



## Tema 3. Constructores de tipos

---

Para definir tipos compuestos a partir de otros tipos

**Tupla:**  $(T_1, T_2, \dots, T_n) \equiv T_1 \times T_2 \times \dots \times T_n$

➤ Estructura de datos heterogénea

**Lista:**  $[T]$

➤ Estructura de datos homogénea

**Función:**  $T_1 \rightarrow T_2$



## Tuplas $(T_1, T_2, \dots, T_n)$

---

Ejs:  $(\text{'a'}, (7,8)) :: (\text{Char}, (\text{Int}, \text{Int}))$   
 $(\text{"hola"}, 8.7, \text{True}) :: (\text{String}, \text{Float}, \text{Bool})$

➤ Patrones de tuplas:

$(x,y)$  ó  $(x,y,z)$  ó  $(x,(y,z))$  .....

➤ Definición de funciones por ajuste de patrones:

$\text{fst } (x,y) = x$

$\text{snd } (x,y) = y$

( $\text{fst}$  y  $\text{snd}$  predefinidas para los pares)



## Ejemplo con tuplas

---

`raices :: (Float,Float,Float) -> (Float, Float)`

`raices (a,b,c)`

`| a == 0`                    `= error "no es de 2 grado"`

`| e < 0`                    `= error "raices complejas"`

`| otherwise`              `= ((-b-s)/d, (-b+s)/d)`

where                    `e = b*b - 4*a*c`

`s = sqrt e`

`d = 2*a`

`? raices (1,3,2)`

`(-2.0,-1.0)`



## Ejemplo con tuplas / Currificación

Definimos  $\text{minimo} :: (\text{Int}, \text{Int}) \rightarrow \text{Int}$

$\text{minimo } (x,y) = \text{if } x < y \text{ then } x \text{ else } y$

su versión currificada es:

$\text{minimoc} :: \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$

$\text{minimoc } x \ y = \text{if } x < y \text{ then } x \text{ else } y$

- *Expresiones correctas:*     $\text{minimo } (5,2)$      $\text{minimoc } 5 \ 2$
- *Expresiones incorrectas:*     $\text{minimo } 5 \ 2$      $\text{minimoc } (5,2)$



## Funciones “curry” y “uncurry”

Funciones predefinidas en Haskell:

$$\text{curry } f \ x \ y = f \ (x,y)$$
$$\text{uncurry } g \ (x,y) = g \ x \ y$$

$\text{curry } f$  es la versión currificada de  $f :: (T_1, T_2) \rightarrow T$

$\text{uncurry } g$  es la versión descurrificada de  $g :: T_1 \rightarrow T_2 \rightarrow T$

Ej: Para la función  $\text{minimo} :: (\text{Int}, \text{Int}) \rightarrow \text{Int}$

$\text{curry minimo}$  es equivalente a  $\text{minimoc} :: \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$

- *Exp. correctas:*  $\text{minimo } (5,2)$        $(\text{curry minimo}) \ 5 \ 2$



## Listas [T]

---

Ejs:     $[3,4,5,9] :: [\text{Int}]$

$[3,4,5,9] == 3:4:5:9:[]$

$['a','4','+'] :: [\text{Char}]$

- Patrones de listas:

$[]$        lista vacía

$(x:s)$     lista con cabeza  $x$  y resto  $s$

- Definición de funciones por ajuste de patrones:

$\text{head } (x:s) = x$                       (  $\text{head } [] = \text{error}$  )

$\text{tail } (x:s) = s$                       (  $\text{tail } [] = \text{error}$  )



## Ejemplo con listas

-- Elementos positivos de una lista

positLista :: [Int] -> [Int]

positLista [] = []

positLista (x:s) = if x>0 then x: positLista s  
                  else positLista s

} script

? positLista [3,-2,9,-5,0]  
[3,9]

---

*NOTA:* script con comentarios, declaraciones y definiciones



## Tipos sinónimos

**type** <tipo> = <expresión de tipo>

- Nuevo nombre para un tipo => Mayor legibilidad
- **type** String = [Char]      “hola” == ['h','o','l','a']

Ejs:    type Biblioteca = [Libro]  
         type Libro = (Titulo, FechaPub, Autor)  
         type Autor = (Nombre, Apellido)  
         type FechaPub = Int  
         type Nombre = String  
         type Apellido = String  
         type Titulo = String





## Ejemplo “biblioteca”

---

libro1 = (“Haskell”, 1998, (“Richard”, “Bird”)) :: Libro

libro2 = (“Miranda”, 1995, (“Alan”, “Turner”)) :: Libro

.....

mibib = [libro1, libro2, ...] :: Biblioteca

**darAutor :: Libro → Autor**

darAutor (t,f,a) = a

**obtenerTitulos :: Biblioteca → [Titulo]**

obtenerTitulos [] = []

obtenerTitulos ((t,f,a):res) = t: obtenerTitulos res



# Polimorfía

---

- Tipo de las listas es **polimórfico**:  $[\alpha]$   
con funciones polimórficas como  $\text{head}:: [\alpha] \rightarrow \alpha$   
(donde  $\alpha$  es una **variable de tipo**)
- Tipo de las tuplas es **polimórfico**:  $(\alpha, \beta)$   $(\alpha, \beta, \gamma)$  .....  
con funciones polimórficas como  $\text{fst}:: (\alpha, \beta) \rightarrow \alpha$   
(donde  $\alpha, \beta$  son **variables de tipo**)
- La función error es **polimórfica**:  $\text{error}:: \text{String} \rightarrow \alpha$

---

*NOTA:* Usaremos letras griegas  $\alpha, \beta, \chi, \dots$  para las variables de tipo aunque en Haskell son a,b,c,....



## Instanciación de tipos

---

Se obtiene al sustituir de \*forma consistente\* las variables (de tipo) por tipos.

Ejs:

$(\text{Char}, \text{Char})$  es instancia de  $(\alpha, \beta)$

$(\text{Int}, [\text{Int}])$  es instancia de  $(\alpha, \beta)$

$[\text{Char}] \rightarrow \text{Char}$  es instancia de  $[\alpha] \rightarrow \alpha$

pero

$[\text{Char}] \rightarrow \text{Int}$  NO es instancia de  $[\alpha] \rightarrow \alpha$

---

(\*) todas las apariciones de una variable por el mismo tipo



## Inferencia de tipos (1)

- Se infiere el tipo más general (si existe) para las funciones definidas por el usuario.

Ej: Para  $f\ x\ y = (\text{head}\ x, y)$  se infiere como tipo  
 $f :: [\alpha] \rightarrow \beta \rightarrow (\alpha, \beta)$

- Preguntas para cualquier instanciación del tipo de  $f$

? f ['a','e','u'] "ko"

('a', "ko") :: (Char, [Char])

donde

$\uparrow$   
 $\alpha$

$\uparrow$   
 $\beta$

? f ['4'] True

('4', True) :: (Char, Bool)

$\uparrow$   
 $\alpha$

$\uparrow$   
 $\beta$



## Inferencia de tipos (2)

---

? f 'a' 3

$f :: [\alpha] \rightarrow \beta \rightarrow (\alpha, \beta)$

ERROR: No concuerda el tipo esperado con el real

\* Expresión : f 'a' 3

\* Término : 'a'

\* Tipo real : Char

\* Tipo esperado :  $[\alpha]$

? head ['c', "mesa"]

$\text{head} :: [\alpha] \rightarrow \alpha$

ERROR: No concuerda el tipo esperado con el real

\* Subexpresión : ['c', "mesa"]

\* Término : "mesa"

\* Tipo real : String

\* Tipo esperado : Char



## Comprobación de tipos (1)

- Se comprueba si el tipo declarado es una instancia del tipo inferido.

Ej:  $f :: [\text{Char}] \rightarrow \text{Integer} \rightarrow (\text{Char}, \text{Integer})$   
 $f\ x\ y = (\text{head}\ x, y)$

Comprobación correcta:

$[\text{Char}] \rightarrow \text{Integer} \rightarrow (\text{Char}, \text{Integer})$  es instancia de  
 $[\alpha] \rightarrow \beta \rightarrow (\alpha, \beta)$  para  $\alpha = \text{Char}$  y  $\beta = \text{Integer}$

? f ['a','e','u'] 3

('a',3) :: (Char, Integer)

? f [4,5] True

ERROR: No concuerda el tipo  
esperado con el real .....



## Comprobación de tipos (2)

---

- Se obtiene un error si el tipo declarado no es una instancia del inferido por el sistema

$f :: [\text{Char}] \rightarrow \text{Integer} \rightarrow (\text{Bool}, \text{Integer})$

$f\ x\ y = (\text{head}\ x, y)$

Comprobación: error

- Se obtiene un error si no existe tipo inferido para una definición dada (sin declaración)

$g\ x\ y = (\text{head}\ x, x + y)$

Comprobación: error