ACF 84 — Lean 4 Proof Dossier

The Placed Source

Exogenous Preparation of the Substituted-Chrysene Assembly, and the Equality of Its Reachable Set with the Endogenous One

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

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

Proof file: `acf-84-proof.lean` Verification harness: `verify_A84.py`

Kernel Certification

Item	Value
Proof file	`acf-84-proof.lean`
Lean version	`4.12.0` (core only, no Mathlib)
Namespace	`ChryseneFormalism.Article84`
Sections	S1–S14
Definitions	58
Theorems	171
Proved by `decide`	171
Depending on no axioms	**171**
Depending on `propext`	0
Depending on `Quot.sound` or `funext`	0
`sorry`	0
`native_decide`	0
Elaboration time	77 s
Exit code	0

What This Certifies

The Lean 4.12.0 kernel independently verifies 171 claims from Article 84. Each is stated over `Nat` and discharged by `decide`: the goal is a decidable proposition, the kernel evaluates a verified Boolean decision procedure, and the proof is accepted only when that evaluation yields `true`. No external oracle; no `native_decide`; no `sorry`; no Mathlib import.

The file assumes nothing at all. An `#print axioms` audit over every theorem reports *"does not depend on any axioms"* in all 171 cases: not `propext`, not `Quot.sound`, not `funext`, not `Classical.choice`. The customary three-axiom footprint quoted in Lean dossiers would overstate what is assumed here, so it is not quoted.

The audit runs *inside* the file. Its last section puts every theorem to `#print axioms`, so a run of `lean acf-84-proof.lean` is not silent: it prints 171 lines, each of which must read "does not depend on any axioms," and then exits 0. That transcript, not a silent run, is the certificate.

What a clean audit means here, and what it still does not

Article 84 is formal, so for this file a kernel-checked theorem *is* a result — which is the opposite of the position the Article 82 proof dossier had to take. Article 82's data are measurements made by other people, entering under a premise about the world that no kernel reaches; this article's data are definitions of the corpus, and every figure it prints follows from them. There is no H_{data} here, and nothing in the file stands in for one.
That makes the limit of this file a different one, and it is worth stating before the theorem index.

Article 84 carries six named hypotheses. H_{frame} asserts an asymmetric reference, without which the alignment bit is not readable at all; H_{place} asserts that an exogenous agent can set a unit at a prescribed site, and is a stipulation that names no mechanism; H_{draw}, H_{ν}, H_{nf} and H_{br} fix the draw, the registration weight, the normal form and the bridging. The article's positive results rest on them, and *none of the six is a theorem of this file.*
So the 171 theorems below certify the combinatorics of the placed source and the arithmetic the article performs on it: that the frame group has twenty orbits on a six-site word, that a blind unit suffices for a blind contact and that the converse fails in 2790 of 13 689 ordered contacts, that the registry term is $\ell - 1$ bits and the species term 23.0 of them at $\rho^* = 200$. They do not certify that an agent exists which can address those twenty classes, and a reader who takes "171 of 171 axiom-free" as settling the engineering has read the wrong thing into it. The mechanism is a Target, and the article says so twice before it says anything else.

Encoding

Everything in this article is already an integer. Article 84 prints costs in bits and probabilities in scientific notation, both of which look as though they need real numbers. Neither does. The admitted fractions at the stipulated $\rho^* = 200$ and $p_3/p_4 = 1/10$ are ratios of integers — $f_{\mathcal{D}} = 2000/2011$, $f_{\mathcal{I}} = 1/2011$, $f_{\mathcal{T}} = 10/2011$ — so every probability is rational; the frame orbits have sizes two and four, so the bit cost of a registry class is an integer and no logarithm is taken to find it; and the quantities that genuinely are not rational are reached by brackets rather than formed. The whole file therefore lives in `Nat`, and no rational encoding appears in it at all.

Words, arm words and the inversion. A word of length ℓ over $\{0, 1\}$ is a natural below 2^{ℓ}, bit i of w being $w/2^i \bmod 2$. An arm word is a natural below 4^4 whose base-four digit at position j is the arm type at locant $3(j + 1)$, with the digits off the family held at zero, so a word and its family determine each other's content without a list. Axiom A2's inversion ι — $3 \leftrightarrow 9$, $6 \leftrightarrow 12$ — is the index involution $0 \leftrightarrow 2$, $1 \leftrightarrow 3$, and acts on locant masks and on arm words by the two definitions `iotaMask` and `iotaWord`. Both are checked to be involutions (S2, S5) rather than assumed to be.

Signs, without negatives. Nothing is subtracted anywhere in this file. `Nat`'s subtraction truncates, and a truncated subtraction inside a certificate is a certificate of something else. So the ℓ-bit complement is stated as the equation `flipN 6 w + w + 1 = 64` and built bit by bit rather than as $63 - w$; a rounding is a pair of strict inequalities rather than an absolute value; the layer cut of S14 is the cross-multiplied equation $5\,159\,780\,352 \times 16 = 339\,738\,624 \times 243$ rather than a quotient; and the exponent $2r - 1$ of the logarithm bracket is cleared by doubling the whole bracket through, which is why `log2Rounds` below reads as it does.

Three brackets, no irrational formed.

`roundsTo` $n\,d\,r\,s$	n/d rounds to r/s	$2ns < 2rd + d \ \wedge\ 2rd < 2ns + d$
`log2Rounds` $n\,d\,r\,s$	$\log_2(n/d)$ rounds to r/s	$2^{2r}d^{2s} \le 2n^{2s} < 4 \cdot 2^{2r}d^{2s}$
`digitsIs` $n\,k$	n has k decimal digits	$10^k \le 10n \ \wedge\ n < 10^k$

The first is $2|ns - rd| < d$ written out in both directions. The second is the bracket $2^{2r-1}d^{2s} \le n^{2s} < 2^{2r+1}d^{2s}$ doubled, so that no exponent has to be decremented over `Nat`. The third is $10^{k-1} \le n < 10^k$ with the lower end multiplied through. Two further definitions, `sciNeg` and `sciPos`, carry a scientific-notation figure to `roundsTo`, and which of the two applies depends on the sign of the exponent.

The binary-logarithm bracket. `log2Rounds` is this file's characteristic device and the counterpart of the antilogarithm bracket of the Article 82 proof file, *inverted*. That paper's figures are logarithms of measured ratios, and its bracket is raised to clear the logarithm; this paper's figures are costs in bits, and the bracket is raised to form one. Rather than approximate $\log_2$ of anything, the bracket is raised to the $2s$, which clears the logarithm and leaves a comparison of two exact integers. They are large: certifying the species term at $\rho^* = 200$ compares two integers of 397 digits. The kernel's arbitrary-precision arithmetic settles them without approximation, which is precisely what no floating-point computation can do.

The cost is negligible, which is worth recording because it is counter-intuitive. All twelve bracket certificates of S12 and S13 together elaborate in under a tenth of a second. Almost the whole 77-second elaboration of this file is one exhaustive search: the 13 689 ordered contacts of S7, each reduced by the kernel. That search was first written with the registration number counted over all sixteen (locant, arm-type) pairs and with the packed arm word and the family mask recomputed inside the inner loop; it exhausted the elaborator's budget. Counting over the four locants instead — which is what the registration number is a count over — and lifting the word and the mask out of the loop brought it to about fifty seconds.

Why the surjectivity claims are Booleans. Four claims of this file are statements that something exists — that χ is onto the square, that the agreement word is onto, that every frame class is reached, that the registry coordinate is onto. The obvious encoding is a bounded existential, and Lean's core library decides $\exists m, m < n \wedge p\,m$ through `Nat.decidableExistsLT`; that instance is built with `decidable_of_iff`, which carries `propext` and `Quot.sound` into the proof term. Written that way, those four theorems were *not* axiom-free, and the audit said so. They are therefore written with a Boolean search `anyB` instead, which assumes nothing; the claim is the same and the audit is empty. This is recorded because the first draft of the file had the weaker form and the audit is what caught it.

What is not encoded. Four things, each of which the harness carries instead.

The *bookkeeping* of the harness's block P15 — six hypotheses, four Targets, eight precedence rulings — has no section here, on purpose. A count of the article's own bookkeeping is not a mathematical claim, and a kernel certificate of $6 = 6$ would be a decoration rather than a result. The file has fourteen sections to the harness's fifteen for that reason alone.

The *draw law* is not built. The harness constructs it over all 20 736 configurations of a four-site column and checks it normalised and of full support at four values of (θ, γ); this file certifies the

multinomial identity that makes it normalised, the factor of sixteen that forbidding one registry value costs, and the two extreme word masses. A mistake in the construction would fail in the harness, not here.

The *enumeration of the frame orbits stops at length eight*, where the harness carries it to twelve. The kernel's evaluation of the length-twelve search overflows its stack, and the closed form — checked integral at every length to twelve, and against the enumeration at every length to eight — is what the article quotes at twelve and sixteen in any case. S3 says which is which.

And H_{frame}, which is what makes the alignment bit readable at all, is named, argued and unmeasured. Nothing here reaches it.

Theorem Index

Section	Name	Statement (informal)
S1	`rounding_accepts_a_correct_value`	1/3 rounds to 0.33
	`rounding_rejects_the_value_below`	and not to 0.32
	`rounding_rejects_the_value_above`	nor to 0.34
	`log2_accepts_an_integral_logarithm`	$\log_2 8 = 3.0$
	`log2_rejects_two_point_nine`	not 2.9
	`log2_rejects_three_point_one`	nor 3.1
	`log2_accepts_a_nonintegral_logarithm`	$\log_2 10 = 3.3$, so the bracket is not only for powers of two
	`log2_rejects_three_point_four`	and not 3.4
	`digits_accepts_four` …	the digit bracket, and its two rejections
	`digits_rejects_five`	
	`sci_accepts_a_negative_exponent`	$1/2011 = 4.97 \times 10^{-4}$
	`sci_rejects_the_mantissa_above`	and not 4.98
	`sci_accepts_a_positive_exponent`	the other branch, $12.273 = 1.2 \times 10^{1}$
	`sci_rejects_the_mantissa_above_positive`	and not 1.3×10^{1}
	`wrong_sci_form_rejects_a_correct_value`	**the guard**: the power of ten in the target rejects a value the correct form accepts
S2	`iota_is_an_involution_on_indices`	$\iota^2 = \mathrm{id}$ on the four locants
	`iota_fixes_D_setwise`	$\iota\mathcal{D} = \mathcal{D}$
	`iota_fixes_T_setwise`	$\iota\mathcal{T} = \mathcal{T}$
	`iota_does_not_fix_I`	$\iota\mathcal{I} \neq \mathcal{I}$
	`I_is_the_only_family_iota_moves`	so there is exactly one frame carrier
	`family_sizes`	two, three and four locants
	`arm_words_at_two_types`	$m(m+1)/2$, m^3, $(m^4+m^2)/2$ at $m = 2$, enumerated
	`arm_words_at_three_types`	the same at $m = 3$
	`arm_words_at_four_types`	and at $m = 4$
	`twenty_one_arm_words_at_two_types`	21 in all
	`seventy_eight_arm_words_at_three_types`	78 in all
	`per_node_alphabet_is_three_hundred_and_twelve`	$4 \times 78 = 312$ site data
	`delivered_alphabet_is_twelve`	twelve on the delivered stock
	`twelve_to_the_eight_has_nine_digits` …	12^n at $n = 8, 16, 27, 54$, by digit bracket
	…`fiftyfour_has_fiftynine_digits`	
S3	`flip_is_the_complement`	**sign-free**: $\mathrm{flip}\, w + w + 1 = 64$
	`flip_is_an_involution`	and $\mathrm{flip}^2 = \mathrm{id}$
	`rev_is_an_involution`	as is the reversal
	`flip_and_rev_commute`	so Fr is the Klein four-group
	`flip_has_no_fixed_word`	0 fixed points

Section	Name	Statement (informal)
	`rev_fixes_eight_words`	8
	`fliprev_fixes_eight_words`	8
	`burnside_is_integral`	$(64{+}0{+}8{+}8) \equiv 0 \bmod 4$
	`burnside_gives_twenty`	and the quotient is twenty
	`twenty_registry_classes_at_six`	**the enumeration agrees**: 20 orbit representatives
	`no_orbit_is_a_singleton`	every orbit has size ≥ 2
	`orbits_have_size_two_or_four`	and size exactly two or four
	`eight_orbits_of_size_two`	8 of them
	`twelve_orbits_of_size_four`	and 12
	`the_orbits_partition_the_words`	$8{\cdot}2 + 12{\cdot}4 = 64$
	`closed_form_agrees_with_the_enumeration`	$N(\ell)$ against the enumeration, $1 \leq \ell \leq 8$
	`closed_form_is_integral`	and divisible by four to $\ell = 12$
	`classes_at_six`	$N(6) = 20$
	`classes_at_twelve`	$N(12) = 1056$
	`classes_at_sixteen`	$N(16) = 16\,512$
S4	`the_encoding_decodes_to_itself`	$2a + b$ recovers the letter
	`chi_is_additive_on_the_square`	on all sixteen pairs
	`chi_is_an_automorphism_of_the_square`	onto, by Boolean search
	`chi_is_injective`	and injective
	`first_coordinate_vanishes_on_the_purines`	**the letter set first**: $\{A, G\}$
	`second_coordinate_is_one_on_the_three_bond_letters`	and $\{C, G\}$
	`the_two_sets_are_different`	they are not the same distinction
	`chi_prime_is_an_automorphism_too`	the draft's $(a, a{+}b)$ is one as well
	`chi_prime_splits_A_and_C_from_G_and_U`	but it splits $\{A, C\}$
	`that_split_is_neither_distinction`	**Rem. "A correction"**: which is why a draft was wrong
	`chi_has_order_three`	$\chi^3 = \mathrm{id}$, $\chi^2 \neq \mathrm{id}$
	`chi_prime_is_an_involution`	so the two maps are not the same map
S5	`agr_lands_in_the_shorter_alphabet`	$\mathrm{agr} : 2^6 \to 2^5$
	`agr_is_onto`	and onto
	`agr_is_flip_invariant`	$\mathrm{agr}(\mathrm{flip}\, w) = \mathrm{agr}\, w$
	`every_fibre_of_agr_has_two_elements`	two-to-one
	`each_fibre_is_a_flip_pair`	and the fibre is the flip pair
	`agr_carries_rev_to_reversal`	equivariantly
	`frame_orbits_are_rev_orbits_of_the_agreement_word`	at every length to eight
	`twenty_agreement_classes`	20 reversal orbits at $\ell = 6$
	`the_palindromes_number_eight`	8
	`burnside_over_rev`	$(32 + 8)/2 = 20$
	`rev_alone_on_six_sites_gives_thirtysix`	36
	`thirtysix_is_not_twenty`	which is *not* the registry class count
	`iota_is_an_involution_on_subsets`	on all sixteen subsets of the locants
	`four_subsets_are_iota_fixed`	4

Section	Name	Statement (informal)
	`the_fixed_subsets_are_the_unions_of_the_two_orbits`	$\emptyset$, $\{3,9\}$, $\{6,12\}$, all four
	`ten_iota_orbits_on_the_subsets`	10
	`burnside_on_the_subsets`	$(16+4)/2 = 10$
S6	`delivered_words_are_arm_words`	the three delivered words are well formed
	`blind_words_at_two_types`	**Thm. "Unit blindness"**: m, 0, m^2 at $m = 2$
	`blind_words_at_three_types`	at $m = 3$
	`blind_words_at_four_types`	and at $m = 4$
	`delivered_D_word_is_blind`	*ee* is blind
	`delivered_T_word_is_blind`	*eeee* is blind
	`delivered_I_word_is_not_blind`	and *eee* is not
	`stabiliser_orders`	order two on $\mathcal{D}$, four on $\mathcal{T}$
S7	`the_indexing_agrees_with_the_enumeration`	9, 27, 81 words, two ways
	`the_indexing_is_injective`	no index names two words
	`one_hundred_and_seventeen_unit_types`	117
	`thirteen_thousand_six_hundred_and_eightynine_contacts`	$117^2 = 13\,689$, counted by the loop that tests them
	`one_blind_unit_suffices`	**Thm. "Contact blindness"**: 13 689 cases, no violation
	`the_converse_fails_in_2790_contacts`	and the converse fails in 2790
	`the_converse_failures_are_a_minority`	under a quarter of them
	`one_sensitive_family_pair_on_the_delivered_stock`	exactly one of the nine
	`the_sensitive_pair_is_I_and_I`	and it is $\mathcal{I}$–$\mathcal{I}$
	`the_I_I_contact_reads_three_and_two`	$\nu = 3$ aligned, 2 inverted
	`the_D_D_contact_reads_two_either_way`	2 either way
	`the_T_T_contact_reads_four_either_way`	and 4 either way
S8	`the_admitted_fractions_sum_to_one`	$2000 + 1 + 10 = 2011$
	`f_I_rounds_to_497e_minus_4`	$f_{\mathcal{I}} = 4.97 \times 10^{-4}$
	`f_I_does_not_round_to_498e_minus_4`	and not 4.98
	`the_II_frequency_is_one_over_2011_squared`	$1/4\,044\,121$
	`the_II_frequency_rounds_to_25e_minus_7`	2.5×10^{-7}
	`the_II_frequency_does_not_round_to_24e_minus_7`	and not 2.4
	`the_sensitive_contact_is_rarer_than_one_in_four_million`	rarer than $1/4{\times}10^6$
S9	`a_blind_stock_reads_alike_in_both_alignments`	**the exponents are equal**, so the law is uniform at every Δ

Section	Name	Statement (informal)
	`the_heritable_field_has_two_to_the_fiftyfour_values`	2^{54}
	`two_to_the_minus_fiftyfour_rounds_to_555e_minus_17`	5.55×10^{-17}
	`two_to_the_fiftyfour_has_seventeen_digits`	17 digits
S10	`twelve_letters_in_the_delivered_alphabet`	$3 \cdot 2 \cdot 2$
	`the_configuration_set_of_four_sites`	$12^4 = 20\,736$
	`the_draw_law_is_normalised`	the multinomial identity
	`forbidding_one_registry_value_cuts_the_support`	**positivity is load-bearing**: a factor of sixteen
	`the_cut_support_is_a_proper_subset`	and the cut set is strictly smaller
	`the_least_word_mass_rounds_to_239e_minus_16`	2.39×10^{-16}
	`the_greatest_word_mass_rounds_to_382e_minus_3`	3.82×10^{-3}
	`their_ratio_is_exactly_two_thousand_to_the_fourth`	2000^4, an integer and not an approximation
	`the_ratio_rounds_to_160e13`	1.60×10^{13}
	`every_class_is_reached`	**Thm. "Equality of the reachable sets"**: every word lies in a representative's orbit
	`the_registry_coordinate_is_onto`	and the registry coordinate is free
S11	`the_agreement_alphabet_of_six_sites_has_thirtytwo_letters`	all 2^5 are realised
	`five_bits_at_the_codon_length`	$\ell - 1 = 5$
	`fiftythree_bits_at_the_heritable_minimum`	$\ell - 1 = 53$
	`the_class_cost_is_four_or_five`	a class costs four or five bits
	`the_class_cost_times_its_orbit_fills_the_word_set`	because the orbit sizes are powers of two
	`the_edge_computation_agrees`	an orbit has size two iff its agreement word is a palindrome
	`the_edge_orbit_fills_the_agreement_alphabet`	so the cost is the same on sites and on edges
S12	`the_probability_at_rho_one_hundred`	$10^{12}/1011^6$
	`the_probability_at_rho_two_hundred`	$8{\times}10^{12}/2011^6$
	`the_first_probability_rounds_to_936e_minus_7`	9.36×10^{-7}
	`the_second_probability_rounds_to_121e_minus_7`	1.21×10^{-7}
	`the_species_term_at_rho_one_hundred_is_twenty_bits`	20.0 bits
	`it_is_not_nineteen_point_nine`	not 19.9
	`it_is_not_twenty_point_one`	nor 20.1
	`the_species_term_at_rho_two_hundred_is_twentythree_bits`	**397-digit comparison**: 23.0 bits

Section	Name	Statement (informal)
	`it_is_not_twentytwo_point_nine`	not 22.9
	`it_is_not_twentythree_point_one`	nor 23.1
	`the_total_at_rho_one_hundred_is_twentyfive_bits`	25.0 bits, bracketed in one step
	`the_total_at_rho_two_hundred_is_twentyeight_bits`	28.0 bits, likewise
S13	`the_share_at_rho_one_hundred_is_twenty_per_cent`	20 per cent at *both* bracket ends
	`the_share_at_rho_one_hundred_is_not_twentyone`	and not 21 at either
	`the_share_at_rho_two_hundred_is_eighteen_per_cent`	18 per cent at both
	`the_share_at_rho_two_hundred_is_not_nineteen`	and not 19
	`the_share_falls_as_the_species_term_grows`	the registry share is the smaller of the two
S14	`the_heritable_minimum_is_fiftyfour`	3×3×6
	`the_least_complete_box_is_twentyseven`	3^3
	`the_lateral_adjacencies_number_seventytwo`	72
	`the_stacking_relations_number_fortyfive`	45
	`the_relations_of_a_box_number_one_hundred_and_seventeen`	117 in all
	`a_layer_has_four_corners_four_edges_and_one_centre`	the degree profile of a 3×3 layer
	`the_degrees_exhaust_the_layer`	and it accounts for all nine sites
	`a_corner_admits_twelve_letters`	12
	`an_edge_site_admits_eight`	8
	`the_centre_admits_four`	4
	`the_per_site_counts_over_the_layer`	four, four and one site of each
	`the_unconstrained_layer`	$12^9 = 5\,159\,780\,352$
	`the_realisable_layer`	$339\,738\,624$
	`the_layer_cut_is_exactly_243_over_16`	**cross-multiplied**, so nothing is divided
	`the_layer_cut_rounds_to_152_tenths`	which is 15.2
	`the_six_layer_cut_is_243_over_16_to_the_sixth`	$243^6/16^6$
	`the_six_layer_cut_rounds_to_12e7`	1.2×10^7
	`the_realisable_count_has_fiftytwo_digits`	52 digits
	`the_unconstrained_count_has_fiftynine_digits`	against 59
	`the_two_counts_differ_by_seven_digits`	a gap of seven decimal orders

Section	Name	Statement (informal)
	`four_to_the_sixteen_has_ten_digits` … `…fiftyfour_has_thirtythree_digits`	the interior lattice alphabet at $n = 16, 27, 54$

Proof Listing: `acf-84-proof.lean`

`acf-84-proof.lean` — complete source

```
/-
  ================================================================
  acf-84-proof.lean

  Formal proof file for
    Annals of the Chrysene Formalism, Article 84
    "The Placed Source: Exogenous Preparation of the Substituted-Chrysene
     Assembly, and the Equality of Its Reachable Set with the Endogenous
     One"
    Charles D. Schaper, Ph.D., Schaper Systems Press

  Lean 4.12.0, core only.  No Mathlib.  No `sorry`.  No `native_decide`.
  Every theorem is stated over `Nat` and discharged by `decide`: the goal
  is a decidable proposition, the kernel evaluates a verified Boolean
  decision procedure, and the proof is accepted only when that evaluation
  returns `true`.

  Companion harness: verify_A84.py.  Its blocks P1-P14 are this file's
  sections S1-S14.  Its block P15 -- the count of hypotheses, targets and
  precedence rulings -- has no counterpart here on purpose: a count of
  the article's own bookkeeping is not a mathematical claim, and a
  kernel certificate of `6 = 6` would be a decoration rather than a
  result.

  ENCODING, in four sentences.
  * Words of length l over {0,1} are naturals below 2^l, bit i of w being
    w / 2^i % 2.  The flip is the l-bit complement and the reversal is the
    digit reversal, both defined by recursion on l rather than by
    subtraction.
  * An arm word on a family is a natural below 4^4 whose base-four digit
    at position j is the arm type at locant 3(j+1); digits off the family
    are held at zero, so a word and its family determine each other's
    content without a list.
  * The inversion iota of Axiom A2 is the locant involution 3<->9, 6<->12,
    which is the index involution 0<->2, 1<->3.
  * Nothing is subtracted anywhere.  A complement is stated as the
    equation flip w + w + 1 = 2^l, a rounding as a pair of strict
    inequalities, and a ratio as a cross-multiplied equation, so Nat's
    truncating subtraction never appears on either side of a certificate.
  ================================================================
-/

set_option exponentiation.threshold 4000
set_option maxRecDepth 20000
set_option maxHeartbeats 4000000

namespace ChryseneFormalism.Article84

-- ================================================================
--  S0.  Counting and summing, tail-recursively
-- ================================================================
--
--  Every exhaustive claim below is a count, and every count is this
--  accumulator.  It is tail-recursive so that the kernel's evaluation of
```

```
--  a search over several thousand cases does not grow the stack with it.

def b2n (b : Bool) : Nat := if b then 1 else 0

def cntGo (f : Nat → Bool) : Nat → Nat → Nat
  | 0, acc => acc
  | n + 1, acc => cntGo f n (acc + b2n (f n))

def cnt (f : Nat → Bool) (n : Nat) : Nat := cntGo f n 0

def sumGo (f : Nat → Nat) : Nat → Nat → Nat
  | 0, acc => acc
  | n + 1, acc => sumGo f n (acc + f n)

def sumf (f : Nat → Nat) (n : Nat) : Nat := sumGo f n 0

/-- The four locant indices, summed over without a list. -/
def four (g : Nat → Bool) : Nat := b2n (g 0) + b2n (g 1) + b2n (g 2) + b2n (g 3)

/-- A bounded search, as a Boolean rather than as an existential.
    `Nat.decidableExistsLT` of the core library decides `∃ m, m < n ∧ p m`
    through `decidable_of_iff`, which carries `propext` and `Quot.sound`
    into the proof term; this does not, and every surjectivity claim
    below is written with it so that the axiom audit stays empty. -/
def anyB (f : Nat → Bool) : Nat → Bool
  | 0 => false
  | n + 1 => f n || anyB f n

-- ============================================================================
--  S1.  The integer devices
-- ============================================================================
--
--  Three brackets carry every figure Article 84 prints.  Each replaces a
--  statement about a real number with a comparison of exact integers.

/-- `n/d` rounds to `r/s`.  Written as two strict inequalities, which is
    `2|ns - rd| < d` without the subtraction.  Half-way cases fail. -/
def roundsTo (n d r s : Nat) : Prop :=
  2 * (n * s) < 2 * (r * d) + d ∧ 2 * (r * d) < 2 * (n * s) + d

/-- `log2 (n/d)` rounds to `r/s`, for `n/d >= 1`.  The bracket
    `2^(2r-1) d^(2s) <= n^(2s) < 2^(2r+1) d^(2s)` is doubled through, so
    the exponent `2r - 1` never has to be formed over `Nat`. -/
def log2Rounds (n d r s : Nat) : Prop :=
  2 ^ (2 * r) * d ^ (2 * s) ≤ 2 * n ^ (2 * s) ∧
    2 * n ^ (2 * s) < 4 * (2 ^ (2 * r) * d ^ (2 * s))

/-- `n` has exactly `k` decimal digits.  `10^(k-1) <= n` is carried as
    `10^k <= 10n`. -/
def digitsIs (n k : Nat) : Prop := 10 ^ k ≤ 10 * n ∧ n < 10 ^ k

/-- `n/d` rounds to `(mant/scale) * 10^(-e)`.  Rounding to a mantissa is
    rounding of the value divided by its power of ten, so the power
    multiplies the value and NOT the target.  Holding it against the
    target instead keeps the tolerance at a half-unit of the target's
    last place while the value grows, which is the error an earlier form
    of the harness made; theorem `wrong_sci_form_rejects_a_correct_value`
    below is what forecloses it. -/
def sciNeg (n d mant scale e : Nat) : Prop := roundsTo (n * 10 ^ e) d mant scale
```

```
/-- The same for a positive exponent: the power divides the value, so it
    multiplies the denominator. -/
def sciPos (n d mant scale e : Nat) : Prop := roundsTo n (d * 10 ^ e) mant scale

instance (n d r s : Nat) : Decidable (roundsTo n d r s) := by
  unfold roundsTo; infer_instance
instance (n d r s : Nat) : Decidable (log2Rounds n d r s) := by
  unfold log2Rounds; infer_instance
instance (n k : Nat) : Decidable (digitsIs n k) := by
  unfold digitsIs; infer_instance
instance (n d m s e : Nat) : Decidable (sciNeg n d m s e) := by
  unfold sciNeg; infer_instance
instance (n d m s e : Nat) : Decidable (sciPos n d m s e) := by
  unfold sciPos; infer_instance

theorem rounding_accepts_a_correct_value : roundsTo 1 3 33 100 := by decide
theorem rounding_rejects_the_value_below : ¬ roundsTo 1 3 32 100 := by decide
theorem rounding_rejects_the_value_above : ¬ roundsTo 1 3 34 100 := by decide

theorem log2_accepts_an_integral_logarithm : log2Rounds 8 1 30 10 := by decide
theorem log2_rejects_two_point_nine : ¬ log2Rounds 8 1 29 10 := by decide
theorem log2_rejects_three_point_one : ¬ log2Rounds 8 1 31 10 := by decide
theorem log2_accepts_a_nonintegral_logarithm : log2Rounds 10 1 33 10 := by
    decide
theorem log2_rejects_three_point_four : ¬ log2Rounds 10 1 34 10 := by decide

theorem digits_accepts_four : digitsIs 1000 4 := by decide
theorem digits_rejects_three : ¬ digitsIs 1000 3 := by decide
theorem digits_rejects_five : ¬ digitsIs 1000 5 := by decide

theorem sci_accepts_a_negative_exponent : sciNeg 1 2011 497 100 4 := by decide
theorem sci_rejects_the_mantissa_above : ¬ sciNeg 1 2011 498 100 4 := by decide
theorem sci_accepts_a_positive_exponent : sciPos 12273 1000 12 10 1 := by decide
theorem sci_rejects_the_mantissa_above_positive : ¬ sciPos 12273 1000 13 10 1 :=
     by decide

/-- The guard.  Putting the power of ten into the target rather than into
    the value rejects a value the correct form accepts, so the two forms
    are not interchangeable and the choice is not a matter of taste. -/
theorem wrong_sci_form_rejects_a_correct_value : ¬ roundsTo 12273 1000 120 10 :=
     by decide

-- ============================================================================
--  S2.  The delivered alphabet and the configuration count
-- ============================================================================
--
--  Article 84, Definition "Site datum" and Theorem "Finiteness and the
--  configuration count".  A site carries a species, an arm word within
--  that species, and a letter of Article 40's structural alphabet Sigma,
--  of order four.  Arm words are counted up to iota, because a word and
--  its image name the same compound.

/-- `4 ^ i` for `i < 4`, by table, so no power is evaluated in an inner
    loop. -/
def pw4 (i : Nat) : Nat := if i = 0 then 1 else if i = 1 then 4 else if i = 2
    then 16 else 64

/-- `2 ^ i` for `i < 4`. -/
```

```
def pw2 (i : Nat) : Nat := if i = 0 then 1 else if i = 1 then 2 else if i = 2
    then 4 else 8

/-- The arm type at locant index `j` of the packed arm word `w`. -/
def armOf (w j : Nat) : Nat := w / pw4 j % 4

/-- Bit `i` of a four-bit locant mask. -/
def bitOf (m i : Nat) : Nat := m / pw2 i % 2

/-- The inversion iota on locant indices: 0<->2 is 3<->9, 1<->3 is
    6<->12. -/
def iotaIdx (i : Nat) : Nat :=
  if i = 0 then 2 else if i = 1 then 3 else if i = 2 then 0 else 1

/-- Families: 0 is D = {6,12}, 1 is I = {3,6,12}, 2 is T = {3,6,9,12}. -/
def maskOf (f : Nat) : Nat := if f = 0 then 10 else if f = 1 then 11 else 15

/-- Membership in a family's locant set, by table. -/
def inFam (f j : Nat) : Bool :=
  if f = 0 then j == 1 || j == 3 else if f = 1 then !(j == 2) else true

/-- iota acting on a locant mask. -/
def iotaMask (m : Nat) : Nat :=
  bitOf m 0 * 4 + bitOf m 1 * 8 + bitOf m 2 * 1 + bitOf m 3 * 2

/-- iota acting on a packed arm word: the digit at `j` becomes the digit
    that stood at `iotaIdx j`. -/
def iotaWord (w : Nat) : Nat :=
  armOf w 2 + armOf w 3 * 4 + armOf w 0 * 16 + armOf w 1 * 64

/-- `w` is an arm word of family `f` over `m` arm types: in-family digits
    below `m`, off-family digits zero. -/
def wordOK (f m w : Nat) : Bool :=
  let ok := fun j => if inFam f j then decide (armOf w j < m) else armOf w j ==
     0
  ok 0 && ok 1 && ok 2 && ok 3

/-- All arm words of family `f` over `m` types. -/
def rawCount (f m : Nat) : Nat := cnt (wordOK f m) 256

/-- Arm words of an iota-stable family, counted up to iota: the words
    that are least in their iota-orbit. -/
def upToIota (f m : Nat) : Nat :=
  cnt (fun w => wordOK f m w && decide (w ≤ iotaWord w)) 256

theorem iota_is_an_involution_on_indices : ∀ i, i < 4 → iotaIdx (iotaIdx i) = i
    := by decide
theorem iota_fixes_D_setwise : iotaMask (maskOf 0) = maskOf 0 := by decide
theorem iota_fixes_T_setwise : iotaMask (maskOf 2) = maskOf 2 := by decide
theorem iota_does_not_fix_I : iotaMask (maskOf 1) ≠ maskOf 1 := by decide
theorem I_is_the_only_family_iota_moves :
    cnt (fun f => !(iotaMask (maskOf f) == maskOf f)) 3 = 1 := by decide

theorem family_sizes : four (inFam 0) = 2 ∧ four (inFam 1) = 3 ∧ four (inFam 2)
    = 4 := by decide

/-- Article 79's three arm-word formulas, enumerated rather than quoted,
    at two, three and four arm types. -/
theorem arm_words_at_two_types :
```

```
    upToIota 0 2 = 2 * 3 / 2 ∧ rawCount 1 2 = 2 ^ 3 ∧ upToIota 2 2 = (2 ^ 4 + 2
        ^ 2) / 2 := by
  decide
theorem arm_words_at_three_types :
    upToIota 0 3 = 3 * 4 / 2 ∧ rawCount 1 3 = 3 ^ 3 ∧ upToIota 2 3 = (3 ^ 4 + 3
        ^ 2) / 2 := by
  decide
theorem arm_words_at_four_types :
    upToIota 0 4 = 4 * 5 / 2 ∧ rawCount 1 4 = 4 ^ 3 ∧ upToIota 2 4 = (4 ^ 4 + 4
        ^ 2) / 2 := by
  decide

theorem twenty_one_arm_words_at_two_types :
    upToIota 0 2 + rawCount 1 2 + upToIota 2 2 = 21 := by decide
theorem seventy_eight_arm_words_at_three_types :
    upToIota 0 3 + rawCount 1 3 + upToIota 2 3 = 78 := by decide

theorem per_node_alphabet_is_three_hundred_and_twelve :
    4 * (upToIota 0 3 + rawCount 1 3 + upToIota 2 3) = 312 := by decide
theorem delivered_alphabet_is_twelve : 3 * 4 = 12 := by decide

theorem twelve_to_the_eight_has_nine_digits : digitsIs (12 ^ 8) 9 := by decide
theorem twelve_to_the_sixteen_has_eighteen_digits : digitsIs (12 ^ 16) 18 := by
    decide
theorem twelve_to_the_twentyseven_has_thirty_digits : digitsIs (12 ^ 27) 30 :=
    by decide
theorem twelve_to_the_fiftyfour_has_fiftynine_digits : digitsIs (12 ^ 54) 59 :=
    by decide

-- ============================================================================
--  S3.  The frame group and the registry classes
-- ============================================================================
--
--  Article 84, Definition "Frame group" and Theorem "Enumeration of the
--  registry classes".  Fr = <flip, rev>, acting on {0,1}^l.

/-- The l-bit complement, built bit by bit so that nothing is
    subtracted. -/
def flipGo : Nat → Nat → Nat → Nat
  | 0, _, _ => 0
  | k + 1, w, p => (if w % 2 = 0 then p else 0) + flipGo k (w / 2) (p * 2)

def flipN (l w : Nat) : Nat := flipGo l w 1

/-- The digit reversal of an l-bit word. -/
def revGo : Nat → Nat → Nat → Nat
  | 0, _, acc => acc
  | k + 1, w, acc => revGo k (w / 2) (acc * 2 + w % 2)

def revN (l w : Nat) : Nat := revGo l w 0

/-- The size of the Fr-orbit of `w`. -/
def orbSize (l w : Nat) : Nat :=
  let a := w; let b := flipN l w; let c := revN l w; let d := flipN l (revN l w)
  1 + b2n (!(b == a)) + b2n (!(c == a || c == b)) + b2n (!(d == a || d == b || d
      == c))

/-- `w` is the least element of its Fr-orbit. -/
def isMinRep (l w : Nat) : Bool :=
```

```
  decide (w ≤ flipN 1 w) && decide (w ≤ revN 1 w) && decide (w ≤ flipN 1 (revN 1
      w))

/-- `N(l) = (2^l + 2^ceil(l/2) + [l even] 2^(l/2)) / 4`. -/
def closedN (l : Nat) : Nat :=
  (2 ^ l + 2 ^ ((l + 1) / 2) + (if l % 2 = 0 then 2 ^ (l / 2) else 0)) / 4

/-- The flip is the complement: stated as an equation between naturals,
    which is what makes it sign-free. -/
theorem flip_is_the_complement : ∀ w, w < 64 → flipN 6 w + w + 1 = 64 := by
    decide
theorem flip_is_an_involution : ∀ w, w < 64 → flipN 6 (flipN 6 w) = w := by
    decide
theorem rev_is_an_involution : ∀ w, w < 64 → revN 6 (revN 6 w) = w := by decide
theorem flip_and_rev_commute :
    ∀ w, w < 64 → flipN 6 (revN 6 w) = revN 6 (flipN 6 w) := by decide
theorem flip_has_no_fixed_word : cnt (fun w => w == flipN 6 w) 64 = 0 := by
    decide
theorem rev_fixes_eight_words : cnt (fun w => w == revN 6 w) 64 = 8 := by decide
theorem fliprev_fixes_eight_words :
    cnt (fun w => w == flipN 6 (revN 6 w)) 64 = 8 := by decide

/-- Burnside over the Klein four-group, with the fixed-point counts taken
    from the three theorems above rather than asserted. -/
theorem burnside_is_integral : (64 + 0 + 8 + 8) % 4 = 0 := by decide
theorem burnside_gives_twenty : (64 + 0 + 8 + 8) / 4 = 20 := by decide

theorem twenty_registry_classes_at_six : cnt (isMinRep 6) 64 = 20 := by decide
theorem no_orbit_is_a_singleton : ∀ w, w < 64 → 2 ≤ orbSize 6 w := by decide
theorem orbits_have_size_two_or_four :
    ∀ w, w < 64 → orbSize 6 w = 2 ∨ orbSize 6 w = 4 := by decide
theorem eight_orbits_of_size_two :
    cnt (fun w => isMinRep 6 w && orbSize 6 w == 2) 64 = 8 := by decide
theorem twelve_orbits_of_size_four :
    cnt (fun w => isMinRep 6 w && orbSize 6 w == 4) 64 = 12 := by decide
theorem the_orbits_partition_the_words : 8 * 2 + 12 * 4 = 64 := by decide

/-- The closed form, against the enumeration, at every length from one to
    eight.  The enumeration stops at eight here while the harness carries
    it to twelve: the kernel's evaluation of the length-twelve search
    overflows its stack, and the closed form is what the article quotes at
    twelve and sixteen in any case. -/
theorem closed_form_agrees_with_the_enumeration :
    ∀ l, l < 9 → 0 < l → closedN l = cnt (isMinRep l) (2 ^ l) := by decide
theorem closed_form_is_integral :
    ∀ l, l < 13 → 0 < l →
      (2 ^ l + 2 ^ ((l + 1) / 2) + (if l % 2 = 0 then 2 ^ (l / 2) else 0)) % 4 =
          0 := by decide
theorem classes_at_six : closedN 6 = 20 := by decide
theorem classes_at_twelve : closedN 12 = 1056 := by decide
theorem classes_at_sixteen : closedN 16 = 16512 := by decide

-- ======================================================================
--  S4.  The letter assignment and the two base-two coordinates
-- ======================================================================
--
--  Article 84, Proposition "The letter assignment" and Proposition "The
--  two base-two coordinates of a register", with the correction that
--  Remark "A correction" records.  Letters are encoded 2a + b, so
```

```
--  A = 0, C = 1, G = 2, U = 3; the purines {A,G} are {0,2} and the
--  three-bond letters {C,G} are {1,2}.

def aOf (p : Nat) : Nat := p / 2
def bOf (p : Nat) : Nat := p % 2

/-- chi (a,b) = (b, a+b), as the article states it. -/
def chi1 (p : Nat) : Nat := bOf p
def chi2 (p : Nat) : Nat := (aOf p + bOf p) % 2
def chi (p : Nat) : Nat := 2 * chi1 p + chi2 p

/-- chi' (a,b) = (a, a+b), the natural first guess, which a draft of the
    article carried. -/
def chiP1 (p : Nat) : Nat := aOf p
def chiP (p : Nat) : Nat := 2 * chiP1 p + (aOf p + bOf p) % 2

/-- Componentwise addition over Z_2 on the square. -/
def xor2 (p q : Nat) : Nat := 2 * ((aOf p + aOf q) % 2) + ((bOf p + bOf q) % 2)

theorem the_encoding_decodes_to_itself : ∀ p, p < 4 → 2 * aOf p + bOf p = p :=
   by decide
theorem chi_is_additive_on_the_square :
    ∀ p, p < 4 → ∀ q, q < 4 → chi (xor2 p q) = xor2 (chi p) (chi q) := by decide
theorem chi_is_an_automorphism_of_the_square :
    ∀ q, q < 4 → anyB (fun p => chi p == q) 4 = true := by decide
theorem chi_is_injective :
    ∀ p, p < 4 → ∀ q, q < 4 → chi p = chi q → p = q := by decide

/-- The two distinctions, stated as LETTER SETS and located in the
    coordinates afterwards -- the order the article insists on, so that
    the identification cannot be fitted to whichever coordinate is at
    hand. -/
theorem first_coordinate_vanishes_on_the_purines :
    ∀ p, p < 4 → (chi1 p = 0 ↔ (p = 0 ∨ p = 2)) := by decide
theorem second_coordinate_is_one_on_the_three_bond_letters :
    ∀ p, p < 4 → (chi2 p = 1 ↔ (p = 1 ∨ p = 2)) := by decide
theorem the_two_sets_are_different : ¬ (∀ p, p < 4 → (chi1 p = 0 ↔ chi2 p = 1))
   := by decide

theorem chi_prime_is_an_automorphism_too :
    ∀ q, q < 4 → anyB (fun p => chiP p == q) 4 = true := by decide
theorem chi_prime_splits_A_and_C_from_G_and_U :
    ∀ p, p < 4 → (chiP1 p = 0 ↔ (p = 0 ∨ p = 1)) := by decide
theorem that_split_is_neither_distinction :
    ¬ (∀ p, p < 4 → (chiP1 p = 0 ↔ (p = 0 ∨ p = 2))) ∧
      ¬ (∀ p, p < 4 → (chiP1 p = 0 ↔ (p = 1 ∨ p = 2))) := by decide
theorem chi_has_order_three :
    (∀ p, p < 4 → chi (chi (chi p)) = p) ∧ ¬ (∀ p, p < 4 → chi (chi p) = p) :=
       by decide
theorem chi_prime_is_an_involution : ∀ p, p < 4 → chiP (chiP p) = p := by decide

-- ============================================================================
--  S5.  The agreement word, and the quotient it performs
-- ============================================================================
--
--  Article 84, Definition "The agreement word" and Lemma "The agreement
--  word is the quotient by the flip".

/-- `agr w` has bit i equal to w_i + w_{i+1} mod 2.  `k` is the number of
```

```
    agreement letters, one fewer than the number of sites. -/
def agrGo : Nat → Nat → Nat
  | 0, _ => 0
  | k + 1, w => (w + w / 2) % 2 + 2 * agrGo k (w / 2)

def agrN (k w : Nat) : Nat := agrGo k w

/-- Representatives of the rev-orbits on agreement words of length k. -/
def revReps (k : Nat) : Nat := cnt (fun e => decide (e ≤ revN k e)) (2 ^ k)

theorem agr_lands_in_the_shorter_alphabet : ∀ w, w < 64 → agrN 5 w < 32 := by
    decide
theorem agr_is_onto : ∀ e, e < 32 → anyB (fun w => agrN 5 w == e) 64 = true :=
    by decide
theorem agr_is_flip_invariant : ∀ w, w < 64 → agrN 5 (flipN 6 w) = agrN 5 w :=
    by decide
theorem every_fibre_of_agr_has_two_elements :
    ∀ w, w < 64 → cnt (fun v => agrN 5 v == agrN 5 w) 64 = 2 := by decide
theorem each_fibre_is_a_flip_pair :
    ∀ w, w < 64 → ∀ v, v < 64 → agrN 5 v = agrN 5 w → (v = w ∨ v = flipN 6 w) :=
        by decide
theorem agr_carries_rev_to_reversal :
    ∀ w, w < 64 → agrN 5 (revN 6 w) = revN 5 (agrN 5 w) := by decide

/-- Hence the Fr-orbits on words correspond to the rev-orbits on
    agreement words, at every length the enumeration reaches. -/
theorem frame_orbits_are_rev_orbits_of_the_agreement_word :
    ∀ k, k < 8 → cnt (isMinRep (k + 1)) (2 ^ (k + 1)) = revReps k := by decide
theorem twenty_agreement_classes : revReps 5 = 20 := by decide
theorem the_palindromes_number_eight : cnt (fun e => e == revN 5 e) 32 = 8 := by
    decide
theorem burnside_over_rev : (32 + 8) / 2 = 20 := by decide

/-- 36 is the count the quotient by rev alone would give on the six-site
    word, and is not the registry class count. -/
theorem rev_alone_on_six_sites_gives_thirtysix : revReps 6 = 36 := by decide
theorem thirtysix_is_not_twenty : revReps 6 ≠ revReps 5 := by decide

/-- The iota-orbits on the sixteen subsets of the four locants.  A
    different count from the frame-class count, and quoted separately by
    the article. -/
theorem iota_is_an_involution_on_subsets :
    ∀ m, m < 16 → iotaMask (iotaMask m) = m := by decide
theorem four_subsets_are_iota_fixed : cnt (fun m => m == iotaMask m) 16 = 4 :=
    by decide
theorem the_fixed_subsets_are_the_unions_of_the_two_orbits :
    ∀ m, m < 16 → ((m == iotaMask m) = (m == 0 || m == 5 || m == 10 || m == 15))
        := by decide
theorem ten_iota_orbits_on_the_subsets :
    cnt (fun m => decide (m ≤ iotaMask m)) 16 = 10 := by decide
theorem burnside_on_the_subsets : (16 + 4) / 2 = 10 := by decide

-- ============================================================================
--  S6.  The inversion, and which families are flip-blind
-- ============================================================================
--
--  Article 84, Theorem "Unit blindness": over m arm types the blind words
--  number m on D, m^2 on T, and none on I.
```

```
/-- Locant index j witnesses a failure of blindness for the arm word w of
    family f: j is in the family but its iota-image is not, or carries a
    different arm type. -/
def badj (f w j : Nat) : Bool :=
  inFam f j && !(inFam f (iotaIdx j) && armOf w (iotaIdx j) == armOf w j)

/-- pres(S,w) = pres_iota(S,w).  Both sides have |S| elements, so the
    containment this counts is equality. -/
def unitBlind (f w : Nat) : Bool := four (badj f w) == 0

def blindCount (f m : Nat) : Nat := cnt (fun w => wordOK f m w && unitBlind f w)
    256

/-- The arm words the source actually delivers: arm type 1 (`e`) at every
    locant of the family. -/
def deliveredWord (f : Nat) : Nat := if f = 0 then 68 else if f = 1 then 69 else
    85

theorem delivered_words_are_arm_words :
    wordOK 0 2 (deliveredWord 0) = true ∧ wordOK 1 2 (deliveredWord 1) = true ∧
      wordOK 2 2 (deliveredWord 2) = true := by decide

theorem blind_words_at_two_types :
    blindCount 0 2 = 2 ∧ blindCount 1 2 = 0 ∧ blindCount 2 2 = 2 ^ 2 := by
       decide
theorem blind_words_at_three_types :
    blindCount 0 3 = 3 ∧ blindCount 1 3 = 0 ∧ blindCount 2 3 = 3 ^ 2 := by
       decide
theorem blind_words_at_four_types :
    blindCount 0 4 = 4 ∧ blindCount 1 4 = 0 ∧ blindCount 2 4 = 4 ^ 2 := by
       decide

theorem delivered_D_word_is_blind : unitBlind 0 (deliveredWord 0) = true := by
   decide
theorem delivered_T_word_is_blind : unitBlind 2 (deliveredWord 2) = true := by
   decide
theorem delivered_I_word_is_not_blind : unitBlind 1 (deliveredWord 1) = false :=
    by decide

/-- The stabiliser of a blind word is the family's own locant set, of
    order two on D and four on T. -/
theorem stabiliser_orders : four (inFam 0) = 2 ∧ four (inFam 2) = 4 := by decide

-- ======================================================================
--  S7.  Contact blindness, and its one exception
-- ======================================================================
--
--  Article 84, Theorem "Contact blindness" and Corollary "What the
--  registration weight reads, and what it does not".  This is the
--  file's one exhaustive search: 117 unit types, 13,689 ordered
--  contacts, every one of them tested.

/-- The number of arm words of family `f` over three arm types: 9, 27,
    81.  Checked against the enumeration of S2 below. -/
def famSize (f : Nat) : Nat := if f = 0 then 9 else if f = 1 then 27 else 81

/-- The `i`-th arm word of family `f` over three arm types, by placing the
    base-three digits of `i` at the family's locants.  Indexing the words
    rather than filtering 256 of them is what brings the double loop down
```

```
    to the 13,689 contacts that exist. -/
def wordOf (f i : Nat) : Nat :=
  if f = 0 then (i % 3) * 4 + (i / 3 % 3) * 64
  else if f = 1 then (i % 3) + (i / 3 % 3) * 4 + (i / 9 % 3) * 64
  else (i % 3) + (i / 3 % 3) * 4 + (i / 9 % 3) * 16 + (i / 27 % 3) * 64

/-- Locant j contributes to the registration number of an aligned
    contact. -/
def hit (f1 w1 f2 w2 j : Nat) : Bool :=
  inFam f1 j && inFam f2 j && armOf w1 j == armOf w2 j

/-- And of the inverted one. -/
def hitF (f1 w1 f2 w2 j : Nat) : Bool :=
  inFam f1 j && inFam f2 (iotaIdx j) && armOf w1 j == armOf w2 (iotaIdx j)

def nuA (f1 w1 f2 w2 : Nat) : Nat := four (hit f1 w1 f2 w2)
def nuF (f1 w1 f2 w2 : Nat) : Nat := four (hitF f1 w1 f2 w2)

/-- The contact reads differently in the two alignments. -/
def sensitive (f1 w1 f2 w2 : Nat) : Bool := !(nuA f1 w1 f2 w2 == nuF f1 w1 f2 w2
    )

/-- A count over all ordered contacts of the three-arm-type stock. -/
def contactCount (p : Nat → Nat → Nat → Nat → Bool) : Nat :=
  sumf (fun f1 => sumf (fun i1 =>
    sumf (fun f2 => sumf (fun i2 => b2n (p f1 (wordOf f1 i1) f2 (wordOf f2 i2)))
      (famSize f2)) 3) (famSize f1)) 3

theorem the_indexing_agrees_with_the_enumeration :
    famSize 0 = rawCount 0 3 ∧ famSize 1 = rawCount 1 3 ∧ famSize 2 = rawCount 2
        3 := by decide
/-- The number of index pairs `j < i` of one family that name the same arm
    word.  Zero, so the indexing of `wordOf` is injective and the double
    loop below really does run over 117 distinct unit types. -/
def dupCount (f : Nat) : Nat :=
  sumf (fun i => cnt (fun j => decide (j < i) && wordOf f i == wordOf f j) (
     famSize f))
    (famSize f)

theorem the_indexing_is_injective :
    dupCount 0 = 0 ∧ dupCount 1 = 0 ∧ dupCount 2 = 0 := by decide
theorem one_hundred_and_seventeen_unit_types :
    famSize 0 + famSize 1 + famSize 2 = 117 := by decide
theorem thirteen_thousand_six_hundred_and_eightynine_contacts :
    contactCount (fun _ _ _ _ => true) = 13689 := by decide

/-- Contact blindness.  One blind unit suffices: not one of the 13,689
    ordered contacts with a blind participant is sensitive. -/
theorem one_blind_unit_suffices :
    contactCount (fun f1 w1 f2 w2 =>
      (unitBlind f1 w1 || unitBlind f2 w2) && sensitive f1 w1 f2 w2) = 0 := by
        decide

/-- And the converse fails, in 2,790 of them: two sighted units may still
    make an insensitive contact. -/
theorem the_converse_fails_in_2790_contacts :
    contactCount (fun f1 w1 f2 w2 =>
      !unitBlind f1 w1 && !unitBlind f2 w2 && !sensitive f1 w1 f2 w2) = 2790 :=
        by decide
```

```
theorem the_converse_failures_are_a_minority : 2790 * 4 < 13689 := by decide

/-- On the delivered stock exactly one family pair is sensitive, and it is
    I--I. -/
theorem one_sensitive_family_pair_on_the_delivered_stock :
    cnt (fun k => sensitive (k / 3) (deliveredWord (k / 3))
                            (k % 3) (deliveredWord (k % 3))) 9 = 1 := by decide
theorem the_sensitive_pair_is_I_and_I :
    ∀ k, k < 9 → (sensitive (k / 3) (deliveredWord (k / 3))
                            (k % 3) (deliveredWord (k % 3)) = true ↔ k = 4) :=
                               by decide
theorem the_I_I_contact_reads_three_and_two :
    nuA 1 (deliveredWord 1) 1 (deliveredWord 1) = 3 ∧
      nuF 1 (deliveredWord 1) 1 (deliveredWord 1) = 2 := by decide
theorem the_D_D_contact_reads_two_either_way :
    nuA 0 (deliveredWord 0) 0 (deliveredWord 0) = 2 ∧
      nuF 0 (deliveredWord 0) 0 (deliveredWord 0) = 2 := by decide
theorem the_T_T_contact_reads_four_either_way :
    nuA 2 (deliveredWord 2) 2 (deliveredWord 2) = 4 ∧
      nuF 2 (deliveredWord 2) 2 (deliveredWord 2) = 4 := by decide

-- ======================================================================
--  S8.  How often the exception arises
-- ======================================================================
--
--  Article 84, Proposition "How often the exception arises".  The
--  admitted fractions come from p_2 : p_3 : p_4 = rho* : x : 1, and at
--  the article's stipulated rho* = 200, x = 1/10 they are exactly
--  rational: f_D = 2000/2011, f_I = 1/2011, f_T = 10/2011.  Every
--  probability of the article is therefore a ratio of integers.

theorem the_admitted_fractions_sum_to_one : 2000 + 1 + 10 = 2011 := by decide
theorem f_I_rounds_to_497e_minus_4 : sciNeg 1 2011 497 100 4 := by decide
theorem f_I_does_not_round_to_498e_minus_4 : ¬ sciNeg 1 2011 498 100 4 := by
    decide
theorem the_II_frequency_is_one_over_2011_squared : 2011 * 2011 = 4044121 := by
    decide
theorem the_II_frequency_rounds_to_25e_minus_7 : sciNeg 1 4044121 25 10 7 := by
    decide
theorem the_II_frequency_does_not_round_to_24e_minus_7 :
    ¬ sciNeg 1 4044121 24 10 7 := by decide
theorem the_sensitive_contact_is_rarer_than_one_in_four_million :
    4000000 < 4044121 := by decide

-- ======================================================================
--  S9.  The endogenous law on the alignment field
-- ======================================================================
--
--  Article 84, Theorem "The endogenous law on the alignment field".  On a
--  stock whose contacts are blind the registration weight is the same in
--  both alignments, so the induced law is uniform whatever the
--  registration free energy is.  What is certified is the equality of the
--  EXPONENTS, so no exponential is formed and the conclusion holds at
--  every value of Delta.

theorem a_blind_stock_reads_alike_in_both_alignments :
    ∀ k, k < 4 →
      nuA (2 * (k / 2)) (deliveredWord (2 * (k / 2)))
```

```
          (2 * (k % 2)) (deliveredWord (2 * (k % 2))) =
      nuF (2 * (k / 2)) (deliveredWord (2 * (k / 2)))
          (2 * (k % 2)) (deliveredWord (2 * (k % 2))) := by decide

theorem the_heritable_field_has_two_to_the_fiftyfour_values :
    2 ^ 54 = 18014398509481984 := by decide
theorem two_to_the_minus_fiftyfour_rounds_to_555e_minus_17 :
    sciNeg 1 (2 ^ 54) 555 100 17 := by decide
theorem two_to_the_fiftyfour_has_seventeen_digits : digitsIs (2 ^ 54) 17 := by
    decide

-- ======================================================================
--  S10.  Reachability, and the positivity the equality needs
-- ======================================================================
--
--  Article 84, Theorem "Equality of the reachable sets" and Theorem
--  "Positivity of the exit distribution".
--
--  The harness builds the draw law over all 20,736 configurations of a
--  four-site column and checks that it is normalised and of full support
--  at four values of (theta, gamma).  This file certifies the identity
--  that makes it normalised, and the support count; the construction of
--  the law itself is the harness's work and is not reproduced here.

theorem twelve_letters_in_the_delivered_alphabet : 3 * 2 * 2 = 12 := by decide
theorem the_configuration_set_of_four_sites : 12 ^ 4 = 20736 := by decide
theorem the_draw_law_is_normalised :
    (2000 + 1 + 10) ^ 4 * 2 ^ 4 * 2 ^ 4 = 2011 ^ 4 * 256 := by decide

/-- Positivity is not decorative.  Forbidding one registry value cuts the
    support of a four-site column by a factor of sixteen, so the equality
    of the reachable sets fails without it. -/
theorem forbidding_one_registry_value_cuts_the_support :
    6 ^ 4 * 16 = 12 ^ 4 := by decide
theorem the_cut_support_is_a_proper_subset : 6 ^ 4 < 12 ^ 4 := by decide

/-- The extremes of the word mass at theta = gamma = 1/2, exactly
    rational because the admitted fractions are. -/
theorem the_least_word_mass_rounds_to_239e_minus_16 :
    sciNeg 1 (2011 ^ 4 * 256) 239 100 16 := by decide
theorem the_greatest_word_mass_rounds_to_382e_minus_3 :
    sciNeg (2000 ^ 4) (2011 ^ 4 * 256) 382 100 3 := by decide
theorem their_ratio_is_exactly_two_thousand_to_the_fourth :
    2000 ^ 4 = 16 * 10 ^ 12 := by decide
theorem the_ratio_rounds_to_160e13 : sciPos (16 * 10 ^ 12) 1 160 100 13 := by
    decide

/-- Every registry word lies in the orbit of a class representative, so
    every one of the twenty classes is reached.  Reachability is what is
    checked; injectivity is what Article 57's capacity theorem expressly
    withholds, and is not checked anywhere. -/
theorem every_class_is_reached :
    ∀ w, w < 64 → anyB (fun v => isMinRep 6 v &&
      (v == w || v == flipN 6 w || v == revN 6 w || v == flipN 6 (revN 6 w))) 64
          = true := by
  decide
theorem the_registry_coordinate_is_onto :
    ∀ r, r < 2 → anyB (fun a => a % 2 == r) 12 = true := by decide
```

```
-- ====================================================================
--  S11.  The registry term of the selection cost
-- ====================================================================
--
--  Article 84, Theorem "The registry term".  A prescribed alignment field
--  on a column of l sites costs l - 1 bits, because that is the number of
--  contacts and the agreement word has one letter for each.

/-- The agreement alphabet of a six-site column has 2^5 letters: each of
    the 32 is realised. -/
theorem the_agreement_alphabet_of_six_sites_has_thirtytwo_letters :
    cnt (fun e => anyB (fun w => agrN 5 w == e) 64) 32 = 32 := by decide
theorem five_bits_at_the_codon_length : 2 ^ 5 = 32 := by decide
theorem fiftythree_bits_at_the_heritable_minimum : 2 ^ 53 * 2 = 2 ^ 54 := by
    decide

/-- The cost of a CLASS rather than a word.  The orbit sizes are powers of
    two, so the cost is an integer and no logarithm is taken. -/
def classCost (w : Nat) : Nat := if orbSize 6 w == 2 then 5 else 4

theorem the_class_cost_is_four_or_five :
    ∀ w, w < 64 → classCost w = 4 ∨ classCost w = 5 := by decide
theorem the_class_cost_times_its_orbit_fills_the_word_set :
    ∀ w, w < 64 → 2 ^ classCost w * orbSize 6 w = 64 := by decide

/-- The same cost computed on edges.  An orbit has size two exactly when
    the agreement word is a palindrome, which is why the two computations
    agree. -/
theorem the_edge_computation_agrees :
    ∀ w, w < 64 → ((orbSize 6 w == 2) = (agrN 5 w == revN 5 (agrN 5 w))) := by
        decide
theorem the_edge_orbit_fills_the_agreement_alphabet :
    ∀ e, e < 32 → 2 ^ (if e == revN 5 e then 5 else 4) * (if e == revN 5 e then
        1 else 2) = 32 := by
  decide

-- ====================================================================
--  S12.  The species term, by the binary-logarithm bracket
-- ====================================================================
--
--  Article 84, Proposition "The species term".  A six-site alternating
--  column D,T,D,T,D,T drawn from the admitted stream has probability
--  f_D^3 f_T^3, exactly rational, and the cost is its surprisal.  The
--  surprisal is the one quantity of the article that is not rational, and
--  it is pinned by the bracket of S1 rather than approximated.
--
--  This is the file's characteristic device, and the counterpart of the
--  antilogarithm bracket of the Article 82 proof file, inverted: that
--  paper's figures are logarithms of measured ratios and the bracket is
--  raised to clear them, this paper's are costs in bits and the bracket
--  is raised to form them.  Certifying the species term at rho* = 200
--  compares two integers of 397 decimal digits.

theorem the_probability_at_rho_one_hundred :
    1000 ^ 3 * 10 ^ 3 = 10 ^ 12 ∧ 1011 ^ 6 = 1067841840583077561 := by decide
theorem the_probability_at_rho_two_hundred :
    2000 ^ 3 * 10 ^ 3 = 8 * 10 ^ 12 ∧ 2011 ^ 6 = 66141253840394383561 := by
        decide
theorem the_first_probability_rounds_to_936e_minus_7 :
```

```
    sciNeg (10 ^ 12) (1011 ^ 6) 936 100 7 := by decide
theorem the_second_probability_rounds_to_121e_minus_7 :
    sciNeg (8 * 10 ^ 12) (2011 ^ 6) 121 100 7 := by decide

theorem the_species_term_at_rho_one_hundred_is_twenty_bits :
    log2Rounds (1011 ^ 6) (10 ^ 12) 200 10 := by decide
theorem it_is_not_nineteen_point_nine : ¬ log2Rounds (1011 ^ 6) (10 ^ 12) 199 10
    := by decide
theorem it_is_not_twenty_point_one : ¬ log2Rounds (1011 ^ 6) (10 ^ 12) 201 10 :=
    by decide

theorem the_species_term_at_rho_two_hundred_is_twentythree_bits :
    log2Rounds (2011 ^ 6) (8 * 10 ^ 12) 230 10 := by decide
theorem it_is_not_twentytwo_point_nine :
    ¬ log2Rounds (2011 ^ 6) (8 * 10 ^ 12) 229 10 := by decide
theorem it_is_not_twentythree_point_one :
    ¬ log2Rounds (2011 ^ 6) (8 * 10 ^ 12) 231 10 := by decide

/-- The total cost of one coded hexamer, bracketed in one step, so that it
    cannot inherit a rounding error from the species term. -/
theorem the_total_at_rho_one_hundred_is_twentyfive_bits :
    log2Rounds (1011 ^ 6 * 2 ^ 5) (10 ^ 12) 250 10 := by decide
theorem the_total_at_rho_two_hundred_is_twentyeight_bits :
    log2Rounds (2011 ^ 6 * 2 ^ 5) (8 * 10 ^ 12) 280 10 := by decide

-- ======================================================================
--  S13.  The registry share, from the bracket endpoints
-- ======================================================================
--
--  The share of the registry term in the total is an integer over a
--  logarithm, so it is not rational.  It is certified by showing that the
--  share rounds to the printed integer at BOTH ends of the bracket S12
--  establishes for the total, hence everywhere inside it.  The endpoints
--  of the bracket for 25.0 are 499/20 and 501/20; for 28.0, 559/20 and
--  561/20.

theorem the_share_at_rho_one_hundred_is_twenty_per_cent :
    roundsTo 10000 499 20 1 ∧ roundsTo 10000 501 20 1 := by decide
theorem the_share_at_rho_one_hundred_is_not_twentyone :
    ¬ roundsTo 10000 499 21 1 ∧ ¬ roundsTo 10000 501 21 1 := by decide
theorem the_share_at_rho_two_hundred_is_eighteen_per_cent :
    roundsTo 10000 559 18 1 ∧ roundsTo 10000 561 18 1 := by decide
theorem the_share_at_rho_two_hundred_is_not_nineteen :
    ¬ roundsTo 10000 559 19 1 ∧ ¬ roundsTo 10000 561 19 1 := by decide
theorem the_share_falls_as_the_species_term_grows : 18 < 20 := by decide

-- ======================================================================
--  S14.  The box: site classes, bridges, and realisability
-- ======================================================================
--
--  Article 84, Proposition "Site classes and interconnections of a box",
--  reading Article 80's heritable minimum of fifty-four nodes.

/-- The lateral degree of site (x,y) in a 3x3 layer. -/
def latDeg (x y : Nat) : Nat :=
  b2n (decide (0 < x)) + b2n (decide (x < 2)) + b2n (decide (0 < y)) + b2n (
      decide (y < 2))

/-- The letters a site of lateral degree d admits: a family can carry a
```

```
    bridge at each lateral neighbour, so D (two arms) is excluded at
    degree three and above and I (three arms) at degree four. -/
def perSite (d : Nat) : Nat :=
  (b2n (decide (d ≤ 2)) + b2n (decide (d ≤ 3)) + b2n (decide (d ≤ 4))) * 4

theorem the_heritable_minimum_is_fiftyfour : 3 * 3 * 6 = 54 := by decide
theorem the_least_complete_box_is_twentyseven : 3 ^ 3 = 27 := by decide
theorem the_lateral_adjacencies_number_seventytwo :
    2 * 3 * 6 + 3 * 2 * 6 = 72 := by decide
theorem the_stacking_relations_number_fortyfive : 3 * 3 * 5 = 45 := by decide
theorem the_relations_of_a_box_number_one_hundred_and_seventeen :
    45 + 72 = 117 := by decide

theorem a_layer_has_four_corners_four_edges_and_one_centre :
    cnt (fun k => latDeg (k / 3) (k % 3) == 2) 9 = 4 ∧
      cnt (fun k => latDeg (k / 3) (k % 3) == 3) 9 = 4 ∧
      cnt (fun k => latDeg (k / 3) (k % 3) == 4) 9 = 1 := by decide
theorem the_degrees_exhaust_the_layer :
    cnt (fun k => latDeg (k / 3) (k % 3) == 2 || latDeg (k / 3) (k % 3) == 3
                  || latDeg (k / 3) (k % 3) == 4) 9 = 9 := by decide

theorem a_corner_admits_twelve_letters : perSite 2 = 12 := by decide
theorem an_edge_site_admits_eight : perSite 3 = 8 := by decide
theorem the_centre_admits_four : perSite 4 = 4 := by decide
theorem the_per_site_counts_over_the_layer :
    cnt (fun k => perSite (latDeg (k / 3) (k % 3)) == 12) 9 = 4 ∧
      cnt (fun k => perSite (latDeg (k / 3) (k % 3)) == 8) 9 = 4 ∧
      cnt (fun k => perSite (latDeg (k / 3) (k % 3)) == 4) 9 = 1 := by decide

theorem the_unconstrained_layer : 12 ^ 9 = 5159780352 := by decide
theorem the_realisable_layer : 12 ^ 4 * 8 ^ 4 * 4 = 339738624 := by decide

/-- The constraint cuts the layer by exactly 243/16.  Stated as a
    cross-multiplied equation, so no division is performed and nothing is
    rounded. -/
theorem the_layer_cut_is_exactly_243_over_16 :
    5159780352 * 16 = 339738624 * 243 := by decide
theorem the_layer_cut_rounds_to_152_tenths :
    roundsTo 5159780352 339738624 152 10 := by decide

theorem the_six_layer_cut_is_243_over_16_to_the_sixth :
    243 ^ 6 = 205891132094649 ∧ 16 ^ 6 = 16777216 := by decide
theorem the_six_layer_cut_rounds_to_12e7 :
    sciPos 205891132094649 16777216 12 10 7 := by decide
theorem the_realisable_count_has_fiftytwo_digits :
    digitsIs (339738624 ^ 6) 52 := by decide
theorem the_unconstrained_count_has_fiftynine_digits :
    digitsIs ((12 ^ 9) ^ 6) 59 := by decide
theorem the_two_counts_differ_by_seven_digits : 52 + 7 = 59 := by decide

/-- The interior lattice alphabet: tetrasubstituted throughout, so four
    letters a site. -/
theorem four_to_the_sixteen_has_ten_digits : digitsIs (4 ^ 16) 10 := by decide
theorem four_to_the_twentyseven_has_seventeen_digits : digitsIs (4 ^ 27) 17 :=
  by decide
theorem four_to_the_fiftyfour_has_thirtythree_digits : digitsIs (4 ^ 54) 33 :=
  by decide

end ChryseneFormalism.Article84
```

```
-- ====================================================================
--  Kernel axiom audit
-- ====================================================================
--
--  Every theorem above, put to `#print axioms`.  Each line of the output
--  should read "does not depend on any axioms".  The audit is part of the
--  file, so running it is running the certificate.

open ChryseneFormalism.Article84

#print axioms rounding_accepts_a_correct_value
#print axioms rounding_rejects_the_value_below
#print axioms rounding_rejects_the_value_above
#print axioms log2_accepts_an_integral_logarithm
#print axioms log2_rejects_two_point_nine
#print axioms log2_rejects_three_point_one
#print axioms log2_accepts_a_nonintegral_logarithm
#print axioms log2_rejects_three_point_four
#print axioms digits_accepts_four
#print axioms digits_rejects_three
#print axioms digits_rejects_five
#print axioms sci_accepts_a_negative_exponent
#print axioms sci_rejects_the_mantissa_above
#print axioms sci_accepts_a_positive_exponent
#print axioms sci_rejects_the_mantissa_above_positive
#print axioms wrong_sci_form_rejects_a_correct_value
#print axioms iota_is_an_involution_on_indices
#print axioms iota_fixes_D_setwise
#print axioms iota_fixes_T_setwise
#print axioms iota_does_not_fix_I
#print axioms I_is_the_only_family_iota_moves
#print axioms family_sizes
#print axioms arm_words_at_two_types
#print axioms arm_words_at_three_types
#print axioms arm_words_at_four_types
#print axioms twenty_one_arm_words_at_two_types
#print axioms seventy_eight_arm_words_at_three_types
#print axioms per_node_alphabet_is_three_hundred_and_twelve
#print axioms delivered_alphabet_is_twelve
#print axioms twelve_to_the_eight_has_nine_digits
#print axioms twelve_to_the_sixteen_has_eighteen_digits
#print axioms twelve_to_the_twentyseven_has_thirty_digits
#print axioms twelve_to_the_fiftyfour_has_fiftynine_digits
#print axioms flip_is_the_complement
#print axioms flip_is_an_involution
#print axioms rev_is_an_involution
#print axioms flip_and_rev_commute
#print axioms flip_has_no_fixed_word
#print axioms rev_fixes_eight_words
#print axioms fliprev_fixes_eight_words
#print axioms burnside_is_integral
#print axioms burnside_gives_twenty
#print axioms twenty_registry_classes_at_six
#print axioms no_orbit_is_a_singleton
#print axioms orbits_have_size_two_or_four
#print axioms eight_orbits_of_size_two
#print axioms twelve_orbits_of_size_four
#print axioms the_orbits_partition_the_words
```

```
#print axioms closed_form_agrees_with_the_enumeration
#print axioms closed_form_is_integral
#print axioms classes_at_six
#print axioms classes_at_twelve
#print axioms classes_at_sixteen
#print axioms the_encoding_decodes_to_itself
#print axioms chi_is_additive_on_the_square
#print axioms chi_is_an_automorphism_of_the_square
#print axioms chi_is_injective
#print axioms first_coordinate_vanishes_on_the_purines
#print axioms second_coordinate_is_one_on_the_three_bond_letters
#print axioms the_two_sets_are_different
#print axioms chi_prime_is_an_automorphism_too
#print axioms chi_prime_splits_A_and_C_from_G_and_U
#print axioms that_split_is_neither_distinction
#print axioms chi_has_order_three
#print axioms chi_prime_is_an_involution
#print axioms agr_lands_in_the_shorter_alphabet
#print axioms agr_is_onto
#print axioms agr_is_flip_invariant
#print axioms every_fibre_of_agr_has_two_elements
#print axioms each_fibre_is_a_flip_pair
#print axioms agr_carries_rev_to_reversal
#print axioms frame_orbits_are_rev_orbits_of_the_agreement_word
#print axioms twenty_agreement_classes
#print axioms the_palindromes_number_eight
#print axioms burnside_over_rev
#print axioms rev_alone_on_six_sites_gives_thirtysix
#print axioms thirtysix_is_not_twenty
#print axioms iota_is_an_involution_on_subsets
#print axioms four_subsets_are_iota_fixed
#print axioms the_fixed_subsets_are_the_unions_of_the_two_orbits
#print axioms ten_iota_orbits_on_the_subsets
#print axioms burnside_on_the_subsets
#print axioms delivered_words_are_arm_words
#print axioms blind_words_at_two_types
#print axioms blind_words_at_three_types
#print axioms blind_words_at_four_types
#print axioms delivered_D_word_is_blind
#print axioms delivered_T_word_is_blind
#print axioms delivered_I_word_is_not_blind
#print axioms stabiliser_orders
#print axioms the_indexing_agrees_with_the_enumeration
#print axioms the_indexing_is_injective
#print axioms one_hundred_and_seventeen_unit_types
#print axioms thirteen_thousand_six_hundred_and_eightynine_contacts
#print axioms one_blind_unit_suffices
#print axioms the_converse_fails_in_2790_contacts
#print axioms the_converse_failures_are_a_minority
#print axioms one_sensitive_family_pair_on_the_delivered_stock
#print axioms the_sensitive_pair_is_I_and_I
#print axioms the_I_I_contact_reads_three_and_two
#print axioms the_D_D_contact_reads_two_either_way
#print axioms the_T_T_contact_reads_four_either_way
#print axioms the_admitted_fractions_sum_to_one
#print axioms f_I_rounds_to_497e_minus_4
#print axioms f_I_does_not_round_to_498e_minus_4
#print axioms the_II_frequency_is_one_over_2011_squared
#print axioms the_II_frequency_rounds_to_25e_minus_7
```

```
#print axioms the_II_frequency_does_not_round_to_24e_minus_7
#print axioms the_sensitive_contact_is_rarer_than_one_in_four_million
#print axioms a_blind_stock_reads_alike_in_both_alignments
#print axioms the_heritable_field_has_two_to_the_fiftyfour_values
#print axioms two_to_the_minus_fiftyfour_rounds_to_555e_minus_17
#print axioms two_to_the_fiftyfour_has_seventeen_digits
#print axioms twelve_letters_in_the_delivered_alphabet
#print axioms the_configuration_set_of_four_sites
#print axioms the_draw_law_is_normalised
#print axioms forbidding_one_registry_value_cuts_the_support
#print axioms the_cut_support_is_a_proper_subset
#print axioms the_least_word_mass_rounds_to_239e_minus_16
#print axioms the_greatest_word_mass_rounds_to_382e_minus_3
#print axioms their_ratio_is_exactly_two_thousand_to_the_fourth
#print axioms the_ratio_rounds_to_160e13
#print axioms every_class_is_reached
#print axioms the_registry_coordinate_is_onto
#print axioms the_agreement_alphabet_of_six_sites_has_thirtytwo_letters
#print axioms five_bits_at_the_codon_length
#print axioms fiftythree_bits_at_the_heritable_minimum
#print axioms the_class_cost_is_four_or_five
#print axioms the_class_cost_times_its_orbit_fills_the_word_set
#print axioms the_edge_computation_agrees
#print axioms the_edge_orbit_fills_the_agreement_alphabet
#print axioms the_probability_at_rho_one_hundred
#print axioms the_probability_at_rho_two_hundred
#print axioms the_first_probability_rounds_to_936e_minus_7
#print axioms the_second_probability_rounds_to_121e_minus_7
#print axioms the_species_term_at_rho_one_hundred_is_twenty_bits
#print axioms it_is_not_nineteen_point_nine
#print axioms it_is_not_twenty_point_one
#print axioms the_species_term_at_rho_two_hundred_is_twentythree_bits
#print axioms it_is_not_twentytwo_point_nine
#print axioms it_is_not_twentythree_point_one
#print axioms the_total_at_rho_one_hundred_is_twentyfive_bits
#print axioms the_total_at_rho_two_hundred_is_twentyeight_bits
#print axioms the_share_at_rho_one_hundred_is_twenty_per_cent
#print axioms the_share_at_rho_one_hundred_is_not_twentyone
#print axioms the_share_at_rho_two_hundred_is_eighteen_per_cent
#print axioms the_share_at_rho_two_hundred_is_not_nineteen
#print axioms the_share_falls_as_the_species_term_grows
#print axioms the_heritable_minimum_is_fiftyfour
#print axioms the_least_complete_box_is_twentyseven
#print axioms the_lateral_adjacencies_number_seventytwo
#print axioms the_stacking_relations_number_fortyfive
#print axioms the_relations_of_a_box_number_one_hundred_and_seventeen
#print axioms a_layer_has_four_corners_four_edges_and_one_centre
#print axioms the_degrees_exhaust_the_layer
#print axioms a_corner_admits_twelve_letters
#print axioms an_edge_site_admits_eight
#print axioms the_centre_admits_four
#print axioms the_per_site_counts_over_the_layer
#print axioms the_unconstrained_layer
#print axioms the_realisable_layer
#print axioms the_layer_cut_is_exactly_243_over_16
#print axioms the_layer_cut_rounds_to_152_tenths
#print axioms the_six_layer_cut_is_243_over_16_to_the_sixth
#print axioms the_six_layer_cut_rounds_to_12e7
#print axioms the_realisable_count_has_fiftytwo_digits
```

```
#print axioms the_unconstrained_count_has_fiftynine_digits
#print axioms the_two_counts_differ_by_seven_digits
#print axioms four_to_the_sixteen_has_ten_digits
#print axioms four_to_the_twentyseven_has_seventeen_digits
#print axioms four_to_the_fiftyfour_has_thirtythree_digits
```

Console Transcript: the Audit Itself

What a run of the file prints

The 171 lines below are the output of `lean acf-84-proof.lean`, not a separate measurement. Every one of them reads *"does not depend on any axioms"*; there is no other kind of line in the transcript, no warning, and no error. The elapsed time and the exit code are at the foot of it.

$ lean acf-84-proof.lean

```
$ lean --version
Lean (version 4.12.0, x86_64-unknown-linux-gnu, commit dc2533473114, Release)

$ lean acf-84-proof.lean
'ChryseneFormalism.Article84.rounding_accepts_a_correct_value' does not depend on any axioms
'ChryseneFormalism.Article84.rounding_rejects_the_value_below' does not depend on any axioms
'ChryseneFormalism.Article84.rounding_rejects_the_value_above' does not depend on any axioms
'ChryseneFormalism.Article84.log2_accepts_an_integral_logarithm' does not depend on any axioms
'ChryseneFormalism.Article84.log2_rejects_two_point_nine' does not depend on any axioms
'ChryseneFormalism.Article84.log2_rejects_three_point_one' does not depend on any axioms
'ChryseneFormalism.Article84.log2_accepts_a_nonintegral_logarithm' does not depend on any axioms
'ChryseneFormalism.Article84.log2_rejects_three_point_four' does not depend on any axioms
'ChryseneFormalism.Article84.digits_accepts_four' does not depend on any axioms
'ChryseneFormalism.Article84.digits_rejects_three' does not depend on any axioms
'ChryseneFormalism.Article84.digits_rejects_five' does not depend on any axioms
'ChryseneFormalism.Article84.sci_accepts_a_negative_exponent' does not depend on any axioms
'ChryseneFormalism.Article84.sci_rejects_the_mantissa_above' does not depend on any axioms
'ChryseneFormalism.Article84.sci_accepts_a_positive_exponent' does not depend on any axioms
'ChryseneFormalism.Article84.sci_rejects_the_mantissa_above_positive' does not depend on any axioms
'ChryseneFormalism.Article84.wrong_sci_form_rejects_a_correct_value' does not depend on any axioms
'ChryseneFormalism.Article84.iota_is_an_involution_on_indices' does not depend on any axioms
'ChryseneFormalism.Article84.iota_fixes_D_setwise' does not depend on any axioms
'ChryseneFormalism.Article84.iota_fixes_T_setwise' does not depend on any axioms
'ChryseneFormalism.Article84.iota_does_not_fix_I' does not depend on any axioms
'ChryseneFormalism.Article84.I_is_the_only_family_iota_moves' does not depend
```

```
    on any axioms
'ChryseneFormalism.Article84.family_sizes' does not depend on any axioms
'ChryseneFormalism.Article84.arm_words_at_two_types' does not depend on any
    axioms
'ChryseneFormalism.Article84.arm_words_at_three_types' does not depend on any
    axioms
'ChryseneFormalism.Article84.arm_words_at_four_types' does not depend on any
    axioms
'ChryseneFormalism.Article84.twenty_one_arm_words_at_two_types' does not
    depend on any axioms
'ChryseneFormalism.Article84.seventy_eight_arm_words_at_three_types' does not
    depend on any axioms
'ChryseneFormalism.Article84.per_node_alphabet_is_three_hundred_and_twelve'
    does not depend on any axioms
'ChryseneFormalism.Article84.delivered_alphabet_is_twelve' does not depend on
    any axioms
'ChryseneFormalism.Article84.twelve_to_the_eight_has_nine_digits' does not
    depend on any axioms
'ChryseneFormalism.Article84.twelve_to_the_sixteen_has_eighteen_digits' does
    not depend on any axioms
'ChryseneFormalism.Article84.twelve_to_the_twentyseven_has_thirty_digits' does
     not depend on any axioms
'ChryseneFormalism.Article84.twelve_to_the_fiftyfour_has_fiftynine_digits'
    does not depend on any axioms
'ChryseneFormalism.Article84.flip_is_the_complement' does not depend on any
    axioms
'ChryseneFormalism.Article84.flip_is_an_involution' does not depend on any
    axioms
'ChryseneFormalism.Article84.rev_is_an_involution' does not depend on any
    axioms
'ChryseneFormalism.Article84.flip_and_rev_commute' does not depend on any
    axioms
'ChryseneFormalism.Article84.flip_has_no_fixed_word' does not depend on any
    axioms
'ChryseneFormalism.Article84.rev_fixes_eight_words' does not depend on any
    axioms
'ChryseneFormalism.Article84.fliprev_fixes_eight_words' does not depend on any
     axioms
'ChryseneFormalism.Article84.burnside_is_integral' does not depend on any
    axioms
'ChryseneFormalism.Article84.burnside_gives_twenty' does not depend on any
    axioms
'ChryseneFormalism.Article84.twenty_registry_classes_at_six' does not depend
    on any axioms
'ChryseneFormalism.Article84.no_orbit_is_a_singleton' does not depend on any
    axioms
'ChryseneFormalism.Article84.orbits_have_size_two_or_four' does not depend on
    any axioms
'ChryseneFormalism.Article84.eight_orbits_of_size_two' does not depend on any
    axioms
'ChryseneFormalism.Article84.twelve_orbits_of_size_four' does not depend on
    any axioms
'ChryseneFormalism.Article84.the_orbits_partition_the_words' does not depend
    on any axioms
'ChryseneFormalism.Article84.closed_form_agrees_with_the_enumeration' does not
     depend on any axioms
'ChryseneFormalism.Article84.closed_form_is_integral' does not depend on any
    axioms
'ChryseneFormalism.Article84.classes_at_six' does not depend on any axioms
```

```
'ChryseneFormalism.Article84.classes_at_twelve' does not depend on any axioms
'ChryseneFormalism.Article84.classes_at_sixteen' does not depend on any axioms
'ChryseneFormalism.Article84.the_encoding_decodes_to_itself' does not depend
    on any axioms
'ChryseneFormalism.Article84.chi_is_additive_on_the_square' does not depend on
     any axioms
'ChryseneFormalism.Article84.chi_is_an_automorphism_of_the_square' does not
    depend on any axioms
'ChryseneFormalism.Article84.chi_is_injective' does not depend on any axioms
'ChryseneFormalism.Article84.first_coordinate_vanishes_on_the_purines' does
    not depend on any axioms
'ChryseneFormalism.Article84.
    second_coordinate_is_one_on_the_three_bond_letters' does not depend on any
     axioms
'ChryseneFormalism.Article84.the_two_sets_are_different' does not depend on
    any axioms
'ChryseneFormalism.Article84.chi_prime_is_an_automorphism_too' does not depend
     on any axioms
'ChryseneFormalism.Article84.chi_prime_splits_A_and_C_from_G_and_U' does not
    depend on any axioms
'ChryseneFormalism.Article84.that_split_is_neither_distinction' does not
    depend on any axioms
'ChryseneFormalism.Article84.chi_has_order_three' does not depend on any
    axioms
'ChryseneFormalism.Article84.chi_prime_is_an_involution' does not depend on
    any axioms
'ChryseneFormalism.Article84.agr_lands_in_the_shorter_alphabet' does not
    depend on any axioms
'ChryseneFormalism.Article84.agr_is_onto' does not depend on any axioms
'ChryseneFormalism.Article84.agr_is_flip_invariant' does not depend on any
    axioms
'ChryseneFormalism.Article84.every_fibre_of_agr_has_two_elements' does not
    depend on any axioms
'ChryseneFormalism.Article84.each_fibre_is_a_flip_pair' does not depend on any
     axioms
'ChryseneFormalism.Article84.agr_carries_rev_to_reversal' does not depend on
    any axioms
'ChryseneFormalism.Article84.frame_orbits_are_rev_orbits_of_the_agreement_word
    ' does not depend on any axioms
'ChryseneFormalism.Article84.twenty_agreement_classes' does not depend on any
    axioms
'ChryseneFormalism.Article84.the_palindromes_number_eight' does not depend on
    any axioms
'ChryseneFormalism.Article84.burnside_over_rev' does not depend on any axioms
'ChryseneFormalism.Article84.rev_alone_on_six_sites_gives_thirtysix' does not
    depend on any axioms
'ChryseneFormalism.Article84.thirtysix_is_not_twenty' does not depend on any
    axioms
'ChryseneFormalism.Article84.iota_is_an_involution_on_subsets' does not depend
     on any axioms
'ChryseneFormalism.Article84.four_subsets_are_iota_fixed' does not depend on
    any axioms
'ChryseneFormalism.Article84.
    the_fixed_subsets_are_the_unions_of_the_two_orbits' does not depend on any
     axioms
'ChryseneFormalism.Article84.ten_iota_orbits_on_the_subsets' does not depend
    on any axioms
'ChryseneFormalism.Article84.burnside_on_the_subsets' does not depend on any
    axioms
```

```
'ChryseneFormalism.Article84.delivered_words_are_arm_words' does not depend on any axioms
'ChryseneFormalism.Article84.blind_words_at_two_types' does not depend on any axioms
'ChryseneFormalism.Article84.blind_words_at_three_types' does not depend on any axioms
'ChryseneFormalism.Article84.blind_words_at_four_types' does not depend on any axioms
'ChryseneFormalism.Article84.delivered_D_word_is_blind' does not depend on any axioms
'ChryseneFormalism.Article84.delivered_T_word_is_blind' does not depend on any axioms
'ChryseneFormalism.Article84.delivered_I_word_is_not_blind' does not depend on any axioms
'ChryseneFormalism.Article84.stabiliser_orders' does not depend on any axioms
'ChryseneFormalism.Article84.the_indexing_agrees_with_the_enumeration' does not depend on any axioms
'ChryseneFormalism.Article84.the_indexing_is_injective' does not depend on any axioms
'ChryseneFormalism.Article84.one_hundred_and_seventeen_unit_types' does not depend on any axioms
'ChryseneFormalism.Article84.thirteen_thousand_six_hundred_and_eightynine_contacts' does not depend on any axioms
'ChryseneFormalism.Article84.one_blind_unit_suffices' does not depend on any axioms
'ChryseneFormalism.Article84.the_converse_fails_in_2790_contacts' does not depend on any axioms
'ChryseneFormalism.Article84.the_converse_failures_are_a_minority' does not depend on any axioms
'ChryseneFormalism.Article84.one_sensitive_family_pair_on_the_delivered_stock' does not depend on any axioms
'ChryseneFormalism.Article84.the_sensitive_pair_is_I_and_I' does not depend on any axioms
'ChryseneFormalism.Article84.the_I_I_contact_reads_three_and_two' does not depend on any axioms
'ChryseneFormalism.Article84.the_D_D_contact_reads_two_either_way' does not depend on any axioms
'ChryseneFormalism.Article84.the_T_T_contact_reads_four_either_way' does not depend on any axioms
'ChryseneFormalism.Article84.the_admitted_fractions_sum_to_one' does not depend on any axioms
'ChryseneFormalism.Article84.f_I_rounds_to_497e_minus_4' does not depend on any axioms
'ChryseneFormalism.Article84.f_I_does_not_round_to_498e_minus_4' does not depend on any axioms
'ChryseneFormalism.Article84.the_II_frequency_is_one_over_2011_squared' does not depend on any axioms
'ChryseneFormalism.Article84.the_II_frequency_rounds_to_25e_minus_7' does not depend on any axioms
'ChryseneFormalism.Article84.the_II_frequency_does_not_round_to_24e_minus_7' does not depend on any axioms
'ChryseneFormalism.Article84.the_sensitive_contact_is_rarer_than_one_in_four_million' does not depend on any axioms
'ChryseneFormalism.Article84.a_blind_stock_reads_alike_in_both_alignments' does not depend on any axioms
'ChryseneFormalism.Article84.the_heritable_field_has_two_to_the_fiftyfour_values' does not depend on
```

```
    any axioms
'ChryseneFormalism.Article84.
    two_to_the_minus_fiftyfour_rounds_to_555e_minus_17' does not depend on any
     axioms
'ChryseneFormalism.Article84.two_to_the_fiftyfour_has_seventeen_digits' does
    not depend on any axioms
'ChryseneFormalism.Article84.twelve_letters_in_the_delivered_alphabet' does
    not depend on any axioms
'ChryseneFormalism.Article84.the_configuration_set_of_four_sites' does not
    depend on any axioms
'ChryseneFormalism.Article84.the_draw_law_is_normalised' does not depend on
    any axioms
'ChryseneFormalism.Article84.forbidding_one_registry_value_cuts_the_support'
    does not depend on any axioms
'ChryseneFormalism.Article84.the_cut_support_is_a_proper_subset' does not
    depend on any axioms
'ChryseneFormalism.Article84.the_least_word_mass_rounds_to_239e_minus_16' does
     not depend on any axioms
'ChryseneFormalism.Article84.the_greatest_word_mass_rounds_to_382e_minus_3'
    does not depend on any axioms
'ChryseneFormalism.Article84.their_ratio_is_exactly_two_thousand_to_the_fourth
    ' does not depend on any axioms
'ChryseneFormalism.Article84.the_ratio_rounds_to_160e13' does not depend on
    any axioms
'ChryseneFormalism.Article84.every_class_is_reached' does not depend on any
    axioms
'ChryseneFormalism.Article84.the_registry_coordinate_is_onto' does not depend
    on any axioms
'ChryseneFormalism.Article84.
    the_agreement_alphabet_of_six_sites_has_thirtytwo_letters' does not depend
     on any axioms
'ChryseneFormalism.Article84.five_bits_at_the_codon_length' does not depend on
     any axioms
'ChryseneFormalism.Article84.fiftythree_bits_at_the_heritable_minimum' does
    not depend on any axioms
'ChryseneFormalism.Article84.the_class_cost_is_four_or_five' does not depend
    on any axioms
'ChryseneFormalism.Article84.the_class_cost_times_its_orbit_fills_the_word_set
    ' does not depend on any axioms
'ChryseneFormalism.Article84.the_edge_computation_agrees' does not depend on
    any axioms
'ChryseneFormalism.Article84.the_edge_orbit_fills_the_agreement_alphabet' does
     not depend on any axioms
'ChryseneFormalism.Article84.the_probability_at_rho_one_hundred' does not
    depend on any axioms
'ChryseneFormalism.Article84.the_probability_at_rho_two_hundred' does not
    depend on any axioms
'ChryseneFormalism.Article84.the_first_probability_rounds_to_936e_minus_7'
    does not depend on any axioms
'ChryseneFormalism.Article84.the_second_probability_rounds_to_121e_minus_7'
    does not depend on any axioms
'ChryseneFormalism.Article84.
    the_species_term_at_rho_one_hundred_is_twenty_bits' does not depend on any
     axioms
'ChryseneFormalism.Article84.it_is_not_nineteen_point_nine' does not depend on
     any axioms
'ChryseneFormalism.Article84.it_is_not_twenty_point_one' does not depend on
    any axioms
'ChryseneFormalism.Article84.
```

```
    the_species_term_at_rho_two_hundred_is_twentythree_bits' does not depend
    on any axioms
'ChryseneFormalism.Article84.it_is_not_twentytwo_point_nine' does not depend
    on any axioms
'ChryseneFormalism.Article84.it_is_not_twentythree_point_one' does not depend
    on any axioms
'ChryseneFormalism.Article84.the_total_at_rho_one_hundred_is_twentyfive_bits'
    does not depend on any axioms
'ChryseneFormalism.Article84.the_total_at_rho_two_hundred_is_twentyeight_bits'
     does not depend on any axioms
'ChryseneFormalism.Article84.the_share_at_rho_one_hundred_is_twenty_per_cent'
    does not depend on any axioms
'ChryseneFormalism.Article84.the_share_at_rho_one_hundred_is_not_twentyone'
    does not depend on any axioms
'ChryseneFormalism.Article84.the_share_at_rho_two_hundred_is_eighteen_per_cent
    ' does not depend on any axioms
'ChryseneFormalism.Article84.the_share_at_rho_two_hundred_is_not_nineteen'
    does not depend on any axioms
'ChryseneFormalism.Article84.the_share_falls_as_the_species_term_grows' does
    not depend on any axioms
'ChryseneFormalism.Article84.the_heritable_minimum_is_fiftyfour' does not
    depend on any axioms
'ChryseneFormalism.Article84.the_least_complete_box_is_twentyseven' does not
    depend on any axioms
'ChryseneFormalism.Article84.the_lateral_adjacencies_number_seventytwo' does
    not depend on any axioms
'ChryseneFormalism.Article84.the_stacking_relations_number_fortyfive' does not
     depend on any axioms
'ChryseneFormalism.Article84.
    the_relations_of_a_box_number_one_hundred_and_seventeen' does not depend
    on any axioms
'ChryseneFormalism.Article84.
    a_layer_has_four_corners_four_edges_and_one_centre' does not depend on any
     axioms
'ChryseneFormalism.Article84.the_degrees_exhaust_the_layer' does not depend on
     any axioms
'ChryseneFormalism.Article84.a_corner_admits_twelve_letters' does not depend
    on any axioms
'ChryseneFormalism.Article84.an_edge_site_admits_eight' does not depend on any
     axioms
'ChryseneFormalism.Article84.the_centre_admits_four' does not depend on any
    axioms
'ChryseneFormalism.Article84.the_per_site_counts_over_the_layer' does not
    depend on any axioms
'ChryseneFormalism.Article84.the_unconstrained_layer' does not depend on any
    axioms
'ChryseneFormalism.Article84.the_realisable_layer' does not depend on any
    axioms
'ChryseneFormalism.Article84.the_layer_cut_is_exactly_243_over_16' does not
    depend on any axioms
'ChryseneFormalism.Article84.the_layer_cut_rounds_to_152_tenths' does not
    depend on any axioms
'ChryseneFormalism.Article84.the_six_layer_cut_is_243_over_16_to_the_sixth'
    does not depend on any axioms
'ChryseneFormalism.Article84.the_six_layer_cut_rounds_to_12e7' does not depend
     on any axioms
'ChryseneFormalism.Article84.the_realisable_count_has_fiftytwo_digits' does
    not depend on any axioms
'ChryseneFormalism.Article84.the_unconstrained_count_has_fiftynine_digits'
```

```
   does not depend on any axioms
'ChryseneFormalism.Article84.the_two_counts_differ_by_seven_digits' does not
   depend on any axioms
'ChryseneFormalism.Article84.four_to_the_sixteen_has_ten_digits' does not
   depend on any axioms
'ChryseneFormalism.Article84.four_to_the_twentyseven_has_seventeen_digits'
   does not depend on any axioms
'ChryseneFormalism.Article84.four_to_the_fiftyfour_has_thirtythree_digits'
   does not depend on any axioms

exit code 0, elapsed 77.2 s
```

Kernel Axiom Audit

Axiom Inventory

Running `#print axioms` on each of the 171 theorems in `acf-84-proof.lean` yields:

Result	Meaning	Count
no axioms	The proof term reduces to primitive inference steps alone. Nothing is assumed beyond the type theory.	**171**
`propext`	Not used.	0
`Quot.sound`	Not used.	0
`funext`	Not used.	0
`Classical.choice`	Not used.	0
Extra axioms introduced by this proof:		**none**

The audit is the file's last section, so the 171 lines above are the run's own output rather than a separate measurement. Four of them were not clean in the first draft, for the reason the *Encoding* section records: a bounded existential decided through `Nat.decidableExistsLT` carries `propext` and `Quot.sound`. Rewriting those four claims over a Boolean search removed both, and the fact that the audit caught them is the argument for running it inside the file.

The proof files of the corpus, and what their counts mean

Article	Kind	Checks	Theorems	Axiom-free
79	formal	88	55	54
80	formal	99	61	61
81	matching	91	55	55
82	concordance	120	109	109
84	formal	206	171	**171**

The first four rows are the figures the Article 82 proof dossier reports, carried here unchanged; they were measured there and have not been re-measured for this file. The last row was measured here, by the audit above.

The theorem count of this file is the largest of the five and should not be read as the strongest result. The same caution the Article 82 dossier entered against its own count applies here with a different cause. Article 82's count is large because a concordance paper prints a great many figures and the arithmetic behind each is shallow. This file's count is large because the article is combinatorial: an orbit count, a blindness count and a registration number are each cheap to state and cheap to decide, and a paper that enumerates will accumulate certificates faster than a paper that proves. One theorem here — `one_blind_unit_suffices`, the 13 689-case exhaustion of S7 — carries more of the article than twenty of the digit brackets together, and the count does not say so.

Verification Note

The proof file makes no use of `native_decide`, which bypasses the kernel by running compiled code; of `sorry`; of `Classical.choice` or `Classical.em`; or of any Mathlib import. The `decide` tactic in Lean 4 core reduces a goal to a kernel-evaluated Boolean expression, and the kernel itself performs the evaluation.

Three `set_option` lines appear at the head of the file and none of them weakens anything. `exponentiation.threshold` raises the point at which the *elaborator* declines to pre-evaluate a power; past it Lean emits a warning, not an error, and the kernel evaluates the power either way. The bracket for the species term at $\rho^* = 200$ raises an exponent to 460, well past the default of 256, so without the raise the run is noisy rather than wrong. `maxRecDepth` is raised because the default refuses the deeper evaluations outright, and `maxHeartbeats` because the elaborator's default budget is smaller than the 13 689-case search of S7. All three are budgets, not assumptions, and the audit above is the evidence.

To reproduce: install Lean 4.12.0 and run `lean acf-84-proof.lean`. It takes about 77 seconds, about fifty of them in S7, prints 171 audit lines and exits 0. There are no dependencies beyond the Lean core, so no `lakefile` and no package fetch are required.

Scope: three limits, stated plainly

A kernel-checked theorem certifies that a conclusion follows from the definitions in the file. For this article that is a good deal, because the definitions are the corpus's own and there is no measured datum standing between them and the conclusions. Three limits belong with it anyway.

First, the hypotheses are not theorems. The six named hypotheses of Article 84 are premises, and H_{frame} and H_{place} are the two that carry the paper's positive results. S9 through S13 certify what a prescribed alignment field costs; nothing here certifies that an agent can prescribe one, and the article does not claim it can. The blindness theorem of S6 and S7, read backwards, says that any alignment sensor must couple to something other than the arm-registration pattern, because that pattern is provably degenerate. That is a constraint on sensors. It is not a sensor.

Second, reachability is not injectivity. S10 certifies that every one of the twenty frame classes is reached from the configuration set. It does not certify that distinct classes have distinct products, and Article 57's capacity theorem expressly withholds that. A reader who takes the twenty classes for twenty distinguishable products has imported an injectivity no file in this corpus asserts.

Third, both artifacts encode the article's conclusions. The harness fails if contact blindness does not hold and if the reachable sets are not equal; this file's S7 and S10 do the same. That is deliberate and the article's appendix states it: a published verification artifact should stop passing when the claims it supports stop holding. But it makes both files regression tests on the conclusions rather than independent tests of them, and a reader re-deriving a theorem under weaker hypotheses should expect to edit the certificate as well as the proof.