Escuela de Ingeniería Informática
Universidad de Oviedo

Cálculo lambda

Seminario 3

Programación Funcional

Cálculo Lambda

- El cálculo lambda (λ-calculus) es un sistema formal basado en la **definición de funciones** (abstracción) y su **aplicación** (invocación)
- Una **expresión** lambda se define como
  - Una variable **x** (x, y, z, x_1, x_2…)
  - Una abstracción **λx.M** (*M*, *N*, M_1, M_2…)
    donde x es una variable y *M* es una expresión
  - Una aplicación ***M N***
    donde *M* y *N* son expresiones
- Pregunta: ¿Cuál es la funcionalidad de las dos siguientes funciones?
  - `λx.x`
  - `λf.λx.f(fx)`

Lenguaje Universal

- El cálculo lambda se considera como el **lenguaje más pequeño universal de computación**
  - Es universal porque cualquier función computable puede ser expresada y evaluada en él
- Mediante expresiones lambda se pueden representar
  - Constantes y lógica booleana
  - Constantes y aritmética entera y real
  - Funciones con cualquier número de parámetros (currificación)
  - Distintos tipos de datos (tuplas, registros, arrays…)
  - Orientación a Objetos
  - …

Lógica Booleana

- Vamos a ver cómo representar en cálculo lambda las constantes booleanas y sus principales operaciones
- El literal **`true`** puede representarse como

  `true` ≡ λt.λf.t

- Y **`false`** como

  `false` ≡ λt.λf.f

- Pregunta: ¿Qué representan?

Ejercicio 1

- Dadas las dos constantes **`true`** y **`false`**, implemente la función **`if-else`** que implemente el clásico condicional (en versión funcional)
  - ¿Cuántos parámetros recibirá?
  - ¿De qué "tipo" (que tipo de valores alberga) es cada parámetro?
  - ¿Cuál será su cuerpo?
  - Implemente la función **`máximo`** y que use la función **`if-else`** definida (puede usar el operador >)
  - Haga una invocación de ejemplo, mostrando todas las sustituciones

Ejercicio 2

- Dadas las dos constantes **`true`** y **`false`**, implemente una función lambda que implemente el operador **`and`** (**&&** de Java)
  - ¿Cuántos parámetros recibirá?
  - ¿De qué "tipo" (que tipo de valores alberga) es cada parámetro?
  - ¿Cuál será su cuerpo?

Ejercicios 3 y 4

- Siguiendo el mismo método
  1. Implemente la función **`or`** (|| en Java)
  2. Implemente la función **`not`** (! en Java)

Resto de tipos de datos y operandos

- Cualquier tipo de datos y operandos puede representarse en cálculo lambda como funciones de orden superior
  - Nºs naturales, booleanos, enteros, listas…
- A esta representación se la conoce como **Church encoding** debido a Alonzo Church, que fue el primero que codifico datos de esta forma en el cálculo lambda