential`. Source line 32.
- `KomorebiPaperVI.ReducingPackets.compressed_evolution`. Source line 39.

12.4 3.1 / Exact Duhamel refinement identity

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Bochner interval-integral identity for rectangular bounded real operators on Banach spaces, and exact intertwining.

Declared targets.

- `KomorebiPaperVI.Semigroups.duhamel_integrand_derivative`. Source line 57.

- `KomorebiPaperVI.Semigroups.duhamel_rectangular`. Source line 70.
- `KomorebiPaperVI.Semigroups.exact_rectangular_intertwining`. Source line 81.

12.5 3.2 / Norm bound

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

The displayed exponential-growth integral estimate for rectangular bounded real operators, with the contraction specialization.

Declared targets.

- `KomorebiPaperVI.OperatorBounds.growth_duhamel_bound`. Source line 10.
- `KomorebiPaperVI.Semigroups.contraction_duhamel_bound`. Source line 89.

12.6 3.3 / Resolvent refinement identity

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Literal rectangular matrices with left/right inverse equations and central scalar; applies to the stated resolvents.

Declared targets.

- `KomorebiPaperVI.Algebra.rectangular_resolvent_identity`. Source line 85.

12.7 4.1 / Finite Markov semigroup

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

The actual real matrix exponential of any finite generator has nonnegative entries and columns summing to one for nonnegative time; consequently it preserves probability vectors.

Declared targets.

- `KomorebiPaperVI.MarkovSemigroup.markov_nonnegative`. Source line 48.
- `KomorebiPaperVI.MarkovSemigroup.markov_column_sum`. Source line 71.
- `KomorebiPaperVI.MarkovSemigroup.markov_probability`. Source line 84.

12.8 5.1 / Exact lumpability

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Literal rectangular matrix intertwining for every natural-number iterate.

Declared targets.

- `KomorebiPaperVI.Algebra.intertwining_powers`. Source line 53.

12.9 5.2 / Finite-horizon defect formula

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

The literal rectangular finite-sum defect identity and the norm bound for arbitrary rectangular contractions, including horizon zero and zero spaces.

Declared targets.

- `KomorebiPaperVI.Algebra.finite_horizon_defect`. Source line 64.
- `KomorebiPaperVI.OperatorBounds.finite_horizon_contraction`. Source line 50.

12.10 6.1 / Finite conditional-expectation projector

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Literal finite complex weighted conditional expectation: idempotence, self-adjointness for the weighted inner product, and range exactly the block-constant functions. Strictly positive finite weights are explicit.

Declared targets.

- `KomorebiPaperVI.ConditionalExpectation.expect_idempotent`. Source line 49.
- `KomorebiPaperVI.ConditionalExpectation.expect_self_adjoint`. Source line 105.
- `KomorebiPaperVI.ConditionalExpectation.expect_range`. Source line 74.
- `KomorebiPaperVI.ConditionalExpectation.expect_matrix`. Source line 84.

12.11 6.2 / Tower law and martingale packets

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Both concrete conditional-expectation tower identities; arbitrary pairs of nested projector kernels multiply to the coarser kernel; their differences are pairwise orthogonal idempotents and sum to the terminal projector.

Declared targets.

- `KomorebiPaperVI.ConditionalTowers.conditional_kernel_product`. Source line 37.
- `KomorebiPaperVI.ConditionalTowers.nested_kernel_min`. Source line 53.
- `KomorebiPaperVI.ConditionalTowers.martingale_packet_product`. Source line 65.
- `KomorebiPaperVI.ConditionalTowers.martingale_packet_reconstruction`. Source line 72.

12.12 7.1 / Independent-likelihood composition

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Composition with explicit nonzero first and combined evidences; positive evidences imply these hypotheses.

Declared targets.

- `KomorebiPaperVI.FiniteBayes.sequential_evidence`. Source line 37.
- `KomorebiPaperVI.FiniteBayes.posterior_composition`. Source line 45.

12.13 7.2 / Bayesian packet reconstruction

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Literal finite fibers and evidence-weighted reconstruction, including zero-evidence packets; specialize nonnegative weights to likelihood times prior.

Declared targets.

- `KomorebiPaperVI.Coarsening.packet_reconstruction_point`. Source line 90.
- `KomorebiPaperVI.Coarsening.packet_reconstruction`. Source line 99.
- `KomorebiPaperVI.Coarsening.zero_packet_component`. Source line 48.

12.14 7.3 / Likelihood perturbation bound

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

The printed finite weighted L1 perturbation bound under nonnegativity and positive evidence.

Declared targets.

- `KomorebiPaperVI.FiniteBayes.normalized_l1_bound_left`. Source line 73.
- `KomorebiPaperVI.FiniteBayes.likelihood_perturbation_bound`. Source line 102.

12.15 8.1 / Exact effective-likelihood factorization

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Literal finite coarsening and prior-dependent effective likelihood, including zero prior-mass fibers.

Declared targets.

- `KomorebiPaperVI.Coarsening.effective_weight`. Source line 66.
- `KomorebiPaperVI.Coarsening.effective_evidence`. Source line 78.
- `KomorebiPaperVI.Coarsening.effective_factorization`. Source line 83.

12.16 8.2 / Bayesian refinement criterion

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

The all-prior equivalence for strictly positive fine and coarse likelihoods and a surjective partition with at least two packets; necessity is derived from actual two-state priors. Singleton coarsening is automatic.

Declared targets.

- `KomorebiPaperVI.BayesianNecessity.refinement_iff`. Source line 112.
- `KomorebiPaperVI.BayesianNecessity.cross_packet_ratio`. Source line 67.
- `KomorebiPaperVI.BayesianTowers.singleton_commutes`. Source line 30.

12.17 8.4 / Bayesian information-loss packet

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

The literal information-loss matrix factorization, null scale direction, minimum-evidence L1 estimate, and the final fine-to-coarse weighted bound, including zero prior-mass fibers.

Declared targets.

- `KomorebiPaperVI.InformationLoss.information_loss_matrix`. Source line 17.
- `KomorebiPaperVI.InformationLoss.information_loss_null`. Source line 28.
- `KomorebiPaperVI.InformationLoss.information_loss_norm`. Source line 68.
- `KomorebiPaperVI.InformationLoss.total_loss_bound`. Source line 60.
- `KomorebiPaperVI.FiniteBayes.likelihood_perturbation_bound`. Source line 102.

12.18 8.5 / Nested Bayesian packet tower

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

The necessary-and-sufficient adjacent likelihood criterion with singleton exception, and exact Bayesian commutation over every finite span of a varying-state-space tower.

Declared targets.

- `KomorebiPaperVI.BayesianTowers.adjacent_criterion`. Source line 43.
- `KomorebiPaperVI.BayesianTowers.tower_criterion`. Source line 80.
- `KomorebiPaperVI.BayesianTowers.tower_commutes`. Source line 73.
- `KomorebiPaperVI.BayesianTowers.coarsen_comp`. Source line 12.

12.19 8.6 / Canonical coarsest likelihood-resolving partition

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Actual quotient by equality of likelihood profiles, canonical coarse likelihoods and exact closure, unique factorization from every likelihood-resolving partition, and the corrected coarsest nontrivial all-prior-closed property under strict positivity and distinct profiles.

Declared targets.

- `KomorebiPaperVI.BayesianPartitions.packet_eq_iff`. Source line 28.
- `KomorebiPaperVI.BayesianPartitions.canonical_coarsening`. Source line 38.
- `KomorebiPaperVI.BayesianPartitions.canonical_unique_factor`. Source line 46.
- `KomorebiPaperVI.BayesianPartitions.all_prior_family_iff_resolves`. Source line 67.
- `KomorebiPaperVI.BayesianPartitions.canonical_coarsest_nontrivial`. Source line 85.
- `KomorebiPaperVI.BayesianPartitions.canonical_nontrivial`. Source line 91.

12.20 9.1 / Bayesian nonlinear semigroup

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Actual normalized real exponential flow on the finite probability simplex, its semigroup law, and centered multiplicative derivative.

Declared targets.

- `KomorebiPaperVI.BayesianFlow.positive_evidence`. Source line 13.
- `KomorebiPaperVI.BayesianFlow.flow_probability`. Source line 25.
- `KomorebiPaperVI.BayesianFlow.flow_add`. Source line 36.
- `KomorebiPaperVI.BayesianFlow.flow_zero`. Source line 45.
- `KomorebiPaperVI.BayesianFlow.flow_derivative`. Source line 64.

12.21 9.3 / Prediction-update reconstruction

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Literal column-stochastic matrix prediction followed by a nonnegative positive-evidence Bayesian update.

Declared targets.

- `KomorebiPaperVI.FiniteBayes.prediction_probability`. Source line 124.
- `KomorebiPaperVI.FiniteBayes.prediction_update_probability`. Source line 135.

12.22 10.1 / Fractional packet calculus

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Fractional powers and transfer for the given finite spectral resolution, including zero eigenvalues; multiplication/addition laws, packet reduction, and identification with the actual algebra exponential. Finite spectral resolution is the stated starting data.

Declared targets.

- `KomorebiPaperVI.SpectralPackets.fractional_add`. Source line 58.
- `KomorebiPaperVI.SpectralPackets.spectral_transfer_exponential`. Source line 68.
- `KomorebiPaperVI.SpectralPackets.spectral_transfer_add`. Source line 77.
- `KomorebiPaperVI.SpectralPackets.fractional_reduces`. Source line 89.
- `KomorebiPaperVI.SpectralPackets.spectral_transfer_reduces`. Source line 96.

12.23 10.2 / Poisson subordination

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

The actual normalized improper Poisson integral for every nonnegative eigenvalue, including zero; Bochner integrability and equality with the actual exponential of the finite spectral square root.

Declared targets.

- `KomorebiPaperVI.PoissonKernel.kernel_mass`. Source line 35.
- `KomorebiPaperVI.PoissonKernel.heat_integrand_integrable`. Source line 54.

- `KomorebiPaperVI.ReciprocalGaussian.reciprocal_square_gaussian`. Source line 127.
- `KomorebiPaperVI.PoissonSubordination.scalar_subordination`. Source line 42.
- `KomorebiPaperVI.PoissonSubordination.finite_spectral_subordination`. Source line 52.
- `KomorebiPaperVI.PoissonSubordination.operator_subordination`. Source line 74.

12.24 10.3 / Fractional intertwining theorem

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Actual normalized Lebesgue/Bochner resolvent-defect integral for bounded nonnegative self-adjoint operators between distinct complex Hilbert spaces, the exact Holder constant, the alpha*m^(alpha-1) common-gap estimate, and exact fractional intertwining. Real finite matrices use their standard complexification; this ordinary specialization is manually reviewed.

Declared targets.

- `KomorebiPaperVI.BetaNormalization.normalizer_exact`. Source line 80.
- `KomorebiPaperVI.BetaNormalization.normalized_gap_integral`. Source line 164.
- `KomorebiPaperVI.NormalizedFractional.cfc_lebesgue_integral`. Source line 75.
- `KomorebiPaperVI.PositiveResolvent.resolvent_inverse`. Source line 18.
- `KomorebiPaperVI.PositiveResolvent.resolvent_gap_norm`. Source line 45.
- `KomorebiPaperVI.FractionalNormBounds.holder_estimate`. Source line 97.
- `KomorebiPaperVI.FractionalNormBounds.common_gap_estimate`. Source line 67.
- `KomorebiPaperVI.RectangularFractional.literal_resolvent_integral`. Source line 95.
- `KomorebiPaperVI.RectangularFractional.holder_estimate`. Source line 173.
- `KomorebiPaperVI.RectangularFractional.common_gap_estimate`. Source line 145.
- `KomorebiPaperVI.RectangularFractional.exact_fractional_intertwining`. Source line 110.

12.25 10.4 / Fractional semigroup refinement

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Actual fractional exponentials between distinct complex Hilbert spaces, rectangular Duhamel identity, contraction estimate, and the final common-gap bound in terms of the original generator defect.

Declared targets.

- `KomorebiPaperVI.RectangularSemigroup.evolution_exponential`. Source line 19.
- `KomorebiPaperVI.RectangularSemigroup.contraction`. Source line 30.
- `KomorebiPaperVI.RectangularSemigroup.gap_decay`. Source line 35.
- `KomorebiPaperVI.RectangularSemigroup.fractional_duhamel`. Source line 41.
- `KomorebiPaperVI.RectangularSemigroup.fractional_contraction_bound`. Source line 53.
- `KomorebiPaperVI.RectangularSemigroup.generator_gap_bound`. Source line 94.

12.26 10.5 / Fractional refinement towers

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Varying complex Hilbert-space towers: both printed generator criteria imply summability of fractional-defect norms and time-defect norms, with uniform convergence of the real norm series on bounded nonnegative time intervals. Operator-series convergence is additionally proved in the inherited fixed-space formulation; no sum across unrelated operator spaces is asserted.

Declared targets.

- `KomorebiPaperVI.RectangularTowers.generator_holder_summable`. Source line 40.
- `KomorebiPaperVI.RectangularTowers.generator_gap_summable`. Source line 49.
- `KomorebiPaperVI.RectangularTowers.bounded_time_majorant`. Source line 58.
- `KomorebiPaperVI.RectangularTowers.time_norms_summable`. Source line 64.
- `KomorebiPaperVI.RectangularTowers.bounded_time_uniform_convergence`. Source line 72.
- `KomorebiPaperVI.RectangularTowers.holder_tower`. Source line 82.
- `KomorebiPaperVI.RectangularTowers.gap_tower`. Source line 93.
- `KomorebiPaperVI.FractionalTowers.bounded_time_uniform_convergence`. Source line 35.

12.27 11.1 / Exact discrete Green reconstruction at grid nodes

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

The literal finite Dirichlet tridiagonal matrix, its actual inverse, the scaled discrete Laplacian inverse, and the exact grid-node continuum Green identity for h=1/(n+1).

Declared targets.

- `KomorebiPaperVI.DiscreteGreen.green_second_difference`. Source line 22.
- `KomorebiPaperVI.DiscreteGreen.tridiagonal_inverse_kernel`. Source line 85.
- `KomorebiPaperVI.DiscreteGreen.inverse_eq_kernel`. Source line 109.
- `KomorebiPaperVI.DiscreteGreen.laplacian_inverse`. Source line 115.
- `KomorebiPaperVI.DiscreteGreen.grid_reconstruction`. Source line 122.

12.28 11.2 / Second-order low-mode convergence

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

The exact bound 0 <= w^2 - 4/h^2 sin(w*h/2)^2 <= w^4*h^2/12 for every nonzero spacing, with w=k*pi. This uniform bound implies the printed O(h^2) assertion as positive h tends to zero.

Declared targets.

- `KomorebiPaperVI.LowModeConvergence.sine_square_error`. Source line 8.
- `KomorebiPaperVI.LowModeConvergence.low_mode_error`. Source line 25.

12.29 12.1 / Discrete Feynman–Kac path formula

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

The actual potential-weighted matrix power equals the expectation over all finite paths; the path law is nonnegative and has total mass one for column-stochastic K. Product transition probabilities, the sum over departure-state potentials, and the terminal-state index are proved explicitly.

Declared targets.

- `KomorebiPaperVI.FeynmanKac.path_probability_nonnegative`. Source line 41.
- `KomorebiPaperVI.FeynmanKac.path_probability_mass`. Source line 48.
- `KomorebiPaperVI.FeynmanKac.path_probability_product`. Source line 85.
- `KomorebiPaperVI.FeynmanKac.path_potential_sum`. Source line 95.
- `KomorebiPaperVI.FeynmanKac.terminal_last`. Source line 104.
- `KomorebiPaperVI.FeynmanKac.feynman_kac_path_formula`. Source line 74.

12.30 13.1 / Schur–Feshbach projected resolvent

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Literal finite complex block matrices: under invertibility of the full resolvent matrix and the hidden resolvent block, the retained block of the actual inverse is the inverse Schur complement; invertibility equivalence is included.

Declared targets.

- `KomorebiPaperVI.SchurFeshbach.resolvent_blocks`. Source line 14.
- `KomorebiPaperVI.SchurFeshbach.projected_resolvent`. Source line 21.
- `KomorebiPaperVI.SchurFeshbach.schur_invertible`. Source line 40.

12.31 13.2 / Memory/self-energy correspondence

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

For arbitrary finite complex matrices, including nonnormal matrices and nilpotent Jordan parts, the strict spectral half-plane implies actual Bochner integrability and decay of the complex-damped exponential. Both inverse identities are derived by the improper-integral fundamental theorem. The literal rectangular retained/hidden memory integral equals U*(zI-A)^(-1)*V, with the actual matrix inverse and no supplied growth bound.

Declared targets.

- `KomorebiPaperVI.ScalarDecay.exp_pow_integrable`. Source line 16.
- `KomorebiPaperVI.ScalarDecay.exp_pow_tendsto_zero`. Source line 25.
- `KomorebiPaperVI.NonnormalMemory.exp_mul_component`. Source line 38.
- `KomorebiPaperVI.NonnormalMemory.spectral_decay`. Source line 99.
- `KomorebiPaperVI.NonnormalMemory.spectral_laplace_inverse`. Source line 114.
- `KomorebiPaperVI.MatrixMemory.rectangular_memory_self_energy`. Source line 19.

12.32 14.1 / Semigroup survival identities

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Actual finite Metzler-subgenerator exponential: survival derivative and strict positivity at finite nonnegative time, monotonicity, integrated hazard, existence of the survival limit, total defective density mass 1 minus that limit, and the eventual-absorption specialization.

Declared targets.

- `KomorebiPaperVI.Survival.survival_derivative`. Source line 54.
- `KomorebiPaperVI.Survival.survival_positive`. Source line 30.
- `KomorebiPaperVI.Survival.survival_antitone`. Source line 69.
- `KomorebiPaperVI.Survival.survival_hazard`. Source line 82.
- `KomorebiPaperVI.Survival.survival_has_limit`. Source line 100.
- `KomorebiPaperVI.Survival.defective_density_mass`. Source line 115.
- `KomorebiPaperVI.Survival.eventual_absorption_density`. Source line 129.

12.33 15.1 / Naturality under exact intertwining

Recorded status. Complete scoped statement (inherited v0.5.0 ledger).

Canonical basis-free finite Taylor-jet functional calculus on generalized eigenspaces, with uniqueness, truncation stability, dependence only on spectral jets, admissible local Cauchy circles in every declared open holomorphic neighborhood, and independence of those circles. Exact naturality holds for literal rectangular complex matrices without assuming a contour representation or one globally convergent Taylor series. The paper now states this standard finite-dimensional definition explicitly; real matrices are treated by the previously disclosed ordinary complexification specialization.

Declared targets.

- `KomorebiPaperVI.CanonicalJets.calculus_on_component`. Source line 28.
- `KomorebiPaperVI.CanonicalJets.jet_stable`. Source line 46.
- `KomorebiPaperVI.CanonicalJets.canonical_naturality`. Source line 81.
- `KomorebiPaperVI.HolomorphicFinite.canonical_unique`. Source line 17.
- `KomorebiPaperVI.HolomorphicFinite.calculus_spectrum_congr`. Source line 38.
- `KomorebiPaperVI.HolomorphicFinite.cauchyCoefficient_eq`. Source line 58.
- `KomorebiPaperVI.HolomorphicFinite.canonical_contours_exist`. Source line 96.
- `KomorebiPaperVI.HolomorphicFinite.canonical_contours_independent`. Source line 113.
- `KomorebiPaperVI.HolomorphicFinite.matrix_holomorphic_naturality`. Source line 129.

13 All 341 supporting theorem declarations

The inventory is generated from the preserved v0.5.0 source contract. “Support” means the declaration is included in the audited target inventory but is not named directly as an endpoint in the 33-row manuscript ledger. It can still support a ledger endpoint. This is not a proof-dependency computation. Source lines refer to the immutable production files, not the display excerpts in this guide.

13.1 Algebra: 9 theorem declarations

Source: `KomorebiPaperVI/Algebra.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.Algebra.block_reconstruction`	12; 2.1
`KomorebiPaperVI.Algebra.block_multiplication`	16; support
`KomorebiPaperVI.Algebra.packet_semigroup_convolution`	28; support
`KomorebiPaperVI.Algebra.difference_idempotent`	35; support
`KomorebiPaperVI.Algebra.difference_annihilates_coarse`	39; support
`KomorebiPaperVI.Algebra.tower_telescope`	44; support
`KomorebiPaperVI.Algebra.intertwining_powers`	53; 5.1
`KomorebiPaperVI.Algebra.finite_horizon_defect`	64; 5.2
`KomorebiPaperVI.Algebra.rectangular_resolvent_identity`	85; 3.3

13.2 BanachDefects: 3 theorem declarations

Source: `KomorebiPaperVI/BanachDefects.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.BanachDefects.duhamel_derivative`	8; support
`KomorebiPaperVI.BanachDefects.duhamel`	21; support
`KomorebiPaperVI.BanachDefects.contraction_bound`	29; support

13.3 BayesianFlow: 8 theorem declarations

Source: `KomorebiPaperVI/BayesianFlow.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.BayesianFlow.positive_evidence`	13; 9.1
`KomorebiPaperVI.BayesianFlow.flow_probability`	25; 9.1
`KomorebiPaperVI.BayesianFlow.likelihood_add`	30; support
`KomorebiPaperVI.BayesianFlow.flow_add`	36; 9.1
`KomorebiPaperVI.BayesianFlow.flow_zero`	45; 9.1
`KomorebiPaperVI.BayesianFlow.likelihood_derivative`	52; support
`KomorebiPaperVI.BayesianFlow.evidence_derivative`	56; support

Declaration	Line; ledger
`KomorebiPaperVI.BayesianFlow.flow_derivative`	64; 9.1

13.4 BayesianNecessity: 9 theorem declarations

Source: `KomorebiPaperVI/BayesianNecessity.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.BayesianNecessity.pair_weighted_sum`	13; support
`KomorebiPaperVI.BayesianNecessity.pair_probability`	19; support
`KomorebiPaperVI.BayesianNecessity.pair_fiber_weight`	25; support
`KomorebiPaperVI.BayesianNecessity.pair_coarsen`	35; support
`KomorebiPaperVI.BayesianNecessity.pair_coarse_posterior`	54; support
`KomorebiPaperVI.BayesianNecessity.cross_packet_ratio`	67; 8.2
`KomorebiPaperVI.BayesianNecessity.all_packet_ratio`	85; support
`KomorebiPaperVI.BayesianNecessity.positive_evidence`	101; support
`KomorebiPaperVI.BayesianNecessity.refinement_iff`	112; 8.2

13.5 BayesianPartitions: 9 theorem declarations

Source: `KomorebiPaperVI/BayesianPartitions.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.BayesianPartitions.packetMap_surjective`	14; support
`KomorebiPaperVI.BayesianPartitions.packetLikelihood_apply`	24; support
`KomorebiPaperVI.BayesianPartitions.packet_eq_iff`	28; 8.6
`KomorebiPaperVI.BayesianPartitions.packetLikelihood_positive`	33; support
`KomorebiPaperVI.BayesianPartitions.canonical_coarsening`	38; 8.6
`KomorebiPaperVI.BayesianPartitions.canonical_unique_factor`	46; 8.6
`KomorebiPaperVI.BayesianPartitions.all_prior_family_iff_resolves`	67; 8.6
`KomorebiPaperVI.BayesianPartitions.canonical_coarsest_nontrivial`	85; 8.6
`KomorebiPaperVI.BayesianPartitions.canonical_nontrivial`	91; 8.6

13.6 BayesianTowers: 7 theorem declarations

Source: `KomorebiPaperVI/BayesianTowers.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.BayesianTowers.coarsen_comp`	12; 8.5
`KomorebiPaperVI.BayesianTowers.singleton_commutes`	30; 8.2
`KomorebiPaperVI.BayesianTowers.adjacent_criterion`	43; 8.5

Declaration	Line; ledger
`KomorebiPaperVI.BayesianTowers.compose_commutes`	57; support
`KomorebiPaperVI.BayesianTowers.coarsen_id`	70; support
`KomorebiPaperVI.BayesianTowers.tower_commutes`	73; 8.5
`KomorebiPaperVI.BayesianTowers.tower_criterion`	80; 8.5

13.7 BetaNormalization: 15 theorem declarations

Source: `KomorebiPaperVI/BetaNormalization.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.BetaNormalization.beta_value`	12; support
`KomorebiPaperVI.BetaNormalization.change_derivative`	33; support
`KomorebiPaperVI.BetaNormalization.change_injective`	38; support
`KomorebiPaperVI.BetaNormalization.change_image`	46; support
`KomorebiPaperVI.BetaNormalization.change_integrand`	60; support
`KomorebiPaperVI.BetaNormalization.rational_integral`	70; support
`KomorebiPaperVI.BetaNormalization.normalizer_exact`	80; 10.3
`KomorebiPaperVI.BetaNormalization.sine_factor_positive`	88; support
`KomorebiPaperVI.BetaNormalization.scalar_fractional_integral`	95; support
`KomorebiPaperVI.BetaNormalization.beta_real_integral`	99; support
`KomorebiPaperVI.BetaNormalization.weighted_beta_value`	116; support
`KomorebiPaperVI.BetaNormalization.gap_change_integrand`	123; support
`KomorebiPaperVI.BetaNormalization.unit_gap_integral`	132; support
`KomorebiPaperVI.BetaNormalization.gap_integral`	143; support
`KomorebiPaperVI.BetaNormalization.normalized_gap_integral`	164; 10.3

13.8 CanonicalJets: 8 theorem declarations

Source: `KomorebiPaperVI/CanonicalJets.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.CanonicalJets.calculus_on_component`	28; 15.1
`KomorebiPaperVI.CanonicalJets.power_kills`	38; support
`KomorebiPaperVI.CanonicalJets.jet_stable`	46; 15.1
`KomorebiPaperVI.CanonicalJets.shift_intertwines`	58; support
`KomorebiPaperVI.CanonicalJets.power_intertwines`	65; support
`KomorebiPaperVI.CanonicalJets.jet_intertwines`	75; support
`KomorebiPaperVI.CanonicalJets.canonical_naturality`	81; 15.1
`KomorebiPaperVI.CanonicalJets.holomorphic_naturality`	98; support

13.9 Coarsening: 21 theorem declarations

Source: `KomorebiPaperVI/Coarsening.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.Coarsening.coarsen_mass`	15; support
`KomorebiPaperVI.Coarsening.coarsen_nonnegative`	21; support
`KomorebiPaperVI.Coarsening.coarsen_probability`	29; support
`KomorebiPaperVI.Coarsening.coarsen_div`	34; support
`KomorebiPaperVI.Coarsening.coarsen_posterior`	43; support
`KomorebiPaperVI.Coarsening.zero_packet_component`	48; 7.2
`KomorebiPaperVI.Coarsening.zero_packet_weighted`	57; support
`KomorebiPaperVI.Coarsening.effective_weight`	66; 8.1
`KomorebiPaperVI.Coarsening.effective_nonnegative`	73; support
`KomorebiPaperVI.Coarsening.effective_evidence`	78; 8.1
`KomorebiPaperVI.Coarsening.effective_factorization`	83; 8.1
`KomorebiPaperVI.Coarsening.packet_reconstruction_point`	90; 7.2
`KomorebiPaperVI.Coarsening.packet_reconstruction`	99; 7.2
`KomorebiPaperVI.Coarsening.blockwise_weight`	106; support
`KomorebiPaperVI.Coarsening.blockwise_evidence`	115; support
`KomorebiPaperVI.Coarsening.blockwise_sufficiency`	120; support
`KomorebiPaperVI.Coarsening.scaled_blockwise_sufficiency`	126; support
`KomorebiPaperVI.Coarsening.one_packet_posterior`	135; support
`KomorebiPaperVI.Profiles.profile_equivalence`	147; support
`KomorebiPaperVI.Profiles.likelihood_resolving_iff_refines`	153; support
`KomorebiPaperVI.Profiles.exact_profile_factorization`	159; support

13.10 ComplexMemory: 7 theorem declarations

Source: `KomorebiPaperVI/ComplexMemory.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.ComplexMemory.damped_eq_shifted`	18; support
`KomorebiPaperVI.ComplexMemory.shifted_decay`	29; support
`KomorebiPaperVI.ComplexMemory.integral_transfer_right`	43; support
`KomorebiPaperVI.ComplexMemory.laplace_inverse`	55; support
`KomorebiPaperVI.ComplexMemory.memory_self_energy`	77; support
`KomorebiPaperVI.ComplexMemory.normal_growth`	91; support
`KomorebiPaperVI.ComplexMemory.normal_memory_self_energy`	101; support

13.11 ComplexSemigroups: 4 theorem declarations

Source: `KomorebiPaperVI/ComplexSemigroups.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.ComplexSemigroups.duhamel_integrand_derivative`	13; support
`KomorebiPaperVI.ComplexSemigroups.duhamel_rectangular`	29; support
`KomorebiPaperVI.ComplexSemigroups.exact_rectangular_intertwining`	40; support
`KomorebiPaperVI.ComplexSemigroups.contraction_duhamel_bound`	48; support

13.12 ConditionalExpectation: 11 theorem declarations

Source: `KomorebiPaperVI/ConditionalExpectation.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.ConditionalExpectation.fiber_mass_cast`	17; support
`KomorebiPaperVI.ConditionalExpectation.fiber_pull`	23; support
`KomorebiPaperVI.ConditionalExpectation.fiber_mass_ne`	32; support
`KomorebiPaperVI.ConditionalExpectation.expect_constant`	39; support
`KomorebiPaperVI.ConditionalExpectation.expect_block_constant`	46; support
`KomorebiPaperVI.ConditionalExpectation.expect_idempotent`	49; 6.1
`KomorebiPaperVI.ConditionalExpectation.expect_fixes_iff`	53; support
`KomorebiPaperVI.ConditionalExpectation.expect_range`	74; 6.1
`KomorebiPaperVI.ConditionalExpectation.expect_matrix`	84; 6.1
`KomorebiPaperVI.ConditionalExpectation.kernel_weighted_symmetry`	94; support
`KomorebiPaperVI.ConditionalExpectation.expect_self_adjoint`	105; 6.1

13.13 ConditionalTowers: 7 theorem declarations

Source: `KomorebiPaperVI/ConditionalTowers.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.ConditionalTowers.weighted_inner_separates`	11; support
`KomorebiPaperVI.ConditionalTowers.fine_after_coarse`	19; support
`KomorebiPaperVI.ConditionalTowers.coarse_after_fine`	26; support
`KomorebiPaperVI.ConditionalTowers.conditional_kernel_product`	37; 6.2
`KomorebiPaperVI.ConditionalTowers.nested_kernel_min`	53; 6.2
`KomorebiPaperVI.ConditionalTowers.martingale_packet_product`	65; 6.2
`KomorebiPaperVI.ConditionalTowers.martingale_packet_reconstruction`	72; 6.2

13.14 ContourNaturality: 4 theorem declarations

Source: `KomorebiPaperVI/ContourNaturality.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.ContourNaturality.circle_mul_left`	20; support
`KomorebiPaperVI.ContourNaturality.circle_mul_right`	31; support
`KomorebiPaperVI.ContourNaturality.resolvent_intertwines`	44; support
`KomorebiPaperVI.ContourNaturality.common_contour_naturality`	57; support

13.15 DiscreteGreen: 9 theorem declarations

Source: `KomorebiPaperVI/DiscreteGreen.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.DiscreteGreen.green_zero`	13; support
`KomorebiPaperVI.DiscreteGreen.green_end`	15; support
`KomorebiPaperVI.DiscreteGreen.green_second_difference`	22; 11.1
`KomorebiPaperVI.DiscreteGreen.sparse_sum`	52; support
`KomorebiPaperVI.DiscreteGreen.tridiagonal_mul`	65; support
`KomorebiPaperVI.DiscreteGreen.tridiagonal_inverse_kernel`	85; 11.1
`KomorebiPaperVI.DiscreteGreen.inverse_eq_kernel`	109; 11.1
`KomorebiPaperVI.DiscreteGreen.laplacian_inverse`	115; 11.1
`KomorebiPaperVI.DiscreteGreen.grid_reconstruction`	122; 11.1

13.16 FeynmanKac: 10 theorem declarations

Source: `KomorebiPaperVI/FeynmanKac.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.FeynmanKac.sum_paths_succ`	30; support
`KomorebiPaperVI.FeynmanKac.transfer_entry`	37; support
`KomorebiPaperVI.FeynmanKac.path_probability_nonnegative`	41; 12.1
`KomorebiPaperVI.FeynmanKac.path_probability_mass`	48; 12.1
`KomorebiPaperVI.FeynmanKac.path_expectation_zero`	57; support
`KomorebiPaperVI.FeynmanKac.path_expectation_step`	61; support
`KomorebiPaperVI.FeynmanKac.feynman_kac_path_formula`	74; 12.1
`KomorebiPaperVI.FeynmanKac.path_probability_product`	85; 12.1
`KomorebiPaperVI.FeynmanKac.path_potential_sum`	95; 12.1
`KomorebiPaperVI.FeynmanKac.terminal_last`	104; 12.1

13.17 FiniteBayes: 14 theorem declarations

Source: `KomorebiPaperVI/FiniteBayes.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.FiniteBayes.normalized_mass`	15; support
`KomorebiPaperVI.FiniteBayes.posterior_mass`	19; support
`KomorebiPaperVI.FiniteBayes.posterior_nonnegative`	23; support
`KomorebiPaperVI.FiniteBayes.evidence_scale`	27; support
`KomorebiPaperVI.FiniteBayes.posterior_scale`	31; support
`KomorebiPaperVI.FiniteBayes.sequential_evidence`	37; 7.1
`KomorebiPaperVI.FiniteBayes.posterior_composition`	45; 7.1
`KomorebiPaperVI.FiniteBayes.posterior_difference`	53; support
`KomorebiPaperVI.FiniteBayes.scale_null_direction`	65; support
`KomorebiPaperVI.FiniteBayes.normalized_l1_bound_left`	73; 7.3
`KomorebiPaperVI.FiniteBayes.likelihood_perturbation_bound`	102; 7.3, 8.4
`KomorebiPaperVI.FiniteBayes.posterior_probability`	120; support
`KomorebiPaperVI.FiniteBayes.prediction_probability`	124; 9.3
`KomorebiPaperVI.FiniteBayes.prediction_update_probability`	135; 9.3

13.18 FractionalIntegral: 3 theorem declarations

Source: `KomorebiPaperVI/FractionalIntegral.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.FractionalIntegral.common_measure_defect`	14; support
`KomorebiPaperVI.FractionalIntegral.scalar_normalizer_positive`	32; support
`KomorebiPaperVI.FractionalIntegral.scalar_normalized_representation`	36; support

13.19 FractionalNormBounds: 6 theorem declarations

Source: `KomorebiPaperVI/FractionalNormBounds.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.FractionalNormBounds.integrand_saturation`	16; support
`KomorebiPaperVI.FractionalNormBounds.integrand_low_bound`	29; support
`KomorebiPaperVI.FractionalNormBounds.integrand_gap_bound`	44; support
`KomorebiPaperVI.FractionalNormBounds.integrand_high_bound`	61; support
`KomorebiPaperVI.FractionalNormBounds.common_gap_estimate`	67; 10.3
`KomorebiPaperVI.FractionalNormBounds.holder_estimate`	97; 10.3

13.20 FractionalResolvent: 4 theorem declarations

Source: `KomorebiPaperVI/FractionalResolvent.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.FractionalResolvent.cfc_integrand`	13; support
`KomorebiPaperVI.FractionalResolvent.integrand_defect`	26; support
`KomorebiPaperVI.FractionalResolvent.normalized_defect`	38; support
`KomorebiPaperVI.FractionalResolvent.exact_fractional_intertwining`	59; support

13.21 FractionalScalarBounds: 5 theorem declarations

Source: `KomorebiPaperVI/FractionalScalarBounds.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.FractionalScalarBounds.low_integral`	8; support
`KomorebiPaperVI.FractionalScalarBounds.high_integral`	14; support
`KomorebiPaperVI.FractionalScalarBounds.split_domain`	20; support
`KomorebiPaperVI.FractionalScalarBounds.split_norm_bound`	33; support
`KomorebiPaperVI.FractionalScalarBounds.optimize_split`	56; support

13.22 FractionalSemigroup: 8 theorem declarations

Source: `KomorebiPaperVI/FractionalSemigroup.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.FractionalSemigroup.power_nonnegative`	19; support
`KomorebiPaperVI.FractionalSemigroup.exponential_cfc`	21; support
`KomorebiPaperVI.FractionalSemigroup.exponential_contraction`	33; support
`KomorebiPaperVI.FractionalSemigroup.exponential_gap`	42; support
`KomorebiPaperVI.FractionalSemigroup.evolution_transfer`	55; support
`KomorebiPaperVI.FractionalSemigroup.fractional_duhamel`	58; support
`KomorebiPaperVI.FractionalSemigroup.fractional_contraction_bound`	69; support
`KomorebiPaperVI.FractionalSemigroup.fractional_gap_bound`	81; support

13.23 FractionalTowers: 4 theorem declarations

Source: `KomorebiPaperVI/FractionalTowers.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.FractionalTowers.bounded_time_majorant`	15; support

Declaration	Line; ledger
KomorebiPaperVI.FractionalTowers.time_defects_norm_summable	21; support
KomorebiPaperVI.FractionalTowers.time_defects_summable	30; support
KomorebiPaperVI.FractionalTowers.bounded_time_uniform_convergence	35; 10.5

13.24 HolomorphicContours: 3 theorem declarations

Source: KomorebiPaperVI/HolomorphicContours.lean.

Declaration	Line; ledger
KomorebiPaperVI.HolomorphicContours.resolvent_continuousAt	15; support
KomorebiPaperVI.HolomorphicContours.holomorphic_circle_integrable	31; support
KomorebiPaperVI.HolomorphicContours.holomorphic_common_contour_naturality	42; support

13.25 HolomorphicFinite: 11 theorem declarations

Source: KomorebiPaperVI/HolomorphicFinite.lean.

Declaration	Line; ledger
KomorebiPaperVI.HolomorphicFinite.canonical_unique	17; 15.1
KomorebiPaperVI.HolomorphicFinite.holomorphic_on_component	25; support
KomorebiPaperVI.HolomorphicFinite.component_mem_spectrum	33; support
KomorebiPaperVI.HolomorphicFinite.calculus_spectrum_congr	38; 15.1
KomorebiPaperVI.HolomorphicFinite.cauchyCoefficient_eq	58; 15.1
KomorebiPaperVI.HolomorphicFinite.cauchyCoefficient_independent	70; support
KomorebiPaperVI.HolomorphicFinite.admissible_circle	77; support
KomorebiPaperVI.HolomorphicFinite.canonical_cauchy_representation	88; support
KomorebiPaperVI.HolomorphicFinite.canonical_contours_exist	96; 15.1
KomorebiPaperVI.HolomorphicFinite.canonical_contours_independent	113; 15.1
KomorebiPaperVI.HolomorphicFinite.matrix_holomorphic_naturality	129; 15.1

13.26 InformationLoss: 6 theorem declarations

Source: KomorebiPaperVI/InformationLoss.lean.

Declaration	Line; ledger
KomorebiPaperVI.InformationLoss.lossMatrix_apply	12; support
KomorebiPaperVI.InformationLoss.information_loss_matrix	17; 8.4
KomorebiPaperVI.InformationLoss.information_loss_null	28; 8.4
KomorebiPaperVI.InformationLoss.fiber_loss_bound	35; support

Declaration	Line; ledger
`KomorebiPaperVI.InformationLoss.total_loss_bound`	60; 8.4
`KomorebiPaperVI.InformationLoss.information_loss_norm`	68; 8.4

13.27 LowModeConvergence: 2 theorem declarations

Source: `KomorebiPaperVI/LowModeConvergence.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.LowModeConvergence.sine_square_error`	8; 11.2
`KomorebiPaperVI.LowModeConvergence.low_mode_error`	25; 11.2

13.28 MarkovSemigroup: 8 theorem declarations

Source: `KomorebiPaperVI/MarkovSemigroup.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.MarkovSemigroup.matrix_powers_nonnegative`	12; support
`KomorebiPaperVI.MarkovSemigroup.exponential_nonnegative`	20; support
`KomorebiPaperVI.MarkovSemigroup.shifted_generator_nonnegative`	29; support
`KomorebiPaperVI.MarkovSemigroup.scalar_exponential`	40; support
`KomorebiPaperVI.MarkovSemigroup.markov_nonnegative`	48; 4.1
`KomorebiPaperVI.MarkovSemigroup.generator_power_columns`	63; support
`KomorebiPaperVI.MarkovSemigroup.markov_column_sum`	71; 4.1
`KomorebiPaperVI.MarkovSemigroup.markov_probability`	84; 4.1

13.29 MartingalePackets: 2 theorem declarations

Source: `KomorebiPaperVI/MartingalePackets.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.MartingalePackets.packet_product`	12; support
`KomorebiPaperVI.MartingalePackets.packet_sum`	40; support

13.30 MatrixMemory: 1 theorem declarations

Source: `KomorebiPaperVI/MatrixMemory.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.MatrixMemory.rectangular_memory_self_energy`	19; 13.2

13.31 MemoryLaplace: 5 theorem declarations

Source: `KomorebiPaperVI/MemoryLaplace.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.MemoryLaplace.decay_integrable`	13; support
`KomorebiPaperVI.MemoryLaplace.decay_tendsto_zero`	22; support
`KomorebiPaperVI.MemoryLaplace.integral_generator_transfer`	33; support
`KomorebiPaperVI.MemoryLaplace.shifted_laplace_right_inverse`	45; support
`KomorebiPaperVI.MemoryLaplace.memory_integral`	51; support

13.32 NonnormalMemory: 8 theorem declarations

Source: `KomorebiPaperVI/NonnormalMemory.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.NonnormalMemory.left_shift_power`	16; support
`KomorebiPaperVI.NonnormalMemory.component_spectrum`	24; support
`KomorebiPaperVI.NonnormalMemory.exp_mul_component`	38; 13.2
`KomorebiPaperVI.NonnormalMemory.damped_mul_component`	64; support
`KomorebiPaperVI.NonnormalMemory.component_decay`	76; support
`KomorebiPaperVI.NonnormalMemory.spectral_decay`	99; 13.2
`KomorebiPaperVI.NonnormalMemory.spectral_laplace_inverse`	114; 13.2
`KomorebiPaperVI.NonnormalMemory.nonnormal_self_energy`	142; support

13.33 NormalizedFractional: 3 theorem declarations

Source: `KomorebiPaperVI/NormalizedFractional.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.NormalizedFractional.scalar_measure`	19; support
`KomorebiPaperVI.NormalizedFractional.cfc_explicit_measure`	32; support
`KomorebiPaperVI.NormalizedFractional.cfc_lebesgue_integral`	75; 10.3

13.34 OperatorBounds: 3 theorem declarations

Source: `KomorebiPaperVI/OperatorBounds.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.OperatorBounds.growth_duhamel_bound`	10; 3.2
`KomorebiPaperVI.OperatorBounds.contraction_powers`	40; support
`KomorebiPaperVI.OperatorBounds.finite_horizon_contraction`	50; 5.2

13.35 PoissonKernel: 9 theorem declarations

Source: `KomorebiPaperVI/PoissonKernel.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.PoissonKernel.inverse_half_square`	14; support
`KomorebiPaperVI.PoissonKernel.kernel_substitution`	18; support
`KomorebiPaperVI.PoissonKernel.kernel_integrable`	28; support
`KomorebiPaperVI.PoissonKernel.kernel_mass`	35; 10.2
`KomorebiPaperVI.PoissonKernel.kernel_nonnegative`	43; support
`KomorebiPaperVI.PoissonKernel.heat_integrand_bounds`	47; support
`KomorebiPaperVI.PoissonKernel.heat_integrand_integrable`	54; 10.2
`KomorebiPaperVI.PoissonKernel.zero_mode_subordination`	67; support
`KomorebiPaperVI.PoissonKernel.finite_spectral_integrable`	72; support

13.36 PoissonSubordination: 6 theorem declarations

Source: `KomorebiPaperVI/PoissonSubordination.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.PoissonSubordination.scaled_reciprocal_gaussian`	9; support
`KomorebiPaperVI.PoissonSubordination.heat_substitution`	30; support
`KomorebiPaperVI.PoissonSubordination.scalar_subordination`	42; 10.2
`KomorebiPaperVI.PoissonSubordination.finite_spectral_subordination`	52; 10.2
`KomorebiPaperVI.PoissonSubordination.operator_integrand`	65; support
`KomorebiPaperVI.PoissonSubordination.operator_subordination`	74; 10.2

13.37 PositiveResolvent: 9 theorem declarations

Source: `KomorebiPaperVI/PositiveResolvent.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.PositiveResolvent.spectral_positive`	13; support
`KomorebiPaperVI.PositiveResolvent.resolvent_inverse`	18; 10.3
`KomorebiPaperVI.PositiveResolvent.resolvent_left_inverse`	24; support
`KomorebiPaperVI.PositiveResolvent.resolvent_right_inverse`	37; support
`KomorebiPaperVI.PositiveResolvent.resolvent_gap_norm`	45; 10.3
`KomorebiPaperVI.PositiveResolvent.resolvent_norm`	53; support
`KomorebiPaperVI.PositiveResolvent.resolvent_defect`	57; support
`KomorebiPaperVI.PositiveResolvent.saturation_cfc`	71; support
`KomorebiPaperVI.PositiveResolvent.saturation_norm`	79; support

13.38 ReciprocalGaussian: 13 theorem declarations

Source: `KomorebiPaperVI/ReciprocalGaussian.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.ReciprocalGaussian.phi_derivative`	8; support
`KomorebiPaperVI.ReciprocalGaussian.phi_injective`	14; support
`KomorebiPaperVI.ReciprocalGaussian.phi_surjective`	21; support
`KomorebiPaperVI.ReciprocalGaussian.gaussian_jacobian_integrable`	35; support
`KomorebiPaperVI.ReciprocalGaussian.gaussian_jacobian_integral`	45; support
`KomorebiPaperVI.ReciprocalGaussian.reciprocal_derivative`	59; support
`KomorebiPaperVI.ReciprocalGaussian.reciprocal_image`	65; support
`KomorebiPaperVI.ReciprocalGaussian.reciprocal_injective`	74; support
`KomorebiPaperVI.ReciprocalGaussian.reciprocal_phi`	81; support
`KomorebiPaperVI.ReciprocalGaussian.reciprocal_gaussian_symmetry`	86; support
`KomorebiPaperVI.ReciprocalGaussian.phi_gaussian_integrable`	100; support
`KomorebiPaperVI.ReciprocalGaussian.phi_gaussian_integral`	113; support
`KomorebiPaperVI.ReciprocalGaussian.reciprocal_square_gaussian`	127; 10.2

13.39 RectangularFractional: 10 theorem declarations

Source: `KomorebiPaperVI/RectangularFractional.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.RectangularFractional.normalized_integral`	23; support
`KomorebiPaperVI.RectangularFractional.low_bound`	42; support
`KomorebiPaperVI.RectangularFractional.resolvent_defect`	59; support
`KomorebiPaperVI.RectangularFractional.literal_integrand`	80; support

Declaration	Line; ledger
KomorebiPaperVI.RectangularFractional.literal_resolvent_integral	95; 10.3
KomorebiPaperVI.RectangularFractional.exact_fractional_intertwining	110; 10.3
KomorebiPaperVI.RectangularFractional.gap_bound	121; support
KomorebiPaperVI.RectangularFractional.high_bound	139; support
KomorebiPaperVI.RectangularFractional.common_gap_estimate	145; 10.3
KomorebiPaperVI.RectangularFractional.holder_estimate	173; 10.3

13.40 RectangularSemigroup: 8 theorem declarations

Source: KomorebiPaperVI/RectangularSemigroup.lean.

Declaration	Line; ledger
KomorebiPaperVI.RectangularSemigroup.evolution_exponential	19; 10.4
KomorebiPaperVI.RectangularSemigroup.evolution_cfc	23; support
KomorebiPaperVI.RectangularSemigroup.contraction	30; 10.4
KomorebiPaperVI.RectangularSemigroup.gap_decay	35; 10.4
KomorebiPaperVI.RectangularSemigroup.fractional_duhamel	41; 10.4
KomorebiPaperVI.RectangularSemigroup.fractional_contraction_bound	53; 10.4
KomorebiPaperVI.RectangularSemigroup.fractional_gap_bound	63; support
KomorebiPaperVI.RectangularSemigroup.generator_gap_bound	94; 10.4

13.41 RectangularTowers: 9 theorem declarations

Source: KomorebiPaperVI/RectangularTowers.lean.

Declaration	Line; ledger
KomorebiPaperVI.RectangularTowers.holder_constant_positive	21; support
KomorebiPaperVI.RectangularTowers.holder_majorant	27; support
KomorebiPaperVI.RectangularTowers.generator_holder_summable	40; 10.5
KomorebiPaperVI.RectangularTowers.generator_gap_summable	49; 10.5
KomorebiPaperVI.RectangularTowers.bounded_time_majorant	58; 10.5
KomorebiPaperVI.RectangularTowers.time_norms_summable	64; 10.5
KomorebiPaperVI.RectangularTowers.bounded_time_uniform_convergence	72; 10.5
KomorebiPaperVI.RectangularTowers.holder_tower	82; 10.5
KomorebiPaperVI.RectangularTowers.gap_tower	93; 10.5

13.42 ReducingPackets: 3 theorem declarations

Source: `KomorebiPaperVI/ReducingPackets.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.ReducingPackets.generator_commutes`	12; 2.2
`KomorebiPaperVI.ReducingPackets.reducing_exponential`	32; 2.2
`KomorebiPaperVI.ReducingPackets.compressed_evolution`	39; 2.2

13.43 ScalarDecay: 3 theorem declarations

Source: `KomorebiPaperVI/ScalarDecay.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.ScalarDecay.norm_exp_pow`	11; support
`KomorebiPaperVI.ScalarDecay.exp_pow_integrable`	16; 13.2
`KomorebiPaperVI.ScalarDecay.exp_pow_tendsto_zero`	25; 13.2

13.44 SchurFeshbach: 3 theorem declarations

Source: `KomorebiPaperVI/SchurFeshbach.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.SchurFeshbach.resolvent_blocks`	14; 13.1
`KomorebiPaperVI.SchurFeshbach.projected_resolvent`	21; 13.1
`KomorebiPaperVI.SchurFeshbach.schur_invertible`	40; 13.1

13.45 Semigroups: 11 theorem declarations

Source: `KomorebiPaperVI/Semigroups.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.Semigroups.transfer_zero`	18; 1.2
`KomorebiPaperVI.Semigroups.transfer_add`	20; 1.2
`KomorebiPaperVI.Semigroups.transfer_derivative_right`	26; 1.2
`KomorebiPaperVI.Semigroups.transfer_derivative_left`	29; 1.2
`KomorebiPaperVI.Semigroups.transfer_continuous`	33; support
`KomorebiPaperVI.Semigroups.transfer_intertwining`	36; support
`KomorebiPaperVI.Semigroups.exponential_packet_convolution`	44; 2.1
`KomorebiPaperVI.Semigroups.duhamel_integrand_derivative`	57; 3.1
`KomorebiPaperVI.Semigroups.duhamel_rectangular`	70; 3.1

Declaration	Line; ledger
`KomorebiPaperVI.Semigroups.exact_rectangular_intertwining`	81; 3.1
`KomorebiPaperVI.Semigroups.contraction_duhamel_bound`	89; 3.2

13.46 SpectralPackets: 9 theorem declarations

Source: `KomorebiPaperVI/SpectralPackets.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.SpectralPackets.resolve_mul`	12; support
`KomorebiPaperVI.SpectralPackets.resolve_left_packet`	19; support
`KomorebiPaperVI.SpectralPackets.resolve_right_packet`	25; support
`KomorebiPaperVI.SpectralPackets.exp_resolution`	39; support
`KomorebiPaperVI.SpectralPackets.fractional_add`	58; 10.1
`KomorebiPaperVI.SpectralPackets.spectral_transfer_exponential`	68; 10.1
`KomorebiPaperVI.SpectralPackets.spectral_transfer_add`	77; 10.1
`KomorebiPaperVI.SpectralPackets.fractional_reduces`	89; 10.1
`KomorebiPaperVI.SpectralPackets.spectral_transfer_reduces`	96; 10.1

13.47 Survival: 11 theorem declarations

Source: `KomorebiPaperVI/Survival.lean`.

Declaration	Line; ledger
`KomorebiPaperVI.Survival.survival_zero`	23; support
`KomorebiPaperVI.Survival.state_nonnegative`	26; support
`KomorebiPaperVI.Survival.survival_positive`	30; 14.1
`KomorebiPaperVI.Survival.density_nonnegative`	48; support
`KomorebiPaperVI.Survival.survival_derivative`	54; 14.1
`KomorebiPaperVI.Survival.survival_antitone`	69; 14.1
`KomorebiPaperVI.Survival.density_continuous`	78; support
`KomorebiPaperVI.Survival.survival_hazard`	82; 14.1
`KomorebiPaperVI.Survival.survival_has_limit`	100; 14.1
`KomorebiPaperVI.Survival.defective_density_mass`	115; 14.1
`KomorebiPaperVI.Survival.eventual_absorption_density`	129; 14.1

14 Local module imports and locked dependencies

All 47 production modules are imported by `KomorebiPaperVI.lean`. The table gives direct local imports only; Mathlib dependencies remain recorded in the source. Import edges describe module organization, not the complete dependency graph of individual proofs.

Module (targets)	Local imports
`Algebra` (9)	Mathlib only
`BanachDefects` (3)	`Semigroups`
`BayesianFlow` (8)	`FiniteBayes`
`BayesianNecessity` (9)	`Coarsening`
`BayesianPartitions` (9)	`BayesianNecessity`
`BayesianTowers` (7)	`BayesianNecessity`
`BetaNormalization` (15)	Mathlib only
`CanonicalJets` (8)	Mathlib only
`Coarsening` (21)	`FiniteBayes`
`ComplexMemory` (7)	`MemoryLaplace`
`ComplexSemigroups` (4)	`Semigroups`; `FractionalSemigroup`; `RectangularFractional`
`ConditionalExpectation` (11)	`Coarsening`
`ConditionalTowers` (7)	`ConditionalExpectation`; `Algebra`; `MartingalePackets`
`ContourNaturality` (4)	`Algebra`
`DiscreteGreen` (9)	Mathlib only
`FeynmanKac` (10)	Mathlib only
`FiniteBayes` (14)	Mathlib only
`FractionalIntegral` (3)	Mathlib only
`FractionalNormBounds` (6)	`FractionalResolvent`; `FractionalScalarBounds`
`FractionalResolvent` (4)	`NormalizedFractional`; `PositiveResolvent`
`FractionalScalarBounds` (5)	`BetaNormalization`
`FractionalSemigroup` (8)	`Semigroups`; `BanachDefects`
`FractionalTowers` (4)	`FractionalSemigroup`
`HolomorphicContours` (3)	`ContourNaturality`
`HolomorphicFinite` (11)	`CanonicalJets`
`InformationLoss` (6)	`Coarsening`
`LowModeConvergence` (2)	Mathlib only
`MarkovSemigroup` (8)	Mathlib only
`MartingalePackets` (2)	`Algebra`
`MatrixMemory` (1)	`NonnormalMemory`
`MemoryLaplace` (5)	`Semigroups`
`NonnormalMemory` (8)	`ComplexMemory`; `CanonicalJets`; `ScalarDecay`
`NormalizedFractional` (3)	`BetaNormalization`; `FractionalIntegral`
`OperatorBounds` (3)	`Semigroups`
`PoissonKernel` (9)	`SpectralPackets`
`PoissonSubordination` (6)	`PoissonKernel`; `ReciprocalGaussian`
`PositiveResolvent` (9)	`FractionalSemigroup`
`ReciprocalGaussian` (13)	Mathlib only
`RectangularFractional` (10)	`FractionalNormBounds`
`RectangularSemigroup` (8)	`ComplexSemigroups`

Module (targets)	Local imports
`RectangularTowers` (9)	`RectangularSemigroup`
`ReducingPackets` (3)	`Semigroups`
`ScalarDecay` (3)	Mathlib only
`SchurFeshbach` (3)	Mathlib only
`Semigroups` (11)	`Algebra`
`SpectralPackets` (9)	`Semigroups`
`Survival` (11)	`MarkovSemigroup`; `Semigroups`

14.1 Nine pinned packages

Dependency	Locked revision
mathlib	`aeb72f1e9981b6d35b426b16dca7759be3e7ba06`
plausible	`d9598f07b1bc701f1e3aae163d2681c1fd978793`
LeanSearchClient	`ba67e212be1197b84c1f1f6299488a10a3002713`
importGraph	`d8823026ac7ef130c253089d95685f9877b95323`
proofwidgets	`a8acbfd87375ff4abe14ce09db5b7664d383bc7f`
aesop	`18889deb9e83ea7420ef51c160d6f88552e744e3`
Qq	`507746ab8f4b643ccdacb2ec4cdb5853fa9f8ab3`
batteries	`4cac2177c37f5530c4da76aa8e4307f3fc9e4dcb`
Cli	`ab3a82db9fea14cf0fd7f5a2de650f4b534640af`