

Valores, expresiones y reducción

Programación funcional

Docentes:

- ▶ Daniel Ciolek
- ▶ Federico Sawady

Sitio:

- ▶ <https://sites.google.com/site/pftpiunq/>

Evaluación:

- ▶ Parcial / Recuperatorio
- ▶ TP
- ▶ Promoción (7+) / Integrador

Preguntas

¿Qué es un programa?

¿Qué es un modelo de cómputo?

¿Cuáles son las propiedades deseables de un programa?

Preguntas

¿Qué es un programa?

Programa

Descripción ejecutable.

¿Qué es un modelo de cómputo?

¿Cuáles son las propiedades deseables de un programa?

Preguntas

¿Qué es un programa?

Programa

Descripción ejecutable.

¿Qué es un modelo de cómputo?

Modelo de cómputo

Modelo formal que explica el proceso por el cuál a partir de un input se obtiene un output.

¿Cuáles son las propiedades deseables de un programa?

Preguntas

¿Qué es un programa?

Programa

Descripción ejecutable.

¿Qué es un modelo de cómputo?

Modelo de cómputo

Modelo formal que explica el proceso por el cuál a partir de un input se obtiene un output.

¿Cuáles son las propiedades deseables de un programa?

- ▶ Correctitud
- ▶ Eficiencia
- ▶ Claridad/Simplicidad
- ▶ Modificabilidad/Extensibilidad
- ▶ Otras...



Alonso Church

Paradigma de programación inspirado en el lambda cálculo.

Repaso:

Definición de elementos:

```
data Day   = Mon | Tue | Wed | Thu | Fri | Sat | Sun
data Bool = False | True
data Nat   = Zero | Suc Nat
```

Definición de funciones:

```
isWeekend Sat = True
isWeekend Sun = True
isWeekend _   = False
```

```
not True  = False
not False = True
```

```
inc n = Suc n
```

Visión denotacional:

- ▶ **Valores:** elementos abstractos.
- ▶ **Expresiones:** construcciones sintácticas (correctas) que denotan valores.

Términos (T):

- ▶ Variables: x
- ▶ Funciones: $(\lambda x \rightarrow T)$
- ▶ Aplicación: $T_1 T_2$
- ▶ Pattern-Matching¹: `case T of {C1 -> T1; ...; CN -> TN}`

Syntactic sugar:

`f x = T` \equiv `f = $\lambda x \rightarrow T$`

`g C1 = T1`

`g C2 = T2` \equiv `g = $\lambda x \rightarrow$ case x of {C1 -> T1; C2 -> T2}`

`if T1 then T2 else T3` \equiv `case T1 of {True -> T2; False -> T3}`

¹Versión simplificada.

Variables libres vs ligadas

Una variable se dice “ligada” si aparece dentro del alcance de un lambda, en caso contrario contrario se dice “libre”.

Ejemplo:

```
(\x -> x) y  --  x esta ligada, y esta libre
```

! Una expresión con variables libres es inválida.

Reducción: reglas de reescritura

$$\frac{D \cup \{x = T\}, x}{T} \text{ Expansión de definición}$$

$$\frac{D, (\lambda x \rightarrow T1) T2}{T1[x \leftarrow T2]} \text{ Beta reducción}^2$$

$$\frac{D, \text{case } C_i \text{ of } \{C1 \rightarrow T1; \dots; C_n \rightarrow T_n\}}{T_i} \text{ Matching}^2$$

Forma Normal (FN)

Término que no se puede reducir más.

²Versión simplificada (sin parámetros).

Reducción: selección del redex

- ▶ Eager o Aplicativo: Primero lo más interno (eg. argumentos).
- ▶ Lazy o Normal: Primero lo más externo.
- ▶ Otros...

```
zero x = Zero
```

```
inf = Suc inf
```

```
zero inf  --  a que reduce?
```

Reducción: propiedades

- ▶ **Normalización:** Si se llega a una FN, es siempre la misma independientemente del orden de reducción.
- ▶ **Confluencia:** El orden normal llega a una FN si existe al menos una manera de llegar a una FN.
- ▶ **Transparencia referencial:** El resultado de las funciones depende sólo de sus parámetros.

Desafío

¿Cómo podemos definir una función aleatoria `random`?