Programació funcional

Albert Rubio

Especialitat de Computació
Grau en Enginyeria Informàtica

FIB

Pla de la sessió

- Sistema de tipus de Haskell
- Tipus predefinits
- Tipus polimòrfics.
- Creació de tipus de dades
- Classes de tipus
- Comprovació/Inferència de tipus

Sistema de tipus en Haskell

- Tipus predefinits
- Tipus definits per l'usuari
- Polimorfisme paramètric
- Formes més potents de polimorfisme
- Classes de tipus y la sobrecàrrega
- Instanciació de classes.

Tipus predefinits

Com ja hem vist existeixen una sèrie de tipus predefinits:

- Int, Integer,...
- Char
- Bool
- Funcions: a -> ... -> a
- Llistes, Tuples

```
    5  :: Integer
   'a' :: Char
   inc :: Integer -> Integer
[1,2,3] :: [Integer]
('b',4) :: (Char,Integer)
```

Tipus predefinits

Tenim altres tipus predefinits, per exemple per manegar errors:

```
data Maybe a = Just a | Nothing
```

Tenim opercions com ara

```
isJust :: Maybe a -> Bool
isNothing :: Maybe a -> Bool
fromJust :: Maybe a -> a
fromMaybe :: a -> Maybe a -> a Source
```

Per exemple podem definir

```
arrel:: Arbre a -> Maybe a
arrel Abuit = Nothing
arrel (Node x _ _) = Just x
```

Tipus polimòrfics

Utilització de variables de tipus que poden pendre qualsevol valor.

```
mida :: [a] -> Integer
```

Les variables de tipus estan implicitament quantificades universalment.
Noteu que ens permet escriure llistes de qualsevol tipus, però no llistes heterogènies.

Aquest tipus de polimorfisme s'anomena *polimorfisme paramètric*.

ghc ens permet usar el constructor de tipus `forall` explicitament, usant per exemple el flag `-XExplicitForAll`

```
mida :: forall a. [a] -> Integer
```

Creació de tipus de dades

```
data Color = Vermell | Verd | Blau | Violat
```

Es poden definir tipus recursius com `Arbe`

Alternativa

```
newtype Dolars = Dolars Int

llistaDolars :: [Dolars]
llistaDolars = [Dolars 3, Dolars 27]
```

És com data quan hi ha un únic constructor de tipus.

Tot newtype es pot escriure com data, però no el contrari

```
data Dolars = Dolars Int
```

Creació de tipus de dades

No s'ha de confondre amb els Tipus sinònims:

```
type String  = [Char]
type Person  = (Name,Address)
type Name    = String
type AssocList a b  = [(a,b)]
```

Són com els typedef de C++

És a dir, els dos tipus són intercanviables.

Creació de tipus polimòrfics amb forall

ghi ens permet també usar el `forall` dins de les definicions de tipus de dades amb constructors.

```
data ShowBox = forall s. SB s

heteroList :: [ShowBox]
heteroList = [SB 'a', SB 5, SB True]
```

Per poder usar-lo s'ha d'activar per exemple el flag `-XExistentialQuantification`

D'aquesta manera podem treballar amb llistes heterogènies.

Classes de tipus

Les classes de Haskell no són tipus, sino categories de tipus.
(Type classes) predefinides

```
class Eq a where
  (==) :: a -> a -> Bool
elem  :: (Eq a) => a -> [a] -> Bool
```

És un mecanisme similar als Interfaces de Java.

- És la forma de tenir sobrecàrrega en Haskell.
- És una altre forma de polimorfisme.

Classes de tipus

```
class  Eq a  where
  (==), (/=)                  :: a -> a -> Bool
  x /= y                      =  not (x == y)
class  (Eq a) => Ord a  where
  (<), (<=), (>=), (>)  :: a -> a -> Bool
  max, min                    :: a -> a -> a
  x < y                    =  x <= y && x /= y
```

Altres classes

Show, Read,..

Classes de tipus

Instanciació:

```
instance Eq a => Eq (Arbre a) where
  x == y                    =  x ‘iguals‘ y
```

Instanciació predefinida: *deriving*

```
data Carta = Two | Three | Four
    | Five | Six | Seven | Eight | Nine | Ten
    | Jack | Queen | King | Ace
   deriving (Read,  Show, Enum, Eq, Ord)
```

Comprovació/Inferència de tipus

- En anglès “Type checking”

  És l’acció de comprovar si un terme té un determinat tipus

  - Dinàmic (en temps d’execució).
  - Estàtic (en temps de compilació).

- En anglès “Type inference”

  És l’acció d’inferir el tipus d’un terme a partir d’informació de contexte

  - declaracions
  - aplicacions de funcions

  Intenta inferir el tipus més general.