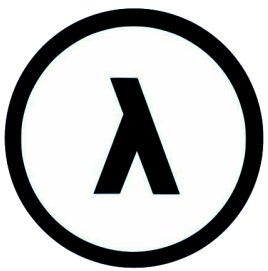
Funciones de Orden Superior

Ignacio Ballesteros
Luis Eduardo Bueso

https://github.com/edububa/haskell_course

8 de noviembre de 2017



Índice

Introducción

Curried functions

Las funciones en Haskell reciben **un único argumento**.

```
Prelude> max 4 5
5
Prelude> (max 4) 5
5
```

Si observamos el tipo de max:

```
max :: (Ord a) => a -> a -> a
```

que es equivalente a:

```
max :: (Ord a) => a -> (a -> a)
```

Introducción

Curried functions

Las siguientes funciones son equivalentes:

```haskell
multThree :: (Num a) => a -> a -> a -> a
multThree x y z = x * y * z

multThree' :: (Num a) => a -> (a -> (a -> a))
multThree' x y z = x * y * z
```

Entonces... ¿Qué sucede cuando llamamos a una función con menos argumentos de los que deberíamos?

```
*Main> mult2y4 = multThree 2 4
*Main> mult2y4 3
24
*Main>
```

Introducción

Definición

Una **función de orden superior** es aquella que recibe otra función como argumento y/o devuelve otra función como resultado.

Esto es posible gracias a que las funciones son **ciudadanos de primera clase**

Ejemplos:

```haskell
applyTwice :: (a -> a) -> a -> a
applyTwice f x = f (f x)

flip :: (a -> b -> c) -> (b -> a -> c)
flip f = g
  where g x y = f y x
```

Estructuras de datos

Definición

Estas son las implementaciones de las estructuras de datos que vamos a utilizar:

- **Listas**

```
data [] a = [] | a : [a]
```

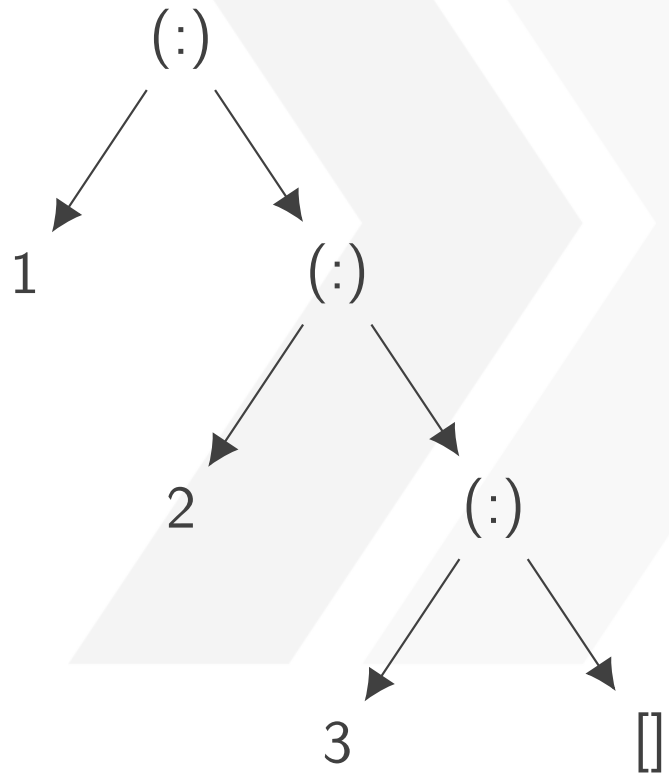
- **Árboles Binarios**

```
data BinaryTree a = Leaf
      | Node (BinaryTree a) a (BinaryTree a)
```

Estructuras de datos

Representación gráfica

Así es como representamos gráficamente la lista [1,2,3] en Haskell:

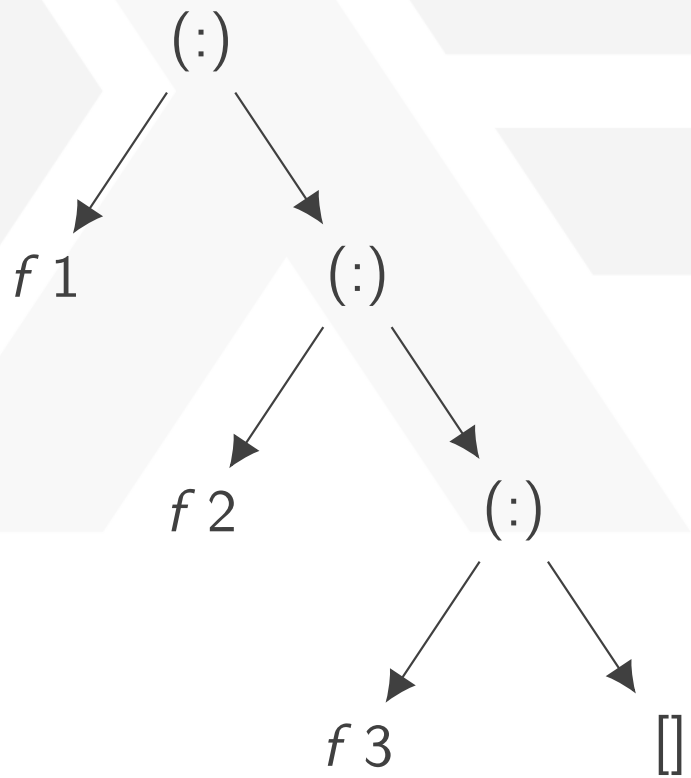


Funciones de orden Superior

map

La función *map* es una función de orden superior que aplica f a una estructura de datos.

Su representación gráfica sobre la lista [1,2,3]:



Su tipo es:

```
map :: (a -> b) -> [a] -> [b]
--        f          xs
```

Funciones de orden Superior

filter

La función *filter* es una función que recibe un predicado y una lista y devuelve la lista filtrada.

```
filter :: (a -> Bool) -> [a] -> [a]
--          predicado        xs
```

Ejemplo:

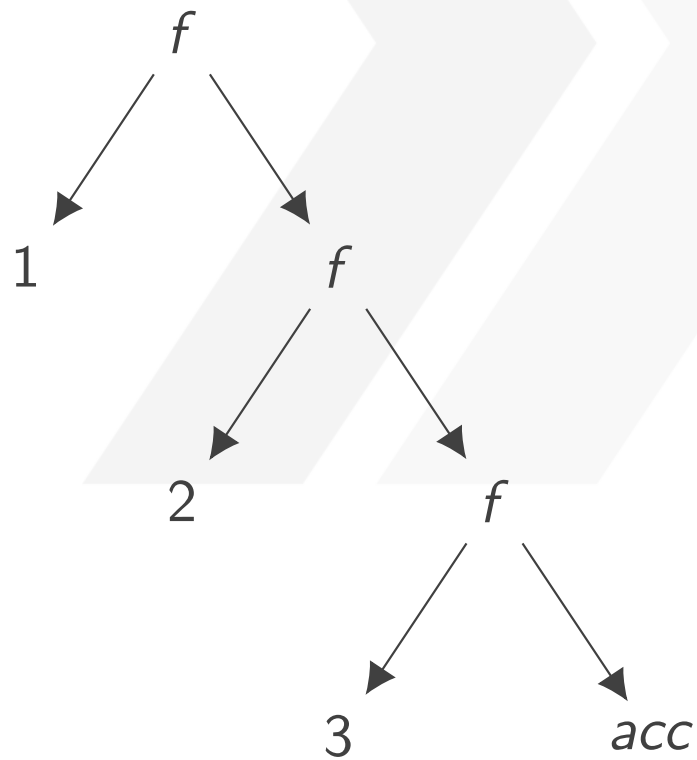
```
Prelude> :t even
even :: Integral a => a -> Bool
Prelude> :t odd
odd :: Integral a => a -> Bool
Prelude> filter even [1,2,3,4]
[2,4]
Prelude> filter odd [1,2,3,4]
[1,3]
```

Funciones de orden Superior

fold

La función *fold* es una función de orden superior que permite compactar una estructura de datos a un solo valor.

```haskell
foldr :: (a -> b -> b) -> b -> [a] -> b
--               f          acc    xs
```



Funciones anónimas

Las **funciones anónimas** son funciones que no tienen identificador. Usos:

- Como argumentos de funciones de orden superior.
- Para construir el resultado de funciones de orden superior.

Su sintaxis es la siguiente:

```
\x y -> x + y
```

Donde:

- \ indica el comienzo de la función.
- Entre \ y -> se encuentran sus argumentos.
- Detrás de -> se encuentra el cuerpo de la función

Recursos adicionales

Learn You a Haskell for Great Good!

`http://learnyouahaskell.com/chapters`