



UNIVERSITAT DE  
BARCELONA

Facultat de Matemàtiques  
i Informàtica

GRADO EN MATEMÀTICAS

Trabajo final de grado

---

# El producto tensorial de conjuntos dendroidales

---

Autor: Roger Brascó Garcés

Director: Dr. Javier J. Gutiérrez

Realizado en: Departamento de Matemáticas  
e Informática

Barcelona, 24 de enero de 2022

## Resumen

The main goal of this project is to understand the tensor product operation in the category of dendroidal sets. To do so, we introduce the notion of shuffles between two trees, and show how they are used in order to describe the tensor product. We begin with a quick review about categories, functors and operads, focusing on the basic definitions and constructions. Then, we introduce the formalism of trees, its relation to coloured operads and the definition of the dendroidal category  $\Omega$ , which serves as the indexing category for defining dendroidal sets as a presheaf category. We also show how the dendroidal category  $\Omega$  extends the simplicial category  $\Delta$  via the inclusion of linear trees, and the relation between simplicial sets and dendroidal sets. Finally, we develop a Python algorithm that generates the complete set of shuffles between two trees and prints their planar representations.

## Agradecimientos

A todas las personas que me apoyaron e hicieron posible que este trabajo se realice con éxito. En especial a mi tutor por compartirme sus conocimientos. Y para recalcar, toda mi familia y amigos por acompañarme en este proceso.

# Contents

<b>1</b>	<b>Nociones previas</b>	<b>1</b>
1.1	Categorías . . . . .	1
1.1.1	Funtores . . . . .	2
1.2	Opéradas en conjuntos . . . . .	3
1.2.1	Opéradas coloreadas . . . . .	3
<b>2</b>	<b>Conjuntos Simpliciales</b>	<b>5</b>
2.1	Complejos simpliciales . . . . .	5
2.1.1	Morfismos simpliciales . . . . .	5
2.1.2	Complejos simpliciales ordenados y sus caras . . . . .	6
2.2	Conjuntos Delta . . . . .	6
2.2.1	Definición categórica de los conjuntos Delta . . . . .	7
2.3	Conjuntos simpliciales y sus morfismos . . . . .	7
2.3.1	Definición categórica de los conjuntos simpliciales . . . . .	8
<b>3</b>	<b>Conjuntos Dendroidales</b>	<b>10</b>
3.1	Árboles como opéradas . . . . .	10
3.1.1	Formalismo de árboles . . . . .	10
3.1.2	Árboles planares . . . . .	11
3.2	Morfismos en $\Omega_p$ . . . . .	12
3.2.1	Caras . . . . .	12
3.2.2	Degeneraciones . . . . .	13
3.2.3	Identidades dendroidales . . . . .	14
3.3	Árboles no planares . . . . .	16
3.3.1	Prehaz de estructuras planares . . . . .	18
3.4	Conjuntos Dendroidales . . . . .	18
3.5	Producto tensorial de conjuntos dendroidales . . . . .	21
3.5.1	Producto tensorial de Boardman–Vogt . . . . .	21
3.5.2	Producto tensorial de conjuntos dendroidales . . . . .	22
<b>4</b>	<b>Shuffles de árboles</b>	<b>24</b>
4.1	Producto tensorial de árboles lineales . . . . .	24
4.2	Producto tensorial de árboles . . . . .	25
4.2.1	Shuffles y conjuntos de shuffles . . . . .	25
4.2.2	Producto tensorial de árboles . . . . .	29
4.3	Shuffle de árboles en Python . . . . .	30



# 1 Nociones previas

En esta sección vamos a exponer unas nociones previas que nos van a ayudar durante todo el trabajo. Las definiciones las hemos extraído de [1, Capítulo 1].

## 1.1 Categorías

**Definición 1.1.** Una *categoría*  $\mathcal{C}$  consiste en:

- Una clase  $\text{Ob}(\mathcal{C})$ , cuyos elementos llamaremos *objetos* de la categoría.
- Para cada par de objetos  $A, B \in \text{Ob}(\mathcal{C})$  un conjunto  $\mathcal{C}(A, B)$  de *morfismos* o *flechas* de  $A$  a  $B$ .
- Para cada tres objetos  $A, B, C \in \text{Ob}(\mathcal{C})$  una *función de composición*

$$\mathcal{C}(B, C) \times \mathcal{C}(A, B) \xrightarrow{\circ} \mathcal{C}(A, C)$$

que envía el par  $(g, f)$  a  $g \circ f$ .

- Para cada objeto  $A$ , un elemento  $\text{id}_A \in \mathcal{C}(A, A)$  que llamaremos la *identidad* en  $A$ .

Además, esta estructura cumple los siguientes axiomas:

- *Asociatividad.* La función de composición es asociativa, esto es, dados  $f \in \mathcal{C}(A, B)$ ,  $g \in \mathcal{C}(B, C)$  y  $h \in \mathcal{C}(C, D)$ , se cumple que  $(h \circ g) \circ f = h \circ (g \circ f)$ .
- *Unidad.* La identidad es un elemento neutro para la composición, es decir, para toda  $f \in \mathcal{C}(A, B)$  tenemos que  $f \circ \text{id}_A = f = \text{id}_B \circ f$ .

A menudo, denotaremos un objeto  $A$  de  $\mathcal{C}$  como  $A \in \mathcal{C}$ , en vez de  $A \in \text{Ob}(\mathcal{C})$  y un morfismo  $f \in \mathcal{C}(A, B)$  como  $f: A \rightarrow B$ . Una categoría  $\mathcal{C}$  es *pequeña* si  $\text{Ob}(\mathcal{C})$  es un conjunto.

**Ejemplo 1.2.** Los siguientes son algunos ejemplos de categorías.

- (i) La categoría  $\text{Set}$  cuyos objetos son todos los conjuntos y cuyos morfismos son las aplicaciones entre conjuntos
- (ii) La categoría  $\text{Grp}$  cuyos objetos son los grupos y cuyos morfismos son los morfismos de grupo.
- (iii) La categoría  $\text{Top}$  cuyos objetos son los espacios topológicos y cuyos morfismos son las aplicaciones continuas.

**Definición 1.3.** Dada una categoría  $\mathcal{C}$ , podemos definir su *categoría opuesta*  $\mathcal{C}^{\text{op}}$  de la siguiente manera. Los objetos de  $\mathcal{C}^{\text{op}}$  son los mismos que los de  $\mathcal{C}$ , los morfismos cambian de dirección  $\mathcal{C}^{\text{op}}(A, B) = \mathcal{C}(B, A)$  y la función de composición es  $f \circ^{\text{op}} g = g \circ f$ .

### 1.1.1 Funtores

**Definición 1.4.** Sean  $\mathcal{C}$  y  $\mathcal{D}$  dos categorías. Un *functor*  $F$  de  $\mathcal{C}$  en  $\mathcal{D}$ , que denotaremos por  $F: \mathcal{C} \rightarrow \mathcal{D}$  consiste en:

- Una aplicación  $\text{Ob}(\mathcal{C}) \rightarrow \text{Ob}(\mathcal{D})$ . La imagen de un objeto  $A$  de  $\mathcal{C}$  la denotaremos por  $F(A)$
- Para cada par de objetos  $A, B \in \mathcal{C}$  una aplicación

$$\mathcal{C}(A, B) \longrightarrow \mathcal{D}(F(A), F(B)).$$

La imagen de un morfismo  $f: A \rightarrow B$  por esta aplicación la denotaremos por  $F(f): F(A) \rightarrow F(B)$ .

Además, estas aplicaciones son compatibles con la composición y la unidad, esto es, se cumplen los siguientes axiomas:

- Dados  $f \in \mathcal{C}(A, B)$  y  $g \in \mathcal{C}(B, C)$  se cumple que  $F(g \circ f) = F(g) \circ F(f)$ .
- Para todo objeto  $A \in \mathcal{C}$  se cumple que  $F(\text{id}_A) = \text{id}_{F(A)}$ .

**Observación 1.5.** La noción de functor que acabamos de llamar también *functor covariante* de  $\mathcal{C}$  en  $\mathcal{D}$ . Un functor de  $\mathcal{C}^{\text{op}}$  en  $\mathcal{D}$  se llama *functor contravariante* de  $\mathcal{C}$  en  $\mathcal{D}$ . Observar que si  $F$  es un functor contravariante de  $\mathcal{C}$  en  $\mathcal{D}$  y  $f: A \rightarrow B$  es un morfismo en  $\mathcal{C}$ , entonces  $F(f): F(B) \rightarrow F(A)$ .

**Ejemplo 1.6.** Dado un conjunto  $X$  cualquiera, podemos construir el grupo libre en los elementos de este conjunto  $F(X)$ . Esto define un functor  $F: \text{Set} \rightarrow \text{Grp}$ .

**Definición 1.7.** Sea  $F: \mathcal{C} \rightarrow \mathcal{D}$  un functor entre dos categorías  $\mathcal{C}$  y  $\mathcal{D}$ . Dados un par de objetos  $A, B \in \mathcal{C}$  consideremos la aplicación

$$F_{A,B}: \mathcal{C}(A, B) \longrightarrow \mathcal{D}(F(A), F(B)).$$

- Diremos que  $F$  es un functor *fiel* si para cada par de objetos  $A, B \in \mathcal{C}$  la aplicación  $F_{A,B}$  es inyectiva.
- Diremos que  $F$  es un functor *pleno* si para cada par de objetos  $A, B \in \mathcal{C}$  la aplicación  $F_{A,B}$  es exhaustiva.
- Diremos que  $F$  es un functor *plenamente fiel* si para cada par de objetos  $A, B \in \mathcal{C}$  la aplicación  $F_{A,B}$  es biyectiva.

## 1.2 Opéradas en conjuntos

Para cada  $n \geq 0$ , denotaremos por  $\Sigma_n$  el grupo simétrico de  $n$  letras (en el caso  $n = 0, 1$ ,  $\Sigma_n$  será el grupo trivial). Este apartado usa las definiciones de [3, Capítulo 1].

**Definición 1.8.** Una *opérada*  $P$  consiste en una sucesión de conjuntos  $\{P(n)\}_{n \geq 0}$  junto con la siguiente estructura:

- Un elemento *unidad*  $1 \in P(1)$ .
- Un *producto composición*

$$P(n) \times P(k_1) \times \cdots \times P(k_n) \longrightarrow P(k)$$

para cada  $n$  y  $k_1, \dots, k_n$  tal que  $k = \sum_{i=1}^n k_i$ .

- Para cada  $\sigma \in \Sigma_n$  una *acción por la derecha*  $\sigma^*: P(n) \rightarrow P(n)$ .

Además el producto composición es asociativo, equivariante y compatible con la unidad.

**Definición 1.9.** Dadas dos opéradas  $P$  y  $Q$ , un morfismo de opéradas  $f: P \rightarrow Q$  consiste en aplicaciones  $f_n: P(n) \rightarrow Q(n)$  para cada  $n \geq 0$  compatibles con el producto composición, la unidad y la acción del grupo simétrico.

### 1.2.1 Opéradas coloreadas

La noción de opérada coloreada generaliza a la vez el concepto de categoría y de opérada.

**Definición 1.10.** Sea  $C$  un conjunto, cuyos elementos llamaremos colores. Una opérada  $C$ -coloreada  $P$  consiste en, para cada  $(n+1)$ -tupla de colores  $(c_1, \dots, c_n, c)$  con  $n \geq 0$ , un conjunto  $P(c_1, \dots, c_n; c)$  (que representará el conjunto de operaciones cuyas entradas están coloreadas por los colores  $c_1, \dots, c_n$  y cuya salida esta coloreada por  $c$ ), junto con la siguiente estructura:

- Un elemento *unidad*  $1_c \in P(c; c)$  para cada  $c \in C$ .
- Un *producto composición*

$$\begin{aligned} P(c_1, \dots, c_n; c) \otimes P(d_{1,1}, \dots, d_{1,k_1}; c_1) \otimes \cdots \otimes P(d_{n,1}, \dots, d_{n,k_n}; c_n) \\ \longrightarrow P(d_{1,1}, \dots, d_{1,k_1}, \dots, d_{n,1}, \dots, d_{n,k_n}; c) \end{aligned}$$

para cada  $(n+1)$ -tupla de colores  $(c_1, \dots, c_n; c)$  y  $n$  tuplas cualesquiera

$$(d_{1,1}, \dots, d_{1,k_1}; c_1), \dots, (d_{n,1}, \dots, d_{n,k_n}; c_n)$$

- Para cada elemento  $\sigma \in \Sigma_n$  una *acción*

$$\sigma^*: P(c_1, \dots, c_n; c) \longrightarrow P(c_{\sigma(1)}, \dots, c_{\sigma(n)}; c).$$

Además el producto composición es asociativo, equivariante y compatible con las unidades.



**Definición 1.11.** Sea  $P$  una opéada  $C$ -coloreada y  $Q$  una opéada  $D$ -coloreada. Un morfismo de opéadas  $f: P \rightarrow Q$  consiste en una aplicación entre los conjuntos de colores  $f: C \rightarrow D$  y aplicaciones

$$f_{c_1, \dots, c_n; c}: P(c_1, \dots, c_n; c) \longrightarrow Q(f(c_1), \dots, f(c_n); c)$$

compatibles con el producto composición, las unidades y la acción del grupo simétrico.

Denotaremos por  $\text{Oper}$  la categoría cuyos objetos son operadas coloreadas y cuyos morfismos son los morfismos de operadas coloreadas.

*lógica*

**Ejemplo 1.12.** Si  $C = \{*\}$ , entonces una opéada  $C$ -coloreada es lo mismo que una opéada. Si  $P$  es una opéada  $C$ -coloreada tal que solamente tiene operaciones de aridad uno, es decir  $P(c_1, \dots, c_n; c) = \emptyset$  si  $n \neq 1$ , entonces  $P$  es una categoría, cuyo conjunto de objetos es  $C$ .

**Ejemplo 1.13.** Una categoría pequeña  $\mathcal{C}$  se puede ver como una opéada coloreada en  $\text{Set}$ . El conjunto de colores es el conjunto de objetos de  $\mathcal{C}$  y las operaciones son operaciones unitarias. La composición del producto viene determinado por la composición de morfismos en  $\mathcal{C}$ .

A la inversa, las operaciones unitarias en una opéada  $C$ -coloreada  $P$  dan una categoría cuyos objetos son elementos de  $C$ .

Estas dos construcciones definen los funtores adjuntos:

$$j_!: \text{Cat} \rightleftarrows j^*: \text{Oper}$$

donde  $\text{Cat}$  denota la categoría de categorías pequeñas.

## 2 Conjuntos Simpliciales

Los conjuntos simpliciales son esencialmente una generalización de los complejos simpliciales geométricos con una topología elemental. En esta sección daremos la base para llegar a entender como se forman los conjuntos simpliciales. Esta sección usa las definiciones de [2, Capítulo 2 y 3]

### 2.1 Complejos simpliciales

**Definición 2.1.** Un  $n$ -simplex geométrico es ~~una~~ <sup>la</sup> envoltura convexa de  $n + 1$  puntos geoméricamente independientes  $\{v_0, \dots, v_n\}$  en un espacio euclideo cualquiera. Es decir, colección de  $n$  vectores  $v_1 - v_0, \dots, v_n - v_0$  son linealmente independientes. Los puntos  $v_i$  los llamaremos *vértices*. <sup>toda cara de dimensión  $n$</sup>

Observamos que un  $n$ -simplex es homeomorfo a una esfera de  $n$  dimensiones.

**Definición 2.2.** Una *cara geométrica* de un  $n$ -simplex formado por los vértices  $\{v_0, \dots, v_n\}$  es la envoltura convexa formada por un subconjunto de dichos vértices.

**Definición 2.3.** Un *complejo simplicial geométrico*  $X$  en  $\mathbb{R}^n$  es una colección de simplices de varias dimensiones en  $\mathbb{R}^n$  tales que

- Toda cara de un simplex en  $X$  también está en  $X$ .
- La intersección de dos cualesquiera simplices de  $X$ , es una cara en ambos, si no es vacía.

Sea  $X$  un complejo simplicial geométrico, denotaremos por  $X^k$  al complejo simplicial geométrico formado por todos los  $k$ -simplex de  $X$ . Observamos que  $X^0 = \{v_i\}_{i \in I}$  donde  $I$  es un conjunto de índices. Entonces, podemos pensar ~~como~~ todo elemento de  $X^k$  como un subconjunto de  $X^0$  de cardinalidad  $k + 1$ . Es decir, un subconjunto  $\{v_{i_0}, \dots, v_{i_k}\} \subset X^0$  es un *elemento* de  $X^k$ . Para toda colección ~~contable~~ <sup>finita</sup> de vértices  $\{v_0, \dots, v_n\}$  que forma un simplex lo denotaremos como el simplex  $[v_0, \dots, v_n]$ .

**Definición 2.4.** Un *complejo simplicial abstracto*  $X$  es un conjunto de vértices  $X^0$  junto con los conjuntos  $X^k$  formados por ~~todos los~~ subconjuntos de  $X^0$  de cardinalidad  $k + 1$ , para todo  $k \in \mathbb{N}$ . Estos conjuntos deben cumplir que todo subconjunto de cardinalidad  $j + 1 \leq k$  de un elemento de  $X^k$  es un elemento de  $X^j$ . Es decir, ~~para~~ todo elemento de  $X^k$  es un  $k$ -simplex abstracto y ~~que~~ todas sus caras son simplices en  $X$ .

#### 2.1.1 Morfismos simpliciales

En este apartado definiremos ~~el morfismo~~ <sup>los morfismos</sup> entre dos complejos simpliciales geométricos. Este morfismo <sup>es</sup> será una herramienta clave para pasar de los complejos simpliciales a los conjuntos simpliciales.

**Definición 2.5.** <sup>Sean</sup> Sea  $K$  y  $L$  dos complejos simpliciales geométricos. Un *morfismo simplicial*  $f: K \rightarrow L$  los vertices  $\{v_i\}$  de  $K$  a los vertices  $\{f(v_i)\}$  de  $L$  de manera que si  $[v_{i_0}, \dots, v_{i_k}]$  es un simplex de  $K$ , entonces  $f(v_{i_0}), \dots, f(v_{i_k})$  son vértices (no todos únicos) de un simplex en  $L$ .   
<sup>envía</sup>  <sup>$\$K\$$</sup>   <sup>$\$L\$$</sup>

**Ejemplo 2.6.** Sea  $[v_0, v_1, v_2]$  un 2-simplex y  $[v_0, v_1]$  una de sus 1-cara. Consideramos el morfismo simplicial  $f: [v_0, v_1, v_2] \rightarrow [v_0, v_1]$  determinado por  $f(v_0) = v_0$ ,  $f(v_1) = v_1$ ,  $f(v_2) = v_1$ . Observamos en la siguiente figura que el morfismo colapsa el 2-simplex a un 1-simplex.

no hay figura!

Observamos que tal definición de morfismos simpliciales prevalece para los complejos simpliciales abstractos. De ahora en adelante simplemente usaremos el término complejo simplicial.

## 2.1.2 Complejos simpliciales ordenados y sus caras

**Definición 2.7.** Un *complejo simplicial ordenado*  $X$  es un complejo simplicial cuyo conjunto de vértices  $X^0$  está totalmente ordenado. Es decir, la notación  $[v_{i_0}, \dots, v_{i_k}]$  es un simplex si y solo si  $v_{i_j} < v_{i_l}$  para todo  $j < l$ .

**Definición 2.8.** Un *n-simplex ordenado* es un n-simplex con los vértices ordenados. Denotaremos el n-simplex ordenado por  $[\Delta^n]$ . Para simplificar, normalmente se renombran los vértices con los números  $0, 1, \dots, n$ , de tal manera que  $|\Delta^n| = [0, \dots, n]$ .

**Definición 2.9.** Sea  $X$  un complejo simplicial ordenado. Las *caras* son una colección de morfismos  $\delta_0, \dots, \delta_n: X^n \rightarrow X^{n-1}$  determinados por  $[0, \dots, n] \mapsto [0, \dots, \hat{j}, \dots, n]$ . Es decir, envían un n-simplex a su  $(n-1)$ -cara asociada al vértice  $j$ . Para complejos simpliciales ordenados en general,  $0 \leq j \leq n$

Simplex

$$\delta_j: X^n \longrightarrow X^{n-1}$$

$$[v_{i_0}, \dots, v_{i_n}] \longmapsto [v_{i_0}, \dots, \hat{v}_{i_j}, \dots, v_{i_n}]$$

Observamos que si  $i < j$ ,  $\delta_i \delta_j = \delta_{j-1} \delta_i$ . En efecto,  $\delta_i \delta_j [0, \dots, n] = [0, \dots, \hat{i}, \dots, \hat{j}, \dots, n] = \delta_{j-1} \delta_i [0, \dots, n]$

## 2.2 Conjuntos Delta

$\Delta$ -sets

intermedio

En este apartado veremos que los conjuntos Delta  $\Delta$ -sets son un intermedio entre complejos simpliciales y conjuntos simpliciales.

**Definición 2.10.** Un *conjunto Delta* consiste de una secuencia de conjuntos de  $i$ -simplices  $X_0, X_1, \dots, X_i, \dots$  y, para cada  $n \geq 0$ , las funciones  $\delta_i: X_{n+1} \rightarrow X_n$ ,  $\forall 0 \leq i \leq n+1$ , que cumplen  $\delta_i \delta_j = \delta_{j-1} \delta_i$ , si  $i \leq j$ . Formando el siguiente diagrama

$$\begin{array}{ccccc} & \delta_0 & & \delta_0 & \\ X_0 & \xleftarrow{\quad} & X_1 & \xleftarrow{\quad} & X_2 \dots \\ & \delta_1 & & \delta_1 & \\ & & & \delta_2 & \end{array}$$

Podemos representarlo mediante

### 2.2.1 Definición categórica de los ~~conjuntos~~ Delta

En este apartado introduciremos la definición de los ~~conjuntos~~ Delta como categorías cuyos objetos son simples y los morfismos son morfismos simpliciales.

**Definición 2.11.** La categoría  $\hat{\Delta}$  es una categoría cuyos objetos son los conjuntos estrictamente ordenados finitos  $[n] = \{0, \dots, n\}$  y los morfismos son las funciones, que mantienen el orden estrictamente,  $f: [m] \rightarrow [n]$ . Podemos pensar que sea la inclusión de un  $m$ -simplex como cara de un  $n$ -simplex. Para todo  $0 \leq i \leq n$  consideramos los morfismos:

$$d_i: [n] \longrightarrow [n+1] \\ \{0, \dots, n\} \longmapsto \{0, \dots, \hat{i}, \dots, n+1\}$$

**Definición 2.12.** La categoría  $\hat{\Delta}^{op}$ , es la categoría opuesta de  $\hat{\Delta}$ , cuyos objetos son los conjuntos estrictamente ordenados finitos  $[n] = \{0, \dots, n\}$  y los morfismos son las funciones, que mantienen el orden estrictamente,  $f: [n] \rightarrow [m]$ . Podemos pensar que sea la extracción de la cara  $m$ -simplex de un  $n$ -simplex. Para todo  $0 \leq i \leq n$  consideramos los morfismos:

$$D_i: [n] \longrightarrow [n-1] \\ \{0, \dots, n\} \longmapsto \{0, \dots, \hat{i}, \dots, n\}$$

**Definición 2.13.** Un conjunto Delta es un funtor covariante  $X: \hat{\Delta}^{op} \rightarrow \text{Set}$ , equivalentemente es un funtor contravariante  $X: \hat{\Delta} \rightarrow \text{Set}$ . Es decir, un funtor contravariante  $\hat{\Delta} \rightarrow \text{Set}$  asigna ~~un~~ objeto  $[n]$  de  $\hat{\Delta}$  a un conjunto de simplices  $X_n$  de  $\text{Set}$ , y asigna cada función que mantiene el orden estrictamente  $[m] \rightarrow [n]$  de  $\hat{\Delta}$  a una cara  $X_n \rightarrow X_m$ , haciendo una inclusión de una  $m$ -cara de cada simplex en  $X_n$  a un simplex de  $X_m$ .

**Ejemplo 2.14.** Sean  $[2]$  y  $[3]$  objetos de  $\hat{\Delta}$  y sea  $d_1: [2] \rightarrow [3]$  una función que mantiene el orden estrictamente, determinada por  $d_1(0) = 0$ ,  $d_1(1) = 2$  y  $d_1(2) = 3$ . Ahora, aplicando el funtor contravariante obtenemos los conjuntos de 2 y 3-simplices  $X_2$  y  $X_3$ , y la cara  $\delta_1: X_3 \rightarrow X_2$ .

## 2.3 Conjuntos simpliciales y sus morfismos

Antes de poder definir cómo se forman los conjuntos simpliciales, tenemos que introducir la noción de degeneraciones de simplices y sus degeneraciones como morfismos.

**Definición 2.15.** Un  $n$ -simplex degenerado es un  $n$ -simplex  $[v_0, \dots, v_n]$  cuyos vértices pueden estar repetidos, es decir, existe algún  $i$  y  $j$  tal que  $v_i = v_j$ . Por ejemplo, el simplex  $[0, 1, 1]$ .

**Definición 2.16.** Sea  $X$  un complejo simplicial ordenado. Las degeneraciones son una colección de morfismos  $\sigma_0, \dots, \sigma_n: X^n \rightarrow X^{n+1}$  determinados por  $[0, \dots, n] \mapsto [0, \dots, j, j, \dots, n]$ . Es decir, envían un  $n$ -simplejo a su  $(n+1)$ -simplejo degenerado asociado al vértice  $j$ . Para complejos simpliciales ordenados en general,  $0 \leq j \leq n$

$$\sigma_j: X^n \longrightarrow X^{n+1} \\ [v_{i_0}, \dots, v_{i_n}] \longmapsto [v_{i_0}, \dots, v_{i_j}, v_{i_j}, \dots, v_{i_n}]$$

Observamos que si  $i < j$ ,  $\sigma_i \sigma_j = \sigma_{j+1} \sigma_i$ . En efecto,

$$\sigma_i \sigma_j [0, \dots, n] = [0, \dots, i, i, \dots, j, j, \dots, n] = \sigma_{j+1} \sigma_i [0, \dots, n]$$

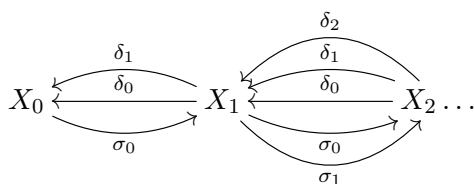
*Exposición*

**Definición 2.17.** Un conjunto simplicial es una ~~secuencia~~ *secuencia* de conjuntos de  $i$ -simplices  $X_0, X_1, \dots, X_i, \dots$ , y para cada  $n \geq 0$ , ~~las~~ funciones  $\delta_i : X_n \rightarrow X_{n-1}$  y  $\sigma_i : X_n \rightarrow X_{n+1}$ ,  $\forall 0 \leq i \leq n$ , que cumplen:

- (1)  $\delta_i \delta_j = \delta_{j-1} \delta_i, i < j$
- (2)  $\delta_i \sigma_j = \sigma_{j-1} \delta_i, i < j$
- (3)  $\delta_j \sigma_j = \delta_{j+1} \sigma_j = id$
- (4)  $\delta_i \sigma_j = \sigma_j \delta_{i-1}, i > j+1$
- (5)  $\sigma_i \sigma_j = \sigma_{j+1} \sigma_i, i \leq j$

*Los podemos representar mediante*

Formando el siguiente diagrama



### 2.3.1 Definición categórica de los conjuntos simpliciales

Como en los conjuntos Delta, daremos una definición categórica de los conjuntos simpliciales. Antes de ello definiremos la categoría  $\Delta$  y su opuesta.

**Definición 2.18.** La categoría  $\Delta$  es una categoría cuyos objetos son los conjuntos ordenados finitos  $[n] = \{0, \dots, n\}$  y los morfismos son las funciones, que mantienen solamente el orden,  $f : [m] \rightarrow [n]$ . Para todo  $0 \leq i \leq n$  consideramos los morfismos:

$$\begin{aligned} d_i : [n] &\longrightarrow [n+1] \\ \{0, \dots, n\} &\longmapsto \{0, \dots, \hat{i}, \dots, n+1\} \\ s_i : [n+1] &\longrightarrow [n] \\ \{0, \dots, n+1\} &\longmapsto \{0, \dots, i, \widehat{i+1}, \dots, n\} \end{aligned}$$

**Definición 2.19.** Categoría  $\hat{\Delta}^{op}$

Sea la categoría  $\Delta^{op}$ , la categoría opuesta de  $\Delta$ , cuyos objetos son los conjuntos ordenados finitos  $[n] = \{0, \dots, n\}$  y los morfismos son las funciones, que mantienen solamente el orden,  $f : [m] \rightarrow [n]$ . Para todo  $0 \leq i \leq n$  consideramos los morfismos:

$$\begin{aligned} D_i : [n] &\longrightarrow [n-1] \\ \{0, \dots, n\} &\longmapsto \{0, \dots, \hat{i}, \dots, n\} \\ S_i : [n] &\longrightarrow [n+1] \\ \{0, \dots, n\} &\longmapsto \{0, \dots, i, i, \dots, n\} \end{aligned}$$

Observamos que  $D_i$  y  $S_i$  corresponden a las caras y degeneraciones, respectivamente, y a su vez son los morfismos opuestos a  $d_i$  y a  $s_i$  que se encargan de incluir un  $n$ -simplex en un  $(n+1)$ -simplex como cara y de unir los vértices en las posiciones  $i$  y  $i+1$  de un  $(n+1)$ -simplex, respectivamente.

**Definición 2.20.** Un *conjunto simplicial* es un funtor covariante  $X: \Delta^{op} \rightarrow \text{Set}$ , equivalentemente es un funtor contravariante  $X: \Delta \rightarrow \text{Set}$ . Usaremos la notación  $\Delta[n] = \Delta(-, [n])$ . ✓

$$\begin{aligned}\Delta[n]: \Delta &\longrightarrow \text{Set} \\ [m] &\longmapsto \Delta([m], [n])\end{aligned}$$

**Ejemplo 2.21.** Vamos a calcular el conjunto simplicial  $\Delta[2] = \Delta(-, [2])$  2

- $\Delta([0], [2])$  tiene tres 0-simplices  $\{[0], [1], [2]\}$
- $\Delta([1], [2])$  tiene 6 1-simplices, tres degenerados generados por la degeneración  $\sigma_0$ ,  $\{[0, 0], [1, 1], [2, 2]\}$ ; y tres no degenerados  $\{[0, 1], [0, 2], [1, 2]\}$  que tienen las caras  $\delta_0$  y  $\delta_1$ , por ejemplo  $\delta_0([0, 1]) = [0]$  y  $\delta_1([0, 1]) = [1]$  que están en  $\Delta([0], [2])$ .
- $\Delta([2], [2])$  tiene 10 2-simplices, 9 degenerados generados por las degeneraciones  $\sigma_0$  y  $\sigma_1$ ,  $\{[0, 0, 0], [1, 1, 1], [2, 2, 2], [0, 0, 1], [0, 0, 2], [1, 1, 2], [0, 1, 1], [0, 2, 2], [1, 2, 2]\}$ ; y uno no degenerado  $\{[0, 1, 2]\}$  que tienen las caras  $\delta_0$ ,  $\delta_1$  y  $\delta_2$ , por ejemplo  $\delta_0([0, 1, 2]) = [1, 2]$ ,  $\delta_1([0, 1, 2]) = [0, 2]$  y  $\delta_2([0, 1, 2]) = [0, 1]$  que están en  $\Delta([1], [2])$  ✓
- $\Delta([3], [2])$ , y en adelante, tendrá todos degenerados.

### 3 Conjuntos Dendroidales

En esta sección vamos a introducir las nociones necesarias para poder describir los conjuntos dendroidales mediante el uso de árboles. Para ello tendremos que hablar sobre la formalización de árboles como opéradas coloreadas y los morfismos posibles entre árboles. Finalmente, podremos definir los conjuntos dendroidales como una categoría de prehaces.

#### 3.1 Árboles como opéradas

Antes de nada, vamos a definir una terminología para los árboles como opéradas coloreadas que la usaremos a lo largo del trabajo. Este apartado usa las definiciones de [3, Capítulo 2.1].

##### 3.1.1 Formalismo de árboles

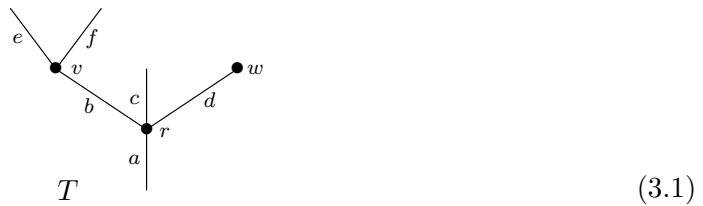
Un *árbol* es un grafo no vacío, finito, conexo y sin lazos. Llamaremos *vértices exteriores* a los vértices que tienen solamente una arista adyacente. Todos los árboles que consideraremos tendrán *raíz*, es decir, para cada árbol existe un vértice exterior, llamado *output* o *salida*. El conjunto de vértices exteriores restantes lo llamaremos *inputs* o *entradas*. Este último conjunto puede ser vacío y no contiene el vértice output.

Para dibujar dichos árboles, borraremos los vértices output e inputs de la figura. De tal manera que los vértices restantes serán los *vértices* del árbol. Dado un árbol  $T$ , definimos el conjunto de vértices como  $V(T)$  y el conjunto de aristas como  $E(T)$ .

Llamaremos *hojas* o *aristas externas* a las aristas adyacentes de los vértices inputs y *raíz* a la arista adyacente del vértice output. De tal manera que las aristas restantes las llamaremos *aristas internas*. Podemos observar que existe una dirección clara en cada árbol, desde las hojas hasta la raíz.

Sea  $v$  un vértice de un árbol finito con raíz, definimos  $\text{out}(v)$  como la única arista de salida y  $\text{in}(v)$  como el conjunto de aristas de entrada, observamos que este último conjunto puede ser vacío. Llamaremos la *valencia* de  $v$  a la cardinalidad del conjunto  $\text{in}(v)$ .

Finalmente, la siguiente figura es un árbol de ejemplo:



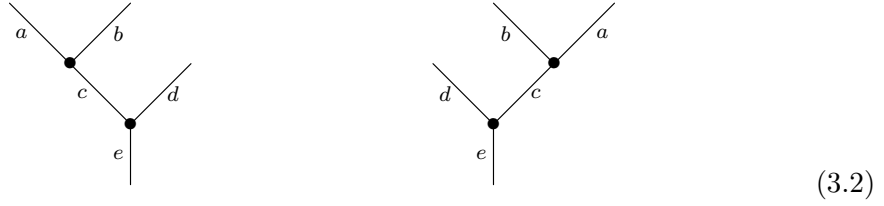
Observamos que hemos eliminado el vértice output de la arista  $a$  y los vértices inputs en las aristas  $e$ ,  $f$  y  $c$ . Este árbol tiene tres vértices  $r$ ,  $v$  y  $w$  con valencia 3, 2 y 0, respectivamente. Este árbol tiene tres hojas  $e$ ,  $f$  y  $c$ , y dos aristas internas  $b$  y  $d$ . Finalmente, la raíz es la arista  $a$ .

### 3.1.2 Árboles planares

Los árboles planares sirven para simplificar y poder definir los morfismos que vienen a continuación de manera más sencilla, pero todo lo que vamos a definir ahora será válido en el caso de árboles no planares.

**Definición 3.1.** Un *árbol planar con raíz* es un árbol con raíz  $T$  dotado con un orden lineal del conjunto  $\text{in}(v)$  para cada  $v$  de  $T$ .

**Observación 3.2.** El orden de los conjuntos  $\text{in}(v)$  se obtiene de la idea de dibujar los árboles en un plano. Es decir, para dibujar un árbol siempre pondremos la raíz debajo y las hojas arriba ordenando de izquierda a derecha. Observamos con esta técnica que tendremos varias representaciones planares del mismo árbol. Por ejemplo,

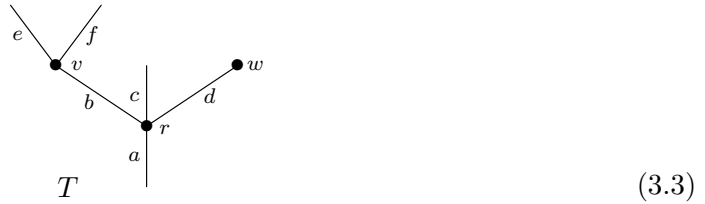


**Definición 3.3.** Denotaremos por  $\eta$  o *unitario* el árbol que tiene una única arista y ningún vértice.

**Definición 3.4.** Sea  $T$  un árbol planar con raíz. Denotaremos la opéada coloreada no-simétrica generada por  $T$  como  $\Omega_p(T)$ . El conjunto de colores de  $\Omega_p(T)$  es el conjunto de aristas  $E(T)$  de  $T$  y las operaciones están generadas por los vértices del árbol. Es decir, para cada vértice  $v$  con entradas  $e_1, \dots, e_n$  y salida  $e$ , definimos una operación  $v \in \Omega_p(T)(e_1, \dots, e_n; e)$ . Las otras operaciones son las operaciones unitarias y las operaciones obtenidas por composición.

**Observación 3.5.** Para todo  $e_1, \dots, e_n, e$ , el conjunto de operaciones  $\Omega_p(T)(e_1, \dots, e_n; e)$  contiene como mucho un solo elemento.

**Ejemplo 3.6.** Vamos a realizar la descripción completa de la opéada asociada al árbol  $T$ :



La operada  $\Omega_p(T)$  tiene seis colores  $a, b, c, d, e$ , y  $f$ . Las operaciones generadoras son  $v \in \Omega_p(T)(e, f; b)$ ,  $w \in \Omega_p(T)(-; d)$  y  $r \in \Omega_p(T)(b, c, d; a)$ . Mientras que las otras operaciones son las operaciones unitarias  $1_a, 1_b, \dots, 1_f$  y las operaciones composición  $r \circ_1 v \in \Omega_p(T)(e, f, c, d; a)$ ,  $r \circ_2 w \in \Omega_p(T)(b, c; a)$  y

$$(r \circ_1 v) \circ_3 w = (r \circ_2 w) \circ_1 v \in \Omega_p(T)(e, f, c; a)$$

**Definición 3.7.** La *categoría de árboles planares con raíz*  $\Omega_p$  es la subcategoría plena de la categoría de opéadas coloreadas no-simétricas cuyos objetos son  $\Omega_p(T)$  para cada árbol  $T$ .

Podemos pensar que  $\Omega_p$  es una categoría cuyos objetos son árboles planares con raíz. Sean  $S$  y  $T$  dos árboles planares con raíz, el conjunto de morfismos  $\Omega_p(S, T)$  es dado por los morfismos entre opéadas coloreadas no-simétricas de  $\Omega_p(S)$  a  $\Omega_p(T)$ .



**Observación 3.8.** La categoría  $\Omega_p$  extiende la categoría simplicial  $\Delta$ . Para todo  $n \geq 0$  se define un *árbol lineal*  $L_n$  como un árbol planar con  $n+1$  aristas y  $n$  vértices  $v_1, \dots, v_n$ , donde la valencia de todos los vértices es uno. Es decir, es un árbol cuyos vértices solo tienen una arista de entrada.



Denotaremos este árbol por  $[n]$ . Toda aplicación que mantiene el orden de manera que envíe  $\{0, \dots, n\}$  a  $\{0, \dots, m\}$ , define un morfismo  $[n] \rightarrow [m]$  en la categoría  $\Omega_p$ . De esta manera obtenemos un encaje

$$\Delta \xhookrightarrow{u} \Omega_p$$

Este encaje es un funtor plenamente fiel. Podemos observar que para toda flecha  $S \rightarrow T$  en  $\Omega_p$ , si  $T$  es lineal entonces  $S$  también lo es.

### 3.2 Morfismos en $\Omega_p$

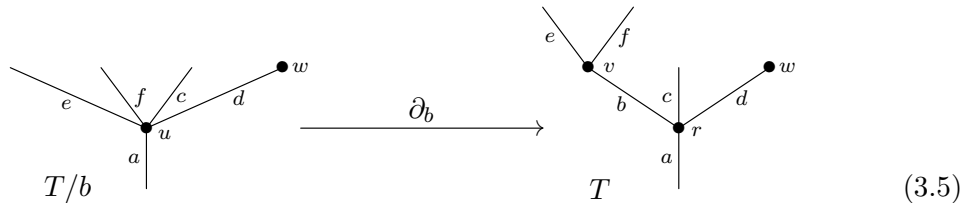
En los siguientes apartados vamos a tratar con todos los tipos de morfismos en  $\Omega_p$  y dar una descripción más explícita. Este apartado usa las definiciones de [3, Capítulo 2.2].

#### 3.2.1 Caras

Sea  $T$  un árbol planar con raíz.

**Definición 3.9.** Una *cara interna* asociada a una arista interna  $b$  en  $T$  es una función  $\partial_b: T/b \rightarrow T$  en  $\Omega_p$ , donde  $T/b$  es el árbol que se obtiene al contraer la arista  $b$  de  $T$ .

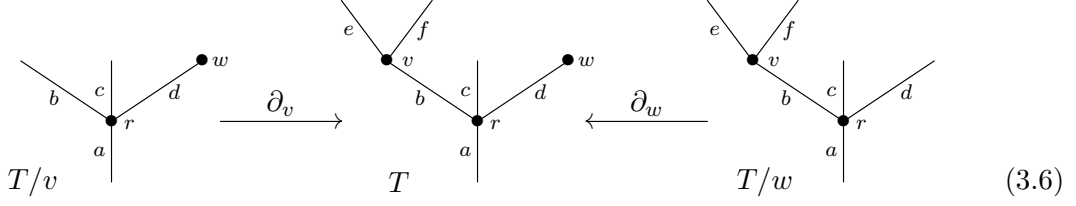
A nivel de opéradas, esta función es una inclusión de los colores y de las operaciones generadoras de  $\Omega_p(T/b)$ , excepto por la operación  $u$ , que se envía a la composición  $r \circ_b v$ , donde  $r$  y  $v$  son dos vértices en  $T$  con la arista  $b$  entre ellos, y  $u$  es el vértice correspondiente en  $T/b$ . Tomamos la siguiente figura para visualizar la función.



**Definición 3.10.** Una *cara externa* asociada a un vértice  $v$  en  $T$ , con solo una arista interna adyacente, es una función  $\partial_v: T/v \rightarrow T$  en  $\Omega_p$ , donde  $T/v$  es el árbol que se obtiene al cortar el vértice  $v$  de  $T$  con todas sus aristas externas.

A nivel de opéradas, esta función es una inclusión de los colores y de las operaciones generadoras de  $\Omega_p(T/v)$ , donde  $r$  y  $v$  son dos vértices en  $T$  con la arista  $b$  entre ellos, y  $u$

es el vértice correspondiente en  $T/b$ . Tenemos dos tipos de cara externa que mostramos en las siguientes figuras.



**Observación 3.11.** Con esta última definición no queda excluida la posibilidad de cortar la raíz. Esta situación solo será posible si la raíz tiene solamente una arista interna adyacente. Entonces, no todo árbol  $T$  tiene una cara externa asociada a su raíz.

**Observación 3.12.** Vale la pena mencionar un caso en especial, la inclusión del árbol sin vértices  $\eta$  en un árbol con un vértice, llamado *corola*. En este caso tendremos  $n + 1$  caras si la corola tiene  $n$  hojas. La opérada  $\Omega_p(\eta)$  consiste solamente de un color y la operación identidad de dicho color. Entonces, una función de opéradas  $\Omega_p(\eta) \rightarrow \Omega(T)$  es simplemente elegir un color de una corola  $T$ .

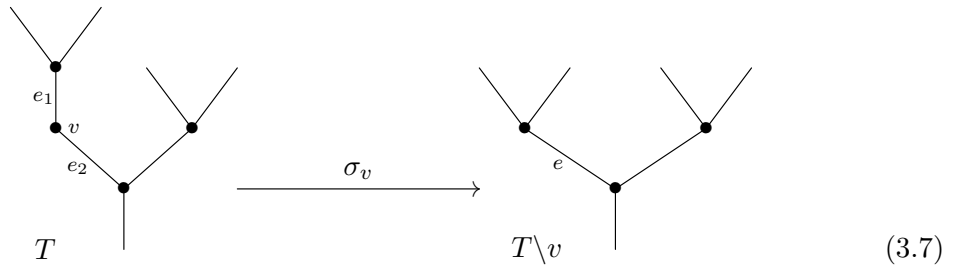
Para concluir, llamaremos *caras* tanto a las caras internas como a las caras externas.

### 3.2.2 Degeneraciones

Sea  $T$  un árbol planar con raíz y  $v$  un vértice de valencia uno en  $T$ .

**Definición 3.13.** Una *degeneración* asociada al vértice  $v$  es una función  $\sigma_v: T \rightarrow T \setminus v$  en  $\Omega_p$ , donde  $T \setminus v$  es el árbol que se obtiene al cortar el vértice  $v$  y juntar las dos aristas adjuntas en una nueva arista  $e$ .

A nivel de opéradas, la función envía los colores  $e_1$  y  $e_2$  de  $\Omega_p(T)$  al color  $e$  de  $\Omega_p(T \setminus v)$  y envía la operación generativa  $v$  a la operación identidad  $id_e$ , mientras que es la identidad para los colores y operaciones generativas restantes. Tomamos la siguiente figura para visualizar la función.



**Observación 3.14.** Las caras y las degeneraciones generan todos los morfismos de la categoría  $\Omega_p$ .

El siguiente lema es una generalización en  $\Omega_p$  del lema en la categoría  $\Delta$ , diciendo que toda flecha en dicha categoría se puede escribir como composición de degeneraciones seguidas por caras.

**Lema 3.15.** *Toda flecha  $f: S \rightarrow T$  en  $\Omega_p$  descompone, salvo isomorfismos, como*

$$\begin{array}{ccc} S & \xrightarrow{f} & T \\ & \searrow \sigma & \uparrow \partial \\ & & H \end{array}$$

donde  $\sigma: S \rightarrow H$  es una composición de degeneraciones y  $\partial: H \rightarrow T$  es una composición de caras.

*Proof.* Es una demostración análoga a la del lema 3.21. □

### 3.2.3 Identidades dendroidales

En este apartado vamos a dar las relaciones entre los morfismos generadores de  $\Omega_p$ . Las identidades que obtenemos generalizan las identidades simpliciales de la categoría  $\Delta$ .

#### Relaciones elementales de caras

Sea  $\partial_a: T/a \rightarrow T$  y  $\partial_b: T/b \rightarrow T$  dos caras internas distintas de  $T$ . Seguidamente tenemos las caras internas  $\partial_a: (T/b)/a \rightarrow T/b$  y  $\partial_b: (T/a)/b \rightarrow T/a$ . Observamos que  $(T/a)/b = (T/b)/a$ , entonces el siguiente diagrama conmuta:

$$\begin{array}{ccc} (T/a)/b & \xrightarrow{\partial_b} & T/a \\ \partial_a \downarrow & & \downarrow \partial_a \\ T/b & \xrightarrow{\partial_b} & T \end{array}$$

Mostramos esta relación mediante las siguientes figuras:

$$\begin{array}{ccccc} \begin{array}{c} \text{Diagram 1: } (T/a)/b \end{array} & \xrightarrow{\partial_b} & \begin{array}{c} \text{Diagram 2: } T/a \end{array} & \xrightarrow{\partial_a} & \begin{array}{c} \text{Diagram 3: } T \end{array} \\ (T/a)/b & & T/a & & T \end{array} \quad (3.8)$$

$$\begin{array}{ccccc} \begin{array}{c} \text{Diagram 4: } (T/b)/a \end{array} & \xrightarrow{\partial_a} & \begin{array}{c} \text{Diagram 5: } T/b \end{array} & \xrightarrow{\partial_b} & \begin{array}{c} \text{Diagram 6: } T \end{array} \\ (T/b)/a & & T/b & & T \end{array} \quad (3.9)$$

Sea  $\partial_v: T/v \rightarrow T$  y  $\partial_w: T/w \rightarrow T$  dos caras externas distintas de  $T$ , y asumimos que  $T$  tiene como mínimo tres vértices. Seguidamente tenemos las caras externas  $\partial_v: (T/w)/v \rightarrow T/w$  y  $\partial_w: (T/v)/w \rightarrow T/v$ . Observamos que  $(T/v)/w = (T/w)/v$ , entonces el siguiente diagrama conmuta:

$$\begin{array}{ccc} (T/v)/w & \xrightarrow{\partial_w} & T/v \\ \partial_v \downarrow & & \downarrow \partial_v \\ T/w & \xrightarrow{\partial_w} & T \end{array}$$

Mostramos esta relación mediante las siguientes figuras:

$$\begin{array}{ccccc}
 \begin{array}{c} \diagup \quad \diagdown \\ \bullet \\ | \\ (T/v)/w \end{array} & \xrightarrow{\partial_w} & \begin{array}{c} \diagup \quad \diagdown \\ \bullet \\ | \\ T/v \end{array} & \xrightarrow{\partial_v} & \begin{array}{c} \bullet \quad \bullet \\ \diagdown \quad \diagup \\ \bullet \\ | \\ T \end{array} \\
 & & & & (3.10)
 \end{array}$$

$$\begin{array}{ccccc}
 \begin{array}{c} \diagup \quad \diagdown \\ \bullet \\ | \\ (T/w)/v \end{array} & \xrightarrow{\partial_v} & \begin{array}{c} \bullet \quad \bullet \\ \diagdown \quad \diagup \\ \bullet \\ | \\ T/w \end{array} & \xrightarrow{\partial_w} & \begin{array}{c} \bullet \quad \bullet \\ \diagdown \quad \diagup \\ \bullet \\ | \\ T \end{array} \\
 & & & & (3.11)
 \end{array}$$

En el caso que  $T$  solo tenga dos vértices, existe un diagrama conmutativo similar mediante la inclusión de  $\eta$  a una  $n$ -corola. Existe un último caso para combinar una cara interna con una cara externa, y viceversa; así obteniendo un diagrama conmutativo similar, pero se debe tener cuenta dos condiciones excluyentes. Sean  $\partial_v: T/v \rightarrow T$  y  $\partial_e: T/e \rightarrow T$  una cara externa y una cara interna de  $T$ . Tenemos que la combinación de estas dos caras existen si:

- Si la arista  $e$  no es adyacente al vértice  $v$ .
- Si la arista  $e$  si es adyacente al vértice  $v$ , entonces existe otro vértice  $w$  adyacente a la arista  $e$ .

### Relaciones elementales de degeneraciones

Sea  $\sigma_v: T \rightarrow T \setminus v$  y  $\sigma_w: T \rightarrow T \setminus w$  dos degeneraciones distintas de  $T$ . Seguidamente tenemos las degeneraciones  $\sigma_v: T \setminus w \rightarrow (T \setminus w) \setminus v$  y  $\sigma_w: T \setminus v \rightarrow (T \setminus v) \setminus w$ . Observamos que  $(T \setminus v) \setminus w = (T \setminus w) \setminus v$ , entonces el siguiente diagrama conmuta:

$$\begin{array}{ccc}
 T & \xrightarrow{\sigma_w} & T \setminus w \\
 \sigma_v \downarrow & & \downarrow \sigma_v \\
 T \setminus v & \xrightarrow{\sigma_w} & (T \setminus v) \setminus w
 \end{array}$$

Mostramos esta relación mediante las siguientes figuras:

$$\begin{array}{ccccc}
 \begin{array}{c} \diagup \quad \diagdown \quad \diagup \quad \diagdown \\ \bullet \quad \bullet \\ | \quad | \\ v \quad w \\ \diagdown \quad \diagup \\ \bullet \\ | \\ T \end{array} & \xrightarrow{\sigma_v} & \begin{array}{c} \diagup \quad \diagdown \quad \diagup \quad \diagdown \\ \bullet \quad \bullet \\ | \quad | \\ e \quad w \\ \diagdown \quad \diagup \\ \bullet \\ | \\ T \setminus v \end{array} & \xrightarrow{\sigma_w} & \begin{array}{c} \diagup \quad \diagdown \quad \diagup \quad \diagdown \\ \bullet \quad \bullet \\ | \quad | \\ e \quad c \\ \diagdown \quad \diagup \\ \bullet \\ | \\ (T \setminus v) \setminus w \end{array} \\
 & & & & (3.12)
 \end{array}$$

$$\begin{array}{ccccc}
 \begin{array}{c} \diagup \quad \diagdown \quad \diagup \quad \diagdown \\ \bullet \quad \bullet \\ | \quad | \\ v \quad w \\ \diagdown \quad \diagup \\ \bullet \\ | \\ T \end{array} & \xrightarrow{\sigma_w} & \begin{array}{c} \diagup \quad \diagdown \quad \diagup \quad \diagdown \\ \bullet \quad \bullet \\ | \quad | \\ v \quad c \\ \diagdown \quad \diagup \\ \bullet \\ | \\ T \setminus w \end{array} & \xrightarrow{\sigma_v} & \begin{array}{c} \diagup \quad \diagdown \quad \diagup \quad \diagdown \\ \bullet \quad \bullet \\ | \quad | \\ e \quad c \\ \diagdown \quad \diagup \\ \bullet \\ | \\ (T \setminus w) \setminus v \end{array} \\
 & & & & (3.13)
 \end{array}$$

## Relaciones combinadas

Sea  $\sigma_v: T \rightarrow T \setminus v$  una degeneración y  $\partial: T' \rightarrow T$  es una cara de tal manera que la degeneración  $\sigma_v: T' \rightarrow T' \setminus v$  esta bien definida. Entonces existe una cara  $\partial: T' \setminus v \rightarrow T \setminus v$  determinada por el mismo vértice o arista que  $\partial: T' \rightarrow T$ . Además, el siguiente diagrama conmuta:

$$\begin{array}{ccc} T & \xrightarrow{\sigma_v} & T \setminus v \\ \partial \uparrow & & \uparrow \partial \\ T' & \xrightarrow{\sigma_v} & T' \setminus v \end{array}$$

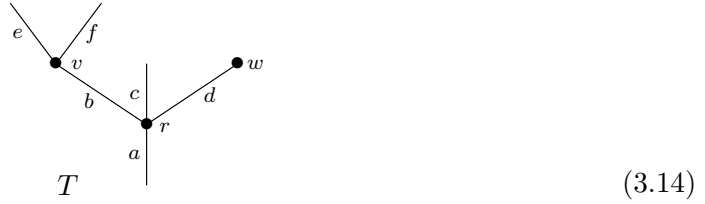
Sea  $\sigma_v: T \rightarrow T \setminus v$  una degeneración y  $\partial: T' \rightarrow T$  es una cara interna en una arista adyacente a  $v$  o una cara externa en  $v$ , si es posible. Entonces, tenemos que  $T' = T \setminus v$  y la composición  $T \setminus v \xrightarrow{\partial} T \xrightarrow{\sigma_v} T \setminus v$  es la función identidad  $id_{T \setminus v}$ .

### 3.3 Árboles no planares

Una vez dada la definición de los morfismos en  $\Omega_p$ , vamos a dar una generalización de dichos morfismos en  $\Omega$ . Este apartado usa las definiciones de [3, Capítulo 2.3].

**Definición 3.16.** Sea  $T$  un árbol no-planar. Denotaremos la opéada coloreada simétrica generada por  $T$  como  $\Omega(T)$ . El conjunto de colores de  $\Omega(T)$  es el conjunto de aristas  $E(T)$  de  $T$ . Las operaciones están generadas por los vértices del árbol, y el grupo simétrico de  $n$  letras  $\Sigma_n$  actúa en cada operación de  $n$  entradas permutando el orden de las entradas. Es decir, para cada vértice  $v$  con entradas  $e_1, \dots, e_n$  y salida  $e$ , definimos una operación  $v \in \Omega(T)(e_1, \dots, e_n; e)$ . Las otras operaciones son las operaciones unitarias, las operaciones obtenidas por composición y la acción del grupo simétrico.

**Ejemplo 3.17.** Consideramos la figura del siguiente árbol  $T$ :



La opéada  $\Omega(T)$  tiene seis colores  $a, b, c, d, e$ , y  $f$ . Las operaciones generadoras son las mismas que las operaciones generativas en  $\Omega_p(T)$ . Observamos que toda operación de  $\Omega_p(T)$  son operaciones de  $\Omega(T)$ , pero no a la inversa ya que hay más operaciones en  $\Omega(T)$  obtenidas por la acción del grupo simétrico. Por ejemplo, sea  $\sigma$  la transposición de dos elementos de  $\Sigma_2$ , entonces tenemos una operación  $v \circ \sigma \in \Omega(f, e; b)$ .

**Observación 3.18.** Sea  $T$  cualquier árbol, entonces  $\Omega(T) = \Sigma(\Omega_p(\bar{T}))$ , donde  $\bar{T}$  es una representación planar de  $T$  y  $\Sigma$  representa todas las acciones del grupo simétrico posibles aplicadas a las entradas de las operaciones. De hecho, se elige una estructura planar de  $T$  como generador de  $\Omega(T)$ .

**Definición 3.19.** La categoría de árboles con raíz  $\Omega$  es la subcategoría plena de la categoría de opéadas coloreadas cuyos objetos son  $\Omega(T)$  para todo árbol  $T$ .

Podemos pensar que  $\Omega$  es una categoría cuyos objetos son árboles con raíz. Sean  $S$  y  $T$  dos árboles con raíz, el conjunto de morfismos  $\Omega(S, T)$  es dado por los morfismos entre opéradas coloreadas de  $\Omega(S)$  a  $\Omega(T)$ .

**Observación 3.20.** Los morfismos de la categoría  $\Omega$  son generados por las caras y las degeneraciones, análogas al caso planar, y los isomorfismos no planares.

**Lema 3.21.** *Toda flecha  $f: S \rightarrow T$  en  $\Omega$  descompone como*

$$\begin{array}{ccc} S & \xrightarrow{f} & T \\ \sigma \downarrow & & \uparrow \partial \\ S' & \xrightarrow{\varphi} & T' \end{array}$$

donde  $\sigma: S \rightarrow S'$  es una composición de degeneraciones,  $\varphi: S' \rightarrow T'$  es isomorfismo, y  $\partial: T \rightarrow T'$  es una composición de caras.

*Proof.* Vamos a demostrarlo por inducción sobre el número de vértices de  $S$  y  $T$ . Si  $S$  y  $T$  no tienen vértices, entonces  $S = T = \eta$  y  $f$  es la identidad. Podemos asumir que  $f$  envía la raíz de  $S$  a la raíz de  $T$ ; de lo contrario podemos factorizar  $f$  como una función  $S \rightarrow T'$  que conserva la raíz seguido de otra función  $T' \rightarrow T$  que es una composición de caras externas. También podemos asumir que  $f$  es un epimorfismo en las hojas, de lo contrario podemos factorizar  $f$  como  $S \rightarrow T/v \xrightarrow{\partial_d} T$ , donde  $v$  es el vértice que está debajo de la hoja en  $T$  pero no está en la imagen de  $f$ .

Sean  $a$  y  $b$  son dos aristas de  $S$  tales que  $f(a) = f(b)$ , entonces  $a$  y  $b$  deberían estar en la misma rama de  $S$  y  $f$  envía los vértices intermedios a sus identidades.

Podemos factorizar  $f$  como una sobreyección seguido de una inyección en los colores, ya que  $f$  es una función de opéradas coloreadas. Esto corresponde a una factorización en  $\Omega$ ,

$$S \xrightarrow{\psi} S' \xrightarrow{\xi} T$$

donde  $\psi$  es una composición de degeneraciones y  $\xi$  es biyectiva en las hojas, envía la raíz de  $S'$  a la raíz de  $T$  y es inyectiva en los colores.

Si  $\xi$  es sobreyectiva en los colores, entonces  $\xi$  es un isomorfismo. Si  $\xi$  no es sobreyectiva, entonces existe una arista  $e$  de  $T$  que no está en la imagen de  $\xi$ . Como  $e$  debe ser una arista interna (no una hoja), podemos factorizar  $\xi$  como

$$S' \xrightarrow{\xi'} T/e \xrightarrow{\partial_e} T$$

Ahora continuamos por inducción sobre la función  $\xi'$ . □

### 3.3.1 Prehaz de estructuras planares

Para poder relacionar la categoría  $\Delta$  con las categorías  $\Omega_p$  y  $\Omega$ , necesitamos un morfismo que envía árboles a su conjunto de representaciones planares.

Sea  $P : \Omega^{\text{op}} \rightarrow \text{Set}$  el prehaz en  $\Omega$  que envía cada árbol a su conjunto de representaciones planares. Recordamos que la categoría  $\Omega \backslash P$  es la categoría cuyos objetos son pares  $(T, x)$  con  $x \in P(T)$ . Sean  $(T, x)$  y  $(S, y)$  dos objetos, un morfismo entre ellos es dado por un morfismo  $f : T \rightarrow S$  en  $\Omega$ , tal que  $P(f)(y) = x$ . Entonces, tenemos que  $\Omega \backslash P = \Omega_p$  y existe una proyección  $v : \Omega_p \rightarrow \Omega$ . Tenemos el siguiente triángulo conmutativo:

$$\begin{array}{ccc} \Delta & \xrightarrow{u} & \Omega_p \\ & \searrow i & \downarrow v \\ & & \Omega \end{array}$$

Donde  $i$  es un encaje plenamente fiel de  $\Delta$  en  $\Omega$ , que envía el objeto  $[n]$  de  $\Delta$  al árbol lineal  $L_n$  de  $\Omega$ , para todo  $n \geq 0$ .

Hemos podido ver que las dos categorías,  $\Omega_p$  y  $\Omega$ , extienden la categoría  $\Delta$ , gracias a ver los objetos de  $\Delta$  como árboles lineales. Además, se puede obtener  $\Delta$  como la categoría coma de  $\Omega_p$  o  $\Omega$ . Sea  $\eta$  un árbol en  $\Omega$  que no contiene ningún vértice y tan solo una arista, y sea  $\eta_p$  su representación planar en  $\Omega_p$ . Si  $T$  es un árbol cualquiera en  $\Omega$ , entonces  $\Omega(T, \eta)$  consiste en un solo morfismo o es el conjunto vacío, dependiendo si  $T$  es un árbol lineal o no. Pasa lo mismo con  $\Omega_p$  y  $\eta_p$ . Entonces,  $\Omega \backslash \eta = \Omega_p \backslash \eta_p = \Delta$ .

### 3.4 Conjuntos Dendroidales

En este apartado vamos a introducir nociones básicas y terminología para la categoría de los conjuntos dendroidales. Describiremos la categoría de los conjuntos dendroidales y los conjuntos dendroidales planares como categorías de prehaces en  $\Omega$  y  $\Omega_p$ , respectivamente. Hemos visto la relación entre estas categorías con la categoría de conjuntos simpliciales y la categoría de las opéradas, mediante una adjunción natural de funtores entre ellas. Más adelante definiremos un nervio dendroidal, desde opéradas hacia conjuntos dendroidales, generalizando así la construcción clásica del nervio, desde categorías pequeñas hacia conjuntos simpliciales. Este apartado usa las definiciones de [3, Capítulo 3.1].

**Definición 3.22.** La categoría  $dSets$  de *conjuntos dendroidales* es la categoría de prehaces en  $\Omega$ . Los objetos son funtores  $\Omega^{\text{op}} \rightarrow \text{Set}$  y los morfismos vienen dados por las transformaciones naturales. La categoría  $pdSet$  de *conjuntos dendroidales planares* esta definida de manera análoga intercambiando  $\Omega$  por  $\Omega_p$ .

Un conjunto dendroidal  $X$  viene definido como un conjunto  $X(T)$ , denotado por  $X_T$ , para cada árbol  $T$ , conjuntamente con una función  $\alpha^* : X_T \rightarrow X_S$  para cada morfismo  $\alpha : S \rightarrow T$  en  $\Omega$ . Como  $X$  es un funtor, entonces  $(id)^* = id$  y si  $\alpha : S \rightarrow T$  y  $\beta : R \rightarrow S$  son morfismos en  $\Omega$ , entonces  $(\alpha \circ \beta)^* = \beta^* \circ \alpha^*$ . El conjunto  $X_T$  lo llamaremos conjunto de *déndrices con forma T*, o simplemente conjunto de  $T$ -déndrices.

Sean  $X$  y  $Y$  dos conjuntos dendroidales, un *morfismo de conjuntos dendroidales*  $f : X \rightarrow Y$  viene definido por funciones  $f : X_T \rightarrow Y_T$ , para cada árbol  $T$ , conmutando con las funciones de estructura. Es decir, si  $\alpha : S \rightarrow T$  es cualquier morfismo en  $\Omega$  y  $x \in X_T$ , entonces  $f(\alpha^*x) = \alpha^*f(x)$ .

Decimos que  $Y$  es un *subconjunto dendroidal* de  $X$  si para cada árbol  $T$  tenemos que  $Y_T \subseteq X_T$  y la inclusión  $Y \hookrightarrow X$  es un morfismo de conjuntos dendroidales.

**Definición 3.23.** Un déndrice  $x \in X_T$  se llama *degenerado* si existe otro déndrice  $y \in X_S$  y una degeneración  $\sigma: T \rightarrow S$  tal que  $\sigma^*(y) = x$ .

Existen inclusiones canónicas y restricciones evidentes

$$\begin{array}{ccc} \Delta & \xrightarrow{u} & \Omega_p \\ & \searrow i & \downarrow v \\ & & \Omega \end{array} \qquad \begin{array}{ccc} sSets & \xleftarrow{u^*} & pdSets \\ & \nwarrow i^* & \uparrow v^* \\ & & dSets \end{array}$$

Donde todos tienen adjuntos por la derecha e izquierda

$$\begin{array}{ccc} sSets & \xrightleftharpoons[u^*]{u!} & pdSets \\ & \nwarrow i^* & \uparrow v^* \\ & & dSets \end{array} \qquad \begin{array}{ccc} sSets & \xrightleftharpoons[u^*]{u_*} & pdSets \\ & \nwarrow i_* & \uparrow v_* \\ & & dSets \end{array}$$

Que vienen dados por las extensiones de Kan correspondientes [5, Capítulo X, Apartado 3]. Por ejemplo, el funtor  $i^*$  envía un conjunto dendroidal  $X$  al conjunto simplicial

$$i^*(X)_n = X_{i([n])}$$

Su adjunto por la izquierda  $i_!: sSets \rightarrow dSets$  es la extensión por cero, y envía un conjunto simplicial  $X$  a un conjunto dendroidal dado por

$$i_!(X)_T = \begin{cases} X_n & \text{si } T \cong i([n]) \\ \emptyset & \text{si } T \not\cong i([n]) \end{cases}$$

Podemos ver que  $i_!$  es plenamente fiel y que  $i^*i_!$  es el funtor identidad en los conjuntos simpliciales.

Sea  $T$  un árbol. El  $T$ -déndrice estándar es el prehaz del representable  $\Omega(-, T)$ . Lo denotaremos por  $\Omega[T]$ .

El funtor  $\Omega \rightarrow Oper$  que envía un árbol  $T$  a la opéada coloreada  $\Omega(T)$  induce, por extensión de Kan, la siguiente adjunción

$$\tau_d: dSets \rightleftarrows Oper: N_d$$

El funtor  $N_d$  se llama *nervio dendroidal*. Para toda opéada  $P$ , su nervio dendroidal es el conjunto dendroidal

$$N_d(P)_T = Oper(\Omega(T), P)$$

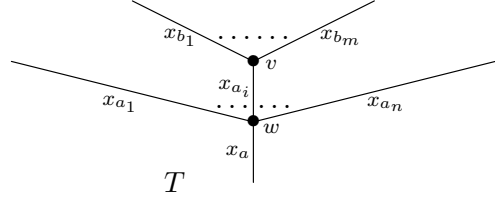
Este funtor es plenamente fiel y  $N_d(\Omega(T)) = \Omega[T]$  para cada árbol  $T$  en  $\Omega$ . También extiende el nervio de categorías a conjuntos simpliciales. Sea  $\mathcal{C}$  una categoría cualquiera y  $j_!(\mathcal{C})$  es la opéada coloreada asociada, entonces

$$i^*(N_d(j_!(\mathcal{C}))) = N(\mathcal{C})$$



Sea  $X$  un conjunto dendroidal. Nos referimos a la adjunción por la izquierda  $\tau_d(X)$  como la *opérada generada por  $X$* . Sabemos que el conjunto de colores de  $\tau_d(X)$  es igual que el conjunto  $X_\eta$ . Las operaciones de la opérada son generadas por los elementos de  $X_{C_n}$ , donde  $C_n$  es la  $n$ -ésima corola, con las siguientes relaciones:

- (i)  $s(x_a) = \text{id}_{x_a} \in \tau_d(X)(x_a; a_a)$  si  $x_a \in X_\eta$  y  $s = \sigma^*$ , donde  $\sigma$  es la degeneración  $\sigma: C_1 \rightarrow \eta$ .
- (ii) Si  $T$  es un árbol de la forma


(3.15)

y  $x \in X_T$ , entonces  $d_w(x) \circ_{x_{a_i}} d_v(x) = d_{x_{a_i}}(x)$ , donde

$$\begin{aligned} d_w(x) &\in \tau_d(X)(x_{a_1}, \dots, x_{a_n}; x_a) \\ d_v(x) &\in \tau_d(X)(x_{b_1}, \dots, x_{b_m}; x_{a_i}) \\ d_{x_{a_i}}(x) &\in \tau_d(X)(x_{a_1}, \dots, x_{a_{i-1}}, x_{b_1}, \dots, x_{b_m}, x_{a_{i+1}}, \dots, x_{a_n}; x_a) \end{aligned}$$

y  $d_w = \partial_w^*$  viene inducido por la cara asociada al vértice de la raíz  $w$ ;  $d_v = \partial_v^*$  viene inducido por la cara externa asociada al vértice  $v$ ; y  $d_{x_{a_i}} = \partial_{x_{a_i}}^*$  viene inducido por la cara interna asociada a la arista  $x_{a_i}$ .

Entonces,  $\tau_d(\Omega[T]) = \Omega(T)$  para todo árbol  $T$  en  $\Omega$ . También extiende el funtor  $\tau: sSets \rightarrow Cat$  adjunto al nervio simplicial, es decir, para todo conjunto simplicial  $X$

$$\tau(X) = j^* \tau_d(i_!(X))$$

En particular, tenemos el siguiente diagrama de funtores adjuntos

$$\begin{array}{ccc} sSets & \xrightleftharpoons[i^*]{i_!} & dSets \\ N \uparrow \downarrow \tau & & N_d \uparrow \downarrow \tau_d \\ Cat & \xrightleftharpoons[j^*]{j_!} & Oper \end{array}$$

Tenemos las siguientes relaciones conmutativas salvo isomorfismos:

$$\begin{aligned} \tau N &= \text{id}, \tau_d N_d = \text{id}, i^* i_! = \text{id}, j^* j_! = \text{id} \\ j_! \tau &= \tau_d i_!, N j^* = i^* N_d, i_! N = N_d j_! \end{aligned}$$

### 3.5 Producto tensorial de conjuntos dendroidales

En este apartado vamos a introducir el producto tensorial entre conjuntos dendroidales gracias al producto tensorial de Boardman–Vogt. Este apartado usa las definiciones de [3, Capítulo 4.1 y 4.2].

#### 3.5.1 Producto tensorial de Boardman–Vogt

**Definición 3.24.** Sea  $P$  una opéada simétrica  $C$ -coloreada, y sea  $Q$  una opéada simétrica  $D$ -coloreada. El *producto tensorial de Boardman–Vogt*  $P \otimes_{BV} Q$  es una opéada  $(C \times D)$ -coloreada definida en terminos de generadores y relaciones de la siguiente manera. Para cada color  $d \in D$  y cada operación  $p \in P(c_1, \dots, c_n; c)$  existe un generador

$$p \otimes d \in P \otimes_{BV} Q((c_1, d), \dots, (c_n, d); (c, d))$$

De manera análoga, para cada color  $c \in C$  y cada operación  $q \in Q(d_1, \dots, d_m; d)$  existe un generador

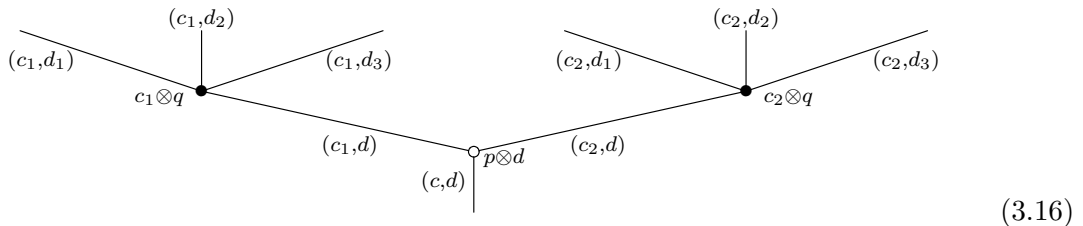
$$c \otimes q \in P \otimes_{BV} Q((c, d_1), \dots, (c, d_m); (c, d))$$

Estos generadores están sujetos a las siguientes relaciones:

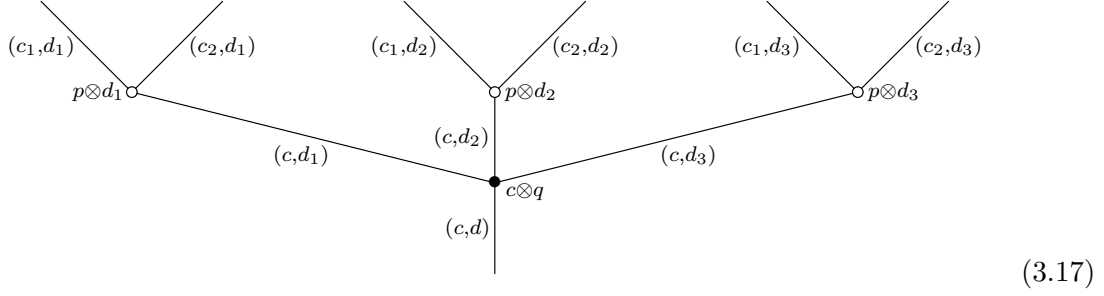
- (i)  $(p \otimes d) \circ ((p_1 \otimes d), \dots, (p_n \otimes d)) = (p \circ (p_1, \dots, p_n)) \otimes d$ .
- (ii)  $\sigma^*(p \otimes d) = (\sigma^*p) \otimes d$ , para cada  $\sigma \in \Sigma_n$ .
- (iii)  $(c \otimes q) \circ ((c \otimes q_1), \dots, (c \otimes q_m)) = c \otimes (q \circ (q_1, \dots, q_m))$ .
- (iv)  $\sigma^*(c \otimes q) = c \otimes (\sigma^*q)$ , para cada  $\sigma \in \Sigma_m$ .
- (v)  $\sigma_{n,m}^*((p \otimes d) \circ ((c_1 \otimes q), \dots, (c_n \otimes q))) = (c \otimes q) \circ ((p \otimes d_1), \dots, (p \otimes d_m))$ , donde  $\sigma_{n,m} \in \Sigma_{nm}$  es una permutación que describimos a continuación. Consideramos el conjunto  $\Sigma_{nm}$  como el conjunto de biyecciones del conjunto  $\{0, 1, \dots, nm-1\}$ . Cada elemento de dicho conjunto se puede escribir como  $kn + j$  de manera única para  $0 \leq k < m$  y  $0 \leq j < n$ ; y, análogamente, se puede escribir como  $km + j$  para  $0 \leq k < n$  y  $0 \leq j < m$ . Finalmente, la permutación  $\sigma_{n,m}$  la definimos de tal manera que  $\sigma_{n,m}(kn + j) = jm + k$ .

**Observación 3.25.** Tenemos que las relaciones (i) y (ii) implican que para cada color  $d \in D$  la función  $P \rightarrow P \otimes_{BV} Q$  es un morfismo de opéadas, que viene dada por  $p \mapsto p \otimes d$ . De manera análoga, tenemos que las relaciones (iii) y (iv) implican que para cada color  $c \in C$  la función  $Q \rightarrow P \otimes_{BV} Q$  es un morfismo de opéadas, que viene dada por  $q \mapsto c \otimes q$ .

**Ejemplo 3.26.** Vamos a ilustrar la relación (v), también llamada como la *relación de intercambio* con las siguientes figuras. Suponemos que  $n = 2$  y  $m = 3$ . Representamos mediante el siguiente árbol la operación de la izquierda de la relación (v), antes de aplicar la permutación  $\sigma_{2,3}^*$



Representamos mediante el siguiente árbol la operación de la derecha de la relación (v)



Observamos que la permutación  $\sigma_{2,3}$  corresponde a la permutación (2 4 5 3) de  $\Sigma_6$ . Hemos pintado los vértices de las operaciones en  $P$  de color blanco y los vértices de las operaciones en  $Q$  de color negro.

### 3.5.2 Producto tensorial de conjuntos dendroïdais

La categoría de los conjuntos dendroïdais es una categoría de prehaces, y por lo tanto cartesiana con el producto definido en cada nivel  $(X \times Y)_T = X_T \times Y_T$ . El producto cartesiano de los conjuntos dendroïdais extiende el producto cartesiano de conjuntos simpliciales, es decir, para cada par de conjuntos simpliciales  $X$  e  $Y$ , tenemos que

$$i_!(X \times Y) \cong i_!(X) \times i_!(Y).$$

**Definición 3.27.** Para todo par de árboles  $T$  y  $S$  en  $\Omega$ , el *producto tensorial* de los representables  $\Omega[T]$  y  $\Omega[S]$  se define como

$$\Omega[T] \otimes \Omega[S] = N_d(\Omega(T) \otimes_{BV} \Omega(S))$$

Donde  $N_d$  es el nervio dendroïdal,  $\Omega(T)$  y  $\Omega(S)$  son las opéradas coloreadas asociadas a los árboles  $T$  y  $S$ , respectivamente; y  $\otimes_{BV}$  es el producto tensorial de Boardman–Vogt.

Esto define un producto tensorial en toda la categoría de conjuntos dendroïdais, ya que es una categoría de prehaces y entonces cada objeto es un colímite canónico de representables y  $\otimes$  conserva colímites en cada variable.

**Definición 3.28.** Sean  $X$  e  $Y$  dos conjuntos dendroïdais y sea  $X = \lim_{\rightarrow} \Omega[T]$  y  $Y = \lim_{\rightarrow} \Omega[S]$  sus expresiones canónicas como colímites de representables. Entonces, definimos el *producto tensorial*  $X \otimes Y$  como

$$X \otimes Y = \lim_{\rightarrow} \Omega[T] \otimes \lim_{\rightarrow} \Omega[S] = \lim_{\rightarrow} N_d(\Omega(T) \otimes_{BV} \Omega(S))$$

Sabemos que este producto tensorial es cerrado gracias a la teoría general de categorías [6], y el conjunto de  $T$ -déndrices del hom interno viene definido por

$$\text{Hom}_{dSets}(X, Y)_T = dSets(\Omega[T] \otimes X, Y)$$

Para cada par  $X$  e  $Y$  de conjuntos dendroïdais y para cada árbol  $T$  en  $\Omega$ .

**Teorema 3.29.** La categoría de conjuntos dendroïdais admite una estructura monoidal, simétrica y cerrada. Esta estructura monoidal está únicamente determinada (salvo isomorfismo) por la propiedad de que existe un isomorfismo natural

$$\Omega[T] \otimes \Omega[S] \cong N_d(\Omega(T) \otimes_{BV} \Omega(S))$$

para cada par  $T$  y  $S$  de objetos de  $\Omega$ . La unidad del producto tensorial es el conjunto dendroïdal representable  $\Omega[\eta] = i_!(\Delta[0]) = U$ .

**Proposición 3.30.** *Tenemos las siguientes propiedades:*

(i) *Para cada par  $X$  e  $Y$  de conjuntos simpliciales, existe un isomorfismo natural*

$$i_!(X) \otimes i_!(Y) \cong i_!(X \times Y)$$

(ii) *Para cada par  $X$  e  $Y$  de conjuntos simpliciales, existe un isomorfismo natural*

$$\tau_d(X \otimes Y) \cong \tau_d(X) \otimes_{BV} \tau_d(Y)$$

(iii) *Para cada par  $P$  e  $Q$  de opéradas coloreadas, existe un isomorfismo natural*

$$\tau_d(N_d(P) \otimes N_d(Q)) \cong P \otimes_{BV} Q$$

*Proof.* (i) Basta con ver que la propiedad se mantiene en los representables en  $sSets$ . Si vemos  $[n]$  y  $[m]$  de  $\Delta$  como categorías, entonces tenemos

$$j_!([n] \times [m]) \cong j_!([n]) \otimes_{BV} j_!([m])$$

Entonces tenemos la siguiente cadena de isomorfismos naturales

$$\begin{aligned} i_!(\Delta[n] \times \Delta[m]) &\cong i_!(N([n]) \times N([m])) \cong i_!(N([n] \times [m])) \\ &\cong N_d(j_!([n] \times [m])) \cong N_d(j_!([n]) \otimes_{BV} j_!([m])) \\ &\cong N_d(\Omega(L_n) \otimes_{BV} \Omega(L_m)) \cong \Omega[L_n] \otimes \Omega[L_m] \\ &\cong i_!(\Delta[n]) \otimes i_!(\Delta[m]) \end{aligned}$$

Donde  $L_n$  y  $L_m$  son dos árboles lineales con  $n$  y  $m$  vértices, y  $n + 1$  y  $m + 1$  aristas; respectivamente.

(ii) Basta con ver que la propiedad se mantiene en los representables en  $dSets$ . Tenemos la siguiente cadena de isomorfismos naturales, usando el isomorfismo natural  $\tau_d N_d \cong \text{id}$

$$\begin{aligned} \tau_d(\Omega[T] \otimes \Omega[S]) &\cong \tau_d N_d((\Omega(T) \otimes_{BV} \Omega(S))) \cong \Omega(T) \otimes_{BV} \Omega(S) \\ &\cong \tau_d(\Omega[T]) \otimes_{BV} \tau_d(\Omega[S]) \end{aligned}$$

(iii) Análogamente siguiendo (ii) pero remplazando  $X$  por  $N_d(P)$  y  $Y$  por  $N_d(Y)$ .  $\square$

## 4 Shuffles de árboles

En esta sección describiremos el producto tensorial  $\Omega[S] \otimes \Omega[T]$  para todo par de árboles  $S$  y  $T$  de  $\Omega$ . Para ello deberemos introducir la noción de conjunto de shuffles de  $S$  y  $T$ , que será una gran ayuda para encontrar el producto tensorial. Así podremos entender qué es el producto tensorial de conjuntos dendroidales. Esta sección usa las definiciones y ejemplos de [3, Capítulo 4.3].

### 4.1 Producto tensorial de árboles lineales

Antes de ver el producto tensorial para árboles en general, estudiaremos el caso de árboles lineales ya que resulta una tarea más sencilla y la podemos relacionar con el producto cartesiano de conjuntos simpliciales.

Sean  $S = L_n$  y  $T = L_m$  dos árboles lineales, entonces por la Proposición 3.30(i),

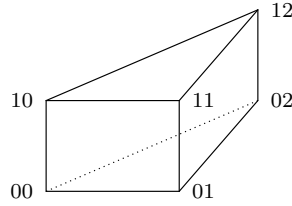
$$\Omega[L_n] \otimes \Omega[L_m] = i_!(\Delta[n]) \otimes i_!(\Delta[m]) \cong i_!(\Delta[n] \times \Delta[m])$$

Los simplices no degenerados del producto de dos representables en conjuntos simpliciales se calculan mediante un *shuffle*. Un  $(n, m)$ -*shuffle* es un camino de longitud máxima en el conjunto parcialmente ordenado  $[n] \times [m]$ . Los  $(n + m)$ -simplices no degenerados de  $\Delta[n] \times \Delta[m]$  corresponden a los  $(n, m)$ -shuffles. De hecho,

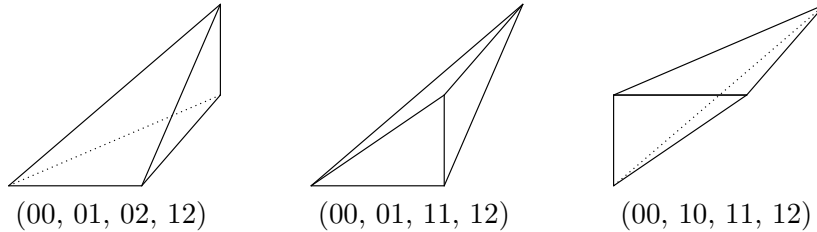
$$\Delta[n] \times \Delta[m] = \bigcup_{(n, m)} \Delta[n + m]$$

Donde la unión recorre todos los posibles  $(n, m)$ -shuffles.

**Ejemplo 4.1.** Sean  $n = 2$  y  $m = 1$ . Existen tres  $(2, 1)$ -shuffles en  $[2] \times [1]$ ,  $(00, 01, 02, 12)$ ,  $(00, 01, 11, 12)$  y  $(00, 10, 11, 12)$ . Tenemos la siguiente figura que representa  $\Delta[2] \times \Delta[1]$



Podemos ver que cada  $(2, 1)$ -shuffle corresponde a un tetraedro, y nos dan una descomposición de  $\Delta[2] \times \Delta[1]$  como la unión de tres copias de  $\Delta[3]$ :

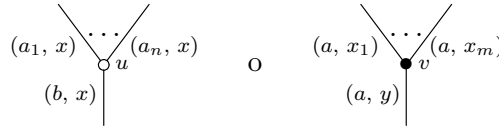


## 4.2 Producto tensorial de árboles

### 4.2.1 Shuffles y conjuntos de shuffles

En este apartado vamos a introducir la noción de un shuffle entre dos árboles y la colección de todos ellos. También daremos ejemplos extensos de como calcular dichos shuffles.

**Definición 4.2.** Sea  $S$  y  $T$  dos objetos de  $\Omega$ . Un *shuffle* de  $S$  y  $T$  es un árbol  $R$  cuyo conjunto de aristas es un subconjunto de  $E(S) \times E(T)$ . La raíz de  $R$  es  $(a, x)$ , donde  $a$  es la raíz de  $S$  y  $x$  es la raíz de  $T$ , y sus hojas son todos los pares  $(l_S, l_T)$ , donde  $l_S$  es una hoja de  $S$  y  $l_T$  es una hoja de  $T$ . Los vértices son de la forma



Donde  $u$  es un vértice de  $S$  con entradas  $a_1, \dots, a_n$  y salida  $b$ , y  $v$  es un vértice de  $T$  con entradas  $x_1, \dots, x_m$  y salida  $y$ . Nos referiremos a los dos tipos de vértices como *vértices blancos* y *vértices negros*, respectivamente. Para diferenciarlos visualmente los pintaremos con  $\circ$  y  $\bullet$ , respectivamente.

Observamos que existe una biyección entre los shuffles de dos árboles lineales  $L_n$  y  $L_m$  con los  $(n, m)$ -shuffles de  $[n] \times [m]$ .

**Definición 4.3.** Sean  $S$  y  $T$  dos árboles. El *conjunto de shuffles de  $S$  y  $T$*  es la colección de todos los shuffles posibles entre  $S$  y  $T$ . La cardinalidad de este conjunto la denotaremos por  $sh(S, T)$ .

**Proposición 4.4.** El número de shuffles  $sh(S, T)$  de dos árboles  $S$  y  $T$  satisface tres propiedades:

- (i) *Simétrico:*  $sh(S, T) = sh(T, S)$
- (ii) *Unitario:* Si  $T$  es un árbol unitario  $\eta$ , entonces  $sh(S, \eta) = 1$
- (iii) *Inducción:* Si  $S = C_n[S_1, \dots, S_n]$  y  $T = C_m[T_1, \dots, T_m]$ , entonces

$$sh(S, T) = \prod_{i=1}^n sh(S_i, T) + \prod_{j=1}^m sh(S, T_j),$$

donde  $C_n$  y  $C_m$  son  $n$  y  $m$ -corolas, respectivamente; y  $C_n[S_1, \dots, S_n]$  es una  $n$ -corola que cada hoja  $i$ -ésima la conectamos con la raíz del árbol  $S_i$ . [4, Proposición 3.1].

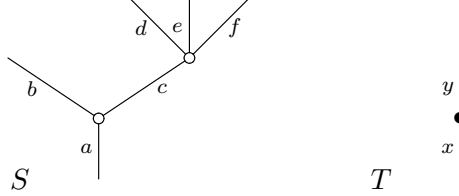
*Proof.* Las propiedades (i) y (ii) son obvias. La propiedad (iii) viene de que el conjunto de nombres de las hojas de un shuffle del tipo  $(s, t)$ , donde  $s$  y  $t$  son aristas de  $S$  y  $T$  respectivamente, es el producto cartesiano de las hojas de  $S$  y  $T$ .  $\square$

Observamos que el número de shuffles entre dos árboles cualesquiera augmenta mucho según el tamaño de los árboles y también parece difícil encontrar una fórmula cerrada que de una aproximación precisa del número, excepto en árboles especiales. [4, Observación 3.2].

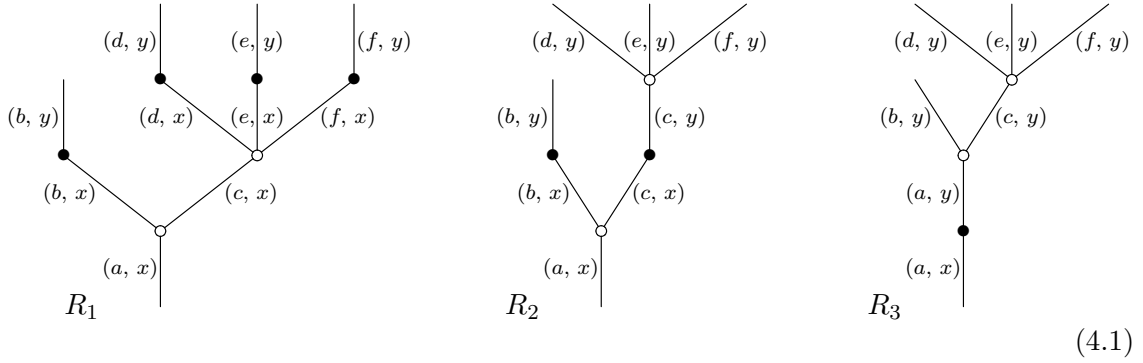
*En general es un problema abierto encontrar esta fórmula cerrada*

Sean  $S = L_m$  y  $T = L_n$  árboles lineales con  $m$  y  $n$  vértices, respectivamente. Denotamos  $\lambda(m, n)$  por  $sh(L_m, L_n)$ . De la Proposición 4.4 tenemos que  $\lambda(m, 0) = 1 = \lambda(0, n)$  y  $\lambda(m, n) = \lambda(m-1, n) + \lambda(m, n-1)$ , que es la relación inductiva que define el coeficiente binomial  $\binom{m+n}{n} = \binom{m-1+n}{n} + \binom{m+n-1}{n-1}$ . Entonces, el número de shuffles para árboles lineales es  $sh(L_m, L_n) = \binom{m+n}{n}$ .

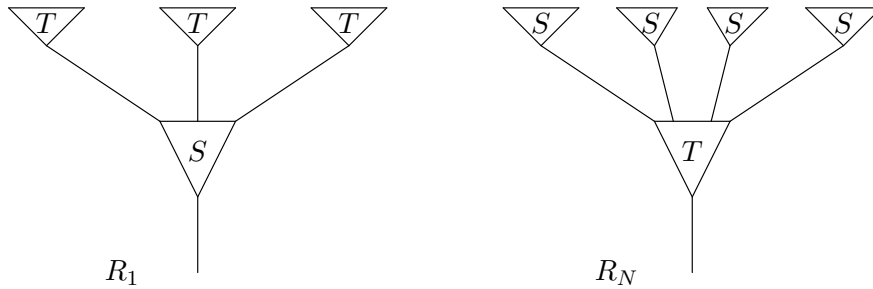
**Ejemplo 4.5.** Sean  $S$  y  $T$  los árboles



El conjunto de shuffles de  $S$  y  $T$  consiste de los siguientes tres árboles:

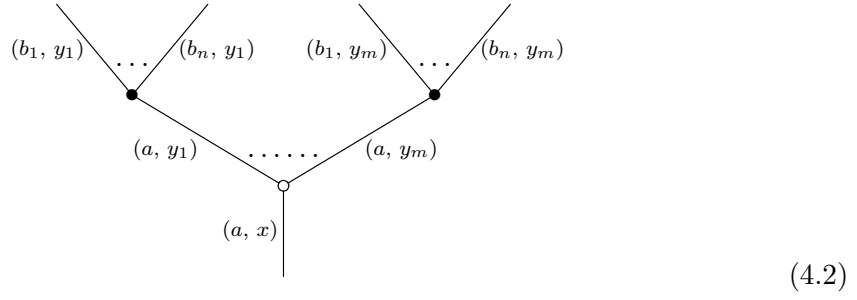


El conjunto de shuffles de  $S$  y  $T$  está *parcialmente ordenado*. El árbol minimal  $R_1$  en el conjunto parcialmente ordenado se obtiene mediante la inserción de una copia del árbol negro  $T$  en cada entrada del árbol blanco  $S$ . Es decir, primero hacemos una copia del árbol  $S$  de la forma  $S \otimes r_T$ , donde todas sus aristas han sido renombradas como  $(\_, r_T)$ , siendo  $r_T$  la raíz del árbol  $T$ . Luego hacemos una copia del árbol  $T$  de la forma  $l \otimes T$ , para toda hoja  $l$  de  $S$ ; donde todas sus aristas han sido renombradas como  $(l, \_)$ . Finalmente, obtenemos el árbol  $R_1$  encajando las últimas copias encima de las hojas de la forma  $(l, r_T)$  de la primera copia. El árbol maximal  $R_N$  en el conjunto parcialmente ordenado se obtiene mediante la inserción de una copia del árbol blanco  $S$  en cada entrada del árbol negro  $T$ . Los árboles  $R_1$  y  $R_n$  deberían lucir de la siguiente manera

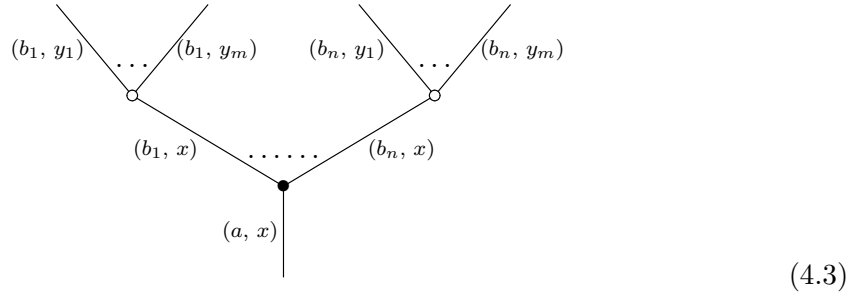


Existen los *shuffles intermedios*  $R_k$  ( $1 < k < N$ ) entre  $R_1$  y  $R_N$  obtenidos filtrando los vértices negros en  $R_1$  hacia la raíz del árbol mediante intercambios con los vértices blancos. Todo  $R_k$  se obtiene desde un  $R_l$  anterior. Es decir, cada intercambio se basa en

transformar una configuración de  $R_l$

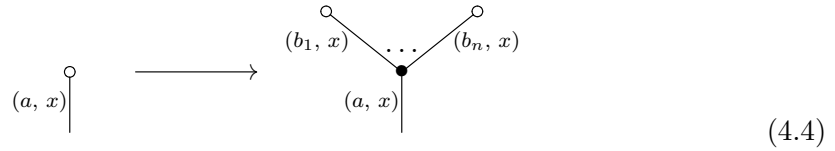


A una configuración de  $R_k$

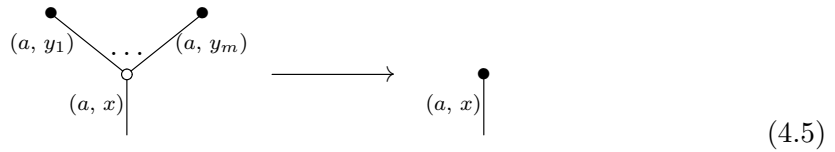


Si un shuffle  $R_k$  se obtiene the otro shuffle  $R_l$  mediante la norma de arriba, entonces decimos que  $R_k$  se obtiene mediante *un solo intercambio* y lo denotaremos por  $R_l \leq R_k$ . Así, obtenemos un orden parcial en el conjunto de todos los shuffles.

Tenemos que especificar el caso de un intercambio con un árboles sin entradas, es decir,  $n = 0$  o  $m = 0$ . Si  $m = 0$  y  $n \neq 0$ , entonces tenemos el intercambio



Si  $n = 0$  y  $m \neq 0$ , entonces tenemos el intercambio



Finalmente, si  $n = m = 0$ , entonces tenemos el intercambio

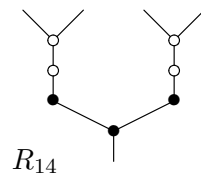
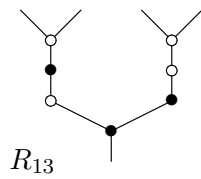
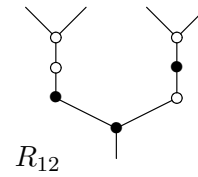
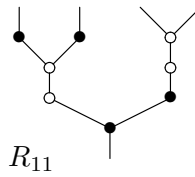
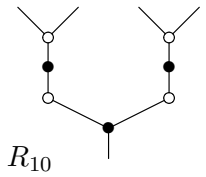
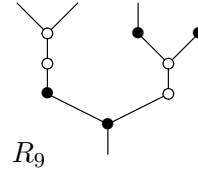
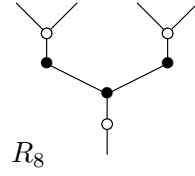
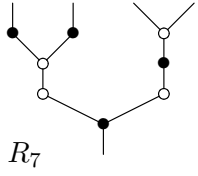
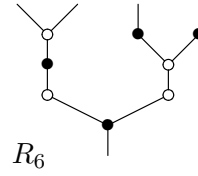
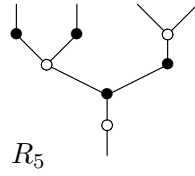
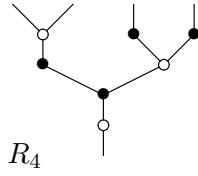
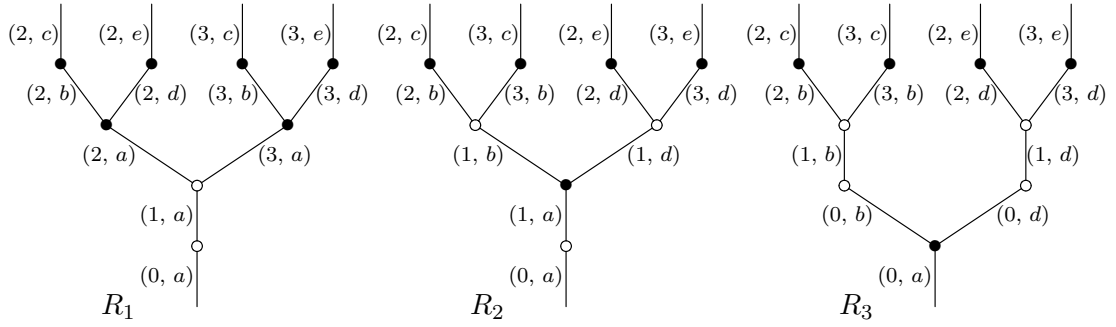




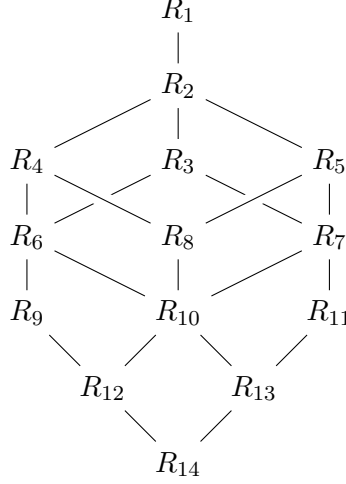
**Ejemplo 4.6.** Sean  $S$  y  $T$  los árboles



Existen catorce shuffles  $R_1, \dots, R_{14}$  de  $S$  y  $T$ . Mostramos una lista completa de ellos. Marcaremos los nombres de las aristas en los tres primeros shuffles.



Tenemos la siguiente estructura dentro del conjunto parcialmente ordenado.



#### 4.2.2 Producto tensorial de árboles

Ahora podemos dar una descripción completa del producto tensorial entre dos objetos representables en  $\Omega$  mediante el cálculo de su conjunto de shuffles.

**Lema 4.7.** *Para todo shuffle  $R_i$  de  $S$  y  $T$  tenemos un monomorfismo*

$$m: \Omega[R_i] \hookrightarrow \Omega[S] \otimes \Omega[T]$$

*El subconjunto dendroidal, que viene dado por la imagen de este monomorfismo, lo denotaremos  $m(R_i)$ .*

*Proof.* Los vértices del conjunto dendroidal  $\Omega[R_i]$  son las aristas del árbol  $R_i$ . La función  $m$  envía aristas nombradas como  $(a, x)$  en  $R_i$  a la arista con el mismo nombre en  $\Omega[S] \otimes \Omega[T]$ . Un morfismo  $\Omega[R] \rightarrow X$ , de un representable de un conjunto dendroidal a un conjunto dendroidal arbitrario, es un monomorfismo desde que el morfismo  $\Omega[R]_\eta \rightarrow X_\eta$  lo es en los vértices.

□

**Corolario 4.8.** *Para todo objeto  $T$  y  $S$  en  $\Omega$ , tenemos que*

$$\Omega[S] \otimes \Omega[T] = \bigcup_{i=1}^N m(R_i)$$

*donde la unión recorre todos los posibles shuffles de  $S$  y  $T$ .*

### 4.3 Shuffle de árboles en Python

No es complicado ver que tanto encontrar el producto tensorial de conjuntos dendroidales o, equivalentemente, calcular el conjunto de shuffles para dos árboles cualesquiera, resulta una tarea tediosa si los árboles son grandes. Para tal problema, el uso de un programa informático, capaz de almanezar grandes cantidades de información al momento de ejecución, nos resulta cómodo, fácil y rápido.

En este apartado vamos a describir de manera breve el código del paquete que hemos desarrollado para poder tratar con opéradas, árboles, shuffles y finalmente con el conjunto de shuffles. También, el código incluye una función para formar figuras con el paquete *xypic* de LaTeX de un árbol mediante una descripción completa.

Finalmente, dicho código se puede encontrar tanto en el Anexo B como en el repositorio público de código de Github: Trees Shuffling. Hace falta comentar que habrán diferencias entre el código completo que podréis encontrar en el anexo y los pseudocódigos usados a continuación, ya que nos hemos quedado con la estructura fundamental del algoritmo para facilitar la lectura.

#### Clases del paquete

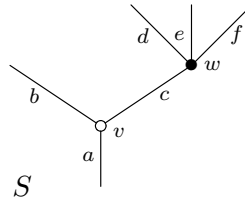
**Definición 4.9.** Una *clase* es una abstracción de propiedades y funciones de un objeto en concreto. Siguiendo tal definición, tenemos las siguientes clases en nuestro paquete:

- *Operad*: Espacio que guarda los colores y las operaciones que forman un árbol.
- *Tree*: Clase abstracta que define un árbol con propiedades tipo: raíz, hojas...
- *TreeMerger*: Clase para juntar dos árboles  $S$  y  $T$ , uno encima del otro.
- *TreeManipulator*: Clase para buscar y hacer intercambios.
- *ShuffleLattice*: Clase para generar todos los shuffles entre dos árboles  $S$  y  $T$ .

#### Utilidades

Antes de describir las clases que forman el paquete, debemos comentar que existe un fichero llamado *utils.py* donde hay una colección de funciones y algoritmos útiles que usaremos a lo largo del código.

La función *string\_to\_tree\_space* servirá como la puerta de entrada a los árboles. Tiene como entrada una descripción completa de un árbol y devuelve una instancia de la clase *Tree*. Por ejemplo, la cadena de caracteres " $vW(b, c; a)|wB(d, e, f; c)$ " describe el árbol  $S$



donde el vértice  $v$  de color blanco ( $W$ ) tiene como entradas  $b$  y  $c$ , y salida  $a$ ; y el vértice  $w$  de color negro ( $B$ ) tiene como entradas  $d$ ,  $e$  y  $f$ , y salida  $c$ . Durante la explicación vamos llamar *operaciones* a " $vW(b, c; a)$ " o " $wB(d, e, f; c)$ ".

El algoritmo *sorted* servirá para ordenar las operaciones que generan un árbol. El orden que vamos a describir nos asegura que todo color de salida en cada operación, aparezca como entrada de otra operación en la izquierda, salvo el color correspondiente a la raíz ya que esa operación en concreto será la primera. Seguidamente podemos ver el pseudocódigo.

---

**Algoritmo 1** Pseudocódigo del algoritmo para ordenar las operaciones de un árbol  $T$ . Podéis encontrar el código completo en el anexo.

---

**Input:** Requires an array of operations of a tree

```

1: function SORTED(operations)
2:    $n \leftarrow$  Length of operations
3:    $i \leftarrow 0$  ▷ Starting index
4:   while  $i < n$  do
5:     for  $j$  in  $[i + 1, \dots, n)$  do
6:       if operations[ $i$ ].trunk in operations[ $j$ ].branches then
7:         operation  $\leftarrow$  Pop operation on index  $i$  from operations
8:         operations.insert(operation, j) ▷ Insert operation on new index
9:         break
10:      else ▷ Note that this else only occurs when the for hasn't been broken
11:         $i \leftarrow i + 1$ 
12:   return operations

```

**Output:** Returns the list of operations sorted

---

Por ejemplo, todas las siguientes descripciones hablan de un mismo árbol  $S$ :

1. "2W(3; 2)|1W(2, 4, 6; 1)|3W(5; 4)|4W(7; 6)|0W(1; 0)"
2. "1W(2, 4, 6; 1)|2W(3; 2)|3W(5; 4)|0W(1; 0)|4W(7; 6)"
3. "3W(5; 4)|2W(3; 2)|4W(7; 6)|0W(1; 0)|1W(2, 4, 6; 1)"

Si aplicamos el algoritmo *sorted* a estas entradas, obtendremos la siguiente cadena de caracteres "0W(1; 0)|1W(2, 4, 6; 1)|4W(7; 6)|3W(5; 4)|2W(3; 2)" que describe el árbol  $S$  de manera estándar.

## Clase *Tree*

La clase *Tree* tiene las propiedades que define un árbol como una operada coloreada.

- *trunk*: Color de salida.
- *branches*: Lista de colores de entrada.
- *node*: Nombre de la operación.

Por ejemplo, sea  $S = "vW(b, c; a)"$ , entonces el tronco (*trunk*) es el color  $a$ , las ramas (*branches*) son los colores  $b$  y  $c$ ; y el *node* es el nombre  $vW$ .

Sea  $S$  un árbol con más de una operación, cada operación será una instancia de *Tree* y estarán relacionadas entre si mediante los colores de las ramas (*branches*) y los troncos (*trunk*). Es decir, sea  $S = "vW(b, c; a)|wB(d, e, f; c)"$ , la rama  $c$  de la operación " $vW(b, c; a)$ " esta relacionada con el tronco  $c$  de la operación " $wB(d, e, f; c)$ ".

De esta manera tendremos una estructura de árbol en forma de grafo, que podremos explotar con los algoritmos que vienen a continuación. Y además, tendremos todo el árbol a mano mediante la operación que contienen la raíz. Falta mencionar que usaremos la clase *uTree* que describe un árbol sin *node* ni *branches*.

### Clase *TreeMerger*

Esta clase esta formada de unos algoritmos, que no describiremos, para poder unir dos árboles de la misma manera descrita en el apartado 4.2.1. Es decir, esta clase genera el shuffle  $R_1$  si usamos los dos árboles  $S$  y  $T$  de entrada; y a la inversa, genera el shuffle  $R_N$  si usamos los árboles  $T$  y  $S$  de entrada.

### Clase *TreeManipulator*

Esta clase es importante para la clase que explicaremos a continuación *ShuffleLattice*, ya que será su alimento. Es decir, esta clase nos busca en un shuffle  $R$  los vértices disponibles para realizar un intercambio y nos realiza tal intercambio.

El algoritmo de búsqueda se llama *find\_percolations*. Es el encargado de buscar los vértices disponibles para realizar un intercambio en un shuffle  $R$  cualquiera, de la misma manera descrita en el apartado 4.2.1. Es un algoritmo recursivo donde va acumulando las operaciones donde se encuentran los intercambios posibles. Seguidamente podemos ver el pseudocódigo.

---

**Algoritmo 2** Pseudocódigo del algoritmo para encontrar los vértices para hacer un intercambio de un shuffle  $R$ . Podéis encontrar el código completo en el anexo.

---

**Input:** Requires a *Tree*  $R$  and an optional array *found*

```

1: function FIND_PERCOLATIONS( $R$ , found)      ▷ Note that  $R$  is the rooted operation
2:   if found is None then
3:     found  $\leftarrow$  Initialize empty array
4:   if  $R$ .node is from  $S$ .operations then
5:     for branch in  $R$ .branches do                                ▷ Note that branch is a Tree
6:       if branch.node not in  $T$ .operations then
7:         break
8:       else                                ▷ Note that this else only occurs when the for hasn't been broken
9:         found  $\leftarrow$  Append  $R$ 
10:    for  $R'$  in  $R$ .branches do
11:      find_percolations( $R'$ , found)                                ▷ Note that  $R'$  is a Tree
12:    return found

```

**Output:** Returns the list *found* of locations

---

La función de acción se llama *make\_percolation*. Es la encargada de realizar el intercambio, previamente encontrado por el algoritmo *find\_percolations*. Es decir, realiza un cambio de operaciones y un cambio de los nombres de las aristas afectadas siguiendo la norma de intercambios.

## Clase *ShuffleLattice*

Finalmente, la clase *ShuffleLattice* es la clase más importante del paquete que usa directamente o indirectamente todas las clases que hemos comentado anteriormente. Esta clase genera todos los shuffles  $R_i$  entre dos árboles cualesquiera  $S$  y  $T$ .

La clase tiene una propiedad llamada *dictionary* donde vamos acumulando todos los shuffles encontrados. También tiene una propiedad llamada *skeleton* donde guardamos las relaciones entre shuffles  $R_l < R_k$  y así poder tener la estructura del conjunto de shuffles parcialmente ordenado.

La clase se basa en la ejecución del algoritmo *generate\_shuffle*, que básicamente es el algoritmo clásico de búsqueda en anchura, más conocido como *BFS* en inglés. Es decir, es un algoritmo de búsqueda no informada que se alimenta con nuestro algoritmo *find\_percolations* y actúa con la función *make\_percolation*. Seguidamente podemos ver el pseudocódigo.

---

**Algoritmo 3** Pseudocódigo del algoritmo para generar todos los shuffles entre  $S$  y  $T$ . Podéis encontrar el código completo en el anexo.

---

**Input:** Requires the class instance of *ShuffleLattice*

```
1: function GENERATE_SHUFFLES(self)
2:   queue  $\leftarrow$  Append  $R_1$  ▷ Note that  $R_1$  is the first shuffle
3:   self.dictionary  $\leftarrow$  Save initial shuffle
4:   while queue not empty do
5:     shuffle  $\leftarrow$  Pop the first shuffle in queue
6:     for location in find_percolations(shuffle) do
7:       new_shuffle  $\leftarrow$  Apply make_percolation on location
8:       if new_shuffle not in self.dictionary then
9:         queue  $\leftarrow$  new_shuffle Append new shuffle
10:      self.dictionary  $\leftarrow$  Save new shuffle
```

**Output:** The algorithm does not return anything because the shuffles have been stored on the class property *dictionary*

---

Finalmente para esta clase, comentaremos el algoritmo *sh* que usa la Proposición 4.4. Este algoritmo devuelve el número de shuffles entre dos árboles  $S$  y  $T$ . Seguidamente podemos ver el pseudocódigo.

---

**Algoritmo 4** Pseudocódigo del algoritmo para computar el número de shuffles entre  $S$  y  $T$ . Podéis encontrar el código completo en el anexo.

---

**Input:** Requires two trees  $S$  and  $T$

```
1: function SH( $S, T$ )
2:   if  $S$  is a unitary Tree or  $T$  is a unitary Tree then return 1
3:   prodS  $\leftarrow$  1 & prodT  $\leftarrow$  1
4:   for branch in S.get_branches() do
5:     prodS  $\leftarrow$  prodS * sh(branch,  $T$ ) ▷ Note that every branch is a Tree
6:   for branch in T.get_branches() do
7:     prodT  $\leftarrow$  prodT * sh( $S$ , branch) ▷ Note that every branch is a Tree
8:   return prodS + prodT
```

**Output:** Returns the sum of the products

---

**Observación 4.10.** Para usar el paquete tendremos que cortar los vértices que no tienen entradas, ya que simplifica el código de buscar los intercambios y hacerlos. Además, tener o no dichos vértices no varía la cardinalidad del conjunto de shuffles. Por ejemplo,

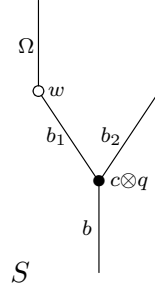


### Figuras en LaTeX

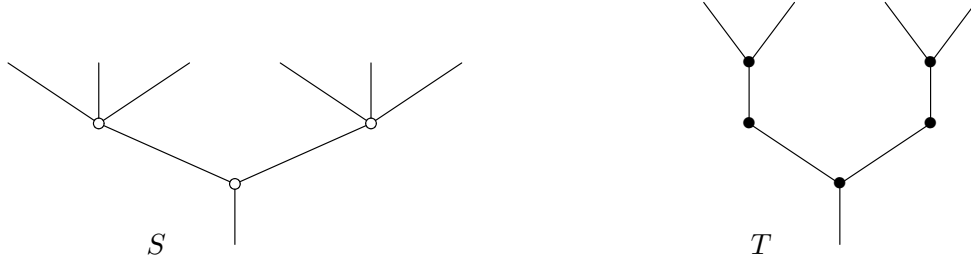
Como se ha podido observar, es necesario usar representaciones gráficas de los árboles durante todo el trabajo, ya que facilita entender como actúan los morfismos o que son los shuffles, entre otros casos.

Hemos usado el paquete *xypic* para hacer las figuras. Dichas figuras se forman mediante la descripción de puntos en el plano cartesiano y segmentos que unen dichos puntos. Hemos creado un fichero *latex\_gen.py* que es una colección de funciones para poder convertir una descripción completa de un árbol cualquiera a la descripción necesaria del paquete *xypic*, ya que la clase *Tree* genera una estructura de grafo. La función principal es *tree\_to\_latex*.

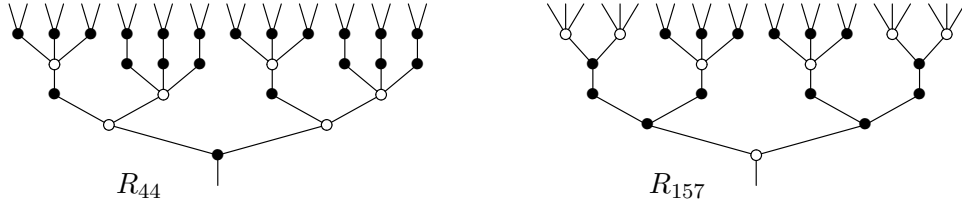
La descripción completa de un árbol admite símbolos de LaTeX para que sean compilados. Por ejemplo, sea  $S = "c \backslash \otimes qB(b_{-1}, b_{-2}; b) | wW(\backslash \backslash \Omega; b_{-1})"$ , si usámos la función obtendremos la siguiente figura



**Ejemplo 4.11.** Para acabar esta sección, pondremos un ejemplo para enseñar la utilidad del paquete. Sean  $S$  y  $T$  los árboles



Este sería un ejemplo tedioso de calcular el conjunto de shuffles ya que según la función  $sh$  existen 296 shuffles diferentes. Encontramos todos ellos mediante la clase *ShuffleLattice*, mostramos los shuffles  $R_{44}$  y  $R_{157}$



Se pueden encontrar todos los shuffles de este ejemplo en el Anexo A.



## 5 Conclusiones

La idea principal de este trabajo era poder entender el producto tensorial de conjuntos dendroidales, para ello hemos ido siguiendo un hilo que parte de las nociones básicas de categorías y opéradas y acaba con una técnica alternativa para encontrar dicho producto tensorial.

Una vez dadas las nociones básicas, el trabajo sigue con la definición de la categoría simplicial  $\Delta$ . Dicha categoría se forma mediante los simplexes como objetos y las caras y degeneraciones como los morfismos.

El trabajo se basa en un formalismo de árboles como opéradas coloreadas. Este formalismo nos permite tener una definición estricta de qué es un árbol, formado de sus entradas y salida. Y así, quedan bien definidos los morfismos de árboles como las caras y las degeneraciones. Con estos morfismos de opéradas coloreadas y con los árboles planares como objetos obtenemos la categoría  $\Omega_p$ . Tenemos que la categoría de árboles no planares  $\Omega$  viene de la categoría  $\Omega_p$  considerando que ahora los árboles admiten la acción del grupo simétrico en sus entradas.

Para poder introducir la definición de la categoría de conjuntos dendroidales, tenemos que introducir la noción de prehaces en  $\Omega$ . Para ello necesitamos un morfismo que envía árboles a su conjunto de representaciones planares, de esta manera tenemos el prehaz  $P$  en  $\Omega$ . Entonces, tenemos que  $\Omega \setminus P = \Omega_p$  y en consecuencia, existe una proyección de  $\Omega_p$  a  $\Omega$ . De esta manera podemos ver que las categorías dendroidales extienden la categoría simplicial mediante el encaje de  $\Delta$  en  $\Omega_p$  y la anterior proyección. Finalmente, un conjunto dendroidal  $X$  es un conjunto de dendrices con forma  $T$  para todo árbol  $T$  en  $\Omega$  junto las funciones asociadas a los morfismos de  $\Omega$ .

El producto tensorial de Boardman–Vogt introduce un producto distinto al producto cartesiano dentro de las opéradas coloreadas y gracias al nervio dendroidal podemos enviar un árbol como opéradas coloreadas a su representable en los conjuntos dendroidales. De esta manera, si definimos dos conjuntos dendroidales como los colímites de representables, podemos encontrar el producto tensorial de dichos conjuntos dendroidales encontrando el producto tensorial de Boardman–Vogt de dos opéradas coloreadas y luego aplicar el nervio dendroidal.

Encontrar el producto tensorial no es tarea fácil, así que necesitamos introducir la noción del conjunto de shuffles, donde un shuffle es simplemente un árbol como opéradas coloreadas generado por dos iniciales, manteniendo la configuración de cada uno de ellos a nivel de vértices. Es decir, la copia de un vértice blanco de la primera opéradas en un shuffle tendrá las mismas entradas que tenía en su opéradas original pero con la diferencia que estas entradas estarán conectadas al output de las copias de un vértice negro de la segunda opéradas. Luego, podemos generar todo el conjunto de shuffles distintos siguiendo la norma de intercambio. Una vez que encontramos el conjunto de shuffles de dos árboles cualesquiera podemos definir el producto tensorial de sus representables en conjuntos dendroidales como la unión de todos los shuffles encontrados.

Tenemos una función recursiva que nos calcula cuantos shuffles hay entre dos árboles sin generarlos en sí. Nos preguntamos si es posible encontrar una fórmula general de dicha función recursiva. Si trabajamos con los árboles lineales tenemos un caso especial donde sí existe una fórmula cerrada, resulta que el problema se reduce a una relación inductiva muy conocida, el coeficiente binomial.

Finalmente, generar todos los shuffles es una tarea relativamente fácil pero costosa de tiempo ya que el número de shuffles augmenta mucho según el tamaño de los árboles. Por eso hemos ~~decidido~~ ~~de~~ desarrollar un paquete en Python para que genere el conjunto de shuffles entre dos árboles cualesquiera. Para realizar el paquete hemos abstraído las nociones de opéradas coloreadas, de árboles como opéradas coloreadas y de los shuffles de árboles. Gracias al paquete hemos podido ilustrar todas las figuras que representan los árboles y los ejemplos de los conjuntos de shuffles.

*que aparecen en el trabajo*

## Bibliografía

- [1] Tom Leinster: *Basic Category Theory*, Cambridge Studies in Advanced Mathematics, Vol. 143, Cambridge University Press, 2014.
- [2] Greg Friedman: *An elementary illustrated introduction to simplicial sets*, Rocky Mountain J. Math. 42 (2012), no. 2, 353-423, arXiv:0809.4221 [math.AT].
- [3] Ieke Moerdijk and Bertrand Toën: *Simplicial Methods for Operads and Algebraic Geometry*, Springer Basel AG 2010.
- [4] Eric Hoffbeck and Ieke Moerdijk: *Shuffles of trees*, arXiv:1705.03638 [math.CO].
- [5] Saunders Mac Lane: *Categories for the Working Mathematician*, Springer Science+Business Media New York 1978.
- [6] G. M. Kelly: *Basic Concepts of Enriched Category Theory*, London Math. Soc. Lecture Notes, vol. 64, Cambridge University Press, Cambridge, 1982.