Estrategias de evaluación

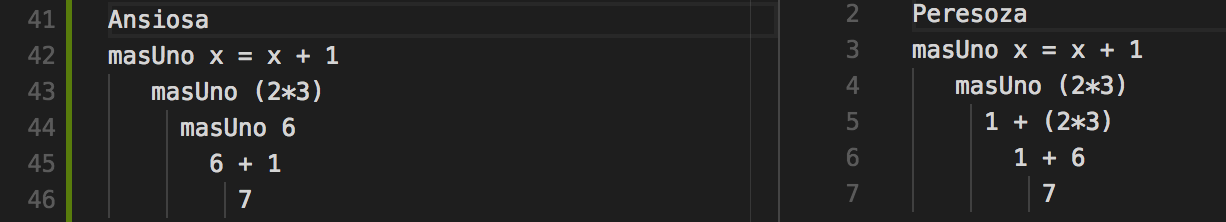
## Conceptos clave

* Redex
* Evaluación ansiosa
* Evaluación perezosa
* Funciones estrictas
* Listas infinitas

## Introducción a la clase

La idea del tema que vamos a tratar a continuación es saber qué se tiene que tener en cuenta para determinar el orden en en que aplicarán las funciones de una expresión.

Empecemos con el siguiente ejemplo:



A una expresión que consta de una función aplicada a uno o más parámetros y que puede ser “reducida” aplicando dicha función la vamos a llamar **Redex** (Reducible Expression).

## De adentro hacia afuera

Una de las estrategias más comunes es comenzar desde adentro hacia afuera (**innermost evaluation**), esta estrategia elige el redex que está “más adentro” entendiendo por esto al redex que no contiene otro redex. Si existe más de un redex que cumple dicha condición se elige el que está más a la izquierda.

Esta estrategia me asegura que los parámetros de una función están completamente evaluados antes de que la función sea aplicada. Por eso se dice que los parámetros se pasan por valor. **Call-by-value**

## De afuera hacia adentro

Otra de las estrategias más comunes es comenzar desde afuera hacia adentro (**outtermost evaluation**), esta estrategia elige el redex que está “más afuera” entendiendo por esto al redex que no esta contenido en otro redex. Si existe más de un redex que cumple dicha condición se elige el que está más a la izquierda.

Usando esta estrategia las funciones se aplican antes que los parámetros sean evaluados. Por esto se dice que los parámetros se pasan por nombre.

**Call-by-name**



## Infinity evaluations

Se tiene:

**inf = 1 + inf**

Sabiendo que:

fst (x,\_) = x

Evaluar (ansiosa y perezosamente) si termina o no la siguiente expresión:

**fst (1,inf)**

Lo mismo para:

naturalesDesde x = x : naturalesDesde (x+1)

take 0 \_ = []

take \_ [] = []

take n (x:xs) = x : (take (n-1) xs)

Evaluar

**take 3 (naturalesDesde 1)**

**Y ahora?**

head (filter (3<) [1..])

Si bien la expresión filter (3<) [1..] no termina (seguiría buscando cuáles son mayores a 3 infinitamente), como lo que primero se evalúa es el head y se difiere la ejecución del filtrado, la ejecución va a terminar en cuanto el filter encuentre su primer elemento que pertenezca a la solución que es el 4.

## **Conslusiones**

**Lazy Evaluation**: con esta estrategia los parámetros solo se resuelven cuando son necesarios (y son evaluados solo lo necesario). También conocida como **evaluación perezosa** o **diferida**.

A la estrategia call-by-value (y sus variantes) también se las conoce como Eager Evaluation. **Eager Evaluation**: con esta estrategia los parámetros tienen que resolverse antes de aplicar la función. También conocida como **evaluación ansiosa**.

## Referencias

* **Estrategias de evaluación**

[**http://wiki.uqbar.org/wiki/articles/estrategias-de-evaluacion.html**](http://wiki.uqbar.org/wiki/articles/estrategias-de-evaluacion.html)