CURSO DE PROGRAMACION SCALA
Sesión 13

Sergio Couto Catoira

Índice

- Continuación de gestión del estado

Ejercicio:

- Del mismo modo que hemos definido map2 para combinar dos estados, define una función para combinar una lista de transiciones a una única.

- Usa la función anterior para redefinir ints
  - Usa el método List.fill(n)(x) que crea una lista de x con tamaño n

Ejercicio:

- Del mismo modo que hemos definido map2 para combinar dos estados, define una función para combinar una lista de transiciones a una única.

- Usa la función anterior para redefinir ints
  - Usa el método List.fill(n)(x) que crea una lista de x con tamaño n

Ejercicio:

- Define el método flatMap
  - def flatMap[A,B](f: Rand[A])(g: A => Rand[B]): Rand[B]

- Usa la función anterior para definir una función que devuelva un entero no negativo menor a un número dado
  - def nonNegativeLessThan(n: Int): Rand[Int]

Ejercicio:

- Reimplementa las funciones map y map2 con flatMap

- Usando las funciones definidas en la clae RNG, implementa una función rollDice que simule el lanzamiento de un dado. Debe devolver un numero aleatorio entre 1 y 6 (ambos incluidos)

Estado

- Define un tipo State que generalice Rand. Hazlo covariante en el valor.
  - type State[S, +A] = ???

- State es una función que *transporta* un estado al ejecutarse.
- Podría crearse como case class en lugar de como tipo
- Podría redefinirise Rand como un subtipo de State
  - type Rand[A] = State[RNG, A]

Estado

- Define las funciones unit, map, map2 , flatMap y sequence para State. No todas podrán incluirse en la case class, 2 de ellas tendrás que meterlas en un companion object

- Si no definiste map y map2 usando flatMap, hazlo.