Programación funcional

Funciones

$$\begin{array}{rcl} f : \mathcal{A} & \to & \mathcal{B} \\ f(x) & \mapsto & \ldots \end{array}$$

Ejemplos: $sucesor$, $sumaCuadrados$ (2 argumentos) y pi (constante)

$$\begin{array}{rcl|rcl|rcl} sucesor : \mathbb{Z} & \to & \mathbb{Z} & sumaCuadrados : \mathbb{Z} \times \mathbb{Z} & \to & \mathbb{Z} & \pi & : & \mathbb{R} \\ sucesor(x) & \mapsto & x+1 & sumaCuadrados(x,y) & \mapsto & x^2+y^2 & \pi & \mapsto & 3.1415927\ldots \end{array}$$

✓ Evaluación de una función para ciertos valores de la variable:

$$sucesor(1) \Longrightarrow 2 \quad sumaCuadrados(2,3) \Longrightarrow 13$$

Sesiones y declaraciones

• Programar = especificar los pasos para resolver un problema.

• Solución a un problema = valor calculado a partir de ciertos datos.

En programación funcional, ordenador = calculadora o *evaluador*

Existen un conjunto de funciones *predefinidas* (PRELUDE)

$$\text{PRELUDE>}\ 1+2$$
$$3 :: \ Integer$$

Las líneas anteriores representan un *diálogo* con el evaluador.

✓ El siguiente ejemplo utiliza valores reales:

$$\text{PRELUDE>}\ cos\ pi$$
$$-1.0 :: \ Double$$

Sólo se usan paréntesis cuando es necesario:

$$\text{PRELUDE>}\ cos\ (2 * pi)$$
$$1.0 :: \ Double$$

$$\text{PRELUDE>}\ [1..5]$$
$$[1,2,3,4,5] :: \ [Integer]$$

$$\text{PRELUDE>}\ sum\ [1..10]$$
$$55 :: \ Integer$$

✓ El siguiente ejemplo muestra cómo se usan funciones de más de un argumento:

$$\text{PRELUDE>}\ mod\ 10\ 3$$
$$1 :: \ Integer$$

$$\text{PRELUDE>}\ mod\ 10\ (3+1)$$
$$2 :: \ Integer$$

- HASKELL es un lenguaje *fuertemente tipificado*.

  Proporciona un rico conjunto de elementos predefinidos.
  Este conjunto es ampliable vía declaraciones o definiciones:

$$\begin{array}{ll} sucesor \quad :: Integer \to Integer & -- \text{declaración de tipo} \\ sucesor\ x \ = x + 1 & -- \text{cómo computar el valor de la función} \end{array}$$

  MAIN> $sucesor\ 3$
  $4 :: Integer$
  MAIN> $10 * sucesor\ 3$
  $40 :: Integer$

Las siguientes declaraciones muestra la definición de una función de dos argumentos:

$$\begin{array}{l} sumaCuadrados \quad :: Integer \to Integer \to Integer \\ sumaCuadrados\ x\ y \ = x * x + y * y \end{array}$$

MAIN> $sumaCuadrados\ 2\ 3$
$13 :: Integer$
MAIN> $sumaCuadrados\ (2 + 2)\ 3$
$25 :: Integer$

✓ Los tipos de los distintos argumentos aparecen separados por el *constructor de tipo* $\to$.

✓ El último tipo es el del resultado.

Reducción de expresiones

- El evaluador *simplifica* la expresión original todo lo posible y muestra el resultado.
- La simplificación se produce, en general, tras varios pasos:

$$\begin{aligned} &cuadrado &&:: Integer \rightarrow Integer \\ &cuadrado\ x &&= x * x \end{aligned}$$

podemos calcular el valor de la expresión:

$$\begin{array}{ll} & 2 + \underline{cuadrado\ 3} \\ \Longrightarrow & \{\text{! por la definición de } cuadrado\} \\ & 2 + (\underline{3 * 3}) \\ \Longrightarrow & \{\text{! por el operador } (*)\} \\ & \underline{2 + 9} \\ \Longrightarrow & \{\text{! por el operador } (+)\} \\ & 11 \end{array}$$

Cada paso es una *reducción*.

✓ Un *redex* es cada parte de la expresión que pueda reducirse.

✓ Cuando una expresión no puede ser reducida más se dice que está en *forma normal*.

✓ La labor del evaluador consiste en:

- ▶ mientras quede algún *redex* en la expresión
  seleccionar un *redex* y
  reducirlo

Una vez alcanzada la forma normal, el evaluador muestra el resultado.

¿Qué ocurre si hay más de un *redex*?.

Por ejemplo $cuadrado\ (cuadrado\ 3)$

• Podemos reducir la expresión desde dentro hacia fuera (primero los *redexes* internos):

$$
\begin{array}{ll}
& cuadrado(\underline{cuadrado\ 3}) \\
\Longrightarrow & \text{! por la definición de } cuadrado \\
& cuadrado(\underline{3 * 3}) \\
\Longrightarrow & \text{! por el operador } (*) \\
& \underline{cuadrado\ 9} \\
\Longrightarrow & \text{! por la definición de } cuadrado \\
& \underline{9 * 9} \\
\Longrightarrow & \text{! por el operador } (*) \\
& 81
\end{array}
$$

Esta estrategia presenta algunos problemas.

• Una estrategia mejor consiste en reducir la expresión desde fuera hacia dentro (primero los *redexes* externos):

$$cuadrado\ \boxed{x} \Longrightarrow \boxed{x} * \boxed{x}$$

No es necesario evaluar previamente el argumento para aplicar la definición de la función $cuadrado$.

$$
\begin{array}{ll}
& \underline{cuadrado(cuadrado\ 3)} \\
\Longrightarrow & \text{! por la definición de } cuadrado \\
& \underline{(cuadrado\ 3)} * (cuadrado\ 3) \\
\Longrightarrow & \text{! por la definición de } cuadrado \\
& (\underline{3 * 3}) * (cuadrado\ 3) \\
\Longrightarrow & \text{! por la definición de } (*) \\
& 9 * (\underline{cuadrado\ 3}) \\
\Longrightarrow & \text{! por la definición de } cuadrado \\
& 9 * (\underline{3 * 3}) \\
\Longrightarrow & \text{! por el operador } (*) \\
& \underline{9 * 9} \\
\Longrightarrow & \text{! por el operador } (*) \\
& 81
\end{array}
$$

Para reducir la expresión $e_1 * e_2$ hay que reducir previamente e_1 y e_2. ($*$ es *estricto*).

Transparencia referencial

Sea cual sea la estrategia seguida en las reducciones, el resultado final (el valor 81) es el mismo:

✓ Si aparecen varios *redexes*, podemos elegir cualquiera.

✓ Sin embargo, la reducción de un *redex* equivocado puede que no conduzca a la forma normal de una expresión:

$$\begin{array}{ll} \mathit{infinito} & :: \ \mathit{Integer} \\ \mathit{infinito} & = 1 \ + \ \mathit{infinito} \\ \mathit{cero} & :: \ \mathit{Integer} \ \rightarrow \ \mathit{Integer} \\ \mathit{cero}\ x & = 0 \end{array}$$

- Si en cada momento elegimos el *redex* más interno:

$$\begin{array}{ll} & \mathit{cero}\ \underline{\mathit{infinito}} \\ \Longrightarrow & \text{! por definición de } \mathit{infinito} \\ & \mathit{cero}\ (1 + \underline{\mathit{infinito}}) \\ \Longrightarrow & \text{! por definición de } \mathit{infinito} \\ & \mathit{cero}\ (1 + (1 + \underline{\mathit{infinito}})) \\ \Longrightarrow & \text{! por definición de } \mathit{infinito} \\ & \dots \end{array}$$

y la evaluación no terminaría nunca.

- Si en cada paso elegimos el *redex* más externo:

$$\begin{array}{ll} & \underline{\mathit{cero}\ \mathit{infinito}} \\ \Longrightarrow & \text{! por definición de } \mathit{cero} \\ & 0 \end{array}$$

$$\forall\, n :: \ \mathit{Integer}\ \centerdot\ \mathit{cero}\ n \ \Longrightarrow\ 0.$$

En particular, $\mathit{cero}\ \mathit{infinito} \ \Longrightarrow\ 0.$

La estrategia utilizada para seleccionar el *redex* es crucial.

Órdenes de reducción aplicativo y normal

orden de reducción = Estrategia que indica qué *redex* hay que seleccionar en cada paso de la reducción.

Existen varios órdenes de reducción. Dos de los más interesantes:

- Orden *aplicativo*
- orden *normal*.

- **Orden *aplicativo***:

Seleccionar siempre el *redex* más interno y más a la izquierda.

✓ Esta estrategia de reducción es conocida como *paso de parámetros por valor* (*call by value*).

✓ A los evaluadores que utilizan este orden se los llama *estrictos* o *impacientes*.

- Problemas del *paso por valor*

✓ A veces, se efectúan reducciones que no son necesarias:

$$\begin{array}{ll} & \textit{cero}\ (\underline{10 * 4}) \\ \Longrightarrow & !\ \text{por el operador}\ (*) \\ & \underline{\textit{cero}\ 40} \\ \Longrightarrow & !\ \text{por definición de}\ \textit{cero} \\ & 0 \end{array}$$

✓ Problema aún mayor: puede no conducir a la forma normal de la expresión. Por ejemplo:

cero infinito

- **Orden de reducción *normal***

Consiste en seleccionar el *redex* más externo y más a la izquierda.

✓ Esta estrategia de reducción se conoce como *paso de parámetros por nombre* (*call by name*).

✓ A los evaluadores que utilizan el orden normal se los llama *no estrictos*.

✓ El *paso por nombre* es *normalizante*

Evaluación perezosa

- Con *paso por nombre* ciertas expresiones se reducen varias veces.

  En $cuadrado\ (cuadrado\ 3)$ la expresión:

  $(3\ *\ 3)$

  es calculada 2 veces.
- Esto no ocurre en la reducción que usa *paso por valor*
- La estrategia de *paso de parámetros por necesidad* (*call by need*), también conocida como *evaluación perezosa*, soluciona este problema.

✓ La evaluación perezosa consiste en utilizar *paso por nombre* y recordar los valores de los argumentos ya calculados para evitar recalcularlos:

$\underline{cuadrado\ (cuadrado\ 3)}$
$\Longrightarrow$! por la definición de $cuadrado$
$a * a$ **donde** $a = \underline{cuadrado\ 3}$
$\Longrightarrow$! por la definición de $cuadrado$
$a * a$ **donde** $a = \underline{b * b}$ **donde** $b = 3$
$\Longrightarrow$! por el operador $(*)$
$\underline{a * a}$ **donde** $a = 9$
$\Longrightarrow$! por el operador $(*)$
81