Combinatorial Species
18.204 paper

Anthony Wang

December 17, 2025

1 Introduction

What is the derivative of the list data type? That might sound like an absurd question, but in the 1990s, computer scientists discovered that data types like lists, binary trees, and multisets can surprisingly be manipulated like algebraic objects. Taking the derivative of the list data type is in fact a well-defined operation and even has a nice combinatorial interpretation.

First, we must understand data types. In computer science, the two main ways of creating more complex data types out of simpler data types are by using sum types and product types.

Definition 1.1. A *sum type* $A+B$ is a data type that either contains an element a of type A or an element b of type B.

Definition 1.2. A *product type* AB, also denoted $A \cdot B$, is a data type that contains the pair (a, b) where a is of type A and b is of type B.

For example, consider an ordered list of integers. This list is either the empty list, or an integer concatenated with another list. Clearly, it is a sum type. In one case, we have an empty list, which we will represent as 1. This is because the product of an empty list with any data type is still isomorphic to the original data type, so the empty list is analogous to 1. In the other case, we have a pair of an integer and a list, which is a product type. Thus,

$$\mathsf{List}(\mathsf{Int}) = 1 + \mathsf{Int} \cdot \mathsf{List}(\mathsf{Int}).^{1}$$

If we have a list of x where x is some specific data type, then we get

$$\begin{aligned} \mathsf{List}(x) &= 1 + x\mathsf{List}(x) \\ &= \frac{1}{1-x}. \end{aligned}$$

[1] Technically, $\mathsf{List}(\mathsf{Int})$ is the least fixed point of this equation, since it has multiple solutions.

Now we can take the derivative to get

$$\begin{aligned}\frac{d\mathsf{List}(x)}{dx} &= \frac{1}{(1-x)^2} \\ &= \mathsf{List}(x)^2.\end{aligned}$$

Thus, the derivative of the list data type is two lists.

However, it also seems like we performed many illegal and dubious operations, since we never defined subtraction or division for data types, and we differentiated with respect to x which is not a real number. Still, we can try to find a combinatorial interpretation for our answer using dual numbers.

Definition 1.3. A *dual number* is an expression of the form $a + b\epsilon$ where $\epsilon \neq 0$ and $\epsilon^2 = 0$.

Intuitively, ϵ behaves like an infinitesimal. The following property of dual numbers is famously used by forward-mode automatic differentiation:

Lemma 1.1. Let f be a sufficiently differentiable real-valued function. Then

$$\frac{df}{dx}\epsilon = f(x+\epsilon) - f(x).$$

Proof. Use Taylor expansion. □

This lemma suggests that

$$\frac{df}{dx} = \frac{f(x+\epsilon) - f(x)}{\epsilon},$$

which resembles the definition of differentiation using limits, but sadly division by ϵ is undefined, as it would involve division by zero. However, we can still use it because we have not yet reached the rigorous part of this paper yet.

The data type equivalent of ϵ is a value where when paired with itself, it annihilates the pair. Consider the equation

$$\frac{d\mathsf{List}(x)}{dx} = \frac{\mathsf{List}(x+\epsilon) - f(x)}{\epsilon}.$$

$\mathsf{List}(x+\epsilon)$ can be interpreted as a list where every element is either of type x or ϵ, and if multiple ϵs appear in the list, then the list annihilates itself. Thus, $\mathsf{List}(x+\epsilon)$ is the data type consisting of all lists with at most one ϵ. If we subtract $\mathsf{List}(x)$, we obtain all lists with exactly one ϵ. Finally, we can divide by ϵ to remove the ϵ from all lists. Therefore, the derivative of the list data type is a list with a hole in it, which splits it into two lists.

Again, this explanation is intuitively nice but also involves illegal and dubious operations. Fortunately, these deep connections between data types and algebra can be made rigorous using the beautiful theory of *combinatorial species*, first introduced by André Joyal in 1981[2][5]. The key idea is to view data types as bijection-preserving functions from sets of labels to sets of structures.

Operations on these functions can be interpreted both combinatorially and algebraically, using exponential generating functions. Combinatorial species even have some applications for computer programming, such as automated test data generation and manipulation of data types.

In Section 2, we will formally define combinatorial species. Furthermore, in Section 3, we will define various algebraic operations on species. Section 4 adapts the Lagrange inversion theorem from real analysis and proves it for combinatorial species. Finally, in Section 5, we will prove Cayley's formula in three lines using species and the Lagrange inversion theorem, and state a few extensions in Section 6.

This paper assumes basic knowledge of calculus and category theory[3].

2 Definitions

Definition 2.1. A *combinatorial species* F is a functor from the category of finite sets and bijections to the category of finite sets and total functions.

Intuitively, we can view F as a function that maps a finite set of labels U to a finite set of structures $F[U]$, as well as mapping bijections $U \to V$ to $F[U] \to F[V]$.

Definition 2.2. Two species F and G are *equivalent*, denoted as $F = G$, if there exists a natural transformation from F to G.

Intuitively, this means that F and G are isomorphic.

Definition 2.3. Two species F and G are *equipotent*, denoted as $F \equiv G$, if $|F[U]| = |G[U]|$ for all finite sets U.

Definition 2.4. The *exponential generating function* (EGF) for a species F is the formal power series

$$F(x) = \sum_{n=0}^{\infty} |F[n]| \frac{x^n}{n!}$$

where $F[n] = F[\{1, 2, \cdots, n\}]$.

For example, the combinatorial species $\mathsf{List}[U]$ maps a set of labels U to ordered lists of length $|U|$ where each element is uniquely assigned one of the labels. Thus, its EGF is

$$\begin{aligned} \mathsf{List}(x) &= \sum_{n=0}^{\infty} |\mathsf{List}[n]| \frac{x^n}{n!} \\ &= \sum_{n=0}^{\infty} n! \frac{x^n}{n!} \\ &= \frac{1}{1-x}. \end{aligned}$$

This is exactly the function we obtained in the introduction!

Figure 1: The list species mapping $E = \{0, 1, 2\}$ to the 6 possible lists of length $|E|$.

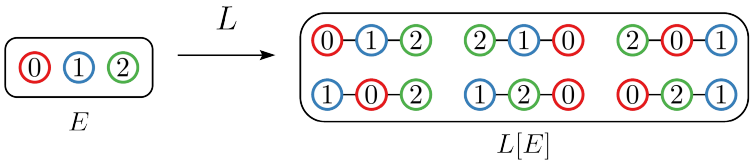
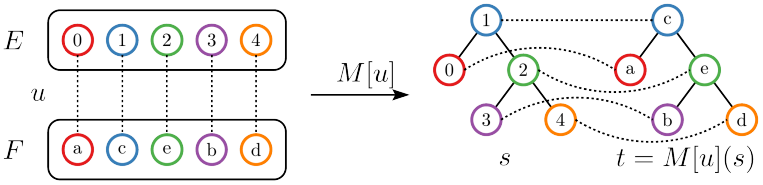


Figure 2: The tree species M mapping a bijection u between E and F to a bijection $M[u]$ between trees s and t.



3 Operations

Now that we have defined combinatorial species and EGFs, we can define various operations for species.

Definition 3.1. The *sum* of species F and G is the species

$$(F + G)[U] = F[U] \sqcup G[U]$$

where $\sqcup$ denotes the disjoint union.

This corresponds to the sum type in computer science. The number of structures for $F + G$ with size n is simply the sum of the number of structures for F and for G of size n, so its EGF is

$$(F + G)(x) = F(x) + G(x).$$

Definition 3.2. The *product* of species F and G is the species

$$(FG)[U] = \bigsqcup_{A \sqcup B = U} F[A] \times G[B].$$

This is the product type from computer science. The EGF of the product is

$$(FG)(x) = F(x)G(x).$$

Definition 3.3. The *composition* of species F and G such that $G(0) = 0$ is

$$(F \circ G)[U] = \bigsqcup_{\pi} (F(\pi) \times \prod_{V \in \pi} G(V))$$

where π is an unordered partition of U.

Figure 3: The product of two species

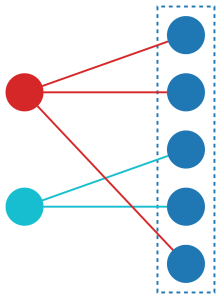
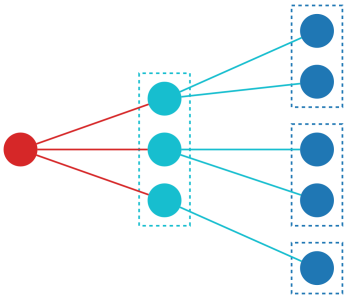


Figure 4: The composition of two species



This corresponds to composition of data types in computer science. The EGF is

$$(F \circ G)(x) = F(G(x)).$$

Definition 3.4. The *derivative* of a species is

$$F'[U] = F[U \sqcup \{\star\}]$$

where $\star$ is some distinguished element not in U.

Intuitively, this corresponds to making a hole in the structures. For a term in the EGF, we have

$$\frac{d}{dx} F[n] \frac{x^n}{n!} = \frac{d}{dx} F[n] \frac{x^{n-1}}{(n-1)!},$$

so thus differentiating the EGF decreases the size of the structure by one. Thus, the EGF of the derivative of a species is

$$F'(x) = \frac{dF(x)}{dx}.$$

The derivative of a species satisfies all the usual properties of the derivative, such as the sum, product, and chain rules.

4 Lagrange Inversion Theorem

In real analysis, the Lagrange inversion theorem gives the Taylor expansion of the inverse of an analytic function. There is an analogous result in the theory of combinatorial species, which can be proved combinatorially. First, we will state some definitions and prove some helpful lemmas.

Definition 4.1. An *endofunction* is a function whose domain equals its codomain.

Figure 5: The derivative of a species

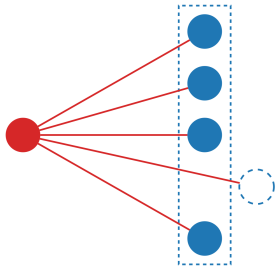
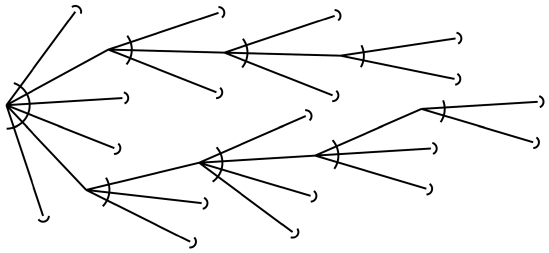


Figure 6: An R-enriched tree, where the arcs on each vertex denote the R species



Definition 4.2. An endofunction $\phi : E \to E$ is *R-enriched* if each of its fibers $\phi^{-1}\{x\}$ for $x \in E$ is equipped by the R species.

Lemma 4.1. The species $C_R = X \cdot R'[A_R]$ is equivalent to that of R-enriched contractions, where a contraction is an endofuction that ultimately becomes constant.

Proof. Let $\phi : E \to E$ be an R-enriched contraction and x_0 be the point of convergence of ϕ. Then we can partition E into $\{x_0\}$, which corresponds to the X, and $E - \{x_0\}$, which is an R'-assembly of A_R structures, where the derivative in R' is because of the loop at x_0. This is a bijection, as shown in Figure 7. □

Lemma 4.2. Let S be the species of permutations and D_R be the species of R-enriched endofunctions. Then

$$S(C_R) = D_R.$$

Proof. Any permutation can be decomposed into cycles. If we equip each node of the cycle with an R-enriched contraction, then we obtain an R-enriched endofunction. This process is reversible so it is a bijection, as shown in Figure 8. □

Figure 7: Visual proof of the first lemma

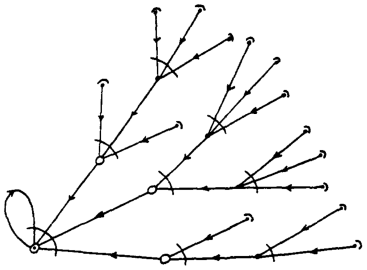
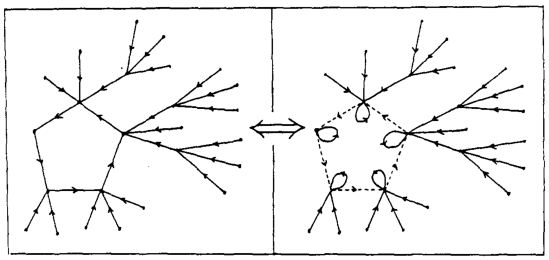


Figure 8: Visual proof of the second lemma



Lemma 4.3 (Labelle's Repartitioning Lemma)**.** For any species G, a $G(A_R)D_R$-structure on a set E can be interpreted as partitioning E into $E = E_1 \sqcup E_2$ such that E_1 is equipped with a G-structure γ and E_2 is equipped with an R-enriched function λ from E_2 to E. Furthermore, we have

$$(G(A_R)D_R)[n] = (GR^n)[n].$$

Proof. Let F_1 be the $G(A_R)$ portion of the $G(A_R)D_R$ species and F_2 be the D_R portion. Joyal's original paper says to "meditate" on Figure 9 to complete the proof.

□

Theorem 4.1 (Lagrange Inversion Theorem)**.** Let R, F be species and A_R be the species of R-enriched rooted trees. Then for $n \geq 1$, we have

$$F(A_R)[n] = F'R^n[n-1].$$

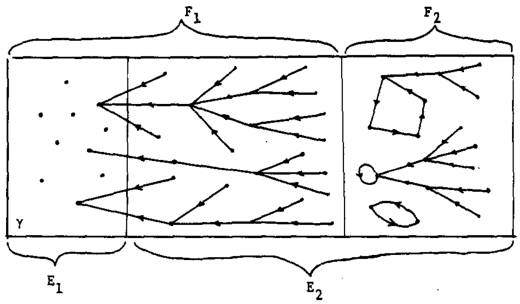
Proof. First, note that the derivative of a power series has the property

$$F(A_R)[n] = F(A_R)'[n-1].$$

Using the chain rule, we have

$$F(A_R)' = F'(A_R)A_R'.$$

Figure 9: Visual proof of Labelle's repartitioning lemma



We will first compute A'_R. An R-enriched rooted tree can be split into the root and its children, so

$$A_R = X \cdot R(A_R).$$

Taking the derivative, we get

$$\begin{aligned} A'_R &= R(A_R) + X \cdot R'(A_R)A'_R \\ &= R(A_R) + C_R A'_R \\ &= \frac{R(A_R)}{1 - C_R}. \end{aligned}$$

We recognize $\frac{1}{1-C_R}$ as the species of lists of C_Rs. The species of lists $\frac{1}{1-X}$ is equipotent to the species $S(X)$ because there are the same number of ordered lists of n labels as permutations of n labels. However, they are not equivalent because all lists of the same length are equal up to relabelling, but not for permutations since a relabelling cannot map cycles of different sizes. There exists a bijection between lists and permutations, but it is not natural in the sense of a category theoretic natural transformation. Using Lemma 4.2, we get

$$\begin{aligned} A'_R &\equiv R(A_R)S(C_R) \\ &\equiv R(A_R)D_R. \end{aligned}$$

Thus, we get

$$F(A_R)' \equiv F'(A_R)R(A_R)D_R.$$

Finally, we can conclude using Labelle's repartitioning lemma that

$$\begin{aligned} F(A_R)[n] &= F(A_R)'[n-1] \\ &= F'(A_R)R(A_R)D_R[n-1] \\ &= (F'R)(A_R) \cdot D_R[n-1] \\ &= (F'R)R^{n-1}[n-1] \\ &= F'R^n[n-1]. \end{aligned}$$

□

5 Cayley's Formula

One application of the Lagrange inversion formula for combinatorial species is proving Cayley's formula, a classic result in combinatorics.

Theorem 5.1 (Cayley's Formula)**.** There are n^{n-2} labeled unrooted trees on n labeled vertices.

Proof. This is equivalent to showing that there are n^{n-1} rooted trees on n labeled vertices, because we have n ways to choose a root for an unrooted tree.

Let A be the species of rooted trees and F be the species of rooted forests. Let E be the species of sets, which has EGF e^x. A rooted forest is a set of rooted trees, so

$$F = E(A).$$

Additionally, a rooted tree is a product of the root and its subtrees, which form a forest, so

$$A = X \cdot E(A).$$

We can now apply the Lagrange inversion theorem to compute $E(A)$. Since $A_E = A$, let $R = E$. Then

$$\begin{aligned} E(A_R)[n] &= (E' \cdot E^n)[n-1] \\ &= (E \cdot E^n)[n-1] \\ &= E^{n+1}[n-1] \\ &= (n+1)^{n-1}. \end{aligned}$$

Thus, the EGF of A is

$$\begin{aligned} A(x) &= x \sum_{n=0}^{\infty} (n+1)^{n-1} \frac{x^n}{n!} \\ &= \sum_{n=1}^{\infty} n^{n-1} \frac{x^n}{n!} \end{aligned}$$

Therefore, there are n^{n-1} labeled rooted trees and n^{n-2} unlabeled rooted trees on n vertices. □

Alternatively, we could have algebraically manipulated the EGFs for these species to get

$$x = A(x)e^{-A(x)}.$$

Then, we can apply the real analysis version of the Lagrange inversion theorem to compute $A(x)$, which explains the connection between the combinatorial theorem that we proved and its real analysis counterpart.

6 Extensions

6.1 Cycle index series

An EGF counts the number of labeled structures of a certain size, but we can also define an *ordinary generating function* that counts unlabeled structures of a certain size. These are both special cases of a more general generating function called the *cycle index series*, which is useful for many applications such as proving the Pólya enumeration theorem.[1] Combinatorial species provide a rigorous framework for dealing with the relationship between labeled and unlabeled structures.

6.2 Virtual species

We defined the operations of addition and multiplication for species, which forms a semiring. We can complete this into a ring using *virtual species*, which are equivalence classes of the equivalence relation on pairs of species such that $(F, G) \sim (H, K)$ if and only if $F + K = G + H$.[4] We can then rigorously define subtraction and division of species, as well as a combinatorial logarithm which generalizes the notion of structures built from connected components. This finally enables us to rigorously justify the algebraic manipulations in the introduction.

References

[1] Andy Hardt, Pete McNeely, Tung Phan, and Justin M. Troyka. Combinatorial species and graph enumeration, 2013.

[2] André Joyal. Une théorie combinatoire des séries formelles. *Advances in Mathematics*, 42(1):1–82, 1981.

[3] Bartosz Milewski. The Dao of Functional Programming. `https://github.com/BartoszMilewski/DaoFP`, 2025.

[4] Brent A. Yorgey. Species and functors and types, oh my! In *Proceedings of the Third ACM Haskell Symposium on Haskell*, Haskell '10, page 147–158, New York, NY, USA, 2010. Association for Computing Machinery.

[5] Brett Yorgey. A combinatorial theory of formal series (English translation with commentary). `http://ozark.hendrix.edu/~yorgey/pub/series-formelles.pdf`, 2019.