

ESTRUCTURA DE DATOS

TEMA 2

16/10/2017

Las funciones de orden superior son funciones que toman otra función como argumento o devuelve una función como resultado.

Una lista es una estructura polimórfica de datos homogéneos.

El símbolo ":" es un constructor de datos, ya que no tiene un cuerpo, sino que devuelve un dato.

Se usa type para asignar nombres definidos por el usuario a tipos concretos (por comodidad).

23/10/2017

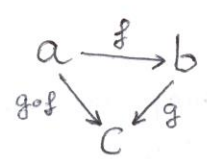
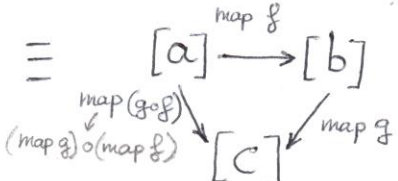
rara ($_ : \beta : \gamma : \mathbb{Z}S$) \leftarrow Patrón

Las funciones que manejan listas suelen tener definidas una ecuación para la lista vacía y otra para una lista ($x : xs$).

$$(:) \neq (++) \Rightarrow (:) :: a \rightarrow [a] \rightarrow [a] \neq (++) :: [a] \rightarrow [a] \rightarrow [a]$$

Algoritmo de recursión de cola: aquel que funciona como un algoritmo iterativo, al llegar al caso base se resuelve.
A diferencia de un algoritmo recursivo, se van eliminando los contextos de la pila.

La función map aplica una función f a todos los elementos de una lista.

$$\begin{aligned} \text{map} &:: (a \rightarrow b) \rightarrow [a] \rightarrow [b] \\ \text{map } f [] &= [] \\ \text{map } f (x:xs) &= f x : \text{map } f xs \end{aligned}$$

$$a \xrightarrow{f} b \quad \equiv \quad [a] \xrightarrow{\text{map } f} [b]$$

$$a \xrightarrow{\Delta a} a \equiv [a] \xrightarrow{\text{map } \Delta a} [a]$$

La función filter aplica una condición p (predicado) a todos los elementos de una lista, y devuelve otra lista con aquellos elementos que cumplen la condición anterior.

$\text{filter} :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$

$\text{filter } p [] = []$

$\text{filter } p (x:xs) \mid p\ x == \text{True} = x : \text{filter } p\ xs$
 $\mid \text{otherwise} = \text{filter } p\ xs$

λ -Funciones: funciones sin nombre (anónimas) que son definidas como argumento. Se expresan como " $\lambda x \rightarrow (\dots)$ ".

$f(x) = x + 1 \Rightarrow f = \lambda x. x + 1$

$f(3) = 3 + 1 \Rightarrow (\lambda x. x + 1)(3)$

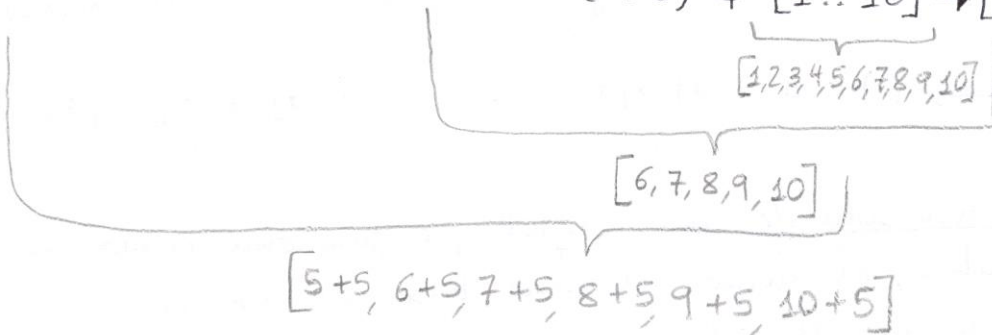
argumento de
 $f \$ x = f\ x$

$h \$ g \$ f\ x = h(g(f(x)))$

$[a..b] \rightarrow$ lista desde a hasta b

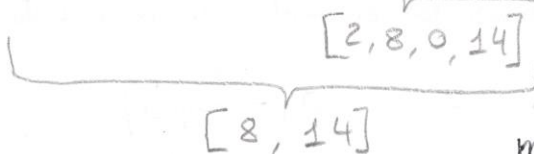
Ejemplo:

$\text{map } (\lambda x \rightarrow x + 5) \$ \text{filter } (x > 5) \$ [1..10] \rightarrow [11, 12, 13, 14, 15]$



25/10/2017

$\text{filter } (> 3) (\text{map } (*2) [1, 4, 0, 7]) \rightarrow [8, 14]$



map mantiene la longitud de la lista.
 filter no mantiene la longitud de la lista.

Las λ -funciones de n argumentos se definen " $\underbrace{\lambda x y z (\dots) n}_{\text{separadas por espacios}} \rightarrow (\dots)$ ".

Las funciones de orden superior `foldr` y `foldl` usan funciones binarias, ya que necesitan que su función argumento tenga un caso base.

$$\begin{aligned} \text{suma } \overset{\text{argumento}}{xs} &= \text{foldr } \underbrace{(\lambda e s \rightarrow e + s)}_{\lambda} \underbrace{0}_{\text{caso base}} \underbrace{xs}_{\text{lista (argumento)}} \\ &= \text{foldr } (+) \quad 0 \quad xs \end{aligned}$$

$$\text{algunoPar } xs = \text{foldr } (\lambda e s \rightarrow \text{even } e \parallel s) \text{ False } xs$$

$$\text{map } xs = \text{foldr } (\lambda e s \rightarrow f e : s) [] xs$$

$$\text{filter } xs = \text{foldr } (\lambda e s \rightarrow \text{if } p \text{ then } e : s \text{ else } s) [] xs$$

$$\text{inversa } xs = \text{foldr } (\lambda e s \rightarrow s ++ [e]) [] xs$$

$$\text{concat} :: [[a]] \rightarrow [a]$$

$$\text{concat } [] = []$$

$$\text{concat } (xs:xss) = xs ++ \text{concat } xss$$

Dada una lista, cuyos elementos son listas, devuelve otra lista con todos los elementos de esas listas: $[[a,b],[c,d],[e]] \rightarrow [a,b,c,d,e]$

$$\text{concat}^{xs} = \text{foldr } (\lambda e s \rightarrow e ++ s) [] xs$$

$$\text{lenght } xs = \text{foldr } (\lambda e s \rightarrow 1 + s) 0 xs$$

También puede pasarse una función incompleta, para que el argumento restante sea un valor de la lista.

Ejemplo:

$$g \times y = \text{mod } x \ y$$

$$\text{map } (g \ 2) \ [1, 2, 3] \rightarrow [1, 1, 0]$$

$$\text{map } (g \ 2) \ [\underline{1}, \underline{2}, \underline{3}] =$$

$$= g \ 2 \ \underline{1} : g \ 2 \ \underline{2} : g \ 2 \ \underline{3} : [] =$$

$$= 1 : 1 : 0 : []$$