

Representación de Datos Abstractos en λ -cálculo

Mauro J. Jaskelioff

11 de febrero de 2004

En λ -cálculo los datos se representan por términos en forma normal.

Para representar un data de Haskell (o cualquier tipo de datos abstractos) se puede utilizar la siguiente técnica, que aplicaremos (como ejemplo) al siguiente tipo de datos

```
data Lista a = Nil | Cons a (Lista a)
```

Si tenemos n constructores, donde cada constructor C_i ($i = 1 \dots n$) tendrá su correspondiente aridad $ar(C_i)$. Entonces el término $C_i(t_1, \dots, t_{ar(C_i)})$ se puede representar como:

$$\lambda x_1 \dots x_n. x_i t_1 \dots t_{ar(C_i)}$$

En nuestro ejemplo tenemos dos constructores, Nil de aridad 0 y Cons de aridad 2. Por lo tanto los términos quedan representados de la siguiente manera:

$$\begin{aligned} \text{Nil} &\equiv \lambda x y. x \\ \text{Cons } A \ B &\equiv \lambda x y. yAB \end{aligned}$$

Otros ejemplos

Los booleanos, y los pares quedan representados en su forma tradicional

```
data Bool = True | False
data P a b = Pair a b
```

$$\begin{aligned} \text{True} &\equiv \lambda x y. x \\ \text{False} &\equiv \lambda x y. y \\ \text{Pair } A \ B &\equiv \lambda x. xAB \end{aligned}$$

El operador case

Mediante esta codificación el case se puede representar de la siguiente manera:

```
case E of Nil ->F ; Cons A B ->G  $\equiv EF(\lambda ab.G)$ 
```

Si interpretamos el if-then-else como una caso particular del case sobre booleanos obtenemos la representación usual:

```
case E of True ->F; False ->G  $\equiv EFG \equiv \text{if } E \text{ then } F \text{ else } G$ 
```