Structural Induction for the Collatz Conjecture: Established Results, Residual Domains, and Open Problems

Y.Ueoka
Independent Researcher, Corresponding Author

Tsumugi (ChatGPT)
AI Co-researcher

29 November 2026 — Preprint, version 1.0

Abstract

We synthesize the ABN/CPE programme for the accelerated Collatz map, distinguishing common-rank certificates from descent certificates for other scores. We record the final residual outer envelope, including its unbounded mixed-parity sector, with explicit separation of analytic transfer arguments and historical finite premises. Two results from the last short trial are reconstructed: decidability of coverage by a prescribed finite collection of fixed-exponent coalescence rules on a suitable counting language, and a uniform-time obstruction on the high-count residual outer envelope, without a bound on inverse exponents. The latter family already admits vacancy descent. We also reconstruct an earlier finite-history obstruction to forward descent under the first component of the common score. Neither an obstruction on the exact residual nor the Collatz conjecture itself is established. This is a synthesis of established results and explicitly conditional historical records, not a new convergence claim.

1 Purpose and relation to the five papers

The signed-power alternating binary notation (ABN) and its Collatz phase expression (CPE) run description provide structural observables distinct from ordinary binary digits. The objective of the programme was universal finite-time reduction by a well-founded order, not necessarily a decrease in integer value. Papers A–E [1, 2, 3, 4, 5] organize 116 numbered results (including reused or classical facts, not 116 independent novelty claims). Paper A develops the representation and exact updates; B establishes finite-time descent and finite compression; C constructs witness-preserving scores and induction; D studies powers of three, distributions and transport; E treats lifted trajectories, extraction and interfaces. This paper preserves their connections, the last historical coverage ledger, and the scope of the final trial. It does not reopen the search for a complete proof. The mathematical record ends at R271 [6]. Publication identifiers and source maps are maintained in the accompanying archive.

2 Induction and the distinction between scores

On positive odd integers put $F(n) = (3n+1)/2^{s(n)}$, $s(n) = \nu_2(3n+1)$, $S_t = \sum_{j<t} s(F^j(n))$. For ordinary binary digits b_j with $b_{-1} = 0$ and eventual upper zeros, the signed ABN digits are $b_{j-1} - b_j$. Their support, not the ordinary binary word, is the occupied-cell mask $E(n) = n \,\mathrm{XOR}\, (2n)$. It has low-to-high run word $C(r_0)K(d_0)\cdots K(d_{m-1})C(r_m)$. Here all run lengths are positive, $m \geq 0$, and $C(r)$ and $K(d)$ denote occupied and vacant runs of the stated lengths. Write $\mathrm{bitlen}(0) = 0$, $\mathrm{bitlen}(v) = \lfloor \log_2 v \rfloor + 1$ for positive integers, $x_+ = \max(x, 0)$, and $\mathbf{1}_P = 1$ if P holds and 0 otherwise. Set $h = H_C = \sum r_i$, $Z = \sum d_i$, $B = h + Z$, $p_i = r_i \bmod 2$, $\bar{r}_0 = 2 - p_0$, $H = h - r_0 + \bar{r}_0$, $\widehat{B} = H + Z$. The count h is even (binary transitions come in pairs), and H is a positive even integer. Normalize by deleting $r_0 - \bar{r}_0$ lowest occupied cells and shifting down the remaining word, then decoding its unique positive odd integer. In normalized words $r_0 = \bar{r}_0$. The normalization ρ preserves the next odd iterate, vacancy count and higher run data (P6-351–353; Paper B); $s(n) = r_0$. The base $Z = 0$

consists of one even occupied run and has $F(n) = 1$. For a positive odd input, reaching 1 in finitely many accelerated steps is equivalent to ordinary Collatz convergence; positive even inputs reduce to odd ones by division.

Proposition 2.1 (P6-425; vacancy induction)**.** *The Collatz conjecture is equivalent to the assertion that every n with $Z(n) > 0$ has some $t \geq 1$ with $Z(F^t(n)) < Z(n)$.*

Proof. Convergence supplies an endpoint with $Z = 0$. Conversely apply strong induction on the nonnegative integer Z, using the solved base. Finite sublevel sets of Z on unnormalized integers are not required, nor is a uniform time bound. □

Define the final first component $C = 24(Z + p(H))$ with

$$p(H) = \begin{cases} 0 & H \leq 28, \\ 25(H-29)/24 & 30 \leq H \leq 52, \\ 2(H-28) & H \geq 54. \end{cases}$$

For the bounded label $\ell = \ell_{130}$, use the following explicit rule (Paper C, P6-635, 644, 647, 649, 652, 653). Outside the nonbase all-$d_i = 1$ domain it is zero. In that domain let $P = \sum_{i\geq 1}(\lfloor r_i/2 \rfloor - 1)_+$. Apply the table in order, assigning zero if no condition applies.

ℓ	Condition
6	all higher $r_i \in \{1, 2\}$
5,4	$p_0 = 0$, $r_m = 2, 4$ respectively, $P \leq \mathbf{1}_{r_1=1}$
3,2	$p_0 = 0$, $r_m = 3, 1$ respectively, $r_1 = 1$, all higher $r_i \leq 3$
1	$p_0 = 1$, $r_1 \geq 2$, $\max_{i\geq 1} r_i \geq 3$, $P + p_m - 1 + \mathbf{1}_{r_1=2} = 0$, and either $r_1 \in \{2, 5\}$ or $r_m \in \{1, 4\}$

Put $L = 7[Z(Z+1)/2 + m] + \ell$, $W = \Xi_6 = H - 2 + \sum_{j=0}^{5} Z(F^j(n))$, and $\Psi = (C, L, W)$ with lexicographic order. The order of L equals that of (Z, m, ℓ) because $0 \leq m \leq Z$ and $0 \leq \ell \leq 6$.

Proposition 2.2 (P6-1041,1050; common-rank induction, Paper C)**.** *The common order is well-founded with base $Z = 0$. If every nonbase n admits a positive odd z and $T, J \geq 0$ such that $F^T(n) = F^J(z)$ and $\Psi(z) <_{\text{lex}} \Psi(n)$, then the conjecture holds.*

Proof. All three components are nonnegative integers. A finite lexicographic product of well-orders is well-founded: an infinite descending sequence would eventually have constant first, then constant second, then decreasing third component. At the base all components vanish and outside it $C > 0$. The explicit integer realization in Paper C is

$$\Phi = CA(C)D(C) + LD(C) + W, \quad A(c) = 7c(c+3)/2 + 7, \quad D(c) = 7c + 205.$$

Here $A(c)$ is integral since $c(c+3)$ is even for integer c. For the normalized representative $x = \rho(n)$, $F(n) = F(x) \leq (3x+1)/2 \leq 2x$; normalization does not increase width. Thus $\widehat{B}(F(n)) \leq \widehat{B}(n) + 1$. This and $p(H) \geq H - 29$ give $\widehat{B}(n) \leq C/24 + 29$ and $Z(F^j(n)) \leq C/24 + 27 + j$ since $H \geq 2$. Hence $0 \leq W \leq 7C/24 + 204 < D(C)$. As $m \leq Z \leq C$, $0 \leq L < A(C)$. At fixed $c = C$, the term $LD(c) + W$ is in $[0, A(c)D(c))$ and respects the lexicographic order on (L, W). For $c' < c$, $(c'+1)A(c')D(c') \leq cA(c)D(c)$ by positivity and monotonicity. Thus Φ exactly realizes the lexicographic order. Its normalized sublevel sets are finite since C is bounded there and $\widehat{B} \leq C/24 + 29$; this finiteness is not needed for the well-foundedness of the triple. By well-founded induction z converges, so the common endpoint and n converge. Forward strict descent is the case $J = 0$; coalescence need not be forward descent. □

For a fixed order define $\mathcal{C}_\Psi$ as the set of nonbase inputs with a mathematical certificate of the displayed form in the selected established library. Distinguish this from the semantic set of all inputs for which such a witness exists; no algorithm for either full set is claimed. In particular a strict W

witness is automatically preserved only with C, L unchanged (or if an earlier component already decreases). A strict Z witness is not automatically a C witness: its change is $24(\Delta Z + p(H') - p(H))$. A strict L witness is preserved if C is unchanged or decreases. The set of inputs certified for other scores may overlap $\mathcal{C}_\Psi$; it is not a disjoint convergence classification. Normalization preserves these common components but does not identify all numerical integers of the same score.

3 The residual outer envelope at R270–R271

Here we record the historical ledger, conditional on its original finite certificates, not an independent re-certification of every historical finite branch. Every stated exclusion is subject to its original certificate and domain. Let $\mathcal{N}$ denote the normalized positive odd nonbase inputs and let $\mathcal{R}_\Psi = \mathcal{N} \setminus \mathcal{C}_\Psi$. The exact complement $\mathcal{R}_\Psi$ of the entire certified library was not computed. Conditional on the historical finite premises, the union of the sectors below contains $\mathcal{R}_\Psi$; membership in that union alone is not evidence of absence of a witness.

Let $\epsilon_i = p_i \text{ XOR } \mathbf{1}_{i=0}$, $I_i = \mathbf{1}_{d_i=1}$,

$$U = 1 + p_m + \sum_{i<m} [2 + \epsilon_i(1 - 2I_i)], \qquad \delta = 1 + p_m - r_0$$

for normalized words. The historical common-score exclusions include $H \le 10$, $H > \max(55, 3m + 3)$,

$$H > \max(28, 54 + \delta, U + \delta),$$

the special width bounds of P6-930, and the equality-breaking conditions of P6-1042–1044 (see Paper C). The strict lower bound on H involving U and δ is a one-step sufficient condition from P6-571/928; its negation is not a test for failure of every later witness.

Sector	Residual candidate conditions (outer envelope, not exact complement)
A	All r_i even; $12 \le H \le 52$, $3 \le m \le 13$, $H \ge 2(m+1)+2$; $d_0 \le 76$, $d_1 < D'_1(r_1, d_0) \le 896$; for $m = 13$, $d_0 = 1$. For $m = 3$, additionally $d_0 = 1$, $r_1 \in \{20, 22, \ldots, 46\}$, $d_2 < D'_2$.
B1	Some odd r_i; $12 \le H \le 28$, $1 \le m \le H - 1$.
B2	Same parity condition; $30 \le H \le 52$, $1 \le m \le H - 1$.
B3	Same parity condition; $H = 54$, $1 \le m \le 53$; exclude $\delta = -1$, $U \le 54$.
B4	Same parity condition; $H \ge 56$, $\lceil (H-3)/3 \rceil \le m \le H - 1$, $H \le U + \delta$, some $d_i \ge 2$.

B1–B3 also obey $H \le \max(28, 54 + \delta, U + \delta)$; further original exclusions apply. The individual thresholds D'_1, D'_2 are those of P6-1059, not unrestricted functions to be extrapolated. The table gives the sharper historical envelope A; the full threshold tables are not available here. A coarser envelope with explicit membership conditions is obtained by retaining all other A conditions and replacing $d_1 < D'_1$ by $d_1 < 896$, and, for $m = 3$, $d_2 < D'_2$ by $d_2 < 4084$. The containment follows from the historical bounds $D'_1 \le 896$ and $D'_2 \le D_2 \le 4084$ (P6-1057,1059), conditional on those certificates. Within the all-even domain, the one-band, two-band and $m \ge 14$ sectors are recorded as certified for the common rank (P6-1046,1053,930). This $m \ge 14$ exclusion does not extend to arbitrary mixed-parity words. The sharper historical three-band envelope of A was reported to contain 413,237,001 points; R269 removed 16,610,680 points from the preceding envelope of 429,847,681, by 13,993,441 direct checks plus extension arguments (P6-1062). These are historical computer-assisted counts, not an independently reproduced enumeration of the sharper envelope, an enumeration of the coarse envelope, or a count of exact residuals. A with $m \ge 4$ still has potentially unbounded later band lengths. B1–B3 have finitely many occupied run lists but potentially unbounded bands. B4 has neither a uniform occupied count nor band-count bound from the ledger.

For arbitrary parity, the original W certificates cover $m = 1, 2, 3$ within 32,46,103 steps respectively (P6-781,782,802; Paper B); they do not imply the endpoints have $m \le 3$ or the common order decreases. P6-1052 and P6-1055–1059 also exclude suitable lower prefixes with arbitrary upper

words, including mixed parity. The R270 two-step theorem P6-1063 (Paper C) excludes every normalized high-H all-unit-band word, regardless of the number and parity of runs. These statements describe certificates, not all-state convergence.

Proposition 3.1 (R269 local boundary identity)**.** *For normalized nonbase words let* $o = \sum p_i$, $e = \sum_{i=1}^{m}(r_i - 2 + p_i)/2$, $a = \sum_{i<m} \epsilon_i I_i$. *Then*

$$H = 2(m+1) - o + 2e, \qquad U + \delta - H = 2(o - e - a) - 1 - p_0 + p_m.$$

Consequently $H \le U + \delta$ *is equivalent to* $e + a \le o - 1 + \mathbf{1}_{p_0=0, p_m=1}$.

Proof. Use $r_0 = 2 - p_0$, the definition of e, and $\sum_{i<m} \epsilon_i = o - p_m + 1 - 2p_0$. Substitution yields both equalities. All counts are integers; taking the floor of the upper bound for $e + a$ gives the final condition. No orbit assertion follows from the identity alone. □

Call the explicit B4 conditions above $\mathcal{E}_4$; they include the R269 boundary and R270 long-band requirement, but not the subtraction of every earlier certificate. This is the outer envelope used in the following obstruction. The exact residual of the certified library within it is not known.

3.1 Analytic transfer of finite certificates

The late coverage claims rely on two distinct operations: a replacement by a coalescing lower-ranked state and the extension of a forward descent witness through longer K-bands. The first operation uses the lifting theorem P6-1051 (Paper C); the second uses P6-565 (Paper B). Neither requires a decrease of the numerical integer. For clarity write $b(x) = \operatorname{bitlen} E(x)$ and $h(x) = H_C(x)$, reserving $H(x)$ for the normalized count.

Proposition 3.2 (P6-1054/1058; same-time lower replacement)**.** *Let* x, v *be positive odd integers with* $F^t(x) = F^t(v)$, $t \ge 0$, *and cumulative division exponents* S, R *respectively. Assume* $H(v) \le H(x)$ *and* $R - h(v) < S - h(x)$. *Define*

$$D_* = \max(1, S + 1 - b(x), S - R + b(v) + 1 - b(x)).$$

For every positive odd Q *and every integer* $d \ge D_*$, *the input* $n = 2^{b(x)+d}Q + x$ *admits a coalescing state* $z = 2^{b(x)+d-S+R}Q + v$ *with* $F^t(z) = F^t(n)$ *and* $C(z) < C(n)$.

Proof. Put $k = b(x) + d$, $K = k - S + R$. The threshold gives $k > b(x)$, $k > S$, and $K > b(v)$. The upper terms of the two lifted trajectories are respectively $2^{k-S}3^tQ$ and $2^{K-R}3^tQ$ at time t: at each step the upper term is divisible by a higher power of 2 than the exact division exponent of the lower iterate. Hence the endpoints agree. At the initial states the upper and lower masks have disjoint supports separated by a positive gap. This gives

$$Z(z) - Z(n) = R - S + h(x) - h(v), \qquad H(z) - H(n) = H(v) - H(x).$$

The former is strictly negative and the latter nonpositive, so monotonicity of p gives strict C descent in the comparison. The threshold and the difference do not depend on Q. This is a sufficient coalescence certificate, not a forward descent of n. □

If x, v have individually verified first arrival times t_x, t_v at 1, with respective cumulative exponents S_x, S_v, define $a(x) = S_x - 2t_x - h(x)$ and similarly for v. Waiting at the fixed odd point 1 adds two to the exponent per step. Thus $a(v) < a(x)$ and $H(v) \le H(x)$ imply the proposition with $t = \max(t_x, t_v)$ (P6-1054). This argument uses the verified arrivals of the particular lower words, not convergence of arbitrary integers. For P6-1055, $x = 1$, $v = 31$ give $t = 39$, $S = 78$, $R = 67$, $h(x) = h(v) = H(x) = H(v) = 2$, and $b(v) = 6$. The sufficient threshold is $d \ge 77$, with $\Delta C = -264$; the upper word has no parity restriction beyond representing a positive odd integer. This fixed lower trajectory was recalculated in R45. The historical P6-1053 example with lower inputs $x = 11{,}184{,}809$ and $v = 26{,}623$ was also replayed: $T = 175$, $J = 113$, cumulative exponents

$S = 301$ and $R = 194$, threshold 277 and $\Delta C = -225$. This is an *unequal-time* example, not an application of the same-time proposition: with $k = b(x) + 277$ and $Q = 5$, use $n = 2^k Q + x$, $z = 2^{k-S+R}3^{T-J}Q + v$, so the upper parts at the common endpoint are equal. The quoted score difference is the verified value for this specific pair. This does not verify the other 223 certificate rows or the finite complement.

Proposition 3.3 (P6-1061; simultaneous extension of two gaps)**.** *For positive odd x, q, u, put $k = b(x) + d_1$, $l = k + b(q) + d_2$ and $n = x + 2^k q + 2^l u$. Fix $t \geq 1$, let $x_t = F^t(x)$ and let S be its cumulative division exponent. Write $q_t = 3^t q$, $u_t = 3^t u$. If*

$$d_1 \geq \max(1, S + b(x_t) + 1 - b(x)), \qquad d_2 \geq \max(1, b(q_t) + 1 - b(q)),$$

then $F^t(n) = x_t + 2^{k-S} q_t + 2^{l-S} u_t$. The difference $C(F^t(n)) - C(n)$ is independent of both d_1, d_2 on this region.

Proof. The first threshold implies $k > S$, so the same valuation argument as above applies at every intermediate step. The two thresholds separate the three endpoint masks, and $d_1, d_2 \geq 1$ separate the initial masks. Their raw occupied counts add, while only the lowest block contributes the normalization correction. The total endpoint width is $l - S + b(u_t)$ and the initial width is $l + b(u)$. Consequently the difference of first components is

$$\begin{aligned} 24\big[& - S + b(u_t) - b(u) + h(x) + h(q) + h(u) \\ & - h(x_t) - h(q_t) - h(u_t) \\ & + p(H(x_t) + h(q_t) + h(u_t)) - p(H(x) + h(q) + h(u))\big]. \end{aligned}$$

No gap length occurs in this expression. If it is negative, the entire rectangle of longer gaps is certified at the same time. Moreover the thresholds are at most $2t + 1$: $Y_j = 2^{S_j} F^j(x)$ satisfies $Y_{j+1} = 3Y_j + 2^{S_j} \leq 4Y_j$, hence $S + b(x_t) \leq b(x) + 2t$, and $3^t q \leq 4^t q$ gives $b(q_t) \leq b(q) + 2t$. □

In P6-1060/1062 the more conservative P6-565 bound is sufficient: whenever a directly verified forward descent at time t has a band of length at least $2t + 1$, that band's arbitrary extension preserves the entire occupied run list and adds a constant to Z at all corresponding times, hence preserves the C difference. The same holds for simultaneous extension of several such bands. Skipped finite points must lie in an actually certified extension region; the mere existence of an upper bound on a gap does not certify its shorter values.

3.2 Late coverage: conditional dependence on finite evidence

The following is a concordance of original reported coverage, not a new full-domain computational verification. R264–R269 source proofs and available reproduction instructions are retained in the full working archive, not all in the focused review distribution. The original historical SQLite catalogue lists large certificate tables, checking programs and completion logs whose exact bytes are missing from the R44/R45 working archive. Of its 96 entries for R264–R270, 27 match a file in that working archive by SHA256 and 69 do not; these counts were also confirmed in the full R47 archive. The focused R47 review distribution contains exact matches for 9 of the 96 entries (87 absent), omitting a further 18 entries present in the full archive. These are counts of catalogue entries, not distinct files. The original catalogue and the two archive inventories must not be conflated. The analytic transfers above are directly established; the reported finite premises below require the original evidence or independent exhaustive rechecking before they can be presented as reproduced results.

Source	Reported coverage or finite premise	Dependence / evidence status
1047–49	All-even two/three bands with $d_0 \geq 22$; two-band remainder at most 224 lower-word families, $d_1 \geq 267$.	Isolation and P6-565; R264 tables and finite checks unavailable here.
1053	All-even two-band domain; 224 tail certificates and 3,607 finite points.	P6-1048/1049,1051/1052; R265 certificates unavailable.
1055–57	Even bottom, $d_0 \geq 77$; 1,748 two-run prefixes; 1,157,378 three-run prefixes in the specified finite domain.	P6-1054; 1055 fixed orbit checked; R266/267 tables unavailable.
1059	Improved second/third thresholds; 227,122 remaining lower words for third-band tables.	P6-1056–58; retained and improved rows, R268 tables unavailable.
1060	Three all-even bands, $d_0 \neq 1$ or $d_0 = 1, r_1 \leq 8$; 465,409,300 finite envelope points.	Prior boundary and tail certificates, P6-565, 11,612,671 reported direct checks; unavailable completion logs.
1062	Further $d_0 = 1, r_1 = 10, 12, 14, 16, 18$; 16,610,680 finite envelope points.	P6-1059/1060 and 565; 13,993,441 reported direct checks; unavailable completion logs.
1063	All unit bands, $H \geq 56$, one or two forward steps.	Analytic proof in Paper C; original R270 code and output present and replayed exactly in R45.

For the second-band threshold in P6-1056 the original domain is $r_1 \in \{2, 4, \ldots, 46\}$ and $1 \leq d_0 \leq 76$, with normalized bottom $C(2)$. For the third-band threshold in P6-1057 it is additionally $r_1 + r_2 \leq 48$, $1 \leq d_0 \leq 21$, $1 \leq d_1 < D_1(r_1, d_0)$ and r_2 positive even. P6-1059 retains the old witnesses while improving the thresholds and restricting the third-band table to $d_1 < D_1'$. The analytic transfer is independent of the rest of the upper word, but these lower-prefix restrictions must not be dropped. The original bounds $D_1' \leq 896$, $D_2' \leq 4084$ are reported bounds from finite tables, not new analytic bounds derived here.

The count reconciliation is $429,847,681 - 16,610,680 = 413,237,001$. The 71 rectangles from P6-1061 are within the P6-1062 completed sectors and are not added again. Interrupted attempts for $r_1 = 20, 42, 44, 46$ are not included. No exact residual cardinality, unconditional verification of the missing finite premises, or new coverage is claimed. The machine-readable dependency and evidence inventories accompany the full working archive; the R48 evidence inventory distinguishes the full R47 archive from its focused review distribution.

4 Decidability of prescribed coalescence coverage (R271)

Theorem 4.1 (P6-1064, reconstructed)**.** *Fix a finite list of pairs of finite lists of positive integer exponents (a, b) (empty lists allowed), and an effectively specified domain of positive odd binary words recognized by finite-state paths with additive counters and a Presburger acceptance condition. Assume the specified path counters faithfully represent the domain (existential path acceptance with no spurious counter values). It is decidable whether every input in the domain has a lower-Ψ coalescence candidate with one of these pairs of realized exponent lists. If not, a finite counterexample input can be recovered. The statement concerns verification of a given list, not existence of a covering list.*

Proof. Let the list lengths be T, J, cumulative sums S_i, R_i, totals S, R, and $A = \sum_{i<T} 3^{T-1-i} 2^{S_i}$, $D = \sum_{i<J} 3^{J-1-i} 2^{R_i}$. The affine iteration identity (Paper E) forces the unique candidate

$$z = \frac{2^R(3^T n + A) - 2^S D}{2^S 3^J}.$$

The required exponent histories are determined by residues modulo 2^{S+1} and 2^{R+1} respectively: exact division by 2^s transfers congruence modulo 2^{s+k} to congruence modulo 2^k. The extra bit beyond the total exponent ensures that the final quotient is odd, including the empty-history case.

Positive oddness is checked separately. For each affine relation $z = \alpha n + \beta$, a positive candidate satisfies $z \le (\alpha + |\beta|)n$. Use a common constant number P of upper padding zeros so $2^P \ge \max(1, \alpha + |\beta|)$ for the finite list. Low-to-high binary input words have initial and terminal 1, then a delimiter and P zeros. Candidate tracks have the same number of bit slots, allowing upper zeros. The delimiter does not update the bit clock.

A fixed integer equation $un + vz + c = 0$ is checked by the finite carry recursion $c_0 = c$, $c_{i+1} = (c_i + ux_i + vy_i)/2$, requiring integral carries and terminal carry zero. Conversely, if the full integer equation holds, each lower partial sum (including c) is divisible by the corresponding power of two, so all carries are integral. The carries are bounded by $\max(|c|, |u| + |v|)$; telescoping proves equivalence to the integer equation. Combining this with residue and word tests gives an unweighted finite-state relation. Its projection to the input track is regular, so the complement recognizes absence of a valid candidate. This complementation is made *before* adding any score inequalities.

For valid candidates and inputs, stream the binary words of F^j for $0 \le j \le 5$. Each one-step stage uses the finite state $(p, c, h_*) \in \{0,1\}^3$ initially $(0, 1, 0)$; on bit x set $q = x + p + c$, $y = q \bmod 2$, and update $(x, \lfloor q/2 \rfloor, h_* \vee y)$. Suppress leading low-order zero output until the first 1. This computes $3n + 1$ and removes exactly its low zero bits. Cascading five stages streams the iterates, with downstream clocks advanced only when an output bit is emitted. For a positive input, at most two extra upper zeros suffice to finish its carry and previous-bit contribution; further zeros are emitted after the first 1. More explicitly, after the last potentially nonzero input bit, a stage consumes at most two zeros before its state has previous bit and carry both zero. Since its input represents a positive integer, the stage has emitted its first 1 by then, and every subsequent zero is passed downstream. Induction gives at most ten input zeros to finish all five stages and one further zero to finish their masks. Twelve extra input zeros therefore suffice. They are added after the common candidate padding. This is the finite-state construction of A105 (Paper D).

The domain recognizer reads only the canonical input word; it finishes at the delimiter and is frozen during all padding and flushing. The arithmetic relation and validity tests finish after the P padding zeros, while the metric streams additionally receive the twelve zeros. For each stream form the mask bits $x_i \,\mathrm{XOR}\, x_{i-1}$ with previous bit zero initially. Guess its last occupied bit, require all subsequent mask bits zero, and count width only through that bit. The correct accepting end is unique: the guess is permitted only on a 1, exactly once, and no later 1 is accepted. Count width B through the guessed last 1, occupied cells h, initial occupied run r_0, and occupied run starts (subtract one for m). These give $Z = B - h$ and $H = h - r_0 + 2 - p_0$. Stream the actual iterates; normalization is applied only to their scores, not as an intermediate change of the orbit. Store initial parity and finite tests for the bounded label: comparisons of higher run lengths with 1 through 5 and the capped value $\min(P_{\mathrm{ex}}, 2)$ of the label excess $P_{\mathrm{ex}} = \sum_{i \ge 1}(\lfloor r_i/2 \rfloor - 1)_+$ from Section 2. All label conditions follow from these tests; in particular the equality for label 1 can hold only at excess at most one. The mask counts for six streams compute W by an integer linear expression. The piecewise affine C and comparison of (C, Z, m, ℓ, W) are Presburger conditions. No unbounded carry or exponent is stored in finite control for the streamed score computation.

For each prescribed pair, choose either the regular no-candidate branch or the valid candidate branch with $\Psi(z) \ge \Psi(n)$. Since the candidate is unique, these exactly express failure of that rule. A finite synchronized product of these branches and the domain recognizer expresses simultaneous failure. Keep domain acceptance counters and all comparison conditions as a joint Presburger formula on additive transition counts. There is no appeal to complementation of a general weighted projection.

For completeness, a finite directed multigraph of product states can be tested for an accepting path with such counts. Fix initial and accepting vertices s, t, enumerate nonnegative edge multiplicities k_e subject to flow balance $\mathrm{out}(v) - \mathrm{in}(v) = \mathbf{1}_{v=s} - \mathbf{1}_{v=t}$ and connected undirected support containing both endpoints (separately allow the empty path when $s = t$). Connected support is a finite disjunction over edge subsets. These conditions are necessary and sufficient: replicate each edge with its multiplicity and add a closing edge from t to s when distinct. The balanced connected directed graph has an Euler circuit, which when cut at the closing edge gives a path with the re-

quired counts. When $s = t$ use the circuit directly. All counters are linear in k_e, so existence with the acceptance and failure constraints is a Presburger sentence. Its satisfiability is decidable, and a satisfying count vector and Euler path recover a counterexample. Unsatisfiability certifies the prescribed coverage. □

The coarse B4 envelope is definable in this manner: all run-length comparisons, parities, existence of a long band and the additive counts in Section 3 use finite control and linear constraints. This does not establish that the entire historical certified library can be subtracted within this language. The complete product generator and Presburger certificate engine have not been implemented.

5 A uniform-time obstruction on the outer envelope

Theorem 5.1 (P6-1065, reconstructed)**.** *For any nonnegative integers T_*, J_* there exists a normalized $n \in \mathcal{E}_4$ such that for every positive odd z, $0 \le T \le T_*$, $0 \le J \le J_*$,*

$$F^T(n) = F^J(z) \quad \Longrightarrow \quad \Psi(z) \ge_{\text{lex}} \Psi(n).$$

There is no bound on inverse exponents. Thus no finite list of fixed-exponent rules directly covers the entire explicit outer envelope, even if replaced by arbitrary exponent lists of uniformly bounded lengths.

Proof. Let $N = \max(T_*, J_*)$. Choose $L_0 \ge 5$ with $3^{J_*} \mid b = 2^{L_0} - 1$ and $b > 3^{2N}$; powers of 2 modulo 3^{J_*} are periodic (when $J_* = 0$ the divisibility condition is automatic). For all indicated T, J, $q = 3^{T-J}b$ is a positive odd integer. If $T \ne J$ then q is not Mersenne: when $T > J$, equality $3^{T-J}b = 2^u - 1$ would have $u \ge L_0$ and force $3^{T-J} \equiv 1 \pmod{2^{L_0}}$, impossible by the size bound. When $T < J$, $b = 3^{J-T}(2^u - 1)$ implies $u < L_0$ and $2^u > 3^{J-T}$ by the size bound; reduction modulo 2^u is again impossible. Thus $h(q) \ge 4$ for unequal times, whereas $h(b) = 2$ (two mask cells exactly characterizes a positive odd Mersenne number).

Choose $K \ge T_* + 4$, $g \ge \max_{T,J} \text{bitlen}(q) + 4$, and define

$$V = \sum_{i=0}^{M-1} 2^{ig}, \qquad n = 2^K bV - 1,$$

where $M \ge 27$ will be increased below. For $t \le T_*$, $F^t(n) = 2^{K-t}3^t bV - 1$ by repeated exponent-one steps. Suppose $F^J(z) = F^T(n)$, and denote the actual inverse-side exponent list in forward order by $a_0, \ldots, a_{J-1}$, total R. Applying the inverse maps to -1 defines $-v$ with $1 \le v \le 2^{R-J}$, since each positive auxiliary map $v \mapsto (2^a v + 1)/3$ maps $v \ge 1$ into $[1, 2^{a-1}v]$. The affine inverse composition gives

$$z = 2^{k'} qV - v, \qquad k' = K + R - T \ge R - J + 4.$$

The first term is even and integral; hence a valid positive odd z forces v to be a positive odd integer. The spacing inequality separates the low mask from the upper structure even for arbitrarily large inverse exponents.

Complementing the lower k' ordinary bits of $v - 1$, and writing the upper ordinary block as $qV - 1$, gives the exact mask identities

$$E(z) = E(v-1) + 1 + 2^{k'}(E(qV-1) + 1),$$
$$E(qV-1) + 1 = E(q-1) + 1 + \sum_{i=1}^{M-1} 2^{ig} E(q).$$

In each sum supports are disjoint, by $k' \ge R - J + 4$ and the choice of g. The boundary terms 1 account for the complemented low block and the upper interface; $v - 1$ and $qV - 1$ are even. The uppermost occupied bit gives $B(z) = k' + (M-1)g + \text{bitlen}(q) + 1$. The normalized count

and width are obtained by deleting the same number of lowest occupied cells. For $J \geq 1$, write $\gamma = a_0 - (2 - (a_0 \bmod 2))$ for the removed bottom cells of z. Its first occupied run has length a_0 and lies in the low mask, since bitlen $E(v-1) < k'$ and there is a vacant cell before the upper mask, so the number of occupied bits of that low mask is at least a_0; in particular popcount$(E(v-1)+1) - \gamma \geq 1$. Also $R - \gamma \geq J$. For $J = 0$, $v = 1, R = 0, \gamma = 0$, and both inequalities remain valid.

At the initial state $E(b-1)+1 = 2^{L_0} + 3$, $E(b) = 2^{L_0} + 1$. Thus $H(n) = 2M + 2$, $\widehat{B}(n) = K + (M-1)g + L_0 + 1$. Its lowest run is one, and its word consists of this low cell, a first upper block with runs 2 and 1, and $M - 1$ upper blocks each with two runs of length one. The initial gap is $K - 1$, the first internal gap is $L_0 - 2$, subsequent within-block gaps are $L_0 - 1$, and interblock gaps are $g - L_0 - 1$. Each is at least three. Hence $m = o = 2M$, $p_0 = p_m = 1$, $e = a = 0$. Here $H = 2M + 2 \geq 56$, $3m \geq H - 3$, $m \leq H - 1$, and $e + a = 0 \leq o - 1$; the local identity implies $H \leq U + \delta$. Thus $n \in \mathcal{E}_4$.

If $T \neq J$, each of the last $M - 1$ blocks has at least four cells, and the first upper block and normalized low block each have at least one cell. Hence $H(z) - H(n) \geq 2M - 4$. The width difference satisfies

$$\widehat{B}(z) - \widehat{B}(n) = R - T - \gamma + \text{bitlen}(q) - L_0 \geq J - T + \text{bitlen}(q) - L_0.$$

Let $D_* = \max(\{0\} \cup \{T - J + L_0 - \text{bitlen}(q)\}_{T,J})$. Take $M \geq 27$ with $2M - 4 > D_*$. Both occupied counts are in the high branch where $C/24 = \widehat{B} + H - 56$, so $C(z) > C(n)$ uniformly in the inverse exponents.

If $T = J = 0$, then $z = n$. If $T = J \geq 1$, $q = b$ and the upper mask is merely translated. Consequently $H(z) - H(n) = \text{popcount}(E(v-1)+1) - \gamma - 1 \geq 0$ and $\widehat{B}(z) - \widehat{B}(n) = R - T - \gamma \geq 0$. A strict inequality gives $C(z) > C(n)$. At equality of both differences, a_0 is odd and all remaining exponents are one: indeed $R - T - \gamma = 1 - (a_0 \bmod 2) + \sum_{i=1}^{J-1}(a_i - 1)$. The inverse map for a specified exponent is single-valued. Inverting the last $J - 1$ common steps yields $F(z) = F(n)$; the one-step coalescence classification (P6-353) gives $z = 4^j n + (4^j - 1)/3$, $j = (a_0 - 1)/2 \geq 0$. This extends only the bottom run by $2j$ cells, preserving normalization and the entire forward tail. Thus $\Psi(z) = \Psi(n)$. All possibilities are exhausted. □

The family has every higher C-run of length one or two and all K-lengths at least three. Therefore the unbounded short-C corollary P6-577 (Paper B) supplies a strict Z witness within two steps for each member. This is not a proof that the family lacks a common-rank witness at longer times, or that it survives subtraction of all previously certified common-rank inputs. In particular neither this result nor the finite-history obstruction P6-982 shows a fixed orbit without descent forever. The result rules out a uniform bound over this explicit envelope; quantifiers must not be interchanged.

5.1 Earlier forward obstruction with arbitrary prescribed histories

The R242 result P6-982 predates R271. Unlike the preceding theorem, it allows arbitrary forward exponent words but addresses only strict increase of the first score at the forward iterates, not comparison with all coalescing states. It does not require membership in $\mathcal{E}_4$.

Theorem 5.2 (P6-982, reconstructed)**.** *Given positive integers $s_0, \ldots, s_T$ with $T \geq 1$, there exists a positive odd integer n such that $s(F^t(n)) = s_t$ for $0 \leq t \leq T$ and $C(F^t(n)) > C(n)$ for $1 \leq t \leq T$.*

Proof. Write $S_t = \sum_{j<t} s_j$. First choose a positive odd x realizing the prescribed history. Starting with 1 at time $T + 1$, apply the rational inverse maps $y \mapsto (2^{s_j} y - 1)/3$ in reverse order. Their outputs are odd rational numbers with odd denominators and thus define odd residue classes modulo powers of two. Take a positive odd representative x of the initial class modulo $2^{S_{T+1}+1}$. At each step, congruence to the rational state modulo 2^{s_j+k} implies exact exponent s_j and congruence of the divided output modulo 2^k for $k \geq 1$. This proves realization of the entire list. Put $x_t = F^t(x)$ and $e_t = s_t - (2 - (s_t \bmod 2))$.

Choose $\lambda > \text{bitlen}(3^T) + 1$ and $b = 2^\lambda - 1$. For $0 \leq t \leq T$, let $u = 3^t - 1$, an even nonnegative integer. The identity

$$3^t b = 2^\lambda u + (2^\lambda - 1 - u)$$

expresses the lower λ ordinary binary bits as the complement of u and the upper block as u. The width condition guarantees a zero upper bit of u within the lower block. The lower complement contributes $E(u)+1$ and the upper block contributes $2^\lambda(E(u)+1)$: their boundary bits are 1 because u is even and its highest possible bit lies strictly below position $\lambda-1$. The supports are separated at the λ boundary. Thus $h(3^t b) = 2 + 2h(3^t - 1)$ (with $h(0) = 0$), which is 2 at $t = 0$ and at least 6 for $t \geq 1$.

Choose $K > S_{T+1}+1$ and $K > \mathrm{bitlen}(2^{S_t}x_t)+1$ for all $0 \leq t \leq T$. Choose $g > \lambda+\mathrm{bitlen}(3^T)+1$. For an integer $M \geq 27$ set

$$n = x + \sum_{i=0}^{M-1} 2^{K+ig} b.$$

The added term is divisible by $2^{S_{T+1}+1}$, so the history is unchanged. The affine iteration identity yields $F^t(n) = x_t + \sum_i 2^{K+ig-S_t} 3^t b$. The choices of K, g ensure disjoint occupied masks with intervening zeros at every such time. Hence

$$H_t = h(x_t) + Mh(3^t b) - e_t, \qquad H_0 = h(x) + 2M - e_0.$$

All $H_t \geq 54$ since $h(x_t) - e_t$ is a normalized positive odd occupied count. Also $2^{S_t}F^t(n) \geq n$, giving $B(F^t(n)) - B(n) \geq -S_t$. Subtracting normalization corrections from the widths yields $\widehat{B}(F^t(n)) - \widehat{B}(n) \geq -S_t + e_0 - e_t$. In the high branch $C/24 = \widehat{B} + H - 56$, so for every $1 \leq t \leq T$,

$$\frac{C(F^t(n)) - C(n)}{24} \geq 4M - S_t - h(x) - 2 \max_{0 \leq j \leq T} e_j.$$

The right hand side is positive simultaneously for all t by taking M sufficiently large. This is the original R242 construction with its score rescaled by 24. □

Both obstruction theorems have a finite time horizon chosen before the initial input. Neither supplies a single input with strict non-descent for all times. The upper isolated blocks used in P6-982 are initial resources; the theorem does not establish their indefinite regeneration. It also makes no claim about survival of earlier certificates.

6 Reproduction, source status, and open questions

The R271 finite test was replayed in R44 using its original Python code. The JSON was byte-identical: 52,224 stream-state checks, 85 instances of the P6-1055 affine coalescence, and seven groups of sampled inverse histories for the obstruction (1,314 valid and 6,456 invalid candidates, counted per group). These tests neither implement the general decision procedure nor replace the proofs. The exact group counts remain in the JSON. R45 also replayed the retained R270 verification exactly (35,731 checks); the large R264–R269 finite evidence is unavailable in this working archive. The analytic transfer propositions in Section 3 are reconstructed from their original proofs.

Future mathematical questions include preserving old witnesses across distinct scores; proving reductions with structurally dependent unbounded coalescence times; certifying the exact residual relative to the chosen library; and finding a global mechanism for mixed long/short bands. These are open tasks, not assertions that a strategy must succeed. R271 added no new certified descent coverage. No further proof search is undertaken here.

Source concordance and attribution. Section 2: P6-425,1041,1050 (Paper C also contains the common-rank realization). Section 3: R269-GLOBAL-SCOPE and R270 ledger, P6-1053,1059–1063 and the cited prior certificates. Section 3 transfer propositions: P6-1054/1058 (R266/R268), P6-1061 (R269), with original derivations by ChatGPT and English reconstruction by Tsumugi (ChatGPT). Sections 4–5: R271 original addendum, pages 634–638 of the 638-page research note, P6-1064/1065; Section 5.1: R242, P6-982, pages 557–558 of the same note, original construction and proof by ChatGPT, building on the R215 finite-history construction by Astra and the isolation

programme initiated by Y.Ueoka. The original research ideas of ABN/CPE, isolation, structural scores and gluing are credited to Y.Ueoka; the concrete R271 proposal and original proofs and tests were recorded under Tsumugi (ChatGPT). The present English reconstruction and replay are by Tsumugi (ChatGPT). The entire prior archive is preserved, including corrections, abandoned approaches and original attribution. The R47 manuscript received a separate-context ChatGPT logical review (Minor Revision); R48 addresses its three required clarifications without changing the eight numbered statements. This is not human or organizational peer review, nor a certification of the missing historical finite premises. No comprehensive novelty claim is made.

Reading guide to the archived programme

Paper	Subject and stable archive location	Results
A	Canonical representations and exact structural dynamics; `history/A_R18_frozen/`	10
B	Finite-time structural descent, two-step vacancy balances, finite compression and window scores; `history/B_R24_frozen/`	20
C	Witness-preserving gluing and variational induction; `history/C_R30_frozen/`	21
D	Powers of three and structural transport; `history/D_R36_frozen/`	31
E	Coupled dynamics, lifted trajectories and exact branching laws; `history/E_R43_frozen/`	34

The frozen papers each contain their source concordance. In the full working archive, the final research note is `short_trial/R271/MAIN_NOTE.tex` and its 638-page PDF, together with the full theorem index. The full note, index and R271 status record were not included in the focused R47 review distribution; the listed paths are locations in the full archive, not promises about a standalone PDF or every supplement. The index records references and occurrences, not automatic proof certifications. The R46 editorial audit and source map distinguish the three obstruction/decision results, with their precise quantifiers and source dependencies. The final ledger is the R270 status record under `references/`, supplemented by `short_trial/R271/RESEARCH_STATUS.md`. The R271 addendum, original verification script, JSON, and the R44 replay are archived there and under `paper_F_sources/`. The R45 dependency inventory records the unavailable late finite certificates; their original reported conclusions are distinguished from reproduced checks. The accompanying publication index records the frozen local files and reported identifiers, not verified copies of the remotely published files. The R47 review report, independent finite checks and R48 response are retained in the full working archive. The reviewer-side finite checks were replayed byte-for-byte in R48; they do not replace the analytic proofs or missing finite certificates. No new proof search was undertaken. The references below identify the local frozen versions; published identifiers are recorded as previously reported, not independently verified remote copies. No comprehensive literature or priority review is claimed here.

References

[1] Y. Ueoka and Tsumugi (ChatGPT), *Alternating Binary Notation and Collatz Phase Expression: Canonical Representations and Exact Structural Dynamics*, Paper A, R18, v1.0.1 (2026). Reported identifier: aixiv.261128.000003.

[2] Same authors, *Finite-Time Structural Descent in the Accelerated Collatz Map: Two-Step Vacancy Balances, Finite Compression, and Window Scores*, Paper B, R24, v1.0 (2026). Reported identifier: aixiv.261128.000004.

[3] Same authors, *Witness-Preserving Gluing and Variational Induction in the Collatz Problem*, Paper C, R30, v1.0 (2026). Reported identifier: aixiv.261128.000005.

[4] Same authors, *Powers of Three and Structural Transport in the Collatz Map: Exact Window Distributions and Boundary Estimates*, Paper D, R36, v1.0 (2026). Reported identifier: aixiv.261128.000006.

[5] Same authors, *Coupled Collatz Dynamics in Alternating Binary Notation: Lifted Trajectories and Exact Branching Laws*, Paper E, R43, v1.0 candidate (2026); publication unconfirmed.

[6] ABN/CPE collaborative research record, Phase 6, R271 (29 November 2026), archived 638-page research note and theorem index; in particular R242, R269–R271. Local primary record, not an independently reviewed publication.

A Reproducible finite checks

This appendix provides the complete source of the five available check scripts and embeds the unmodified scripts and their saved stdout files in the PDF. It follows the reproducibility format of Paper E. Run each separately with Python 3.13.5 (standard library only), with assertions enabled (without `-O`); for example, `python R271_VERIFY.py`. Wrapped listing lines are continuations of a source line. R49 replayed all five scripts and compared their stdout byte for byte with the saved results. The original R270 and R271 checks, R45 and R46 editorial checks, and the separate-context R47 reviewer checks retain their respective authorship. No new independent review of this final edition is claimed.

Script	Scope of checks
R270	35,731 checks of the local boundary identity and recorded descent branches.
R271	52,224 stream-state checks; 85 affine coalescences; seven bounded samples of inverse histories (1,314 valid and 6,456 invalid candidates).
R45	Two historical fixed coalescence examples and arithmetic of reported coverage counts.
R46	Six prescribed-history regression examples of P6-982.
R47 reviewer	6,000 normalization/rank inputs, 5,988 boundary identity inputs, 1,200 two-gap comparisons and seven obstruction-family vacancy descent checks.

These are finite checks, not replacements for the analytic proofs. In particular the R271 script does not implement the general decision procedure in Theorem 4.1, and the R45 count arithmetic does not re-certify the reported finite covers. The unavailable historical R264–R269 evidence described in Section 3 remains unavailable; the scripts do not recover it or determine the exact residual. No additional proof exploration or descent coverage was undertaken in R49.

File fingerprints. The local reproducibility directory `reproducibility_F/` in the companion archive contains these files, a README and a SHA-256 manifest. The archive is not asserted to be publicly deposited. The fingerprints of the embedded files are:

File	SHA-256
R270_VERIFY.py	d05b3d8cd058f37bbb538b44dbba7f342c1bee4d28c63f1e1fa1b093959e69c0
R270_REPLAY_R45.txt	4b5d92acccdd987745a094927398753ec48179f9c1118b2dbe3537ca756a1a61
R271_VERIFY.py	22a84352e22e801f5ba0d914c7b343c4d73bedc34312dd2a6d131746d297bc80
VERIFICATION_R271.json	4ae2d5b0c830ebe04c4ec0e9fa102992345076f98944f5dab2309c1804fdba7c
verify_r45.py	a430ee43b0e45103199967c6b04596554f8b6a4bc1918684950ff29d84efef85
verify_r45.json	9f11accfb039c8f2b5097dc746c858307e945d7f185aced6d5e7ce8dcf7e7d87
verify_r46.py	58eb0baba1fa2c5035b6838211ce9e06c7f3a60762fb813d4686e387c97ca1a4
verify_r46.json	f9681ca869ea7f3f6eb033e8f97002eb55926364014de0ade2a2b0cc0108fdbf
independent_checks.py	0aa2ca41a4a0abdfd68e5e1924235d001aae1fea37106021e2be6458c33d98ed
independent_checks.json	66b53dd5f54c8e6d37f4b24a0cb76d6dcd9393bc23abaa24845b21f86f4a0653

R270_VERIFY.py

```
from random import Random
from collections import Counter
from itertools import product,groupby
import json

def F(n):
    v=3*n+1
    return v//(v&-v)
def state(n):
    a=[len(list(g)) for _,g in groupby(bin(n^(n<<1))[2:][::-1])]
    r,d=a[::2],a[1::2]
    H=sum(r)-r[0]+(2-r[0]%2); Z=sum(d); m=len(d)
    p=0 if H<=28 else 25*(H-29) if H<=52 else 48*(H-28)
    return 24*Z+p,H,Z,m,r,d

```

```
def num(r):
    j=0;n=0;sgn=-1
    for i,a in enumerate(r):
        for _ in range(a):
            n+=sgn*(1<<j);j+=1;sgn=-sgn
        if i<len(r)-1:j+=1
    assert n>0 and n%2
    return n

def check(r,stats):
    if sum(r)%2:return
    n=num(r); a=state(n);assert a[4]==r
    if a[1]<56:return
    b=state(F(n));c=state(F(F(n)))
    # use only first component or R1042, not any unknown label
    def cert(x):return x[0]<a[0] or x[0]==a[0] and x[1]<=a[1] and x[2]==a[2] and x[3]<a[3]
    assert cert(b) or cert(c),(r,a,b,c)
    stats['first_C' if b[0]<a[0] else 'first_band' if cert(b) else 'second_C' if c[0]<a[0] else
    'second_band']+=1
    # independently check analytic H and width identity for all d=1
    p0,pm=r[0]%2,r[-1]%2;e=sum((x-(2-x%2))//2 for x in r[1:]);H=a[1]
    if b[2]>0:
        corr=b[4][0]-(2-b[4][0]%2)
        assert b[1]-H==2*(p0+pm-1-e)-corr
        assert b[1]+b[2]-H-a[2]==p0+pm-1-corr
    if p0==pm==1 and r[1]>=2 and b[2]>0 and c[2]>0:
        aa=sum(x%2 and x>=3 for x in r[1:]);I=int(r[1]==2)
        assert sum(c[4])-H==2*(aa-e-I),(r,a,b,c)
        assert c[1]+c[2]<=H+a[2]
    stats['checked']+=1

def main():
    rng=Random(270); stats=Counter()
    for i in range(20000):
        m=rng.randint(20,150);r=[rng.choice([1,2]) for _ in range(m+1)];r[-1]+=sum(r)%2;check(r,
    stats)
        for _ in range(i%3):r[rng.randrange(len(r))]+=2
        check(r,stats)
    for m in range(2,101,2):
        for r in [[2]+[1]*m,[1,2]+[1]*(m-1),[1,3]+[1]*(m-1),[1,4]+[1]*(m-1)]:check(r,stats)
    for j in range(27,200):check([1,2]+[1]*(2*j-1),stats)
    print(json.dumps(stats,indent=2))
    for j in (27,40,100):
        n=num([1,2]+[1]*(2*j-1));print(j,[state(x)[:4] for x in (n,F(n),F(F(n)))])
if __name__=='__main__':main()
```

R271_VERIFY.py

```
"""Finite checks only; proof in R271_APPEND.tex. Python stdlib, deterministic."""
from itertools import product, groupby
from fractions import Fraction
from random import Random
import json

def E(n): return n ^ (2*n)
def F(n):
    y=3*n+1; s=(y & -y).bit_length()-1
    return y>>s,s

def orbit(n,t):
    a=[]
    for _ in range(t): n,s=F(n); a.append(s)
    return n,tuple(a)

def runs(n):
```

```
    lengths=[len(list(g)) for _,g in groupby(bin(E(n))[2:][::-1])]
    return lengths[::2], lengths[1::2]

def label(r,d):
    if not d or any(x!=1 for x in d): return 0
    p=sum(max(x//2-1,0) for x in r[1:]); first,last=r[1],r[-1]
    if max(r[1:])<=2: return 6
    if r[0]%2==0:
        if last in (2,4) and p<=int(first==1): return {2:5,4:4}[last]
        if first==1 and max(r[1:])<=3 and last in (1,3): return {3:3,1:2}[last]
    elif first>=2 and p+last%2-1+int(first==2)==0 and (first in (2,5) or last in (1,4)): return
    1
    return 0

def p24(H): return 0 if H<=28 else 25*(H-29) if H<=52 else 48*(H-28)
def stats(n):
    r,d=runs(n); H=sum(r)-r[0]+2-r[0]%2; Z=sum(d)
    return H,Z,len(d),label(r,d)
def psi(n):
    H,Z,m,l=stats(n); W=H-2
    for _ in range(6):
        e=E(n); W+=e.bit_length()-e.bit_count(); n=F(n)[0]
    return 24*Z+p24(H),Z,m,l,W

def stream_metrics(n):
    """A105 cascade and mask accumulation; terminal zeros stripped offline in this test.
    Full automaton construction (including unique end guess) is in proof, not implemented here
    ."""
    state=[[0,1,0] for _ in range(5)]; words=[[] for _ in range(6)]
    for i in range(n.bit_length()+12):
        x=(n>>i)&1; words[0].append(x)
        for j in range(5):
            p,c,h=state[j]; q=x+p+c; y=q%2; h=h or y
            state[j]=[x,q//2,h]
            if not h: break
            words[j+1].append(y); x=y
    out=[]
    for w in words:
        prev=0; mask=[]
        for x in w: mask.append(x^prev); prev=x
        while mask and mask[-1]==0: mask.pop()
        assert mask and mask[0]==mask[-1]==1
        a=[len(list(g)) for _,g in groupby(mask)]; r,d=a[::2],a[1::2]
        H=sum(r)-r[0]+2-r[0]%2
        out.append((H,sum(d),len(d),label(r,d)))
    return out

def coeff(a):
    S=0; A=0
    for s in a: A=3*A+2**S; S+=s
    return S,A

def affine(n,a,b):
    S,A=coeff(a); R,D=coeff(b)
    return Fraction(2**R*(3**len(a)*n+A)-2**S*D,2**S*3**len(b))
def carry(n,z,a,b):
    S,A=coeff(a); R,D=coeff(b)
    u=2**R*3**len(a); v=-2**S*3**len(b); c=2**R*A-2**S*D
    bound=max(abs(c),abs(u)+abs(v))
    for i in range(max(n.bit_length(),z.bit_length())):
        c+=u*((n>>i)&1)+v*((z>>i)&1)
        if c%2: return False
        c//=2; assert abs(c)<=bound
    return c==0

```

```
def make(Tm,Jm):
    N=max(Tm,Jm); L=5
    while (2**L-1)%3**Jm or 2**L-1<=3**(2*N): L+=1
    b=2**L-1
    q={(T,J):b*3**(T-J) if T>=J else b//3**(J-T) for T in range(Tm+1) for J in range(Jm+1)}
    D=max([0]+[T-J+L-x.bit_length() for (T,J),x in q.items()]); M=max(27,(D+4)//2+1)
    K=Tm+4; g=max(x.bit_length() for x in q.values())+4; U=sum(2**(i*g) for i in range(M))
    n=2**K*b*U-1
    r,d=runs(n); H,Z,m,l=stats(n)
    assert r[0]==r[-1]==1 and H==2*M+2 and m==2*M and min(d)>=3
    assert sum(x%2 for x in r)==2*M and sum((x-(2-x%2))//2 for x in r[1:])==0
    assert H>=56 and 3*m>=H-3 and m<=H-1
    assert orbit(n,Tm)[1]==(1,)*Tm
    return n,q,K,g,U,L,M,D

def check():
    rng=Random(271); out={}
    count=0
    for n in list(range(1,16384,2))+[rng.getrandbits(rng.randrange(1,1025))|1 for _ in range
    (512)]:
        a=stream_metrics(n); x=n
        for s in a: assert s==stats(x); x=F(x)[0]; count+=1
        H,Z,m,l=a[0]
        assert psi(n)==(24*Z+p24(H),Z,m,l,H-2+sum(s[1] for s in a))
    out['stream_state_comparisons']=count
    cases=0
    for k in (79,80,91,127,200):
        for Q in [1,3,7,19,31]+[rng.getrandbits(256)|1 for _ in range(12)]:
            n=2**k*Q+1; z=2**(k-11)*Q+31
            y,a=orbit(n,39); w,b=orbit(z,39)
            assert y==w and a==(2,)*39 and sum(b)==67
            assert affine(n,a,b)==z and carry(n,z,a,b) and not carry(n,z+2,a,b)
            assert psi(z)[0]-psi(n)[0]==-264
            cases+=1
    out['P6-1055_valid_cases']=cases
    out['P6-1065_cases']=[]
    for Tm,Jm in [(0,0),(0,3),(3,0),(1,1),(2,3),(4,4),(5,3)]:
        n,qs,K,g,U,L,M,D=make(Tm,Jm); npsi=psi(n); valid=invalid=unequal=equal=0
        for (T,J),q in qs.items():
            end,a=orbit(n,T)
            choices=list(product(range(1,5),repeat=J))
            if J: choices += [tuple(rng.randrange(1,81) for _ in range(J)) for _ in range(100)]
            for b in choices:
                z=affine(n,a,b); v=Fraction(1)
                for s in reversed(b): v=(2**s*v+1)/3
                R=sum(b); assert 1<=v<=2**(R-J)
                assert z==2**(K+R-T)*q*U-v
                if z.denominator!=1 or z<=0 or z.numerator%2==0: invalid+=1; continue
                z=int(z)
                if orbit(z,J)!=(end,b): invalid+=1; continue
                valid+=1; assert carry(n,z,a,b)
                v=int(v); k=K+R-T
                assert E(z)==E(v-1)+1+2**k*(E(q*U-1)+1)
                assert E(q*U-1)+1==E(q-1)+1+sum(2**(i*g)*E(q) for i in range(1,M))
                assert psi(z)>=npsi
                if T!=J: assert psi(z)[0]>npsi[0]; unequal+=1
                elif psi(z)[0]==npsi[0]: assert psi(z)==npsi; equal+=1
        out['P6-1065_cases'].append(dict(Tmax=Tm,Jmax=Jm,L=L,M=M,D=D,input_bits=n.bit_length(),
    valid=valid,invalid=invalid,unequal_strict=unequal,equal_first=equal))
    out['status']='all assertions passed; finite tests, not universal graph construction or
    proof'
    return out

if __name__=='__main__': print(json.dumps(check(),indent=2))
```

verify_r45.py

```
"""Recheck fixed historical examples and published count arithmetic; no search."""
import json

def step(n):
    v=3*n+1;s=0
    while v%2==0:v//=2;s+=1
    return v,s

def iterate(n,t):
    S=0
    for _ in range(t):n,s=step(n);S+=s
    return n,S

def count(n):
    e=n^(n<<1); b=e.bit_length();h=e.bit_count();r=0
    while (e>>r)&1:r+=1
    H=h-r+2-r%2
    return b,h,H,b-h

def C(n):
    _,_,H,Z=count(n)
    return 24*Z+(0 if H<=28 else 25*(H-29) if H<=52 else 48*(H-28))

def main():
    t=39; x=1;v=31
    assert iterate(x,t)==(1,78) and iterate(v,t)==(1,67)
    assert iterate(v,t-1)[0]!=1
    assert count(x)==(2,2,2,0) and count(v)==(6,2,2,4)
    D=max(1,78+1-2,78-67+6+1-2);assert D==77
    # Fixed historical example from R265/P6-1053, not certificate generation.
    x=11184809;v=26623;Q=5;T=175;J=113
    assert iterate(x,T)==(1,301) and iterate(v,J)==(1,194)
    assert iterate(x,T-1)[0]!=1 and iterate(v,J-1)[0]!=1
    b=count(x)[0];assert b==25
    qp=Q*3**(T-J)
    D2=max(267,301+1-b,301-194+count(v)[0]+1-b);assert D2==277
    n=(Q<<(b+D2))+x;z=(qp<<(b+D2-301+194))+v
    assert iterate(n,T)[0]==iterate(z,J)[0] and C(z)-C(n)==-225
    assert sum([138,6762])==6900 and sum([2116,1214584])==1216700
    assert 294*21==6174 and 6174*256==1580544 and 224*10==2240
    assert 5950+224==6174 and 45+166+13==224
    assert sum([2603587,3461773,3952030,1595281])==11612671
    assert sum([42810972,103013651,316853035,2731642])==465409300
    assert sum([2441410,1877116,938971,6718163,2017781])==13993441
    assert sum([2999577,2797764,1363107,7057950,2392282])==16610680
    assert 429847681-16610680==413237001
    print(json.dumps({'P6-1055':{'t':39,'S':78,'R':67,'D':D,'delta_C':-264},'P6-1053
    _fixed_example':{'T':T,'J':J,'S':301,'R':194,'D':D2,'delta_C':C(z)-C(n),'coalescence':True
    },'reported_count_arithmetic':'consistent; not a certificate replay'},indent=2))
if __name__=='__main__':main()
```

verify_r46.py

```
"""Fixed finite regression examples of the existing P6-982 proof, not a search."""
from fractions import Fraction

import json

def E(n): return n^(2*n)
def h(n): return E(n).bit_count()
def step(n):
    a=3*n+1; s=0
    while a%2==0: a//=2; s+=1
```

```
    return a,s

def counts(n):
    a=E(n); r=0
    while (a>>r)&1: r+=1
    gamma=r-(2-r%2); H=h(n)-gamma; B=a.bit_length()-gamma
    assert H>=54
    return H,B,24*(B+H-56)

def check(word):
    T=len(word)-1; S=[0]
    for s in word: S.append(S[-1]+s)
    y=Fraction(1)
    for s in reversed(word): y=(2**s*y-1)/3
    mod=2**(S[-1]+1); x=(y.numerator*pow(y.denominator,-1,mod))%mod
    assert x>0 and x%2==1
    xs=[x]
    for s in word:
        z,t=step(xs[-1]); assert t==s; xs.append(z)
    lam=(3**T).bit_length()+2; b=2**lam-1
    K=max(S[-1]+2,max((2**S[t]*xs[t]).bit_length()+2 for t in range(T+1)))
    g=lam+(3**T).bit_length()+2
    e=[s-(2-s%2) for s in word]
    M=max(27,(S[T]+h(x)+2*max(e))//4+1)
    n=x+sum(2**(K+i*g)*b for i in range(M)); H0,B0,C0=counts(n)
    z=n; diffs=[]; bounds=[]
    for t,s in enumerate(word):
        if t<=T:
            q=3**t*b
            assert h(q)==2+2*h(3**t-1)
            assert z==xs[t]+sum(2**(K+i*g-S[t])*q for i in range(M))
            H,B,C=counts(z); assert H==h(xs[t])+M*h(q)-e[t]
            assert B-B0>=-S[t]+e[0]-e[t]
            if t:
                bound=4*M-S[t]-h(x)-2*max(e)
                assert C-C0>=24*bound>0
                diffs.append(C-C0); bounds.append(24*bound)
        z,actual=step(z); assert actual==s
    return dict(history=word,initial_bits=n.bit_length(),M=M,score_differences=diffs,
    lower_bounds=bounds)

if __name__=='__main__':
    words=[(1,1),(2,2),(1,1,2,1),(1,1,2,1,1,2,1),(3,7,2,8,1),(31,1,16,5,2)]
    print(json.dumps({'cases':[check(w) for w in words],'status':'all fixed regression checks
    passed; no new coverage'},indent=2))
```

independent_checks.py

```
"""Independent finite audit checks for Paper F R47; no original script imports."""
import random,json
from itertools import groupby
rng=random.Random(470029)
def F(n):
    v=3*n+1; s=(v&-v).bit_length()-1
    return v>>s,s
def it(n,t):
    S=0
    for _ in range(t): n,s=F(n); S+=s
    return n,S
def metrics(n):
    # Read transitions from ordinary binary digits, including final upper zero.
    bits=[int(x) for x in bin(n)[:1:-1]]+[0]
    mask=[x^y for x,y in zip(bits,[0]+bits)]
    a=[len(list(g)) for _,g in groupby(mask)]
    r,d=a[::2],a[1::2]; H=sum(r)-r[0]+2-r[0]%2; Z=sum(d); m=len(d)
```

```
    return r,d,H,Z,m,len(mask),sum(r)
def pen(H): return 0 if H<=28 else 25*(H-29) if H<=52 else 48*(H-28)
def rank(n):
    r,d,H,Z,m,B,h=metrics(n); p=sum(max(x//2-1,0) for x in r[1:]); l=0
    if d and all(x==1 for x in d):
        if max(r[1:])<=2:l=6
        elif r[0]%2==0:
            if r[-1] in [2,4] and p<=int(r[1]==1):l={2:5,4:4}[r[-1]]
            elif r[1]==1 and max(r[1:])<=3 and r[-1] in [1,3]:l={3:3,1:2}[r[-1]]
        elif r[1]>=2 and max(r[1:])>=3 and p+r[-1]%2-1+int(r[1]==2)==0 and (r[1] in [2,5] or r[-1] in [1,4]):l=1
    W=H-2; x=n
    for _ in range(6): W+=metrics(x)[3]; x=F(x)[0]
    C=24*Z+pen(H); L=7*(Z*(Z+1)//2+m)+l
    A=7*C*(C+3)//2+7; D=7*C+205
    assert 0<=L<A and 0<=W<D
    return (C,L,W),C*A*D+L*D+W
def norm(n):
    y,s=F(n); return ((1<<(2-s%2))*y-1)//3
ns=list(range(1,10000,2))+[rng.getrandbits(rng.randrange(1,513))|1 for _ in range(1000)]
records=[]; boundaries=0
for n in ns:
    r,d,H,Z,m,B,h=metrics(n); y,s=F(n); v=norm(n)
    assert s==r[0] and F(v)[0]==y and v%2==1 and rank(n)==rank(v)
    rr,dd,HH,ZZ,mm,BB,hh=metrics(v)
    assert (rr[1:],dd,HH,ZZ)==(r[1:],d,H,Z)
    assert metrics(norm(y))[5]<=BB+1
    if m:
        p=[a%2 for a in rr]; eps=[a^(i==0) for i,a in enumerate(p[:-1])]; I=[int(a==1) for a in d]; o=sum(p); e=sum((a-2+a%2)//2 for a in rr[1:]); a=sum(x*y for x,y in zip(eps,I)); U=1+p[-1]+sum(2+x*(1-2*y) for x,y in zip(eps,I)); delta=1+p[-1]-rr[0]
        assert H==2*(m+1)-o+2*e and U+delta-H==2*(o-e-a)-1-p[0]+p[-1]
        assert (H<=U+delta)==(e+a<=o-1+int(p[0]==0 and p[-1]==1));boundaries+=1
    records.append(rank(n))
assert [v for _,v in sorted(records)]==sorted(v for _,v in records)
for i in range(300):
    x,q,u=[rng.randrange(1,2**20,2) for _ in range(3)]; t=rng.randrange(1,12); xt,S=it(x,t); qt=3**t*q; ut=3**t*u
    bx,bq=[metrics(a)[5] for a in (x,q)]; d1=max(1,S+metrics(xt)[5]+1-bx); d2=max(1,metrics(qt)[5]+1-bq); assert max(d1,d2)<=2*t+1
    ds=[]
    for a,b in [(0,0),(1,0),(0,1),(13,17)]:
        k=bx+d1+a; l=k+bq+d2+b; n=x+(q<<k)+(u<<l); y,R=it(n,t)
        assert R==S and y==xt+(qt<<(k-S))+(ut<<(l-S));ds.append(rank(y)[0][0]-rank(n)[0][0])
    assert len(set(ds))==1
# Explicitly verify P6-577 on the seven displayed obstruction constructions.
short=[]
for T,J in [(0,0),(0,3),(3,0),(1,1),(2,3),(4,4),(5,3)]:
    L=5
    while (2**L-1)%3**J or 2**L-1<=3**(2*max(T,J)): L+=1
    b=2**L-1; qs=[b*3**t//3**j for t in range(T+1) for j in range(J+1)]; g=max(q.bit_length() for q in qs)+4; D=max([0]+[t-j+L-(b*3**t//3**j).bit_length() for t in range(T+1) for j in range(J+1)]); M=max(27,(D+4)//2+1); n=(b*sum(1<<(i*g) for i in range(M))<<(T+4))-1
    z=[metrics(it(n,t)[0])[3] for t in range(3)]; assert min(z[1:])<z[0];short.append({'Tmax':T,'Jmax':J,'delta_Z1':z[1]-z[0],'delta_Z2':z[2]-z[0]})
print(json.dumps({'normalization_rank_inputs':len(ns),'boundary_identity_inputs':boundaries,'two_gap_families':300,'two_gap_comparisons':1200,'obstruction_family_Z_checks':short,'status':'pass; finite checks only'},indent=2))
```