Mapping Uncharted Symmetries: Machine Discovery in Combinatorics

Eugenio Cainelli[*]
University of Bologna
eugenio.cainelli2@unibo.it

Lorenzo Luccioli
University of Bologna
lorenzo.luccioli2@unibo.it

Alessandro Iraci
Pegaso University
alessandro.iraci@unipegaso.it

Michele D'Adderio
University of Pisa
michele.dadderio@unipi.it

Giovanni Paolini[*]
University of Bologna
g.paolini@unibo.it

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 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.

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 1). 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.

We introduce two machine learning methods for SLURP: MapSeek-Functional and MapSeek-Symbolic. To demonstrate their effectiveness and the soundness of the SLURP setup, we describe a

[*]Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

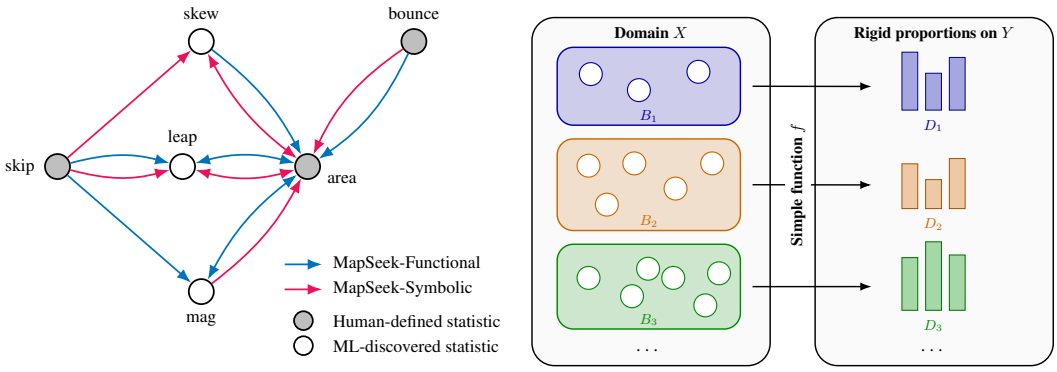


Figure 1: Left: atlas of notable statistics (nodes) for $k = 3$. An edge $f \to g$ indicates that, starting from statistic f, one of our methods discovers g as a partner statistic, so that (f, g) provides a combinatorial interpretation of Narayana polynomials $N_{n,3}(q, t)$. Right: a SLURP instance where the domain X is partitioned into bags, and supervision is provided via per-bag output distributions.

successful application to the combinatorial study of the q, t-Narayana polynomials $N_{n,k}(q, t)$ 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]

We regard the SLURP formulation, the expert-in-the-loop discovery workflow, and the resulting mathematics as our core contributions, rather than the learning algorithms themselves. Our broader goal is to build flexible tools that, in the hands of expert mathematicians, can leverage human intuition and extend their reach: ML proposes candidate statistics, which mathematicians then interpret, generalize, and use to prove theorems (Table 2).

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 [1, 10]. Separately, a growing line of

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

research employs ML not as a prover but as an exploratory tool for discovering mathematical objects, constructing counterexamples, and formulating conjectures. Examples include constructing extremal combinatorial objects [40, 33, 8, 5], studying algebraic invariants [15], computing enumerative invariants [21], finding formulas for mathematical constants [35], and synthesizing Lyapunov functions for dynamical systems [2]; see also [41]. 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 [23], a Transformer was trained to approximate the *zeta map* [19], recovering a new algorithmic interpretation of this known map by analyzing attention weights; in [6], 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 [31, 34, 22, 7, 26]. 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 learn a classifier with good generalization, 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) and symbolic regression (SR). The objective of recovering explicit symbolic formulas under symbolic constraints situates SLURP within the broader scope of PS [18, 39, 9]. 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. Methodologically, MapSeek-Symbolic is close to neural-guided program search [43, 4] and deep SR [29, 28], where a generative model proposes candidates. Nevertheless, SLURP differs from typical PS and SR 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 as follows: the domain X is partitioned into bags $B_1, B_2, \ldots$, and each B_i comes with a target distribution D_i on Y (Figure 1). The goal is to find $f\colon X \to Y$ such that:

(1) The outputs on 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. Let $X = \mathbb{N}$ and partition it into consecutive bags $B_m = \{10m, 10m+1, \ldots, 10m+9\}$, for $m \geq 0$. Suppose that $Y = \{0, 1\}$ and that, for every bag B_m, the target distribution requires

exactly five zeros and five ones: $D_m(0) = D_m(1) = 5$. 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. Two possible simple solutions are $f(n) = n \bmod 2$ and $g(n) = \lfloor (n \bmod 10)/5 \rfloor$. The task is to find an exact solution whose description is simple enough to expose structure.

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. Third, we do not fix a formal measure of simplicity: any formal criterion (e.g., expressibility in a chosen language) would likely exclude some meaningful functions. Our two methods address this in complementary ways: MapSeek-Symbolic restricts the search to a prescribed formula language and favors shorter expressions, whereas MapSeek-Functional searches more broadly and relies on the subsequent distillation of its output into a uniform rule.

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 fixed random validation subset of X that is not used for gradient updates. Note that the validation set still undergoes the constraint projection step, so the setup is transductive: validation elements are held out from training, but not from the constraints.

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, or as a form of constrained pseudo-labeling. The stopping criterion is strict: a training run is considered successful only if the rigid proportion constraints are perfectly satisfied. In particular, the model must also fit the constraints on the validation elements, on which it is never trained, 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, but it appears to filter out less interpretable runs.

After convergence, the model f_θ encodes a function $X \to Y$, given as a finite vector of predictions; in our experiments, we are sometimes able to distill this function into a simple symbolic expression. Distillation can be partly automated via LLMs or symbolic regression (Section 6.1).

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 is aggregated to bias the subsequent generation 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 [19, 13]. 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 $\text{stat}_1, \text{stat}_2 \colon X \to \mathbb{N}$ (called *statistics*) such that

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

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:

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

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 [16], 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 [3, 14]. They refine the classical Narayana numbers $N_{n,k}(1, 1)$, which count many families of objects, including *parallelogram polyominoes* and *noncrossing partitions*.

In [3], a combinatorial interpretation is given 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:

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

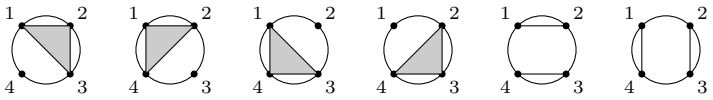


Figure 2: Noncrossing partitions $\mathsf{NC}(4,3)$. The corresponding q,t-Narayana is $q^2+qt+t^2+q+t+1$.

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 2), 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:

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

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 $u=\mathrm{skip}(\pi)$; within each group, the number of partitions π with $\mathrm{stat}(\pi)=v$ 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 $u=\mathrm{skip}(\pi)$. 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.

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 $i=0,\dots,n-1$ 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. M is used as an attention mask: in alternating layers, self-attention is restricted to entries with $M_{ij}=1$ (the final layer is unmasked). The attention logits from the last layer are 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 output, kept as a floating point number for the constraint projection step.

Training procedure. Within each family (same (m,u)), we sort all partitions, including validation ones, by the current model's output and assign the prescribed multiset of pseudo-labels by rank. 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. A run is successful when, after an epoch, the multiset of rounded model outputs equals the prescribed one in every family.

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 $n \geq 5$: $\sum_{\beta \in \pi} (\text{sum}(\beta) - \max(\beta)) - \frac{k(k-1)}{2}$. 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 deduce the following formula:

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

(the second formula being the limit case $b = c$ of the first). Interestingly, this resembles the t-statistic, which pairs with area on certain objects, yielding an interpretation of other q, t-polynomials [17].

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):

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

(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 3). 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.

Distilling a rule. We found the formulas for mag and leap with LLM assistance, discarding rules based on lookup tables or ad hoc cases. An LLM and PySR [11], run on all distinct successful outputs, find uniform rules valid for larger n only for leap, mag, and the first-cluster medoid for $h = 2$ (Appendix C). Thus MapSeek-Functional is best seen as a generator of candidate statistics, whose distillation relies on expert judgment.

Ablation study on validation set size. A peculiarity of MapSeek-Functional is that the size of the validation set is a hyperparameter that seems to trade off success rate and simplicity. Table 1 shows its effect in the $h = 2$ setup, pooling two independent batches of 50 runs (reported separately in Appendix C). As the validation split increases, the success rate drops. The number of runs matching mag exactly stays roughly constant, which is consistent with a larger split filtering out less interpretable solutions; given the small counts, we regard this as a qualitative observation.

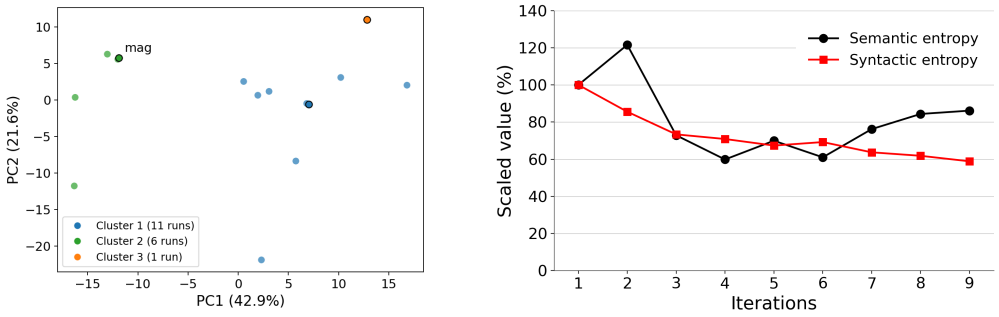


Figure 3: Left: PCA scatter plot of the 18 successful MapSeek-Functional runs of the original batch (refined $N_{14,3}$ dataset, $h = 2$, 5% validation). Cluster medoids are highlighted. Right: semantic and syntactic entropy of a representative run. Values are scaled with respect to the first iteration.

Table 1: Validation-set ablation in MapSeek-Functional ($h = 2$, 100 runs per setting), with 95% Wilson intervals [42].

Val	# Success	95% CI	# Exact mag
0%	98	[93.0%, 99.4%]	3
1%	82	[73.3%, 88.3%]	4
2%	61	[51.2%, 70.0%]	5
5%	28	[20.1%, 37.5%]	2
10%	2	[0.6%, 7.0%]	1

6.2 Statistic discovery with MapSeek-Symbolic

Formulas and evaluation. We take $\Sigma = \{a, b, c, d, i, 0, 1, 2, 3, +, -, \%, (,), \vee, \wedge, \neg, >, <, =\}$ as our alphabet. 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 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:

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

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 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 [37]. 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 [30]. Therefore, we moved to a controlled symbolic approach using the deep cross-entropy method (CEM) with a generative sequence model.

Architecture and training. We run the deep CEM using a Transformer decoder with 4 layers, 8 attention heads, hidden dimension 256, feedforward dimension 256, and a character-level tokenizer. 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 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 formulas from this trained model, evaluate their distances, extract a new elite subset, and repeat. 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 following new statistic is discovered:

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

Training dynamics. To understand the training dynamics of the deep CEM approach, we monitor population diversity across iterations through the following metrics (Figure 3): 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 [12].

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

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

Table 2: Machine and human contributions to the mathematical results of this paper.

Result	Machine contribution	Human contribution
mag	MapSeek-Functional (warm start, or $h = 2$ homogeneity)	Heuristic prior; distillation
leap	MapSeek-Functional ($h = 1$ homogeneity) and MapSeek-Symbolic	Distillation; naming
skew	MapSeek-Symbolic	Interpretation; naming
skip–leap bijection	leap as a key clue	Construction and proof
mingarc (all k)	skew as the starting point	Discovery of mingarc

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 in our public repository (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 (see Table 2). First, we prove an exchanging bijection for $(\text{skip}, \text{leap})$, hence giving the first combinatorial proof of the symmetry $N_{n,3}(q,t) = N_{n,3}(t,q)$. For $a < b \le c < d$ 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 8). We formalize that this bijection exchanges skip and leap; combined with the Lean proof that $(\text{skip}, \text{leap})$ is distributed according to $N_{n,3}(q,t)$ for all n, this gives a fully verifiable solution to Open Problem 4.1 for $k = 3$. 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 prove and formalize this pairing in Lean (Appendix B).

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.

Limitations. Our successful experiments concern a single family (q,t-Narayana polynomials with $k = 3$). Both pipelines rely on expert-designed representations, constraints, inductive biases, and formula languages: this is intentional, but it limits automation and transfer to other domains. Validation is transductive (Appendix C tests an inductive variant), and its effect is not separated from those of warm start and homogeneity. Distillation needs expert judgment, simplicity has no domain-independent measure, and some analyses rely on few runs. For $k = 4$, MapSeek-Symbolic never reaches zero distance and MapSeek-Functional yields outputs we could not distill (Appendix C).

Future directions. The main mathematical challenge lies in more difficult q,t-combinatorics problems, such as discovering new $N_{n,k}(q,t)$ interpretations for $k \ge 4$. 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. bounce, defined in Appendix A). Other open problems that can be phrased as SLURP instances include the area–bounce bijection for q,t-Catalan polynomials [19], type B q,t-Catalan combinatorics [38], and the Shareshian–Wachs conjecture [36]. Statistics similar to mag, such as the t-statistic of [17], suggest further applications. We have not yet attempted any of these. 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.

Acknowledgments and Disclosure of Funding

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.

References

[1] Tudor Achim, Alex Best, Alberto Bietti, Kevin Der, Mathïs Fédérico, Sergei Gukov, Daniel Halpern-Leistner, Kirsten Henningsgard, Yury Kudryashov, Alexander Meiburg, Martin Michelsen, Riley Patterson, Eric Rodriguez, Laura Scharff, Vikram Shanker, Vladmir Sicca, Hari Sowrirajan, Aidan Swope, Matyas Tamas, Vlad Tenev, Jonathan Thomm, Harold Williams, and Lawrence Wu. 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.

[2] Alberto Alfarano, François Charton, and Amaury Hayat. Global lyapunov functions: a long-standing open problem in mathematics, with symbolic transformers. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 37: 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.

[3] Jean-Christophe Aval, Michele D'Adderio, Mark Dukes, Angela Hicks, and Yvan Le Borgne. 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.

[4] Shraddha Barke, Emmanuel Anaya Gonzalez, Saketh Ram Kasibatla, Taylor Berg-Kirkpatrick, and Nadia Polikarpova. HYSYNTH: context-free LLM approximation for guiding program synthesis. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL https://doi.org/10.52202/079017-0499.

[5] Gergely Bérczi, Baran Hashemi, and Jonas Klüver. Flow-based extremal mathematical structure discovery. *CoRR*, abs/2601.18005, 2026. URL https://arxiv.org/abs/2601.18005.

[6] Davis Brown, Jesse He, Helen Jenne, Henry Kvinge, and Max Vargas. 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.

[7] Róbert Istvan Busa-Fekete, Heejin Choi, Travis Dick, Claudio Gentile, and Andrés Muñoz Medina. Easy Learning from Label Proportions. *CoRR*, abs/2302.03115, 2023. doi: 10.48550/ARXIV.2302.03115. URL https://doi.org/10.48550/arXiv.2302.03115.

[8] François Charton, Jordan S. Ellenberg, Adam Zsolt Wagner, and Geordie Williamson. Patternboost: Constructions in mathematics with a little help from AI. *CoRR*, abs/2411.00566, 2024. doi: 10.48550/ARXIV.2411.00566. URL https://doi.org/10.48550/arXiv.2411.00566.

[9] Swarat Chaudhuri, Kevin Ellis, Oleksandr Polozov, Rishabh Singh, Armando Solar-Lezama, and Yisong Yue. Neurosymbolic programming. *Found. Trends Program. Lang.*, 7(3):158–243, 2021. doi: 10.1561/2500000049. URL https://doi.org/10.1561/2500000049.

[10] Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, Cheng Ren, Jiawei Shen, Wenlei Shi, Tong Sun, He Sun, Jiahui Wang, Siran Wang, Zhihong Wang, Chenrui Wei, Shufa Wei, Yonghui Wu,

Yuchen Wu, Yihang Xia, Huajian Xin, Fan Yang, Huaiyuan Ying, Hongyi Yuan, Zheng Yuan, Tianyang Zhan, Chi Zhang, Yue Zhang, Ge Zhang, Tianyun Zhao, Jianqiu Zhao, Yichi Zhou, and Thomas Hanwen Zhu. 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.

[11] Miles D. Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *CoRR*, abs/2305.01582, 2023. doi: 10.48550/ARXIV.2305.01582. URL https://doi.org/10.48550/arXiv.2305.01582.

[12] Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. 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.

[13] Michele D'Adderio, Alessandro Iraci, and Anna Vanden Wyngaerd. 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.

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

[15] Alex Davies, Petar Velickovic, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomasev, Richard Tanburn, Peter W. Battaglia, Charles Blundell, András Juhász, Marc Lackenby, Geordie Williamson, Demis Hassabis, and Pushmeet Kohli. Advancing mathematics by guiding human intuition with AI. *Nature*, 600(7887):70–74, 2021. doi: 10.1038/S41586-021-04086-X. URL https://doi.org/10.1038/s41586-021-04086-x.

[16] Adriano M. Garsia and James Haglund. 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).

[17] Eugene Gorsky, Graham Hawkes, Anne Schilling, and Julianne Rainbolt. Generalized q,t-Catalan numbers. *Algebraic Combinatorics*, 3(4):855–886, 2020. URL https://arxiv.org/abs/1905.10973.

[18] Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. Program synthesis. *Found. Trends Program. Lang.*, 4(1-2):1–119, 2017. doi: 10.1561/2500000010. URL https://doi.org/10.1561/2500000010.

[19] James Haglund. *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.

[20] Mark Haiman. 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.

[21] Baran Hashemi, Roderic G. Corominas, and Alessandro Giacchetto. Can transformers do enumerative geometry? In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://arxiv.org/abs/2408.14915.

[22] Shreyas Havaldar, Navodita Sharma, Shubhi Sareen, Karthikeyan Shanmugam, and Aravindan Raghuveer. 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.

[23] Xiaoyu Huang, Blake Jackson, and Kyu-Hwan Lee. 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.

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

[25] Mathlib community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, pages 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.

[26] Shinnosuke Matsuo, Ryoma Bise, Seiichi Uchida, and Daiki Suehiro. Learning from label proportion with online pseudo-label decision by regret minimization. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*, pages 1–5. IEEE, 2023. doi: 10.1109/ICASSP49357.2023.10097069. URL https://doi.org/10.1109/ICASSP49357.2023.10097069.

[27] Leonardo de Moura and Sebastian Ullrich. 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*, pages 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.

[28] T. Nathan Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P. Santiago, Daniel M. Faissol, and Brenden K. Petersen. Symbolic regression via deep reinforcement learning enhanced genetic programming seeding. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021.

[29] Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Claudio P. Santiago, Soo K. Kim, and Joanne T. Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://arxiv.org/abs/1912.04871.

[30] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A Field Guide to Genetic Programming*. lulu.com, 2008. ISBN 978-1-4092-0073-4. URL http://www.gp-field-guide.org.uk/.

[31] Novi Quadrianto, Alexander J. Smola, Tibério S. Caetano, and Quoc V. Le. 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.

[32] Jeffrey B. Remmel and Andrew T. Wilson. 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.

[33] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/S41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

[34] Clayton Scott and Jianxin Zhang. Learning from Label Proportions: A Mutual Contamination Framework. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[35] Michael Shalyt, Uri Seligmann, Itay Beit Halachmi, Ofir David, Rotem Elimelech, and Ido Kaminer. Unsupervised discovery of formulas for mathematical constants. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL https://doi.org/10.52202/079017-3596.

[36] John Shareshian and Michelle L. Wachs. Chromatic quasisymmetric functions. *Adv. Math.*, 295:497–551, 2016. doi: 10.1016/j.aim.2015.12.018. URL https://doi.org/10.1016/j.aim.2015.12.018.

[37] Sara Silva and Ernesto Costa. 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.

[38] Christian Stump. q, t-Fuß–Catalan numbers for finite reflection groups. *J. Algebraic Combin.*, 32(1):67–97, 2010. doi: 10.1007/s10801-009-0205-0. URL https://doi.org/10.1007/s10801-009-0205-0.

[39] Jennifer J. Sun, Megan Tjandrasuwita, Atharva Sehgal, Armando Solar-Lezama, Swarat Chaudhuri, Yisong Yue, and Omar Costilla-Reyes. Neurosymbolic programming for science. *CoRR*, abs/2210.05050, 2022. doi: 10.48550/ARXIV.2210.05050. URL https://doi.org/10.48550/arXiv.2210.05050.

[40] Adam Zsolt Wagner. Constructions in combinatorics via neural networks. *CoRR*, abs/2104.14516, 2021. URL https://arxiv.org/abs/2104.14516.

[41] Geordie Williamson. 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.

[42] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *J. Amer. Statist. Assoc.*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953. URL https://doi.org/10.1080/01621459.1927.10502953.

[43] Lisa Zhang, Gregory Rosenblatt, Ethan Fetaya, Renjie Liao, William E. Byrd, Matthew Might, Raquel Urtasun, and Richard Zemel. Neural guided constraint logic programming for program synthesis. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, Canada*, pages 1737–1746, 2018.

A q,t-combinatorics background & Narayana models

Triggered by the discovery of the famous Macdonald polynomials [24], 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 [20]. 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 [19, 13]. 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 [3]. 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 4). The set of all polyominoes of size $m \times n$ is denoted by $\mathsf{PP}(m,n)$.

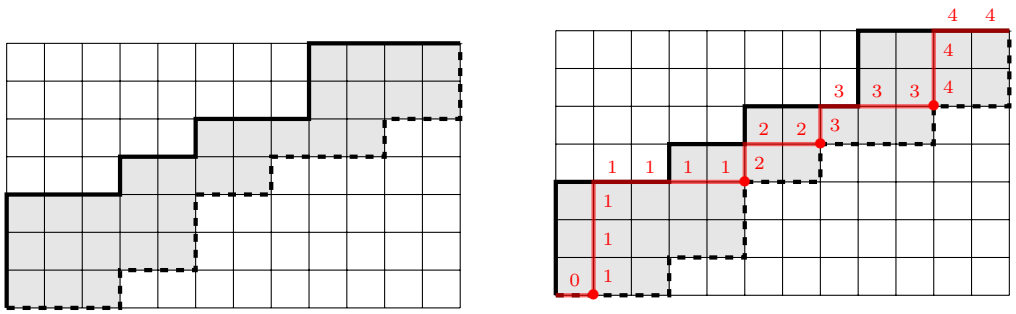


Figure 4: A 12×7 polyomino with area $30 - 18 = 12$ (left) and its bounce path (right); the bounce statistic is equal to $41 - 18 = 23$.

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).

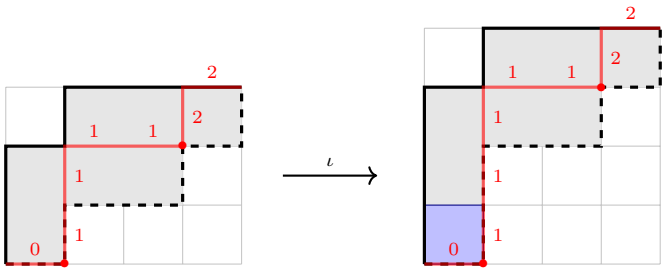
Parallelogram polyominoes have a *bounce path*. To compute it, start from $(0,0)$, draw a single east step, and then follow this algorithm: draw north steps until the path hits the polyomino's ceiling; then draw east steps until the path hits the polyomino's floor; repeat until it reaches (m,n), as shown in Figure 4.

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 [3]. Both pairs $(\text{area}, \text{bounce})$ and $(\text{dinv}, \text{area})$ 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.



$P \in \mathsf{PP}(4,3)$ $\quad$ $\iota(P) \in \mathsf{PP}(4,4)$

Figure 5: 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.

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*

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

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 [3, 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 5. 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 $\mathrm{area}(\iota(P)) = \mathrm{area}(P)$.

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 $\mathrm{bounce}(\iota(P)) = \mathrm{bounce}(P)$.

Thus, for every pair (a,b), the number of polyominoes with $(\mathrm{area}, \mathrm{bounce}) = (a,b)$ 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 6).

We omit the proof that η is a bijection; it is formalized in Lean via `Phi` and its inverse `Psi` (Appendix B). 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 $n \geq k \geq 1$:

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

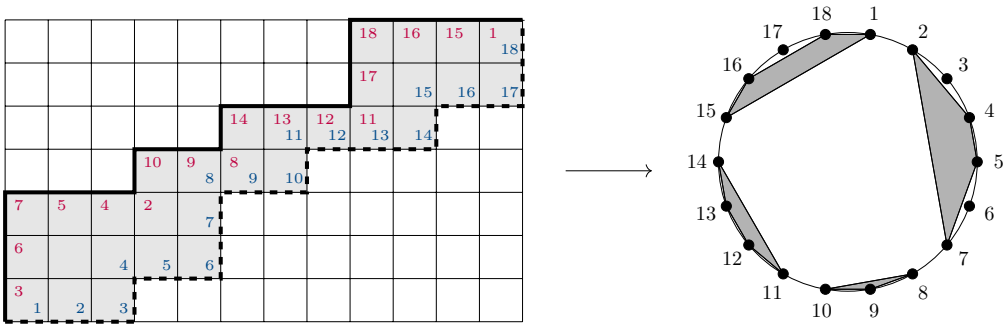


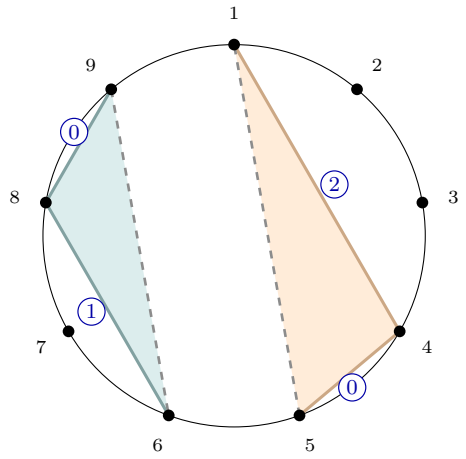
Figure 6: An example of the bijection η.

This is closely related to the known *inv* statistic on ordered set partitions [32].

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 $b_{i+1} - b_i - 1$. 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 7.



$\pi = \{1, 4, 5\} \mid \{2\} \mid \{3\}$
$\mid \{6, 8, 9\} \mid \{7\}$
$\mathrm{skip}(\pi) = (2 + 0) + (1 + 0) = 3$
$\{1, 4, 5\} : (4 - 1 - 1) + (5 - 4 - 1),$
$\{6, 8, 9\} : (8 - 6 - 1) + (9 - 8 - 1).$
Blue labels mark the counted edges; the gray dashed edges are ignored.

Figure 7: 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 our repository).

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 8 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.

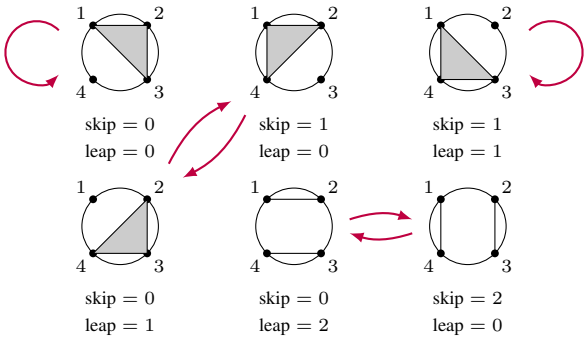


Figure 8: Exchanging bijection for $n = 4$ and $k = 3$.

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:

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

This is proved in Lean as `NC.skip_mingarc_pair` (Appendix B), by composing η (Definition A.4) with a bijection on $\mathsf{NC}(n,k)$ that turns area into skip and showing that the composite turns bounce into mingarc.

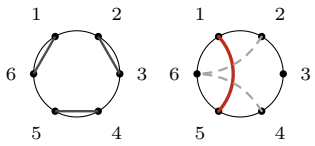
The case $k = 3$. There are exactly two candidate arcs. They overlap when the partition contains a triple block or a nested pair; in the non-nested case they are disjoint and both are kept. The statistic therefore evaluates to

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

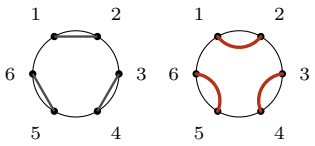
and in the nested case

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

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.



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

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 [27] for a detailed description of the system. Our formalization builds on Mathlib, the community-maintained library of definitions and theorems for Lean [25].

To accelerate the formalization effort, we made extensive use of *Aristotle*, Harmonic's Lean formalization system [1], 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.

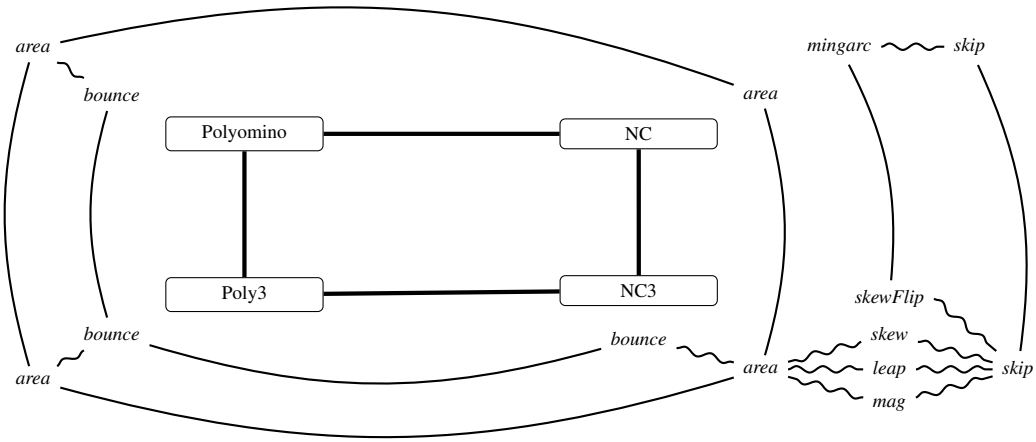


Figure 10: Overview of the objects and statistics covered by the Lean code. Thick edges indicate equivalences of types, and are intended with specific m, n, and k values. Curved thin edges indicate transport of statistics across these maps; squiggly edges indicate pairing relationships among statistics. All edges correspond to statements whose proofs are formalized.

At a high level, the development follows the structure summarized in Figure 10. 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 the core objects, statistics, and proof steps needed for the $k = 3$ results and for the all-k $(\text{skip}, \text{mingarc})$ pairing. For a reader-oriented entry point, we recommend opening `Formalization.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 $\leftrightarrow$ NC equivalence given by `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`).
- **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`, by transporting the $(\text{area}, \text{bounce})$ pairing on `Polyomino` through `Phi` and the inverse of `NC.maximaPacketsMinimumGapsEquiv`. 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`$\leftrightarrow$`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.

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, run the following from the repository root:

```
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 (`propext`, `Classical.choice`, and `Quot.sound`); these do not indicate missing proofs. For example, `#print axioms NC.skip_mingarc_pair` reports exactly these three axioms.

C Additional experimental results

Validation-size ablation. Table 3 reports separately the two independent batches of 50 runs that are pooled in Table 1: the original experiment and its replication with fresh seeds. The median disagreements are computed on successful runs only, so they are unreliable when there are few successes: at 10% validation, they rely on two runs in the original experiment, and the replication has no successful run. Clustering the 10 successful outputs of the replication at 5% validation, as in Section 6.1, gives two clusters of sizes 6 and 4, which correspond to the two largest clusters of the original experiment: one medoid is exactly mag, and the other differs from the medoid of the first cluster of Figure 3 on only 2 of the 2366 partitions. Clustering the 28 successful outputs of both batches at 5% validation gives clusters of sizes 18, 9, and 1, which enlarge the original clusters of Figure 3.

Held-out bags. To test an inductive setting, we held out 7 whole bags, about 5% of the partitions, from both training and constraint projection ($h = 2$, no warm start). All 50 runs satisfied the constraints on the training bags, but only 3 also satisfied them on the held-out bags (6%, with 95% Wilson confidence interval $[2.1\%, 16.2\%]$). One of these 3 runs yields the medoid of the first cluster of Figure 3, while we could not distill the other two. Thus inductive generalization is possible with MapSeek-Functional, but these few runs give no evidence that holding out whole bags favors outputs that are easier to distill.

Distilling a rule with an LLM. We originally found the formulas for mag and leap by inspecting the prediction vectors, using an LLM to propose candidate rules, and then simplifying and verifying them. To assess this step more systematically, we considered all distinct outputs of the successful runs of the homogeneity experiments of Section 6.1: 10 distinct outputs across 14 successful runs for $h = 1$, and 17 distinct outputs across 18 successful runs for $h = 2$. We gave each of them as input to GPT-5.6-sol (with extra-high reasoning effort, through Codex), using a prompt of the following form:

Table 3: The two batches of the validation-size ablation (refined $N_{14,3}$ skip-pairing dataset, $h = 2$, with 50 runs per setting and batch), with 95% Wilson confidence intervals for the success rate.

Batch	Val	# Success	95% CI	Med. disagree between runs	Med. disagree with mag	# Exact mag
Original	0%	49	[89.5%, 99.6%]	14.8%	18.6%	1
	1%	42	[71.5%, 91.7%]	11.7%	18.6%	2
	2%	27	[40.4%, 67.0%]	15.3%	15.3%	3
	5%	18	[24.1%, 49.9%]	16.9%	16.8%	1
	10%	2	[1.1%, 13.5%]	0.13%	0.06%	1
Replication	0%	49	[89.5%, 99.6%]	18.6%	18.9%	2
	1%	40	[67.0%, 88.8%]	10.4%	18.7%	2
	2%	34	[54.2%, 79.2%]	12.0%	18.6%	2
	5%	10	[11.2%, 33.0%]	10.7%	18.7%	1
	10%	0	[0.0%, 7.1%]	–	–	0

```
Find a simple formula or algorithm to describe this assignment of values to noncrossing partitions of rank 2:
((1, 2, 3), (4,), (5,), (6,), (7,), (8,), (9,), (10,), (11,), (12,), (13,), (14,)) 0
((1,), (2, 3, 4), (5,), (6,), (7,), (8,), (9,), (10,), (11,), (12,), (13,), (14,)) 2
[...]
```

The results are summarized in Table 4. Most proposed rules fit the $n = 14$ prediction vectors exactly, but usually through ad hoc cases or lookup tables. Upon manual inspection, only one rule per setting is a uniform formula or algorithm with no ad hoc cases or lookup tables: leap for $h = 1$ and mag for $h = 2$. The average runtime was 7.2 minutes per output (between 2.6 and 18.8 minutes).

Distilling a rule with symbolic regression. We ran PySR [11] on the same outputs, with 16 populations of 50 expressions, maximum complexity 50, and 9 minutes per output. We used arithmetic operations, comparisons, conditionals, floor, ceiling, and absolute value as primitives, and restricted constants to integers (with free continuous constants disabled). The three PySR formulas that remain correct when checked for larger n (Table 4) describe leap, mag, and the medoid of the first cluster of Figure 3. Note that the choice of primitives injects a human prior: in particular, we included floor and ceiling because the formula for mag involves rounding, and PySR might otherwise have missed it.

Table 4: Distilling rules from the distinct outputs of successful MapSeek-Functional runs (h-homogeneity experiments on the refined $N_{14,3}$ skip-pairing dataset). We report how many proposed rules fit the outputs exactly for $n = 14$, how many of them remain correct when checked for larger n, and (for the LLM) how many are uniform rules with no ad hoc cases or lookup tables.

		LLM			PySR	
h	Distinct outputs	Exact fit	Larger n	Uniform	Exact fit	Larger n
1	10	9	8	1 (leap)	2	1
2	17	15	12	1 (mag)	3	2

The case $k = 4$. We also ran both methods on the $N_{12,4}$ skip-pairing dataset, consisting of 9075 partitions. MapSeek-Symbolic never reached zero constraint distance. MapSeek-Functional had no successful runs with 5% or 10% validation, and 130 successful runs out of 1000 with 1% validation and warm start. These 130 runs all produced distinct outputs (the median pairwise agreement is 50.4%, and even the closest pair agrees on only 75.4% of the partitions), and we could not distill any of them into a uniform formula or algorithm. This suggests that the current representations and implementations of MapSeek do not yet handle this more complex case.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state all contributions: the SLURP framework, the two methods (MapSeek-Functional and MapSeek-Symbolic), the new combinatorial interpretations of q,t-Narayana polynomials, the exchanging bijection solving Open Problem 4.1 for $k = 3$, the (skip, mingarc) interpretation of $N_{n,k}(q,t)$ for all k, and the Lean 4 formalization. The introduction also clarifies the respective roles of the ML methods and of the mathematicians. These match the ML and mathematical results presented.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 8 discusses limitations: the successful experiments concern a single mathematical family, both pipelines rely on expert-designed representations and inductive biases, validation is transductive, distillation requires expert judgment, there is no domain-independent measure of simplicity, some analyses rely on few successful runs, and both methods fail for $k = 4$ (see Appendix C for quantitative results).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The main mathematical results, including the all-k (skip, mingarc) pairing, are formally proved and machine-verified in Lean 4 (Appendix B), with proof sketches in Section 7 and Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 6 provides full architecture details, optimizer choices, hyperparameters, dataset sizes, and training procedures for both MapSeek-Functional and MapSeek-Symbolic. The skip-pairing dataset is released with code to regenerate it for any n, k.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code for MapSeek, the skip-pairing benchmark, and the Lean 4 formalization are publicly released (see the link in the introduction).

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 6 details model architectures, optimizers, batch sizes, validation split sizes, number of iterations, and all relevant hyperparameters for both methods; the additional experiments are described in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The primary metric is exact constraint satisfaction (binary success per run). We report success counts with 95% Wilson confidence intervals (Table 1 and Appendix C), pooling two independent batches of runs in the validation-size ablation; statistics computed on few successful runs are presented as descriptive.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [No]

Justification: The paper does not report the type of hardware (CPU/GPU), memory, or execution times for the MapSeek experiments; only the runtimes of the distillation analysis are reported (Appendix C). The experiments are relatively lightweight given the small datasets (2366 and 9075 partitions, small Transformer models).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: This paper presents foundational research in machine learning and combinatorics. It does not involve human subjects, sensitive data, or applications with negative societal impact, and conforms fully with the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is foundational research in mathematical discovery with no foreseeable negative societal impact. The positive impact is to advance ML-assisted mathematics; there is no direct path to harmful applications.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released assets (code, dataset generator, and Lean 4 proofs) pose no misuse risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper uses Lean 4 (Apache 2.0), Mathlib (Apache 2.0), PySR (Apache 2.0), and standard ML libraries such as PyTorch, all of which are open-source and properly cited or standard infrastructure. We used an LLM (GPT-5.6-sol, through Codex) for distillation under its terms of service. All mathematical results and datasets used are based on prior work that is cited in the references.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces the skip-pairing benchmark dataset (described in Section 5) and the Lean 4 formalization of all mathematical results (described in Appendix B). Both are documented in the paper and publicly released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects, so no IRB approval is required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: An LLM was used to assist with the autoformalization of mathematical results into Lean 4, under the guidance of one of the authors, who is a Lean expert. We also used LLMs to help distill the outputs of MapSeek-Functional into formulas (Section 6.1 and Appendix C). This is documented in the paper.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.