Research Paper Based on
"Species and Functors and Types, Oh My!"

Combinatorial Species

Jonathan Dygert

October 31, 2019

1 Introduction

Combinatorial species were introduced by French mathematician André Joyal in his 1981 paper, "Une théorie combinatoire des séries formelles", to unify branches of combinatorics. The theory of combinatorial species is still young, and the community is not yet in consensus on its utility.

Conveniently, it provides a great mathematical representation for data structures. This is of great use to computer scientists, especially when working with expressive type systems. This notation allows us to easily compare different structures, reason about their possible contents, perform operations such as differentiation, and develop transformations between structures.

Consider a linked list, which can be represented as $L = 1 + X \bullet L$. This can be read as follows:

$$\underbrace{L}_{\text{A list}} \quad \underbrace{=}_{\text{is}} \quad \underbrace{1}_{\text{empty}} \quad \underbrace{+}_{\text{or}} \quad \underbrace{X}_{\text{an element}} \quad \underbrace{\bullet}_{\text{and}} \quad \underbrace{L}_{\text{another list}}$$

We can expand this recursive definition by replacement and algebraic manipulation to get an equivalent expression $L = 1 + X + X^2 + X^3 + \cdots$. This tells us that the list can have 0 elements, 1 element, 2 elements, and so on. This also shows us that a linked list is equivalent (more specifically, isomorphic) to an array-based list, which stores elements consecutively.

In addition to its possible utilities for combinatorics, this framework is invaluable for investigating the fundamental properties of data structures, and serves to better strengthen the understanding of types in computer science. Additionally, it has applications in automated testing, by providing a method to generate inputs to test on, either randomly or by exhaustion. [4]

2 Combinatorial Species

2.1 What is a species?

Informally, a species is "a family of structures parameterized by a set of labels which identify locations in the structures." [4] A species maps a given finite set of labels into a set of structures containing those labels. The species must be ignorant of the contents of the labels, and it must be possible to relabel without changing the structure itself.

Definition 1. A *species* F is a pair of mappings, $F_\bullet$ and $F_\leftrightarrow$, where

- $F_\bullet$ maps a finite set U to a finite set of structures, $F_\bullet[U]$, which can be "built from" the labels, commonly called the set of F-structures over U.
- $F_\leftrightarrow$ "lifts" a bijection $\sigma : U_1 \leftrightarrow U_2$ between two label sets onto a bijection between F-structures,

  $$F_\leftrightarrow[\sigma] : F_\bullet[U_1] \leftrightarrow F_\bullet[U_2]$$

  This mapping must be functorial. That is, it should satisfy

  - $F_\leftrightarrow[id] = id$
  - $F_\leftrightarrow[\sigma_1 \circ \sigma_2] = F_\leftrightarrow[\sigma_1] \circ F_\leftrightarrow[\sigma_2]$

  where id is the identity mapping and $\circ$ is composition.

[1, 4]

$F_\bullet$ is typically written as F for simplicity. Because $F_\leftrightarrow$ is functorial, the values of the labels are unimportant. The important property of the labels is that they are distinct, so that we can distinguish different locations of the structure.

A species corresponds to a data structure generic over a single unrestricted type parameter, such as Java's `LinkedList<T>`. However, the labels do not correspond to the data held by the data structure. Instead, they should be thought of as names for the locations within the structure.

2.2 Basic Species

Zero

The species 0 yields no structures no matter what labels it is given. This corresponds to a type which cannot be created. This is less commonly encountered in programming, but formally is the type of infinite loops or a function that terminates the program, and it can also be useful in generic contexts to eliminate the variant of a sum type.

One

The species 1 yields a single structure when given no labels. In other words, it holds no data, but nonetheless can be created. This corresponds to the return type `void` in Java, which can be thought of as nothing, or more precisely, a unit type that can always be created from nothing.

Singleton

The species X yields a single structure when given a single label. In other words, it holds exactly one piece of data. Conceptually, this corresponds to a type that is no different from the type it is holding.

2.3 Operators

Sum

Informally, species sum represents the word "or". We present a choice between two species. Species sum corresponds to a type sum or disjoint (tagged) union. Given species F and G, and the set of labels U, the structures of $F + G$ over U are each either an F-structure or a G-structure along with a tag that specifies which. Formally,

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

where $\uplus$ is disjoint union over sets. [4]

We can also generalize 0 and 1 to any positive natural number n by making n a species that takes no labels and produces n distinct structures. A simple way to define n is as the species sum of n total 1 species. As an example, a Boolean (with values true and false) would correspond to the species $1 + 1 = 2$.

Product

Informally, species product represents the word "and". We will take a structure from both species. The product of species corresponds to type product, or pairing. Given species F and G, and set of labels U, $(F \bullet G)[U]$ is the set of all pairs of structures with U split between F and G. That is,

$$(F \bullet G)[U] = \{\ (x, y) \mid U_1 \uplus U_2 = U,\ x \in F[U_1],\ y \in G[U_2]\ \}$$

A structure of $(F \bullet G)[U]$ will be a pair of structures, one from F and one from G, with the labels of U split between the pair.

2.4 Properties

Species addition and multiplication has the typical properties we expect of these operations. They are associative, commutative, and have identities of zero and one, respectively. Multiplication distributes over addition, and zero is an annihilator for multiplication. These properties are not proven here in interest of space.

2.5 Regular Species

A regular species can be defined equivalently by either its construction or its symmetry.

The species we have seen so far are all regular. By using using addition and multiplication, we can construct more regular species. We are also implicitly using the least fixed point operator μ when we refer to a species in its definition. For example, $L = 1 + X \bullet L$ would be written as $L = \mu\ell.1 + X \bullet \ell$ when using the least fixed point operator. These are equivalent due to the implicit species theorem, which is omitted here for space. [4]

Definition 2. A species is *regular* if it can be expressed in terms of 0, 1, X, $+$, $\bullet$, and μ (self-reference). [4]

A species is also regular if it has no non-trivial symmetries. In other words, all of its possible structures will change if the labels are rearranged.

Definition 3. A species F is *regular* if, for all structures $f \in F[U]$, the only permutation σ such that $F_{\leftrightarrow}[\sigma](f) = f$ is the identity permutation. [4]

This definition involving symmetry is often more useful for testing if a species defined by some idea is regular.

3 Irregular Species

3.1 Simple Irregular Species

Set

E is the species of a set, or unordered collection. For any set of labels, there will be exactly one structure, the set itself. E is irregular, since it has every possible symmetry. No matter what order the labels are in, we will get the same set.

Cycle

C is the species of a directed cycle. This is a nonempty loop that goes in a specific direction. C is irregular because any permutation that rotates the labels along the loop is symmetric.

3.2 Other Operations

Composition

Intuitively, species composition represents the word "of". That is, $F \circ G$ will be F-structures made of G-structures. To form a structure of $(F \circ G)[U]$, split U into non-empty partitions and send each of those partitions separately to G. Then, collect all those G-structures and send them to F.

For example, $L \circ E$ is a list of sets, and $R = 1 + X \bullet (L \circ R)$ is a rose tree, a tree where the nodes have an unspecified number of children.

Differentiation

Informally, taking the derivative of a species puts a single "hole" in the structures, a distinguished location not holding any data. That is,

$$F'[U] = F[U \cup \{*\}]$$

where $*$ is a label distinct from all elements of U. [4] Below are some examples of derivatives of common species.

- $L' = L^2$. If we put a hole in a list, we split it into two lists.
- $E' = E$. If we put a hole in a set, it does not change the set.
- $C' = L$. If we put a hole in a cycle, we break the loop into a list.

Species differentiation holds its expected properties compared to differentiation in calculus.

- $1' = 0$
- $X' = 1$
- $(F + G)' = F + G$
- $(F \bullet G)' = F \bullet G' + F' \bullet G$
- $(F \circ G)' = (F' \circ G) \bullet G'$

3.3 Compound Irregular Species

Powerset

The species $E \bullet E$ or E^2 corresponds to the powerset. Note that the labels given to this species are split between the two sets, so any given structure will consist of a subset of the labels and its complement.

Partition

The species $E \circ E_+$ is the set of nonempty sets, and thus corresponds to set partitioning.

Combination

Let E_m be E restricted to sets of size m. We can represent $\binom{n}{k}$ by the species $E_k \bullet E_{n-k}$

4 Generating Functions

A species F can be associated with an exponential generating function

$$F(x) = \sum_{n \geq 0} f_n \frac{x^n}{n!}$$

where f_n is the number of distinct labeled F-structures of size n. [4] Here are the functions corresponding to some species we have seen so far.

$$0(x) = \frac{0}{0!}x^0 + \frac{0}{0!}x^1 + \cdots = 0$$

$$1(x) = \frac{1}{0!}x^0 = 1$$

$$X(x) = \frac{1}{1!}x^1 = x$$

$$L(x) = \sum_{n \geq 0} \frac{n!}{n!}x^n = \frac{1}{1-x}$$

$$E(x) = \sum_{n \geq 0} \frac{1}{n!}x^n = e^x$$

$$C(x) = \sum_{n \geq 0} \frac{(n-1)!}{n!}x^n = -\log(1-x)$$

Note that the sum, product, differentiation, and composition of species corresponds to the same operations on the generating functions, e.g. $F + G$ is associated with $F(x) + G(x)$. We can confirm our earlier intuitions on the derivatives of species with the following remarks.

$$L'(x) = \frac{1}{(1-x)^2} = L(x)^2$$

$$E'(x) = e^x = E(x)$$

$$C'(x) = \frac{1}{1-x} = L(x)$$

Although the exponential generating function may be the most obvious and useful series, there are other power series that can be associated with species such as the type-generating series, cycle indicator series, and molecular series. These series allow us to demonstrate different properties and compare species in different ways, such as different kinds of equalities. For instance, two species may have the same exponential generating series, but a different type-generating series, helping us to see how they are distinguished. [1, 3]

5 Conclusion

Combinatorial species are an interesting young branch of combinatorics, with many extensions such as weighted, multi-sort, virtual, and molecular species. [4] Species provide computer scientists a great method for understanding data structures and can be applied to automated testing to help improve software correctness.

References

[1] François Bergeron et al. *Combinatorial species and tree-like structures.* Vol. 67. Cambridge University Press, 1998.

[2] André Joyal. "Une théorie combinatoire des séries formelles". In: *Advances in Mathematics* 42.1 (1981), pp. 1–82. DOI: `10.1016/0001-8708(81)90052-9`.

[3] Gilbert Labelle. "Some new computational methods in the theory of species". In: *Combinatoire énumérative.* Springer, 1986, pp. 192–209.

[4] Brent Yorgey. "Species and Functors and Types, Oh My!" In: vol. 45. Nov. 2010, pp. 147–158. DOI: `10.1145/2088456.1863542`.