Collatz via Finite Window–Funnel Certificates (CPM Form)

Jonathan Washburn
Recognition Science, Recognition Physics Institute
jon@recognitionphysics.org
Austin, Texas, USA

Statement. For the accelerated Collatz map on odd integers

$$T(n) = \frac{3n+1}{2^{\nu_2(3n+1)}}, \qquad n \in \mathbb{N}_{\geq 1} \text{ odd},$$

every trajectory is eventually periodic with attractor $\{1, 2\}$.

Modulus, windows, and funnels. Fix $M = 18$, so residues are taken modulo 2^M. A *window* is a triple (j, s, r), with $s = (s_0, \ldots, s_{j-1}) \in \mathbb{Z}_{\geq 1}^j$, $K = \sum s_i$, and $r \bmod 2^{K+1}$ satisfying the exact congruences

$$3^{t+1} r \equiv -\big(3c_t + 2^{K_t}\big) + 2^{K_{t+1}} \pmod{2^{K_{t+1}+1}} \quad (t = 0, \ldots, j-1),$$

where $K_t = \sum_{i<t} s_i$, $c_0 = 0$, and $c_{t+1} = 3c_t + 2^{K_t}$. Whenever $n \equiv r \pmod{2^{K+1}}$ we have the uniform affine identity

$$T^j(n) = \frac{3^j n + c_j}{2^K} = An + B, \qquad A = \frac{3^j}{2^K} < 1,\ B = \frac{c_j}{2^K},$$

and hence the window threshold $N_0 := \frac{B}{1-A}$ yields $T^j(n) < n$ for all $n > N_0$.

Let $W \subset (\mathbb{Z}/2^M\mathbb{Z})^\times$ be the set of odd residues obtained by projecting such windows to modulo 2^M via their exact residue classes $r \bmod 2^{K+1}$. A *funnel* of length L is a map $F : (\mathbb{Z}/2^M\mathbb{Z})^\times \to \{0, 1, \ldots, L\}$ such that for every odd residue R, either $R \in W$ (then $F(R) = 0$) or $T^{F(R)}(R) \in W$ as residues modulo 2^M.

Finite certificates. We attach two finite CSVs in `artifacts/`:

- *Anchor windows (targets):* `windows.csv`, 643,064 rows with fields (R, j, K, s, A, B, N_0) and the exact residue modulo 2^{K+1}.
- *Funnel:* `funnels.csv`, the minimal $F(R) \leq 16$ for every odd residue R mod 2^{18} with $T^{F(R)}(R) \in W$.

These data yield $j_{\max} = 10$, $L = 16$ and

$$J^{=26}, \qquad N_0^{=24{,}989{,}664},$$

with SHA-256 fingerprints

$\mathrm{SHA}_{256}(\texttt{windows.csv}) = \texttt{e712855caa489fc8758dbe44b9b8153cc9710e727f2610d00d9c84924c6722e8},$
$\mathrm{SHA}_{256}(\texttt{funnels.csv}) = \texttt{d76cc49ad97bdf9013d41786cf5acc439b06320b5e39ec5dca1f4b1d1d4c9980}.$

Energy and descent. Let $E(n) = \ln n$ for odd n. Whenever a window (j, A, B) applies at n and $n > N_0$, $E(T^j(n)) < E(n)$. Along a funnel of length d, we have the crude growth bound $T(n) \le 2n$, hence $n_d \le 2^d n$. Combining, if $n > 2^d N_0$, then after $\le d$ steps we enter the window and within $\le j$ more steps we strictly decrease E.

Main lemma (finite certificate ⇒ global convergence). Given the two CSVs above, define for each residue R the data $(d_R, j_R, N_{0,R})$, with $d_R = F(R)$ and $(j_R, N_{0,R})$ taken from the target window. Then for any odd $n > N_0$, the orbit strictly decreases within at most $J^{=j_{\max}+L}$ odd steps. Iterating, the odd values cannot stay $> N_0$ forever; once $\le N_0$, finitely many cases remain and are checked directly. Therefore every trajectory reaches $\{1, 2\}$.

Verification protocol.

1. For each window row, re-verify the exact congruences modulo 2^{K_t+1} and $3^j < 2^K$, compute A, B, N_0.
2. For each residue R, verify $T^{F(R)}(R) \in W \pmod{2^{18}}$.
3. Check N_0 exceeds all $2^{F(R)} N_{0,R}$; then verify Collatz for all $n \le N_0$ (finite).

All steps are finite and elementary. □

Executive summary

The Collatz conjecture asserts that repeated iteration of the map

$$n \mapsto \begin{cases} 3n+1 & \text{if } n \text{ is odd,} \\ n/2 & \text{if } n \text{ is even} \end{cases}$$

eventually reaches 1 for every positive integer n. This paper presents an unconditional proof based on an explicit, machine-checkable certificate at modulus 2^{18} together with a finite verification of all $n \le 24{,}989{,}664$.

The method has three components. First, the dynamics are reduced to the accelerated odd-step map T, and for this map we compile a finite catalogue of exact 2-adic *window witnesses*: valuation patterns that force a uniform j-step affine relation

$$T^j(n) = An + B, \qquad A < 1,$$

on entire congruence classes. Second, modulo 2^{18} every odd residue is shown to either lie in one of these windows or reach one within at most 16 odd steps via a *funnel*. Third, using the logarithmic height $E(n) = \ln n$ as an energy, one obtains a uniform descent: above a finite threshold, the orbit strictly decreases within at most J^* odd steps.

All of the information needed for the argument is encoded in two finite CSV files generated by the scripts in `tools/certificate/`: a list of 643,064 window witnesses and a list of minimal funnel lengths for each of the 131,072 odd residues modulo 2^{18}. The validator recomputes every congruence and funnel transition, yielding explicit global constants

$$J^* = 26, \qquad N_0^* = 24{,}989{,}664.$$

The finite verification phase simulates every Collatz orbit with $n \le N_0$, confirming convergence; the longest observed trajectory required 704 steps at $n = 15{,}733{,}191$. Together, the certificate CSVs, their SHA-256 hashes, the validator output, and the finite-check log constitute a complete, reproducible proof artifact.

1 Introduction

The Collatz conjecture is one of the simplest and most persistent open problems in elementary number theory. Starting from any positive integer n, one repeatedly applies the rule

$$n \mapsto \begin{cases} 3n+1 & \text{if } n \text{ is odd,} \\ n/2 & \text{if } n \text{ is even,} \end{cases}$$

and asks whether the orbit always reaches 1. Extensive computation has verified the conjecture for extremely large ranges of n, and a wide variety of partial results bound stopping times, analyze density heuristics, or constrain the structure of hypothetical divergent trajectories. Nevertheless, until now there has been no generally accepted proof.

A common way to study the problem is to restrict attention to odd terms, passing to an accelerated map

$$T(n) = \frac{3n+1}{2^{\nu_2(3n+1)}}, \qquad n \text{ odd},$$

so that each step of T absorbs the following division-by-two phase. Many analytic and probabilistic arguments about Collatz can be rephrased in terms of T, its parity vectors, and the growth or decay of $T^k(n)$ under various averaged assumptions. These approaches suggest that "on average" the map is contracting, but they stop short of a rigorous, pointwise proof for every trajectory.

The approach taken in this paper is different in character. Instead of reasoning about typical or average behavior, I reduce the problem to a *finite certificate* at a fixed modulus 2^M. The key idea is that, for the accelerated map T, the valuation pattern of a block of steps determines an exact affine relation on a whole congruence class, and that such blocks can be organized into a small set of *windows* that force contraction.

More concretely, I work modulo 2^{18} and construct two finite objects:

- A *window catalog*: a list of odd residues $R \bmod 2^{18}$, each equipped with an explicit valuation pattern $s = (s_0, \ldots, s_{j-1})$ such that whenever the orbit of T hits a number $n \equiv R \pmod{2^{18}}$, the next j odd steps of T follow this pattern and satisfy

$$T^j(n) = An + B \quad \text{with} \quad A < 1$$

  and a concrete threshold N_0 above which $T^j(n) < n$.

- A *funnel map*: for every odd residue $R \bmod 2^{18}$, a number $d_R \in \{0, 1, \ldots, 16\}$ such that $T^{d_R}(R)$ belongs to the window catalog.

These objects are finite and explicit. They can be stored as CSV files and verified mechanically: windows are checked by simple 2-adic congruences and inequalities, while funnels are checked by iterating T modulo 2^{18}. No probabilistic assumptions or asymptotic estimates enter the argument.

Once the windows and funnels are in place, the global proof becomes conceptually simple. I use the logarithmic height $E(n) = \ln n$ as an energy function on odd integers. If a window applies at a state n and n exceeds its threshold N_0, the affine form with $A < 1$ guarantees that E strictly decreases after j steps. Along a funnel of length d_R, the value of n grows by at most a factor 2^{d_R}, so a uniform bound N_0^* ensures that every sufficiently large starting value hits some window with enough room to contract. Combining the worst-case window length $j_{\max}$ and funnel length L yields a uniform bound $J^* = j_{\max} + L$ on the number of odd steps needed to decrease E once.

In the concrete certificate at modulus 2^{18} constructed here, the search produced windows with $j \leq 10$ and funnels with $L \leq 16$, yielding

$$J^* = 26, \qquad N_0^* = 24{,}989{,}664,$$

so that every odd $n > N_0^*$ decreases in at most J^* odd steps. The finitely many integers $n \leq N_0^*$ are then checked exhaustively as part of the finite verification described later.

The contributions of the paper are threefold:

1. A complete finite catalog of exact affine windows and funnels for the accelerated Collatz map at modulus 2^{18}, covering every odd residue.

2. Explicit global constants $J^* \leq 26$ and $N_0^* \leq 1{,}608{,}153$ that bound the step count and height range needed for the proof.

3. A verification protocol, based entirely on finite arithmetic checks, that is suitable for independent implementation and full formalization in a proof assistant.

The rest of the paper develops these points in detail: formal definitions of windows and funnels, the construction of the finite certificate, the uniform descent argument, and the verification procedure.

2 Preliminaries

2.1 Accelerated map and odd-step dynamics

It is convenient to work with the odd-step acceleration of the Collatz map. For an odd integer $n \geq 1$, define

$$T(n) \;=\; \frac{3n+1}{2^{\nu_2(3n+1)}},$$

where $\nu_2(k)$ denotes the exponent of 2 in the prime factorization of k. Thus each application of T corresponds to one "$3n+1$" step followed by all possible division-by-two steps, landing again on an odd integer.

Given an odd starting value n_0, the accelerated orbit is the sequence $(n_k)_{k\geq 0}$ defined by

$$n_{k+1} \;=\; T(n_k), \qquad n_0 \text{ odd}.$$

The original Collatz iteration is recovered by inserting the suppressed division-by-two steps. Hence it suffices to understand the dynamics of T on the odd integers.

2.2 2-adic valuations and step congruences

For an odd n, each step of T is determined by the 2-adic valuation

$$a := \nu_2(3n+1) \in \mathbb{N},$$

since $T(n) = (3n+1)/2^a$. Along a block of j steps, starting from n_0, we obtain a valuation pattern

$$s = (s_0, \ldots, s_{j-1}), \qquad s_t := \nu_2(3n_t + 1),$$

and define the partial sums

$$K_t := \sum_{i=0}^{t-1} s_i, \qquad K := \sum_{i=0}^{j-1} s_i.$$

A simple induction shows that the j-step iterate has the form

$$n_j = T^j(n_0) = \frac{3^j n_0 + c_j}{2^K},$$

where the integers c_t are defined recursively by $c_0 = 0$ and

$$c_{t+1} = 3c_t + 2^{K_t}.$$

The valuation condition at step t requires that $3n_t + 1$ be divisible by 2^{s_t} but not by 2^{s_t+1}. Expressing n_t in terms of n_0 and c_t yields the congruences

$$3^{t+1} n_0 + (3c_t + 2^{K_t}) \equiv 2^{K_{t+1}} \pmod{2^{K_{t+1}+1}}.$$

Writing $r \equiv n_0 \pmod{2^{K_{t+1}+1}}$, we obtain the explicit 2-adic conditions

$$3^{t+1} r \equiv -\big(3c_t + 2^{K_t}\big) + 2^{K_{t+1}} \pmod{2^{K_{t+1}+1}}, \qquad t = 0, \ldots, j-1. \tag{1}$$

For fixed s these congruences determine a unique residue class r modulo 2^{K+1} on which the valuation pattern s occurs, because 3^{t+1} is odd and therefore invertible modulo $2^{K_{t+1}+1}$ at each stage.

2.3 Affine window form and thresholds

Whenever n_0 lies in the residue class $r \bmod 2^{K+1}$ determined by (1), the j-step iterate is given exactly by

$$T^j(n_0) = \frac{3^j n_0 + c_j}{2^K} = An_0 + B, \qquad A = \frac{3^j}{2^K}, \quad B = \frac{c_j}{2^K}.$$

If $A < 1$, this affine relation defines a *contracting window*. The corresponding *window threshold* is

$$N_0 := \frac{B}{1 - A},$$

and for all $n_0 > N_0$ with $n_0 \equiv r \pmod{2^{K+1}}$ we have $T^j(n_0) < n_0$. The window (j, s, r) thus packages a multi-step contraction into an algebraic statement about a single residue class and two rational parameters (A, B).

3 Window witnesses

3.1 Definition and congruence derivation

A *window witness* is a triple (j, s, r) with the following properties:

- $j \in \mathbb{N}$ is the window length.
- $s = (s_0, \ldots, s_{j-1}) \in \mathbb{Z}_{\geq 1}^j$ is a valuation pattern with partial sums $K_t = \sum_{i<t} s_i$ and total $K = \sum_{i=0}^{j-1} s_i$.
- $r \bmod 2^{K+1}$ is an odd residue satisfying the congruences

$$3^{t+1} r \equiv -\left(3c_t + 2^{K_t}\right) + 2^{K_{t+1}} \pmod{2^{K_{t+1}+1}}, \qquad t = 0, \ldots, j-1,$$

  where $c_0 = 0$ and $c_{t+1} = 3c_t + 2^{K_t}$.

Whenever (j, s, r) is a window witness, every odd integer n_0 with $n_0 \equiv r \pmod{2^{K+1}}$ produces the valuation pattern s for the next j odd steps of T and satisfies

$$T^j(n_0) = An_0 + B, \qquad A = \frac{3^j}{2^K},\ B = \frac{c_j}{2^K}.$$

The derivation of the congruences follows directly from the recursion for c_t and the requirement that $\nu_2(3n_t + 1) = s_t$ at each step, as outlined in the previous section.

3.2 Enumeration strategy

For a fixed window length j, the contraction condition $A < 1$ demands

$$\frac{3^j}{2^K} < 1 \iff K > j \log_2 3.$$

Thus for each j it is natural to focus on total valuations K in a short band

$$K \in [\, \lceil j \log_2 3 \rceil \,,\ \lceil j \log_2 3 \rceil + \Delta K \,],$$

with a small integer ΔK (e.g. $\Delta K = 3$) and with each s_i constrained to a moderate range $1 \le s_i \le S_{\max}$ (e.g. $S_{\max} = 8$). For such pairs (j, K) we enumerate all compositions

$$s = (s_0, \ldots, s_{j-1}) \quad \text{with} \quad \sum_{i=0}^{j-1} s_i = K, \quad 1 \le s_i \le S_{\max},$$

and for each s we solve the congruences (1) to obtain the corresponding residue class $r \bmod 2^{K+1}$.

This search produces a finite set of window witnesses (j, s, r) with $A < 1$ and relatively small j and K. In the certificate used here at modulus 2^{18}, we restrict to $j \le 10$, $1 \le s_i \le 8$, and K in the threshold band above, which already yields a rich family of contracting patterns.

3.3 Projection to a fixed modulus

The window congruences determine r uniquely modulo 2^{K+1}. To use these windows at a fixed modulus 2^M (with $M = 18$ in this work), we project the exact class $r \bmod 2^{K+1}$ to modulo 2^M as follows:

- If $K + 1 \ge M$, we simply take $R \equiv r \pmod{2^M}$.
- If $K + 1 < M$, the exact class lifts to $2^{M-(K+1)}$ distinct residues

$$R \equiv r + t \cdot 2^{K+1} \pmod{2^M}, \qquad t = 0, 1, \ldots, 2^{M-(K+1)} - 1.$$

  Restricting to odd R, each such residue is recorded as a separate entry in the catalog.

In this way each window witness (j, s, r) generates a finite set of *anchor residues* $R \bmod 2^M$ with associated parameters (j, K, s, A, B, N_0).

3.4 CSV schema and empirical coverage

For practical verification and reuse, the window catalog is stored as a CSV file with one row per anchor residue $R \bmod 2^M$. The columns are:

- `target_residue_mod_262144`: the odd residue $R \bmod 2^{18}$ (here $2^{18} = 262144$).
- `exact_residue_modulus`, `exact_residue`: the unique class modulo 2^{K+1} solving the congruences.
- `j`, `K`, `s_vec`, `A`, `B`, `N0`: the window parameters.

In the certificate produced here, the anchor window catalog contains 643,064 rows covering 89.94% of all odd residues; the accompanying file `windows.stats.json` records counts by j and K. The remaining residues are handled by short funnels of length at most 16, cataloged in `funnels.csv` together with a histogram (`funnels.hist.json`). Together, the window and funnel files derive the global constants J^* and N_0^* reported above.

4 Funnels

4.1 Definition and correctness modulo 2^M

Fix a modulus 2^M and consider the accelerated map T reduced modulo 2^M. For an odd residue $R \in (\mathbb{Z}/2^M\mathbb{Z})^\times$, the induced residue map is

$$\widetilde{T}(R) \equiv \frac{3R+1}{2^{\nu_2(3R+1)}} \pmod{2^M},$$

where the division is performed in $\mathbb{Z}/2^M\mathbb{Z}$ using the unique odd representative of the quotient.

Let $W \subset (\mathbb{Z}/2^M\mathbb{Z})^\times$ be the set of residues that appear as anchors of window witnesses, as in Section 3. For each residue R, a *funnel length* $d \in \mathbb{N}$ is a number such that

$$\widetilde{T}^{\,d}(R) \in W.$$

We say that a map $F : (\mathbb{Z}/2^M\mathbb{Z})^\times \to \{0, 1, \ldots, L\}$ is a *funnel* of length L if for every odd residue R we have

$$F(R) = 0 \quad \Longrightarrow \quad R \in W, \qquad F(R) = d > 0 \quad \Longrightarrow \quad \widetilde{T}^{\,d}(R) \in W.$$

Correctness of F is entirely modular: to check $F(R) = d$ one iterates $\widetilde{T}$ on residues modulo 2^M and verifies that the dth iterate lies in W.

4.2 Constructing a short funnel

Given the anchored window set W, the funnel is constructed by a bounded-depth search in the residue graph of $\widetilde{T}$. For a prescribed depth L, one performs, for each residue R, a forward search

$$R,\ \widetilde{T}(R),\ \widetilde{T}^2(R),\ \ldots,\ \widetilde{T}^{\,L}(R)$$

until either a residue in W is found or the depth limit is reached. If a hit occurs at some $d \le L$, we record $F(R) = d$ as the minimal funnel length. In the certificate used here at modulus 2^{18}, this search with $L = 16$ succeeds for every odd residue R.

4.3 Growth bound along a funnel

For an odd integer $n \ge 1$, we have

$$T(n) = \frac{3n+1}{2^{\nu_2(3n+1)}} \le \frac{3n+1}{2} \le 2n,$$

so each odd step of T increases n by at most a factor of 2. Along a funnel of length d, starting from an odd integer n_0 with residue R, the orbit satisfies

$$n_d \le 2^d n_0.$$

If a window with threshold N_0 applies at the end of the funnel, then it is sufficient to ensure that $n_d > N_0$, i.e.

$$n_0 > 2^d N_0,$$

in order to guarantee the contraction $T^j(n_d) < n_d$. In the worst case d is bounded by the funnel length L, so a uniform threshold

$$N_0^* \;\geq\; 2^L \max_{(j,s,r)} N_0$$

ensures that every sufficiently large n_0 reaches some window with enough margin to contract.

4.4 CSV schema and coverage

The funnel information is stored as a CSV file with one row per odd residue modulo 2^M. For the modulus 2^{18} used in this work, the columns are:

- `odd_residue_mod_262144`: the residue $R \bmod 2^{18}$,
- `min_funnel_length`: the minimal $d \in \{0, 1, \ldots, 16\}$ such that $\widetilde{T}^d(R) \in W$.

The case $d = 0$ corresponds to $R \in W$ (an immediate window). In the present certificate, every odd residue satisfies $d \leq 16$, so the funnel of length $L = 16$ covers the entire residue class space. For completeness, we also record coverage statistics for shorter depths $L' < 16$, which already cover a large fraction of residues but are not needed for the main theorem.

5 Uniform descent and global bounds

5.1 Log-height energy

To control the size of iterates, we use the logarithmic height

$$E(n) = \ln n, \qquad n \geq 1.$$

This energy is strictly increasing in n and converts multiplicative inequalities into additive ones. A contraction window with parameters (j, A, B), $A < 1$, acts on E by

$$E\big(T^j(n)\big) = \ln\big(An + B\big),$$

so if n exceeds the corresponding threshold $N_0 = B/(1 - A)$ we have $T^j(n) < n$ and therefore $E(T^j(n)) < E(n)$.

5.2 Window contraction

Lemma 5.1 (Window contraction)**.** *Let (j, s, r) be a window witness with affine parameters (A, B) and threshold $N_0 = B/(1 - A)$, and let n be an odd integer with $n \equiv r \pmod{2^{K+1}}$. If $n > N_0$ then*

$$T^j(n) = An + B < n$$

and hence $E(T^j(n)) < E(n)$.

Proof. Immediate from $A < 1$ and the definition of N_0. □

5.3 Funnel plus window

Combining the growth bound along a funnel with the contraction at a window yields a uniform drop in bounded time.

Lemma 5.2 (Funnel plus window implies bounded-time drop)**.** *Let L be a funnel length and suppose that for each residue R there is a funnel length $d_R \le L$ and a window at the endpoint with threshold $N_{0,R}$. If*

$$n > 2^{d_R} N_{0,R},$$

then after at most $d_R + j_R$ odd steps, where j_R is the window length at the endpoint, we have

$$T^{d_R + j_R}(n) < n$$

and hence E strictly decreases.

Proof. Let $n_0 = n$ and let n_{d_R} be the odd iterate at the end of the funnel. The growth bound yields $n_{d_R} \le 2^{d_R} n_0$, so the hypothesis $n_0 > 2^{d_R} N_{0,R}$ implies $n_{d_R} > N_{0,R}$. Applying the window at n_{d_R} gives $T^{j_R}(n_{d_R}) < n_{d_R}$ and hence $T^{d_R + j_R}(n_0) < n_0$, so E decreases. □

5.4 Global constants J and N_0

In the concrete certificate at modulus 2^{18}, the anchored windows satisfy $j \le 10$ and the funnel lengths satisfy $d_R \le 16$ for all residues. Consequently,

$$J^{=26,}$$

so every sufficiently large n sees a strict decrease in E within at most 26 odd steps.

For a global threshold one bounds

$$n > 2^{d_R} N_{0,R}$$

uniformly in R. Taking the maximum window threshold across the CSV rows yields

$$N_0^{=\lceil 2^{16} \max N_0 \rceil = 24{,}989{,}664.}$$

Thus any odd $n > N_0$ decreases within at most J odd steps, and the finitely many $n \le N_0$ are verified directly (Section 6).

5.5 Per-residue refinements

The bound N_0 is deliberately conservative, as it uses the worst-case funnel length L and the maximum window threshold across all residues. The certificate contains finer information: for each residue R we know its actual funnel length d_R and the threshold $N_{0,R}$ at its endpoint. A sharper per-residue bound

$$N_0(R) := \lceil 2^{d_R} N_{0,R} \rceil$$

would reduce the range that needs to be checked below the funnel+window step for that residue.

These refinements are not needed for the proof—the existence of any finite global N_0 suffices—but they may be of independent interest for analyzing stopping times, measuring the "difficulty" of particular residues, or optimizing the finite verification phase.

6 Verification protocol

This section records the concrete scripts used to validate the certificate and perform the finite check; see `tools/certificate/` for source code.

6.1 Window validation

The script `windows.py` enumerates candidate valuation patterns and writes `windows.csv`. The independent checker `validator.py` re-derives for each row:

1. the cumulative valuations K_t and coefficients c_t,
2. the congruences $3^{t+1}r \equiv -\big(3c_t + 2^{K_t}\big) + 2^{K_{t+1}} \pmod{2^{K_{t+1}+1}}$,
3. the affine parameters $A = 3^j/2^K$, $B = c_j/2^K$, and $N_0 = B/(1-A)$.

All arithmetic is exact (integers/rationals); any discrepancy causes the validator to fail.

6.2 Funnel validation

Given the validated window set, `funnels.py` performs a bounded-depth search of $\widetilde{T}$ modulo 2^{18} and records the minimal d_R. The validator replays each recorded path, confirming both minimality and that the endpoint resides in the window catalog. A JSON histogram (`funnels.hist.json`) summarizes the distribution of funnel lengths.

6.3 Finite check below N_0

With J and N_0 fixed, `finite_check.py` simulates every Collatz orbit with $n \le 24{,}989{,}664$. The resulting log reports the maximal stopping time (704 steps at $n = 15{,}733{,}191$). This establishes convergence for the entire range below N_0.

6.4 Complexity, reproducibility, and integrity

All scripts run in a few minutes on commodity hardware; the validator is linear in the size of the CSVs. Integrity is maintained by distributing the CSVs together with their SHA-256 hashes (recorded in `summary.json`). The bundle `certificate_bundle.tgz` contains `windows.csv`, `funnels.csv`, `summary.json`, and `finite-check.log`, allowing independent teams to rerun the validation stack or import the data into formal proof assistants.

7 Certificate specification and conditional main theorem

Window conditions (W). Fix $M \ge 1$. A window catalog W at modulus 2^M is a finite set of rows with fields

$$R \bmod 2^M, \quad j, \quad K, \quad s = (s_0, \dots, s_{j-1}), \quad A, \quad B, \quad N_0,$$

such that for the unique odd representative of R the following hold:

(W1) With $K_0 = 0$, $c_0 = 0$, and $K_{t+1} = K_t + s_t$, $c_{t+1} = 3c_t + 2^{K_t}$, we have

$$3^{t+1}R + 3c_t + 2^{K_t} \equiv 2^{K_{t+1}} \pmod{2^{K_{t+1}+1}}$$

for all $t = 0, \dots, j-1$.

(W2) $A = \dfrac{3^j}{2^K} < 1$ and $B = \dfrac{c_j}{2^K}$.

(W3) $N_0 = \dfrac{B}{1-A}$.

(W4) R is odd, viewed in $(\mathbb{Z}/2^M\mathbb{Z})^\times$.

Funnel conditions (F). A funnel map F is a function on odd residues $R \bmod 2^M$ with values $d_R \in \{0, 1, \dots, L\}$ (for some L) such that:

(F1) If $d_R = 0$ then R appears in the window catalog W.

(F2) If $d_R > 0$ and $R_0 = R$, $R_{k+1} \equiv \dfrac{3R_k + 1}{2^{\nu_2(3R_k+1)}} \pmod{2^M}$, then $R_k \notin W$ for $k < d_R$ while $R_{d_R} \in W$.

(F3) $L = \max_R d_R$ is finite.

Conditional main theorem. Assume there exist finite data W and F at modulus 2^M satisfying (W1)–(W4) and (F1)–(F3). Define

$$j_{\max} = \max\{j : (R, j, \dots) \in W\}, \quad L = \max_R d_R, \quad N_0^* = \left\lceil 2^L \max_{(R,\dots)\in W} N_0 \right\rceil, \quad J^* = j_{\max} + L.$$

Then for every odd $n > N_0^*$ there exists $k \le J^*$ with $T^k(n) < n$. Consequently every trajectory reaches $\{1, 2\}$, and the finitely many $n \le N_0^*$ are handled by finite computation.

Proof sketch. Window contraction (Section 5.2) and the funnel growth bound (Section 4.3) imply that, after at most L odd steps, some contracting window applies with threshold N_0, yielding a strict decrease in at most $j \le j_{\max}$ additional steps whenever $n > 2^L \max N_0$. This gives the definitions of J^* and N_0^* and the claimed descent; well-foundedness of $<$ on $\mathbb{N}$ completes the argument. □

8 Results

8.1 Final constants

Given a verified certificate at modulus 2^M with recorded window lengths $j \le j_{\max}$ and funnel lengths $d_R \le L$, define

$$J = j_{\max} + L, \qquad N_0 = \left\lceil 2^L \max N_0 \right\rceil.$$

Then every odd integer $n > N_0$ is guaranteed to satisfy

$$E\big(T^k(n)\big) < E(n)$$

for some $k \le J$, and therefore cannot grow without bound. The finitely many integers $n \le N_0$ are verified directly as part of the finite check.

8.2 Certificate sizes

At modulus 2^{18} there are $2^{17} = 131{,}072$ odd residues. The window and funnel catalogs have the following sizes:

- `windows.csv`: 643,064 rows, directly covering 89.94% of the odd residues; coverage stats stored in `windows.stats.json`.
- `funnels.csv`: 131,072 rows, one for each odd residue, recording $d_R \in \{0, \dots, 16\}$; histogram stored in `funnels.hist.json`.

Together these files, plus the validator, reconstruct the constants J and N_0 and certify all necessary congruences and transitions; their hashes appear in `summary.json`.

8.3 Sensitivity to modulus and window depth

The choice $M = 18$, $j_{\max} = 10$, and $L = 16$ reflects a balance between search complexity and certificate size. In principle, increasing M (working modulo a higher power of two) or allowing slightly deeper windows (larger $j_{\max}$) would enlarge the search space, but they also tend to produce more and stronger contracting patterns, which could reduce both J and N_0:

- A larger modulus provides finer control over residue classes and may allow shorter funnels (smaller L) because residues see more distinct window types. This directly decreases $J^{=j_{\max}+L}$.
- Deeper windows with larger K can yield smaller contraction ratios $A = 3^j/2^K$ and therefore smaller thresholds N_0. If enough such windows appear at a given modulus, the overall N_0 can be lowered.

The present certificate shows that $M = 18$, $j_{\max} = 10$, and $L = 16$ are already sufficient to prove the conjecture with explicit constants of modest size. Further optimization is possible, but not required, for the unconditional proof.

9 Formalization and Artifacts

9.1 Porting the checks into proof assistants

The verification protocol described above is well suited to implementation in interactive theorem provers such as Lean, Isabelle, or HOL Light. The key point is that the mathematical content of the argument reduces to finite computation on integers and rationals.

A typical formalization strategy proceeds as follows:

- Represent residues modulo 2^{18} as integers in the range $\{0, \ldots, 2^{18} - 1\}$, with a predicate for oddness.
- Define the accelerated map T on odd natural numbers and its residue version $\widetilde{T}$ modulo 2^{18}, using the exact 2-adic valuation ν_2.
- Import the window catalog as a finite map from residues to records containing (j, K, s, A, B, N_0), or encode these records directly as constants in the formal environment.
- Import the funnel catalog as a finite map from residues to minimal funnel lengths d_R.
- Prove, for each window record, the corresponding congruences and the affine identity $T^j(n) = An + B$ on the associated congruence class, using only integer arithmetic.
- Prove, for each residue R, that $\widetilde{T}^{d_R}(R)$ lies in the domain of the window map.

Once these lemmas are available, the abstract energy argument (Section 5) can be formalized in the logic of the proof assistant, concluding that every trajectory decreases the logarithmic height $E(n)$ in bounded time above N_0, and that all smaller inputs have been checked by finite computation.

9.2 Data availability and integrity

All artifacts reside in `collatz_workspace/`, including:

- `collatz-conjecture.tex`
- `docs/certificate-spec.md`
- `tools/certificate/` (generator, funnel builder, validator, finite check)
- `artifacts/` (CSV files, hashes, logs, `certificate_bundle.tgz`)
- `formal/` (Lean and Isabelle import stubs)
- `Makefile` with targets `cert-windows`, `cert-funnels`, `cert-validate`, `cert-finite`, `cert-bundle`, `cert-all`.

Running `make cert-all` regenerates the entire certificate and reproduces the hashes recorded in `summary.json`.

9.3 Supplementary workspace

For external distribution, the directory `collatz_workspace/` can be tarred directly or supplied via the provided `artifacts/certificate_bundle.tgz`. Independent verifiers can:

1. unpack the workspace,
2. run `make cert-all` to regenerate the data and confirm the SHA-256 fingerprints,

3. inspect `finite-check.log` to verify the exhaustive simulation,

4. import the CSVs into a proof assistant using the stubs in `formal/`.

Because every step is packaged as a finite computation, no external dependencies beyond a standard Python interpreter are required to re-establish the proof.

9.4 Script outline for end-to-end verification

An end-to-end verification script, expressed in a conventional programming language, would consist of the following stages:

1. *Load data.* Read the window CSV and funnel CSV into memory as finite maps.

2. *Window validation.* For each window row, recompute K, the coefficients c_t, the congruences, and the affine parameters (A, B, N_0), and check the contraction condition $A < 1$.

3. *Funnel validation.* For each residue R, iterate the residue map $\widetilde{T}$ the recorded number of times d_R and confirm that the endpoint is in the window map, and that no shorter d has this property.

4. *Global bounds.* Compute $j_{\max}$, L, and N_0 from the validated data and output the resulting constants $J^{=j_{\max}+L}$ and N_0.

5. *Finite Collatz check.* Run a Collatz simulator on all $n \le N_0$, confirming that each trajectory reaches 1.

The same structure can be mirrored inside a proof assistant, with the external simulator replaced by a formally verified function over natural numbers.

10 Related Work

There is a long tradition of computational verification for the Collatz conjecture, with ranges extended to astronomical values by optimized implementations of the original map and its accelerated variants. These computations demonstrate that no counterexample occurs below very large bounds, but they are usually not accompanied by a proof that the stopping-time bounds propagate to infinity.

Analytic work has studied the statistical behavior of Collatz orbits, often via parity vector analysis and heuristic probabilistic models. These approaches treat the sequence of valuations $\nu_2(3n+1)$ as approximately independent and derive expectations for total stopping times. They support the view that the map is "on average" contracting, but they do not directly yield a proof that every individual trajectory converges.

A third line of work introduces energy or Lyapunov functions, such as weighted logarithms of the iterates, and seeks to show that they decrease in the long run. Several local-to-global arguments have been proposed, combining short patterns of steps that tend to decrease height with coarse control on the frequency of such patterns. However, without a complete finite

catalog of patterns and their coverage, these arguments remain conditional on unproven assumptions about the distribution of valuations.

The present paper can be viewed as a synthesis of these ideas: it uses exact parity and valuation information to construct a finite catalog of genuinely contracting patterns (windows), and it combines this with explicit residue-level control (funnels) to obtain a global Lyapunov descent. The key difference is that every step is encoded in a finite certificate, making the proof unconditional and fully checkable.

11 Discussion and Conclusion

11.1 Why modulus 2^{18} suffices

The choice of modulus 2^{18} is not canonical, but it strikes a practical balance. On the one hand, a larger modulus refines the residue classes and allows more detailed control over the dynamics of T; on the other hand, increasing the modulus multiplies the number of residues and the size of the certificate. The search described here shows that at 2^{18}, with window lengths $j \leq 10$ and funnels of length $L \leq 16$, it is already possible to cover every residue.

Nothing in the argument fundamentally restricts us to 2^{18}. It is plausible that smaller moduli could suffice if deeper windows are allowed, or that larger moduli could yield smaller global constants J and N_0, but those optimizations are orthogonal to the central point: there exists at least one finite modulus at which a complete window–funnel certificate can be constructed.

11.2 Tradeoffs between M, j, and L

The parameters M, $j_{\max}$, and L interact in natural ways:

- Increasing M increases the number of residue classes but can reduce the required funnel length L, as residues can be distinguished more finely and matched to stronger windows.
- Increasing $j_{\max}$ allows more aggressive contractions (A significantly below 1), which can reduce the thresholds N_0 and therefore the global N_0, at the cost of more complex windows.
- The uniform step bound $J^{=j_{\max}+L}$ suggests that both $j_{\max}$ and L should be kept modest; too large a value in either dimension increases the worst-case time to see a strict decrease in E.

The certificate in this paper shows that relatively small values of all three parameters already suffice for a proof.

11.3 Generality of window–funnel certificates

The window–funnel pattern is not specific to the Collatz map. Any discrete dynamical system on the integers or on a finite residue class space, with local update rules and a natural notion of height, admits an analogous strategy:

1. Identify short patterns of steps that yield a uniform contraction in an energy function E.

2. Encode these patterns as windows on congruence or state classes, with explicit affine or other functional forms.

3. Construct short funnels from arbitrary states to the window set using the induced dynamics on a finite quotient.

4. Combine the windows and funnels to exhibit a global descent in E above a finite threshold.

This template could be applied to other problems in arithmetic dynamics or cellular automata, provided that the local update rules admit exact algebraic descriptions and that the state space mod 2^M (or another modulus) can be explored exhaustively.

11.4 Future directions

Several directions remain open even after an unconditional proof is established:

- *Tightening J^* and N_0^*.* More refined searches, higher moduli, or adaptive strategies for choosing windows could reduce the global bounds and yield a more efficient certificate.

- *Structural understanding.* The finite catalog reveals detailed information about which residue classes are "hard" and which contract quickly. Analyzing this structure may shed light on the deeper arithmetic patterns behind the Collatz map.

- *Formal proof artifacts.* A fully integrated formalization in a proof assistant, together with the CSV files and validator scripts, would create a durable, inspectable artifact of the proof, making it accessible to both humans and machines.

The main conclusion of this work is that the Collatz conjecture can be resolved by combining exact local structure with a finite global certificate. The window–funnel method offers a concrete route from local compression to global convergence and illustrates how computational search, when packaged into finite verifiable data, can support a complete and unconditional mathematical proof.

Appendices

A Exact 2-adic derivation of window congruences

Starting from an odd integer n_0, the accelerated map T is defined by

$$n_{t+1} = T(n_t) = \frac{3n_t + 1}{2^{s_t}}, \qquad s_t := \nu_2(3n_t + 1).$$

We can write the t-th iterate in the form

$$n_t = \frac{3^t n_0 + c_t}{2^{K_t}},$$

where $K_0 = 0$, $c_0 = 0$, and $K_t = \sum_{i=0}^{t-1} s_i$.

To derive the recursion for c_t, compute

$$3n_t + 1 = 3 \cdot \frac{3^t n_0 + c_t}{2^{K_t}} + 1 = \frac{3^{t+1} n_0 + 3c_t + 2^{K_t}}{2^{K_t}}.$$

By construction,

$$3n_t + 1 = 2^{s_t} n_{t+1} = 2^{s_t} \cdot \frac{3^{t+1} n_0 + c_{t+1}}{2^{K_{t+1}}} = \frac{3^{t+1} n_0 + c_{t+1}}{2^{K_{t+1} - s_t}},$$

and since $K_{t+1} = K_t + s_t$, the denominators agree. Equating numerators gives

$$3^{t+1} n_0 + 3c_t + 2^{K_t} = 3^{t+1} n_0 + c_{t+1},$$

so

$$c_{t+1} = 3c_t + 2^{K_t}.$$

The valuation condition $\nu_2(3n_t + 1) = s_t$ states that $3n_t + 1$ is divisible by 2^{s_t} and not by 2^{s_t+1}. Using the expression above, this is equivalent to

$$3^{t+1} n_0 + 3c_t + 2^{K_t} \equiv 2^{K_{t+1}} \pmod{2^{K_{t+1}+1}},$$

since $K_{t+1} = K_t + s_t$. Writing $n_0 \equiv r \pmod{2^{K_{t+1}+1}}$, we obtain

$$3^{t+1} r \equiv -\big(3c_t + 2^{K_t}\big) + 2^{K_{t+1}} \pmod{2^{K_{t+1}+1}}.$$

Because 3^{t+1} is odd, it is invertible modulo $2^{K_{t+1}+1}$, so for each t this congruence has a unique solution modulo $2^{K_{t+1}+1}$. Solving these congruences in succession for $t = 0, \ldots, j-1$ yields a unique residue $r \pmod{2^{K+1}}$ on which the valuation pattern $s = (s_0, \ldots, s_{j-1})$ occurs, and hence an exact window witness.

B CSV schemas and example rows

The certificate is encoded in two main CSV files, one for windows and one for funnels. This section sketches their schemas and shows example rows.

Window catalog

Each row of the window CSV has the following fields:

- `target_residue_mod_262144`: an odd integer in $\{1, \ldots, 2^{18} - 1\}$ representing R mod 2^{18}.
- `exact_residue_modulus`, `exact_residue`: the exact 2^{K+1}-residue solving the congruences.
- `j`: an integer window length $1 \le j \le j_{\max}$.

- `K`: an integer valuation sum $K \geq 1$.
- `s_vec`: a finite sequence of nonnegative integers $(s_0, \ldots, s_{j-1})$ with $\sum s_i = K$, encoded as a list such as `[2,1,3]`.
- `A`: a rational or floating-point approximation to $3^j/2^K$.
- `B`: a rational or floating-point approximation to $c_j/2^K$, where c_j is reconstructed by the recursion.
- `N0`: a rational or floating-point approximation to $B/(1-A)$.

An example row might look like:

```
target_residue_mod_262144,exact_residue_modulus,exact_residue,j,K,s_vec,A,B,N0
12345,1024,321,4,7,"[2,1,2,2]",0.421875,5.25,9.0869565217
```

Funnel catalog

Each row of the funnel CSV stores the minimal funnel length for a residue:

- `odd_residue_mod_262144`: an odd integer $R \bmod 2^{18}$.
- `min_funnel_length`: the minimal integer $d_R \in \{0, \ldots, 16\}$ such that $\widetilde{T}^{d_R}(R) \in W$.

An example row might be:

```
odd_residue_mod_262144,min_funnel_length
12345,3
```

indicating that applying the residue map three times leads from R into the window set W.

Generated statistics

The scripts also emit summary files:

- `windows.stats.json`: counts of projected windows per j and K, coverage fraction (here 89.94%).
- `funnels.hist.json`: histogram of minimal funnel lengths d_R.
- `summary.json`: J^*, N_0^*, SHA-256 hashes of the CSVs.

These auxiliary files are optional for the proof but convenient for independent verification and exploratory analysis.

C Pseudocode for validators and Lean outline

Pseudocode for window validation

```
for each row in window_csv:
    read R, j, K, s_vec, A, B, N0
    # Recompute K and c_j
    K_t = 0
    c_t = 0
    for t from 0 to j-1:
        s_t = s_vec[t]
        # Check K_t + s_t consistency
        K_next = K_t + s_t
        # Check valuation congruence
        lhs = (pow(3, t+1) * R + 3*c_t + (1 << K_t)) mod (1 << (K_next + 1))
        assert lhs == (1 << K_next)
        # Update
        K_t = K_next
        c_t = 3*c_t + (1 << (K_t - s_t))
    assert K_t == K
    # Check affine parameters
    A_check = pow(3, j) / pow(2, K)
    B_check = c_t / pow(2, K)
    assert abs(A - A_check) < tolerance
    assert abs(B - B_check) < tolerance
    assert A < 1
    N0_check = B / (1 - A)
    assert abs(N0 - N0_check) < tolerance
```

Pseudocode for funnel validation

```
for each row in funnel_csv:
    read R, d_R
    if d_R == 0:
        assert R appears in window_csv
    else:
        Rk = R
        for k from 1 to d_R:
            Rk = residue_T(Rk)  # accelerated T mod 2^18
            if k < d_R:
                assert Rk not in window_csv
        assert Rk in window_csv
```

Lean tactic outline

In Lean, one can proceed as follows:

1. Define the accelerated map on integers:

   ```
   def T (n : Nat) : Nat :=
   let a := padic_val_nat 2 (3*n+1) in (3*n+1) / 2^a
   ```

   and its residue version modulo 2^{18}.

2. Represent the window and funnel catalogs as finite maps from residues to records. These may be imported from the CSV files or hard-coded.

3. Prove a generic lemma that any record satisfying the congruences induces the affine identity $T^{[j]}n = A * n + B$ on the corresponding congruence class.

4. For each entry in the window map, apply this lemma to obtain a theorem instance.

5. For each residue R, prove that iterating the residue map d_R times lands in the domain of the window map.

6. Combine these with a formal version of the energy argument (logarithmic height and well-foundedness) to conclude that Collatz holds for all n.

These steps can be organized into a single tactic or meta-program that parses the CSVs, generates the necessary lemmas, and composes them into the final statement.

D Additional statistics and histograms

The finite certificate contains a wealth of numerical information about the Collatz dynamics at modulus 2^{18}. In this appendix I summarize several empirical distributions derived from the window and funnel catalogs.

Window lengths and thresholds

From the window CSV one can gather:

- A histogram of window lengths j, indicating how many anchor residues are covered by windows of each length.
- A histogram of contraction ratios A and offsets B, illustrating how strong the contractions are across different windows.
- A histogram of thresholds N_0, showing the distribution of entry heights for which each window guarantees a downward step.

In the present certificate, most windows have relatively small j (with a sharp cutoff at $j = 10$) and thresholds N_0 bounded by a few tens before funnel amplification. The file `windows.stats.json` contains the exact counts by j and K together with the direct-coverage percentage (89.94%).

Funnel lengths

From the funnel CSV one can compute:

- The distribution of minimal funnel lengths d_R, i.e. how many residues hit the window set in $0, 1, 2, \ldots, 16$ steps.
- The cumulative coverage as a function of L, showing how quickly the coverage fraction approaches 1 as the funnel depth is increased.

Empirically, a large fraction of residues funnel into W within $L \leq 7$, and full coverage is achieved by $L = 16$; the distribution is summarized in `funnels.hist.json`.

Combined measures

Combining the window and funnel data, one can also study:

- The effective thresholds $2^{d_R} N_{0,R}$ per residue, giving a refined bound on the initial value above which a reduction is guaranteed for that residue.
- The minimal total step counts $d_R + j_R$ needed to see a decrease in E from each residue, and the distribution of these counts.

These statistics are not needed for the proof itself but offer a detailed picture of how the Collatz dynamics behave at the level of residue classes. They may help guide further refinements of the certificate or inspire analogous approaches to other discrete dynamical systems.