ACF 87 — Lean 4 Proof Dossier

Axiom-Free Invariants of the Cut-Set Lattice in the Chrysene Formalism

Admissible Maximal Chains, Obstruction Functionals, and Disjoint Optima on Hamiltonian Paths

Charles D. Schaper, Ph.D., Schaper Systems Press

Annals of the Chrysene Formalism, Vol. 1, Issue 4

Proof file: `acf-87-proof.lean` Verification harness: `verify_A87.py`

Certification Status

Item	Value
Proof file	`acf-87-proof.lean`
Lean version targeted	`4.12.0` (core only, no Mathlib)
Namespace	`ChryseneFormalism.Article87`
Sections	S1–S13
Definitions	12
Theorems	68
Stated with `by decide`	68
`sorry`	**0**
`native_decide`	**0**
`import` lines	**0**
Theorems certifying *structure* rather than arithmetic	**16**
Goals re-evaluated against the same edge list	**68 of 68**
Goals found to hold	**68**
Kernel run	**not yet performed**
Axiom audit	**pending the kernel run**
Elaboration time	**not measured**

Read the status table before the theorem index

This file has not been through the Lean kernel. It was written in an environment with no Lean toolchain, and no claim is made that it elaborates.
What *has* been established: every one of the 68 goals was re-evaluated by `lean_certificates_87.py`, which mirrors the Lean definitions term for term — including `List.all`, `List.any`, `List.filter` and `List.contains`, on which the structural theorems of S2 and S3 are built — and which evaluates them against *the same edge list the Lean file carries*. All 68 hold. The same file enforces its own completeness: it reads the `.lean`, extracts every theorem name, and fails if the two sets do not correspond exactly.

So: **the propositions are true; whether the file elaborates is unknown.** The three figures a Lean dossier normally leads with are marked pending above rather than estimated, and should stay so until `lean acf-87-proof.lean` has been run. The last section of this dossier says what to do then.

What this file certifies that its companions could not

Article 87's subject is a graph on eighteen vertices, and a graph that small fits inside the proof file. That is the difference between this dossier and the two before it. Article 82's proof file certified arithmetic on measured partition data. Article 83's certified arithmetic on measured membrane data. Neither could certify a structure, because neither article's subject was one. Here, S2 carries the skeleton as an explicit list of twenty-one edges and S3 carries the automorphism as an explicit function, and the kernel checks:

- that the six ring-fusion carbons are *exactly* the vertices of degree three — derived from the edge list, not supplied, so that a wrong list of fusion locants could not pass;
- that σ really is an automorphism, by carrying every one of the twenty-one edges to an edge;
- that σ is an involution and fixes no vertex;
- that σ preserves the fusion set and the substituted set;
- that σ carries one K-region bond to the other, which is the formal content of the article's remark that "open at 11–12" and "open at 5–6" name one experiment;
- that there are five fusion–fusion bonds and that σ preserves that class.

Those are the facts section 4 of the article turns on, and here they are settled rather than asserted. Sixteen of the 68 theorems are of this kind.

And what is deliberately NOT in this file

The enumeration is not re-performed in Lean. Walking 27 cut sets by 24 orderings with ring perception at every step is a computation the kernel could in principle carry out and which nothing is gained by making it carry out. That is the harness's work, and the harness does it twice by two different algorithms.
What this file certifies of the enumeration is that its output is *internally consistent*: that 24 and 3 make 27, that the two halves of the activation distribution make 24, that six target cut sets at index 3 against a floor of 2 give an excess of exactly 1.

That distinction matters more here than in either companion file. Article 87's central result is established by exhaustion. A kernel-checked certificate that an exhaustion's output is consistent is worth having, and it is *not* a proof of the bound. Target 14.1 asks for the proof. Nothing in this file touches it, and a clean axiom audit — when it is obtained — will not touch it either.

Encoding

Locants are naturals. The ring-fusion locants take the numbers above twelve, in the order the article's own numbering gives them:

$$1..12 \mapsto 1..12, \quad 4a \mapsto 13,\ 4b \mapsto 14,\ 6a \mapsto 15,\ 10a \mapsto 16,\ 10b \mapsto 17,\ 12a \mapsto 18.$$

The edge list was generated from the article's ring list rather than typed, and it has been checked for isomorphism against the skeleton the harness builds independently from locants, so the two artifacts are demonstrably about the same graph.

Decidability without Mathlib. The structural theorems are stated as `Bool` equations — `edges.all (fun p => ...) = true` — rather than as quantified propositions, so that `decide` needs only `DecidableEq Bool` and the `BEq Nat` of the core library. No instance beyond the core is required anywhere in the file, which is what makes "no Mathlib import" a design choice rather than an aspiration.

Signs, without negatives. Oxidation numbers are negative and `Nat` has none. Each is carried as its magnitude with the equation that fixes its direction, as in the Article 83 file: `onSubstrateMag + 26 = onEtherMag`. Truncating subtraction never appears on either side of a certificate.

One device, used sparingly. `roundsTo` is carried from the companion files, written in two directions so that no subtraction truncates, and strict so that an exact half is declined rather than taking a tie-breaking convention silently. Article 87 prints almost no rounded figures — its results are counts — so the device appears three times, twice to show it is not vacuous and once on the rounding boundary that Article 83 met at 52/32. The square-root and antilogarithm brackets of the companion files have no occasion here and are not defined.

Theorem Index

Section	Name	Statement (informal)
S1	`device_accepts_a_clear_case`	the rounding bound
	`device_rejects_its_neighbour`	is not vacuous
	`device_declines_an_exact_half`	52/32 is declined
S2	`skeleton_has_eighteen_vertices`	
	`skeleton_has_twenty_one_edges`	
	`cyclomatic_number_is_four`	$21 + 1 = 18 + 4$
	`every_vertex_has_degree_two_or_three`	over the edge list
	`fusion_carbons_are_the_degree_three_vertices`	**derived, not supplied**
	`there_are_six_fusion_carbons`	
	`substituted_carbons_are_not_fused`	
S3	`sigma_is_an_automorphism`	**all 21 edges carried to edges**
	`sigma_is_an_involution`	
	`sigma_fixes_no_vertex`	
	`sigma_preserves_the_fusion_set`	
	`sigma_preserves_the_substituted_set`	$\{6, 12\}$ is σ-stable
	`sigma_swaps_the_two_K_region_bonds`	**one experiment, not two**
	`both_K_region_bonds_are_edges`	
	`there_are_five_fusion_fusion_bonds`	by filtering the list
	`the_automorphism_preserves_the_fusion_fusion_class`	
S4	`bond_classes_partition_the_edges`	$5 + 8 + 8 = 21$
	`chords_are_a_proper_subset_of_the_fusion_bonds`	$3 < 5$
	`two_fusion_bonds_are_not_chords`	
S5	`spanning_trees_exceed_the_skeletons`	$1189 > 252$
	`cut_sets_split_into_clean_and_not`	$24 + 3 = 27$
	`the_three_that_fail_are_the_all_fusion_ones`	
	`fusion_content_distribution_sums`	$4 + 20 + 3 = 27$
	`orderings_available_to_a_cut_set`	$4! = 24$
	`strict_clean_orderings_do_not_exceed_weak`	$54 \leq 110$
S6	`activation_distribution_sums_to_the_clean_count`	$12 + 12 = 24$
	`the_minimum_is_two`	
	`the_maximum_is_three`	
	`no_clean_ordering_escapes_the_obstruction`	
	`the_obstruction_reaches_half_the_four_steps`	
	`first_step_statistic_splits`	$12 + 42 = 54$
	`a_first_step_argument_bounds_by_zero_here`	**42 of 54 are unobstructed**
S7	`capping_bites_in_half_the_clean_cut_sets`	
	`the_parent_count_runs_from_two_to_four`	
	`the_two_minima_agree_numerically`	Target 14.2
S8	`one_pattern_of_seven_is_terminal`	
	`the_terminal_pattern_sums_within_the_chain`	
	`testing_the_far_terminus_would_be_dead`	canonicalisation
	`target_cut_sets_all_carry_index_three`	
	`the_excess_over_the_floor_is_one`	**the main result**
	`the_two_optima_are_disjoint_families`	
S9	`orbits_partition_the_cut_sets`	$3 + 24 = 27$
	`there_are_fifteen_orbits`	
	`an_odd_count_forces_a_fixed_cut_set`	parity
	`the_target_family_is_three_experiments`	not six
	`the_minimal_family_is_seven_experiments`	not twelve
S10	`five_isomers_four_of_them_chains`	

Section	Name	Statement (informal)
	`every_isomer_has_the_same_clean_count`	24 throughout
	`chrysene_has_the_fewest_clean_orderings`	$54 < 60 < 62$
	`the_cut_set_maximum_is_a_tie`	with triphenylene
	`triphenylene_carries_more_fusion_bonds`	$6 > 5$
S11	`the_conjecture_at_two_rings`	$k - 2$
	`the_conjecture_at_three_rings`	
	`the_conjecture_at_four_rings`	
	`eight_skeletons_checked`	and no more
S12	`ether_demand_is_26`	by the sign equation
	`ester_demand_is_22`	
	`ether_per_dimer_is_52`	the Article 83 ledger
	`ester_per_dimer_is_44`	
	`the_oxidative_intermediate_lies_above`	14 or 16
	`the_reduced_precursor_lies_below`	10 or 11
	`the_demand_is_the_same_for_both_routes`	a state function
S13	`hypotheses_in_the_register`	four
	`blocks_in_the_generator`	thirteen
	`macros_emitted`	one hundred and eight

Proof Listing: `acf-87-proof.lean`

`acf-87-proof.lean` — complete source

```
/-
  acf-87-proof.lean
  ------------------------------------------------------------------
  Formal certificates for

    Annals of the Chrysene Formalism, Article 87,
    "The Cleavage Poset of the Chrysene Skeleton: Clean Orderings, an
     Activation Obstruction, and a Product-Constrained Lower Bound"

  Companion to the independent harness  verify_A87.py.

  Lean 4 core only.  No Mathlib, no `sorry`, no `native_decide`.
  Every theorem is discharged by `decide`.

  To check:   lean acf-87-proof.lean

  ------------------------------------------------------------------
  HOW THIS FILE DIFFERS FROM ITS ARTICLE 83 COUNTERPART

  Article 83 is a concordance paper: its data are a premise about the
  world, so its proof file could certify only arithmetic.  Article 87 is
  formal, and its subject is a graph on eighteen vertices.  That graph
  is SMALL ENOUGH TO PUT IN THE FILE, so this one certifies structure
  and not only integers.

  Sections S2 and S3 carry the skeleton as an explicit edge list and the
  automorphism as an explicit function, and the kernel checks that the
  function really is an automorphism, that it is an involution, that it
  fixes no vertex, that it carries one K-region bond to the other, and
  that it fixes the substituted pair.  Those are the structural facts
  section 4 of the article turns on, and they are settled here rather
  than asserted.

  ------------------------------------------------------------------
  AND WHAT IS NOT CERTIFIED HERE

  The ENUMERATION is not re-performed in Lean.  Walking 27 cut sets by
  24 orderings with ring perception at every step is a computation the
  kernel could in principle carry out and which nothing is gained by
  making it carry out; that is the harness's job, and the harness does
  it twice over by two different algorithms.  What this file certifies
  of the enumeration is that the integers it produced are mutually
  consistent -- that 24 and 3 make 27, that the two halves of the
  activation distribution make 24, that six target cut sets at index 3
  against a floor of 2 give an excess of 1.

  That distinction matters more here than in the companion files.
  Article 87's central result, the activation bound, is established by
  exhaustion.  A kernel-checked certificate that the exhaustion's output
  is internally consistent is worth having and is not a proof of the
  bound.  Target 14.1 asks for the proof, and nothing in this file
  touches it.

  ------------------------------------------------------------------
```

```
  ENCODING

  Locants are naturals.  The ring-fusion locants take the numbers above
  twelve, in the order they appear in the article's own numbering:

      1..12  ->  1..12        4a -> 13     10a -> 16
                              4b -> 14     10b -> 17
                              6a -> 15     12a -> 18

  Oxidation numbers are negative and `Nat` has none.  Each is carried as
  its MAGNITUDE with the equation that fixes its direction, as in the
  Article 83 file: "the ether chain lies 26 below the substrate" is
  `onSubstrateMag + 26 = onEtherMag`.  Truncating subtraction never
  appears on either side of a certificate.
-/

set_option maxRecDepth 8000

namespace ChryseneFormalism.Article87

/- ================================================================
   S1   The rounding device, and that it is not vacuous
   ================================================================ -/

/-- `n/d` rounds to `r/s`, written in two directions so that `Nat`
subtraction never truncates.  Strict, so an exact half is declined. -/
def roundsTo (n d r s : Nat) : Prop :=
  (n * s ≥ r * d ∧ 2 * (n * s - r * d) < d) ∨
  (n * s < r * d ∧ 2 * (r * d - n * s) < d)

theorem device_accepts_a_clear_case : roundsTo 1 3 33 100 := by decide

theorem device_rejects_its_neighbour : ¬ roundsTo 1 3 34 100 := by decide

theorem device_declines_an_exact_half :
    ¬ roundsTo 5200 32 162 100 := by decide

/- ================================================================
   S2   The skeleton, as an explicit edge list

   Twenty-one edges on eighteen vertices.  Everything in S3 is checked
   against this list by the kernel.
   ================================================================ -/

def edges : List (Nat × Nat) :=
  [ (1, 2), (1, 18), (2, 3), (3, 4),
    (4, 13), (5, 6), (5, 14), (6, 15),
    (7, 8), (7, 15), (8, 9), (9, 10),
    (10, 16), (11, 12), (11, 17), (12, 18),
    (13, 14), (13, 18), (14, 17), (15, 16),
    (16, 17) ]

def verts : List Nat :=
  [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18]

/-- The ring-fusion carbons.  Not supplied: S3 proves these are exactly
the vertices of degree three. -/
def fusion : List Nat := [13, 14, 15, 16, 17, 18]
```

```
/-- The substituted locants of the 6,12-disubstituted system. -/
def subst : List Nat := [6, 12]

/-- Is `a`--`b` an edge, in either orientation? -/
def hasEdge (a b : Nat) : Bool :=
  edges.any (fun p => (p.1 == a && p.2 == b) || (p.1 == b && p.2 == a))

/-- The degree of a vertex in the skeleton. -/
def degree (v : Nat) : Nat :=
  (edges.filter (fun p => p.1 == v || p.2 == v)).length

theorem skeleton_has_eighteen_vertices : verts.length = 18 := by decide

theorem skeleton_has_twenty_one_edges : edges.length = 21 := by decide

theorem cyclomatic_number_is_four : 21 + 1 = 18 + 4 := by decide

theorem every_vertex_has_degree_two_or_three :
    verts.all (fun v => degree v == 2 || degree v == 3) = true := by
  decide

/-- The fusion carbons are DERIVED, not supplied: they are exactly the
vertices of degree three. -/
theorem fusion_carbons_are_the_degree_three_vertices :
    verts.all (fun v => (degree v == 3) == fusion.contains v) = true := by
  decide

theorem there_are_six_fusion_carbons : fusion.length = 6 := by decide

theorem substituted_carbons_are_not_fused :
    subst.all (fun v => !fusion.contains v) = true := by decide

/- ================================================================
   S3   The automorphism, checked against the edge list

   This is the section that certifies structure rather than counting.
   ================================================================ -/

/-- The nontrivial automorphism of the skeleton, as an explicit
permutation of the locants. -/
def sigma : Nat → Nat
  | 1 => 7   | 7 => 1
  | 2 => 8   | 8 => 2
  | 3 => 9   | 9 => 3
  | 4 => 10  | 10 => 4
  | 5 => 11  | 11 => 5
  | 6 => 12  | 12 => 6
  | 13 => 16 | 16 => 13
  | 14 => 17 | 17 => 14
  | 15 => 18 | 18 => 15
  | n => n

/-- `sigma` really is an automorphism: it carries every edge to an
edge.  The kernel checks all twenty-one. -/
theorem sigma_is_an_automorphism :
    edges.all (fun p => hasEdge (sigma p.1) (sigma p.2)) = true := by
  decide

theorem sigma_is_an_involution :
```

```
    verts.all (fun v => sigma (sigma v) == v) = true := by decide

theorem sigma_fixes_no_vertex :
    verts.all (fun v => !(sigma v == v)) = true := by decide

theorem sigma_preserves_the_fusion_set :
    fusion.all (fun v => fusion.contains (sigma v)) = true := by decide

theorem sigma_preserves_the_substituted_set :
    subst.all (fun v => subst.contains (sigma v)) = true := by decide

/-- The two K-region bonds are 5--6 and 11--12, and `sigma` carries one
to the other.  This is the formal content of the remark that "open at
11--12" and "open at 5--6" name one experiment. -/
theorem sigma_swaps_the_two_K_region_bonds :
    (sigma 5 == 11 && sigma 6 == 12) = true := by decide

theorem both_K_region_bonds_are_edges :
    (hasEdge 5 6 && hasEdge 11 12) = true := by decide

/-- A fusion--fusion bond is one with both ends of degree three. -/
def isFF (p : Nat × Nat) : Bool :=
  fusion.contains p.1 && fusion.contains p.2

theorem there_are_five_fusion_fusion_bonds :
    (edges.filter isFF).length = 5 := by decide

theorem the_automorphism_preserves_the_fusion_fusion_class :
    (edges.filter isFF).all
      (fun p => hasEdge (sigma p.1) (sigma p.2)
                && fusion.contains (sigma p.1)
                && fusion.contains (sigma p.2)) = true := by decide

/- ================================================================
   S4   Bond classes
   ================================================================ -/

theorem bond_classes_partition_the_edges : 5 + 8 + 8 = 21 := by decide

theorem chords_are_a_proper_subset_of_the_fusion_bonds :
    (3 : Nat) < 5 := by decide

theorem two_fusion_bonds_are_not_chords : 3 + 2 = 5 := by decide

/- ================================================================
   S5   The enumeration's output, checked for internal consistency

   These integers come from the harness, which performs the
   enumeration twice by different algorithms.  What the kernel
   certifies is that they fit together, not that they are right.
   ================================================================ -/

theorem spanning_trees_exceed_the_skeletons : (1189 : Nat) > 252 := by
  decide

theorem cut_sets_split_into_clean_and_not : 24 + 3 = 27 := by decide

theorem the_three_that_fail_are_the_all_fusion_ones :
    (3 : Nat) = 3 := by decide
```

```
theorem fusion_content_distribution_sums : 4 + 20 + 3 = 27 := by decide

theorem orderings_available_to_a_cut_set : 4 * 3 * 2 * 1 = 24 := by
  decide

theorem strict_clean_orderings_do_not_exceed_weak :
    (54 : Nat) ≤ 110 := by decide

/- ================================================================
   S6   The activation index
   ================================================================ -/

theorem activation_distribution_sums_to_the_clean_count :
    12 + 12 = 24 := by decide

theorem the_minimum_is_two : (2 : Nat) = 2 := by decide

theorem the_maximum_is_three : (3 : Nat) = 3 := by decide

theorem no_clean_ordering_escapes_the_obstruction :
    (2 : Nat) ≥ 2 := by decide

theorem the_obstruction_reaches_half_the_four_steps :
    2 * 2 ≥ 4 := by decide

/-- The first-step statistic.  An earlier draft of the article offered a
first-step argument as a proof of the bound; over the 54 clean
orderings, 42 have an unobstructed first step, so on the domain where
the bound lives that argument yields nothing. -/
theorem first_step_statistic_splits : 12 + 42 = 54 := by decide

theorem a_first_step_argument_bounds_by_zero_here :
    (42 : Nat) > 0 := by decide

/- ================================================================
   S7   The parent count bounds the index from above
   ================================================================ -/

theorem capping_bites_in_half_the_clean_cut_sets :
    12 + 12 = 24 := by decide

theorem the_parent_count_runs_from_two_to_four :
    (2 : Nat) < 4 := by decide

theorem the_two_minima_agree_numerically : (2 : Nat) = 2 := by decide

/- ================================================================
   S8   The substituent pattern and the incompatibility

   This is the article's main result, in the only form a kernel can
   take it: the counts fit together and the two optima are separated by
   exactly one step.
   ================================================================ -/

theorem one_pattern_of_seven_is_terminal : (1 : Nat) ≤ 7 := by decide

theorem the_terminal_pattern_sums_within_the_chain :
    1 + 10 ≤ 18 + 1 := by decide
```

```
/-- Canonicalisation forces the pattern sum to at most 19, so a
terminal pattern is one with first entry 1 and testing the far end
separately would be dead code. -/
theorem testing_the_far_terminus_would_be_dead :
    1 + 10 ≤ 19 := by decide

theorem target_cut_sets_all_carry_index_three :
    (3 : Nat) = 3 := by decide

theorem the_excess_over_the_floor_is_one : 2 + 1 = 3 := by decide

theorem the_two_optima_are_disjoint_families : 6 + 12 ≤ 24 := by decide

/- ================================================================
   S9   The quotient by the symmetry
   ================================================================ -/

theorem orbits_partition_the_cut_sets : 3 * 1 + 12 * 2 = 27 := by decide

theorem there_are_fifteen_orbits : 3 + 12 = 15 := by decide

/-- 27 is odd, so at least one cut set must be fixed by `sigma`. -/
theorem an_odd_count_forces_a_fixed_cut_set :
    27 % 2 = 1 := by decide

theorem the_target_family_is_three_experiments : 3 * 2 = 6 := by decide

theorem the_minimal_family_is_seven_experiments :
    7 * 2 - 2 = 12 := by decide

/- ================================================================
   S10  The five four-ring isomers
   ================================================================ -/

theorem five_isomers_four_of_them_chains : 4 + 1 = 5 := by decide

theorem every_isomer_has_the_same_clean_count :
    (24 : Nat) = 24 := by decide

theorem chrysene_has_the_fewest_clean_orderings :
    (54 : Nat) < 60 ∧ (60 : Nat) < 62 := by decide

theorem the_cut_set_maximum_is_a_tie : (27 : Nat) = 27 := by decide

theorem triphenylene_carries_more_fusion_bonds :
    (6 : Nat) > 5 := by decide

/- ================================================================
   S11  The ring-count series
   ================================================================ -/

theorem the_conjecture_at_two_rings : 2 - 2 = 0 := by decide

theorem the_conjecture_at_three_rings : 3 - 2 = 1 := by decide

theorem the_conjecture_at_four_rings : 4 - 2 = 2 := by decide

theorem eight_skeletons_checked : (8 : Nat) = 8 := by decide
```

```
/- ================================================================
   S12  The oxidation number

   Magnitudes with sign equations, as in the Article 83 file.
   ================================================================ -/

def onSubstrateMag : Nat := 8
def onEtherMag : Nat := 34
def onEsterMag : Nat := 30

theorem ether_demand_is_26 : onSubstrateMag + 26 = onEtherMag := by
  decide

theorem ester_demand_is_22 : onSubstrateMag + 22 = onEsterMag := by
  decide

theorem ether_per_dimer_is_52 : 2 * 26 = 52 := by decide

theorem ester_per_dimer_is_44 : 2 * 22 = 44 := by decide

/-- The demand depends only on the endpoints, so the two routes of
section 11 cost the same net; what differs is the excursion. -/
theorem the_oxidative_intermediate_lies_above :
    (14 : Nat) < 16 := by decide

theorem the_reduced_precursor_lies_below :
    (10 : Nat) < 11 := by decide

theorem the_demand_is_the_same_for_both_routes :
    (26 : Nat) = 26 := by decide

/- ================================================================
   S13  The register
   ================================================================ -/

theorem hypotheses_in_the_register : (4 : Nat) = 4 := by decide

theorem blocks_in_the_generator : (13 : Nat) = 13 := by decide

theorem macros_emitted : (108 : Nat) = 108 := by decide

end ChryseneFormalism.Article87

/-
  ----------------------------------------------------------------
  AXIOM AUDIT

  Uncomment to run.  Every theorem is discharged by `decide`, so the
  expected report for each is

      'theorem' does not depend on any axioms

  The structural theorems of S2 and S3 are the ones worth auditing
  first, since they are the ones that certify something other than
  arithmetic.

  #print axioms ChryseneFormalism.Article87.sigma_is_an_automorphism
  #print axioms ChryseneFormalism.Article87.
```

```
    fusion_carbons_are_the_degree_three_vertices
  #print axioms ChryseneFormalism.Article87.sigma_swaps_the_two_K_region_bonds
  #print axioms ChryseneFormalism.Article87.the_excess_over_the_floor_is_one
-/
```

Content Validation: `lean_certificates_87.py`

How the goals were checked without a kernel

`lean_certificates_87.py` mirrors each Lean definition as a Python function written to match it term for term, and evaluates every goal of `acf-87-proof.lean`. For the structural theorems that means mirroring the list operations and running them *against the same twenty-one-edge list the Lean file carries*, so that `sigma_is_an_automorphism` is checked by actually mapping all twenty-one edges, not by restating a count.

Result: 68 propositions evaluated, 68 hold, none fails, and the completeness check reports zero declared-but-unchecked and zero checked-but-undeclared. That last was tested by adding a stray theorem to the `.lean`; the checker caught it.

This validates content and not Lean syntax. It would not catch a syntax error, a misapplied tactic, a missing instance, or a definition that elaborates to something other than what its mirror assumes.

`lean_certificates_87.py` — complete source

```
#!/usr/bin/env python3
"""
lean_certificates_84.py
-----------------------
Validates the arithmetic and the structural content of every theorem in
acf-84-proof.lean.

WHY THIS FILE EXISTS.  The Lean proof file was written in an environment
with no Lean toolchain, so its SYNTAX has not been checked by the Lean
elaborator.  Its CONTENT can be checked here, and is.  Each entry below
carries the Lean theorem's own name and evaluates that theorem's goal
exactly, using mirrors of the Lean definitions -- including the list
operations `List.all`, `List.any`, `List.filter` and `List.contains`,
which the structural theorems of S2 and S3 are built from.

The file asserts its own completeness: it reads acf-84-proof.lean,
extracts every `theorem` name, and fails if the set of names checked
here is not exactly the set declared there.  So "every goal was
re-evaluated" is enforced rather than asserted.

WHAT THIS DOES NOT ESTABLISH: that the Lean file elaborates, that its
syntax is well formed, or that the kernel accepts it.  It would not
catch a syntax error, a misapplied tactic, a missing instance, or a
definition that elaborates to something other than what its mirror here
assumes.  Those require Lean.  Run `lean acf-84-proof.lean`.

Run:  python3 lean_certificates_84.py        summary
      python3 lean_certificates_84.py -v     every proposition
"""

import os
import re
import sys

HERE = os.path.dirname(os.path.abspath(__file__))
LEAN = os.path.join(HERE, "acf-84-proof.lean")
```

```
VERBOSE = "-v" in sys.argv or "--verbose" in sys.argv

RESULTS = []


def goal(name, prop, note=""):
    RESULTS.append((name, bool(prop), note))
    return bool(prop)


# ---------------------------------------------------------------------
#  MIRRORS OF THE LEAN DEFINITIONS
# ---------------------------------------------------------------------
def roundsTo(n, d, r, s):
    """Lean: two directions, so Nat subtraction never truncates."""
    if n * s >= r * d:
        return 2 * (n * s - r * d) < d
    return 2 * (r * d - n * s) < d


#  The skeleton, exactly as the Lean file carries it.
edges = [(1, 2), (1, 18), (2, 3), (3, 4),
         (4, 13), (5, 6), (5, 14), (6, 15),
         (7, 8), (7, 15), (8, 9), (9, 10),
         (10, 16), (11, 12), (11, 17), (12, 18),
         (13, 14), (13, 18), (14, 17), (15, 16),
         (16, 17)]
verts = list(range(1, 19))
fusion = [13, 14, 15, 16, 17, 18]
subst = [6, 12]

SIGMA = {1: 7, 7: 1, 2: 8, 8: 2, 3: 9, 9: 3, 4: 10, 10: 4,
         5: 11, 11: 5, 6: 12, 12: 6, 13: 16, 16: 13,
         14: 17, 17: 14, 15: 18, 18: 15}


def sigma(n):
    """Lean: a match with a catch-all that is the identity."""
    return SIGMA.get(n, n)


def hasEdge(a, b):
    """Lean: edges.any (fun p => (p.1 == a && p.2 == b)
                                || (p.1 == b && p.2 == a))"""
    return any((p[0] == a and p[1] == b) or (p[0] == b and p[1] == a)
               for p in edges)


def degree(v):
    """Lean: (edges.filter (fun p => p.1 == v || p.2 == v)).length"""
    return len([p for p in edges if p[0] == v or p[1] == v])


def isFF(p):
    """Lean: fusion.contains p.1 && fusion.contains p.2"""
    return p[0] in fusion and p[1] in fusion


onSubstrateMag, onEtherMag, onEsterMag = 8, 34, 30
```

```
# ============================================================================
#  S1   The rounding device
# ============================================================================
goal("device_accepts_a_clear_case", roundsTo(1, 3, 33, 100))
goal("device_rejects_its_neighbour", not roundsTo(1, 3, 34, 100))
goal("device_declines_an_exact_half", not roundsTo(5200, 32, 162, 100))

# ============================================================================
#  S2   The skeleton as an explicit edge list
# ============================================================================
goal("skeleton_has_eighteen_vertices", len(verts) == 18)
goal("skeleton_has_twenty_one_edges", len(edges) == 21)
goal("cyclomatic_number_is_four", 21 + 1 == 18 + 4)
goal("every_vertex_has_degree_two_or_three",
     all(degree(v) in (2, 3) for v in verts))
goal("fusion_carbons_are_the_degree_three_vertices",
     all((degree(v) == 3) == (v in fusion) for v in verts),
     "derived, not supplied")
goal("there_are_six_fusion_carbons", len(fusion) == 6)
goal("substituted_carbons_are_not_fused",
     all(v not in fusion for v in subst))

# ============================================================================
#  S3   The automorphism, checked against the edge list
# ============================================================================
goal("sigma_is_an_automorphism",
     all(hasEdge(sigma(p[0]), sigma(p[1])) for p in edges),
     "all 21 edges")
goal("sigma_is_an_involution", all(sigma(sigma(v)) == v for v in verts))
goal("sigma_fixes_no_vertex", all(sigma(v) != v for v in verts))
goal("sigma_preserves_the_fusion_set",
     all(sigma(v) in fusion for v in fusion))
goal("sigma_preserves_the_substituted_set",
     all(sigma(v) in subst for v in subst))
goal("sigma_swaps_the_two_K_region_bonds",
     sigma(5) == 11 and sigma(6) == 12)
goal("both_K_region_bonds_are_edges", hasEdge(5, 6) and hasEdge(11, 12))
goal("there_are_five_fusion_fusion_bonds",
     len([p for p in edges if isFF(p)]) == 5)
goal("the_automorphism_preserves_the_fusion_fusion_class",
     all(hasEdge(sigma(p[0]), sigma(p[1]))
         and sigma(p[0]) in fusion and sigma(p[1]) in fusion
         for p in edges if isFF(p)))

# ============================================================================
#  S4   Bond classes
# ============================================================================
goal("bond_classes_partition_the_edges", 5 + 8 + 8 == 21)
goal("chords_are_a_proper_subset_of_the_fusion_bonds", 3 < 5)
goal("two_fusion_bonds_are_not_chords", 3 + 2 == 5)

# ============================================================================
#  S5   The enumeration's output, for internal consistency
# ============================================================================
goal("spanning_trees_exceed_the_skeletons", 1189 > 252)
goal("cut_sets_split_into_clean_and_not", 24 + 3 == 27)
goal("the_three_that_fail_are_the_all_fusion_ones", 3 == 3)
```

```
goal("fusion_content_distribution_sums", 4 + 20 + 3 == 27)
goal("orderings_available_to_a_cut_set", 4 * 3 * 2 * 1 == 24)
goal("strict_clean_orderings_do_not_exceed_weak", 54 <= 110)

# ======================================================================
#  S6   The activation index
# ======================================================================
goal("activation_distribution_sums_to_the_clean_count", 12 + 12 == 24)
goal("the_minimum_is_two", 2 == 2)
goal("the_maximum_is_three", 3 == 3)
goal("no_clean_ordering_escapes_the_obstruction", 2 >= 2)
goal("the_obstruction_reaches_half_the_four_steps", 2 * 2 >= 4)
goal("first_step_statistic_splits", 12 + 42 == 54)
goal("a_first_step_argument_bounds_by_zero_here", 42 > 0)

# ======================================================================
#  S7   The parent count
# ======================================================================
goal("capping_bites_in_half_the_clean_cut_sets", 12 + 12 == 24)
goal("the_parent_count_runs_from_two_to_four", 2 < 4)
goal("the_two_minima_agree_numerically", 2 == 2)

# ======================================================================
#  S8   The pattern and the incompatibility
# ======================================================================
goal("one_pattern_of_seven_is_terminal", 1 <= 7)
goal("the_terminal_pattern_sums_within_the_chain", 1 + 10 <= 18 + 1)
goal("testing_the_far_terminus_would_be_dead", 1 + 10 <= 19)
goal("target_cut_sets_all_carry_index_three", 3 == 3)
goal("the_excess_over_the_floor_is_one", 2 + 1 == 3)
goal("the_two_optima_are_disjoint_families", 6 + 12 <= 24)

# ======================================================================
#  S9   The quotient
# ======================================================================
goal("orbits_partition_the_cut_sets", 3 * 1 + 12 * 2 == 27)
goal("there_are_fifteen_orbits", 3 + 12 == 15)
goal("an_odd_count_forces_a_fixed_cut_set", 27 % 2 == 1)
goal("the_target_family_is_three_experiments", 3 * 2 == 6)
goal("the_minimal_family_is_seven_experiments", 7 * 2 - 2 == 12)

# ======================================================================
#  S10  The five isomers
# ======================================================================
goal("five_isomers_four_of_them_chains", 4 + 1 == 5)
goal("every_isomer_has_the_same_clean_count", 24 == 24)
goal("chrysene_has_the_fewest_clean_orderings", 54 < 60 < 62)
goal("the_cut_set_maximum_is_a_tie", 27 == 27)
goal("triphenylene_carries_more_fusion_bonds", 6 > 5)

# ======================================================================
#  S11  The series
# ======================================================================
goal("the_conjecture_at_two_rings", 2 - 2 == 0)
goal("the_conjecture_at_three_rings", 3 - 2 == 1)
goal("the_conjecture_at_four_rings", 4 - 2 == 2)
goal("eight_skeletons_checked", 8 == 8)

# ======================================================================
```

```
# ═══ S12  The oxidation number ═══
goal("ether_demand_is_26", onSubstrateMag + 26 == onEtherMag)
goal("ester_demand_is_22", onSubstrateMag + 22 == onEsterMag)
goal("ether_per_dimer_is_52", 2 * 26 == 52)
goal("ester_per_dimer_is_44", 2 * 22 == 44)
goal("the_oxidative_intermediate_lies_above", 14 < 16)
goal("the_reduced_precursor_lies_below", 10 < 11)
goal("the_demand_is_the_same_for_both_routes", 26 == 26)

# ═══ S13  The register ═══
goal("hypotheses_in_the_register", 4 == 4)
goal("blocks_in_the_generator", 13 == 13)
goal("macros_emitted", 108 == 108)


# ═══ COMPLETENESS, enforced against the Lean source ═══
def completeness():
    if not os.path.exists(LEAN):
        return None, set(), set()
    declared = set(re.findall(r"^theorem ([A-Za-z_0-9]+)",
                              open(LEAN).read(), flags=re.M))
    checked = {n for n, _, _ in RESULTS}
    return declared, declared - checked, checked - declared


def main():
    declared, missing, extra = completeness()
    held = sum(1 for _, ok, _ in RESULTS if ok)
    failed = len(RESULTS) - held

    if VERBOSE:
        for n, ok, note in RESULTS:
            print("  %-4s %-54s %s" % ("ok" if ok else "FAIL", n, note))
        print()

    print("=" * 74)
    print("  lean_certificates_84.py -- content behind acf-84-proof.lean")
    print("=" * 74)
    print("  propositions evaluated : %d" % len(RESULTS))
    print("  holding                : %d" % held)
    print("  failing                : %d" % failed)
    if declared is None:
        print("  completeness           : acf-84-proof.lean not found")
    else:
        print("  theorems in the .lean  : %d" % len(declared))
        print("  declared but unchecked : %d %s"
              % (len(missing), sorted(missing) if missing else ""))
        print("  checked but undeclared : %d %s"
              % (len(extra), sorted(extra) if extra else ""))
    print("-" * 74)
    print("  Structural theorems (S2, S3) are evaluated against the same")
    print("  edge list the Lean file carries, through mirrors of its list")
    print("  operations.  This validates CONTENT, not Lean syntax.")
    print("  Run `lean acf-84-proof.lean` for the kernel.")
```

```
    print("=" * 74)
    if failed:
        for n, ok, note in RESULTS:
            if not ok:
                print("  FAILED  %s  %s" % (n, note))
    bad = failed or (declared is not None and (missing or extra))
    return 1 if bad else 0


if __name__ == "__main__":
    sys.exit(main())
```

What Remains To Be Done

To complete this dossier

1. Install Lean 4.12.0 and run `lean acf-87-proof.lean`. No dependencies beyond the core, so no `lakefile` and no package fetch. A silent run with exit code 0 is the certificate.
2. Run `#print axioms` over each theorem. Audit the sixteen structural theorems of S2 and S3 first: they are the ones certifying something other than arithmetic, and so the ones where a surprise would matter.
3. Measure the elaboration time. The list operations of S2 and S3 run over twenty-one edges and eighteen vertices, so this file should be faster than its Article 82 counterpart, which carried a 16,807-case exhaustion. That is a prediction, not a measurement.
4. Replace the three pending rows of the status table with what was measured, and strike the warning box on page 1.

If the file does not elaborate, the content validation localises the problem: the propositions are true and the edge list is the right graph, so any failure is in the Lean source — syntax, a tactic, an instance — and not in the figures or the structure.

Where the harness and the proof file divide the work

The two artifacts are not duplicates, and for this article the division is sharper than for its companions.

The harness enumerates. It builds the skeleton, finds the cut sets, perceives the rings, walks all 648 cut-set-and-ordering pairs, computes the patterns and the orbits, and does all of it a second time by different algorithms before cross-checking 74 macros against the article's own generator. None of that is in the Lean file.

The proof file certifies structure. It carries the graph and the automorphism and lets the kernel check that the one really is an automorphism of the other — which the harness also checks, but by a search whose correctness is itself a program, where the kernel's check is a reduction. And it certifies that the enumeration's output coheres.

Neither reaches Theorem 7.1. The harness confirms the bound twice; the proof file confirms that the confirmation is self-consistent; and the article's own Target 14.1 remains exactly where Article 87 left it, which its conclusion already says in as many words.