Emergent Assembly Chirality and Flip Kinematics

Euclidean Stabilizers of the Frame Classes, Enantiotopic Faces of the C_{2h} Core, and Initiator Capping of the Interior Lattice

Lean 4 Formal Proof Companion

Charles D. Schaper, Ph.D.
Annals of the Chrysene Formalism, Volume 1, Number 4
Article 91, November 2026

1 Formal scope and result

This is a **partial formal proof companion** to revised Article 91. Lean 4.24.0 successfully compiled **38 named theorems**. Two prove list identities for arbitrary lengths: applying the complement twice returns the word, and the alignment field is invariant under complement. The remaining theorems are exact finite certificates for the specified coordinates and reduced models. They use kernel-checked `decide`, not `native_decide`, and contain no `sorry`, `admit`, custom axiom or unsafe declaration.

A fresh `#print axioms` audit reports only Lean's standard propositional extensionality axiom for the two general list identities. The thirty-six finite certificates have no axiom dependencies. This is a claim about these theorem declarations, not a formalization of all analytical or physical claims in the manuscript.

2 Definitions and coverage

Words.
Boolean lists encode site words. `agr` takes adjacent exclusive differences, `flip` complements each bit and `words` generates every word of a fixed length. Reversal equivariance is checked on every length-six word; the general reversal theorem is proved in the manuscript, not formalized here.

Counts.
The file proves 32 length-five fields and eight palindromes, 64 length-eleven palindromes, pair counts 12, 992 and 16,256 at the distinguished lengths, and the integer equality giving the $l = 54$ fraction 2^{-26}. The general orbit-counting formula is not proved in Lean.

Geometry.
The eighteen centered pairs encode $(u/4, \sqrt{3}v/4)$. Their moments give $I_{xx} = 354/16$, $I_{yy} = 1014/16$ and $I_{xy} = -182\sqrt{3}/16$. The carbon half-turn, four equal-radius candidates and failure of every candidate reflection are checked. Rational images are represented by integer numerators and denominators, avoiding floating-point geometry. The continuous-to-finite completeness reduction is explained in the verification companion and manuscript; it is not itself formalized in Lean.

Core operations.
The four diagonal operations (a, a, b) are represented by sign pairs. No proper element among them exchanges the faces. The assertion that these are the complete $O(3)$ stabilizer uses the manuscript's geometric proof.

Sign configurations.
Boolean signs are grouped by the two-element locant orbits. The half-turn swaps each pair, and sign reversal models the horizontal reflection. Stabilizer counts, locked energy gaps and all partner energy-exchange identities for D and T are checked.

Thermal counterexample.
The anti pair $(-,+)$ and its half-turn have registry-zero energy -2ϵ and registry-one energy 2ϵ under correlated sampling. Four exact theorems certify the failure of marginal symmetry alone. The general weighted partition-sum reindexing proof remains in the manuscript.

Cap and supply.
The asymmetric cap locant image, failure of half-turn invariance and cleared-denominator resource equalities are certified. The full conditional geometric chirality theorem is not claimed as a Lean result.

3 Reproduction and trust boundary

Use the supplied `lean-toolchain`, or install the official Lean 4.24.0 release. Run

```
lean FormalProof_A91.lean
lean Audit_A91.lean
```

The second file repeats the exact proof source and prints the axiom dependencies of each named theorem. Both commands completed with exit code zero in the release check. The accompanying JSON and log record the compiler version, source hash and declarations checked. Only `Std` is imported; no Mathlib installation is required.

The standalone LaTeX source extracts the identical Lean source through `filecontents*`. Typesetting alone does not compile Lean. The included source is sufficient to reproduce the formal declarations separately from Overleaf.

4 Concordance with the manuscript

The certificates support the revised conditional realization framework. They do not assume that a bare C_{2h} core has two physically distinct half-turn states, nor infer a chemical realization from a word orbit. The manuscript's faithful, chirality-graded decorated realization hypothesis remains an explicit hypothesis. The thermal model requires a second-partner invariant joint weight. The code and formal proof record the correlated counterexample so that the earlier overstatement cannot be restored unnoticed.

5 Complete Lean source

FormalProof_A91.lean

```
import Std

/- Article 91: finite certificates for the revised model.
   Lean 4.24.0, core library only. No molecular-realization claim. -/
set_option maxRecDepth 100000
set_option maxHeartbeats 0
namespace Article91

def flip (w : List Bool) : List Bool := w.map Bool.not
def agr : List Bool -> List Bool
  | a :: b :: rest => (a != b) :: agr (b :: rest)
  | _ => []
def words : Nat -> List (List Bool)
  | 0 => [[]]
  | n+1 => (words n).map (fun w => false :: w) ++ (words n).map (fun w => true :: w)
def pal (w : List Bool) : Bool := w == w.reverse
def palCount (n : Nat) : Nat := ((words n).filter pal).length

theorem flip_twice (w : List Bool) : flip (flip w) = w := by
  induction w with
  | nil => rfl
  | cons a w ih => cases a <;> simp_all [flip]

theorem agr_flip (w : List Bool) : agr (flip w) = agr w := by
  induction w using agr.induct with
  | case1 a b rest ih => cases a <;> cases b <;> simp_all [agr, flip]
  | case2 w h => cases w with
    | nil => rfl
    | cons a rest => cases rest with
      | nil => rfl
      | cons b tail => exact False.elim (h a b tail rfl)

def sixEquivariance : Bool := (words 6).all fun w =>
  agr w.reverse == (agr w).reverse &&
  agr (flip w.reverse) == (agr w).reverse

theorem six_equivariance : sixEquivariance = true := by decide

theorem six_fields : (words 5).length = 32 := by decide
theorem six_palindromes : palCount 5 = 8 := by decide
theorem twelve_palindromes : palCount 11 = 64 := by decide
theorem six_pair_count : (2^5 - 2^3)/2 = 12 := by decide
theorem twelve_pair_count : (2^11 - 2^6)/2 = 992 := by decide
theorem sixteen_pair_count : (2^15 - 2^8)/2 = 16256 := by decide
theorem fiftyfour_fraction : (2^27 : Nat) * 2^26 = 2^53 := by decide

-- Centered coordinates: Euclidean point (u/4, sqrt(3)*v/4).
def points : List (Prod Int Int) :=
 [(-7,5),(-5,7),(-1,7),(1,5),(5,1),(7,-1),
  (7,-5),(5,-7),(1,-7),(-1,-5),(-5,-1),(-7,1),
  (-5,3),(-1,-1),(-1,3),(1,-3),(1,1),(5,-3)]
def norm (p : Prod Int Int) : Int := p.1*p.1 + 3*p.2*p.2

theorem eighteen_points : points.length = 18 := by decide
theorem centered_x : (points.map Prod.fst).sum = 0 := by decide
```

```
theorem centered_y : (points.map Prod.snd).sum = 0 := by decide
theorem moment_xx : (points.map fun p => p.1*p.1).sum = 354 := by decide
theorem moment_yy : (points.map fun p => 3*p.2*p.2).sum = 1014 := by decide
theorem moment_xy : (points.map fun p => p.1*p.2).sum = -182 := by decide
theorem carbon_halfturn : (points.all fun p => points.contains (-p.1,-p.2)) = true := by
    decide
theorem equal_radius_candidates :
 (points.filter fun q => norm q == norm (-7,1)).length = 4 := by decide

-- Integer numerator and denominator for a reflection mapping p to q.
def imageNumerator (v q : Prod Int Int) : Prod (Prod Int Int) Int :=
 let p : Prod Int Int := (-7,1)
 let a := (p.1+q.1,p.2+q.2)
 if a == (0,0) then
   let d := norm p
   let t := p.1*v.1+3*p.2*v.2
   ((d*v.1-2*t*p.1,d*v.2-2*t*p.2),d)
 else
   let d := norm a
   let t := a.1*v.1+3*a.2*v.2
   ((2*t*a.1-d*v.1,2*t*a.2-d*v.2),d)
def imageInSet (v q : Prod Int Int) : Bool :=
 let (num,d) := imageNumerator v q
 points.any fun w => num.1 == d*w.1 && num.2 == d*w.2

theorem all_candidate_reflections_fail :
 ((points.filter fun q => norm q == norm (-7,1)).all fun q =>
    !(points.all fun v => imageInSet v q)) = true := by decide

-- The four signs of the ideal C2h operations (a,a,b).
def coreOps : List (Prod Int Int) := [(1,1),(-1,1),(1,-1),(-1,-1)]
theorem no_proper_facial_core_op :
 (coreOps.all fun op => !(op.2 == -1 && op.1*op.1*op.2 == 1)) = true := by decide

-- Signs are encoded in pairs corresponding to the iota orbits.
def swap : List Bool -> List Bool
 | a :: b :: rest => b :: a :: swap rest
 | other => other
def sign (b : Bool) : Int := if b then 1 else -1
def dotSigns (a b : List Bool) : Int :=
 ((a.zip b).map fun p => sign p.1 * sign p.2).sum
def energy (a b : List Bool) (eta : Bool) : Int :=
 -(dotSigns a (if eta then swap b else b))
def antiCount : List Bool -> Nat
 | a :: b :: rest => (if a == b then 0 else 1) + antiCount rest
 | _ => 0
def c2count (n : Nat) := ((words n).filter fun w => swap w == w).length
def invcount (n : Nat) := ((words n).filter fun w => flip (swap w) == w).length

theorem D_c2 : c2count 2 = 2 := by decide
theorem D_ci : invcount 2 = 2 := by decide
theorem T_c2 : c2count 4 = 4 := by decide
theorem T_ci : invcount 4 = 4 := by decide
theorem T_mixed : 2^4 - c2count 4 - invcount 4 = 8 := by decide

def gapCheck (n : Nat) : Bool := (words n).all fun w =>
 energy w w true - energy w w false == 4 * (antiCount w : Int)
theorem D_gap : gapCheck 2 = true := by decide
theorem T_gap : gapCheck 4 = true := by decide
```

```
def exchangeCheck (n : Nat) : Bool := (words n).all fun a =>
 (words n).all fun b =>
 energy a (swap b) false == energy a b true &&
 energy a (swap b) true == energy a b false

theorem D_energy_exchange : exchangeCheck 2 = true := by decide
theorem T_energy_exchange : exchangeCheck 4 = true := by decide

def anti : List Bool := [false,true]
theorem correlated_anti_E0 : energy anti anti false = -2 := by decide
theorem correlated_anti_E1 : energy anti anti true = 2 := by decide
theorem correlated_swapped_E0 : energy (swap anti) (swap anti) false = -2 := by decide
theorem correlated_swapped_E1 : energy (swap anti) (swap anti) true = 2 := by decide

-- Locant masks under iota (3,9) and (6,12).
def iota (n : Nat) : Nat :=
 if n == 3 then 9 else if n == 9 then 3 else
 if n == 6 then 12 else if n == 12 then 6 else n
def cap := [3,6,12]
theorem cap_image : cap.map iota = [9,12,6] := by decide
theorem cap_not_fixed : !(cap.all fun x => cap.contains (iota x)) = true := by decide
theorem cap_budget : (208 : Int) - 10 = 198 := by decide
theorem nine_cap_threshold : (208 : Int) * 10 - 208 = 9 * 208 := by decide
theorem eighteen_cap_threshold : (208 : Int) * 19 - 208 = 18 * 208 := by decide

end Article91
```