

nth: Open-Source Generic Combinatorial (Un)ranking via the Symbolic Method

Marcos Tomaszewski **Gustavo Zambonin**
Jean Everson Martina Ricardo Custódio

Universidade Federal de Santa Catarina

July 21, 2026

Software for Enumerative and Analytic Combinatorics
ICMS 2026

An abstract combinatorial class

Definition ([FS09, p. 16])

A *combinatorial class* $(\mathcal{A}, \mathcal{B}, \dots)$ is a finite or denumerable set on which a *size function* is defined, where

- ❶ the size of an element is a non-negative integer;
- ❷ the number of elements of any given size is finite.

The size of an element $\alpha \in \mathcal{A}$ is denoted $|\alpha|$.

Definition ([FS09, p. 16])

The *counting sequence* of $\mathcal{A}$ is $(A_n)_{n \geq 0}$, where $A_n = |\{\alpha \in \mathcal{A} : |\alpha| = n\}|$.

Such sequences can be characterized by a formal power series.

Generating functions

In an “unlabeled” class the atoms (classes of size 1) are indistinguishable.

Definition ([FS09, p. 19])

The *ordinary* generating function (OGF) of a sequence A_n is $\sum_{n \geq 0} A_n z^n$; the OGF of class $\mathcal{A}$ is $\sum_{\alpha \in \mathcal{A}} z^{|\alpha|}$.

In a “labeled” class the n atoms bear distinct labels 1 to n , and we count it analogously.

Definition ([FS09, p. 97])

The *exponential* generating function (EGF) of a sequence A_n is $\sum_{n \geq 0} A_n \frac{z^n}{n!}$; the EGF of class $\mathcal{A}$ is $\sum_{\alpha \in \mathcal{A}} \frac{z^{|\alpha|}}{|\alpha|!}$.

Admissible constructions & the symbolic method

Definition ([FS09, p. 22])

Let Φ be an m -ary construction $\mathcal{A} = \Phi[\mathcal{B}^{(1)}, \dots, \mathcal{B}^{(m)}]$. Then Φ is *admissible* if and only if A_n only depends on $(B_n^{(1)}), \dots, (B_n^{(m)})$.

The *symbolic method* [FS09] turns each admissible construction into a *transfer rule*, an operator on the generating functions.

We describe a class by a *grammar*, a few equations over atoms and constructions. Plane binary trees, for instance, are

$$\mathcal{B} = \text{UNION}(\mathcal{Z}, \text{PROD}(\mathcal{B}, \mathcal{B})),$$

a leaf, or a node over two subtrees.

Basic admissibility, unlabeled universe

Theorem ([FS09, Ch. I])

Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be unlabeled classes with OGFs A, B, C . Then

$$\begin{aligned}\text{UNION} &\Rightarrow A = B + C, & \text{MSET} &\Rightarrow A = \prod_{n \geq 1} (1 - z^n)^{-B_n}, \\ \text{PROD} &\Rightarrow A = B C, & \text{PSET} &\Rightarrow A = \prod_{n \geq 1} (1 + z^n)^{B_n}, \\ \text{SEQ} &\Rightarrow A = \frac{1}{1-B}, & \text{CYC} &\Rightarrow A = \sum_{k \geq 1} \frac{\varphi(k)}{k} \log \frac{1}{1-B(z^k)}, \\ \text{SUBST} &\Rightarrow A = B(C(z)), & \text{POINT} &\Rightarrow A = z \partial_z B.\end{aligned}$$

Pointing means “pointing at a distinguished atom”; substitution means “substitute elements of $\mathcal{C}$ for atoms of $\mathcal{B}$ ”.

Basic admissibility, labeled universe

Theorem ([FS09, Ch. II])

Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be labeled classes with EGFs A, B, C . Then

$$\text{UNION} \Rightarrow A = B + C, \quad \text{BOXPROD} \Rightarrow A = \int_0^z (\partial_t B) C dt,$$

$$\text{PROD} \Rightarrow A = B C, \quad \text{SET} \Rightarrow A = e^B,$$

$$\text{SEQ} \Rightarrow A = \frac{1}{1-B}, \quad \text{CYC} \Rightarrow A = \log \frac{1}{1-B},$$

$$\text{SUBST} \Rightarrow A = B(C(z)), \quad \text{POINT} \Rightarrow A = z \partial_z B.$$

Definition ([FS09, p. 139])

The *boxed product* (alternatively $\mathcal{B}^\square \star \mathcal{C}$) is the subset of $\text{PROD}(\mathcal{B}, \mathcal{C})$ whose *smallest label* lies in $\mathcal{B}$ (assuming $B_0 = 0$).

Combinatorial (un)ranking

Definition ([KS98])

A *ranking* of the size- n objects $\mathcal{C}_n$ is a bijection $\mathbf{R} : \mathcal{C}_n \rightarrow [c_n - 1]$. It fixes a *total order*, with s before t when $\mathbf{R}(s) < \mathbf{R}(t)$; its inverse $\mathbf{U} = \mathbf{R}^{-1}$ is *unranking*.

Theorem ([Mol05, Thms. 9, 10])

The worst-case time complexity of unranking for objects of size n in any admissible class using lexicographic ordering is $\mathcal{O}(n^2)$.

Theorem ([Mol05, Thms. 16, 17])

The worst-case time complexity of unranking for objects of size n in any admissible class using boustrophedonic ordering is $\mathcal{O}(n \log n)$.

Why (un)ranking, why generic?

Unranking *decompresses* a rank back to its object in that order. As a special case it gives exact-size *uniform sampling*.

(Simply *draw* $\lceil \lg c_n \rceil$ random bits and evaluate **U**.)

By generic we mean **R** and **U** come from the grammar alone, with no *ad hoc* code per structure.

No open, maintained, permissively licensed tool does this, so we built our own, standing on the shoulders of giants.

Three lineages of automatic combinatorics software

The *Maple* line: LUO [FSZ91] $\rightarrow$ gfun [SZ94], Gaïa [Zim94] $\rightarrow$ combstruct [FS95]. It is still shipped in Maple, but generic (un)ranking was never added [Mis03].

The *MuPAD* line: CS [DDZ98] $\rightarrow$ MuPAD-Combinat [HT04] $\rightarrow$ SageMath, our closest ancestor. MathWorks bought MuPAD in 2008, and grammar-driven generation never reached Sage.

For an alternative line of samplers, we have *Boltzmann* [Duc+04], at approximate size: GenRGenS [PTD06] $\rightarrow$ Paganini [BBD18] $\rightarrow$ CombOL [Eri26], our modern yardstick.

Enter nth

`nth` is an open-source (MIT) C library for generic (un)ranking from a grammar, over the aforementioned admissible operators.

We provide an ergonomic API and CLI, as well as Python and SageMath bindings.

We benchmark counting and drawing against `combstruct` and `CombOL`, and draw faster over a useful range of sizes.

`nth` depends on FLINT [The26] for arbitrary-precision integral, rational and polynomial types to calculate counts and ranks.

Deriving rank from the grammar

Martínez and Molinero [MM01] derive **R** and **U** from *any* labeled grammar, partitioning the rank space at each node.

They hand out label sets by binomial coefficients, and rank the subsets with the combinatorial number system [Knu11].

Molinero [Mol05] extended this to the *unlabeled* world, where there are no labels to distribute and components are canonicalized instead.

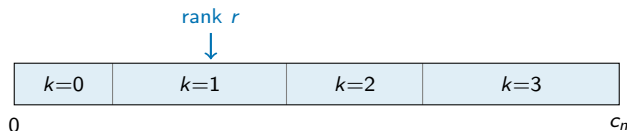
(The elusive exception is unlabeled CYC.)

Counting, then ranking

First `nth` counts: for every symbol it builds the exact coefficient array $[c_0, \dots, c_N]$, with FLINT's truncated power series.

To rank an object we recurse on its grammar. At each node the construction cuts the rank space $[0, c_n)$ into blocks, one per way of building the object; the block widths are those counts.

We keep the block holding our rank, and recurse.



The “ordered” constructions

$\text{PROD}(\mathcal{A}, \mathcal{B})$ splits by the left size k : the offset $\sum_{j < k} a_j b_{n-j}$ counts every object with a smaller left part.

(When labeled, a binomial also picks which labels go left.)

$\text{UNION}(\mathcal{A}, \mathcal{B})$ splits at a_n : the rank sits in the $\mathcal{A}$ block $[0, a_n)$ or the $\mathcal{B}$ block $[a_n, a_n + b_n)$.

$\text{SEQ}(\mathcal{A})$ splits by the number of parts, and then by their sizes.

Breaking symmetry: the boxed product

SET and CYC put no order on their components, so ranking has nothing to split on. The boxed product breaks the tie.

That one component is now distinguished, and the rest ranks exactly like PROD. We choose which labels join it, rank the component, then rank what remains.

A single device then covers SET, CYC, and BOXPROD.

The unlabeled world

Without labels there is nothing to hand out, so the unordered constructions *canonicalize* their components instead.

A shared decoder numbers every component of size $\leq n$, and reads a SET off as an increasing list of those numbers.

That table can grow large, so we cap it at 2^{24} and return a clean error past the cap.

Do we trust it?

We recomputed every entry of the legacy Encyclopedia of Combinatorial Structures [Lum18] against the full OEIS.

About 1018 of 1075 agree, but the mismatches are the instructive part. We also have sign and offset differences to take into account.

We also let `nth fuzz` itself, generating random grammars and keeping those whose counts hit an OEIS sequence.

Every entry is then checked (not proved) in three ways: its counts match OEIS, ranking and unranking round-trip in both orders, and all distinct ranks give distinct objects.

Experiments

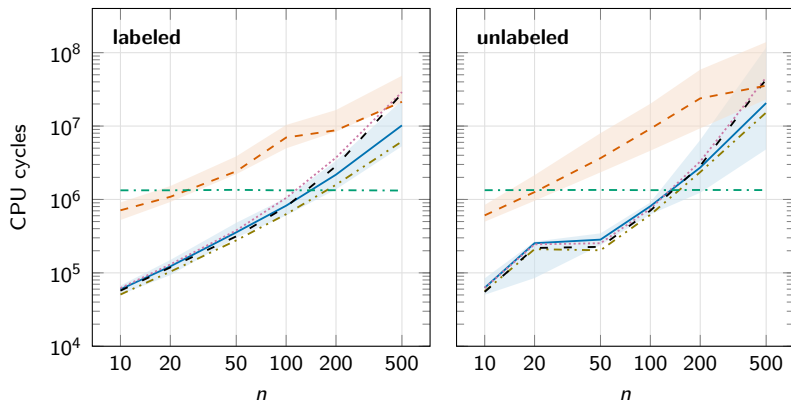
We take the 20 classes from the `combstruct` manual. This selection also closely matches the benchmarks of Molinero's thesis [Mol05].

We also include plane-binary and Motzkin trees so that CombOL is exercised too.

On the next slide, we have the geometric mean of the performance for all such classes. The bands are interquartile ranges.

Unsurprisingly, the graphs match the theory: lexicographic order is usually slower, so is `combstruct`, and Boltzmann is constant time.

Results



n th solid, no memo. dotted. boustrophedonic dash-dot-dotted, both loosely dashed; combstruct dashed, CombOL dash-dotted.

Where this goes next

We are actively working on the unranking of unlabeled CYC, and formally verifying the algorithms using Lean 4.

We invite ports of the C API to other languages, such as Julia or even Maple, as well as making this a first-class SageMath interface.

Partially supported by CAPES (Finance Code 001), FAPESC (Edital 62/2024), and the ON-RCPN of Brazil.

Thank you! Any questions?

<https://github.com/zambonin/nth>

<https://github.com/zambonin/nth-tools>

References I

- [BBD18] Maciej Bendkowski, Olivier Bodini, and Sergey Dovgal. “Polynomial tuning of multiparametric combinatorial samplers”. In: *Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*. Jan. 2018, pp. 92–106. DOI: 10.1137/1.9781611975062.9.
- [DDZ98] Alain Denise, Isabelle Dutour, and Paul Zimmermann. “CS: a MuPAD package for counting and randomly generating combinatorial structures”. In: *Proceedings of the 10th Conference on Formal Power Series and Algebraic Combinatorics (FPSAC'98)*. June 1998, pp. 195–204.
- [Duc+04] Philippe Duchon, Philippe Flajolet, Guy Louchard, and Gilles Schaeffer. “Boltzmann Samplers for the Random Generation of Combinatorial Structures”. In: *Combinatorics, Probability and Computing* 13.4-5 (July 2004), pp. 577–625. DOI: 10.1017/S0963548304006315.
- [Eri26] Casper Asbjørn Eriksen. *CombOL: The Combinatorial Objects Library*. 2026. URL: <https://gitlab.com/casbjorn/combol>.
- [FS09] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. 1st. Cambridge University Press, 2009. ISBN: 9780521898065. DOI: 10.1017/CB09780511801655.

References II

- [FS95] Philippe Flajolet and Bruno Salvy. “Computer algebra libraries for combinatorial structures”. In: *Journal of Symbolic Computation* 20.5 (Nov. 1995), pp. 653–671. DOI: 10.1006/jsco.1995.1070.
- [FSZ91] Philippe Flajolet, Bruno Salvy, and Paul Zimmermann. “Automatic average-case analysis of algorithms”. In: *Theoretical Computer Science* 79.1 (Feb. 1991), pp. 37–109. DOI: 10.1016/0304-3975(91)90145-R.
- [HT04] Florent Hivert and Nicolas M. Thiéry. “MuPAD-Combinat, an open-source package for research in algebraic combinatorics”. In: *Séminaire Lotharingien de Combinatoire* 51 (2004). Article B51z.
- [Knu11] Donald E. Knuth. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*. 1st. Addison-Wesley Professional, 2011. ISBN: 9780201038040.
- [KS98] Donald L. Kreher and Douglas R. Stinson. *Combinatorial Algorithms: Generation, Enumeration, and Search*. 1st. CRC Press, 1998. ISBN: 9781003068006.
- [Lum18] Jérémie Lumbroso. *Encyclopedia of Combinatorial Structures Dataset*. 2018. URL: <https://github.com/jlumbroso/encyclopedia-of-combinatorial-structures-data>.

References III

- [Mis03] Marni Mishna. “Attribute grammars and automatic complexity analysis”. In: *Advances in Applied Mathematics* 30.1-2 (Feb. 2003), pp. 189–207. DOI: 10.1016/S0196-8858(02)00542-0.
- [MM01] Conrado Martínez and Xavier Molinero. “A Generic Approach for the Unranking of Labeled Combinatorial Classes”. In: *Random Structures & Algorithms* 19.3-4 (Oct. 2001), pp. 472–497. DOI: 10.1002/rsa.10025.
- [Mol05] Xavier Molinero. “Ordered generation of classes of combinatorial structures”. PhD thesis. Universitat Politècnica de Catalunya, July 2005.
- [PTD06] Yann Ponty, Michel Termier, and Alain Denise. “GenRGenS: software for generating random genomic sequences and structures”. In: *Bioinformatics* 22.12 (Mar. 2006), pp. 1534–1535. DOI: 10.1093/bioinformatics/btl113.
- [SZ94] Bruno Salvy and Paul Zimmermann. “GFUN: a Maple package for the manipulation of generating and holonomic functions in one variable”. In: *ACM Transactions on Mathematical Software* 20.2 (June 1994), pp. 163–177. DOI: 10.1145/178365.178368.
- [The26] The FLINT team. *FLINT: Fast Library for Number Theory*. Version 3.6.0. 2026. URL: <https://flintlib.org>.

References IV

- [Zim94] Paul Zimmermann. “Gaïa: a package for the random generation of combinatorial structures”. In: *MapleTech* 1.1 (1994), pp. 38–46.