K-Theory of the Sequence Algebra, Spectral Dimension Three, and Path-Calibrated Spectral Metrics

Finite Certificates and Analytic Scope

Formal Proof Companion

Charles D. Schaper, Ph.D.
Annals of the Chrysene Formalism, Volume 1, Number 4
Article 95, November 2026

1 Formal scope and result

Lean 4.24.0 freshly compiles **42 named finite theorems**. Every theorem has an empty axiom dependency list. No `sorry`, user axiom or native-evaluation axiom is used. Proofs by `decide` are checked by Lean's kernel against the explicit finite definitions.

This is a partial formal companion. It does not formalize operator K-theory, the Bernoulli-shift theorem, infinite spectral traces or Weyl asymptotics, a Connes-distance supremum, general graph duality, or chemistry. In particular, a verified rank recursion is not a formal proof of the crossed-product classification. The mathematical manuscript carries those analytic proofs and dependencies separately.

2 Definitions and theorem coverage

Conditional ranks.
Rank recursion through dimension eight matches a separately defined binomial exterior count. A two-coordinate shear demonstrates the obstruction to inferring a middle action from its endpoint actions.

Finite matrix algebra.
Integer 2×2 Pauli squares and anticommutation are certified. All 49 potential pairs in the finite range $0, \ldots, 6$ satisfy the explicit edge-commutator and squared-singular-value identities. Rational edge scaling is checked separately in Python.

Source metric witnesses.
The explicit option-valued source distance has 21 finite ordered pairs and 204 infinite pairs. All 15 diagonal and all 225 symmetry cases are proved. Distance potentials are Lipschitz on the three edges and attain every finite separation. Component indicators have zero edge seminorm and separate every disconnected pair. These are finite witnesses, not a formalized supremum theorem.

Faithful representation dimensions.
Nine isolated scalar blocks plus three two-dimensional edge blocks give dimension 15; the product spinor-grading fibre is 120.

Cube metric.
The base-three coordinate distance on 27 vertices has zero diagonal, symmetry, diameter six,

and the triangle inequality for every triple. Its equality to shortest-path distance is proved in the manuscript and independently checked in Python, not assumed to be formalized here.

Bypass and finite spectral counts.
The triangle model $\min(w, 2)$ is bounded throughout the tested range, including $w = 100$. Direct integer enumeration gives lattice-ball counts $1, 7, 33, 123$ at radii $0, 1, 2, 3$. No finite radius range establishes the infinite spectral dimension.

3 Reproduction and audit

Install Lean 4.24.0 from its official release and run `python audit_lean.py` with `lean` on the executable path, or set the `LEAN` environment variable. No Mathlib dependency is required. The audit checks the source SHA-256 pin, compiles the file, verifies the axiom report for every named theorem, and records the result in `formal_audit.json`. The source is embedded in this PDF and reproduced below.

4 Complete executable source

Article 95 Lean source

```
/- Article 95. Lean 4.24 core-only finite certificates.
No C*-algebras, infinite Weyl asymptotics, Connes metric supremum,
or molecular realization are formalized here. -/
namespace Article95
set_option maxRecDepth 100000
set_option maxHeartbeats 8000000

def rankPair : Nat -> Prod Nat Nat
| 0 => (1,0)
| n+1 => let p := rankPair n; (p.1+p.2,p.1+p.2)
def binom : Nat -> Nat -> Nat
| _,0 => 1
| 0,_+1 => 0
| n+1,k+1 => binom n k + binom n (k+1)
def evenRank (n : Nat) := ((List.range (n+1)).filter (fun k => k%2==0)).foldl (
    fun a k => a+binom n k) 0
def oddRank (n : Nat) := ((List.range (n+1)).filter (fun k => k%2==1)).foldl (
    fun a k => a+binom n k) 0

theorem rank_zero : rankPair 0 = (1,0) := by decide
theorem rank_one : rankPair 1 = (1,1) := by decide
theorem rank_two : rankPair 2 = (2,2) := by decide
theorem rank_three : rankPair 3 = (4,4) := by decide
theorem exterior_check : forall n : Fin 9, rankPair n.val = (evenRank n.val,
    oddRank n.val) := by decide
theorem bott_rank : binom 3 2 = 3 := by decide

-- Endpoint actions need not determine an extension action.
def shear (p : Prod Int Int) := (p.1+p.2,p.2)
theorem shear_subobject : shear (1,0) = (1,0) := by decide
theorem shear_quotient : (shear (0,1)).2 = 1 := by decide
theorem shear_nonidentity : shear (0,1) != (0,1) := by decide

-- Integer 2x2 matrices represented in row-major order.
```

```
abbrev Mat := Prod Int (Prod Int (Prod Int Int))
def mm (a b : Mat) : Mat :=
 (a.1*b.1+a.2.1*b.2.2.1,
  a.1*b.2.1+a.2.1*b.2.2.2,
  a.2.2.1*b.1+a.2.2.2*b.2.2.1,
  a.2.2.1*b.2.1+a.2.2.2*b.2.2.2)
def ma (a b : Mat) : Mat := (a.1+b.1,a.2.1+b.2.1,a.2.2.1+b.2.2.1,a.2.2.2+b
    .2.2.2)
def neg (a : Mat) : Mat := (-a.1,-a.2.1,-a.2.2.1,-a.2.2.2)
def mt (a : Mat) : Mat := (a.1,a.2.2.1,a.2.1,a.2.2.2)
def diag (u v : Int) : Mat := (u,0,0,v)
def sx : Mat := (0,1,1,0)
def sz : Mat := (1,0,0,-1)
def cm (a b : Mat) := ma (mm a b) (neg (mm b a))
theorem sx_square : mm sx sx = diag 1 1 := by decide
theorem sz_square : mm sz sz = diag 1 1 := by decide
theorem anticommutation : ma (mm sx sz) (mm sz sx) = diag 0 0 := by decide
theorem edge_commutator : forall u v : Fin 7,
 cm sx (diag (Int.ofNat u.val) (Int.ofNat v.val)) =
 (0,Int.ofNat v.val-Int.ofNat u.val,Int.ofNat u.val-Int.ofNat v.val,0) := by
    decide
theorem singular_squares : forall u v : Fin 7,
 let c := cm sx (diag (Int.ofNat u.val) (Int.ofNat v.val))
 let q := (Int.ofNat u.val-Int.ofNat v.val)^2
 mm (mt c) c = diag q q := by decide

-- Source has three edges and nine isolated vertices. Every diagonal is zero.
def srcDist (x y : Nat) : Option Nat :=
 if x==y then some 0 else
 if x/3==y/3 && x/3>=2 && x%3!=0 && y%3!=0 then some 1 else none
def srcFinite := (List.range 15).foldl (fun acc x => acc+
 ((List.range 15).filter (fun y => (srcDist x y).isSome)).length) 0
def srcPot (x z : Nat) : Nat := (srcDist x z).getD 0
def srcComponent (x z : Nat) : Nat := if (srcDist x z).isSome then 1 else 0
def gap (a b : Nat) := a-b+(b-a)
theorem source_diagonal : forall x : Fin 15, srcDist x.val x.val = some 0 := by
    decide
theorem source_symmetry : forall x y : Fin 15, srcDist x.val y.val = srcDist y.
    val x.val := by decide
theorem source_finite_count : srcFinite = 21 := by decide
theorem source_infinite_count : 15*15-srcFinite = 204 := by decide
theorem source_rejected_diagonal : srcDist 0 0 = some 0 := by decide
theorem source_isolate_separation : srcDist 0 1 = none := by decide
theorem source_edge : srcDist 7 8 = some 1 := by decide
theorem source_slices_separate : srcDist 7 10 = none := by decide
theorem faithful_dimension : 2*3+9 = 15 := by decide
theorem product_dimension : 8*(2*3+9) = 120 := by decide
theorem source_potential_lipschitz : forall x : Fin 15, forall n : Fin 3,
 gap (srcPot x.val (3*(n.val+2)+1)) (srcPot x.val (3*(n.val+2)+2)) <= 1 := by
    decide
theorem source_potential_attains : forall x y : Fin 15,
 (srcDist x.val y.val).isSome = true ->
 gap (srcPot x.val x.val) (srcPot x.val y.val) = (srcDist x.val y.val).getD 0 :=
     by decide
theorem component_zero_lipschitz : forall x : Fin 15, forall n : Fin 3,
 srcComponent x.val (3*(n.val+2)+1) = srcComponent x.val (3*(n.val+2)+2) := by
    decide
theorem component_separates : forall x y : Fin 15,
 srcDist x.val y.val = none -> gap (srcComponent x.val x.val) (srcComponent x.
```

```
    val y.val) = 1 := by decide

-- 3x3x3 box represented by base-three coordinates.
def coord (x k : Nat) := (x / 3^k)%3
def boxDist (x y : Nat) := gap (coord x 0) (coord y 0) + gap (coord x 1) (coord
    y 1) + gap (coord x 2) (coord y 2)
theorem box_diagonal : forall x : Fin 27, boxDist x.val x.val = 0 := by decide
theorem box_symmetric : forall x y : Fin 27, boxDist x.val y.val = boxDist y.val
     x.val := by decide
theorem box_diameter : boxDist 0 26 = 6 := by decide
theorem box_bound : forall x y : Fin 27, boxDist x.val y.val <= 6 := by decide
theorem box_triangle : forall x y z : Fin 27, boxDist x.val z.val <= boxDist x.
    val y.val + boxDist y.val z.val := by decide

-- An expensive edge in a triangle retains a two-unit bypass.
def triangleDistance (w : Nat) := min w 2
theorem bypass_at_one : triangleDistance 1 = 1 := by decide
theorem bypass_at_hundred : triangleDistance 100 = 2 := by decide
theorem bypass_bounded : forall w : Fin 101, triangleDistance w.val <= 2 := by
    decide

-- Small exact lattice-ball counts by direct enumeration.
def squareOffset (r n : Nat) : Nat := gap r n * gap r n
def ball (r : Nat) := (List.range (2*r+1)).foldl (fun a x => a+
 (List.range (2*r+1)).foldl (fun b y => b+
 ((List.range (2*r+1)).filter (fun z => squareOffset r x+squareOffset r y+
    squareOffset r z<=r*r)).length) 0) 0
theorem ball_zero : ball 0 = 1 := by decide
theorem ball_one : ball 1 = 7 := by decide
theorem ball_two : ball 2 = 33 := by decide
theorem ball_three : ball 3 = 123 := by decide
theorem zero_mass_one : 8*4*ball 1 = 224 := by decide
theorem source_old_count_rejected : srcFinite != 12 := by decide
end Article95

#print axioms Article95.rank_zero
#print axioms Article95.rank_one
#print axioms Article95.rank_two
#print axioms Article95.rank_three
#print axioms Article95.exterior_check
#print axioms Article95.bott_rank
#print axioms Article95.shear_subobject
#print axioms Article95.shear_quotient
#print axioms Article95.shear_nonidentity
#print axioms Article95.sx_square
#print axioms Article95.sz_square
#print axioms Article95.anticommutation
#print axioms Article95.edge_commutator
#print axioms Article95.singular_squares
#print axioms Article95.source_diagonal
#print axioms Article95.source_symmetry
#print axioms Article95.source_finite_count
#print axioms Article95.source_infinite_count
#print axioms Article95.source_rejected_diagonal
#print axioms Article95.source_isolate_separation
#print axioms Article95.source_edge
#print axioms Article95.source_slices_separate
#print axioms Article95.faithful_dimension
#print axioms Article95.product_dimension
```

```
#print axioms Article95.source_potential_lipschitz
#print axioms Article95.source_potential_attains
#print axioms Article95.component_zero_lipschitz
#print axioms Article95.component_separates
#print axioms Article95.box_diagonal
#print axioms Article95.box_symmetric
#print axioms Article95.box_diameter
#print axioms Article95.box_bound
#print axioms Article95.box_triangle
#print axioms Article95.bypass_at_one
#print axioms Article95.bypass_at_hundred
#print axioms Article95.bypass_bounded
#print axioms Article95.ball_zero
#print axioms Article95.ball_one
#print axioms Article95.ball_two
#print axioms Article95.ball_three
#print axioms Article95.zero_mass_one
#print axioms Article95.source_old_count_rejected
```