Programación Declarativa 2020-2
Combinadores Monádicos

Javier Enríquez Mendoza Favio E. Miranda Perea

Facultad de Ciencias UNAM

4 de junio de 2020

`Control.Monad`

La biblioteca `Control.Monad` contiene algunas funciones definidas para el uso de mónadas.

Estas funciones son muy útiles para facilitar la programación monádica.

Estas funciones se conocen como combinadores monádicos.

(=<<)

El operador (=<<) es equivalente al operador bind pero recibe los parámetros en un orden inverso.

El tipo de este operador es:

```
(=<<) :: (a -> m b) -> m a -> m b
```

Y está implementado de la siguiente forma:

```
(=<<) = flip (>>=)
```

mapM, filterM, foldM

Las funciones `mapM`, `filterM` y `foldM` son nuestros viejos amigos `map` `filter` y `foldl` pero envueltas en mónadas

Sus tipos son:

```
mapM :: (a -> m b) -> [a] -> m [b]
filterM :: (a -> m Bool) -> [a] -> m [a]
foldM :: (a -> b -> m a) -> a -> [b] -> m a
```

Existe una versión alternativa de `mapM` llamada `mapM_` y la única diferencia es que está no regresa ningún valor. Entonces su tipo es

```
mapM_ :: (a -> m b) -> [a] -> m ()
```

Estos combinadores son especialmente usados con `IO`.

sequence

El combinador `sequence` simplemente se encarga de ejecutar secuencialmente una lista de cómputos.

Su tipo es:

```
sequence :: [m a] -> m [a]
```

También existe una versión alternativa que hace lo mismo sin regresar ningún valor llamada `sequence_`. Con el tipo

```
sequence_ :: [m a] -> m ()
```

`liftM`

La función `liftM` toma una función no monádica y la *eleva* a una función monádica.

Esta función tiene el tipo:

```
liftM :: (a -> b) -> m a -> m b
```

Esta función es principalmente utilizada para reducir el código.

when

Está función ejecuta una acción monádica solo cuando se cumple una condición.

Su tipo es:

```
when :: Bool -> m () -> m ()
```

Su comportamiento se puede simular también con `filterM` pero a veces es mas legible y conveniente `when`

Puede verse como un `if` manco.

join

El combinador `join` es el equivalente monádico a `concat` de listas.

De hecho para la mónada lista `join` hace lo mismo que concat, para el resto de las mónadas se comporta de forma similar.

Su tipo es:

```
join :: m (m a) -> m a
```

Este es uno de los combinadores mas importantes, especialmente cuando se habla de transformaciones monádicas.