

# Structure Computer: exact, certified answers to structural questions about discrete dynamical systems

Yuta Ogura  
Independent researcher

Version 1 — October 2026

## Abstract

Structure Computer answers structural questions about discrete dynamical systems given by rational maps—first among them “after how many steps, if ever, does state  $s$  reach state  $t$ ?”—exactly, over finite fields and over the rationals, and attaches to every answer a certificate that an independent checker verifies without trusting the program that computed it. For the Lyness map, the Kahan–Hirota–Kimura discretisation of the Euler top, area-preserving linear maps and a Kahan discretisation of hyperbolic geodesics, the answer is computed from theorems proved in Lean, as a discrete logarithm on a group attached to the level set of the conserved quantities: orbits of  $10^{17}$  states are settled in a fraction of a second, including the answer “never”, which over  $\mathbb{Q}$  no iteration can give. The same procedure (compute, certify, check independently, and label each result with the kind of guarantee it carries) applies to eleven further questions: invertibility, period structure, integrability, discovery and verification of conserved quantities, holonomy, reconstruction of integer invariants, equivalence of paths and of systems, transport of checked results along conjugacies, and the evaluation of closed terms of a type theory in the dynamics. Systems are given by name or as short text files, and a system written in unfamiliar coordinates is recognised as a known family up to affine changes of coordinates, with the change recorded in the certificate and re-verified by the checker. For the Lyness map over  $\mathbb{F}_q$  the checker exists in a second form, written in Lean and proved sound. Long walks can optionally be certified by STARK proofs that are verified in milliseconds. In a brute-force comparison over small fields, 12,156 state pairs agreed with the truth read off the orbits and 5,950 certificates passed the checker; in a tamper test the checker rejected all 1,363 altered certificates; the Lean checker and the independent checker did not disagree on any of 2,004 certificates and 14,025 alterations that both could read. On a 2-core machine, a Lyness state  $7.9 \times 10^{15}$  steps away over  $\mathbb{F}_{2^{61}-1}$  is located, certified and checked in 0.4 s.

## Contents

|          |                                             |          |
|----------|---------------------------------------------|----------|
| <b>1</b> | <b>Introduction</b>                         | <b>2</b> |
| 1.1      | The problem . . . . .                       | 2        |
| 1.2      | Design principles . . . . .                 | 3        |
| 1.3      | Organisation . . . . .                      | 3        |
| <b>2</b> | <b>The question, the answers, the kinds</b> | <b>4</b> |

|           |                                                               |           |
|-----------|---------------------------------------------------------------|-----------|
| <b>3</b>  | <b>Architecture</b>                                           | <b>4</b>  |
| 3.1       | Components . . . . .                                          | 4         |
| 3.2       | The flow of one answer . . . . .                              | 5         |
| <b>4</b>  | <b>Certificates</b>                                           | <b>5</b>  |
| <b>5</b>  | <b>The checker</b>                                            | <b>6</b>  |
| 5.1       | What it verifies for $\text{Id}(s, t)$ . . . . .              | 6         |
| 5.2       | The Lean-verified checker . . . . .                           | 7         |
| <b>6</b>  | <b>What each answer rests on</b>                              | <b>7</b>  |
| <b>7</b>  | <b>Regularity</b>                                             | <b>7</b>  |
| <b>8</b>  | <b>Over the rationals</b>                                     | <b>8</b>  |
| <b>9</b>  | <b>System files and recognition</b>                           | <b>8</b>  |
| 9.1       | The description language . . . . .                            | 8         |
| 9.2       | Recognition as written and up to affine coordinates . . . . . | 9         |
| <b>10</b> | <b>The eleven questions</b>                                   | <b>9</b>  |
| <b>11</b> | <b>STARK proofs of long walks</b>                             | <b>10</b> |
| <b>12</b> | <b>Interfaces and bundles</b>                                 | <b>11</b> |
| <b>13</b> | <b>Evaluation</b>                                             | <b>11</b> |
| 13.1      | Correctness . . . . .                                         | 11        |
| 13.2      | Performance . . . . .                                         | 12        |
| <b>14</b> | <b>Related work</b>                                           | <b>12</b> |
| <b>15</b> | <b>Scope and roadmap</b>                                      | <b>13</b> |

# 1 Introduction

## 1.1 The problem

Consider a state that changes one step at a time under a fixed rational rule  $\sigma$ . One asks whether a given state  $t$  is reached from  $s$  and after how many steps, which quantities are constant along orbits, and whether the rule can be run backwards.

Each of the existing tools answers these questions only in part. A floating-point simulation follows an orbit for a while, but cannot compare states exactly and cannot say “never”. A computer algebra system iterates exactly over  $\mathbb{F}_q$  or  $\mathbb{Q}$ , but an orbit over  $\mathbb{F}_{2^{61}-1}$  may have  $10^{17}$  states and an orbit over  $\mathbb{Q}$  is in general infinite; and what it finds comes without evidence that a third party could check short of repeating the computation. Proof assistants prove theorems about a map; they do not compute the answer for a particular pair of states. The arithmetic methods of dynamical systems over finite fields detect integrability from the statistics of periods and say nothing about individual orbits.

Structure Computer is designed to answer these questions exactly; in time governed by the arithmetic of the group order rather than by the orbit length whenever the system belongs to one of the families for which the theory of [6] applies; with a certificate for every answer;

with an independent checker that is small and free of dependencies, and, for the Lyness map, a second checker proved correct in Lean; and with an explicit label saying what kind of guarantee each answer carries. Table 1 compares it with the existing tools.

|                                         | exact | certificate  | “never”                   | large $\mathbb{F}_q$ | $\mathbb{Q}$ | formal proofs | speed             |
|-----------------------------------------|-------|--------------|---------------------------|----------------------|--------------|---------------|-------------------|
| floating-point simulation               | no    | no           | no                        | –                    | no           | no            | very fast         |
| computer algebra (iteration)            | yes   | no           | small $\mathbb{F}_q$ only | no                   | no           | no            | iteration-bound   |
| proof assistant                         | yes   | proof        | for stated theorems       | –                    | –            | yes           | no computation    |
| arithmetic dynamics (period statistics) | yes   | no           | no                        | statistical          | no           | no            | fast              |
| linear-map orbit certificates [4]       | yes   | yes          | yes                       | linear only          | no           | no            | iteration-bound   |
| Structure Computer                      | yes   | yes, checked | yes                       | yes                  | yes          | yes           | group-order-bound |

Table 1: What each kind of tool offers for the orbit question. “Large  $\mathbb{F}_q$ ” means answering without iterating the orbit for  $q$  up to about  $2^{61}$  (about  $2^{45}$  for the Euler top); “ $\mathbb{Q}$ ” means unconditional negative answers over the rationals when some prime separates the states.

## 1.2 Design principles

**Compute, then check.** The engine that produces an answer is not trusted; it may use any algorithm, any heuristic and a fast native kernel, because every answer is checked afterwards by a separate program from the certificate alone. The checker is the trusted part. It is small and uses only the standard library (the verification of STARK proofs alone is delegated to the Rust verifier); for one family a second checker, written and proved sound in Lean, is available.

**Theorems proved once per family.** Lean does not run when a question is asked. The statement “if the conditions the checker confirms hold, the answer is correct” is proved once for each family [6]; the certificate cites the theorem and carries the data (orders, logarithms, witnesses) that the checker confirms.

**Every answer carries its kind.** *Certain* (a theorem applied, every hypothesis checked), *probably certain* (up to a stated error bound, from a STARK proof), *reconstructed* (unique under a stated bound), *statistical* (an estimate with its statistic) or *none* (no claim, with the reason). An answer is called certain only when it has been confirmed, and a presupposition the system could not discharge is stated in the answer.

**Local and inspectable.** Everything runs on the user’s machine; no network access is involved. Certificates are JSON with a published schema; a bundle of certificates ships with the checker and the Lean sources and is checked offline.

## 1.3 Organisation

Sections 2–6 describe the question, the architecture, the certificates and the checkers; Sections 7–12 the further machinery; Section 13 the evaluation, Section 14 related work and Section 15 scope.

## 2 The question, the answers, the kinds

A *system* is a state space  $K^n$  with a partial rational map  $\sigma$  and its partial inverse. A state is *regular* when neither its forward nor its backward orbit meets a point where the map is undefined. For states  $s, t$  the system computes

$$\text{Id}(s, t) = \{n \in \mathbb{Z} : \sigma^n s = t\},$$

all step counts forwards and backwards—the hom-set from  $s$  to  $t$  in the action groupoid of the dynamics, and the interpretation of the identity type between  $s$  and  $t$  in the groupoid model of type theory [6]. The answer is one of:

- **steps**  $(r, M)$ : for every integer  $n$ ,  $\sigma^n s = t$  if and only if  $n \equiv r \pmod{M}$ ;
- **no path**:  $\sigma^n s \neq t$  for every integer  $n$ ;
- **not regular**:  $s$  is not regular; the answer gives the length of its forward chain and the  $n$  (at most one) with  $f^n s = t$  ( $n \geq 0$ ) or  $g^{-n} s = t$  ( $n < 0$ ) on the chain,  $f, g$  being the partial map and its inverse.

Over  $\mathbb{F}_q$ ,  $q$  may be any odd prime (up to  $2^{63}$  on the native kernel, any size on the reference implementation). Over  $\mathbb{Q}$  the question is asked with a bound  $B$  on  $|n|$ ; “no path” is then either unconditional (some prime separates the states) or “within the bound”.

For  $\text{Id}(s, t)$  the kinds are *certain*, *probably certain* (only when a STARK proof stands in for a walk) and *none*. A certain answer may carry the presupposition “ $s$  and  $t$  are regular”, kept only when the system could neither walk the orbits nor decide regularity by the theorems of Section 7.

## 3 Architecture

### 3.1 Components

**Engine (Python).** Classifies the pair of states (family, level, case of the theorems), computes phases, orders and discrete logarithms, assembles certificates, and runs the questions; symbolic work (inverses, recognition) uses SymPy.

**Kernel (Rust, no external dependencies).** A separate process that communicates over a pipe in JSON lines: orbit walks, nullspaces, orders and logarithms on elliptic curves and in  $\mathbb{F}_q^\times, \mathbb{F}_{q^2}^\times, \mathbb{F}_{q^4}^\times$ , roots, factorisation, primality. Montgomery arithmetic for any odd prime below  $2^{63}$ ; the Mersenne primes  $2^{31} - 1$  and  $2^{61} - 1$  by shift-and-add. Each system states its step as a small straight-line program that the kernel executes. The kernel is untrusted like the engine; a pure-Python reference implementation of every operation remains, and the two are compared in the test suite.

**Checker (Python, standard library only).** Verifies certificates of  $\text{Id}(s, t)$  over  $\mathbb{F}_q$  and  $\mathbb{Q}$ , of the eleven questions, and of STARK statements. It never imports the engine, runs in its own process, and parses system files with its own parser.

**Lean development.** The theorems the certificates cite (1,146 theorems, no sorry, standard axioms only) and the Lean checker with its soundness proof.

**STARK prover and verifier (Rust).** Circle STARKs [3] over Mersenne-31, built on Plonky3, for “ $\sigma^n s = t$  with every step defined”, optional, at  $q = 2^{31} - 1$ .

**Interfaces.** A Python API, a line-oriented JSON protocol on standard input and output, a local HTTP service with jobs, and a command line; a web front end uses the HTTP service.

### 3.2 The flow of one answer

1. The request (system, parameters, states, field or bound, options) is validated and a seed drawn.
2. The engine computes the body of the certificate: classification, phase data, orders with factorisations, logarithms or witnesses of non-membership, regularity entries, and the claim.
3. The body is wrapped in an envelope that restates the question, claim, kind, presupposition and the theorems cited.
4. The checker, in a separate process, recomputes the case from the states, verifies every item of auxiliary data, recomputes the claim and compares.
5. If the checker rejects or the engine fails, the answer is recomputed once with a new seed on the reference implementation (over  $\mathbb{Q}$  also with other primes); if that fails too, the answer is *none* with the checker's reasons. Unchecked results are returned only on explicit request and marked as such.
6. The typed result and the checked certificate are returned.

## 4 Certificates

All certificates are JSON documents with a published JSON Schema.

**The envelope.** Every answer of every question has the same envelope: the question, the input as answered, the seed, the result (kind, value, presupposition, error bound, assumption, statistic, reason), the list of theorems cited anywhere in the body, the body, engine information and how the answer was checked. The checker verifies the body by the rules of its question and then requires the envelope to state exactly what the body proves.

**The body of  $\text{Id}(s, t)$  over  $\mathbb{F}_q$ .** The system and parameters,  $q, s, t$ ; the *method* (phase, invariant, fixed point, vacuous level, walked orbit, conjugacy, or a recognised system file); the *case* of the theorem that applies (e.g. an elliptic Lyness level, case A of the Euler top); the theorems; the claim; the presupposition, or null when discharged; the auxiliary data: the level and translation point (for the Euler top also the scaling, the cyclic shift and the sign-change selections), and for each component of the phase group—an elliptic curve, a multiplicative group of  $\mathbb{F}_q$ ,  $\mathbb{F}_{q^2}$  or  $\mathbb{F}_{q^4}$ , an additive group, or a  $\mathbb{Z}/2$  factor—the order of the translation point with its factorisation and either the logarithm or a witness of non-membership; and the *regularity* entry of each state: a walked cycle (with period), a walked chain (with its length), the phase test (no stuck state of the level in the coset, with the stuck candidates and their witnesses), or non-regularity (a stuck candidate in the coset, with its logarithm) together with the partial orbit. For a system file or a conjugacy, the certificate records the match (including the change of coordinates  $u = Mv + c$ ) and contains the family's certificate for the transformed states. Listing 1 shows the body of Example A.1 of [6] in abridged form; the whole certificate is 4.3 kB.

Listing 1: The body of a steps certificate (abridged);  $P$  is the translation point  $Q$  of [6].

```
{
  "system": "lyness", "params": {"a": 3},
  "q": 2305843009213693951, "s": [2, 5],
  "t": [813066766938016773, 992646061924517494],
  "method": "phase", "case": "lyness/elliptic",
  "theorems": [...],
  "presupposition": null,
  "data": {"k": 18, "P": [36, 2305843009213692799]},
  "components": [
    {
      "type": "ec", "order": 576460752629115698,
      "order_fac": [[2,1],[6673,1],[3612179,1],[11957747,1]],
      "log": 7863588611074867
    }
  ],
  "claim": {"kind": "steps", "residue": 7863588611074867,
    "modulus": 576460752629115698},
  "regularity": {
    "s": {"status": "phase", "k": 18,
      "stuck": [{"dir": "F", "z": [26, 2305843009213693948],
        "components": [{"type": "ec", "order":
          576460752629115698,
          "order_fac": [...], "witness": 2}]}],
      {"dir": "G", "z": [2305843009213693948, 26],
        "components": [...]}]}],
    "t": {...}},
  "kind": "certain"
}
```

**The body over  $\mathbb{Q}$ .** The rational data and the bound; the primes used, each with the certificate of the reduced question; the Chinese-remainder combination; the exact iteration of the candidate step count; the claim; and the reduction theorems cited.

**The body of the other questions.** One format shared by the eleven questions: the question, the system and its source, the inputs, the kind, the claim, the auxiliary data, the theorems, and the error bound, assumption, statistic or reason the kind requires.

## 5 The checker

### 5.1 What it verifies for $\text{Id}(s, t)$

For a certificate by the phase method the checker, from the certificate alone:

- verifies the primality of  $q$  and of every prime in a factorisation (deterministic Miller–Rabin, valid far beyond the field sizes in use; the Lean checker uses Pratt certificates instead);
- recomputes the *case* from the states—for the Lyness map the level, the curve, its discriminant, the nondegeneracy conditions and the nonsingularity of  $\varphi(s), \varphi(t)$ ; for the Euler top the scaling against its defining equations, the level, the curve and the case; for the linear systems the eigenvalue equations—and rejects a certificate whose stated case does not match;
- verifies every order ( $NP = 0$  and  $(N/\ell)P \neq 0$  for every prime  $\ell \mid N$ ; in a multiplicative group also  $N \mid |K^\times|$ ), every logarithm (by exponentiation), and every witness of non-membership (on a curve, that the difference has the wrong order in the  $\ell$ -part or

lies outside the cyclic subgroup there, by an exhaustive baby-step giant-step; in a cyclic group,  $c^N \neq 1$  for the difference  $c$ );

- combines the component classes by the Chinese remainder theorem and requires the result to equal the claimed  $(r, M)$ ; for small residues it also iterates  $\sigma^r s = t$  directly; for “no path” by the level it recomputes the invariants;
- recomputes the stuck candidates of the level itself (for the Euler top, verifying that the root list is complete by the degree of  $\gcd(P_D P_P, m^{|K|} - m)$ , where  $P_D, P_P$  are the stuck-state polynomials of [6] and checking the square roots), verifies each witness or logarithm, re-walks walked cycles and chains, and allows the presupposition to be dropped only when  $s$  is shown regular and  $t$  is shown regular or the path was iterated.

For a walked orbit it re-walks; for a system file or a conjugacy it re-derives the match by polynomial identities (Section 9) and then checks the inner certificate; for the eleven questions it applies the rules of Section 10.

## 5.2 The Lean-verified checker

For the Lyness map over  $\mathbb{F}_q$  the checker exists in a second form, written in core Lean (no Mathlib at run time) and accompanied by a soundness theorem: if it accepts a certificate, then  $q$  is prime and the claim is true of the Lyness flow over  $\mathbb{Z}/q$  as defined in the Lean development—for *steps*, that  $\sigma^n s = t \iff n \equiv r \pmod{M}$  for the regular states; for *no path*, that no  $n$  works; for *not regular*, that  $s$  is not regular with the stated chain length and partial orbit. It verifies Pratt certificates [7] with a proved Lucas test, uses the affine addition formulas proved equal to Mathlib’s group operations, and covers the elliptic and nodal levels with all regularity statuses; other cases it reports as unsupported and leaves to the independent checker. The two checkers can be run side by side on any certificate.

## 6 What each answer rests on

| answer                                                                                                               | rests on                                                                                                                                                     |
|----------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\text{Id}(s, t)$ for the Lyness map over $\mathbb{F}_q$ on elliptic and nodal levels, checked with the Lean checker | the Lean kernel and Mathlib, and the Lean checker’s soundness theorem                                                                                        |
| every other $\text{Id}(s, t)$ (reducible levels, the Euler top, linear systems, $\mathbb{Q}$ )                       | the independent checker’s recomputation of the case and the auxiliary data, and the Lean theorems that state the conclusion under exactly those conditions   |
| the eleven questions                                                                                                 | the independent checker’s rules for each question (identities over $\mathbb{Q}$ re-proved, enumerations re-stepped, ranks recomputed) and the cited theorems |
| answers with STARK proofs ( <i>probably certain</i> )                                                                | the Plonky3 verifier and the constraint system built by the checker from the system itself, with the stated error bound                                      |

Table 2: What each kind of answer rests on. Neither the engine nor the kernel appears in the list.

## 7 Regularity

A rational map is undefined where a denominator vanishes, and a state whose orbit reaches such a point is not regular and lies outside the translation theorems. The system first walks a

bounded distance in each direction; a closed orbit settles regularity, a stuck state gives “not regular” with the walked chain. For large  $q$  it applies the phase test of [6]: on a Lyness level the two stuck states are explicit,  $F = (c, -a)$  and  $G = (-a, c)$  with  $c = -(ka + 1 - a)/(1 - a)$  ( $a \neq 1$ ), and  $s$  is regular if and only if neither lies in the coset  $\varphi(s) + \langle Q \rangle$ : two discrete logarithms, or two witnesses of non-membership, decide regularity either way. For the Euler top the stuck candidates are the at most 56 states  $u_i = \pm\sqrt{1 - m\alpha_i}$  for the roots  $m$  of two polynomials of degree  $\leq 3$  and  $\leq 4$  in the multiplier  $m$ , where the level is  $\alpha \in \mathbb{P}^2$  and  $y(u) = m\alpha$  (see [6]), found by root extraction in  $\mathbb{F}_q$  or  $\mathbb{F}_{q^2}$ ;  $s$  is regular if no stuck candidate lies in its coset, and a stuck candidate in the coset that is stuck yet takes two defined steps in the opposite direction shows it not regular. In the remaining cases the system walks a few steps and, if that does not settle the matter, states the presupposition in the answer.

For a Lyness state shown not regular by the phase, the answer includes its partial orbit: the forward chain ends at the stuck state of least non-negative logarithm, and  $t$  is reached exactly when its own logarithm is defined and smaller (Example A.2 of [6]: a chain of  $8.8 \times 10^{17}$  states on which  $t$  is state number  $4.0 \times 10^{17}$ ). The checker recomputes the stuck candidates and re-derives the chain ends.

## 8 Over the rationals

For rational data the system takes primes above a configurable starting point, skipping primes at which the data are not  $p$ -integral or the reduced  $s$  is not shown regular, and answers the reduced question at each. A per-prime “no path” ends the search: by the reduction theorem of [6], the states are never joined over  $\mathbb{Q}$ . Otherwise the residue classes are combined by the Chinese remainder theorem until the modulus exceeds twice the bound; the single remaining candidate  $n$  is confirmed by exact rational iteration, which the checker repeats, or excluded. For a per-prime “no path” to count, the checker requires the reduced  $s$  to be shown regular at that prime—exactly the hypothesis of the reduction theorem. A starting point above  $2^{31}$  is convenient, since the phase test then usually settles regularity at the first prime.

## 9 System files and recognition

### 9.1 The description language

Besides the named systems (Lyness with  $a$ ; the Euler top (Kahan–Hirota–Kimura) with  $a_1, a_2, a_3, h$ ; the oscillator with  $c$ ; E2 (the Kahan discretisation of hyperbolic geodesics) with its quadratic form; and the Lyness-type, reversed Lyness, linear, parabolic and translation maps, answered through conjugacies), a system is a short text file:

```
name: a rational map of the plane
vars: x, y
params: a = 3, b = 2
map:
  x' = y
  y' = (a + b*y)/x
```

The step is given in one of six ways: a map with an optional inverse; a quadratic vector field with a Kahan step size; use of a named system; a potential with an exponent  $\alpha$  (the Kahan step of the  $\alpha$ -geodesic flow of the Hessian metric of the potential, which may contain logarithms); a lagrangian (the discrete Euler–Lagrange step); or several generator blocks (a group action). Optional blocks declare a prime policy, polynomial constraints, observables, symmetries (matrices or a quadratic form) and questions to ask. The semantics is the partial

flow of the formulas as written:  $\sigma(x) = F(x)$  where  $F$  is defined at  $x$ , the inverse  $G$  at  $F(x)$ , and  $G(F(x)) = x$ ; a division by zero is undefined. A missing inverse is computed and recorded in the certificate. Static checks run before any computation: the singular set, the degree, the regularity of the Kahan, geodesic or Legendre matrix, invariance under the declared symmetries, and the square roots per prime.

## 9.2 Recognition as written and up to affine coordinates

A file whose formulas, as written, are those of a known family and are undefined exactly where the family is, is answered by the family’s theorems; the certificate records the match and contains the family’s certificate. The engine fits candidates at sample points and accepts one only when the numerators and denominators agree as polynomials and the divisors match; the checker re-parses the file and re-checks both.

A file that is a known family only in other coordinates is found too: the Lyness map after any affine change of coordinates; an area-preserving affine planar map with a fixed point (the linear map there, or the parabolic normal forms); and a quadratic field that is the Euler top in affine coordinates, whose centre is the point where every partial derivative vanishes and whose axes are the zeros of the homogeneous part in  $\mathbb{P}^2$ . The candidate transform is accepted only when the transformed formulas are the family’s as written with the same undefined set, and the certificate records  $u = Mv + c$ ; the checker substitutes  $v = M^{-1}(u - c)$  and verifies the polynomial identities. Projective or birational conjugacies are checked when supplied (Section 10, Q9/Q10). Formulas should be written in reduced form, because the checker re-derives every identity by expanding them; formulas with common factors cancelled are checked in seconds, uncanceled ones in minutes.

## 10 The eleven questions

Besides  $\text{Id}(s, t)$  the system answers eleven structural questions through the same flow. Table 3 lists them.

Table 3: The eleven questions and how each is certified.

|    | question         | what is computed; what the checker verifies                                                                                                                                                                                                                                                                                                                                             |
|----|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Q1 | invertibility    | an inverse $G$ with $G \circ F = \text{id}$ , $F \circ G = \text{id}$ as rational identities over $\mathbb{Q}$ (certain); or a reason the map is not birational—degree $\neq 1$ in one variable, vanishing Jacobian, or a deck transformation (certain); else collision counts modulo small primes (statistical). The checker re-proves the identities and recomputes the singular set. |
| Q2 | period structure | over $\mathbb{F}_q$ : full enumeration for small state spaces (certain per prime), else sampled orbits (statistical), each with a proof tag (walked, a phase certificate, or a STARK proof). The checker re-steps every state of an enumeration and re-verifies the samples.                                                                                                            |
| Q3 | integrability    | conserved quantities (Q4) and their Jacobian rank: “integrable” when the rank is $\geq n - 1$ (certain); otherwise the Roberts–Vivaldi period statistics [8] over several primes, with a reversor search (statistical).                                                                                                                                                                 |

|     | question               | what is computed; what the checker verifies                                                                                                                                                                                                                                                                                                                                          |
|-----|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Q4  | conserved quantities   | polynomial invariants up to a degree, found as nullspaces modulo primes near $2^{62}$ , lifted by CRT and rational reconstruction [10], and verified as identities $P \circ F = P$ over $\mathbb{Q}$ (certain), with completeness evidence and rational invariants by invariant-curve and Darboux-polynomial methods. The checker re-proves every identity and recomputes the ranks. |
| Q5  | verification           | a given formula is conserved (an identity, certain) or a counterexample point modulo $q$ is given (certain).                                                                                                                                                                                                                                                                         |
| Q6  | holonomy               | for a system with an orthogonal symmetry, the first return of an orbit and the group element realising it, classified, with its length, twist or rotation as discrete logarithms and its order; for $4 \times 4$ Lorentzian forms a $2 \times 2$ (spinor) representative obtained through the Hodge star (certain; probably certain with a STARK proof).                             |
| Q7  | integer invariants     | an integer step count, winding number or translation recovered from its residues modulo several primes under a stated bound (reconstructed; certain when the residues leave no admissible value within the bound).                                                                                                                                                                   |
| Q8  | equivalence of paths   | whether two words in the generators act equally: by exponent sums, by composing formulas, or by a differing point (certain), else by sampling (statistical).                                                                                                                                                                                                                         |
| Q9  | equivalence of systems | a conjugacy, found up to affine maps or supplied as a projective or birational map, verified as identities (certain), refuted by invariant fingerprints (dimension, step hash, Jacobian, invariant basis, period decompositions), or left statistical.                                                                                                                               |
| Q10 | transport              | a checked result of one system moved to another along a conjugacy $\varphi$ whose kind is <i>certain</i> (invariants composed with $\varphi^{-1}$ , states mapped, holonomy conjugated) and re-checked in full together with its source.                                                                                                                                             |
| Q11 | evaluation             | the value of a closed term of the dynamics theory $(X, s, t, \text{step}, \text{path})$ in the groupoid model: $\text{step}^n s \mapsto \sigma^n s_0$ (certain).                                                                                                                                                                                                                     |

A *library* of systems (two Hénon maps, a radial Störmer–Verlet Kepler step, the McMillan map, a symmetric QRT map, a cubic-kick standard map, a discrete Toda lattice) comes with checked certificates of Q1–Q4; a query system is matched against it by fingerprint and, when Q9 finds a certain conjugacy, the stored certificates are transported by Q10.

## 11 STARK proofs of long walks

At  $q = 2^{31} - 1$  the orbit, period and holonomy questions accept an option by which a walk the checker would otherwise repeat is proved by a Circle STARK [3] and the checker verifies the proof instead. The statement is “ $\sigma^n s = t$  with every step defined”, expressed as a trace with a step counter, inverse witnesses for every division, and the inverse map applied to the next row returning the current state; options express a minimal period, avoidance of a state, a return, or invariants. The constraint system is built by the checker from the system itself, so the prover cannot choose what is proved. An answer resting on a STARK proof is *probably certain* with the soundness error of the proofs ( $\approx 2^{-99}$  per proof) as its error bound. For a walk of  $2^{20} - 1$  Lyness steps the proof is 498 kB, takes 132 s to produce and 0.075 s to verify, against 14 s for the pure-Python checker to re-walk;  $2^{16} - 1$  steps of E2 in six variables,

proved in 12 s and verified in 0.05 s against 2.1 s. The verifier rejected 13 of 13 altered inputs and the checker caught 25 of 25 alterations of certificates with proofs.

## 12 Interfaces and bundles

Each request yields one response: a JSON request naming the system (by name and parameters, or by the text of a system file), the question and its inputs returns the typed result (kind, value, presupposition, error bound, assumption, statistic, reason) with its checked certificate. Long computations run as jobs (submit, status, result, cancel; each in its own process). The protocol is served over standard input and output, one JSON line per request, and over HTTP on the loopback interface with optional bearer authentication. The Python API offers the same with typed result classes—`Certain`, `ProbablyCertain`, `Reconstructed`, `Statistical`, `NoResult`—and a function that raises on anything weaker than a certain answer; batches run on a process pool. Integers beyond  $2^{53}$  are exact end to end. A *bundle* packs certificates with the checker, the schema, the Lean sources, a manifest of the cited theorems with their axioms, and checksums, and is checked offline by its own checker.

## 13 Evaluation

### 13.1 Correctness

- **Brute force.** Over  $\mathbb{F}_q$  for  $q = 11, 13, 17, 19, 23$ , every state of every family and parameter set is classified by walking; for every case of the analysis, regular states  $s$  are sampled with targets  $t$  from the orbit of  $s$ , from its level and from anywhere; 30% of the queries skip the walk so that regularity is established by the phase test, and non-regular states are sampled too. Result: 12,156 pairs, no mismatch with the truth read off the orbits; 5,950 certificates through the checker, no failure.
- **Tamper test.** 200 valid certificates, each altered in several ways, one place at a time (the claimed class, a logarithm, an order, a witness, the scaling, a root, the case, a state, a stuck candidate, a square root, a regularity status): 1,363 alterations, all rejected; no original rejected.
- **Lean checker against independent checker.** 2,004 certificates and 14,025 alterations: no disagreement on any certificate both could read; the Lean checker accepted 1,842 originals and rejected 12,632 alterations.
- **Large fields and  $\mathbb{Q}$ .** At  $q = 2^{31} - 1$ , 48 queries, none undecided, the presupposition discharged by the phase test in 24; over  $\mathbb{Q}$ , 22 examples with primes above 10007 and above  $2^{31}$ , all answered and checked.
- **Kernel against reference.** Steps, walks, orders, logarithms, factorisations and nullspaces of the native kernel against the pure-Python reference at primes up to 9223372036854775783: all equal.
- **Manifest and axioms.** Every theorem cited by the engine exists in the Lean development with the standard axioms (241 cited theorems); the audit of every theorem of the development reports no other axiom.
- **End to end.** From a fresh installation: 26 answers checked against 10,497 ground-truth comparisons, 27 false claims rejected, 25 jobs run, and 24 stages run through HTTP and the API, all without error.

The brute-force, tamper, large-field,  $\mathbb{Q}$ , kernel and STARK tests were re-run for this paper on 22 October 2026 on a 2-core Intel Xeon at 2.1 GHz with the kernel and the prover built from source, and reproduced these figures exactly; the Lean comparisons are from the full regression of 21 October 2026.

### 13.2 Performance

Table 4 gives the ten advanced examples of [6], Appendix A, recomputed for this paper (median of three runs; wall-clock for one command including start-up, computation and the independent check, of which the check is up to 0.2 s). For comparison, the kernel walks Lyness orbits over  $\mathbb{F}_{2^{61}-1}$  at  $2.6 \times 10^6$  steps/s and Euler-top orbits over  $\mathbb{F}_{2^{40}+15}$  at  $2.4 \times 10^6$  steps/s on this machine. The other questions take 0.1–0.6 s (Q3 with its period statistics, 12 s). As a control, a floating-point iteration of the Lyness map with  $a = 1$  (period 5) for 200,000 steps comes within  $5.6 \times 10^{-17}$  of  $s$  at the multiples of 5 but equals it at none of the 40,000: so it cannot decide  $\sigma^n s = t$ .

|    | system, field                                                           | answer                                                   | by iteration        | time [s] |
|----|-------------------------------------------------------------------------|----------------------------------------------------------|---------------------|----------|
| 1  | Lyness $a=3$ , $\mathbb{F}_{2^{61}-1}$                                  | $n \equiv 7.86 \times 10^{15}$ ( $5.76 \times 10^{17}$ ) | 96 y                | 0.39     |
| 2  | same                                                                    | not regular; $t$ at $n = 4.04 \times 10^{17}$            | $1.1 \times 10^4$ y | 18.4     |
| 3  | same, same level                                                        | no path                                                  | 2,300 y             | 0.48     |
| 4  | Lyness $a=3$ , $\mathbb{Q}$ , $(3, 7) \rightarrow (7, 3)$               | no path (prime 10009)                                    | never               | 0.19     |
| 5  | Euler top $(1, 2, -3, \frac{1}{2})$ , $\mathbb{F}_{2^{40}+15}$          | $n \equiv 1.37 \times 10^{11}$ ( $2.75 \times 10^{11}$ ) | 16 h                | 0.35     |
| 6  | same, $\mathbb{Q}$                                                      | no path (prime 10039)                                    | never               | 0.26     |
| 7  | same, $\mathbb{F}_{2^{40}+15}$ ,<br>$(3, 1, 4) \rightarrow (-3, -1, 4)$ | no path                                                  | 32 h                | 0.42     |
| 8  | plane map (affine Lyness), $\mathbb{F}_{2^{61}-1}$                      | as 1                                                     | 96 y                | 0.44     |
| 9  | same, $\mathbb{Q}$                                                      | no path                                                  | never               | 0.26     |
| 10 | 3-variable field (affine Euler top),<br>$\mathbb{F}_{2^{40}+15}$        | as 5                                                     | 16 h                | 0.41     |

Table 4: The advanced examples. “By iteration” is the time to establish the answer by iterating at the measured rate (for “no path” over  $\mathbb{F}_q$ , one full orbit; over  $\mathbb{Q}$ , no finite iteration suffices).

Example 2 is the only slow case; the time goes into computing the group order of its level, where the translation point has order  $3 \cdot 29 \cdot 859 \cdot 30854415206297$ , and the check itself takes at most 0.2 s. Field sizes practical today are  $q \leq 2^{61}$  for the Lyness map and  $q \leq 2^{45}$  for the Euler top, whose curves may be defined only over  $\mathbb{F}_{q^2}$ .

## 14 Related work

*Certifying algorithms* [5] return with each output a witness that a simple checker verifies; our certificates are of this kind, with two additions: the checker’s acceptance condition is the hypothesis of a theorem proved in Lean, and for one family the checker itself is proved. Verified checkers for certificates of untrusted solvers exist for SAT [11], for termination [9], and recently for Markov decision processes [2]; to our knowledge, orbits of rational maps, with elliptic-curve arithmetic inside the certificate, have not been treated before. The combination of computer algebra that finds witnesses with a proof assistant that checks them is also the pattern of Mathlib’s `polyarith`. Succinct proofs of long iterations by STARKs are standard in verifiable delay functions [1] and in proof systems for virtual machines; here they are an optional accelerator of the checker, never the ground of a *certain* answer. Certificates of reachability times for *linear* maps over  $\mathbb{F}_p$  with an independent checker have appeared recently [4]; the present system covers non-linear integrable maps over extension fields and

over the rationals, uses elliptic-curve arithmetic, rests on formally verified theorems, and answers without iterating. The arithmetic detection of integrability of Roberts and Vivaldi [8] is used by Q3 as one of its statistical criteria. The mathematics is in [6].

## 15 Scope and roadmap

The theorems cover four families and their affine conjugates; other systems are answered by bounded iteration. Recognition is up to affine changes of coordinates, with projective and birational conjugacies verified when supplied. The Euler top is practical to  $q \approx 2^{45}$ . Planned work: extending the theorems to further QRT maps and higher-dimensional Kahan discretisations, point counting for the Euler top, and extending the Lean-verified checker to the remaining families and to  $\mathbb{Q}$ .

## Use of AI

This system and this paper were produced with an AI assistant (Claude, Anthropic) in a central role. The author set the goals, decided what to build, made the design decisions and reviewed the results; under the author’s direction the AI assistant wrote most of the engine, the kernel, the checkers, the Lean development, the tests and the text of this paper, and took the measurements reported in Section 13. The author is responsible for the content.

## Acknowledgements

The system builds on Lean 4, Mathlib, HoTTLean, SymPy and Plonky3; the author thanks their contributors.

## References

- [1] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Advances in Cryptology – CRYPTO 2018*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, 2018. doi: 10.1007/978-3-319-96884-1\_25.
- [2] Krishnendu Chatterjee, Tim Quatmann, Maximilian Schäffeler, Maximilian Weininger, Tobias Winkler, and Daniel Zilken. Fixed point certificates for reachability and expected rewards in MDPs. arXiv:2501.11467, 2025.
- [3] Ulrich Haböck, David Levit, and Shahar Papini. Circle STARKs. Cryptology ePrint Archive, Paper 2024/278, 2024.
- [4] Ruge Lin. Algebraic-Loop-Certificates: certificates for reachability times and periods of linear maps over  $\mathbb{F}_p$ . <https://github.com/GoGoKo699/Algebraic-Loop-Certificates>, 2026. Software repository.
- [5] Ross M. McConnell, Kurt Mehlhorn, Stefan Näher, and Pascal Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011. doi: 10.1016/j.cosrev.2010.09.009.
- [6] Yuta Ogura. Step counts in integrable rational maps as discrete logarithms: phases, regularity and reduction, formalised in Lean. Companion paper, same site, 2026.

- [7] Vaughan R. Pratt. Every prime has a succinct certificate. *SIAM Journal on Computing*, 4(3):214–220, 1975. doi: 10.1137/0204018.
- [8] John A. G. Roberts and Franco Vivaldi. Arithmetical method to detect integrability in maps. *Physical Review Letters*, 90(3):034102, 2003. doi: 10.1103/PhysRevLett.90.034102.
- [9] René Thiemann and Christian Sternagel. Certification of termination proofs using CeTA. In *Theorem Proving in Higher Order Logics – TPHOLs 2009*, volume 5674 of *Lecture Notes in Computer Science*, pages 452–468. Springer, 2009. doi: 10.1007/978-3-642-03359-9\_31.
- [10] Paul S. Wang, M. J. T. Guy, and J. H. Davenport. P-adic reconstruction of rational numbers. *ACM SIGSAM Bulletin*, 16(2):2–3, 1982. doi: 10.1145/1089292.1089293.
- [11] Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Theory and Applications of Satisfiability Testing – SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer, 2014. doi: 10.1007/978-3-319-09284-3\_31.