Tarea 5
CC4101 - Lenguajes de Programación

Éric Tanter
Auxiliar: Richard Ibarra

Fecha de Entrega. Viernes 25 de Junio *

1 Monitorear Streams con Automatas (6 pts)

En las tareas anteriores hemos visto automatas y flujos. En esta tarea, vamos a combinar ambos conceptos, para definir automatas que reaccionen al contenido consumido en streams.

1. (1.5pt) Estudie la definición de automatas *con macros* tal como explicado en el capitulo 37 del PLAI. Extienda la definición de un automata tal que, a cada transición, se pueda asociar una expresión que evaluar. Por ejemplo:

```
(define x (automaton a
            (a : (0 -> b | (printf "going to state b~n"))
            (b : (1 -> a | (printf "going to state a~n"))))))
> (x '(0 1 0))
going to state b
going to state a
going to state b
```

2. (1.5pt) Cambie su definición de automatas para que se le pueda entregar su input progresivamente, en vez de una lista:

```
(define x ...)
> (x '0)
going to state b
> (x '1)
going to state a
```

3. (1.5pt) Ahora queremos integrar automatas y streams. En particular, queremos poder envolver un stream con un automata, tal que cada vez que se consume del stream, el valor consumido pasa por el automata.

```
(define (combine stream automata) ...)

> (define (stream-0-1 ...) ;; stream infinito de 0 y 1
> (define y (combine stream-0-1 x))
> (take 2 y)
going to state b
going to state a
(0 1)
```

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

4. (1.5pt) Extienda su definición de automatas tal que uno pueda usar cualquier predicado en la parte izquierda de una transición (se testean en orden de definición). El input leido del stream se debe poder referir usando el nombre `val`, tanto en la condición como en la expresión asociada a la transición. Además, usar `(return res)` en la expresión asociada a una transición hace que el stream combinado retorne `res` en vez del valor realmente consumido del stream.

```
(define x (automaton a
            (a : ((val < 10) -> b | (printf "going to state b~n"))
            (b : ((val < 5)  -> a | (printf "going to state a~n"))
                 ((val >= 5) -> b | (begin (printf "invalid input~a~n" val)
                                           (return #f)))))))
> (define st ...) ;; stream 5 4 10 6 7 2 ...
> (define y (combine st x))
> (take 6 y)
going to state b
going to state a
going to state b
invalid input 7
going to state a
(5 4 10 6 #f 7)
```

2 Call-by-Reference (2 pts)

Ejercicio 14.3.2 del PLAI. (Lea con atención la sección 14.3 del libro antes de contestar las distintas preguntas.)

Esta pregunta es opcional. Le puede servir para acumular hasta 2 puntos extras en tareas.