SumProductAtlas 2.0: a proof-producing database of sum–product pairs up to $n \leq N_{\max}$ with completeness in structural strips

Liz Lemma Future Detective

January 16, 2026

Abstract

We introduce SumProductAtlas 2.0, a public, reproducible, proof-producing atlas of sum–product data for finite sets. For $A \subset \mathbb{N}$ we study the pair $(|A+A|, |AA|)$, where $A + A = \{a + b : a, b \in A\}$ and $AA = \{ab : a, b \in A\}$. Motivated by recent large-scale visualizations and conjectures on the geometry of attainable pairs, we treat computation as a theorem: we release (i) a deterministic verifier with independently checkable certificates, (ii) enumerators based on explicit structural descriptions for small sumset/product set, and (iii) a versioned database with provenance.

For each $n \leq N_{\max}$ we certify all attainable pairs in the additive strip $|A + A| \leq 3n - 3$ for integer sets, and we certify all attainable pairs in the multiplicative strip $|AA| \leq 3n-3$ for sets of positive reals, producing canonical witnesses in each case. In addition we provide certified upper bounds for $\alpha_n := \min_{|A|=n} \max\{|A + A|, |AA|\}$ for all $n \leq N_{\max}$ together with witness sets and verification logs. The atlas is designed for community extension via a submission-and-validation protocol. We conclude with a short list of new conjectures suggested by the verified data, stratified by domain ($\mathbb{N}$ versus $\mathbb{R}_+$) and robust to normalization choices.

Table of Contents

4. 4. Structural enumeration in the additive strip: the Freiman-type container families for $|A + A| \leq 3n - 3$; canonical parameterization; enumeration algorithm $\mathsf{EnumAdd}(n)$; what is proved vs what is computer-audited.

5. 5. Structural enumeration in the multiplicative strip over $\mathbb{R}_+$: geometric-progression models for $|AA| \leq 3n - 3$; algebraic ratios via sparse-polynomial constraints; enumeration algorithm $\mathsf{EnumMul}(n)$; certification of algebraic computations.

6. 6. The database: schema, stored artifacts (witness, certificate, provenance, canonical form), compression and indexing by (n, i, j) and by normalized coordinates; change-log and versioning.

7. 7. Main completeness theorems and their computational audits: statements, proof sketches, and audit summaries (counts, hashes, reproducible run scripts).

8. 8. Certified bounds on α_n and extremal witness tables: how witnesses are generated; how certificates are produced; limitations (upper bounds vs optimality).

9. 9. Conjecture mining and 'River of Ignorance' benchmarks: derived statistics (frontiers, gaps, candidate void inequalities), benchmark tasks for generators, and a roadmap for $n > N_{\max}$.

1 Introduction and goals

For a finite set A in a commutative semiring (typically an additive subgroup of $\mathbb{R}$ together with the usual multiplication), two derived sets encode the basic additive and multiplicative complexity of A, namely the sumset $A+A$ and the product set AA. The classical sum–product phenomenon asserts that a set cannot simultaneously exhibit very small additive doubling and very small multiplicative doubling unless it has substantial algebraic structure. In quantitative form one asks for lower bounds on $\max\{|A+A|, |AA|\}$ in terms of $|A|$, and in structural form one seeks an explicit description of witnesses that approach extremality.

A convenient way to organize this information is to record, for fixed n, the collection of *sum–product pairs*

$$(|A+A|, |AA|) \qquad (|A| = n),$$

and to view these pairs as a discrete "phase diagram" in the (i, j)-plane. Even for moderate n, plotting all realized pairs reveals a characteristic shape: a dense interior region bounded by an outer envelope, together with sparse "ridges" and "islands" corresponding to rigid structural families (near arithmetic progressions, geometric-progression models, and hybrid constructions). The lower envelope is particularly important: it governs the quantity

$$\alpha_n := \min_{|A|=n} \max\{|A+A|, |AA|\},$$

and it identifies candidate extremizers whose structure may persist as n grows. Our aim is to make this diagram concrete, reproducible, and queryable for all n up to a fixed cutoff $N_{\max}$.

The present work introduces the *SumProductAtlas 2.0*, an open, versioned database of sum–product data together with mechanisms that make each database claim independently verifiable. The database is not merely a repository of numerically computed pairs; rather, it is intended to be *proof-producing* in the following operational sense. For each stored witness A the atlas includes, alongside the claimed pair $(|A+A|, |AA|)$, a certificate that a separate checker can verify deterministically to reproduce these cardinalities exactly. This philosophy is guided by a basic constraint: in the range where exhaustive search is feasible and inverse theorems reduce the space of candidates to explicit parameter families, we should not ask the reader to trust a single implementation, a floating-point heuristic, or an opaque computation. Instead we store verifiable artifacts and insist that any published "completeness" statement be backed by auditable enumeration logs.

The atlas is built to support three complementary uses.

(i) Exact small-doubling atlases in explicit strips. The small-doubling regimes $|A+A| \le 3n-3$ and $|AA| \le 3n-3$ are the first nontrivial regions beyond the classical $3n - 4$ threshold at which inverse theorems provide strong, finitely parameterized structural descriptions. In these regimes it becomes realistic to claim *strip completeness*: for a fixed n we can, in principle, generate every witness in the relevant model families (modulo natural symmetries), compute its sum–product pair, and thereby determine exactly which pairs occur in the strip. The atlas is designed so that such claims are explicitly flagged as computer-audited and are accompanied by reproducibility data (enumerator version, canonicalization conventions, and hashes of the enumeration output). The mathematical input is an inverse description that reduces the ambient infinite search space to finitely many parameter families; the computational input is an exhaustive traversal of these families and a deterministic verification of each realized pair.

(ii) Certified upper bounds for α_n. Independently of any completeness claim, the atlas records explicit witnesses A_n giving certified values of $\max\{|A_n + A_n|, |A_n A_n|\}$. This produces certified upper bounds on α_n for each $n \le N_{\max}$. These witnesses can be mined for patterns (divisor-closed sets, smooth-number sets, mixed progression models), and they can serve as starting points for new constructions. The point here is not merely to provide a list of numerical bounds, but to provide a persistent set of checkable examples that can be used in subsequent proofs or in falsification-style searches against conjectured lower bounds.

(iii) A falsifiable framework for structural conjectures. A recurring difficulty in the sum–product literature is that conjectures about extremal structure are easy to state but hard to falsify without reliable data beyond very small n. By storing canonical witnesses and supporting queries over all realized pairs, the atlas provides a controlled environment in which to test structural hypotheses. For instance, one may ask whether all points on a certain boundary component arise from a particular parametric family, or whether certain "forbidden" pairs never occur in a given domain. Since each atlas entry carries a certificate and provenance, such investigations become falsifiable: a single certified counterexample suffices to disprove an overly rigid conjecture, while the absence of counterexamples within a declared complete strip has a clear computational meaning.

A central design decision is the separation of responsibilities among three components.

Deterministic verification. We insist on a deterministic routine that, given a witness description of A, computes the sets $A + A$ and AA exactly and produces a certificate sufficient for a second implementation to re-check the result. This is straightforward for integer witnesses but remains essential for real witnesses, where algebraic encodings and certified comparisons are required to avoid numerical ambiguity. The intended workflow is that atlas entries can be re-verified without reference to the original enumerator, and (conversely) enumerator outputs can be validated independently of the enumerator codebase.

Enumerators constrained by inverse structure. Brute-force enumeration of all n-element subsets of a large interval is infeasible well before $n = 50$. We therefore restrict attention to regimes where the set of all relevant witnesses is captured by explicit structural theorems (Freiman-type statements additively, and geometric-progression containment multiplicatively over $\mathbb{R}_+$). The enumerators do not guess; they traverse all admissible parameter instances in these families, reduce each candidate to a canonical representative to avoid duplicates, and submit it to the verifier. Any claim of equality between the set of produced pairs and the set of *all* pairs in a strip is thereby reduced to the combination of (a) the cited inverse theorem and (b) the exhaustiveness of the parameter traversal, the latter being supported by audit logs.

A versioned, tamper-evident data layer. Since the atlas is intended as a community resource, it must support incremental contributions while preserving the integrity of earlier results. We therefore treat the database as a versioned object: entries are keyed by canonical witnesses, each entry includes metadata identifying the verifier version and enumeration run, and the enumeration outputs are hashed so that independent auditors can confirm that no witnesses were silently omitted or altered. In this setting, "proof-producing" should be read as "proof-carrying data": the mathematical argument reduces correctness to a finite computation, and the computation produces artifacts that can be re-checked and re-indexed as the code evolves.

Two further remarks clarify our scope.

First, we do not aim to resolve the asymptotic sum–product problem; rather, we aim to provide a reliable finite atlas in a range where the boundary between theory and computation is well understood. The cutoff $N_{\max}$ is chosen so that exhaustive enumeration inside the relevant model families is computationally expensive but feasible with careful engineering and auditing. The atlas should be viewed as an infrastructure layer: it supports experimentation, suggests conjectures, and provides certified finite evidence that can be incorporated into rigorous arguments.

Second, the choice of ambient domain matters. Over $\mathbb{N}$ and $\mathbb{Z}$, additive structure is rigid and exact enumeration in a small-sumset strip is meaningful. Over $\mathbb{R}_+$, multiplicative structure admits geometric-progression models with algebraic ratios, and the small-product-set strip becomes the natural counterpart. The atlas is therefore intrinsically multi-domain: it records which pairs are realized in which domain, and it stores witnesses in a representation suitable for deterministic checking in that domain.

In summary, the SumProductAtlas 2.0 is designed to make the finite sum–product landscape up to $N_{\max}$ explicit in a way that is both mathematically grounded (via inverse theorems) and computationally trustworthy (via deterministic verification and auditable enumeration). The remaining sections formalize the notation, domains, and normalization conventions that allow us to state these claims precisely and to store witnesses without redundancy.

2 Notation and domains

We work throughout with finite sets in commutative semirings, and we emphasize from the outset that our objects are *sets* (no multiplicity). For a finite set A in an ambient domain X we write $|A| = n$, and we form the associated sumset and product set

$$A + A := \{a + b : \ a, b \in A\}, \qquad AA := \{ab : \ a, b \in A\}.$$

Their cardinalities will be denoted

$$|A + A| = i, \qquad |AA| = j,$$

and we refer to (i, j) as the *sum–product pair* of A. For a fixed ambient domain X and fixed n we record all realized pairs via

$$\mathrm{SPP}_X(n) \ := \ \big\{(|A + A|, |AA|) : \ A \subset X, \ |A| = n\big\}.$$

When $X = \mathbb{N}$ we abbreviate $\mathrm{SPP}(n) := \mathrm{SPP}_{\mathbb{N}}(n)$. We will frequently move among the domains

$$X \in \{\mathbb{Z}, \mathbb{N}, \mathbb{Q}_+, \mathbb{R}_+\},$$

where $\mathbb{Q}_+$ denotes the positive rationals and $\mathbb{R}_+$ the positive reals. The restriction to positive elements in the multiplicative domains is not cosmetic: it eliminates sign ambiguities and permits geometric-progression models (and hence algebraic encoding of witnesses) without additional bookkeeping.

Two elementary bounds will be used implicitly. For any torsion-free additive setting (in particular for $X \subset \mathbb{R}$) one has

$$2n - 1 \ \le \ |A + A| \ \le \ \frac{n(n + 1)}{2},$$

where the lower bound is realized by arithmetic progressions and the upper bound is realized by sets with all unordered pairwise sums distinct (for instance, geometric progressions with large ratio). The same inequalities hold for $|AA|$ over $X \subset \mathbb{R}_+$, with arithmetic progressions replaced by geometric progressions on the lower end. Our atlas records the full range of realized pairs, but the completeness statements we make are confined to the small-doubling "strips" discussed in the introduction.

A central organizational choice is to separate *existence* questions (which pairs occur) from *representation* questions (how we store witnesses without redundancy). This separation is enabled by invariances of $(|A+A|, |AA|)$ under simple transformations. On the additive side, for $A \subset \mathbb{Z}$ and $t \in \mathbb{Z}$ we have translation invariance

$$(A+t)+(A+t) = (A+A)+2t, \qquad \text{hence} \qquad |(A+t)+(A+t)| = |A+A|.$$

On the multiplicative side, for any $d > 0$ in the relevant domain we have dilation invariance

$$(dA)(dA) = d^2(AA), \qquad \text{hence} \qquad |(dA)(dA)| = |AA|,$$

and similarly $(dA)+(dA) = d(A+A)$ gives $|dA+dA| = |A+A|$. Thus scaling by a positive factor preserves both cardinalities, while translation preserves the additive cardinality and is used primarily to normalize integer witnesses. Over $\mathbb{N}$ we cannot translate arbitrarily without leaving the domain, but for any finite $A \subset \mathbb{N}$ we may translate it into $\mathbb{Z}$, apply a normalization, and then translate back into $\mathbb{N}$ by adding a sufficiently large constant; since only the *cardinalities* of $A + A$ and AA matter, this domain management does not affect which pairs occur.

It is therefore natural to store witnesses only up to these trivial symmetries. Concretely, for integer data we fix a canonicalization convention that collapses all translates and integer dilates of a set to a single representative. Given a nonempty $A \subset \mathbb{Z}$ we define $\mathrm{Can}(A)$ by the following steps:

1. translate by $-\min A$ so that the minimum element becomes 0;
2. divide by $\gcd(A)$ so that the resulting set has $\gcd = 1$;
3. sort the resulting set increasingly and regard it as an ordered n-tuple.

In the event of any remaining ambiguity (for example, if two different inputs lead to the same normalized set), we break ties by lexicographic order of the sorted tuple; in practice this simply means that we store the unique normalized, sorted tuple. When we wish to store a witness as a subset of $\mathbb{N}$ rather than $\mathbb{Z}$, we optionally apply an additional shift by $+1$ so that the minimum element is 1 instead of 0. This final shift is a presentational choice only; it plays no role in the combinatorial invariances.

The purpose of $\mathrm{Can}(A)$ is twofold. First, it makes enumeration auditable: an enumerator can list canonical representatives and hash them, and an auditor can confirm that the list contains no duplicates arising from translation or scaling. Second, it makes the database key stable: a witness is stored under a canonical key, so that independent contributors cannot inadvertently create distinct entries for the same configuration. The mathematical justification for using canonical representatives is immediate from the invariances above: for existence questions about pairs in $\mathrm{SPP}_{\mathbb{Z}}(n)$ (or about the restriction $i \le 3n-3$ in $\mathrm{SPP}(n)$) it suffices to consider canonical witnesses.

For rational or real witnesses we employ analogous normalizations, but we distinguish carefully between *arbitrary* finite subsets of $\mathbb{R}_+$ and those described by structured encodings. In the multiplicative-strip regime our enumeration and storage are based on the geometric-progression containment forced by small multiplicative doubling. Thus, rather than storing an arbitrary list of floating-point numbers, we store a witness in the form

$$A = \{r^{e_1}, \dots, r^{e_n}\} \subset \mathbb{R}_+, \qquad 0 \le e_1 < \cdots < e_n \le L,$$

where $r > 1$ is an algebraic real and $\{e_1, \dots, e_n\}$ is a set of integers. The normalizations in this model are the multiplicative analogues of translation and dilation: scaling A by a constant corresponds to adding a constant to the exponent set if we also adjust by a power of r, and replacing r by r^g corresponds to dividing all exponents by g when possible. Accordingly, we impose the exponent-side conditions

$$\min\{e_1, \dots, e_n\} = 0, \qquad \gcd(e_1, \dots, e_n) = 1,$$

which mirror the integer convention $\min A = 0$ and $\gcd(A) = 1$. We then represent the witness by the pair $(r; e_1, \dots, e_n)$ together with a fixed algebraic specification of r (minimal polynomial and an isolating interval), and we regard two such descriptions as identical if they agree under these normalizations. This is the sense in which our multiplicative witnesses are "canonical": we do not attempt to canonically represent *all* subsets of $\mathbb{R}_+$, but we do canonically represent all those produced by our structured enumeration in the small-product strip.

We stress that $\mathrm{SPP}_X(n)$ depends on the domain X in a genuine way. For example, in $\mathbb{N}$ there are integrality constraints on products, and collisions in AA may be forced by factorization structure; over $\mathbb{R}_+$ one can choose algebraic ratios that create additive collisions inside a geometric progression in ways that have no integer analogue. For this reason we keep the ambient domain explicit in the notation and we state completeness theorems domain-by-domain: the additive strip is treated over $\mathbb{N}$ (where inverse theorems yield explicit integer model families), while the multiplicative strip is treated over $\mathbb{R}_+$ (where containment in a short geometric progression yields a tractable search space of algebraic ratios).

Finally, we record the extremal quantity that motivates much of the atlas:

$$\alpha_n := \min_{A \subset \mathbb{N},\ |A|=n} \max\bigl\{|A+A|,\ |AA|\bigr\}.$$

The database is not designed to certify that a stored witness attains α_n; rather, it is designed so that any stored witness A_n provides an *upper bound*

$\alpha_n \le \max\{|A_n + A_n|, |A_nA_n|\}$, and the supporting data can be checked independently. The notions introduced in this section—the domain-dependent phase diagram $\mathrm{SPP}_X(n)$, the invariances under translation and dilation, and the canonicalization map $\mathrm{Can}(A)$ (together with its geometric-progression analogue in $\mathbb{R}_+$)—are the basic ingredients that make such storage and querying meaningful at scale.

In the next section we formalize the verifier interface and the certificate objects that allow each stored claim to be re-checked deterministically, independently of the enumerators and independently of the particular implementation used to construct the atlas.

3 Certificates and verification

The atlas is intended to be usable without trusting the particular codebase that produced it. To that end we separate the *generation* of candidate witnesses (which may be heuristic, distributed, or domain-specific) from the *verification* of any claimed sum–product pair. The verification layer consists of a deterministic routine Check, together with a certificate object $\mathrm{Cert}(A)$ that contains enough information for an independent implementation to re-check the claim.

The verifier interface

The verifier is specified as a pure function on structured inputs. In its simplest integer form the input is a finite set $A \subset \mathbb{Z}$ (represented by a list of integers with duplicates allowed at the transport level but ignored at the mathematical level), and the output is a triple

$$\mathsf{Check}(A) \;=\; (i, j, \mathrm{Cert}(A)), \qquad \text{where} \qquad i = |A + A|, \;\; j = |AA|.$$

We require that Check be *deterministic* and *total* on all supported inputs: it must either return the triple above or return a structured error indicating that the encoding is invalid (e.g. non-integral data in an integer mode). In particular, there is no randomness, no floating-point arithmetic, and no dependence on external state.

For $A \subset \mathbb{Q}_+$ we treat the input as a list of reduced fractions and compute $A + A$ and AA inside $\mathbb{Q}$ using exact rational arithmetic. For $A \subset \mathbb{R}_+$ in the multiplicative-strip regime we use a structured encoding: a witness is given as

$$A = \{r^{e_1}, \dots, r^{e_n}\}, \qquad 0 \le e_1 < \cdots < e_n,$$

together with an algebraic specification of r (a squarefree integer polynomial $p \in \mathbb{Z}[x]$ with $p(r) = 0$ and an isolating interval $(a, b) \subset \mathbb{Q}$ containing r and no other real root of p). In this mode the verifier treats all elements as algebraic numbers in the number field $\mathbb{Q}(r)$, and again the output is a triple $(i, j, \mathrm{Cert}(A))$ with exact cardinalities.

What a certificate contains

A certificate is not merely the list A itself; it is a *reproducible transcript* of the computation that allows an independent checker to confirm the output without relying on any hidden implementation choices. Concretely, a certificate contains at least the following components.

1. **Witness payload.** An unambiguous encoding of A:
   - for integer and rational inputs, the sorted list of distinct elements;
   - for real inputs in geometric-progression form, the polynomial p, isolating interval (a, b) for r, and the exponent list $(e_1, \dots, e_n)$.
2. **Normalization metadata.** The canonical key under which the witness is stored (e.g. $\mathrm{Can}(A)$ in the integer domain, or the normalized exponent tuple with $\min e_i = 0$ and $\gcd(e_i) = 1$ in the GP model). This is used for database integrity and deduplication, but also as a consistency check that the stored entry is stable under the intended invariances.
3. **Derived-set commitments.** A commitment to the computed sets $A + A$ and AA. When the derived sets are small enough, the certificate may include their explicitly sorted lists. When they are larger, we store a chunked hash commitment: the sorted list is split into fixed-size blocks, each block is hashed, and a top-level hash commits to the ordered list of block hashes. This yields a tamper-evident representation while keeping the certificate size manageable.
4. **Cardinality claims.** The integers i and j claimed to equal $|A + A|$ and $|AA|$, together with the obvious consistency checks (e.g. $2n - 1 \leq i, j \leq n(n + 1)/2$ in torsion-free settings).
5. **Verifier transcript.** A minimal description of the deterministic steps that were executed, sufficient to ensure that a re-checker reconstructs the same intermediate objects. At the level of abstraction relevant here, this consists of (a) the sorting order used on the ambient domain, (b) the canonical encoding used for hashing elements of $A + A$ and AA, and (c) the exact arithmetic backend (integers, reduced rationals, or algebraic numbers with certified comparisons).

We emphasize that the certificate does *not* ask the reader to trust a hash as a substitute for correctness. The hash commitments serve to detect tampering and to permit efficient storage; correctness is established by recomputation of the derived sets (or, when the certificate includes explicit derived lists, by recomputation and equality testing against those lists).

Independent re-checking

We specify an independent routine $\mathsf{ReCheck}$ whose only purpose is to validate certificates. Its contract is:

$$\mathsf{ReCheck}(\mathrm{Cert}(A)) = \mathsf{accept} \quad \iff \quad \text{the certificate implies } i = |A{+}A| \text{ and } j = |AA|.$$

Operationally, $\mathsf{ReCheck}$ parses the witness payload, reconstructs the domain elements exactly, recomputes $A+A$ and AA using the deterministic ordering and encoding rules declared in the transcript, and confirms that the recomputed commitments match those stored in the certificate (either by direct list equality or by verifying the hash tree). It then checks that the recomputed cardinalities equal the claimed (i, j).

In the algebraic-real mode, the only additional issue is *comparison* and *deduplication*. Two algebraic numbers may be equal or ordered in a way that is not evident from floating approximations, and the entire enterprise fails if deduplication is done approximately. Accordingly, $\mathsf{ReCheck}$ requires certified comparisons: given two algebraic numbers represented in $\mathbb{Q}(r)$, we decide equality and ordering by exact field operations together with root isolation (or, equivalently, by refining isolating intervals until disjointness is certified). This ensures that the set operations defining $A + A$ and AA are performed in a mathematically meaningful way.

Reproducibility guarantees

The atlas is versioned, and each entry stores: (i) the canonical key, (ii) the verifier version identifier, and (iii) a content hash of the certificate object. Reproducibility is then understood in the following concrete sense: any party can download the certificate, run an independently written $\mathsf{ReCheck}$, and obtain the same accept/reject outcome. The determinism requirement eliminates the common failure modes of scientific computation (random seeds, platform-dependent floating point, and nondeterministic iteration orders).

We also enforce deterministic ordering conventions at the serialization layer. For example, integer sets are always stored as increasing tuples; rational numbers are stored in reduced form with positive denominators; algebraic numbers are stored by a fixed choice of primitive element (the designated r) and by a fixed normal form for elements of $\mathbb{Q}(r)$ (e.g. reduced coordinates in the power basis modulo the minimal polynomial). These conventions ensure that two independent producers do not obtain incompatible encodings for the same mathematical object.

Threat model and non-goals

Our primary threat model is *untrusted generation* and *implementation diversity*. Contributors may submit incorrect witnesses (accidentally or maliciously), enumerators may be buggy or incomplete, and a single verifier implementation may contain subtle errors. The certificate layer is designed to make such failures detectable: a malicious or incorrect submission must pass ReCheck, which can be implemented independently. We therefore avoid any feature that would force trust in the producer, such as floating-point comparisons, undocumented heuristics, or opaque binary formats.

We do *not* attempt to protect against adversaries who can compromise the distribution channel for both certificates and re-checkers, nor do we attempt to provide cryptographic proofs in the sense of succinct arguments. The certificates are intended to be straightforward, mechanically checkable transcripts whose verification cost is commensurate with recomputing $A+A$ and AA for the relevant $n \le N_{\max}$. This is appropriate for our scale: the atlas is a finite, publicly auditable dataset rather than an indefinitely growing ledger.

With these verification and certification conventions in place, we may treat any accepted atlas entry as a mathematically valid witness for its recorded sum–product pair, independently of how that witness was found. In the subsequent sections we exploit this separation: the structural theorems restrict the search space in the small-doubling strips, the enumerators exhaust that space up to canonicalization, and the verifier–certificate layer turns the resulting computational claims into independently checkable mathematical data.

4 Structural enumeration in the additive strip

Fix n with $3 \le n \le N_{\max}$. In this section we explain the mechanism behind EnumAdd(n), the enumerator responsible for the additive-strip claim, i.e. the regime

$$|A| = n, \qquad |A+A| \le 3n-3, \qquad A \subset \mathbb{N}.$$

The goal of EnumAdd(n) is *not* to search arbitrarily through $\binom{\mathbb{N}}{n}$, but to exhaust, up to canonicalization, the finite set of structural possibilities forced by small doubling. The mathematical input is a Freiman-type inverse theorem in dimension one (recorded abstractly as Lemma 2), which places every such A inside an explicitly describable container family. The enumerator ranges over those containers and the relevant parameters, produces canonical representatives, and passes them to Check to obtain certified pairs $(|A+A|, |AA|)$.

Freiman-type container families at the $3n-3$ threshold

We work in the torsion-free integer setting, so small doubling forces one-dimensional structure. Concretely, Lemma 2 asserts that if $A \subset \mathbb{Z}$ has $|A| = n$ and $|A+A| \le 3n-3$, then A lies in a finite union of model families admitting an explicit parameterization. For the purposes of enumeration we treat the theorem as providing a list of *containers* $C \subset \mathbb{Z}$ of bounded complexity and bounded size (on the order of $2n$), together with constraints that ensure $A \subset C$ and $|A| = n$.

Operationally, the relevant containers are all of "progression-like" type. The dominant family is containment in a short arithmetic progression,

$$A \subset P = \{x, x+d, \dots, x+Ld\}, \qquad L \leq 2n-2,$$

with $d \geq 1$, and with additional constraints on how many points of P are omitted. At the $3n-3$ level, further families occur in which A is the union of two short progressions with the same common difference (equivalently, a progression with a single long gap), or a small list of explicitly bounded exceptional configurations. We do not need to reproduce the full classification here; what matters for the construction is that the theorem reduces the ambient infinite search to a *finite* search over parameter tuples whose ranges depend only on n.

Two features of this reduction are critical for computation. First, the containers are of size $O(n)$, so subsets can be handled by bitmask-style representations when needed. Second, the parameterization isolates a small number of integer degrees of freedom (lengths of progressions, positions of gaps, and similar), so that enumeration can be performed systematically and audited.

Canonical parameterization and removal of trivial symmetries

Enumeration must avoid producing the same witness many times through translation, dilation, or through redundant parameter choices. We therefore build canonicalization into the parameterization stage.

On the witness side we ultimately store A in the integer canonical form $\mathrm{Can}(A)$: translate so $\min A = 0$, divide by $\gcd(A)$ to obtain gcd 1, and then (when working in $\mathbb{N}$) apply the fixed positivity convention. On the container side we impose compatible normalizations. For an arithmetic progression container we may take $x = 0$ and $d = 1$ without loss of generality for existence questions in $\mathbb{Z}$, since dilation by d and translation by x preserve $|A + A|$ and do not affect the subset pattern inside the progression. Thus, in the normalized model we reduce to

$$A \subset \{0, 1, \dots, L\}, \qquad L \leq 2n-2,$$

with $|A| = n$, and we enumerate subsets by their increasing index sets $E = \{e_1 < \cdots < e_n\} \subset \{0, \dots, L\}$. The resulting $A = \{e_1, \dots, e_n\}$ is already translated; the remaining normalization step is the gcd reduction, which we perform after subset selection. (When $\gcd(A) > 1$, the witness is discarded in normalized enumeration since it would canonicalize to a smaller set already encountered.)

For two-progression containers (or progression-with-gap containers) we similarly enforce a unique encoding, e.g. by ordering the blocks by their minimum element, fixing the common difference 1 in the normalized world, and recording the gap position and block lengths as a tuple with a prescribed inequality convention. The essential requirement is that each model instance corresponds to exactly one container in normalized coordinates, so that audit logs can be keyed by parameter tuples.

Enumeration algorithm $\mathsf{EnumAdd}(n)$

We describe $\mathsf{EnumAdd}(n)$ at the level necessary to justify the completeness statement in Theorem B.

1. **Generate normalized containers.** Using the structural theorem, we generate the finite list of container types relevant for $|A+A| \leq 3n-3$, and for each type we iterate over all admissible parameter tuples (e.g. $L \leq 2n-2$ for progression length; allowable gap lengths/positions for two-block models; and the finite exceptional list). Each container is realized concretely as a subset $C \subset \{0, 1, \ldots, M\}$ for some $M = O(n)$.

2. **Enumerate n-subsets inside containers.** For each container C, we enumerate all n-element subsets $A \subset C$ satisfying the side conditions attached to the container family (for example, conditions that ensure the Freiman description is not redundant, or that certain endpoints are included). This enumeration is done in a deterministic order (lexicographic in the exponent/index list) so that the output list is reproducible and admits a stable hash.

3. **Canonicalize and deduplicate.** Each candidate A is mapped to $\mathrm{Can}(A)$, and we discard any candidate whose canonical form has already been produced. This step is essential because distinct containers (or distinct parameter tuples within a family) can produce the same A; the atlas is indexed by canonical witnesses, not by model provenance.

4. **Compute and record sum–product data.** For each new canonical witness we call $\mathsf{Check}(A)$ to obtain $(i, j, \mathrm{Cert}(A))$ with $i = |A+A|$ and $j = |AA|$. We retain only those with $i \leq 3n-3$ (this is automatic for correctly generated model instances, but we enforce it as a consistency check). We then register the realized pair (i, j) and, for atlas purposes, store one witness and certificate for each pair.

The output of $\mathsf{EnumAdd}(n)$ is thus a list of canonical integer witnesses, together with enough provenance information to audit the enumeration: container type, parameter tuple, and the canonical key $\mathrm{Can}(A)$.

What is proved and what is computer-audited

There are two logically distinct components behind the additive-strip completeness claim.

Mathematical reduction (proved, conditional only on cited theorems). Assuming Lemma 2 (the Freiman-type container theorem), we obtain that every $A \subset \mathbb{Z}$ with $|A| = n$ and $|A + A| \le 3n - 3$ lies in one of the finitely parameterized model families. Together with Lemma 1 (invariance under translation and dilation), this justifies the normalized container enumeration: it suffices to enumerate subsets of normalized containers and then canonicalize.

Within this framework we also prove the *soundness* of $\mathsf{EnumAdd}(n)$: every witness it outputs indeed has $|A| = n$ and satisfies $|A + A| \le 3n - 3$, because each candidate arises from a container certified by the structural theorem and is filtered by an explicit recomputation of $|A + A|$ via Check.

Exhaustiveness for specific n (computer-audited). The statement that the atlas contains *exactly all* pairs $(i, j) \in \mathrm{SPP}_{\mathbb{N}}(n)$ with $i \le 3n - 3$ is, at the end of the day, an exhaustive-finitary claim for each $n \le N_{\max}$. Even given the structural theorem, there remains a large but finite enumeration over parameter tuples and subsets, and correctness depends on the absence of implementation gaps (missed parameter ranges, duplicate-elimination bugs, and similar). We therefore treat completeness in the strip as a computer-audited result: the released artifacts include (i) deterministic audit logs listing all enumerated canonical keys $\mathrm{Can}(A)$, (ii) hashes committing to these logs, and (iii) cross-check scripts verifying internal consistency (e.g. that every stored pair in the strip has at least one witness, and every witness satisfies the strip bound).

In particular, the equivalence

$$\{(|A{+}A|, |AA|) : A \subset \mathbb{N},\ |A| = n,\ |A{+}A| \le 3n{-}3\} \;=\; \{(|A{+}A|, |AA|) : A \in \mathcal{W}_{\mathrm{add}}(n)\}$$

is asserted as an audited computational conclusion, not as a purely handwritten derivation. The role of the mathematics is to make this audit feasible: without the container theorem, the left-hand side would not be effectively enumerable.

Finally, we stress that the additive-strip computation is designed to be independently reproducible. Given the published container specification (family list and parameter ranges), an independent implementation can regenerate $\mathcal{W}_{\mathrm{add}}(n)$, apply Check or any independent verifier, and compare the resulting canonical keys and realized pairs against the atlas and its audit hashes. This is the sense in which the additive-strip portion of the atlas is meant to be a verifiable mathematical dataset rather than an opaque table of numbers.

5 Structural enumeration in the multiplicative strip over $\mathbb{R}_+$

Fix n with $3 \le n \le N_{\max}$. In this section we explain the mechanism behind $\mathsf{EnumMul}(n)$, the enumerator responsible for the multiplicative-strip claim over $\mathbb{R}_+$, i.e. the regime

$$|A| = n, \qquad |AA| \le 3n-3, \qquad A \subset \mathbb{R}_+.$$

Unlike the additive strip, where we insist on integer completeness, here we deliberately work over $\mathbb{R}_+$ in order to access geometric-progression models and to avoid integrality constraints that would artificially restrict the set of realizable pairs. The structural input is the multiplicative small-doubling inverse statement recorded abstractly as Lemma 3, which places every such A inside a short geometric progression after scaling. The enumerator ranges over the resulting finite exponent patterns and, when necessary, over a finite set of algebraic ratios forced by additive collisions among geometric-progression elements.

Geometric-progression containers and exponent models

By Lemma 3, if $A \subset \mathbb{R}_+$ has $|A| = n$ and $|AA| \le 3n-3$, then after scaling by a positive constant we may assume that

$$A \subset \{1, r, r^2, \dots, r^L\}$$

for some $r > 0$ and some $L \le 2n-2$. Thus, in normalized coordinates we represent a candidate witness by a pair (E, r) where

$$E = \{e_1 < \cdots < e_n\} \subset \{0, 1, \dots, L\}, \qquad A(E,r) := \{r^{e_1}, \dots, r^{e_n}\}.$$

Two immediate observations make this model computationally effective. First, the length bound $L \le 2n-2$ makes the search over exponent patterns E finite and explicitly bounded in terms of n. Second, for $r \ne 1$ the multiplicative structure is essentially combinatorial: products in $A(E,r)A(E,r)$ correspond to exponent sums in $E+E$, and positivity prevents accidental equalities coming from signs. Concretely, for any fixed E and any fixed $r > 0$ with $r \ne 1$,

$$|A(E,r)A(E,r)| = |E+E|.$$

In particular, the strip constraint $|AA| \le 3n-3$ becomes the purely discrete condition $|E+E| \le 3n-3$, which we can test without any real-arithmetic.

The role of the ratio r is therefore not to control $|AA|$ (which is determined by E), but to control $|A+A|$. For generic $r > 1$, the sums $r^{e_i} + r^{e_j}$ are pairwise distinct for $1 \le i \le j \le n$, hence $|A+A| = n(n+1)/2$. Smaller values of $|A+A|$ can occur only when there are nontrivial collisions among such sums, and these collisions force r to be algebraic of explicitly bounded complexity.

Sparse-polynomial constraints from sum collisions

Suppose that for fixed E the witness $A(E, r)$ has a strict sum collision, i.e. there exist (not necessarily distinct) indices with

$$r^a + r^b = r^c + r^d,$$

where $\{a, b\} \neq \{c, d\}$ and $a, b, c, d \in E$. After dividing by the smallest occurring power of r we reduce to an equation of the form

$$1 + r^j = r^k + r^\ell$$

with integers $0 \leq k \leq \ell < j \leq L$. Rearranging yields a sparse polynomial constraint

$$p(r) := r^j - r^\ell - r^k + 1 = 0.$$

This is the basic mechanism behind Lemma 3: additive non-genericity inside a geometric progression forces the ratio to be a root of a $\{0, \pm 1\}$-polynomial of degree at most $L \leq 2n - 2$. For the purposes of exhaustive enumeration in the multiplicative strip, two finiteness consequences are decisive.

- For a fixed bound L, there are only finitely many exponent quadruples (k, ℓ, j) (and hence finitely many such sparse polynomials) to consider.
- For any such polynomial p, there are only finitely many positive real roots. Each root can be uniquely specified for verification by a minimal polynomial (a factor of p) together with an isolating interval.

We emphasize that we do not attempt to classify *all* algebraic relations among GP sums. Rather, we exploit the contrapositive: if $|A + A|$ is *not* maximal, then at least one collision exists, hence r satisfies at least one sparse polynomial constraint of bounded degree and bounded sparsity. Enumerating these constraints is therefore sufficient to capture all non-generic sumset sizes that can occur within the multiplicative strip.

In implementation we also impose normalization conventions to avoid duplications coming from trivial transformations of the GP model. We fix $r > 1$ (replacing r by $1/r$ reverses exponent order but does not change the induced pair $(|A + A|, |AA|)$), we shift exponents so that $\min E = 0$ (equivalently $1 \in A$ after scaling), and we require $\gcd(E) = 1$ to exclude redundant powering (if $\gcd(E) = g > 1$ then $A(E, r) = A(E/g, r^g)$ and the latter is the canonical representation).

Enumeration algorithm $\mathsf{EnumMul}(n)$

We now describe $\mathsf{EnumMul}(n)$ at the level needed to justify the multiplicative-strip completeness statement in Theorem C. The algorithm is finite because both the exponent range and the family of sparse constraints are bounded explicitly in terms of n.

1. **Enumerate exponent patterns.** For each $L \in \{0, 1, \ldots, 2n-2\}$ we enumerate all n-element subsets $E \subset \{0, 1, \ldots, L\}$ with $\min E = 0$ and $\gcd(E) = 1$. For each such E we compute $j_E := |E + E|$. We keep only those with $j_E \leq 3n - 3$, since these are the only ones capable of producing witnesses in the multiplicative strip.

2. **Generate candidate ratios.** For each retained E we assemble a finite set of candidate ratios $\mathcal{R}(E)$, consisting of:

   - one *generic* placeholder ratio r_{gen} (used to realize the maximal sumset value $|A + A| = n(n+1)/2$ for that E), and
   - all positive real algebraic numbers $r > 1$ that arise as roots of sparse polynomials $r^j - r^\ell - r^k + 1$ with $0 \leq k \leq \ell < j \leq L$ *and* with exponent indices consistent with E in the sense that the corresponding collision can occur among elements of $A(E, r)$.

   Concretely, we enumerate the relevant triples (k, ℓ, j), form the polynomial $p(x) = x^j - x^\ell - x^k + 1$, isolate its real roots > 1, and record each such root by a minimal polynomial together with an isolating interval.

3. **Verify and record pairs.** For each $(E, r) \in \bigcup_E \{E\} \times \mathcal{R}(E)$ we form the witness $A(E, r)$ and call Check to compute (i, j) together with $\mathrm{Cert}(A)$. By construction j should equal j_E; we treat any discrepancy as an error. We retain only those with $j \leq 3n - 3$ (again, this should be automatic once E passes the filter, but we enforce it as a consistency check). For atlas purposes we store one certified witness for each realized pair (i, j).

4. **Canonicalize and deduplicate.** Distinct descriptions can lead to the same realized witness data, especially when multiple sparse constraints define the same algebraic ratio or when a collision constraint is redundant for a given E. We therefore assign a canonical key to each real witness: the sorted exponent list E, together with a canonical algebraic encoding of r (primitive minimal polynomial with integer coefficients, a specified root index, and a rational isolating interval), and we deduplicate by this key.

The output of $\mathsf{EnumMul}(n)$ is thus a finite list $\mathcal{W}_{\text{mul}}(n)$ of canonical real witnesses in geometric-progression form, each accompanied by provenance metadata (the exponent pattern, the defining sparse polynomial(s) used to discover the ratio, and hash references to the enumeration run) and by a deterministic certificate from Check.

Certification of algebraic computations

Because the witnesses live in $\mathbb{R}_+$ and may involve algebraic ratios, certification must address two distinct issues: exact arithmetic on algebraic numbers, and exact deduplication of sums and products.

Following Lemma 4, we encode r by a minimal polynomial $m_r(x) \in \mathbb{Z}[x]$ together with an isolating interval $I_r \subset \mathbb{Q}$ that contains exactly one real root of m_r and is contained in $(1, \infty)$. The witness itself is encoded as the exponent list $E = \{e_1, \ldots, e_n\}$, so that $A = \{r^{e_1}, \ldots, r^{e_n}\}$. The verifier Check then performs the following operations deterministically and in a recheckable manner: compute algebraic representations of r^e for each $e \in E$ (as elements of $\mathbb{Q}(r)$), enumerate all $n(n+1)/2$ sums and products, and deduplicate them by certified equality testing. Equality and ordering comparisons among algebraic numbers are certified via standard exact methods (e.g. squarefree factorization, isolating intervals refined by Sturm sequences), and the certificate includes enough data to reproduce each deduplication decision.

In particular, $\mathrm{Cert}(A)$ contains (i) the algebraic specification of r and the exponent list E, (ii) hashed encodings of the deduplicated lists representing $A + A$ and AA, (iii) the resulting counts (i, j), and (iv) a transcript of the comparison steps used to certify that two algebraic expressions are equal or unequal. An independent checker ReCheck can therefore reconstruct the same deduplicated sets and confirm that the published values satisfy $i = |A + A|$ and $j = |AA|$.

Finally, as in the additive strip, the claim that the atlas contains *exactly all* pairs in the multiplicative strip for each $n \leq N_{\max}$ is treated as computer-audited: the released artifacts include deterministic logs of all enumerated exponent patterns and ratio encodings (with stable canonical keys), together with hashes that commit to the full enumeration output. The mathematics ensures finiteness and provides the sparse-polynomial mechanism that makes enumeration feasible; the audit logs ensure that the finite search has in fact been carried out exhaustively and reproducibly.

6 The database: schema, stored artifacts, indexing, and versioning

We package the atlas as a versioned database whose primary purpose is to make each recorded sum–product pair reproducible from first principles: a user should be able to retrieve a witness A, re-run ReCheck on the stored certificate, and recover the asserted pair $(|A + A|, |AA|)$ without appealing to any unpublished computation. Consequently, our unit of storage is not merely the pair (n, i, j), but an *atlas entry* containing (a) a canonical witness description, (b) a machine-checkable certificate, and (c) provenance and audit metadata linking the entry to a specific enumerator run and verifier version.

Entry schema and canonical identifiers

Each entry is conceptually a tuple

$$\text{Entry} = (n, X, i, j, \text{wit}, \text{cert}, \text{meta}),$$

where $X \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}_+, \mathbb{R}_+\}$ is the ambient domain, $|A| = n$, $(i, j) = (|A + A|, |AA|)$, wit is a domain-appropriate witness encoding of A, cert $= \text{Cert}(A)$ is the certificate produced by Check, and meta collects all remaining non-mathematical information.

To ensure stable deduplication across runs and across contributions, we assign each entry a *canonical key* and derive from it a content hash used as an immutable identifier. For integer-domain witnesses ($X \subseteq \mathbb{Z}$) the canonical key is $\text{Can}(A)$ in the sense fixed earlier (translate to $\min A = 0$ and divide by $\gcd(A)$, with deterministic tie-breaking). For multiplicative-strip real witnesses ($X = \mathbb{R}_+$ in geometric-progression form) we store a canonical key of the form

$$\text{Key}_{\mathbb{R}_+}(A) = \big(E,\ m_r(x),\ I_r,\ \text{root\_index}\big),$$

where $E = \{e_1 < \cdots < e_n\}$ is the exponent set with $\min E = 0$ and $\gcd(E) = 1$, and r is specified by a primitive squarefree minimal polynomial $m_r \in \mathbb{Z}[x]$ together with a rational isolating interval $I_r \subset (1, \infty)$ containing exactly one real root; the additional root index fixes the intended root if m_r has multiple real roots in $(1, \infty)$ after interval refinement.

We emphasize that the canonical key is chosen to be independent of auxiliary discovery data (e.g. which sparse polynomial produced r). Such auxiliary data are recorded in meta, but the key depends only on the witness itself. We then define an entry identifier

$$\text{ID} := \text{SHA256}\big(\text{Serialize}(\text{Key})\big),$$

so that identical witnesses (under canonicalization) always receive the same identifier, and any change in encoding is detectable. The database enforces the uniqueness constraint that, for fixed (n, i, j, X), at most one *active* entry is designated as the atlas representative; if multiple witnesses realize the same pair, only one is selected as representative, but the others may be stored as non-representative aliases keyed by their own ID.

Witness payloads

We store witnesses in a compact, exact format tailored to the ambient domain.

Integer witnesses. For $A \subset \mathbb{N}$ or $A \subset \mathbb{Z}$, the payload is the sorted list $(a_1 < \cdots < a_n)$ of the canonical representative $\text{Can}(A)$, encoded by delta-compression:

$$(a_1,\ a_2 - a_1,\ \ldots,\ a_n - a_{n-1}),$$

with variable-length integers. This format is lossless, stable under re-serialization, and efficiently supports reconstruction, hashing, and sanity checks (monotonicity, gcd $= 1$, min $= 0$). When the atlas additionally stores a positive shift (e.g. to place $A \subset \mathbb{N}$), this shift is treated as derived metadata and is not part of the canonical witness key.

$\mathbb{R}_+$ geometric-progression witnesses. For $A \subset \mathbb{R}_+$ represented as $A = \{r^{e_1}, \dots, r^{e_n}\}$, the witness payload consists of (i) the exponent list E and (ii) the algebraic specification of r described above. We do *not* store floating approximations of the elements of A as authoritative data; numerical approximations may be included for convenience, but the certificate and all re-checking logic are based on exact algebraic-number computations in $\mathbb{Q}(r)$.

Certificates as first-class stored artifacts

A certificate $\mathrm{Cert}(A)$ is stored as an immutable object, separate from the witness payload, because it can be substantially larger. Its role is to allow an independent checker to reproduce $|A+A|$ and $|AA|$ exactly, including the deduplication decisions.

In the integer setting, the certificate contains enough information to reconstruct the enumerated multisets

$$\{a_p + a_q : 1 \le p \le q \le n\}, \qquad \{a_p a_q : 1 \le p \le q \le n\},$$

and to verify the deduplicated cardinalities. In practice we store (a) the witness list, (b) the computed values (i, j), and (c) a deterministic digest of the deduplicated sumset and product set (e.g. a sorted list for small sizes, or a Merkle-tree commitment of a sorted sequence for larger sizes). The rechecker recomputes the full sets and confirms that their digests match the committed digests.

In the algebraic-real setting, the certificate additionally contains a transcript sufficient to certify comparisons of algebraic numbers used in sorting and deduplication (e.g. isolating interval refinements and the decisions they justify). We separate the certificate into a compact *header* (witness specification, (i, j), and global digests) and a *comparison log* whose content-addressed chunks can be streamed during verification. This allows re-checking to be performed with bounded memory while remaining fully deterministic.

Provenance metadata and audit linkage

The field meta is not logically necessary for correctness, but it is necessary for accountability and reproducibility of the *completeness* claims. For each entry we store:

- the producing routine (`EnumAdd` or `EnumMul`) and its parameter record (e.g. n, strip bound, and any internal pruning settings that are logically redundant but operationally relevant);
- the exact verifier identity (semantic version and source commit hash) that produced $\mathrm{Cert}(A)$;
- the enumerator identity (semantic version and source commit hash) and a run identifier;
- timestamps and the submitter/source label (human contributor, automated pipeline, or imported legacy dataset);
- optional discovery information, such as the sparse polynomial(s) that led to a candidate ratio r in the $\mathbb{R}_+$ setting.

Most importantly, each enumerator run publishes an *audit manifest* listing (in canonical order) the keys of all candidates enumerated before filtering by realized pairs, together with hashes committing to that list. Individual entries link back to the manifest via run identifiers and hash references, so that a reader can verify that an entry was not produced by ad hoc search but by the declared exhaustive procedure.

Indexing, compression, and normalized-coordinate views

The atlas is indexed primarily by the triple (n, i, j), since this is the natural coordinate system for $\mathrm{SPP}_X(n)$. Concretely, we maintain per-n indices mapping each realized pair (i, j) to the representative entry identifier, with secondary indices mapping (n, i) to all realized j (and vice versa) for efficient strip queries.

For cross-n analysis we also store derived *normalized coordinates* as exact rationals. The database includes, for each entry, the values

$$\kappa_+(A) := \frac{|A+A|}{n}, \qquad \kappa_\times(A) := \frac{|AA|}{n},$$

together with optional alternative normalizations (e.g. division by $n-1$) used in plotting or heuristic searches. These derived coordinates are indexed to support range queries such as "all witnesses with $\kappa_+ \in [1.7, 1.9]$ and $\kappa_\times \in [1.8, 2.0]$" independently of n, while retaining (n, i, j) as the authoritative exact record.

All large objects (certificates, audit manifests, and optionally large witness lists) are stored in a content-addressed object store and compressed (e.g. using a deterministic compressor configuration). The main database file stores only small fixed-size records and hash pointers to objects. This design yields two benefits: (i) identical objects are deduplicated automatically across releases, and (ii) the integrity of each entry is reducible to hash verification.

Change-log policy and release versioning

We treat each atlas release as an immutable snapshot identified by a top-level manifest hash. A release consists of (a) a manifest enumerating all active representative entries, (b) the full object store addressed by hash, and (c) the source identities (commit hashes) of the verifier and enumerators used to produce the snapshot. The manifest includes, for each active representative pair (n, i, j, X), the corresponding entry identifier ID, so that completeness statements can be phrased as comparisons between a mathematically defined set of pairs and the manifest content.

Corrections are handled by explicit supersession: if an error is found in an entry (or in a verifier version), the entry is not silently modified. Instead we (i) deprecate the entry by marking it inactive in the next release, (ii) add a replacement entry with its own identifier and certificate, and (iii) record a change-log record explaining the reason for supersession and the affected release range. This preserves the ability to reproduce past results exactly while still converging to a correct atlas.

Finally, we impose a stability requirement on canonicalization and serialization: within a major version, the canonical key computation and the byte-level serialization used for hashing are fixed. Any change that would alter ID values for previously stored witnesses triggers a major-version increment, accompanied by a migration document that maps old identifiers to new ones via explicit key equivalences. This policy ensures that database citations and external audit scripts remain valid across minor releases while still allowing principled evolution of the infrastructure.

7 Main completeness theorems and their computational audits

We now state the completeness claims that the atlas is intended to certify, and we describe precisely what is (and is not) meant by "proved" in each case. Conceptually, each completeness statement has two components:

1. a mathematical reduction placing all witnesses in a finite union of explicitly parameterized model families, and
2. an exhaustive computation over those parameter families, with deterministic logging sufficient for an independent party to audit exhaustiveness and to re-check every recorded pair.

The second component is unavoidably computational; we therefore treat it as a *computer-audited theorem*, and we make the audit artifacts part of the release.

Verifier correctness (Theorem A) and the role of certificates

The logical foundation of every stored pair is the verifier correctness statement. In our setting, "verifier correctness" means the following: given a witness description in a supported domain, the routine Check deterministically produces

$$(i, j) = (|A + A|, |AA|)$$

together with a certificate $\mathrm{Cert}(A)$, and an independent routine ReCheck accepts $\mathrm{Cert}(A)$ if and only if the asserted sizes are correct. The atlas does not ask the reader to trust any enumerator logic to compute sizes; the enumerators merely propose candidates, while the verifier and certificate mechanism are the only authority for the counts.

The key design constraint is that certificates must be re-checkable without hidden state. For integer witnesses this is straightforward: ReCheck reconstructs A from the stored list, enumerates the $\binom{n+1}{2}$ sums and products (with $p \leq q$), deduplicates them deterministically, and compares the resulting digests to the committed digests. For algebraic-real witnesses the only additional subtlety is comparison: sorting and deduplication require certified decisions of the form

$$r^{e_a} + r^{e_b} \overset{?}{=} r^{e_c} + r^{e_d}, \qquad r^{e_a} r^{e_b} \overset{?}{=} r^{e_c} r^{e_d},$$

which are resolved via exact arithmetic in $\mathbb{Q}(r)$ and certified isolating-interval refinements. The certificate records the minimal polynomial and an isolating interval for r, together with the comparison transcript required to justify every equality/inequality used by the deterministic deduplication routine. Consequently, the statement "the entry (n, i, j) exists in the atlas" is reducible to running ReCheck on the stored objects.

Additive-strip completeness over $\mathbb{N}$ (Theorem B): reduction and audit

Fix n with $3 \leq n \leq N_{\max}$. Let

$$\mathcal{P}_{\leq 3n-3}(n) := \{(i, j) \in \mathrm{SPP}(n) : i \leq 3n - 3\}.$$

Theorem B asserts that our additive-strip enumerator $\mathsf{EnumAdd}(n)$ produces witnesses realizing *exactly* the set $\mathcal{P}_{\leq 3n-3}(n)$, and that the atlas stores one certified representative for each pair.

The mathematical input is a Freiman-type description of integer sets with small doubling at the threshold $3n - 3$: every $A \subset \mathbb{Z}$ with $|A| = n$ and $|A + A| \leq 3n - 3$ lies in a finite union of explicit model families, parameterized by bounded-complexity data (progression-like containers together with a small number of exceptional elements). We do not re-prove the inverse theorem here; we treat it as a cited structural lemma (Lemma 2) and implement its parameter families verbatim.

The computation then consists of the following deterministic pipeline.

1. *Family enumeration.* For each model family and each admissible parameter tuple, we generate a candidate set A, canonicalize it (via $\mathrm{Can}(A)$), and keep it if it has size n and satisfies the syntactic family constraints.

2. *Size verification.* For each retained candidate, we call Check to obtain $(|A+A|, |AA|)$ and $\mathrm{Cert}(A)$, and we discard candidates with $|A+A| > 3n-3$.

3. *Pair aggregation.* We aggregate realized pairs (i, j), select one representative witness per pair (by a fixed deterministic tie-break rule on canonical keys), and store the representative entry together with its certificate.

The crucial point is how we audit *exhaustiveness* of Step (1) and *non-duplication* across overlapping families. Each enumerator run produces an *audit manifest* consisting of:

- the exact model-family identifiers and the discrete parameter ranges used (which are logically forced by the cited inverse theorem and by n);
- the canonical keys $\mathrm{Can}(A)$ of *all* generated candidates before filtering by realized pairs, listed in a deterministic total order;
- the total candidate count $C_{\text{add}}(n)$, the number of candidates surviving $|A+A| \le 3n-3$, and the number of distinct realized pairs $P_{\text{add}}(n)$;
- a content hash $H_{\text{add}}(n) = \mathrm{SHA256}(\mathrm{Serialize}(\mathrm{Manifest}_{\text{add}}(n)))$ committing to the entire list.

The atlas release manifest includes the values $C_{\text{add}}(n)$ and $P_{\text{add}}(n)$ as exact integers and includes the hash $H_{\text{add}}(n)$ as an immutable reference. The audit logic is then: an independent auditor re-runs $\mathsf{EnumAdd}(n)$, recomputes the manifest, checks that the hash matches, and verifies that every stored representative entry for a pair with $i \le 3n-3$ appears among the verified candidates for that run.

Multiplicative-strip completeness over $\mathbb{R}_+$ (Theorem C): reduction and audit

Fix n with $3 \le n \le N_{\max}$ and define

$$\mathcal{Q}_{\le 3n-3}(n) := \{(i,j) \in \mathrm{SPP}_{\mathbb{R}_+}(n) : j \le 3n-3\}.$$

Theorem C is the multiplicative analogue: EnumMul(n) produces witnesses in $\mathbb{R}_+$ realizing exactly $\mathcal{Q}_{\leq 3n-3}(n)$, and the atlas stores one certified representative per pair.

Here the structural reduction has two steps. First, by a multiplicative small-doubling inverse statement, any $A \subset \mathbb{R}_+$ with $|A| = n$ and $|AA| \leq 3n-3$ is, after scaling, contained in a geometric progression

$$\{1, r, r^2, \ldots, r^L\}, \qquad L \leq 2n-2,$$

so the witness is determined by an exponent set $E \subset \{0, \ldots, L\}$ of size n together with the ratio $r > 1$. Second, if $|A + A|$ is not maximal (i.e. there are nontrivial additive collisions among the r^e), then Lemma 3 implies the existence of indices producing a sparse-polynomial constraint

$$r^j - r^\ell - r^k + 1 = 0,$$

with bounded exponents. Thus, potential ratios r are algebraic of controlled complexity, and candidates can be enumerated by bounded exponent patterns.

Algorithmically, EnumMul(n) enumerates:

- all admissible exponent containers of length $L \leq 2n-2$ (up to the normalization $\min E = 0$ and $\gcd(E) = 1$),
- all bounded sparse-polynomial patterns forced by potential sum collisions, and hence a finite list of candidate minimal polynomials for r,
- all real roots $r > 1$ of these polynomials (specified by isolating intervals), and all exponent subsets E consistent with the intended collision pattern,

and then calls Check (with algebraic-number arithmetic) to compute and certify $(|A + A|, |AA|)$ for each candidate $A = \{r^e : e \in E\}$. The multiplicative strip filter is $|AA| \leq 3n - 3$.

The audit artifacts parallel the additive case but must additionally commit to the algebraic specification of r. Each run produces a manifest containing:

- the complete list of enumerated canonical keys $\mathrm{Key}_{\mathbb{R}_+}(A)$ (exponent set, minimal polynomial, isolating interval, and root index), in deterministic order;
- the total candidate count $C_{\mathrm{mul}}(n)$, the number surviving $|AA| \leq 3n-3$, and the number of realized pairs $P_{\mathrm{mul}}(n)$;
- a content hash $H_{\mathrm{mul}}(n)$ committing to the manifest.

Independently of the enumerator, an auditor can re-check each representative entry by running ReCheck on its certificate; the manifest hash is used only to audit exhaustiveness of the candidate generation.

Reproducibility: deterministic run scripts and cross-checks

Every atlas release includes run scripts that reproduce the manifests and the representative-entry selection from scratch. Concretely, for each n we publish a command line of the form

```
atlas enum-add -n n -sum-bound (3n−3) -emit-manifest
```

and

```
atlas enum-mul -n n -prod-bound (3n−3) -emit-manifest,
```

together with a fixed build recipe (compiler version, exact dependency hashes, and the verifier/enumerator commit identifiers). No randomness is used; any pruning is logically redundant and, if present for performance reasons, is recorded in the manifest and must be verified to be completeness-preserving by explicit coverage checks (e.g. by logging the pruned parameter regions and proving they are subsumed by other enumerated regions).

Beyond manifest-hash agreement, we implement three independent sanity layers:

1. *Small-n brute force.* For n in a range where naive enumeration is feasible, we compare the realized pairs from the model-family enumerator to brute-force enumeration in a bounded ambient interval, after canonicalization. Agreement here validates both the implementation and the intended interpretation of the model families.

2. *Cross-implementation agreement.* We provide an independent reference checker for certificates, and we require exact agreement of (i, j) and digests on a representative sample of witnesses across all $n \leq N_{\max}$.

3. *Internal consistency invariants.* We verify constraints such as $2n-1 \leq |A+A|, |AA| \leq n(n+1)/2$, invariance under allowed normalizations (Lemma 1), and monotonicity/validity conditions on canonical keys.

A completeness theorem in this paper should therefore be read as: *assuming the cited inverse theorem(s) and assuming the published manifests are reproduced by the published deterministic scripts, the atlas contains exactly the claimed set of pairs, each supported by an independently re-checkable certificate.*

8 8. Certified bounds on α_n and extremal witness tables: how witnesses are generated; how certificates are produced; limitations (upper bounds vs optimality).

Certified bounds on α_n and extremal witness tables

The completeness theorems above concern the *exact* set of realizable pairs in the additive and multiplicative strips. Independently of those strip statements, we also use the atlas to derive *certified upper bounds* on the classical quantity

$$\alpha_n := \min_{A \subset \mathbb{N},\, |A|=n} \max\{|A+A|, |AA|\}.$$

Our contribution here is not a new lower-bound argument for α_n, but a mechanism for producing, storing, and re-checking explicit witnesses A_n with small $\max\{|A_n+A_n|, |A_nA_n|\}$, thereby yielding an *unconditional, certificate-backed inequality* $\alpha_n \le u_n$ for each $3 \le n \le N_{\max}$ (Corollary D).

How candidate witnesses are generated. For each fixed n, we maintain a finite pool $\mathcal{G}(n)$ of candidate witnesses $A \subset \mathbb{N}$ of size n. This pool is the union of several deterministically defined sources.

(1) *Strip enumerators.* Every witness produced by $\mathsf{EnumAdd}(n)$ and every integer witness arising from the integer-specializations of $\mathsf{EnumMul}(n)$ (when the algebraic ratio happens to be rational and yields an integer set after scaling) is included in $\mathcal{G}(n)$. This source is exhaustive only under the respective strip constraints, but it is systematic and supplies a large family of structured sets (near-progressions, low-dimensional containers, short geometric-progression traces) that frequently lie close to the lower envelope of $\max\{|A+A|, |AA|\}$.

(2) *Library constructions.* We include a fixed, versioned list of explicit construction templates parametrized by n, such as:

- initial segments $\{1, 2, \ldots, n\}$ and affine images thereof (after canonical reduction);
- multiplicatively structured sets (e.g. divisors of a smooth number, truncated sets of the form $\{p_1^{e_1} \cdots p_t^{e_t} \le B\}$ restricted to size n);
- hybrids formed by gluing a short arithmetic progression to a multiplicative core, followed by deterministic pruning to size n.

These families are not claimed to be exhaustive in any mathematical sense; their role is to provide reproducible baselines and to capture constructions known empirically to balance additive and multiplicative expansion.

(3) *Deterministic local search seeded by structured inputs.* Starting from seeds in (1)–(2), we run deterministic improvement procedures (with no randomness and with fully logged moves) that attempt to reduce the objective

$$F(A) := \max\{|A+A|, |AA|\}$$

subject to $|A| = n$. Concretely, we implement bounded-depth exchange moves of the form

$$A \mapsto (A \setminus \{x\}) \cup \{y\},$$

where x ranges over the elements of A and y ranges over a fixed finite proposal set depending only on n and a scale parameter derived from $\max A$. The acceptance rule is purely greedy with deterministic tie-breaking. We emphasize that this procedure is a *generator* rather than a verifier: it proposes candidates that are subsequently certified by Check, and it is included only to expand the witness pool beyond the strip-enumerated families.

(4) *Community submissions.* The atlas accepts externally proposed A together with optional provenance metadata. Submissions enter $\mathcal{G}(n)$ only after canonicalization and successful verification by Check; duplicates are eliminated by canonical keys. In particular, no trust is placed in the submitter's computed values of $|A+A|$ and $|AA|$.

Selection of the bound witness A_n. Given $\mathcal{G}(n)$, we compute

$$u_n := \min_{A \in \mathcal{G}(n)} F(A),$$

and we select A_n as a deterministic representative attaining this minimum. To ensure reproducibility across releases, we fix a total order refining the objective value, for example by comparing triples

$$\Big(F(A),\ |A+A|,\ \mathrm{Can}(A)\Big)$$

lexicographically. The chosen A_n is then stored as the designated *upper-bound witness* for α_n, together with the full verification certificate produced by $\mathsf{Check}(A_n)$.

This definition makes the statement $\alpha_n \le u_n$ tautological once $F(A_n) = u_n$ is certified. It also makes clear what is and is not being claimed: u_n is the best value found within our declared candidate pool, not necessarily the true minimum defining α_n.

Certificate production and what is recorded. For each stored upper-bound witness A_n, we publish $\mathrm{Cert}(A_n)$ in the same sense as for strip entries. In the integer setting the certificate contains: the sorted list of elements of A_n (in canonical form), deterministic digests of $A_n + A_n$ and A_nA_n (computed from the complete lists of $\binom{n+1}{2}$ sums/products with $p \le q$), and the resulting sizes. The independent checker ReCheck recomputes these lists and verifies agreement with the committed digests. Thus the inequality $\alpha_n \le u_n$ is reduced to a finite re-check with no reliance on enumerator correctness, heuristic search logic, or unstated pruning.

In addition, for each n we record minimal metadata sufficient to interpret the bound in context: the witness source tag (strip enumerator / library / local search / submission), the atlas version identifiers for Check and ReCheck, and the canonicalization transcript (translation and dilation applied). This metadata is not logically necessary for correctness, but it is necessary for reproducibility and for comparing improvements across releases.

Extremal witness tables beyond α_n. Alongside the single-number bounds u_n, we maintain *extremal witness tables* summarizing portions of the Pareto frontier of $\mathrm{SPP}(n)$. The simplest such tables are obtained by fixing a target sumset size i (or product-set size j) and minimizing the other coordinate among stored certified entries:

$$j_{\min}(n;i) := \min\{\, j : (i,j) \in \mathrm{SPP}(n) \text{ and certified in the atlas} \,\},$$

and similarly $i_{\min}(n;j)$. In the additive strip, Theorem B implies that $j_{\min}(n;i)$ is *exact* for every $i \le 3n-3$, because every realizable pair in that region appears in the atlas with a certificate. Likewise, in the multiplicative strip over $\mathbb{R}_+$, Theorem C implies that $i_{\min}(n;j)$ is exact for every $j \le 3n-3$ in the real domain. These extremal tables are therefore a mixture of (i) exact frontier data in the strip regions and (ii) best-known data outside the strips, where completeness is not asserted.

We stress a point of interpretation: the atlas may contain multiple witnesses for the same pair (i,j), but the extremal tables store exactly one representative per selected extremal criterion, again chosen deterministically by canonical keys. This policy keeps the tables stable under re-running the same pipelines, while allowing the underlying database to grow with additional non-extremal examples.

Limitations: upper bounds versus optimality. The bounds u_n are *certified* but not in general *optimal*. There are two distinct reasons.

(1) *Incompleteness outside the strips.* Theorems B and C are completeness statements only in the regions $|A+A| \le 3n-3$ (integers) and $|AA| \le 3n-3$ (reals). The definition of α_n imposes no such restriction, and it is entirely possible that an optimal set A for α_n lies outside both small-doubling regimes simultaneously. Our current methods do not provide an inverse-theorem-based parameterization for this global regime, hence we do not claim to enumerate all candidates relevant to α_n.

(2) *Absence of certified lower bounds.* To prove $\alpha_n = u_n$, one must also show $\alpha_n \ge u_n$, i.e. a lower bound applying to *all* A of size n. The atlas

infrastructure is not, by itself, a lower-bound mechanism. It can support lower-bound proofs by providing counterexample searches and exact frontier data in controlled regions, but any claim of optimality necessarily requires additional mathematics (or a separate exhaustive enumeration covering all A up to canonicalization, which is infeasible at $n \approx 50$ in $\mathbb{N}$ without further structure).

For small n where brute-force enumeration in a bounded interval is feasible, we may optionally include an *auxiliary* label indicating that the best-known witness matches the brute-force optimum within that experiment; however, such labels are explicitly scoped and are not part of the main completeness theorems. In all cases, the only unconditional statement attached to A_n is the certified inequality $\alpha_n \le u_n$.

Release discipline. Finally, we treat the α_n witness table as versioned data. A new release may improve some u_n by adding new generators or new submissions, but any change is accompanied by new certificates and updated provenance metadata. Because ReCheck is deterministic and self-contained, an independent user can validate the entire table entry-by-entry without re-running any enumeration. In this way, the atlas separates three logically distinct components: (i) the truth of computed sizes (certificates), (ii) completeness in specified strip regions (enumeration plus audit manifests), and (iii) heuristic progress on global objectives such as α_n (best-known witnesses with certified values but without optimality claims).

9 9. Conjecture mining and 'River of Ignorance' benchmarks: derived statistics (frontiers, gaps, candidate void inequalities), benchmark tasks for generators, and a roadmap for $n > N_{\max}$.

Conjecture mining and 'River of Ignorance' benchmarks

Beyond serving as a repository of certified witnesses, the atlas is designed to function as a data source for formulating and stress-testing conjectures about the geometry of the set

$$\mathrm{SPP}_X(n) = \{(|A+A|, |AA|) : A \subset X,\ |A| = n\},$$

both in regimes where we can prove completeness (the strip regions) and in regimes where we only have best-known samples. We therefore compute and publish a collection of *derived statistics* from the certified entries, together with benchmark tasks intended to drive the development of better generators and to guide investigations for $n > N_{\max}$.

Certified frontiers and derived envelopes. For each $n \le N_{\max}$ we define the (coordinatewise) Pareto frontier of certified points

$$\mathcal{F}_X(n) := \Big\{(i,j) \in \mathrm{SPP}_X(n) : \nexists\, (i',j') \in \mathrm{SPP}_X(n) \text{ with } i' \le i,\ j' \le j,\ (i',j') \ne (i,j)\Big\},$$

and we compute a certified subset of $\mathcal{F}_X(n)$ from the atlas entries by direct dominance tests. In the additive strip over $\mathbb{N}$, Theorem B upgrades this to an *exact* frontier in the region $i \le 3n-3$; similarly, Theorem C yields an exact frontier in the region $j \le 3n-3$ over $\mathbb{R}_+$. We record, for each feasible i in the additive strip, the exact extremal function

$$j_{\min}(n;i) := \min\{j : (i,j) \in \mathrm{SPP}_{\mathbb{N}}(n)\}, \qquad (2n-1 \le i \le 3n-3),$$

and for each feasible j in the multiplicative strip (real domain) the exact function

$$i^{\mathbb{R}_+}_{\min}(n;j) := \min\{i : (i,j) \in \mathrm{SPP}_{\mathbb{R}_+}(n)\}, \qquad (2n-1 \le j \le 3n-3).$$

These functions are the primary inputs to conjecture mining, since they are not sampling artifacts: in the stated strip ranges they are determined by exhaustive, audited enumeration.

Gap profiles and certified voids inside strips. A recurring phenomenon in small-doubling regimes is that many integer lattice points (i,j) in the ambient rectangle are simply *not* realizable. We therefore publish, for each n, the realized set of pairs in the certified strip and the complementary set of *certified void points*. Concretely, in the additive strip we define

$$\mathcal{V}_{\mathrm{add}}(n) := \{(i,j) : i \le 3n-3,\ (i,j) \notin \mathrm{SPP}_{\mathbb{N}}(n)\}.$$

By Theorem B, membership in $\mathcal{V}_{\mathrm{add}}(n)$ is a theorem (computer-audited): it is not merely "not seen", but "proven absent" within that strip. We summarize $\mathcal{V}_{\mathrm{add}}(n)$ by *gap profiles*, such as the set of missing j-values at each fixed i, the lengths of consecutive missing intervals, and the minimal observed increment of $j_{\min}(n;i)$ as i increases. Analogous statistics are recorded for the multiplicative strip over $\mathbb{R}_+$.

These void profiles are useful in two ways. First, they provide immediate falsification tests for would-be constructions: a proposed witness claiming a forbidden pair must fail ReCheck or must violate the stated strip constraint. Second, they suggest structural rigidity: if, for example, $j_{\min}(n;i)$ only takes values in a sparse arithmetic progression for many i, then it is natural to conjecture that witnesses on the lower envelope belong to a small number of model families.

Candidate void inequalities and extrapolation beyond certified regions. Outside the strip regions we cannot, at present, certify non-realizability. Nevertheless, we can record *candidate void inequalities* suggested by the data and by structural heuristics. A candidate void inequality is an inequality of the form

$$\Phi_n(i,j) \geq 0$$

that holds for all certified points at size n (or for all certified points across a range of n) and is conjectured to hold for all of $\mathrm{SPP}(n)$ (or for all sufficiently large n). We emphasize that such inequalities are labeled as conjectural unless they are implied by a proved completeness statement in a region where $\Phi_n(i,j) < 0$ would force (i,j) into a certified void set.

We find it useful to work in normalized coordinates. One convenient normalization is

$$K_n(m) := \frac{\log m}{\log n}, \qquad \kappa(A) := \big(K_n(|A+A|),\, K_n(|AA|)\big),$$

so that trivial bounds place $\kappa(A)$ in a compact region as n varies. In these coordinates we can compare different n on the same scale and look for stable shapes (e.g. a limiting lower envelope). Candidate void inequalities then take forms such as $\kappa_2 \geq f(\kappa_1)$ with f inferred empirically and constrained by certified strip data at small coordinates.

The 'River of Ignorance' as a benchmark region. We designate as a standing benchmark the task of producing certified witnesses whose normalized statistics fall in the empirically sparse region around

$$\kappa(A) \approx (1.8, 1.9) \quad \text{for} \quad n \approx 40,$$

or more generally in a fixed neighborhood of such points as n varies. Informally, this is a "river" in the $(K_n(|A+A|), K_n(|AA|))$ plane where, despite extensive structured enumeration in the strips and substantial heuristic search, the atlas contains few or no entries. The point of naming such a region is methodological: it prevents us from overfitting generators to regimes already well-covered by near-progressions or short geometric-progression traces.

We treat this benchmark as follows. For each release we publish the best-known certified value of

$$\Delta_n(\kappa_0) := \min_{|A|=n} \|\kappa(A) - \kappa_0\|_\infty$$

for selected targets κ_0 in the river region, where the minimum is taken over certified atlas entries (hence it is an upper bound on the true minimum). A generator improves the benchmark by providing a new certified witness decreasing $\Delta_n(\kappa_0)$, regardless of whether it improves α_n.

Benchmark tasks for generators. To make progress reproducible and comparable, we publish a small collection of benchmark tasks, each posed as a finite search problem whose outputs are validated solely by certificates. Typical tasks include:

- (*Threshold task*) For a given n and target T, produce a certified $A \subset \mathbb{N}$ with $|A| = n$ and $\max\{|A+A|, |AA|\} \le T$.
- (*Box-hitting task*) For a given n and rectangle $[i_1, i_2] \times [j_1, j_2]$, produce a certified witness with $(|A+A|, |AA|)$ in the box.
- (*Frontier-improvement task*) For a given n and fixed i (or j), produce a certified witness improving the current best-known j at that i (or improving i at that j), with the understanding that in strip regions the target is exact and outside strips it is "best-known".
- (*Diversity task*) Produce M pairwise non-canonically-equivalent witnesses whose (i, j) occupy distinct bins in a coarse grid of normalized coordinates, to encourage coverage rather than single-objective optimization.

Each task is scored by the verifier outputs and by a deterministic tie-breaker on $\mathrm{Can}(A)$, and each submitted solution must include $\mathrm{Cert}(A)$ so that the scoreboard is checkable without re-running any generation code.

Mining structural conjectures from canonical witnesses. The atlas entries themselves contain more than just the pair (i, j): they contain canonical representatives and, in the real multiplicative strip, algebraic ratio data. We therefore record additional features for each witness, such as container length in the Freiman model (additive strip), geometric progression length and defining sparse polynomial (multiplicative strip), and simple arithmetic invariants (e.g. smoothness of elements, divisor-closure scores, or additive energy proxies computed exactly from the multiset of sums). Correlating these features with position on the frontier suggests conjectures of the form:

> *Frontier points with $i \le 3n - 3$ are realized only by k-parameter subfamilies of the Freiman models, and the minimizers of $j_{\min}(n; i)$ lie in a distinguished subfamily characterized by an explicit combinatorial pattern.*

We do not attempt to formalize such conjectures within the atlas paper itself; rather, we supply the certified evidence and an interface for extracting the relevant subcollections (by parameter tags) so that the conjectures can be pursued mathematically.

Roadmap for $n > N_{\max}$. The choice of $N_{\max} \in \{50, 60\}$ reflects a balance between exhaustive strip enumeration and practical certification sizes. For $n > N_{\max}$ we envision three complementary extensions.

(1) *Extend certification without extending completeness.* The verifier and certificate formats scale to larger n essentially quadratically in n (due to $\binom{n+1}{2}$ pairwise operations), and thus can support a growing repository of best-known witnesses well beyond $N_{\max}$. This enables meaningful, checkable progress on global objectives (including α_n upper bounds and river benchmarks) even when enumeration is infeasible.

(2) *Push the strip boundaries with additional structure.* On the additive side, one may seek to enlarge the provably enumerable regime beyond $3n - 3$ by incorporating stronger inverse results or by stratifying by additive dimension and enumerating model families with an additional parameter. On the multiplicative side, one may broaden the ratio enumeration mechanism by allowing multiple sparse-polynomial constraints simultaneously (arising from multiple forced sum-collisions) and by systematically managing the resulting algebraic-number degrees.

(3) *Hybrid audited search.* Between full enumeration and unrestricted heuristics lies a middle ground: audited searches constrained by provable containers (e.g. bounded-rank generalized progressions additively, bounded-length geometric progressions multiplicatively) but without claiming that the constraints capture *all* extremizers. Such searches can still produce certified void statements *conditional on the container hypothesis*, and can be used to test the plausibility of conjectures suggesting that optimizers must lie in certain model classes.

In all cases, our guiding principle remains the same: we separate (i) certified truth of individual entries from (ii) completeness claims in explicitly delimited regions, and we make the boundary between these components explicit in both the data schema and the published derived statistics.