

1 Shuffle de árboles (Naippear, Barajear o mezclar?)

En esta sección describiremos el producto tensorial $\Omega[S] \otimes \Omega[T]$ para todo par de árboles S y T de Ω , y así conseguir que se entienda el resultado del producto tensorial de conjuntos dendroidales.

1.1 Producto tensorial de árboles lineales

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 son computados mediante un *shuffle*. Un (n, m) -*shuffle* es un camino de longitud máxima en el conjunto de orden parcial $[n] \times [m]$. Los $(n + m)$ -simplices no degenerados de $\Delta[n] \times \Delta[m]$ corresponde a los (n, m) -shuffles. De hecho,

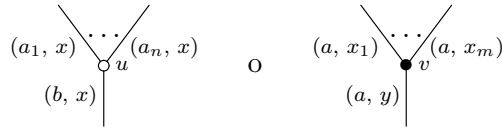
$$\Delta[n] \times \Delta[m] = \bigcup_{(n, m)} \Delta[n + m]$$

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

Ejemplo 1.1. Sea $n \dots$ por hacer

1.2 Producto tensorial de árboles

Definición 1.2. Sea S y T dos objetos de Ω . 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 negres*, 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 $[n] \times [m]$.

1.2.1 Conjunto de shuffles

Definición 1.3. 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 1.4. El número de shuffles $sh(S, T)$ de dos árboles S y T satisface tres propiedades:

$$(i) \quad sh(S, T) = sh(T, S)$$

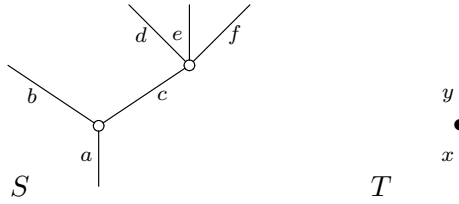
$$(ii) \quad Si \ T \text{ es un árbol unitario } \eta, \text{ entonces } sh(S, \eta) = 1$$

$$(iii) \quad Si \ S = C_n[S_1, \dots, S_n] \text{ y } T = C_m[T_1, \dots, T_m], \text{ entonces}$$

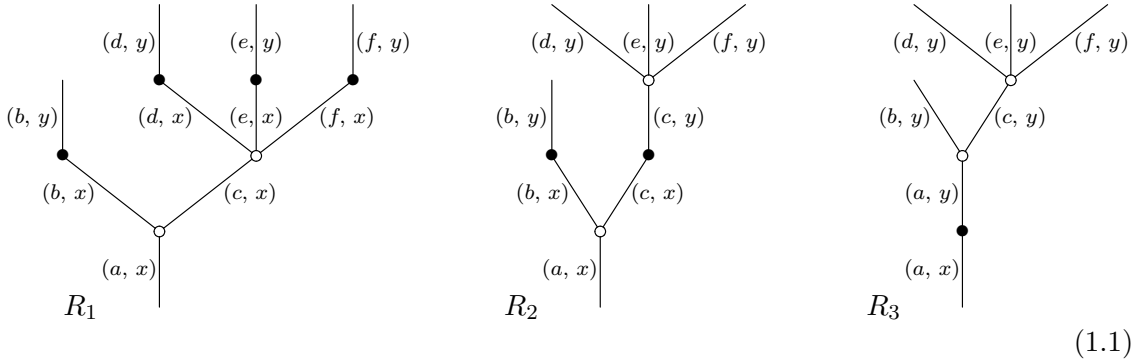
$$sh(S, T) = \prod_{i=1}^n sh(S_i, T) + \prod_{j=1}^m sh(S, T_j)$$

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 -esima la conectamos con la raíz del árbol S_i .

Ejemplo 1.5. Sean S y T los árboles

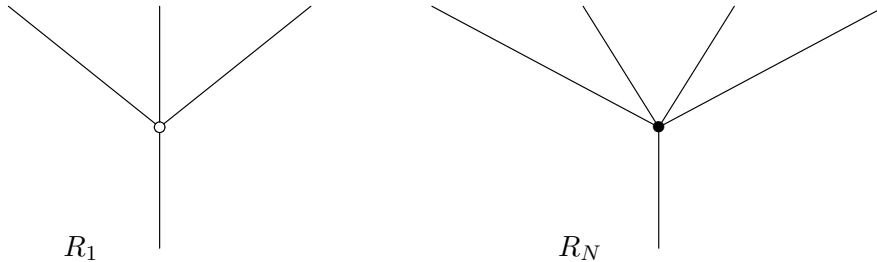


El conjunto de shuffles de S y T consiste de los siguientes tres árboles:

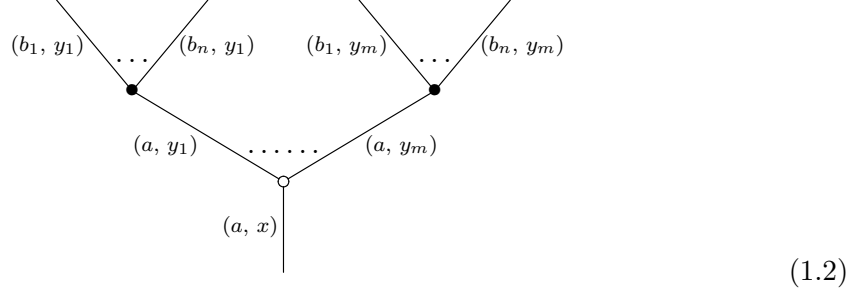


(1.1)

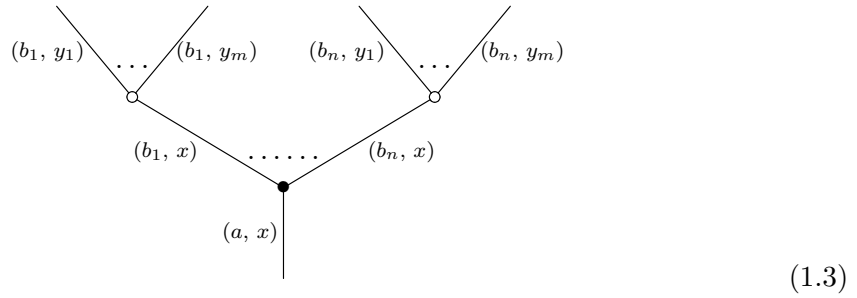
El conjunto de shuffles de S y T es *ordenado parcialmente*. El árbol minimal R_1 en el conjunto ordenado parcialmente se obtiene mediante la inserción de una copia del árbol negro T en cada entrada del árbol blanco S . Es decir, primero hacemos una copia del árbol S de la forma $S \otimes r_T$, donde 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 ordenado parcialmente 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 intermediarios* 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

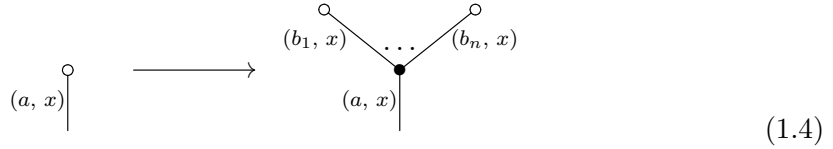


A una configuración de R_k

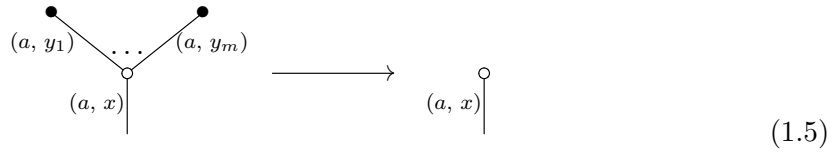


Si un shuffle R_k se obtiene the otro shuffle R_l mediante la norma de arriba, entonces decimos que R_k se obtiene mediante *un solo intercambio* y lo denotaremos por $R_l \leq R_k$. Así, obtenemos un orden parcial en el conjunto de todos los shuffles.

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



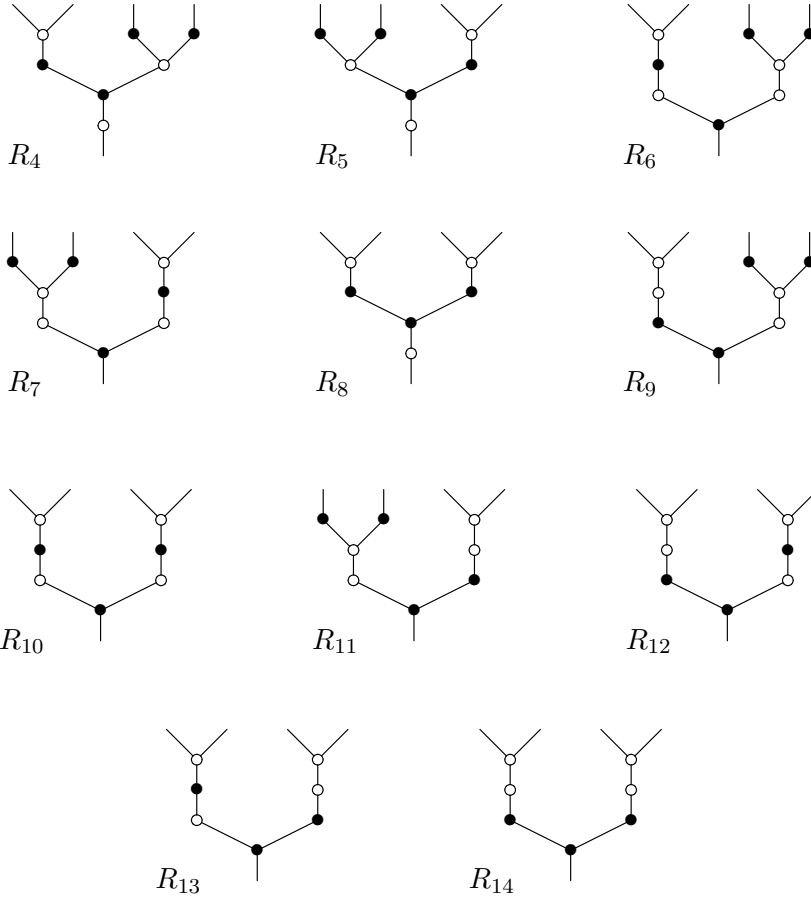
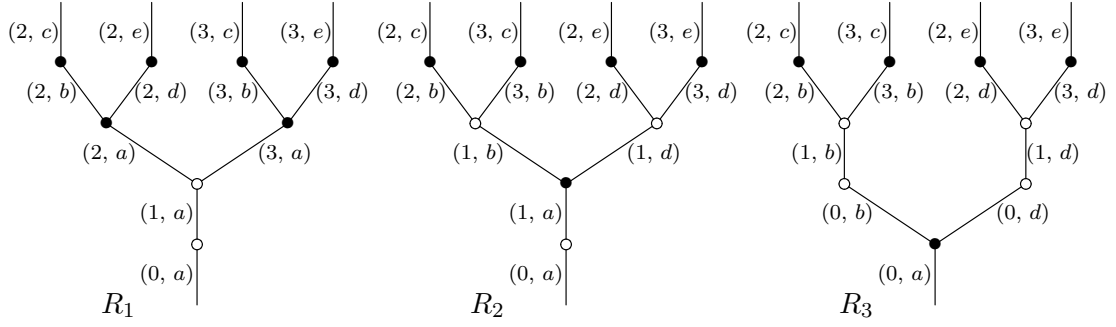
Finalmente, si $n = m = 0$, entonces tenemos el intercambio



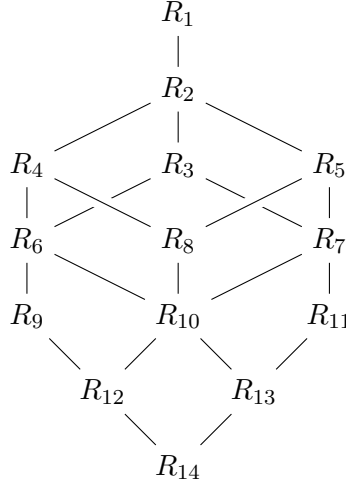
Ejemplo 1.6. Sean S y T los árboles



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 ordenado parcialmente.



Ahora vamos a relacionar el conjunto de shuffles con el producto tensorial de conjuntos dendroidales de representables de dos árboles cualesquiera, de esta manera el cálculo del producto resulta ser más fácil.

Lema 1.7. *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 $\Omega[S] \otimes \Omega[T]$. Vemos que es un monomorfismo. No entiendo. \square

Corolario 1.8. *Para todo objeto T y S en Ω , 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 .

1.3 Shuffle de árboles en Python

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

En esta sección describiré de manera breve el código que he escrito 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 *xy* de *Latex* de un árbol mediante una descripción básica.

Finalmente, dicho código lo podréis encontrar tanto en el Anexo 1 como en el repositorio público de código de Github: Trees Shuffling.

Clases básicas

Definición 1.9. Una *clase informática* es una abstracción de propiedades y funciones de un objeto en concreto.

Siguiendo tal definición, tenemos las siguientes clases básicas:

- Operada
- Arbol