Mapping Uncharted Symmetries: Machine Discovery in Combinatorics

Eugenio Cainelli [*1] **Lorenzo Luccioli** [1] **Alessandro Iraci** [2] **Michele D'Adderio** [3] **Giovanni Paolini** [*1]

Abstract

Inspired by long-standing open problems in algebraic combinatorics, we show that modern machine learning can meaningfully contribute to verifiable mathematical discoveries. In particular, we focus on the construction of simple mathematical functions under exact distributional constraints, a setting we formalize as Simple Learning Under Rigid Proportions (SLURP). We tackle this problem by introducing two methods: MapSeek-Functional, which models the desired function alternating pseudo-labeling and supervised training steps; and MapSeek-Symbolic, designed to directly produce symbolic formulas. We successfully apply both methods to a research problem in algebraic combinatorics, discovering a new combinatorial interpretation of the q, t-Narayana polynomials arising from representation theory. To our knowledge, this is the first such interpretation based on noncrossing partitions. Using one discovered statistic, we find a combinatorial proof of the symmetry of these polynomials in a previously unsolved case. To streamline verification and reproducibility, we release all code, including a formalization of all the mathematical discoveries of this paper in Lean 4.

1. Introduction

Many important problems in mathematics can be framed as function discovery: given two sets X and Y, find a function $f\colon X \to Y$ that satisfies rigid constraints, yet admits a simple description. Modern machine learning is exceptionally good at fitting finite datasets, and its inductive biases often favor low-complexity solutions. In this paper we push these strengths to the limit: we work in settings where success requires *exact* constraint satisfaction rather than high accuracy, and where the desired output is not a black-box predictor but a *symbolic or algorithmic* description, revealing interesting structure and suitable for mathematical verification.

[*]Equal contribution [1]University of Bologna, Bologna, Italy [2]Pegaso University, Naples, Italy [3]University of Pisa, Pisa, Italy. Correspondence to: Eugenio Cainelli <eugenio.cainelli2@unibo.it>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

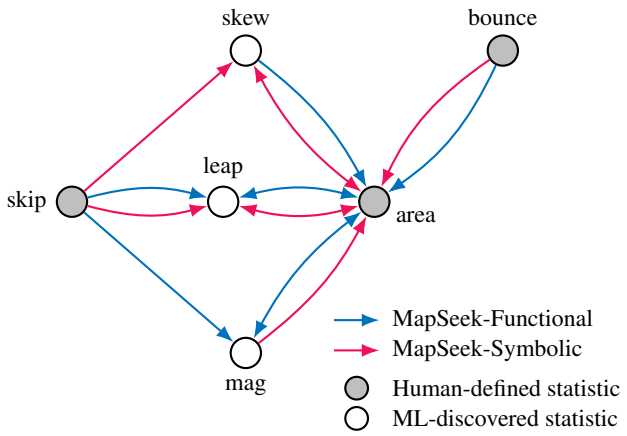


Figure 1. Atlas of notable statistics (nodes). An edge $f \to g$ indicates that, starting from statistic f, one of our methods discovers g as a partner statistic, so that (f, g) provides a combinatorial interpretation of Narayana polynomials $N_{n,3}(q, t)$.

Our driving goal comes from open problems in the field of algebraic combinatorics, asking for functions on combinatorial objects that witness deep algebraic symmetries. Inspired by such problems, we introduce a learning framework called *Simple Learning Under Rigid Proportions* (SLURP), where the task is to discover simple functions that match specified output distributions exactly (Figure 2). In this setting, the constraints deliberately underspecify the function f, so that it is easy to produce example functions, but the challenge is to find simple and mathematically meaningful ones.

To tackle SLURP problems, we propose two radically different machine learning methods. In the first method, *MapSeek-Functional*, we train a model to converge to a desired function via weak supervision from the constraints, alternating pseudo-labeling steps and supervised updates. The second method, *MapSeek-Symbolic*, performs a population-based search on symbolic formulas, iteratively generating and evaluating candidates until an exact solution is discovered.

To demonstrate their effectiveness and the soundness of the SLURP setup, we describe a successful application to the combinatorial study of the q, t-Narayana polynomials

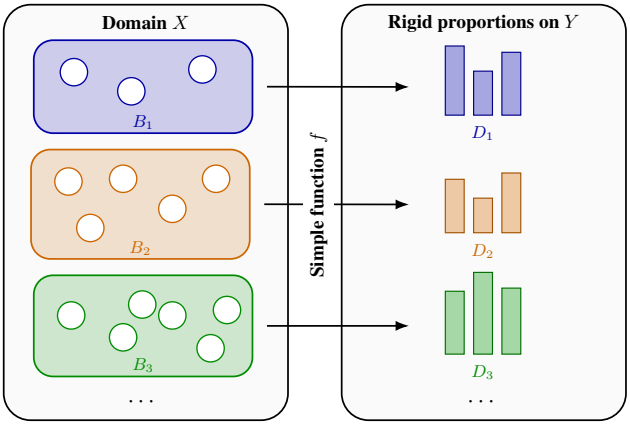


Figure 2. A SLURP instance: the domain X is partitioned into bags, and supervision is provided via per-bag output distributions.

$N_{n,k}(q,t)$ arising from representation theory. In this context, an important problem is to find interpretations of q,t-polynomials in terms of pairs of simple functions (*statistics*) on a set of combinatorial objects; often, one candidate statistic is known while a partner statistic is missing. We find that both methods are effective in discovering statistics and yield new combinatorial interpretations of $N_{n,3}(q,t)$. Successful pairings are represented as directed edges in Figure 1.

One interpretation, independently suggested by both our ML pipelines, leads us to a new combinatorial proof of the symmetry $N_{n,3}(q,t) = N_{n,3}(t,q)$ via an explicit bijection that exchanges the *skip* and *leap* statistics. This constitutes one of the first combinatorial proofs of q,t-symmetry across all of algebraic combinatorics; constructing such exchanging bijections is widely regarded as an extremely difficult open problem. To streamline verification, we formalize the full proof in the Lean 4 proof assistant, which is increasingly becoming a standard tool for formalizing mathematics.

The main contributions of this paper are as follows:

- Drawing inspiration from algebraic combinatorics problems, we formalize the SLURP framework.
- We introduce two methods for SLURP: a functional self-training method (MapSeek-Functional) and a symbolic search method (MapSeek-Symbolic).
- We propose the task of discovering statistics as a testbed for SLURP. In particular, we release the *skip-pairing benchmark*, where the aim is to discover interpretations of q,t-Narayana polynomials.
- We use MapSeek-Functional and MapSeek-Symbolic to discover interpretable and mathematically interesting statistics, and build upon them to derive new combinatorial results.
- We construct an explicit exchanging bijection for the symmetry $N_{n,3}(q,t) = N_{n,3}(t,q)$, thus solving Open Problem 4.1 in the case $k = 3$, and we provide the first interpretation of $N_{n,k}(q,t)$ in terms of noncrossing partitions for all k. We formalize these results in Lean 4.
- We release the code for MapSeek, the skip-pairing benchmark, and the full Lean 4 formalization.[1]

Our broader goal is to build flexible tools that, in the hands of expert mathematicians, can leverage human intuition and extend their reach.

2. Related Work

Machine Learning for Mathematical Discovery. The application of machine learning to mathematics has attracted significant interest in recent years. A substantial body of work focuses on automated theorem proving, often leveraging large language models (Achim et al., 2025; Chen et al., 2025). Separately, a growing line of research employs ML not as a prover but as an exploratory tool for discovering mathematical objects, constructing counterexamples, and formulating conjectures. Examples include searching for extremal graphs (Wagner, 2021), studying algebraic invariants (Davies et al., 2021), and synthesizing Lyapunov functions for dynamical systems (Alfarano et al., 2024); see also (Williamson, 2023). Our contribution stands within this paradigm, with the aim of constructing functions (combinatorial statistics and bijections) that satisfy exact constraints.

Automated Discovery in q,t-Combinatorics. The search for combinatorial interpretations and exchanging bijections is a central challenge in algebraic combinatorics. Recently, ML has been applied to this specific domain: in (Huang et al., 2025), a Transformer was trained to approximate the *zeta map* (Haglund, 2008), recovering a new algorithmic interpretation of this known map by analyzing attention weights; in (Brown et al., 2025), LLMs and evolutionary algorithms were used to generate bijection candidates. However, these contributions have primarily served as proofs of concept, re-deriving established results. To our knowledge, our work is the first to use ML to propose *previously unknown* combinatorial statistics, used to construct an exchanging bijection that solves an open problem in q,t-combinatorics.

Learning from Label Proportions (LLP). Our SLURP framework shares with LLP the use of bag-level distributional constraints rather than individual labels (Quadrianto et al., 2009; Scott & Zhang, 2020; Havaldar et al., 2024; Busa-Fekete et al., 2023). However, the settings differ fundamentally. In LLP, a fixed unknown distribution generates instances, and distributional constraints serve as a proxy for hidden labels; in contrast, for us distributional constraints are the full problem specification. Moreover, LLP aims to

[1] https://github.com/LorenzoLuccioli/qtLearning.

learn a classifier with good generalization, typically allowing approximate constraint satisfaction, whereas SLURP requires exact satisfaction along with a concise symbolic or algorithmic description (and, in our applications, solutions that can be verified on infinite domains).

Program Synthesis (PS). The objective of recovering explicit symbolic formulas under symbolic constraints situates SLURP within the broader scope of PS (Gulwani et al., 2017; Sun et al., 2022; Chaudhuri et al., 2021). Formally, SLURP can be cast as a PS instance by making the hypothesis class explicit, e.g. by defining a domain specific language or bounding syntactic complexity. Nevertheless, SLURP differs from typical PS work because the domains are infinite and the constraints are distributional (bag-level) rather than pointwise. Crucially, standard automated verification tools like SMT solvers cannot check constraint satisfaction over infinite domains. We therefore present SLURP as a proof-driven discovery framework rather than a PS variant: we rely on machine learning to propose candidates from finite data, while the simplicity requirement ensures these candidates are amenable to the mathematical proof.

3. Simple Learning Under Rigid Proportions

Inspired by algebraic combinatorics, we introduce a new learning framework called *Simple Learning Under Rigid Proportions* (SLURP). This framework formalizes settings where the goal is to discover a *simple* function $f\colon X \to Y$ that satisfies *rigid proportion constraints*. We first describe the problem formulation, and then our two approaches.

3.1. Problem Formulation

The SLURP constraints are expressed as follows: the domain X is partitioned into bags $B_1, B_2, \ldots,$ and each bag B_i comes with a target distribution D_i on Y (Figure 2). The goal is to find a function $f\colon X \to Y$ such that:

(1) The outputs on each bag B_i are distributed according to D_i exactly (**rigid proportion constraint**).

(2) The function f is provided in the form of a symbolic description (e.g., a closed formula or algorithm), evaluable within a given time bound (e.g., polynomial), and as short as possible (**simplicity requirement**).

The simplicity requirement is not merely aesthetic but a fundamental necessity. Without it, one could trivially satisfy the rigid proportion constraints by arbitrarily assigning values to domain elements in a way that matches the target distributions, without revealing any mathematical insight.

A Toy Instance. The role of simplicity is easiest to see in a deliberately elementary example. Let $X = \mathbb{N}$ and partition it into consecutive bags

$$B_m = \{10m, 10m+1, \ldots, 10m+9\}, \qquad m \geq 0.$$

Suppose that $Y = \{0, 1\}$ and that, for every bag B_m, the target distribution requires exactly five zeros and five ones:

$$D_m(0) = 5, \qquad D_m(1) = 5.$$

These constraints do not label any individual integer. They only specify how many elements of each bag must be assigned to each output value. Thus exact satisfaction alone is easy to obtain: in every bag one may choose an arbitrary subset of five elements to receive value 0, and send the rest to 1. Such choices solve the finite constraints bag by bag, but are not necessarily explained by a simple rule. Two possible simple solutions are the following:

$$f(n) = n \bmod 2, \quad g(n) = \left\lfloor \frac{n \bmod 10}{5} \right\rfloor.$$

SLURP does not require uniqueness: the data specify only aggregate output multiplicities, not pointwise values. The task is to find an exact solution whose description is simple enough to expose structure.

The problems we introduce in the following sections provide more interesting SLURP instances, where the domain X consists of richer mathematical objects.

Key Characteristics. SLURP differs from traditional learning settings, such as LLP, in several fundamental ways. First, the constraints are *rigid*, meaning that solutions must satisfy them exactly, not approximately. While we can measure how close a function is to satisfying the constraints (to use it as a learning signal, or to track progress during learning), there is no notion of an "almost correct" solution. Any deviation from exact satisfaction means the problem remains unsolved.

Second, the aggregated constraints are not a weak proxy of a ground-truth solution, but rather the true end-goal of our framework. Thus, the problem is underdetermined and multiple simple functions may be viable solutions. The challenge is to discover functions that are both feasible and mathematically interesting.

Learning on Infinite Domains. When the domain X is infinite, as in the practical scenarios we tackle in this paper, we limit the learning process to only use a finite training subset $X_{\text{train}} \subset X$ consisting of a union of a finite number of bags B_i. The symbolic description of a solution $f\colon X_{\text{train}} \to Y$ to the finite SLURP problem can be interpreted as a candidate extension $\tilde{f}\colon X \to Y$; however, the only way to obtain a true guarantee that $\tilde{f}$ satisfies the rigid proportion constraints on the whole infinite domain is to prove it mathematically.

3.2. MapSeek-Functional

In the functional approach to SLURP, we train a model $f_\theta \colon X \to Y$ by alternating two steps: (i) *constraint projection* and (ii) *supervised update*. During constraint projection, we consider the function encoded by the current model f_θ and find the closest function $\hat{f} \colon X \to Y$ satisfying the rigid proportion constraints. Typically, this means solving

$$\min_{\hat{f}} \sum_{x \in X} d\Big(f_\theta(x), \hat{f}(x)\Big)\,,$$

where d is a distance function on Y. Then, in the supervised update, the pseudo-labels $\hat{y} = \hat{f}(x)$ are used as targets to run multiple SGD updates on f_θ; in practice, we run one full epoch at every supervised update, setting aside a validation subset that is not used for gradient updates. Note that the validation set undergoes the constraint projection step.

This procedure is repeated until f_θ satisfies the rigid proportion constraints (so, the projection step does nothing) or a maximum number of iterations is reached and the run is discarded. It can be viewed as a self-training optimization resembling the classical expectation-maximization algorithm.

The stopping criterion is strict: a training run is considered successful only if the rigid proportion constraints are perfectly satisfied. In particular, the model is required to generalize to the validation set in order to fit the constraints, thus accepting only sufficiently well-behaved functions. The size of the validation set is effectively a hyperparameter requiring calibration: enlarging the validation set decreases the probability of convergence, while empirically filtering out less interpretable runs.

After convergence, the model f_θ encodes a function $X \to Y$; in our experiments we are often able to manually distill this function into a symbolic expression. In general, distillation could be automated via symbolic regression or program synthesis (optionally after ensembling multiple runs).

3.3. MapSeek-Symbolic

The symbolic approach begins by defining an alphabet Σ of primitive symbols from which we construct a formal language $\mathcal{L} \subseteq \Sigma^*$ of syntactically valid formulas. Each formula $\varphi \in \mathcal{L}$ defines a function $f_\varphi : X \to Y$ through its explicit algorithmic description, ensuring that functions expressible within $\mathcal{L}$ are inherently simple by virtue of their compact symbolic representation.

We define a distance $\delta : \mathcal{L} \to \mathbb{R}_{\geq 0}$ that quantifies how much f_φ violates the rigid proportion constraints. Our aim is to discover formulas $\varphi \in \mathcal{L}$ that achieve $\delta(\varphi) = 0$, i.e. satisfying the constraints. While any such formula technically solves the problem, we prefer shorter expressions, that are more likely to reveal mathematical structure and admit formal verification on infinite domains.

The search proceeds via an iterative population-based search, typically through the deep cross-entropy method using a generative sequence model g_θ. The process alternates between (i) a *generation phase*, where a batch of candidate formulas is proposed, and (ii) an *evaluation phase*, where the distance δ is computed for each candidate. The information from high-performing candidates—those with the lowest distance values—is aggregated to bias the generation mechanism in the subsequent iteration toward more promising regions of the search space. The loop continues until a formula $\varphi \in \mathcal{L}$ achieving $\delta(\varphi) = 0$ is discovered, returning an exact symbolic solution.

4. q, t-Combinatorics Setup

Within algebraic combinatorics, the area of q, t-combinatorics studies families of bivariate polynomials $F(q, t)$ with nonnegative integer coefficients arising in a variety of mathematical domains such as symmetric functions, representation theory, algebraic geometry, knot theory, probability, and integrable systems (Haglund, 2008; D'Adderio et al., 2021). Two recurring problems motivate much of the subject.

Type 1 problem (combinatorial interpretation): Given $F(q, t)$, find a set X of combinatorial objects and two functions $\mathrm{stat}_1, \mathrm{stat}_2 \colon X \to \mathbb{N}$ (called *statistics*) such that

$$F(q, t) = \sum_{x \in X} q^{\mathrm{stat}_1(x)}\, t^{\mathrm{stat}_2(x)}. \tag{1}$$

Type 2 problem (exchanging bijection). Given a combinatorial interpretation, prove the symmetry $F(q, t) = F(t, q)$ by constructing a bijection $\psi \colon X \to X$ that exchanges the two statistics:

$$\big(\mathrm{stat}_1(\psi(x)), \mathrm{stat}_2(\psi(x))\big) \;=\; \big(\mathrm{stat}_2(x), \mathrm{stat}_1(x)\big).$$

Both types of problems are typically difficult. For example, it took 8 years to find a combinatorial interpretation for the famous q, t-Catalan polynomials (Garsia & Haglund, 2002), whereas a bijective proof of their symmetry remains open. These problems provide a rich source of interesting SLURP instances: one searches for simple statistics and explicit bijections under exact distributional constraints.

4.1. Narayana Polynomials

In this paper we tackle the aforementioned problems on the q, t-Narayana polynomials $N_{n,k}(q, t)$, indexed by integers $n \geq k \geq 1$, arising from the diagonal coinvariants of the symmetric group (Aval et al., 2014; D'Adderio et al., 2022). They refine the classical Narayana numbers $N_{n,k}(1, 1)$, which count many families of objects, including *parallelogram polyominoes* and *noncrossing partitions*.

In (Aval et al., 2014), a combinatorial interpretation is given

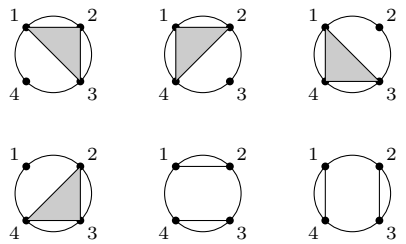


Figure 3. Noncrossing partitions $\mathsf{NC}(4,3)$. The corresponding q,t-Narayana is $N_{4,3}(q,t) = q^2 + qt + t^2 + q + t + 1$.

in terms of polyominoes (see Appendix A), and the symmetry $N_{n,k}(q,t) = N_{n,k}(t,q)$ is proved by algebraic methods. A bijective proof is still missing in general:

Open Problem 4.1. *Find an exchanging bijection proof of the symmetry* $N_{n,k}(q,t) = N_{n,k}(t,q)$.

In this work, we solve Open Problem 4.1 for the first interesting case $k = 3$ (the cases $k \leq 2$ are trivial) via new interpretations of $N_{n,3}(q,t)$ in terms of noncrossing partitions.

4.2. Noncrossing Partitions

A partition of $[n] = \{1, \ldots, n\}$ is a collection of nonempty, pairwise disjoint subsets (called blocks) whose union is $[n]$. For example, $\{\{1,4\},\{2,3,6\},\{5\}\}$ is a partition of $[6]$, and we write it more compactly as $14 \mid 236$, omitting singleton blocks. A partition is *noncrossing* if, when numbers are placed on a circle in increasing order and blocks are drawn by convex hulls, these hulls do not intersect. Noncrossing partitions are counted by the Narayana numbers: the set $\mathsf{NC}(n,k)$ of noncrossing partitions with $n-k+1$ blocks has size $N_{n,k}(1,1)$.

For $k = 3$, we obtain partitions with $n-2$ blocks (Figure 3), which take the following two forms:

- $ab \mid cd$ with $a < b$, $a < c$, and $c < d$;
- abc with $a < b < c$.

We introduce a natural statistic on noncrossing partitions, to be used as a starting point to search for new combinatorial interpretations of Narayana polynomials. This statistic, called *skip*, counts how many numbers are skipped inside each block of a partition, and is defined as follows:

$$\text{skip}(\pi) = \sum_{\beta \in \pi} (\max \beta - \min \beta - |\beta| + 1). \qquad (2)$$

For a visual explanation of the skip statistic, see Figure 9 in the appendix.

5. The Skip-Pairing Benchmark

As a testbed for combinatorial function discovery within the SLURP framework, we propose the concrete problem of constructing simple statistics on noncrossing partitions that, together with the skip statistic (2), provide a combinatorial interpretation of the Narayana polynomials $N_{n,k}(q,t)$.

Practically, this means that noncrossing partitions are grouped by the value $u = \text{skip}(\pi)$; within each group, the number of partitions π with $\text{stat}(\pi) = v$ must equal the coefficient of $q^u t^v$ in $N_{n,k}(q,t)$.

To increase supervision, we optionally refine the constraints via a natural filtration of noncrossing partitions: for a partition π, let $m(\pi)$ be the largest element that belongs to a non-singleton block; group partitions by the pair (m,u) where $m = m(\pi)$ and $u = \text{skip}(\pi)$. In this refined setting, the expected distribution for each group is prescribed by the incremental polynomials $N_{m,k}(q,t) - N_{m-1,k}(q,t)$, which have nonnegative coefficients by Proposition A.3.

To foster research on combinatorial function discovery, we release code to generate the dataset for the skip-pairing problem for any $n \geq k \geq 1$, both in the unrefined and refined setting. In Section 6, we use the refined $N_{14,3}$ dataset, consisting of 2366 partitions grouped into 144 bags, to discover multiple interesting statistics paired with skip for $k = 3$.

6. Experiments and Results

6.1. Statistic Discovery with MapSeek-Functional

Model Architecture. We run the MapSeek-Functional pipeline using a Transformer encoder with 8 layers, 4 attention heads, hidden dimension 128, and feedforward dimension 256. The input to the Transformer is a fixed sequence of n tokens, encoded by taking the real numbers $i/(n-1)$ for $i = 0, \ldots, n-1$ and passing them through a linear embedding layer (with bias).

Each partition π is represented by its block-adjacency matrix $M \in \{0,1\}^{n \times n}$ where $M_{ij} = 1$ if i and j lie in the same block. The partition matrix M enters the model as an attention mask: in alternating layers, self-attention is restricted to entries with $M_{ij} = 1$ (the final layer is unmasked). The model outputs the attention logits from the last layer, averaged over heads and passed through a sigmoid, yielding a score matrix $S \in [0,1]^{n \times n}$. The sum $\sum_{i,j} S_{ij}$ is the model's output, kept as a floating point number for the constraint projection step.

Training Procedure. Within each family (same (m,u)), we sort partitions by the current model's output and assign pseudo-labels matching the required multiplicities from the rigid proportion constraints. If v is the pseudo-label assigned to a partition, we construct a binary $n \times n$ target

matrix by taking the top-v entries of S. The model is trained with a binary cross-entropy loss between S and this target mask. We use the Adam optimizer with learning rate 10^{-4} and batch size 32. In each supervised update step, we run one full training epoch on the current pseudo-labels.

Initial Experiments. We run MapSeek-Functional on the refined $N_{14,3}$ skip-pairing dataset with a budget of 1000 iterations and a 1% validation split. However, we find the outputs of successful runs difficult to distill into simple formulas. Therefore, in subsequent experiments we add extra bias (in the form of warm start) or homogeneity constraints. Note that the ability to incorporate such information into the search allows mathematicians to inject expert knowledge and intuition about the potential structure or properties of the solution. Often a candidate statistic is available but does not satisfy all constraints, and an automated procedure to modify into a working statistic can be extremely helpful.

Experiments with Warm Start. In this setup, we warm-start each run with 20 epochs of supervised pretraining on the following heuristic (human-defined) candidate, which matches the constraints for $n \leq 4$ but not for $n \geq 5$:

$$\sum_{\beta \in \pi} (\mathrm{sum}(\beta) - \max(\beta)) - \tfrac{k(k-1)}{2}.$$

Specifically, the supervised pretraining target for the score matrix S is the $n \times n$ binary matrix T where $T_{ij} = 1$ if i is not the maximum of its block and $1 \leq j \leq i - q(i)$, with $q(i)$ the number of non-largest elements $\leq i$.

We then ran MapSeek-Functional for up to 300 iterations with a 10% validation split. Out of 76 successful runs, 61 runs converged to exactly the same predictions on all 2366 partitions. We name the resulting statistic *mag* (for "magnitude") and manually deduce the following formula:

$$\begin{aligned} \mathrm{mag}(ab \mid cd) &= a + \max(b, c) - 2 - \left\lceil \tfrac{b-a}{2} \right\rceil \\ \mathrm{mag}(abd) &= a + b - 2 - \left\lceil \tfrac{b-a}{2} \right\rceil \end{aligned} \tag{3}$$

(the second formula being the limit case $b = c$ of the first formula). Interestingly, this resembles the t-statistic of (Gorsky et al., 2020), which pairs with the area statistic on certain combinatorial objects, yielding a combinatorial interpretation of other q, t-polynomials.

Experiments with Homogeneity Constraints. As an alternative to warm start, which requires a heuristic candidate, we add supervision to MapSeek-Functional by forcing homogeneity (i.e., translation-equivariance). We say that a statistic is h-homogeneous if, when adding 1 to all elements of each non-singleton block, the statistic increases by h. For example, the skip is 0-homogeneous and the mag is 2-homogeneous. It is easy to see that h-homogeneous statistics paired with skip only exist for $h = 1, 2$ (either of these constraints still leaves huge flexibility).

We enforce this in the projection step, assigning pseudo-labels in increasing value of m while greedily propagating homogeneity.

We ran this setup using a 5% validation split and a budget of 1000 iterations, obtaining 14 successful runs out of 50 for $h = 1$ and 18 out of 50 for $h = 2$.

For $h = 1$, we had 4 runs converging to exactly the same predictions, described by the following formula (we name this the *leap* statistic):

$$\begin{aligned} \mathrm{leap}(ab \mid cd) &= c - 2 + \max(0,\, c - b) \\ \mathrm{leap}(abd) &= b - 2 \end{aligned} \tag{4}$$

(the second formula is the limit case $b = c$ of the first one). Out of the remaining 10 successful runs, 5 differed from leap by less than 0.5%, and the other 5 were much farther from leap and each other ($\geq 5\%$ disagreement).

For $h = 2$, interpretation was slightly less straightforward. We clustered the 18 successful runs, treating runs as 2366-dimensional prediction vectors, using agglomerative hierarchical clustering with average linkage (Figure 4). The highest silhouette score was obtained with 3 clusters of sizes 11, 6, and 1. The medoid of the second cluster yields exactly the mag statistic (3); the medoid of the first cluster yields a modification of mag, with a ± 1 correction based on the parity of $b - a$ when $b < c$.

Overall, for both $h = 1, 2$ we found two ostensibly simple statistics (leap and mag), despite MapSeek-Functional having no explicit bias toward short symbolic descriptions.

Ablation Study on Validation Set Size. A peculiarity of MapSeek-Functional is that the size of the validation set is a hyperparameter trading off success rate and simplicity. Table 1 shows its effect in the $h = 2$ setup.

As the validation split increases, the success rate drops. The number of runs matching mag exactly stays roughly constant, so a larger split appears to filter out less interpretable solutions. Notably, at 10% validation, the two successful runs are nearly identical, and one coincides with mag.

6.2. Statistic Discovery with MapSeek-Symbolic

Formulas and Evaluation. We run MapSeek-Symbolic using the alphabet

$$\Sigma = \{a, b, c, d, i, 0, 1, 2, 3, +, -, \%, (,), \vee, \wedge, \neg, >, <, =\}.$$

A valid formula $\varphi \in \Sigma^*$ defines an integer-valued function f_φ on noncrossing partitions with $k = 3$ by evaluating φ on a 4-tuple (a, b, c, d) (for a 3-block partition abd

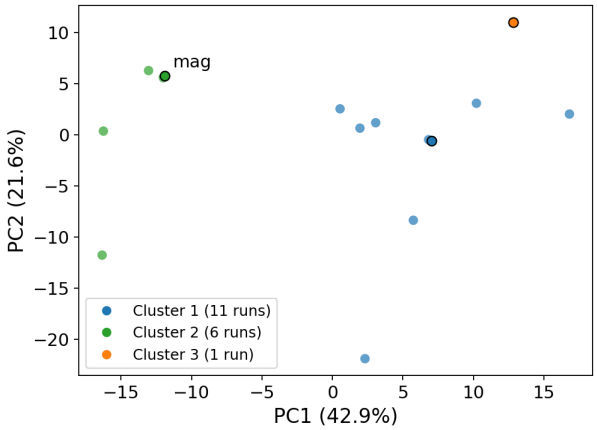


Figure 4. PCA scatter plot of the MapSeek-Functional successful runs (refined $N_{14,3}$ skip-pairing dataset, $h = 2$). Cluster medoids are highlighted. The explained variance of the principal components PC1 and PC2 is shown on the axes.

Table 1. Ablation on validation set size in MapSeek-Functional (refined $N_{14,3}$ skip-pairing dataset, $h = 2$, with 50 runs per setting).

Val	# Success	Med. disagree between runs	Med. disagree with mag	# Exact mag
0%	49	14.8%	18.6%	1
1%	42	11.7%	18.6%	2
2%	27	15.3%	15.3%	3
5%	18	16.9%	16.8%	1
10%	2	0.13%	0.06%	1

we use (a, b, b, d)). Logical operators behave like Python short-circuit operators and can be used to express conditional integer-valued expressions; comparisons produce $0/1$. Given a syntactically valid formula φ, we can treat it as defining a function by summing the formula evaluation over the additional variable i:

$$f_\varphi(\pi) = \sum_{i=1}^{n} \mathrm{eval}\big(\varphi; a, b, c, d, i\big).$$

Distance. We define a distance function $\delta\colon \mathcal{L} \to \mathbb{R}_{\geq 0}$ measuring how far the formula φ deviates from satisfying the constraints. For each bag B_j, the constraints specify a target distribution of output values D_j, prescribed by the refined $N_{14,3}$ skip-pairing dataset. Let $D_j^\varphi = \{f_\varphi(x) \mid x \in B_j\}$ be the distribution of values produced by formula φ on the bag B_j. We define $\mathrm{count}(y, S)$ as the multiplicity of the value y in the distribution S, and δ as a sum of L^1 distances between these distributions:

$$\delta(\varphi) = \sum_j \sum_{y \in Y} \left|\mathrm{count}(y, D_j) - \mathrm{count}(y, D_j^\varphi)\right|. \quad (5)$$

Initial Formulas. We initialize each run by sampling one million unique well-formed formulas, constructed via a recursive grammar that composes arithmetic, comparison, and logical operations. We restrict candidates to the form (expression) % 3; this way, each formula evaluation lies in $\{0, 1, 2\}$ and the resulting value $f_\varphi(\pi)$ lies between 0 and $2n$ (the actual maximum value is $2n - 6$). To maximize search coverage, different runs employ one of 20 distinct hyperparameter configurations, varying recursion depth (2–4), operator probabilities, and length (5–60 characters).

Genetic Algorithm (GA) Baseline. We implemented a GA baseline that optimizes formulas directly using the distance δ as fitness. The GA consistently finds zero-distance formulas yielding the leap statistic (4), but with extreme formula bloat (100+ characters long); this behavior is consistent with the fitness-causes-bloat effect (Silva & Costa, 2009). We applied standard anti-bloat measures (e.g. depth/size limits) but could not obtain consistently compact formulas satisfying the constraints, in accordance with empirical studies that highlight the limits of such regularizers (Poli et al., 2008). Therefore, we moved to a controlled symbolic approach using the deep cross-entropy method (CEM) with a generative sequence model.

Model Architecture. We run the deep CEM pipeline using a Transformer decoder with 4 layers, 8 attention heads, hidden dimension 256, and feedforward dimension 256. We employ a character-level tokenizer.

Training Procedure. Each run begins by evaluating the distance $\delta(\varphi)$ for all 1 million randomly generated formulas in the initial population. We select the top 5% formulas with the lowest distance values to form the elite subset, from which we reserve 1000 for validation and use the remaining as training data for the Transformer model. The model is trained from scratch using standard cross-entropy loss for 60 epochs with the AdamW optimizer, learning rate 5×10^{-5}, weight decay 0.001, and batch size 2048. We select the checkpoint with the lowest validation loss. We then sample 1 million new formulas from this trained model, evaluate their distances, extract a new elite subset, and repeat the process. Crucially, the Transformer is retrained from scratch at each iteration to maintain exploration throughout the search.

Skip-Pairing Experiments. We ran the deep CEM pipeline and consistently obtained successful runs with 7 of the 20 hyperparameter configurations defining the initial population. Remarkably, the runs always converged to the leap statistic in 4 of these configurations, each time discovering syntactically different formulas that represent the same function. In the remaining 3 successful configurations, the

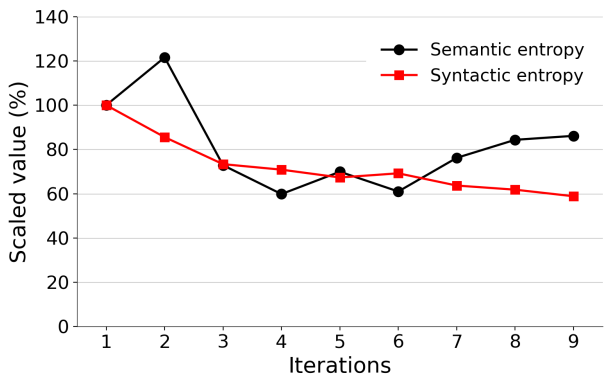


Figure 5. Semantic and syntactic entropy of a representative run. Values are scaled with respect to the first iteration.

following new statistic is discovered:

$$\begin{aligned} \mathrm{skew}(ab \mid cd) &= c - 2 + \begin{cases} a & \text{if } b < c \\ 0 & \text{otherwise} \end{cases} \\ \mathrm{skew}(abd) &= b - 2. \end{aligned} \tag{6}$$

Training Dynamics. To understand the training dynamics of the deep CEM approach, we monitor population diversity across iterations through the following metrics (Figure 5): the entropy of the distribution of unique functions computed by the generated formulas (*semantic*), and the token-level entropy of the model (*syntactic*). Semantic entropy remains relatively stable, indicating that the population does not degenerate into a set of functionally identical clones. In contrast, syntactic entropy exhibits a steady, monotonic decline. This suggests that the model learns to exploit the structural properties of successful formulas, gradually focusing the search space while retaining exploratory variance. This is unlike standard RL on LMs, which suffers from entropy collapse (Cui et al., 2025).

Formula Notation and Stability. The symbolic representation of formulas significantly impacts training stability. We compared three notations of varying compactness: reverse polish notation (most compact, no parentheses), standard infix notation, and infix with explicit whitespace. We find that compactness inversely correlates with stability: RPN runs degraded rapidly, producing illegible formulas after few iterations, while whitespace-enriched notation gave the best results. We hypothesize that explicit structural delimiters (parentheses and spaces) provide cues that help the Transformer preserve coherent formula structure during generation. Without these, the model tends to generate syntactically valid but redundant expressions, such as

$$(((c > i) \,\%\, i \,\%\, 23) \wedge ((b > i) \wedge 2 \wedge 0 \,\%\, 30 + 10 \vee 12)) \,\%\, 3.$$

Wasserstein: L^1 vs Trivial Metric. We investigated whether replacing the distance δ with the sorted L^1 distance would improve convergence. From an optimal transport perspective, these two objectives represent the Wasserstein distance equipped with different ground metrics: the trivial metric, where the cost is 0 for identical points and 1 otherwise, and the 1-dimensional L^1 metric. The trivial metric treats statistic discovery as a classification problem, viewing all output values as equidistant classes; conversely, the L^1 metric interprets the task as regression by rewarding numerical proximity. MapSeek-Symbolic failed to converge in all runs using the sorted L^1 distance, suggesting that the problem is better regarded as a classification task: the constraints demand exact symbolic matching, and the magnitude of numerical violation is not a useful learning signal.

6.3. Extending the Atlas

More combinatorial interpretations of q, t-Narayana polynomials $N_{n,3}(q, t)$ emerge by running MapSeek-Functional and MapSeek-Symbolic on statistics other than skip. Figure 1 shows the most notable statistics and connections between them found by the two methods.

Although both methods used a finite training dataset, we are able to mathematically verify most discovered pairings: for every edge in Figure 1, we prove the joint distribution matches $N_{n,3}(q, t)$ for all n. Proofs are formalized in Lean 4 and available as supplementary material (see Appendix B). This validates our claim that infinite SLURP problems can be effectively solved by applying ML to finite subproblems.

A large number of emerging connections involve the *area* statistic, which (like the *bounce* statistic) is the translation of a known statistic on polyominoes (see Appendix A). This is notable as the area is arguably the simplest known statistic on polyominoes related to q, t-Narayana polynomials, thus suggesting that both methods are biased toward human-designed statistics.

7. ML-Assisted Mathematical Contributions

In addition to discovering several interpretations of $N_{n,3}$ in Section 6, the machine-found statistics also lead to two human mathematical contributions. First, we prove an exchanging bijection for (skip, leap), hence giving the first combinatorial proof of the symmetry $N_{n,3}(q, t) = N_{n,3}(t, q)$. For $a < b \leq c < d$ the four elements in non-singleton blocks, with $b = c$ in the size-3 case, the map reflects b in $[a, c]$ and then reflects a, b, c in $[0, d]$ (Appendix A, Figure 10). In a formula:

$$(a,\ b,\ c,\ d) \mapsto (d - c,\ b + d - a - c,\ d - a,\ d).$$

Moreover, by extending the flipped version of *skew* (its composition with the flip map $i \mapsto n + 1 - i$), we obtain a new

statistic, called *mingarc* (min gap arc), which pairs with skip to give the first combinatorial interpretation of $N_{n,k}(q,t)$ using noncrossing partitions for all k (see Appendix A.4).

We formalize that the above bijection exchanges skip and leap; combined with the Lean proof that (skip, leap) is distributed according to $N_{n,3}(q,t)$ for all n, this gives a fully verifiable solution to Open Problem 4.1 for $k = 3$. We also formalize the skip-mingarc pairing, providing the first combinatorial interpretation of $N_{n,k}(q,t)$ based on noncrossing partitions for all k.

8. Conclusions and Future Directions

This paper shows how ML can be effectively applied to mathematical discovery problems where rigid constraints must be exactly satisfied and simplicity is a key requirement.

Several directions appear immediately. The main mathematical challenge lies in more difficult q,t-combinatorics problems, such as discovering new $N_{n,k}(q,t)$ interpretations for $k \geq 4$. While the statistics and exchanging bijection we obtain for $k = 3$ can be written in closed forms, we expect many natural solutions for larger k to be inherently algorithmic (e.g, the bounce statistic, defined in Appendix A).

On the methodological side, our current $k = 4$ experiments highlight concrete failure modes that must be resolved: MapSeek-Functional converges to solutions seemingly difficult to interpret; MapSeek-Symbolic is unable to reach zero distance from exact constraint match.

More broadly, it remains to understand how SLURP methods explore the space of simple solutions, and whether all solutions a human mathematician would consider meaningful can in fact be found automatically. It would also be beneficial to quantify simplicity and generalization through principled metrics for SLURP problems.

Acknowledgements

The authors are grateful to the organizers of several workshops and conferences where fruitful discussions on this work took place, including *Algebraic Combinatorics in Ancona* (2025), *q,t-Combinatorics in Algebra, Geometry and Topology* (2025), *ItaLean 2025*, and the two *INTRICATO* meetings in Pisa (2025) and Rome (2026). This work was supported by the MUR grant PRIN 2022A7L229 "Algebraic and topological combinatorics" CUP-J53D23003660006. The authors are members of the research group INdAM-GNSAGA.

Impact Statement

This work develops machine learning methods to discover simple, verifiable mathematical functions under exact distributional constraints, with applications to algebraic combinatorics. We do not anticipate direct negative societal impact; the main broader effect is to strengthen machine-assisted mathematical discovery.

References

Achim, T., Best, A., Bietti, A., Der, K., Fédérico, M., Gukov, S., Halpern-Leistner, D., Henningsgard, K., Kudryashov, Y., Meiburg, A., Michelsen, M., Patterson, R., Rodriguez, E., Scharff, L., Shanker, V., Sicca, V., Sowrirajan, H., Swope, A., Tamas, M., Tenev, V., Thomm, J., Williams, H., and Wu, L. Aristotle: IMO-level automated theorem proving. *CoRR*, abs/2510.01346, 2025. doi: 10.48550/ARXIV.2510.01346. URL https://doi.org/10.48550/arXiv.2510.01346.

Alfarano, A., Charton, F., and Hayat, A. Global lyapunov functions: a long-standing open problem in mathematics, with symbolic transformers. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL https://doi.org/10.52202/079017-2969.

Aval, J.-C., D'Adderio, M., Dukes, M., Hicks, A., and Le Borgne, Y. Statistics on parallelogram polyominoes and a q,t-analogue of the Narayana numbers. *J. Combin. Theory Ser. A*, 123:271–286, 2014. ISSN 0097-3165. URL https://doi.org/10.1016/j.jcta.2013.09.001.

Brown, D., He, J., Jenne, H., Kvinge, H., and Vargas, M. Even with AI, bijection discovery is still hard: The opportunities and challenges of openevolve for novel bijection construction. *CoRR*, abs/2511.20987, 2025. doi: 10.48550/ARXIV.2511.20987. URL https://doi.org/10.48550/arXiv.2511.20987.

Busa-Fekete, R., Choi, H., Dick, T., Gentile, C., and Muñoz Medina, A. Easy learning from label proportions. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Chaudhuri, S., Ellis, K., Polozov, O., Singh, R., Solar-Lezama, A., and Yue, Y. Neurosymbolic programming. *Found. Trends Program. Lang.*, 7(3):158–243, 2021. doi: 10.1561/2500000049. URL https://doi.org/10.1561/2500000049.

Chen, L., Gu, J., Huang, L., Huang, W., Jiang, Z., Jie, A., Jin, X., Jin, X., Li, C., Ma, K., Ren, C., Shen, J.,

Shi, W., Sun, T., Sun, H., Wang, J., Wang, S., Wang, Z., Wei, C., Wei, S., Wu, Y., Wu, Y., Xia, Y., Xin, H., Yang, F., Ying, H., Yuan, H., Yuan, Z., Zhan, T., Zhang, C., Zhang, Y., Zhang, G., Zhao, T., Zhao, J., Zhou, Y., and Zhu, T. H. Seed-prover: Deep and broad reasoning for automated theorem proving. *CoRR*, abs/2507.23726, 2025. doi: 10.48550/ARXIV.2507.23726. URL https://doi.org/10.48550/arXiv.2507.23726.

Cui, G., Zhang, Y., Chen, J., Yuan, L., Wang, Z., Zuo, Y., Li, H., Fan, Y., Chen, H., Chen, W., Liu, Z., Peng, H., Bai, L., Ouyang, W., Cheng, Y., Zhou, B., and Ding, N. The entropy mechanism of reinforcement learning for reasoning language models. *CoRR*, abs/2505.22617, 2025. doi: 10.48550/ARXIV.2505.22617. URL https://doi.org/10.48550/arXiv.2505.22617.

D'Adderio, M., Iraci, A., and Vanden Wyngaerd, A. Theta operators, refined delta conjectures, and coinvariants. *Adv. Math.*, 376:60, 2021. ISSN 0001-8708. doi: 10.1016/j.aim.2020.107447. Id/No 107447.

D'Adderio, M., Iraci, A., and Vanden Wyngaerd, A. *Decorated Dyck paths, polyominoes, and the Delta conjecture*, volume 278 of *Memoirs of the American Mathematical Society*. American Mathematical Society, July 2022. ISBN 9781470471705 9781470471576. doi: 10.1090/memo/1370. URL https://www.ams.org/memo/1370/. No. 1370.

Davies, A., Velickovic, P., Buesing, L., Blackwell, S., Zheng, D., Tomasev, N., Tanburn, R., Battaglia, P. W., Blundell, C., Juhász, A., Lackenby, M., Williamson, G., Hassabis, D., and Kohli, P. Advancing mathematics by guiding human intuition with AI. *Nat.*, 600(7887):70–74, 2021. doi: 10.1038/S41586-021-04086-X. URL https://doi.org/10.1038/s41586-021-04086-x.

Garsia, A. M. and Haglund, J. A proof of the q, t-Catalan positivity conjecture. *Discrete Math*, 256(3):677–717, 2002. ISSN 0012-365X. URL https://doi.org/10.1016/S0012-365X(02)00343-6. LaCIM 2000 Conference on Combinatorics, Computer Science and Applications (Montreal, QC).

Gorsky, E., Hawkes, G., Schilling, A., and Rainbolt, J. Generalized q, t-Catalan numbers. *Algebraic Combinatorics*, 3(4):855–886, 2020.

Gulwani, S., Polozov, O., and Singh, R. Program synthesis. *Found. Trends Program. Lang.*, 4(1-2):1–119, 2017. doi: 10.1561/2500000010. URL https://doi.org/10.1561/2500000010.

Haglund, J. *The q, t-Catalan numbers and the space of diagonal harmonics. With an appendix on the combinatorics of Macdonald polynomials*, volume 41 of *Univ. Lect. Ser.* Providence, RI: American Mathematical Society (AMS), 2008. ISBN 978-0-8218-4411-3.

Haiman, M. Hilbert schemes, polygraphs and the Macdonald positivity conjecture. *J. Amer. Math. Soc.*, 14(4):941–1006, 2001. ISSN 0894-0347. URL https://doi.org/10.1090/S0894-0347-01-00373-3.

Havaldar, S., Sharma, N., Sareen, S., Shanmugam, K., and Raghuveer, A. Learning from label proportions: Bootstrapping supervised learners via belief propagation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=KQe9tHd0k8.

Huang, X., Jackson, B., and Lee, K. From black box to bijection: Interpreting machine learning to build a zeta map algorithm. *CoRR*, abs/2511.12421, 2025. doi: 10.48550/ARXIV.2511.12421. URL https://doi.org/10.48550/arXiv.2511.12421.

Macdonald, I. G. A new class of symmetric functions. *Sémin. Lothar. Comb.*, 20:b20a, 41, 1988. ISSN 1286-4889. URL https://eudml.org/doc/120965.

mathlib Community, T. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, pp. 367–381, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370974. doi: 10.1145/3372885.3373824. URL https://doi.org/10.1145/3372885.3373824.

Moura, L. d. and Ullrich, S. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings*, pp. 625–635, Berlin, Heidelberg, 2021. Springer-Verlag. doi: 10.1007/978-3-030-79876-5_37. URL https://doi.org/10.1007/978-3-030-79876-5_37.

Poli, R., Langdon, W. B., and McPhee, N. F. *A Field Guide to Genetic Programming*. lulu.com, 2008. ISBN 978-1-4092-0073-4. URL http://www.gp-field-guide.org.uk/.

Quadrianto, N., Smola, A. J., Caetano, T. S., and Le, Q. V. Estimating labels from label proportions. *J. Mach. Learn. Res.*, 10:2349–2374, 2009. doi: 10.5555/1577069.1755865. URL https://dl.acm.org/doi/10.5555/1577069.1755865.

Remmel, J. B. and Wilson, A. T. An extension of MacMahon's equidistribution theorem to ordered set partitions. *J. Combin. Theory Ser. A*, 134:242–277, 2015. ISSN 0097-3165. doi: 10.1016/j.jcta.2015.03.012. URL https://doi.org/10.1016/j.jcta.2015.03.012.

Scott, C. and Zhang, J. Learning from label proportions: A mutual contamination framework. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Silva, S. and Costa, E. Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genet. Program. Evolvable Mach.*, 10(2):141–179, 2009. doi: 10.1007/S10710-008-9075-9. URL https://doi.org/10.1007/s10710-008-9075-9.

Sun, J. J., Tjandrasuwita, M., Sehgal, A., Solar-Lezama, A., Chaudhuri, S., Yue, Y., and Costilla-Reyes, O. Neurosymbolic programming for science. *CoRR*, abs/2210.05050, 2022. doi: 10.48550/ARXIV.2210.05050. URL https://doi.org/10.48550/arXiv.2210.05050.

Wagner, A. Z. Constructions in combinatorics via neural networks. *CoRR*, abs/2104.14516, 2021. URL https://arxiv.org/abs/2104.14516.

Williamson, G. Is deep learning a useful tool for the pure mathematician? *CoRR*, abs/2304.12602, 2023. doi: 10.48550/ARXIV.2304.12602. URL https://doi.org/10.48550/arXiv.2304.12602.

A. q, t-Combinatorics Background & Narayana Models

Triggered by the discovery of the famous Macdonald polynomials (Macdonald, 1988), q, t-combinatorics studies families of bivariate polynomials $F(q, t)$ with nonnegative integer coefficients arising in a variety of mathematical domains.

Historically, the first examples that attracted an intense research activity occurred in the study of the diagonal coinvariants of the symmetric group (Haiman, 2001). The bivariate polynomials arising from these algebraic objects keep track of the multiplicities of their "building blocks" (irreducible representations): computing these multiplicities is the prototypical problem of representation theory. The q, t-Narayana polynomials $N_{n,k}(q, t)$ are a notable example.

Interest in q, t-combinatorics flourished over the past 25 years. Consequently, a formidable number of new formulas, conjectures, open problems, and interactions with other disciplines have been discovered. Among these new problems, a prominent role is played by the Type 1 and Type 2 problems mentioned in Section 4 (Haglund, 2008; D'Adderio et al., 2021). Our hope is that the methods presented in this paper will lead to breakthroughs in this fascinating area.

The Type 2 problem of proving combinatorially the q, t-symmetry $N_{n,k}(q, t) = N_{n,k}(t, q)$ is one of the outstanding open problems of the theory. A combinatorial interpretation of $N_{n,k}(q, t)$ in terms of *polyominoes*, together with an algebraic proof of its q, t-symmetry, first appeared in (Aval et al., 2014). The relation between q, t-Narayana, polyominoes, and noncrossing partitions is described in the next sections.

A.1. Polyominoes

A *parallelogram polyomino* of size $m \times n$ is a pair of lattice paths from $(0, 0)$ to (m, n), using north and east steps, that meet only in the extremal points (Figure 6). The set of all polyominoes of size $m \times n$ is denoted by $\mathsf{PP}(m, n)$.

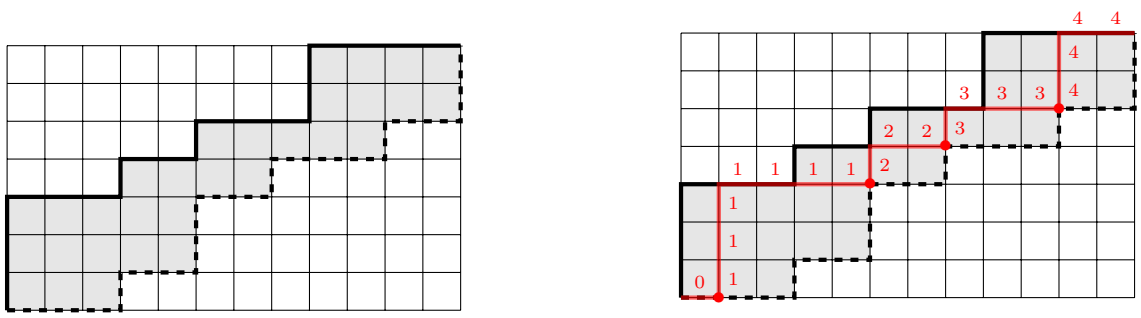


Figure 6. A 12×7 polyomino with area $30 - 18 = 12$ (left) and its bounce path (right); the bounce statistic is equal to $41 - 18 = 23$.

Definition A.1. The *area* statistic on $\mathsf{PP}(m, n)$ is the number of whole squares between the two paths, normalized by subtracting $m + n - 1$ (so that the minimum possible area is 0).

Parallelogram polyominoes have a *bounce path*. To compute it, start from $(0, 0)$, draw a single east step, and then follow this algorithm: draw north steps until the path hits the polyomino's ceiling; then draw east steps until the path hits the polyomino's floor; repeat until it reaches (m, n), as shown in Figure 6.

Definition A.2. The *bounce* statistic on $\mathsf{PP}(m, n)$ is defined as the sum, over all steps of the bounce path, of the number of east-north turns preceding the step; normalize by subtracting $m + n - 1$ (so that the minimum possible bounce is 0).

A third statistic called *dinv* was also introduced in (Aval et al., 2014). Both pairs (area, bounce) and (dinv, area) were proven to yield combinatorial interpretations of the q, t-Narayana polynomials:

$$N_{n,k}(q, t) = \sum_{P \in \mathsf{PP}(n-k+1,k)} q^{\text{area}(P)} t^{\text{bounce}(P)} = \sum_{P \in \mathsf{PP}(n-k+1,k)} q^{\text{dinv}(P)} t^{\text{area}(P)}.$$

In addition, the symmetry $N_{n,k}(q, t) = N_{n,k}(t, q)$ was proved by algebraic methods; a combinatorial proof is still unknown.

The refinement introduced in Section 5 is justified by the following monotonicity property of q, t-Narayana polynomials.

Proposition A.3. *For every fixed $k \geq 1$, each coefficient of $N_{m,k}(q, t)$ is nondecreasing in m. Equivalently, the incremental polynomial*

$$N_{m,k}(q, t) - N_{m-1,k}(q, t)$$

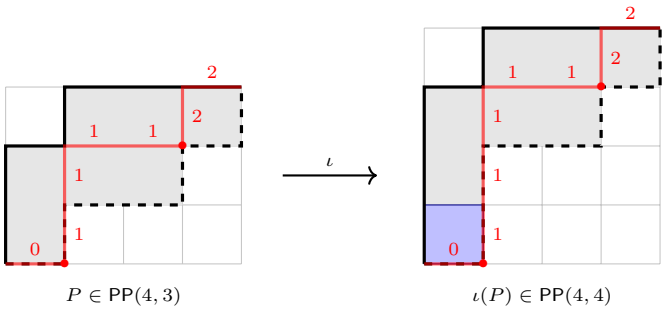


Figure 7. The embedding used in Proposition A.3. A new bottom row is added, but only its leftmost cell is kept, so there is exactly one more square between the two paths. The red bounce path acquires one extra step in its first vertical run.

has nonnegative coefficients.

Proof. It is convenient to draw the model so that the rectangle side depending on m is vertical; this is justified by the symmetry of the q,t-Narayana polynomials in the two rectangle directions (Aval et al., 2014, Theorem 6.2). In this picture, increasing m by 1 amounts to adding one new bottom row. Given a polyomino P, define $\iota(P)$ by adding such a bottom row and keeping only its leftmost cell, as in Figure 7. Equivalently, if the upper path of P is U and the lower path is EL, then the upper and lower paths of $\iota(P)$ are NU and ENL, respectively. This construction is clearly injective.

The unnormalized area increases by exactly 1, because $\iota(P)$ contains exactly one extra cell. But the normalization also increases by 1, since we now subtract one more unit from the enclosing rectangle size. Therefore $\text{area}(\iota(P)) = \text{area}(P)$.

The bounce path of $\iota(P)$ is obtained from the bounce path of P by inserting one extra step in its first vertical run. Hence the unnormalized bounce also increases by 1. Again, this is exactly canceled by the larger normalization term, so $\text{bounce}(\iota(P)) = \text{bounce}(P)$.

Thus, for every pair (a,b), the number of polyominoes with $(\text{area}, \text{bounce}) = (a,b)$ cannot decrease when m increases by 1. In other words, every coefficient of $N_{m,k}(q,t)$ is nondecreasing in m, and the difference $N_{m,k}(q,t) - N_{m-1,k}(q,t)$ has nonnegative coefficients. □

A.2. From Polyominoes to Noncrossing Partitions

Noncrossing partitions $\mathsf{NC}(n,k)$ and parallelogram polyominoes $\mathsf{PP}(n-k+1,k)$ are both counted by the Narayana numbers. A bridge between these two sets of combinatorial objects is given by the following bijection (which, to the best of our knowledge, is not present in the literature).

Definition A.4. Let $\eta\colon \mathsf{PP}(n-k+1,k) \to \mathsf{NC}(n,k)$ be defined as follows. Label from 1 to n the cells adjacent to the floor of a polyomino, going bottom to top, left to right (call these the *indexing labels*). Then label the cells adjacent to the ceiling, again bottom to top, left to right, always writing the largest unused indexing label which is present in the same row or below. The entries in each row are the blocks of the target noncrossing partition (Figure 8).

We omit the proof that η is a bijection. This bijection can be used to translate statistics from polyominoes to noncrossing partitions and vice versa; this is how we defined area and bounce on noncrossing partitions (appearing in Figure 1). For the area, we have the following explicit formula on $\mathsf{NC}(n,k)$ for all $n \geq k \geq 1$:

$$\text{area}(\pi) = \sum_{\beta,\beta' \in \pi} \#\{i \in \beta' \mid i < \max\beta < \max\beta'\}.$$

This is closely related to the known *inv* statistic on ordered set partitions (Remmel & Wilson, 2015).

The skip statistic from (2) can also be read directly from the drawing of a noncrossing partition. If $\beta = \{b_1 < \cdots < b_r\}$ is a block, draw the edges joining consecutive elements in increasing order, namely $b_1b_2, b_2b_3, \ldots, b_{r-1}b_r$, and ignore only the closing edge b_rb_1. Each counted edge b_ib_{i+1} contributes the number of labels strictly between its endpoints, namely

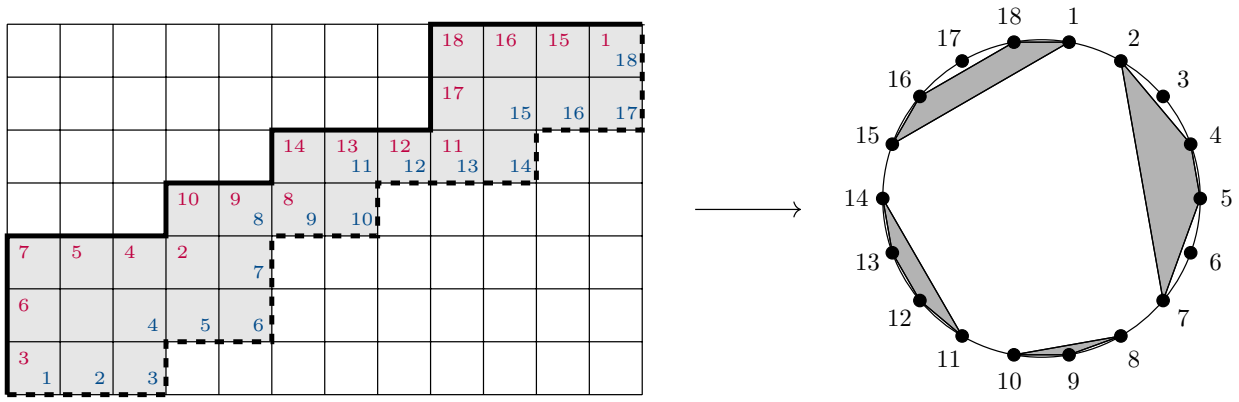
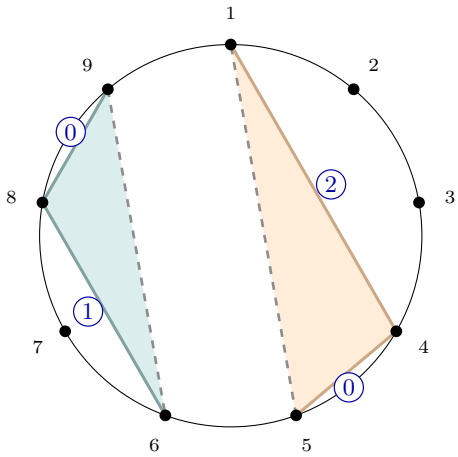


Figure 8. An example of the bijection η.

$b_{i+1} - b_i - 1$. Summing these contributions over all blocks recovers the skip statistic, because

$$\sum_{i=1}^{r-1}(b_{i+1} - b_i - 1) = b_r - b_1 - r + 1 = \max\beta - \min\beta - |\beta| + 1.$$

An example is shown in Figure 9.



$$\pi = \{1,4,5\} \mid \{2\} \mid \{3\} \mid \{6,8,9\} \mid \{7\}$$

$$\mathrm{skip}(\pi) = (2+0) + (1+0) = 3$$

$$\{1,4,5\} : (4-1-1) + (5-4-1),$$
$$\{6,8,9\} : (8-6-1) + (9-8-1).$$

Blue labels mark the counted edges; the gray dashed edges are ignored.

Figure 9. Computing the skip statistic from a noncrossing partition. The contribution of each counted edge is the number of labels strictly between its endpoints.

We derived an explicit formula for the translation of the bounce only for $k = 3$. The bounce–area connections in Figure 1 are obtained by running our ML methods on the *flipped bounce*, i.e., the bounce of the noncrossing partition reflected in the segment $[1, n]$. This flip is convenient in the refined setting (introduced in Section 5): unlike the usual bounce, the flipped bounce is compatible with the filtration by $m(\pi)$ in the sense that it is unchanged when appending singleton blocks at the end. Working with the flipped bounce is essentially equivalent, since there is a simple bijection on $\mathsf{NC}(n, 3)$ that fixes the area and exchanges bounce with flipped bounce; we formalize this bijection in Lean (see supplementary code).

A.3. Exchanging Bijection for $N_{n,3}(q,t)$

The bijection on $\mathsf{NC}(n, 3)$ introduced in Section 7 exchanges the skip and leap statistics, as exemplified in Figure 10 for $n = 4$. This constitutes the first combinatorial proof of the symmetry $N_{n,3}(q,t) = N_{n,3}(t,q)$ for all $n \geq 3$, as verified through formal Lean proofs (Appendix B). Our bijection preserves the block sizes and permutes the gaps between numbers in non-singleton blocks.

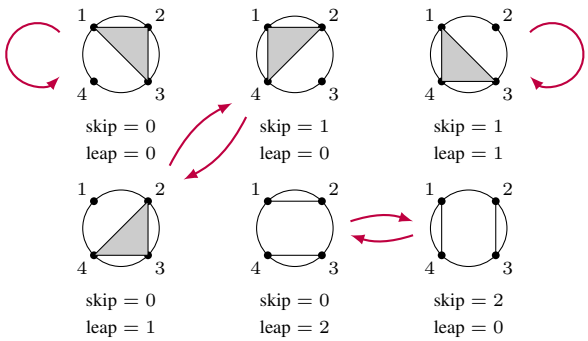


Figure 10. Exchanging bijection for $n = 4$ and $k = 3$.

A.4. The Min Gap Arc Statistic

The *min gap arc* statistic (mingarc) is a human-defined extension of the flipped skew statistic discovered by our ML methods on $\mathsf{NC}(n, 3)$. It is defined on $\mathsf{NC}(n, k)$ for arbitrary k through the arc-selection procedure below and agrees with flipped skew when $k = 3$. Together with skip, it gives a combinatorial interpretation of $N_{n,k}(q, t)$.

Candidate arcs. Let $\pi \in \mathsf{NC}(n, k)$ with $h = n - k + 1$ blocks. Order the block minima and maxima as

$$1 = m_1 < \cdots < m_h, \qquad c_1 < \cdots < c_h = n,$$

and set $m_{h+1} := n + 1$. For each block r let $\nu_r = m_{r+1} - m_r - 1$. List the elements of $[n]$ that are *not* block maxima in increasing order as $x_1 < \cdots < x_{k-1}$ (the *left endpoints*), and form the *right endpoint* list by repeating c_r exactly ν_r times for each r in sequence, yielding a weakly increasing list $y_1 \leq \cdots \leq y_{k-1}$. The *candidate arcs* are the intervals $[x_i, y_i]$.

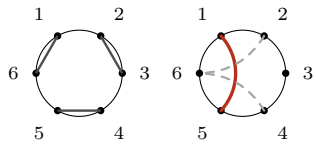
Greedy choice. Scan the candidate arcs left to right. Keep the first arc; thereafter always keep the earliest arc whose left endpoint is greater than the right endpoint of the last kept arc. The *selected arcs* are the ones kept by this greedy procedure, and $\mathcal{I} \subseteq \{1, \ldots, k-1\}$ is the set of their indices.

Statistic. We define mingarc by summing the gaps from the right endpoints of the selected arcs to n:

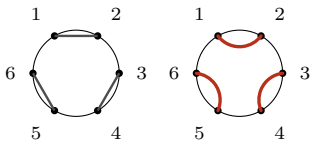
$$\text{mingarc}(\pi) = \sum_{i \in \mathcal{I}} (n - y_i).$$

The (skip, mingarc) pairing yields the q, t-Narayana polynomial:

$$N_{n,k}(q, t) = \sum_{\pi \in \mathsf{NC}(n,k)} q^{\text{skip}(\pi)}\, t^{\text{mingarc}(\pi)}.$$



(a) $\pi = 16 \mid 23 \mid 45$. The selected arc is 1–5; the candidate arcs 2–6 and 4–6 are not selected.



(b) $\pi = 12 \mid 34 \mid 56$. The candidate arcs are disjoint, so all of them are selected.

Figure 11. Min gap arc construction on two noncrossing partitions. Solid red arcs are selected; dashed gray arcs are candidate arcs that are not selected.

The case $k = 3$. There are exactly two candidate arcs. They overlap when the partition contains a triple block or a nested pair; in the non-nested case they are disjoint and both are kept. The statistic therefore evaluates to

$$\text{mingarc}(abc) = n - b - 1, \qquad \text{mingarc}(ab \mid cd) = 2n - b - d \quad (a < b < c < d),$$

and in the nested case

$$\text{mingarc}(ab \mid cd) = n - d - 1 \qquad (a < c < d < b).$$

These are exactly the values of (6) composed with the flip $i \mapsto n + 1 - i$. Hence, on $\mathsf{NC}(n, 3)$, the statistic mingarc is precisely the flipped skew.

B. Formal Lean Proofs

Lean 4 is an interactive theorem prover (proof assistant) and functional programming language for writing machine-checked mathematics. In Lean, mathematical statements are encoded as types and proofs as terms inhabiting those types; correctness is verified by a small trusted kernel via type checking. During development, Lean also supports the keyword `sorry`, which acts as an explicit placeholder for a missing proof term: it allows the file to type-check while recording a proof gap. We refer to (Moura & Ullrich, 2021) for a detailed description of the system. Our formalization builds on Mathlib, the community-maintained library of definitions and theorems for Lean (mathlib Community, 2020).

To accelerate the formalization effort, we made extensive use of *Aristotle*, Harmonic's Lean formalization system (Achim et al., 2025), which closes proof gaps and autoformalizes informal arguments.

This appendix gives a brief guide to the Lean 4 code accompanying this paper. Our goal in using Lean is *verification*: the definitions and proofs that certify the main combinatorial claims are checked by Lean's kernel, making the verification independently reproducible.

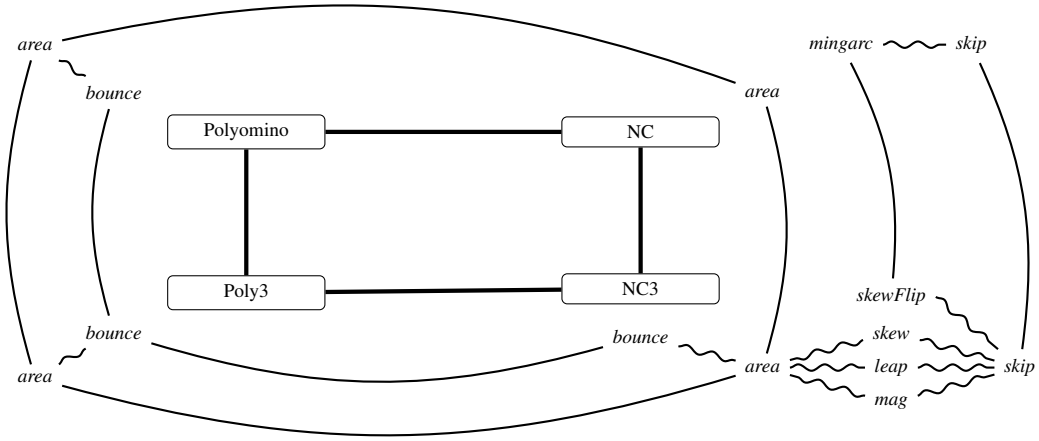


Figure 12. Overview of the objects and statistics covered by the Lean code. Thick edges indicate equivalences of types, and are intended with specific m, n, and k values. Curved thin edges indicate transport of statistics across these maps; squiggly edges indicate pairing relationships among statistics. All edges shown correspond to statements whose proofs are formalized in the current Lean development.

At a high level, the development follows the structure summarized in Figure 12. Two of the boxes (Polyomino and NC) correspond to the main mathematical objects appearing in this paper: parallelogram polyominoes and noncrossing partitions. Both have been formalized in Lean in full generality, for arbitrary parameters m, n, and k. Since our main results concern the case $k = 3$ (equivalently, $m = 3$), we also introduce two specialized definitions (Poly3 and NC3) for those instances. In these specialized representations, objects can be described by three or four natural numbers and are easier to manipulate algebraically. To ensure correctness, we connect the specialized encodings to the general definitions via explicit bijections. The Lean code formalizes (i) maps between these objects and representations and (ii) the way statistics are translated across them. It also formalizes a number of statistic pairings, including the "atlas" connections on NC3 (Figure 1) and the all-k `skip_mingarc` pairing on `NC`.

B.1. What Is Verified

We formalize all the mathematical results found in the paper, as well as the necessary mathematical objects needed to state them and the basic combinatorial theory around them. For a reader-oriented entry point, we recommend opening `qtLearning.lean`, which is a curated index of the main definitions and theorems (in VS Code, the imported filenames

and `#check` declarations are clickable once the file compiles). For instructions on building the project and running the artifact locally, see the file `README.md`.

- **Representations and maps.** Definitions of the main types (`Polyomino` and `NC`) and their specializations (`Poly3` and `NC3`), together with the key connecting maps (`Phi`/`Psi`, `NC.equivNC3`, `Poly3.equivNC3`, `Poly3.equivPolyomino`). The forward/backward maps are implemented throughout, and the stated equivalences are fully proved, including the general Polyomino ↔ NC equivalence `Poly_NC_equiv` built from `Phi` and `Psi`.
- **Statistics.** Definitions of the statistics used in the paper (`area`, `bounce`, `bounceFlip`, `skip`, `leap`, `skew`, `skewFlip`, `mag`, `mingarc`) and lemmas expressing compatibility under the representation maps (e.g., `NC3.toNC_skip`, `NC.toNC3_skip`, `NC3.toNC_area`). All compatibility lemmas needed for the $k = 3$ results are proved, and the full Polyomino-to-NC transport of `area` through `Phi`/`Psi` is proved as `Polyomino.Phi_area` and `NC.Psi_area`.
- **Pairings.** We formalize Narayana pairings by expressing that two pairs of statistics have the same joint distribution as $(\text{area}, \text{bounce})$ on `Polyomino`. The atlas pairings among `area`, `skip`, `leap`, `skew`, `mag`, and `skewFlip` are fully verified on `NC3`; the all-k `skip`–`mingarc` pairing on `NC` is proved as `NC.skip_mingarc_pair`. The pairings on `Polyomino` and on `Poly3` are also formalized. It is noteworthy that the proofs of the core $k = 3$ pairing theorems (e.g., `NC3.skip_leap_pair` and related atlas pairings) were found and formalized end-to-end by Aristotle, without human intervention.
- **Exchanging bijections and q, t-symmetry.** We formalize explicit bijections on `NC3` that exchange pairs of statistics (e.g., `skip`↔`leap`). In particular, the Lean development includes a fully formal, kernel-checked bijective proof of the symmetry $N_{n,3}(q, t) = N_{n,3}(t, q)$ for all $n \geq 3$, which is a special case of Open Problem 4.1.
- **Narayana polynomials.** A lightweight interface for q, t-generating functions of statistics, including `qtPoly` and `EquallyDistributed`$_2$. We also define the q, t-Narayana polynomial `Narayana.qtPoly` as the $(\text{area}, \text{bounce})$ generating function on `Polyomino`. In the Lean code, this polynomial is parameterized using the polyomino convention (m, n) rather than the noncrossing convention (n, k). We intentionally implement this interface as functions $\mathbb{N} \to \mathbb{N} \to \mathbb{N}$ (rather than using Mathlib's polynomial types), since we only need coefficient-level identities coming from finite enumerations. The formalized interface includes the $m = 3$ symmetry theorem `Narayana.qtPoly_3_symm` and the coefficientwise monotonicity theorem `Narayana.incrementalNarayanaPositive` used in Proposition A.3.

Finally, we include some test files, where we define specific instances of polyominoes and noncrossing partitions and check their statistics. This is possible because most of our implementation (including the statistics) is computable, so we can run the calculations in concrete cases. We also include tests comparing the formally computed `Narayana.qtPoly` against generator-produced ground truth up to $n = 14$, along with formal proofs that the two sources agree.

B.2. Formalization Workflow

While Lean provides the final verification guarantee (kernel-checked definitions and proofs), the development process was highly iterative. We made extensive use of Aristotle in two complementary modes:

- **Sorry-filling mode.** During development, some files temporarily contained `sorry` placeholders. Aristotle was used to propose proof terms and tactic scripts to solve these goals. Candidate proofs were then refined by hand when needed, mostly for performance and readability reasons. The sorry-filling mode was mainly used without informal guidance, relying on Aristotle's proof search capabilities. In particular, in the `NC3` specialization, Aristotle was able to autonomously prove the core pairing theorems end-to-end.
- **Autoformalization mode.** This mode takes both a *formal* input (the Lean file being developed, including its imports and local context) and an *informal* input (a text or LaTeX document containing an informal description of the mathematics or a proof sketch), and translates informal mathematics to Lean. As with sorry-filling, the resulting artifacts were iterated on when needed to align definitions, make implicit conditions explicit, and fit proofs to the existing library interface.

In practice, this created a tight loop between (i) writing or refining the informal arguments, (ii) re-running automation to update the Lean files, (iii) consolidating and polishing the resulting proof scripts, and, when some solvable `sorry`s were

identified, (iv) launching a sorry-filling run to eliminate them. The end product is still fully checked by Lean, so the use of automation only affects developer productivity, and not the trust model.

B.3. How to Build the Artifacts

To check the implementation locally, follow the instructions in `README.md`.

The Lean project is pinned to Lean toolchain version `v4.28.0` (see the `lean-toolchain` file in the Lean project directory).

To build it, first change into the directory containing `lakefile.toml`: this is the repository root in a full development checkout, and the `Lean/` directory in the supplementary artifact. Then run:

```
lake exe cache get!
lake build
```

To check whether a specific theorem statement depends on admitted proofs, Lean provides an "axioms" report. In a Lean file, you can run `#print axioms <theoremName>` after the theorem (e.g., on a pairing theorem); if the output mentions `sorryAx`, then the theorem depends (directly or indirectly) on a `sorry`. It is normal for fully proved results in Mathlib-based developments to depend on a small set of standard axioms such as `propext`, `Classical.choice`, and `Quot.sound`; some computational proofs can also report Lean's trusted reduction/compiler axioms. These do not indicate missing proofs.