Arrows

J.F. Cardona McCormick

Universidad EAFIT

24 de Julio de 2009

Agenda

Introducción

- Arrow son una generalización de las Mónadas.

Introducción

- Arrow son una generalización de las Mónadas.
- Suministran una estructura común para bibliotecas.

Introducción

- Arrow son una generalización de las Mónadas.
- Suministran una estructura común para bibliotecas.
- Implementan una noción de computación que es parcialmente estática (Independiente de la entrada).

Introducción

- Arrow son una generalización de las Mónadas.
- Suministran una estructura común para bibliotecas.
- Implementan una noción de computación que es parcialmente estática (Independiente de la entrada).
- Manejan múltiples entradas.

Conceptos básicos

- Funciones:

```
f :: a → b → c
```

Conceptos básicos

- Funciones:

```
f :: a → b → c
```

- Combinadores:

```
<|> :: a → a → a
```

Conceptos básicos

- Funciones:

```
f :: a → b → c
```

- Combinadores:

```
<|> :: a → a → a
```

- Tipos de datos algebraicos.

```
data Árbol a = Hoja a | Nodo (Árbol a) (Árbol a)
```

Conceptos básicos

- Funciones:

```
f :: a → b → c
```

- Combinadores:

```
<|> :: a → a → a
```

- Tipos de datos algebraicos.

```
data Árbol a = Hoja a | Nodo (Árbol a) (Árbol a)
```

- Clases son agrupaciones de tipos.

Conceptos básicos

- Funciones:

```
f :: a → b → c
```

- Combinadores:

```
<|> :: a → a → a
```

- Tipos de datos algebraicos.

```
data Árbol a = Hoja a | Nodo (Árbol a) (Árbol a)
```

- Clases son agrupaciones de tipos.
- Monadas

```
class Monad m where
   (>>=) :: m a -> (a -> m b) -> m b
   (>>) ::  m a -> m b -> m b
   return :: a -> m a
   fail :: String -> m a
```

Clases base

- Clase `cat`

```
class Category cat where
  id  :: cat a a
  (.) :: cat b c -> cat a b -> cat a c
```

Clases base

- Clase `cat`

```
class Category cat where
  id  :: cat a a
  (.) :: cat b c -> cat a b -> cat a c
```

- Métodos de enlace

```
(<<<) :: Category cat
         => cat b c -> cat a b -> cat a c
(>>>) :: Category cat
         => cat a b -> cat b c -> cat a c
```

Clase Arrow

- Clase Arrow

```
class Category a => Arrow a where
  arr    :: (b -> c) -> a b c
  first  :: a b c -> a (b, d) (c, d)
  second :: a b c -> a (d, b) (d, c)
  (***)  :: a b c -> a b' c' -> a (b, b') (c, c')
  (&&&)  :: a b c -> a b c' -> a b (c, c')
```

Clase Arrow

- Clase Arrow

```
class Category a => Arrow a where
  arr    :: (b -> c) -> a b c
  first  :: a b c -> a (b, d) (c, d)
  second :: a b c -> a (d, b) (d, c)
  (***)  :: a b c -> a b' c' -> a (b, b') (c, c')
  (&&&)  :: a b c -> a b c' -> a b (c, c')
```

- Representa una abstración de computación.

Clase Arrow

- Clase Arrow

```
class Category a => Arrow a where
  arr    :: (b -> c) -> a b c
  first  :: a b c -> a (b, d) (c, d)
  second :: a b c -> a (d, b) (d, c)
  (***)  :: a b c -> a b' c' -> a (b, b') (c, c')
  (&&&)  :: a b c -> a b c' -> a b (c, c')
```

- Representa una abstración de computación.
- Identifica los elementos de entrada y salida de la computación.

Clase Arrow

- Clase Arrow

```
class Category a => Arrow a where
  arr    :: (b -> c) -> a b c
  first  :: a b c -> a (b, d) (c, d)
  second :: a b c -> a (d, b) (d, c)
  (***)  :: a b c -> a b' c' -> a (b, b') (c, c')
  (&&&)  :: a b c -> a b c' -> a b (c, c')
```

- Representa una abstración de computación.
- Identifica los elementos de entrada y salida de la computación.
- Generalizan la computación realizada por una función y la transforma en un proceso.

Instancias

```
instance Arrow (->) where
    arr g = g
    first g (x,y) = (g x, y)
```

Instancias

```
newtype Kleisli m a b =
          Kleisli { runKleisli :: a -> m b }

instance (Monad m) =>  Arrow (Kleisli m) where
    arr f = Kleisli (return . f)
    first (Kleisli f) =
          Kleisli ( ~(b,d) -> do c <- f b
                                 return (c,d) )
```

Clase `ArrowChoice`

- La computación con las Mónadas es inflexible, una vez a tomado un camino sigue por este.

Clase `ArrowChoice`

- La computación con las Mónadas es inflexible, una vez a tomado un camino sigue por este.
- Las instancias de clase `ArrowChoice` permite una computación flexible dependiendo del contenido de las previas entradas.

Clase `ArrowChoice`

- La computación con las Mónadas es inflexible, una vez a tomado un camino sigue por este.
- Las instancias de clase `ArrowChoice` permite una computación flexible dependiendo del contenido de las previas entradas.
- El tipo de dato algebraico `Either`

```
data Either a b = Left a | Right b
```

Clase ArrowChoice

- La computación con las Mónadas es inflexible, una vez a tomado un camino sigue por este.
- Las instancias de clase `ArrowChoice` permite una computación flexible dependiendo del contenido de las previas entradas.
- El tipo de dato algebraico `Either`

```
data Either a b = Left a | Right b
```

- Clase `Arrow Choice`

```
class Arrow a => ArrowChoice a where
  left  :: a b c -> a (Either b d) (Either c d)
  right :: a b c -> a (Either d b) (Either d c)
  (+++) :: a b c -> a b' c'
            -> a (Either b b') (Either c c')
  (|||) :: a b d -> a c d
            -> a (Either b c) d
```

Clase `Arrow Apply`

- Permite utilizar arrows que son resultados de previas computación como una función de alto orden.
- Clase `Arrow Apply`

```
class Arrow a => ArrowApply a where
  app :: a (a b c, b) c
```

Clase `Arrow Loop`

- Esta clase describe Arrows que pueden utilizar la recursión para computar resultados.
- Clase `Arrow Loop`

```
class Arrow a => ArrowLoop a where
   loop :: a (b, d) (c, d) -> a b c
```