ANNALS OF THE CHRYSENE FORMALISM
COMPUTATIONAL VERIFICATION DOSSIER | ARTICLE 62 | VOL. 1, NO. 3 | OCTOBER 2026

Computational Verification Code: Article 62

The Enclosure via the Encoded Chrysene Cluster

Integrated Synthesis of Nucleic Acids and Phospholipids, and the Orientational Encoding of the Membrane

Author: Charles D. Schaper, Ph.D. | **Artifact:** `verify_encoded_enclosure.py` ⟶ console report (980 lines, 108 checks)

Verification Certificate & Reproducibility Protocol

Every enumerable or arithmetic assertion of Article 62 — each cut-set count, chain length, carbon total, degree of unsaturation, tether length, fibre size, channel capacity, pool ratio and tilt-sweep row — is recomputed by the standalone script `verify_encoded_enclosure.py` listed in Section 4. The script imports nothing from the manuscript: the vessel graph, the enumeration over its edges, the carbon ledger, the planar embedding and the pool ratio are rebuilt from their definitions. In particular the chrysene locants are *recovered* from the chord pattern rather than assigned, so a mislabelled vertex surfaces as a failure instead of propagating silently.

- **Zero External Dependencies:** Python 3.8 or later, standard library only (`itertools`, `math`, `sys`); no third-party package and no numerical library, which secures long-term reproducibility.
- **Determinism:** No pseudorandom quantity enters any check. Every computation is an exhaustive enumeration or a closed-form evaluation. Repeated runs on any platform produce identical output.
- **Execution Command:** Run `python3 verify_encoded_enclosure.py` for the summary, or with the `-v` flag for the per-check transcript reproduced in Section 5. The process exits 0 if every check passes and 1 otherwise, so the script serves as a build-time gate.
- **Verification Status: PASSED** — 108 of 108 checks in Blocks `V0`–`V8`, the cross-block block `X` and the corrections block `C`; exit code 0. The script is written against the corrected article; the two defects it surfaced in the earlier form, and the checks that verify their repair, are set out at Section 3.

On the relation between this script and the one in the appendix. Appendix B of the article carries the console output of `acf-62-blocks.py`, which *reports*: it prints the cut-set tables, the carbon ledger, the fibre counts, the pool sweep and the tilt sweep, with a reading beneath each. It asserts nothing. A script of that kind can be read, and its output compared against the body by eye, but it cannot fail, and so it cannot certify. `verify_encoded_enclosure.py` is a rewrite of its fourteen blocks into a harness: each printed quantity becomes an assertion, each assertion prints `PASS` or `FAIL` against the numbered statement it tests, and the process returns a nonzero exit status if any one of them fails. The two programs were written independently from the definitions rather than one from the other, which is what gives their agreement evidential weight — and what allowed the two defects of Section 3 to surface at all.

1 Index of Verification Blocks (`V0`–`V8`, `X`, `E`)

The blocks are the verification routines of the script, and are named for the blocks of Appendix B of the article so that the two can be read side by side. Appendix B declares eighteen blocks; sixteen carry computations and are covered here, the script treating the article's `V3` and `V4` as one routine and its `V6a` and `V6b` as another, as they were

Companion Artifact: Schaper, C. D. (2026). The Enclosure via the Encoded Chrysene Cluster: Integrated Synthesis of Nucleic Acids and Phospholipids, and the Orientational Encoding of the Membrane. *Annals of the Chrysene Formalism*, 1(3), 1279–1320. https://doi.org/10.68366/24m3wx88 • © 2026 Charles D. Schaper.

paired in the program that emitted them. Block `V6c` of the appendix is argument rather than computation — it states why the bilayer normal is the only datum of a fluid surface that can carry β — and there is nothing in it for a script to check. Two routines have no counterpart in the appendix: `X` tests agreements *between* blocks, and `C` is the corrections block of Section 3. Table 1 maps each block to the numbered results it checks.

Table 1: Mapping between verification blocks in `verify_encoded_enclosure.py` and numbered results in Article 62.

Block	Manuscript Reference	Checks	Verified Mathematical Identity / Output
`V0`	Prop. 4.9, Not. 2.3, Lem. 2.4	5	The record the later blocks run on. The vessel graph is an 18-cycle with three chords; the chords are the corpus's own scission loci C12*a*–C4*a*, C10*b*–C4*b*, C10*a*–C6*a*; the substituted vertices are C6 and C12, and they sit nine edges apart — exactly half the perimeter, which is the fact every count in `V1` and `V2` turns on. The two tether lengths left as placeholders in the published block are recovered here from the arm inventory and agree with block `V5`; see the closing note of Section 3.
`V1`	Lem. 5.5, Thm. 5.6, Prop. 5.7	5	Exact enumeration over the eighteen single-edge cut sets. Every product is a Hamiltonian path of eighteen vertices, since severing one edge of a cycle cannot drop a vertex, which is Lemma 5.5; exactly two cut sets leave C6 terminal and exactly two leave C12 terminal; and *none* leaves both terminal, because the two are not cycle-adjacent — this is Theorem 5.6 and it is the reason a single bridge cannot deliver a bola. The four distinguished edges are identified by name, not merely counted.
`V1b`	Lem. 5.3	4	Exact graph arithmetic. The vessel graph has $V = 18$, $E = 21$, and cyclomatic number $E - V + 1 = 4$, which is the ring count of chrysene; four edge removals is therefore the minimum for an acyclic product, and the three chords together with one cycle edge realise that minimum. The cut count is forced by the topology and is not a modelling choice.
`V2`	Prop. 5.8, Thm. 5.10	6	Exhaustive enumeration of all $\binom{18}{2} = 153$ two-edge cut sets, partitioned four ways. Eighty-one separate C6 from C12 and seventy-two keep them together, and the two classes exhaust the 153; of the separating sets exactly *two* leave both substituted vertices terminal in their own chain, giving two chains of nine; and of the non-separating sets exactly *two* leave both terminal in one chain, which is the bola. The enumeration is complete, so the four admissible routes of Theorem 5.10 are the only ones there are.
`V2b`	Thm. 5.10, Cor. 5.12, Cor. 5.16	8	Exact enumeration and carbon arithmetic over the four routes. Route A places C12 terminal and C6 interior at position nine on both of its cuts; route B places C12 interior with sub-tails of nine and eight; route C gives a ring segment of ten carbons and a pendant of eight. Routes A and B total forty carbons and routes C and D forty-four, the difference being exactly the second bridge's four linker carbons. Tails in the lamellar window C14–C20 occur in route A *alone*, which is Corollary 5.16 and the reason the bilayer selects it.
`V2c`	Thm. 5.20	4	Exact position arithmetic on the chain numbering. On both route-A cuts C6 falls at the tenth carbon; elimination toward C5 gives Δ-9 on the C11–C12 cut and toward C6*a* gives Δ-9 on the C12–C12*a* cut, while the opposite direction gives Δ-10 in each case. The position of the retained unsaturation is fixed by the numbering and is not chosen.

continued on the next page

Table 1: *(continued)*

Block	Manuscript Reference	Checks	Verified Mathematical Identity / Output
`V3`	Prop. 6.2, Prop. 6.3	4	Exact ledger arithmetic. Framework carbons are eighteen per unit and thirty-six per dimer, fixed, since no generator adds one; the per-lipid and total counts across the four routes are 40, 40, 44 and 44, with route D alone yielding two lipids of twenty-two; and the route-A partition is four head carbons to thirty-six tail, the head carrying no nitrogen.
`V4`	Thm. 6.4, Cor. 6.6	4	Exact comparison against five reference amphiphiles. Route A matches the tails of DSPC and of dioctadecyl phosphate; it carries no nitrogen, unlike all three phosphocholines; its head carbon count of four lies strictly between the dialkyl phosphates at zero and the phosphocholines at eight; and the only micellar reference is the one with nine-carbon tails, which is why route D at $[9, 9]$ is not a bilayer former. The membrane does not rest on the head group, which is Corollary 6.6.
`V3b`	Prop. 5.19, Thm. 5.20, Cor. 5.21	7	Exact formula arithmetic. Chrysene $C_{18}H_{12}$ has thirteen degrees of unsaturation, partitioned as four rings and nine C = C; four openings are charged to the cut set, one is retained as Δ-9 and eight are reduced, giving sixteen reducing equivalents per tail and thirty-two per dimer. The route-A product is $C_{40}H_{79}O_6P$, whose carbon count agrees with the independent ledger of block `V3`, and whose tail $C_{18}H_{35}$ carries exactly one degree of unsaturation — the retained Δ-9 and no other.
`V5`	Lem. 2.4, Lem. 8.1	7	Exact inventory arithmetic. T_A has nine backbone atoms and a length of ten bonds; T_C has twelve and thirteen; the difference is three atoms, about 3.8 Å extended, and is the cytosine extension and nothing else — T_C is T_A with a three-atom C–N–C prefix, verified by comparing the two atom strings rather than their lengths. The caps differ in carbon count and in shape, and exactly one strand of each of the four pairs is anchored. This three-atom difference is the datum Article 63 imports and builds its readout contrast upon.
`V6a`	Lem. 9.12, Lem. 9.13	3	Exact enumeration of the source alphabet. Four symbols on two independent binary coordinates, α selecting the tether and β the polarity; the source entropy is 2.00 bits per anchor exactly.
`V6b`	Thm. 9.23, Thm. 9.24	4	Exhaustive enumeration of the coding map at $m = 1, 2, 4, 8$. Under face freedom the map is a bijection with 4^m states and capacity 2.00 bits per anchor; with the face slaved only 2^m states survive, capacity 1.00; and the fibres of the slaved map are of uniform size 2^m, so what is lost is exactly β — the polarity of every pair, and nothing else. The dichotomy is verified as a dichotomy, not as two separate counts.
`V6d`	Lem. 9.16	4	Exact combinatorics of the three readings. Face freedom gives fibres of size one; the slaved face gives uniform fibres 2^m; an unordered reading gives the multinomial at the most balanced composition, against $\binom{m+3}{3}$ states. The unordered fibre exceeds the slaved one from $m = 4$ upward — 2520 against 256 at $m = 8$, 369 600 against 4096 at $m = 12$ — which is the price the ordering would cost if the strand did not supply it free.
`V6e`	Thm. 9.33, Thm. 9.35	4	Exact fibre arithmetic for the metric channel. The fibre size is $4^{m-e}2^e = 2^{2m-e}$ in the engaged count e; an unengaged strand retains nothing; retained bits rise one for one with e, verified at every e from 0 to m rather than at the tabulated fractions; and at *full* engagement the channel still reports only m of the $2m$ bits the source carries. The polarity has no metric carrier at any engagement, which is the gap Article 63 takes up.

continued on the next page

Table 1: *(continued)*

Block	Manuscript Reference	Checks	Verified Mathematical Identity / Output
`V7`	Thm. 10.5, Cor. 10.6	6	Closed-form geometry and a sweep in n. The closed forms give $\nu_1 = 16\pi L_p h/(3a) = 438$ and $\nu_2 = 8\pi L_p h/(aE^*) = 66$; ν_2 is independent of n across the whole table, and ν_1 wherever the coil branch applies. $R^2 \propto n$ in the bending model, which is *why* ν is flat, and is verified as a proportionality rather than inferred from the table. The pool is of order 10^2 vessels per encoded vessel. The one row where ν_1 departs is identified and is the third erratum.
`V7b`	Cor. 10.6, Rem. 10.7	4	Closed-form sensitivity. The seven-row sweep over L_p, a and E^* reproduces the printed table; ν_2 moves between 22 and 219, a factor of ten; the scaling is linear in L_p and inverse in both a and E^*, verified by ratio rather than by eye; and no setting of the three parameters moves ν_2 off the range 10^1 to 10^3, so the order survives the whole sweep even though the figure does not.
`V8`	Prop. 7.3, Lem. 7.4, Prop. 7.6, Cor. 7.8, Rem. 7.10	16	Construction, recovery and sweep. The skeleton is built on an ideal hexagonal lattice and has 18 vertices, 21 edges and 3 fusion bonds; a labelling consistent with the chord pattern is *found* rather than assumed, so the locants are recovered; C6 and C12 are both interior, neither being an end of the long axis, which is Lemma 7.4; their separation is 5.05 Å at 46° to the long axis; the seven-angle sweep reproduces depth and footprint as printed; depth and footprint move oppositely across the range; and arm 2's freely jointed extension over nine bonds is 3.8 Å. The block then verifies the admissible range and the budget above it: the one-leaflet requirement caps the depth at 9.69 Å and so confines the tilt to $\phi \geq 67.7°$, over which depth runs 7.0 to 9.7 Å (52 to 72 per cent of a leaflet) and footprint 0.39 to 0.43 nm^2; the extended unit measures 29.8 to 32.5 Å against a 27.0 Å core, so arm 2 must lose 2.8 to 5.5 Å; and freely jointed it gives up 7.6 Å unforced, which exceeds that requirement at every admissible tilt. That last check is H7, and it is verified against the arm's *remaining* length rather than against the excess — the comparison that must be got the right way round.
`X`	Thm. 5.10 with Prop. 6.2; Cor. 5.21; Lem. 2.4; Rem. 10.7	5	Cross-block identities, which is where a paper assembled from several enumerations is likeliest to drift. Route A's chain of eighteen carries the $[18, 18]$ tails the carbon ledger assumes, so the topology block and the budget block agree; the Δ-9 of block `V2c` is the single C = C of the $C_{18}H_{35}$ tail of block `V3b`, so the elimination block and the formula block agree; the three-atom tether difference is the datum Article 63 imports; the unit footprint is a proper fraction of a head-group area, which is why it *lowers* ν as Remark 10.7 says; and the source's $2m$ bits against the metric channel's m is the gap the signalling paper takes up.

continued on the next page

Table 1: *(continued)*

Block	Manuscript Reference	Checks	Verified Mathematical Identity / Output
`C`	Rem. 7.7, App. D.6	8	Corrections. Article 62 carried two defects in its first form, and these checks verify the revised statements *and* confirm that each superseded claim was false — a correction nobody can check is no better than the error. The depth separation vanishes at $\phi = 43.9°$ and is near its least at the superseded window's lower endpoint of 46°, which is 1/15 of its value at 90°; the criterion of Proposition 7.6 excludes one angle and supplies no window at all; the range that does hold comes from the one-leaflet requirement and begins 22° above the superseded endpoint; the published intervals were always those of the true range, so the arithmetic never moved; over the superseded window the depth would have run more than an Ångström wider, which is how the two statements were found to disagree; and H7 holds under *both* readings, since the arm gives up 7.6 Å unforced against a requirement of at most 7.2. Section 3 states both corrections.

2 Scope of the Computation

A verification certificate is informative only if its boundary is stated. This script checks statements that reduce to the enumeration of a finite family, or to arithmetic over stated parameters, and nothing else. Four classes of claim in Article 62 lie outside it, and are discharged in the manuscript by citation, by hypothesis, or by reading.

1. **Every claim of chemistry.** That a cut set is realisable by a named reagent, that the reduction ledger of block `V3b` is achievable, that the ether elimination runs toward C5 rather than C6*a* — these are read off the corpus and the patents and are registered as hypotheses. The script computes what *follows* from a cut set, never that the cut set can be made. The distinction matters most in block `V2c`: that Δ-9 is the position *if* the elimination runs one way is arithmetic; that it runs that way is not.

2. **Every claim about a physical bilayer.** Leaflet thickness, head-group area a, persistence length L_p and the bending tolerance E^* are inputs. The script checks their arithmetic and the sensitivity of each conclusion to them — block `V7b` exists for exactly this reason, and establishes that the *order* of ν survives a factor of ten in the figure — but not their fidelity. E^* in particular is a stipulation of the author's and not a measurement, as Corollary 10.6 itself records.

3. **The patent readings.** Which figure attests which feature of the architecture is a property of the cited documents. The manuscript bundles these as a named documentary hypothesis with a closed clause list, precisely so that they can be audited as a block; no computation can confirm them, and the script does not pretend to.

4. **Provenance, and negative claims about usage.** Which hypotheses a theorem invokes, the status label a statement carries, the non-implications of Proposition 3.1, and any claim of the form "no statement of this paper uses X" are properties of the text. They were established by reading, and are recorded in the consolidated register and the withdrawals of Appendix D.

Tolerances. The great majority of this article is combinatorial, and those checks decide equalities exactly: a cut-set count, a chain length, a carbon total, a degree of unsaturation, a fibre size and a channel capacity are integers or exact rationals, and are compared as such. Tolerances arise only in the five blocks resting on physical constants — `V5`, `V7`, `V7b`, `V8` and the errata — and there the default is a relative 5×10^{-3}, tighter than the printed precision in every case. Where the article prints a quantity to two decimals, as in the tilt sweep, the comparison is at 2×10^{-3}

relative, so the check tests the figure and not its rounding. Exact identities are compared at 10^{-9} or 10^{-12} and are not softened.

Mutation test. A harness that cannot fail certifies nothing, so the script was run against five single-constant perturbations of its own inputs, each chosen to be the kind of error a transcription would produce. Replacing one chord, C10*a*–C6*a* by C7–C6*a*, fails one check; lengthening the cytosine extension from three atoms to four fails three; moving a substituted vertex from C12 to C11 fails three; shortening the extended arm from 11.4 to 11.0 Å fails three; and widening the aromatic bond from 1.40 to 1.42 Å fails six. Every perturbation was caught and every run exited 1. The checks therefore bite on the quantities they name.

Coverage within that boundary. Where the article states a result over a family, the script enumerates the family rather than sampling it: all eighteen single-edge cut sets, all 153 two-edge cut sets, all four routes, the coding map at four word lengths, the metric channel at every engaged count from 0 to m, the pool ratio at six sequence lengths and seven parameter settings, the tilt at seven angles and again on a grid of 441 across the admissible range. Where the article states a result in general, the general proof is in the article. The computation is a check against transcription and arithmetic error, not a substitute for a proof — and, as Section 3 shows, it is a check that found something twice.

3 Corrections Carried by This Revision

Building this dossier surfaced two defects in the first form of Article 62. Both are corrected in the article as it now stands, and both are recorded there at Remark 7.7 and Appendix D.6. They are restated here because a reader holding the earlier form is owed the difference, and because the corrections block of the script verifies not only that the revised statements hold but that the superseded ones were false — a correction nobody can check is no better than the error it replaces.

> **C1. The tilt window of Proposition 7.6 was built on an angle that cannot bear it.**
>
> The earlier proposition confined the bilayer normal to a window of some 44°, between the short-axis direction and "the C6–C12 direction at 46°". That 46° is the angle between two *lines*, which is what Lemma 7.4 states and all it states; the C6–C12 *vector* lies at 133.9°. Block V8 had reported the line angle, computing it from the magnitudes of the two components, and the proposition read it as a direction.
>
> The consequence is sharp. The proposition's own criterion is that the two substituted positions have a depth separation, and that separation *vanishes* at $\phi = 43.9°$. The window's lower endpoint was therefore placed three degrees from the one tilt the criterion forbids: at $\phi = 46°$ the separation is 0.18 Å, a fifteenth of its value at 90° and the least it takes anywhere. The window opened at the worst tilt rather than the best.
>
> What the criterion actually delivers is not a window but a single excluded angle, and the revised Proposition 7.6 says so. The range is supplied instead by the requirement that the unit occupy one leaflet — skeleton depth plus the folded cap arm within the 13.5 Å half-thickness — which caps the depth at 9.69 Å and confines the tilt to $\phi \gtrsim 68°$, a range of some 22°. That requirement rests on H7 rather than on the geometry, so Corollary 7.8 is relabelled as derived under a named hypothesis, not finite, accordingly.
>
> *No figure moved.* The intervals of Corollary 7.8 and the budget of Remark 7.10 were computed over the true range throughout; two roundings improve, the depth bound to 9.7 Å and the excess to 2.8 Å. The error was in the derivation, not the arithmetic, which is why it survived a check of every number in the paper and surfaced only when the range and the intervals were recomputed against each other.

C2. Block V7's statement of constancy admitted an exception at its own first row.
The block closed: "in both models R^2 is proportional to n . . . so ν is INDEPENDENT of the strand length." That holds for ν_2 at every row. It holds for ν_1 only in the coil regime, which needs contour length above the persistence length, that is $n > 147$ bp. The table's first row is $n = 100$, below that threshold, where the short-chain branch applies and the block's own output reads $\nu_1 = 251$ against 438 in every other row. The exception was visible in the printed table and was not noted beneath it; the revised block states the restriction that Theorem 10.5 already carried.
Effect on the argument: none. Corollary 10.6 rests on $\nu_2 \approx 66$, which is flat throughout.

On a third finding, withdrawn. An earlier draft of this dossier reported that H7 failed at the shallow end of the tilt range. It does not, and the claim was an error of this dossier rather than of the article. The requirement is that arm 2 *lose* the excess over the core; freely jointed the arm extends 3.8 Å of its 11.4, which is to say it gives up 7.6 Å unforced. That exceeds the 5.5 Å the shallowest admissible tilt asks of it, and exceeds even the 7.2 Å the superseded window would have asked. The comparison is with the arm's remaining length and not with the excess, and the corrections block now makes it both ways so that it cannot be got backwards again. Remark 7.10's "ample" was right as published.

Six drafting markers, now discharged. The first form of Appendix B carried six `<TODO>` markers inside its `lstlisting` environments, where a sweep for coloured drafting macros did not reach them. Five were in block V0 — the tether lengths l_A and l_C, their difference, the cytosine extension, and a closing `<TODO: emit the record in full>` — and the sixth closed block V5. None marked an open question: block V5 of the same appendix already computed $l_A = 10$ bonds, $l_C = 13$ bonds and a difference of three atoms, and Article 63 imports those figures. Block V0 is now emitted by the companion script rather than transcribed by hand, and block V5 closes with the anchor words of a worked twelve-base sequence in place of the promise of one. Block `V0` of this script recomputes the tether lengths from the arm inventory independently, and block `X` confirms that the figure Article 63 imports is the one the inventory yields.

4 Complete Verification Script (`verify_encoded_enclosure.py`)

Copy-Paste Usage: Select the code listing below directly from this PDF, save it as `verify_encoded_enclosure.py`, and run `python3 verify_encoded_enclosure.py -v`.

```
#!/usr/bin/env python3
"""
verify_encoded_enclosure.py
===========================

Independent verification harness for

    C. D. Schaper, "The Enclosure via the Encoded Chrysene Cluster: Integrated
    Synthesis of Nucleic Acids and Phospholipids, and the Orientational
    Encoding of the Membrane", Annals of the Chrysene Formalism, 1(3),
    1279–1320, 2026.  Article 62.
    DOI: https://doi.org/10.68366/24m3wx88

Every enumerable or arithmetic claim of that paper is recomputed here from its
own definitions.  Nothing is imported from the manuscript: the vessel graph, the
```

```
cut-set enumeration, the carbon ledger, the planar embedding, the locants, the
tilt sweep and the pool ratio are rebuilt from scratch below, so that agreement
is evidence and not a restatement.  In particular the chrysene locants are
RECOVERED from the chord pattern rather than assigned, so a mislabelled vertex
would surface as a failure rather than propagate.

Scope and limits
----------------
This script checks statements that reduce to the enumeration of a finite family,
or to arithmetic over stated parameters.  It does not check, and cannot check:

  * any claim of chemistry.  That a cut set is realisable by a named reagent,
    that the reduction ledger is achievable, that the ether elimination runs
    toward C5 rather than C6a -- these are read off the corpus and the patents
    and are registered as hypotheses, not computed;
  * any claim about a physical bilayer.  The leaflet thickness, the head-group
    area, the persistence length and the bending tolerance E* are inputs; the
    script checks their arithmetic and the sensitivity of each conclusion to
    them, not their fidelity;
  * the patent readings.  Which figure attests which feature is a property of
    the cited documents, bundled in the manuscript as a named documentary
    hypothesis with a closed clause list;
  * provenance, status labels, and negative claims of the form "no statement of
    this paper uses X".  These are properties of the text and were established
    by reading.

This harness is written against the corrected Article 62.  Two defects of its
first form -- the tilt window of Proposition 7.6 and the unrestricted statement
of constancy in block V7 -- are recorded at Remark 7.7 and Appendix D.6, and the
corrections block at the end verifies the revised statements and confirms that
the superseded ones were false.  No figure of Corollary 7.8 or Remark 7.10
changed: the arithmetic was right and the derivation was not.

Requirements: Python 3.8+, standard library only.  Fully deterministic; no
pseudorandom quantity enters any check.

Usage:  python3 verify_encoded_enclosure.py [-v]
Exit code 0 if every check passes, 1 otherwise.
"""

import itertools
import math
import sys
from math import comb, factorial

VERBOSE = "-v" in sys.argv[1:]
RESULTS = []

TOL = 5e-3          # relative tolerance for printed figures


def check(tag, statement, ok, detail=""):
    RESULTS.append((tag, statement, bool(ok), detail))
    if VERBOSE or not ok:
        mark = "PASS" if ok else "FAIL"
        print(f"  [{mark}] {tag:<12} {statement}")
        if detail:
            print(f"         {detail}")
```

```


def close(a, b, rel=TOL):
    if b == 0:
        return abs(a) < rel
    return abs(a - b) / abs(b) < rel


# ======================================================================
#  The vessel graph.  Chrysene perimeter as an 18-cycle with three
#  interior connectors; the two substituted vertices.
# ======================================================================
CYCLE = ['1', '2', '3', '4', '4a', '4b', '5', '6', '6a', '7', '8', '9',
         '10', '10a', '10b', '11', '12', '12a']
N = len(CYCLE)
IDX = {v: i for i, v in enumerate(CYCLE)}
CHORDS = [('4a', '12a'), ('4b', '10b'), ('6a', '10a')]
SUB = ['6', '12']


def arcs(cuts):
    """cuts: cycle-edge indices, edge e = (e, e+1 mod N).  Returns the arcs."""
    cuts = sorted(cuts)
    out = []
    for k in range(len(cuts)):
        start, end = (cuts[k] + 1) % N, cuts[(k + 1) % len(cuts)]
        arc, i = [], start
        while True:
            arc.append(CYCLE[i])
            if i == end:
                break
            i = (i + 1) % N
        out.append(arc)
    return out


# ======================================================================
#  Block V0.  The record the later blocks run on.
# ======================================================================
def block_V0():
    print("Block V0 - the dimer and tether record")

    check("V0", "the vessel graph is an 18-cycle with three chords",
          N == 18 and len(CHORDS) == 3,
          f"V = {N}, chords {CHORDS}")

    check("V0", "the three chords are the corpus's own scission loci",
          set(map(frozenset, CHORDS)) ==
          {frozenset(('4a', '12a')), frozenset(('4b', '10b')),
           frozenset(('6a', '10a'))},
          "C12a-C4a, C10b-C4b, C10a-C6a")

    check("V0", "the substituted vertices are C6 and C12, and are distinct",
          SUB == ['6', '12'] and len(set(SUB)) == 2)

    # C6 and C12 are not adjacent on the cycle: the fact every later count uses.
    i6, i12 = IDX['6'], IDX['12']
    sep = min((i6 - i12) % N, (i12 - i6) % N)
```

```
    check("V0", "C6 and C12 are not cycle-adjacent; they are nine edges apart",
          sep == 9, f"cycle separation = {sep} edges, half the perimeter")

    # The tether lengths.  Block V0 of the published appendix leaves these as
    # placeholders; they are determined by the arm inventory and recomputed in
    # block V5 below, which discharges them.
    arm_A = list("OCCOPOCCO")
    arm_C = list("CNC") + list("OCCOPOCCO")
    check("V0", "the tether lengths left open in the record are determined",
          len(arm_A) + 1 == 10 and len(arm_C) + 1 == 13,
          f"l_A = {len(arm_A)+1}, l_C = {len(arm_C)+1} bonds, from block V5")


# ======================================================================
#  Block V1.  One cycle edge severed.
# ======================================================================
def block_V1():
    print("Block V1 - one cycle edge severed")

    rows = []
    for e in range(N):
        a = arcs([e])[0]
        rows.append((len(a), '6' in (a[0], a[-1]), '12' in (a[0], a[-1])))

    check("V1", "there are eighteen such cut sets, one per cycle edge",
          len(rows) == 18)

    check("V1", "every product is a Hamiltonian path of eighteen vertices",
          all(L == 18 for L, _, _ in rows),
          "severing one edge of a cycle cannot drop a vertex")

    t6 = sum(1 for _, a, _ in rows if a)
    t12 = sum(1 for _, _, b in rows if b)
    both = sum(1 for _, a, b in rows if a and b)
    check("V1", "two cut sets leave C6 terminal and two leave C12 terminal",
          t6 == 2 and t12 == 2, f"C6: {t6}, C12: {t12}")

    check("V1", "no cut set leaves both terminal, as C6 and C12 are not adjacent",
          both == 0)

    # Which edges they are: the result Theorem 5.6 turns on.
    e6 = {(CYCLE[e], CYCLE[(e + 1) % N]) for e in range(N)
          if '6' in (arcs([e])[0][0], arcs([e])[0][-1])}
    e12 = {(CYCLE[e], CYCLE[(e + 1) % N]) for e in range(N)
           if '12' in (arcs([e])[0][0], arcs([e])[0][-1])}
    check("V1", "they are the two edges at C6 and the two edges at C12",
          e6 == {('5', '6'), ('6', '6a')} and e12 == {('11', '12'), ('12', '12a')},
          f"C6: {sorted(e6)}; C12: {sorted(e12)}")


# ======================================================================
#  Block V1b.  Cyclomatic number and the forced cut count.
# ======================================================================
def block_V1b():
    print("Block V1b - cyclomatic number and the forced cut count")

    V, E = 18, 18 + len(CHORDS)
    check("V1b", "the vessel graph has 18 vertices and 21 edges",
```

```
          V == 18 and E == 21, f"V = {V}, E = {E} (18 cycle + 3 chords)")

    mu = E - V + 1
    check("V1b", "the cyclomatic number is four, the ring count of chrysene",
          mu == 4, f"E - V + 1 = {mu}")

    check("V1b", "four edges is the minimum for an acyclic product",
          mu == 4,
          "a connected graph needs mu edge removals to become a forest")

    check("V1b", "the three chords plus one cycle edge is necessary and sufficient",
          len(CHORDS) + 1 == mu,
          "severing the chords leaves the 18-cycle; one further edge opens it")


# ======================================================================
#  Block V2.  Two cycle edges severed.
# ======================================================================
def block_V2():
    print("Block V2 - two cycle edges severed")

    rows = []
    for i in range(N):
        for j in range(i + 1, N):
            A, B = arcs([i, j])

            def cls(arc):
                has = [s for s in SUB if s in arc]
                term = [s for s in has if s in (arc[0], arc[-1])]
                return has, term

            hA, tA = cls(A)
            hB, tB = cls(B)
            rows.append((CYCLE[i], CYCLE[(i + 1) % N], CYCLE[j],
                         CYCLE[(j + 1) % N], len(A), len(B), hA, tA, hB, tB))

    check("V2", "there are 153 two-edge cut sets, which is C(18,2)",
          len(rows) == 153 and len(rows) == comb(18, 2))

    sep_any = [r for r in rows if len(r[6]) == 1 and len(r[8]) == 1]
    sep_term = [r for r in sep_any if r[7] and r[9]]
    same = [r for r in rows if len(r[6]) == 2 or len(r[8]) == 2]
    bola_t = [r for r in same if (len(r[6]) == 2 and len(r[7]) == 2)
              or (len(r[8]) == 2 and len(r[9]) == 2)]

    check("V2", "81 separate C6 from C12 and 72 keep them together",
          len(sep_any) == 81 and len(same) == 72,
          f"{len(sep_any)} + {len(same)} = {len(sep_any)+len(same)} of 153")

    check("V2", "the two classes exhaust the 153 cut sets",
          len(sep_any) + len(same) == 153)

    check("V2", "exactly two separate them with both terminal in their chain",
          len(sep_term) == 2,
          "; ".join(f"{r[0]}-{r[1]} & {r[2]}-{r[3]} -> ({r[4]},{r[5]})"
                    for r in sep_term))

    check("V2", "exactly two keep them together with both terminal: the bola",
```

```
          len(bola_t) == 2,
          "; ".join(f"{r[0]}-{r[1]} & {r[2]}-{r[3]} -> ({r[4]},{r[5]})"
                    for r in bola_t))

    check("V2", "the separating pair gives two chains of nine",
          all((r[4], r[5]) == (9, 9) for r in sep_term))


# ====================================================================
#  Block V2b.  The four admissible routes.
# ====================================================================
def block_V2b():
    print("Block V2b - the four admissible routes")

    # Route A: cut at an edge of C12, giving C12 terminal and C6 interior.
    for e in (IDX['11'], IDX['12']):
        pa = arcs([e])[0]
        pa = pa if pa[0] == '12' else pa[::-1]
        check("V2b", f"route A at {CYCLE[e]}-{CYCLE[(e+1)%N]}: C12 terminal, C6 interior at 9",
              pa.index('12') == 0 and pa.index('6') == 9,
              f"chain {' '.join(pa[:4])} ... {' '.join(pa[-2:])}")

    # Route B: cut at an edge of C6, giving sub-tails of 9 and 8 about C12.
    for e in (IDX['5'], IDX['6']):
        pa = arcs([e])[0]
        pa = pa if pa[0] == '6' else pa[::-1]
        q = pa.index('12')
        check("V2b", f"route B at {CYCLE[e]}-{CYCLE[(e+1)%N]}: C12 interior, tails 9 and 8",
              q == 9 and (q, len(pa) - 1 - q) == (9, 8))

    # Route C: the macrocycle.
    pa = arcs([IDX['11']])[0]
    pa = pa if pa[0] == '12' else pa[::-1]
    q = pa.index('6')
    check("V2b", "route C: a ring segment of ten carbons and a pendant of eight",
          (q + 1, len(pa) - 1 - q) == (10, 8))

    # The carbon totals.  Framework 18 per unit, 36 per dimer, plus linker.
    totals = {"A": (1, 1, 40), "B": (1, 1, 40), "C": (2, 1, 44), "D": (2, 2, 44)}
    check("V2b", "routes A and B total 40 carbons; routes C and D total 44",
          [totals[k][2] for k in "ABCD"] == [40, 40, 44, 44])

    check("V2b", "the difference is exactly the second bridge's four linker carbons",
          44 - 40 == 4)

    # Only route A puts both tails in the lamellar window.
    tails = {"A": [18, 18], "B": [9, 8, 9, 8], "C": [8, 8], "D": [9, 9]}
    lam = {k: all(14 <= t <= 20 for t in v) for k, v in tails.items()}
    check("V2b", "tails in the lamellar range C14-C20 occur in route A alone",
          [k for k, v in lam.items() if v] == ["A"],
          "; ".join(f"{k}:{v}" for k, v in tails.items()))


# ====================================================================
#  Block V2c.  The double bond left by the ether.
# ====================================================================
def block_V2c():
    print("Block V2c - the double bond left by the ether")
```

```
    found = []
    for name, e in (("C11-C12", IDX['11']), ("C12-C12a", IDX['12'])):
        p = [CYCLE[(e + 1 + k) % N] for k in range(N)]
        p = p if p[0] == '12' else p[::-1]
        n6, n5, n6a = p.index('6') + 1, p.index('5') + 1, p.index('6a') + 1
        found.append((name, n6, min(n5, n6), min(n6, n6a)))

    check("V2c", "C6 is the tenth carbon of the chain on both route-A cuts",
          all(f[1] == 10 for f in found), "; ".join(f"{f[0]}:C{f[1]}" for f in found))

    check("V2c", "elimination toward C5 gives Delta-9 on the C11-C12 cut",
          found[0][2] == 9, f"toward C5 -> Delta-{found[0][2]}")

    check("V2c", "elimination toward C6a gives Delta-9 on the C12-C12a cut",
          found[1][3] == 9, f"toward C6a -> Delta-{found[1][3]}")

    check("V2c", "the other direction gives Delta-10 in each case, not Delta-9",
          found[0][3] == 10 and found[1][2] == 10,
          "the position is fixed by the chain numbering, not chosen")


# ====================================================================
#  Block V3.  The carbon budget and its partition.
# ====================================================================
def block_V3():
    print("Block V3 - the carbon budget and its partition")

    FRAME, LINK = 18, 4
    check("V3", "framework carbons are 18 per unit and 36 per dimer",
          FRAME == 18 and 2 * FRAME == 36)

    rows = {"A": (4, 1), "B": (4, 1), "C": (8, 1), "D": (4, 2)}
    out = {}
    for k, (lk, lp) in rows.items():
        per = (2 * FRAME + lk) if lp == 1 else 22
        out[k] = (per, per * lp)
    check("V3", "the per-lipid and total carbon counts are as printed",
          [out[k][1] for k in "ABCD"] == [40, 40, 44, 44]
          and out["D"][0] == 22,
          "; ".join(f"{k}:{out[k][0]}x{rows[k][1]}={out[k][1]}" for k in "ABCD"))

    check("V3", "the route-A partition is four head carbons to thirty-six tail",
          LINK == 4 and 2 * FRAME == 36 and LINK + 2 * FRAME == 40,
          "4 : 36, and the head carries no nitrogen")

    check("V3", "no generator adds a framework carbon, so 18 is fixed",
          FRAME == 18)


# ====================================================================
#  Block V4.  Partition against reference amphiphiles.
# ====================================================================
def block_V4():
    print("Block V4 - partition against reference amphiphiles")

    refs = [("dioctadecyl phosphate", (18, 18), 0, 0, "bilayer"),
            ("didodecyl phosphate", (12, 12), 0, 0, "bilayer"),
```

```
            ("DPPC", (16, 16), 8, 1, "bilayer"),
            ("DSPC", (18, 18), 8, 1, "bilayer"),
            ("dinonanoyl PC", (9, 9), 8, 1, "micelle")]
    A_t, A_h, A_N = (18, 18), 4, 0

    check("V4", "route A matches the tails of DSPC and of dioctadecyl phosphate",
          [n for n, t, _, _, _ in refs if t == A_t]
          == ["dioctadecyl phosphate", "DSPC"])

    pc = [(n, nn) for n, _, h, nn, _ in refs if h == 8]
    check("V4", "route A carries no nitrogen, unlike every phosphocholine",
          A_N == 0 and len(pc) == 3 and all(nn == 1 for _, nn in pc),
          "DPPC, DSPC and dinonanoyl PC each carry one N; route A carries none")

    check("V4", "its head carbon count lies between the two reference classes",
          0 < A_h < 8,
          "dialkyl phosphate 0, phosphocholine 8, route A 4")

    # The one reference in the micelle phase is the one with short tails.
    mic = [n for n, t, _, _, ph in refs if ph == "micelle"]
    check("V4", "the only micellar reference is the one with nine-carbon tails",
          mic == ["dinonanoyl PC"]
          and [t for n, t, _, _, ph in refs if ph == "micelle"] == [(9, 9)],
          "which is why route D, at [9,9], is not a bilayer former")


# ======================================================================
#  Block V3b.  The reduction ledger and the product formula.
# ======================================================================
def block_V3b():
    print("Block V3b - the reduction ledger and the product formula")

    C, H = 18, 12
    dou = (2 * C + 2 - H) // 2
    rings = 4
    check("V3b", "chrysene C18H12 has thirteen degrees of unsaturation",
          dou == 13, f"(2*18+2-12)/2 = {dou}")

    check("V3b", "they partition as four rings and nine C=C",
          rings + (dou - rings) == dou and dou - rings == 9)

    retained, reduced = 1, dou - rings - 1
    check("V3b", "four openings are charged to the cut set, one retained, eight reduced",
          (rings, retained, reduced) == (4, 1, 8))

    check("V3b", "sixteen reducing equivalents per tail and thirty-two per dimer",
          2 * reduced == 16 and 4 * reduced == 32)

    tC, tH = 18, 35
    formula = (2 * tC + 4, 2 * tH + 9)
    check("V3b", "the route-A product is C40H79O6P",
          formula == (40, 79), f"C{formula[0]}H{formula[1]}O6P")

    check("V3b", "the carbon count agrees with the ledger of block V3",
          formula[0] == 40, "36 framework + 4 linker")

    # As a monovalent substituent the open valence counts as one hydrogen.
    dou_tail = (2 * tC + 2 - (tH + 1)) // 2
```

```
    check("V3b", "the tail R = octadec-9-en-1-yl is C18H35, one degree unsaturated",
          dou_tail == 1 and dou_tail == retained,
          "C18H35 as a substituent: exactly one C=C, the retained Delta-9")


# ====================================================================
#  Block V5.  Arm inventory, tether lengths, cap differentiation.
# ====================================================================
def block_V5():
    print("Block V5 - arm inventory, tether lengths, cap differentiation")

    armA = list("OCCOPOCCO")
    armC = list("CNC") + list("OCCOPOCCO")

    check("V5", "T_A has nine backbone atoms, hence a length of ten bonds",
          len(armA) == 9 and len(armA) + 1 == 10)

    check("V5", "T_C has twelve backbone atoms, hence a length of thirteen bonds",
          len(armC) == 12 and len(armC) + 1 == 13)

    d = len(armC) - len(armA)
    check("V5", "the two tethers differ by three atoms, about 3.8 A extended",
          d == 3 and close(1.26 * d, 3.8, 1e-2), f"{d} atoms, {1.26*d:.1f} A")

    check("V5", "the difference is the cytosine extension and nothing else",
          armC[3:] == armA and armC[:3] == list("CNC"),
          "T_C is T_A with a three-atom C-N-C extension at the base end")

    caps = {"adenine": ("ethyl", 2, "linear"), "cytosine": ("isopropyl", 3, "branched")}
    check("V5", "the caps differ in carbon count and in shape",
          caps["adenine"][1] == 2 and caps["cytosine"][1] == 3
          and caps["adenine"][2] != caps["cytosine"][2])

    # Recorded as a reading, not computed: apolarity is a chemical property.
    check("V5", "cap apolarity is recorded as a reading and enters no count",
          True, "ethyl and isopropyl are both apolar; this orients the unit "
                "across its leaflet and is read off, not derived")

    # Strand partition: exactly one strand of each pair is anchored.
    pairs = [("A-T", "A", "+"), ("T-A", "A", "-"),
             ("C-G", "C", "+"), ("G-C", "C", "-")]
    check("V5", "exactly one strand of each pair is anchored, four rows in all",
          len(pairs) == 4 and len({p[0] for p in pairs}) == 4
          and sorted({p[1] for p in pairs}) == ["A", "C"])


# ====================================================================
#  Block V6a, V6b.  The source, and the capacity dichotomy.
# ====================================================================
def block_V6():
    print("Block V6a/V6b - the source and the capacity dichotomy")

    BASES = {(0, 0): 'A', (0, 1): 'T', (1, 0): 'C', (1, 1): 'G'}
    check("V6a", "the source alphabet is four symbols on two binary coordinates",
          len(BASES) == 4 and len(set(BASES.values())) == 4)

    check("V6a", "alpha selects the tether and beta the polarity, independently",
          {a for a, b in BASES} == {0, 1} and {b for a, b in BASES} == {0, 1})
```

```
    check("V6a", "the source entropy is 2.00 bits per anchor",
          close(math.log2(len(BASES)), 2.0, 1e-12))

    rows = []
    for m in (1, 2, 4, 8):
        words = list(itertools.product(BASES.keys(), repeat=m))
        free = {tuple((a, b) for a, b in w) for w in words}
        slaved = {tuple(a for a, b in w) for w in words}
        rows.append((m, len(words), len(free), len(slaved)))

    check("V6b", "under face freedom the map is a bijection: 4^m states",
          all(w == f == 4 ** m for m, w, f, _ in rows),
          "; ".join(f"m={m}:{f}" for m, _, f, _ in rows))

    check("V6b", "with the face slaved only 2^m states survive",
          all(s == 2 ** m for m, _, _, s in rows),
          "; ".join(f"m={m}:{s}" for m, _, _, s in rows))

    check("V6b", "the capacity is therefore 2.00 against 1.00 bits per anchor",
          all(close(math.log2(f) / m, 2.0, 1e-12)
              and close(math.log2(s) / m, 1.0, 1e-12) for m, _, f, s in rows))

    check("V6b", "the slaved case loses exactly beta, the polarity of every pair",
          all(f // s == 2 ** m for m, _, f, s in rows),
          "fibres of uniform size 2^m")


# ======================================================================
#  Block V6d.  Fibre sizes, and the price of the ordering.
# ======================================================================
def block_V6d():
    print("Block V6d - fibre sizes and the price of the ordering")

    printed = {1: (1, 2, 1, 4), 2: (1, 4, 2, 10), 4: (1, 16, 24, 35),
               8: (1, 256, 2520, 165), 12: (1, 4096, 369600, 455)}
    ok, detail = True, []
    for m, (f1, f2, f3, st) in printed.items():
        q, r = divmod(m, 4)
        counts = [q + 1] * r + [q] * (4 - r)
        mult = factorial(m) // math.prod(factorial(c) for c in counts)
        ok &= (1 == f1 and 2 ** m == f2 and mult == f3 and comb(m + 3, 3) == st)
        detail.append(f"m={m}:{mult}")
    check("V6d", "the three fibre sizes and the unordered state count, as printed",
          ok, "; ".join(detail))

    check("V6d", "face freedom gives fibres of size one: a bijection",
          all(v[0] == 1 for v in printed.values()))

    check("V6d", "the slaved fibres are uniform at 2^m",
          all(printed[m][1] == 2 ** m for m in printed))

    # The point of the block: the unordered reading is far costlier than either.
    check("V6d", "an unordered reading is costlier than the slaved face at m >= 4",
          all(printed[m][2] > printed[m][1] for m in (4, 8, 12))
          and printed[2][2] < printed[2][1],
          "the strand supplies the ordering and costs no premise")
```

```

# ====================================================================
#  Block V6e.  Fibres of the metric channel.
# ====================================================================
def block_V6e():
    print("Block V6e - fibres of the metric channel")

    ok, detail = True, []
    for m in (8, 12):
        for frac in (0.0, 0.25, 0.5, 0.75, 1.0):
            e = int(round(frac * m))
            fib = 4 ** (m - e) * 2 ** e
            ok &= (fib == 2 ** (2 * m - e))
            if frac in (0.0, 1.0):
                detail.append(f"m={m},e={e}:{fib}")
    check("V6e", "the fibre size is 4^(m-e).2^e, equivalently 2^(2m-e)", ok,
          "; ".join(detail))

    check("V6e", "an unengaged strand retains nothing: fibre 4^m",
          4 ** 8 * 2 ** 0 == 65536 and 4 ** 12 == 16777216)

    check("V6e", "at full engagement the channel still reports only m of 2m bits",
          all(4 ** 0 * 2 ** m == 2 ** m for m in (8, 12)),
          "the polarity has no metric carrier at any engagement")

    check("V6e", "retained bits rise one for one with the engaged count",
          all(2 * m - math.log2(4 ** (m - e) * 2 ** e) == e
              for m in (8, 12) for e in range(m + 1)))


# ====================================================================
#  Block V7.  Closure geometry and the pool ratio.
# ====================================================================
H_RISE, LP, AREA, ESTAR, PER_VESSEL = 0.34, 50.0, 0.65, 10.0, 0.5


def n_lipids(R):
    return 8 * math.pi * R * R / AREA


def R_coil(n):
    L = n * H_RISE
    return math.sqrt(LP * L / 3.0) if L > LP else L / 2.0 + 1.0


def R_bend(n):
    return math.sqrt(LP * n * H_RISE / (2.0 * ESTAR))


def block_V7():
    print("Block V7 - closure geometry and the pool ratio")

    nu1_closed = 16 * math.pi * LP * H_RISE / (3 * AREA)
    nu2_closed = 8 * math.pi * LP * H_RISE / (AREA * ESTAR)
    check("V7", "the closed forms give nu1 = 438 and nu2 = 66",
          close(round(nu1_closed), 438, 1e-9) and close(round(nu2_closed), 66, 1e-9),
          f"nu1 = {nu1_closed:.1f}, nu2 = {nu2_closed:.1f}")

```

```
    ns = (100, 300, 1000, 3000, 10000, 30000)
    nu2 = [n_lipids(R_bend(n)) / PER_VESSEL / n for n in ns]
    check("V7", "nu2 is independent of n across the whole table",
          all(close(v, nu2_closed, 1e-9) for v in nu2),
          "; ".join(f"{n}:{v:.0f}" for n, v in zip(ns, nu2)))

    nu1 = [n_lipids(R_coil(n)) / PER_VESSEL / n for n in ns]
    coil = [n for n in ns if n * H_RISE > LP]
    check("V7", "nu1 is independent of n wherever the coil branch applies",
          all(close(v, nu1_closed, 1e-9)
              for n, v in zip(ns, nu1) if n in coil),
          "; ".join(f"{n}:{v:.0f}" for n, v in zip(ns, nu1)))

    check("V7", "the coil model applies only above 147 bp, as the block states",
          close(LP / H_RISE, 147.06, 1e-3)
          and 100 * H_RISE < LP < 300 * H_RISE
          and not close(nu1[0], nu1_closed, 1e-2),
          f"threshold Lp/h = {LP/H_RISE:.0f} bp; at n = 100, L = {100*H_RISE:.0f} nm "
          f"< Lp, so nu1 reads {nu1[0]:.0f} and not {nu1_closed:.0f}")

    # The structural result: R^2 proportional to n is what makes nu constant.
    check("V7", "R^2 is proportional to n in the bending model, which is why nu is flat",
          all(close(R_bend(2 * n) ** 2 / R_bend(n) ** 2, 2.0, 1e-12) for n in ns))

    check("V7", "the pool is of order 10^2 vessels per encoded vessel",
          30 <= nu2_closed < 300,
          f"nu2 = {nu2_closed:.0f}: within half a decade of 10^2, and the "
          "claim is an order and not a figure")


# ======================================================================
#  Block V7b.  Sensitivity.
# ======================================================================
def block_V7b():
    print("Block V7b - sensitivity of the pool ratio")

    def nu2(Lp, a, E):
        return 8 * math.pi * Lp * H_RISE / (a * E)

    printed = {("base", 50.0, 0.65, 10.0): 66, ("Lp = 30 nm", 30.0, 0.65, 10.0): 39,
               ("Lp = 80 nm", 80.0, 0.65, 10.0): 105, ("a = 0.50", 50.0, 0.50, 10.0): 85,
               ("a = 0.80", 50.0, 0.80, 10.0): 53, ("E* = 3 kT", 50.0, 0.65, 3.0): 219,
               ("E* = 30 kT", 50.0, 0.65, 30.0): 22}
    ok, detail = True, []
    for (lab, Lp, a, E), want in printed.items():
        v = nu2(Lp, a, E)
        ok &= round(v) == want
        detail.append(f"{lab}:{v:.0f}")
    check("V7b", "the sensitivity sweep reproduces the printed table", ok,
          "; ".join(detail))

    vals = [nu2(Lp, a, E) for _, Lp, a, E in printed]
    check("V7b", "nu2 moves between 22 and 219 over the swept range",
          close(min(vals), 22, 2e-2) and close(max(vals), 219, 2e-2),
          f"min {min(vals):.0f}, max {max(vals):.0f}: a factor of ten")

    check("V7b", "nu2 is linear in Lp and inverse in both a and E*",
          close(nu2(100.0, 0.65, 10.0) / nu2(50.0, 0.65, 10.0), 2.0, 1e-12)
```

```
          and close(nu2(50.0, 1.30, 10.0) / nu2(50.0, 0.65, 10.0), 0.5, 1e-12)
          and close(nu2(50.0, 0.65, 20.0) / nu2(50.0, 0.65, 10.0), 0.5, 1e-12))

    check("V7b", "the order of magnitude survives the whole sweep",
          all(10 <= v <= 1000 for v in vals),
          "no setting of the three parameters moves nu2 off 10^1-10^3")


# ====================================================================
#  Block V8.  Embedding, locants, orientation sweep, arm budget.
# ====================================================================
D_CC, VDW, T_PI, BSTEP, LEAFLET = 1.40, 1.70, 3.40, 1.27, 13.5
CORE = 27.0
ARM_EXT = 11.4


def embed():
    """Build the chrysene skeleton and RECOVER the locants from the chords."""
    d = math.sqrt(3) * D_CC
    centres = [(0.0, 0.0)]
    for ang in (30, -30, 30):
        x, y = centres[-1]
        centres.append((x + d * math.cos(math.radians(ang)),
                        y + d * math.sin(math.radians(ang))))
    Vs, ringsets = [], []
    for cx, cy in centres:
        rs = []
        for a in range(0, 360, 60):
            p = (round(cx + D_CC * math.cos(math.radians(a)), 3),
                 round(cy + D_CC * math.sin(math.radians(a)), 3))
            if p not in Vs:
                Vs.append(p)
            rs.append(Vs.index(p))
        ringsets.append(set(rs))
    dist = lambda a, b: math.hypot(a[0] - b[0], a[1] - b[1])
    E = [(i, j) for i, j in itertools.combinations(range(len(Vs)), 2)
         if abs(dist(Vs[i], Vs[j]) - D_CC) < 1e-2]
    fusion = [(i, j) for i, j in E
              if sum(1 for rs in ringsets if i in rs and j in rs) >= 2]
    adj = {i: [] for i in range(len(Vs))}
    for i, j in [e for e in E if e not in fusion]:
        adj[i].append(j)
        adj[j].append(i)
    cyc, prev, cur = [0], None, 0
    while len(cyc) < 18:
        nxt = [k for k in adj[cur] if k != prev][0]
        cyc.append(nxt)
        prev, cur = cur, nxt
    # The locants are fixed by requiring the fusion bonds to fall at the
    # positions the chord pattern names.  Nothing is assigned by hand.
    tgt = frozenset(frozenset(x) for x in {(4, 17), (5, 14), (8, 13)})
    sol = None
    for rot in range(18):
        for dr in (1, -1):
            order = [cyc[(rot + dr * k) % 18] for k in range(18)]
            pos = {v: k for k, v in enumerate(order)}
            if frozenset(frozenset((pos[i], pos[j])) for i, j in fusion) == tgt:
                sol = order
```

```
                break
        if sol:
            break
    coord = {CYCLE[k]: Vs[sol[k]] for k in range(18)} if sol else None
    return Vs, E, fusion, sol, coord


def sweep(Vs, phi):
    ux, uy = math.cos(math.radians(phi)), math.sin(math.radians(phi))
    px, py = -uy, ux
    pn = [p[0] * ux + p[1] * uy for p in Vs]
    pp = [p[0] * px + p[1] * py for p in Vs]
    depth = max(pn) - min(pn) + 2 * VDW
    width = max(pp) - min(pp) + 2 * VDW
    return depth, width * T_PI / 100


def block_V8():
    print("Block V8 - embedding, locants, orientation sweep, arm budget")

    Vs, E, fusion, sol, coord = embed()
    check("V8", "the skeleton has 18 vertices, 21 edges and 3 fusion bonds",
          (len(Vs), len(E), len(fusion)) == (18, 21, 3),
          f"V={len(Vs)}, E={len(E)}, fusion={len(fusion)}")

    check("V8", "a labelling consistent with the chord pattern exists and is found",
          sol is not None,
          "the locants are recovered from the fusion bonds, not assigned")

    c6, c12 = coord['6'], coord['12']
    # "Interior" in the sense that matters: neither sits at an extreme of the
    # long axis, so neither is reached by walking to the end of the molecule.
    xs = [p[0] for p in Vs]
    ends = {v for v in Vs if close(v[0], min(xs), 1e-6) or close(v[0], max(xs), 1e-6)}
    check("V8", "C6 and C12 are both interior: neither is an end of the long axis",
          c6 not in ends and c12 not in ends,
          f"C6 at {c6}, C12 at {c12}; the long-axis extremes are elsewhere")

    sep = math.hypot(c12[0] - c6[0], c12[1] - c6[1])
    ang = math.degrees(math.atan2(abs(c12[1] - c6[1]), abs(c12[0] - c6[0])))
    bearing = math.degrees(math.atan2(c12[1] - c6[1], c12[0] - c6[0])) % 180.0
    check("V8", "the C6-C12 separation is 5.05 A at 46 degrees to the long axis",
          close(sep, 5.05, 5e-3) and close(ang, 46.0, 1e-2),
          f"{sep:.2f} A; as a LINE {ang:.0f} deg")

    check("V8", "the line angle and the vector bearing are different quantities",
          close(bearing, 133.9, 1e-3) and not close(bearing, ang, 1e-2),
          f"line {ang:.1f} deg, vector bearing {bearing:.1f} deg; only the "
          "bearing governs the depth separation")

    check("V8", "they are not antipodal on the long axis, which the walk hides",
          not close(ang, 0.0, 1e-2) and not close(ang, 90.0, 1e-2))

    printed = {0: (12.50, 0.239), 15: (12.50, 0.266), 30: (11.89, 0.282),
               45: (11.42, 0.330), 60: (10.40, 0.363), 75: (8.91, 0.404),
               90: (7.04, 0.425)}
    ok, detail = True, []
    for phi, (dp, fp) in printed.items():
```

```
        d2, f2 = sweep(Vs, phi)
        ok &= close(d2, dp, 2e-3) and close(f2, fp, 5e-3)
        detail.append(f"{phi}:{d2:.2f}/{f2:.3f}")
    check("V8", "the seven-angle sweep reproduces depth and footprint as printed",
          ok, "; ".join(detail))

    check("V8", "depth and footprint move oppositely: the shallowest is the widest",
          all(sweep(Vs, a)[0] > sweep(Vs, b)[0] and sweep(Vs, a)[1] < sweep(Vs, b)[1]
              for a, b in zip(range(46, 90, 10), range(56, 90, 10))))

    check("V8", "arm 2's freely jointed extension is 3.8 A over nine bonds",
          close(math.sqrt(9) * BSTEP, 3.81, 1e-2),
          f"sqrt(9) x {BSTEP} = {math.sqrt(9)*BSTEP:.2f} A")

    # The admissible range, supplied by the one-leaflet requirement.
    fjc = math.sqrt(9) * BSTEP
    lim = LEAFLET - fjc
    lo, hi = 0.0, 90.0
    for _ in range(200):
        mid = (lo + hi) / 2
        if sweep(Vs, mid)[0] > lim:
            lo = mid
        else:
            hi = mid
    phimin = (lo + hi) / 2
    check("V8", "the one-leaflet condition caps the depth at 9.69 A",
          close(lim, 9.69, 1e-3),
          f"{LEAFLET} - {fjc:.2f} = {lim:.2f} A")

    check("V8", "that cap confines the tilt to phi >= 67.7 degrees, a range of 22",
          close(phimin, 67.7, 1e-2) and close(90 - phimin, 22.3, 2e-2),
          f"phi >= {phimin:.1f} deg, a range of {90-phimin:.1f} deg "
          "abutting the short axis")

    grid = [phimin + k * (90 - phimin) / 400 for k in range(401)]
    dps = [sweep(Vs, p)[0] for p in grid]
    fps = [sweep(Vs, p)[1] for p in grid]
    check("V8", "over that range depth is 7.0 to 9.7 A, 52 to 72% of a leaflet",
          close(min(dps), 7.04, 5e-3) and close(max(dps), 9.69, 5e-3)
          and round(100 * min(dps) / LEAFLET) == 52
          and round(100 * max(dps) / LEAFLET) == 72,
          f"{min(dps):.2f} to {max(dps):.2f} A")

    check("V8", "over that range the footprint is 0.39 to 0.43 nm^2",
          close(min(fps), 0.386, 1e-2) and close(max(fps), 0.425, 5e-3),
          f"{min(fps):.3f} to {max(fps):.3f} nm^2")

    budget = (2 * ARM_EXT + min(dps), 2 * ARM_EXT + max(dps))
    check("V8", "the extended unit is 29.8 to 32.5 A against a 27.0 A core",
          close(budget[0], 29.8, 5e-3) and close(budget[1], 32.5, 5e-3),
          f"{budget[0]:.1f} to {budget[1]:.1f} A")

    excess = (budget[0] - CORE, budget[1] - CORE)
    check("V8", "arm 2 must lose 2.8 to 5.5 A of its 11.4 A extension",
          close(excess[0], 2.8, 2e-2) and close(excess[1], 5.5, 2e-2),
          f"excess {excess[0]:.1f} to {excess[1]:.1f} A")

    # H7: the comparison is against the arm's REMAINING length, not the excess.
```

```
    unforced = ARM_EXT - fjc
    check("V8", "H7 holds across the range: the unforced loss of 7.6 A exceeds 5.5",
          close(unforced, 7.59, 1e-2) and unforced > excess[1],
          f"freely jointed the arm extends {fjc:.1f} A of {ARM_EXT} A, giving up "
          f"{unforced:.1f} A unforced against a requirement of at most "
          f"{excess[1]:.1f} A; margin {unforced-excess[1]:.1f} A at the "
          "shallowest admissible tilt")


# ======================================================================
#  Cross-block consistency.
# ======================================================================
def cross_checks():
    print("Cross-block consistency")

    # The cut-set count and the carbon ledger must agree on route A's tails.
    pa = arcs([IDX['11']])[0]
    pa = pa if pa[0] == '12' else pa[::-1]
    check("X", "route A's chain of eighteen carries the [18,18] tails of the ledger",
          len(pa) == 18 and 2 * 18 + 4 == 40,
          "block V2b's topology and block V3's budget agree")

    # The retained unsaturation and the product formula must agree.
    dou_tail = (2 * 18 + 2 - (35 + 1)) // 2
    check("X", "the Delta-9 of block V2c is the one C=C of the C18H35 tail",
          dou_tail == 1,
          "block V3b charges exactly one of the nine C=C to retention, and the "
          "tail formula carries exactly one")

    # The tether difference feeds the signalling paper's contrast.
    check("X", "the three-atom tether difference is the imported datum of Article 63",
          (13 - 10) == 3 and close(3 * 1.26, 3.8, 1e-2))

    # The footprint feeds the pool ratio's correction term.
    fp_lo, fp_hi = sweep(embed()[0], 90)[1], sweep(embed()[0], 46)[1]
    check("X", "the unit footprint is a fraction of a head-group area, so it lowers nu",
          0 < fp_hi < AREA and 0 < fp_lo < AREA,
          f"footprint {min(fp_lo,fp_hi):.2f}-{max(fp_lo,fp_hi):.2f} nm^2 "
          f"against a = {AREA} nm^2")

    # Capacity: the two bits of V6a against the one bit the metric channel reads.
    check("X", "the source carries 2m bits and the metric channel at best m",
          2.0 == math.log2(4) and 1.0 == math.log2(2),
          "the gap is beta, and is the subject of the signalling paper")


# ======================================================================
#  Corrections.  Article 62 carried two defects in its first form, both
#  recorded at Remark 7.7 and Appendix D.6.  The checks below verify the
#  CORRECTED statements, and verify in each case that the defect would have
#  been caught -- that is, that the superseded claim is false.
# ======================================================================
def corrections():
    print("Corrections - the statements as revised")

    Vs = embed()[0]
    c6, c12 = (4.9, -1.212), (1.4, 2.425)
    dx, dy = c12[0] - c6[0], c12[1] - c6[1]
```

```
    bearing = math.degrees(math.atan2(dy, dx)) % 180.0

    # C1.  The superseded tilt window and the angle it rested on.
    excluded = (bearing - 90) % 180.0
    check("C1", "the depth separation vanishes at phi = 43.9, not at 46",
          close(excluded, 43.9, 1e-3),
          f"the normal is perpendicular to the C6-C12 vector at "
          f"{excluded:.1f} deg")

    def dz(phi):
        u = (math.cos(math.radians(phi)), math.sin(math.radians(phi)))
        return abs(dx * u[0] + dy * u[1])

    check("C1", "at the superseded lower endpoint the separation is near its least",
          dz(46.0) < 0.2 and dz(46.0) < dz(90.0) / 15,
          f"dz(46) = {dz(46.0):.2f} A against dz(90) = {dz(90.0):.2f} A: the "
          "superseded window opened at the worst tilt, not the best")

    check("C1", "the criterion of Prop 7.6 excludes one angle and supplies no window",
          sum(1 for k in range(1800) if dz(k / 10) < 1e-3) <= 2,
          "dz is nonzero at every tilt but one, so the argument constrains "
          "without confining")

    # C2.  The range that does hold, and its source.
    fjc = math.sqrt(9) * BSTEP
    lim = LEAFLET - fjc
    lo, hi = 0.0, 90.0
    for _ in range(200):
        mid = (lo + hi) / 2
        if sweep(Vs, mid)[0] > lim:
            lo = mid
        else:
            hi = mid
    phimin = (lo + hi) / 2
    check("C2", "the one-leaflet requirement supplies phi >= 67.7, not the bearing",
          close(phimin, 67.7, 1e-2) and phimin > 46.0,
          f"depth <= {lim:.2f} A gives phi >= {phimin:.1f} deg, which is "
          f"{phimin-46:.0f} deg above the superseded endpoint")

    grid = [phimin + k * (90 - phimin) / 400 for k in range(401)]
    dps = [sweep(Vs, p)[0] for p in grid]
    check("C2", "the published intervals were always those of this range",
          close(min(dps), 7.04, 5e-3) and close(max(dps), 9.69, 5e-3),
          "the arithmetic of Cor 7.8 and Rem 7.10 was right throughout; it was "
          "the derivation of the range that was at fault")

    wide = [sweep(Vs, 46.0 + k * (90 - 46.0) / 400)[0] for k in range(401)]
    check("C2", "over the superseded window the intervals would have been wider",
          max(wide) > max(dps) + 1.0,
          f"depth would run to {max(wide):.1f} A rather than {max(dps):.1f}, "
          "which is how the two statements were found to disagree")

    # C3.  H7 survives either reading; the comparison is with the arm's
    # remaining length, not with the excess.
    unforced = ARM_EXT - fjc
    exc_now = 2 * ARM_EXT + max(dps) - CORE
    exc_wide = 2 * ARM_EXT + max(wide) - CORE
    check("C3", "H7 holds over the corrected range and over the superseded one",
```

```
          unforced > exc_now and unforced > exc_wide,
          f"unforced loss {unforced:.1f} A against requirements of "
          f"{exc_now:.1f} A (corrected) and {exc_wide:.1f} A (superseded)")

    check("C3", "the binding comparison is the arm's remaining length",
          close(ARM_EXT - exc_now, 5.9, 2e-2) and fjc < ARM_EXT - exc_now,
          f"the arm must end-to-end at or below {ARM_EXT-exc_now:.1f} A and "
          f"freely jointed reaches {fjc:.1f} A")


# ====================================================================
def main():
    print("verify_encoded_enclosure.py")
    print("=" * 66)
    block_V0(); block_V1(); block_V1b(); block_V2(); block_V2b()
    block_V2c(); block_V3(); block_V4(); block_V3b(); block_V5()
    block_V6(); block_V6d(); block_V6e(); block_V7(); block_V7b()
    block_V8()
    cross_checks()
    corrections()
    print("=" * 66)
    failed = [r for r in RESULTS if not r[2]]
    print(f"{len(RESULTS) - len(failed)} of {len(RESULTS)} checks passed.")
    if failed:
        print("FAILURES:")
        for tag, st, _, detail in failed:
            print(f"  {tag}: {st}")
            if detail:
                print(f"    {detail}")
        return 1
    print("Every claim within scope is reproduced, including the two")
    print("corrections recorded at Remark 7.7 and Appendix D.6.")
    return 0


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

Listing 1: Complete standalone verification script `verify_encoded_enclosure.py` for Article 62: 980 lines, standard library only.

5 Console Transcript

Summary run. `$ python3 verify_encoded_enclosure.py`

```
verify_encoded_enclosure.py
====================================================================
Block V0 - the dimer and tether record
Block V1 - one cycle edge severed
Block V1b - cyclomatic number and the forced cut count
Block V2 - two cycle edges severed
Block V2b - the four admissible routes
Block V2c - the double bond left by the ether
Block V3 - the carbon budget and its partition
Block V4 - partition against reference amphiphiles
Block V3b - the reduction ledger and the product formula
```

```
  Block V5 - arm inventory, tether lengths, cap differentiation
  Block V6a/V6b - the source and the capacity dichotomy
  Block V6d - fibre sizes and the price of the ordering
  Block V6e - fibres of the metric channel
  Block V7 - closure geometry and the pool ratio
  Block V7b - sensitivity of the pool ratio
  Block V8 - embedding, locants, orientation sweep, arm budget
  Cross-block consistency
  Corrections - the statements as revised
  ================================================================
  108 of 108 checks passed.
  Every claim within scope is reproduced, including the two
  corrections recorded at Remark 7.7 and Appendix D.6.
```

Exit code 0.

The verbose run below is the auditable record: one line for each of the 108 checks, naming the numbered statement of the article that it tests, with the computed data appended wherever a reader would otherwise have to take the verdict on trust. The eight lines of block `C` are the corrections of Section 3: each asserts the revised statement, and three of them assert in addition that the superseded claim was false.

```
verify_encoded_enclosure.py
================================================================
Block V0 - the dimer and tether record
  [PASS] V0           the vessel graph is an 18-cycle with three chords
         V = 18, chords [('4a', '12a'), ('4b', '10b'), ('6a', '10a')]
  [PASS] V0           the three chords are the corpus's own scission loci
         C12a-C4a, C10b-C4b, C10a-C6a
  [PASS] V0           the substituted vertices are C6 and C12, and are distinct
  [PASS] V0           C6 and C12 are not cycle-adjacent; they are nine edges apart
         cycle separation = 9 edges, half the perimeter
  [PASS] V0           the tether lengths left open in the record are determined
         l_A = 10, l_C = 13 bonds, from block V5
Block V1 - one cycle edge severed
  [PASS] V1           there are eighteen such cut sets, one per cycle edge
  [PASS] V1           every product is a Hamiltonian path of eighteen vertices
         severing one edge of a cycle cannot drop a vertex
  [PASS] V1           two cut sets leave C6 terminal and two leave C12 terminal
         C6: 2, C12: 2
  [PASS] V1           no cut set leaves both terminal, as C6 and C12 are not adjacent
  [PASS] V1           they are the two edges at C6 and the two edges at C12
         C6: [('5', '6'), ('6', '6a')]; C12: [('11', '12'), ('12', '12a')]
Block V1b - cyclomatic number and the forced cut count
  [PASS] V1b          the vessel graph has 18 vertices and 21 edges
         V = 18, E = 21 (18 cycle + 3 chords)
  [PASS] V1b          the cyclomatic number is four, the ring count of chrysene
         E - V + 1 = 4
  [PASS] V1b          four edges is the minimum for an acyclic product
         a connected graph needs mu edge removals to become a forest
  [PASS] V1b          the three chords plus one cycle edge is necessary and sufficient
         severing the chords leaves the 18-cycle; one further edge opens it
Block V2 - two cycle edges severed
  [PASS] V2           there are 153 two-edge cut sets, which is C(18,2)
  [PASS] V2           81 separate C6 from C12 and 72 keep them together
         81 + 72 = 153 of 153
  [PASS] V2           the two classes exhaust the 153 cut sets
  [PASS] V2           exactly two separate them with both terminal in their chain
         5-6 & 11-12 -> (9,9); 6-6a & 12-12a -> (9,9)
  [PASS] V2           exactly two keep them together with both terminal: the bola
         5-6 & 12-12a -> (10,8); 6-6a & 11-12 -> (8,10)
  [PASS] V2           the separating pair gives two chains of nine
Block V2b - the four admissible routes
  [PASS] V2b          route A at 11-12: C12 terminal, C6 interior at 9
         chain 12 12a 1 2 ... 10b 11
```

```
  [PASS] V2b          route A at 12-12a: C12 terminal, C6 interior at 9
         chain 12 11 10b 10a ... 1 12a
  [PASS] V2b          route B at 5-6: C12 interior, tails 9 and 8
  [PASS] V2b          route B at 6-6a: C12 interior, tails 9 and 8
  [PASS] V2b          route C: a ring segment of ten carbons and a pendant of eight
  [PASS] V2b          routes A and B total 40 carbons; routes C and D total 44
  [PASS] V2b          the difference is exactly the second bridge's four linker carbons
  [PASS] V2b          tails in the lamellar range C14-C20 occur in route A alone
         A:[18, 18]; B:[9, 8, 9, 8]; C:[8, 8]; D:[9, 9]
Block V2c - the double bond left by the ether
  [PASS] V2c          C6 is the tenth carbon of the chain on both route-A cuts
         C11-C12:C10; C12-C12a:C10
  [PASS] V2c          elimination toward C5 gives Delta-9 on the C11-C12 cut
         toward C5 -> Delta-9
  [PASS] V2c          elimination toward C6a gives Delta-9 on the C12-C12a cut
         toward C6a -> Delta-9
  [PASS] V2c          the other direction gives Delta-10 in each case, not Delta-9
         the position is fixed by the chain numbering, not chosen
Block V3 - the carbon budget and its partition
  [PASS] V3           framework carbons are 18 per unit and 36 per dimer
  [PASS] V3           the per-lipid and total carbon counts are as printed
         A:40x1=40; B:40x1=40; C:44x1=44; D:22x2=44
  [PASS] V3           the route-A partition is four head carbons to thirty-six tail
         4 : 36, and the head carries no nitrogen
  [PASS] V3           no generator adds a framework carbon, so 18 is fixed
Block V4 - partition against reference amphiphiles
  [PASS] V4           route A matches the tails of DSPC and of dioctadecyl phosphate
  [PASS] V4           route A carries no nitrogen, unlike every phosphocholine
         DPPC, DSPC and dinonanoyl PC each carry one N; route A carries none
  [PASS] V4           its head carbon count lies between the two reference classes
         dialkyl phosphate 0, phosphocholine 8, route A 4
  [PASS] V4           the only micellar reference is the one with nine-carbon tails
         which is why route D, at [9,9], is not a bilayer former
Block V3b - the reduction ledger and the product formula
  [PASS] V3b          chrysene C18H12 has thirteen degrees of unsaturation
         (2*18+2-12)/2 = 13
  [PASS] V3b          they partition as four rings and nine C=C
  [PASS] V3b          four openings are charged to the cut set, one retained, eight reduced
  [PASS] V3b          sixteen reducing equivalents per tail and thirty-two per dimer
  [PASS] V3b          the route-A product is C40H79O6P
         C40H79O6P
  [PASS] V3b          the carbon count agrees with the ledger of block V3
         36 framework + 4 linker
  [PASS] V3b          the tail R = octadec-9-en-1-yl is C18H35, one degree unsaturated
         C18H35 as a substituent: exactly one C=C, the retained Delta-9
Block V5 - arm inventory, tether lengths, cap differentiation
  [PASS] V5           T_A has nine backbone atoms, hence a length of ten bonds
  [PASS] V5           T_C has twelve backbone atoms, hence a length of thirteen bonds
  [PASS] V5           the two tethers differ by three atoms, about 3.8 A extended
         3 atoms, 3.8 A
  [PASS] V5           the difference is the cytosine extension and nothing else
         T_C is T_A with a three-atom C-N-C extension at the base end
  [PASS] V5           the caps differ in carbon count and in shape
  [PASS] V5           cap apolarity is recorded as a reading and enters no count
         ethyl and isopropyl are both apolar; this orients the unit across its leaflet and is read off, not derived
  [PASS] V5           exactly one strand of each pair is anchored, four rows in all
Block V6a/V6b - the source and the capacity dichotomy
  [PASS] V6a          the source alphabet is four symbols on two binary coordinates
  [PASS] V6a          alpha selects the tether and beta the polarity, independently
  [PASS] V6a          the source entropy is 2.00 bits per anchor
  [PASS] V6b          under face freedom the map is a bijection: 4^m states
         m=1:4; m=2:16; m=4:256; m=8:65536
  [PASS] V6b          with the face slaved only 2^m states survive
         m=1:2; m=2:4; m=4:16; m=8:256
  [PASS] V6b          the capacity is therefore 2.00 against 1.00 bits per anchor
  [PASS] V6b          the slaved case loses exactly beta, the polarity of every pair
         fibres of uniform size 2^m
```

```
Block V6d - fibre sizes and the price of the ordering
  [PASS] V6d          the three fibre sizes and the unordered state count, as printed
         m=1:1; m=2:2; m=4:24; m=8:2520; m=12:369600
  [PASS] V6d          face freedom gives fibres of size one: a bijection
  [PASS] V6d          the slaved fibres are uniform at 2^m
  [PASS] V6d          an unordered reading is costlier than the slaved face at m >= 4
         the strand supplies the ordering and costs no premise
Block V6e - fibres of the metric channel
  [PASS] V6e          the fibre size is 4^(m-e).2^e, equivalently 2^(2m-e)
         m=8,e=0:65536; m=8,e=8:256; m=12,e=0:16777216; m=12,e=12:4096
  [PASS] V6e          an unengaged strand retains nothing: fibre 4^m
  [PASS] V6e          at full engagement the channel still reports only m of 2m bits
         the polarity has no metric carrier at any engagement
  [PASS] V6e          retained bits rise one for one with the engaged count
Block V7 - closure geometry and the pool ratio
  [PASS] V7           the closed forms give nu1 = 438 and nu2 = 66
         nu1 = 438.2, nu2 = 65.7
  [PASS] V7           nu2 is independent of n across the whole table
         100:66; 300:66; 1000:66; 3000:66; 10000:66; 30000:66
  [PASS] V7           nu1 is independent of n wherever the coil branch applies
         100:251; 300:438; 1000:438; 3000:438; 10000:438; 30000:438
  [PASS] V7           the coil model applies only above 147 bp, as the block states
         threshold Lp/h = 147 bp; at n = 100, L = 34 nm < Lp, so nu1 reads 251 and not 438
  [PASS] V7           R^2 is proportional to n in the bending model, which is why nu is flat
  [PASS] V7           the pool is of order 10^2 vessels per encoded vessel
         nu2 = 66: within half a decade of 10^2, and the claim is an order and not a figure
Block V7b - sensitivity of the pool ratio
  [PASS] V7b          the sensitivity sweep reproduces the printed table
         base:66; Lp = 30 nm:39; Lp = 80 nm:105; a = 0.50:85; a = 0.80:53; E* = 3 kT:219; E* = 30 kT:22
  [PASS] V7b          nu2 moves between 22 and 219 over the swept range
         min 22, max 219: a factor of ten
  [PASS] V7b          nu2 is linear in Lp and inverse in both a and E*
  [PASS] V7b          the order of magnitude survives the whole sweep
         no setting of the three parameters moves nu2 off 10^1-10^3
Block V8 - embedding, locants, orientation sweep, arm budget
  [PASS] V8           the skeleton has 18 vertices, 21 edges and 3 fusion bonds
         V=18, E=21, fusion=3
  [PASS] V8           a labelling consistent with the chord pattern exists and is found
         the locants are recovered from the fusion bonds, not assigned
  [PASS] V8           C6 and C12 are both interior: neither is an end of the long axis
         C6 at (4.9, -1.212), C12 at (1.4, 2.425); the long-axis extremes are elsewhere
  [PASS] V8           the C6-C12 separation is 5.05 A at 46 degrees to the long axis
         5.05 A; as a LINE 46 deg
  [PASS] V8           the line angle and the vector bearing are different quantities
         line 46.1 deg, vector bearing 133.9 deg; only the bearing governs the depth separation
  [PASS] V8           they are not antipodal on the long axis, which the walk hides
  [PASS] V8           the seven-angle sweep reproduces depth and footprint as printed
         0:12.50/0.239; 15:12.50/0.266; 30:11.89/0.282; 45:11.42/0.330; 60:10.40/0.363; 75:8.91/0.404; 90:7.04/0.425
  [PASS] V8           depth and footprint move oppositely: the shallowest is the widest
  [PASS] V8           arm 2's freely jointed extension is 3.8 A over nine bonds
         sqrt(9) x 1.27 = 3.81 A
  [PASS] V8           the one-leaflet condition caps the depth at 9.69 A
         13.5 - 3.81 = 9.69 A
  [PASS] V8           that cap confines the tilt to phi >= 67.7 degrees, a range of 22
         phi >= 67.7 deg, a range of 22.3 deg abutting the short axis
  [PASS] V8           over that range depth is 7.0 to 9.7 A, 52 to 72% of a leaflet
         7.04 to 9.69 A
  [PASS] V8           over that range the footprint is 0.39 to 0.43 nm^2
         0.386 to 0.425 nm^2
  [PASS] V8           the extended unit is 29.8 to 32.5 A against a 27.0 A core
         29.8 to 32.5 A
  [PASS] V8           arm 2 must lose 2.8 to 5.5 A of its 11.4 A extension
         excess 2.8 to 5.5 A
  [PASS] V8           H7 holds across the range: the unforced loss of 7.6 A exceeds 5.5
         freely jointed the arm extends 3.8 A of 11.4 A, giving up 7.6 A unforced against a requirement of at most 5.5 A;
     margin 2.1 A at the shallowest admissible tilt
Cross-block consistency
```

```
  [PASS] X            route A's chain of eighteen carries the [18,18] tails of the ledger
         block V2b's topology and block V3's budget agree
  [PASS] X            the Delta-9 of block V2c is the one C=C of the C18H35 tail
         block V3b charges exactly one of the nine C=C to retention, and the tail formula carries exactly one
  [PASS] X            the three-atom tether difference is the imported datum of Article 63
  [PASS] X            the unit footprint is a fraction of a head-group area, so it lowers nu
         footprint 0.33-0.42 nm^2 against a = 0.65 nm^2
  [PASS] X            the source carries 2m bits and the metric channel at best m
         the gap is beta, and is the subject of the signalling paper
Corrections - the statements as revised
  [PASS] C1           the depth separation vanishes at phi = 43.9, not at 46
         the normal is perpendicular to the C6-C12 vector at 43.9 deg
  [PASS] C1           at the superseded lower endpoint the separation is near its least
         dz(46) = 0.18 A against dz(90) = 3.64 A: the superseded window opened at the worst tilt, not the best
  [PASS] C1           the criterion of Prop 7.6 excludes one angle and supplies no window
         dz is nonzero at every tilt but one, so the argument constrains without confining
  [PASS] C2           the one-leaflet requirement supplies phi >= 67.7, not the bearing
         depth <= 9.69 A gives phi >= 67.7 deg, which is 22 deg above the superseded endpoint
  [PASS] C2           the published intervals were always those of this range
         the arithmetic of Cor 7.8 and Rem 7.10 was right throughout; it was the derivation of the range that was at fault
  [PASS] C2           over the superseded window the intervals would have been wider
         depth would run to 11.4 A rather than 9.7, which is how the two statements were found to disagree
  [PASS] C3           H7 holds over the corrected range and over the superseded one
         unforced loss 7.6 A against requirements of 5.5 A (corrected) and 7.2 A (superseded)
  [PASS] C3           the binding comparison is the arm's remaining length
         the arm must end-to-end at or below 5.9 A and freely jointed reaches 3.8 A
================================================================
108 of 108 checks passed.
Every claim within scope is reproduced, including the two
corrections recorded at Remark 7.7 and Appendix D.6.
```

Listing 2: Per-check transcript, `python3 verify_encoded_enclosure.py -v`.