Tarea 4
CC4101 - Lenguajes de Programación

Éric Tanter
Auxiliar: Richard Ibarra

Fecha de Entrega. Viernes 4 de Junio *

1 Streams

En clase hemos visto como la evaluación perezosa permite trabajar con estructuras infinitas. En un lenguaje con evaluación temprana, es posible definir valores que juegan el papel de una estructura infinita: se llaman *streams* (flujos). Un stream esta definido por 2 operaciones, `head` y `tail`, que permiten respectivamente obtener el elemento en el tope del stream, y obtener el stream una vez consumido el primer elemento. Eso claramente se parece a `car` y `cdr`, pero a la diferencia de listas en Scheme, un stream se evalua perezosamente, a medida que uno va consumiendo sus elementos.

Se obtiene ese comportamiento considerando que un stream es un conjunto de tres elementos (hint: `define-type`):

- un estado actual (de tipo a).
- una función $a \rightarrow b$ que, dado el estado actual, retorna el elemento en el tope del stream.
- una función $a \rightarrow stream$, que, dado el estado actual, retorna el stream en su siguiente estado.

1. (1pt) Defina, en PLAI Scheme, una librería de streams que respete la siguiente interfaz:
   - `(stream state hd tl)`: construye un stream con estado inicial `state`, y funciones de cabeza y cola `hd` y `tl`.
   - `(head stream)`: retorna el elemento en el tope del stream.
   - `(tail stream)`: retorna el stream en su siguiente estado.
   - `(take n stream)`: retorna la lista con los `n` primeros elementos de `stream`.
2. (1pt) Defina los siguientes streams: `ones` (stream infinito de 1), `ints` (stream infinito de los enteros), y `reads` (stream infinito de input del usuario). Ejemplos:

```
> (take 10 ones)
(1 1 1 1 1 1 1 1 1 1 )
> (take 8 ints)
(0 1 2 3 4 5 6 7)
> (take 2 reads)
...input1...
...input2...
(input1 input2)
```

*Ver las reglas de entrega y evaluación en `http://pleiad.cl/teaching/cc4101/reglas`.

2 Call-by-Need and Call-by-Name

La evaluación perezosa que vimos en Haskell también se llama "call-by-need" (llamada por necesidad). Es decir que se evalua la expresión de argumento a lo más una vez, solamente si se necesita.

Existe otra forma de hacer evaluación perezosa conocida como "call-by-name" (llamada por nombre), un mecanismo introducido hace mucho tiempo en el lenguaje ALGOL. Ese mecanismo está teniendo una nueva juventud ahora con su integración en el lenguaje Scala[1].

1. (0pt) Lea *atentativamente* la descripción del mecanismo de call-by-name en Scala:
   `http://www.scala-lang.org/node/138`
   Asegurese de entender bien el primer ejemplo (`whileLoop`).

2. (1pt) Introduzca una sintaxis concreta para definir funciones cuyo parametro sera pasado usando la semántica de call-by-name. Demuestre como usaría su lenguaje (RCFAE extendido) para definir y usar la función `whileLoop` del ejemplo de Scala (asumiendo que tienen `set!` en el lenguaje).

3. (2pt) Implemente el soporte para call-by-name en su interprete RCFAE de tal forma de poder correr el ejemplo anterior. (Como el RCFAE no soporta mutación, agregue `read` y/o `random` y uselos en sus ejemplos.)

[1] Scala es un nuevo lenguaje para la JVM que combina programación funcional y por objeto; o sea, lo mejor de ambos mundos, además de una buena integración con Java. Scala tiene varias caracteristicas que lo hacen muy interesante y varias empresas ya están migrando a Scala (como Twitter).
Ver `http://www.scala-lang.org`