Parcial Número I de Leguajes II

Profesor: Néstor Cataño C.

Programación en el Lambda-Calculus

Suponga que pair es el término lambda para parejas, suponga que first es el término lambda para primer elemento de una pareja y second para el segundo. Estos números son llamados números de Church. Considere además la existencia de números $\underline{0} \equiv \lambda x.\lambda y.\, y$, $\underline{1} \equiv \lambda x.\lambda y.\, xy$, $\underline{2} \equiv \lambda x.\lambda y.\, x\,(x\,y)$, etc. Considere la definición de los términos lambda zz y ss como se introducen a continuación:

$$\underline{zz} \equiv \underline{pair}\ \underline{0}\ \underline{0}$$
$$\underline{ss} \equiv \lambda p.\ \underline{pair}\ (\underline{second}\ p)\ (\underline{+}\ (\underline{second}\ p)\ \underline{1})$$

1. (a) A qué valor evalúa (ss ww), si ww ≡ zz ?
   (b) A qué valor evalúa (ss ww), si ww ≡ (pair 0 1) ?
   (c) A qué valor evalúa (ss ww), si ww ≡ (pair 1 2) ?
   (d) A qué valor evalúa (first (1 ss zz))?
   (e) A qué valor evalúa (first (2 ss zz))?
   (f) A qué valor evalúa (first (3 ss zz))?
   (g) A qué valor evalúa (first (12354 ss zz))?
2. Suponga que kk ≡ λn. first (n ss zz). Utilice kk para escribir un término lambda que tome dos números de Church y devuelva T si los números son iguales y F si no lo son.

Programación en Scheme

1. En clase vimos la definición de `make-stack` en Scheme para la creación de una pila la cual implementaba las funciones `push`, `top`, `pop` y `empty?` así:

```
(define make-stack
  (lambda ()
    (let ((ls '()))
       (lambda (msg . args)
         (cond
           ((eqv? msg 'empty?) (null? ls))
           ((eqv? msg 'push!) (set! ls (cons (car args) ls)))
           ((eqv? msg 'top) (car ls))
           ((eqv? msg 'pop!) (set! ls (cdr ls)))
           (else "oops")))))))
```

De tal manera que:

```
(define mystack1 (make-stack))
(mystack1 'empty?)              ⟹ #t
(mystack1 'push! 'a)
(mystack1 'empty?)              ⟹ #f
(mystack1 'push! 'b)
(mystack1 'push! 'c)
(mystack1 'top)                 ⟹ c
(mystack1 'pop!)
(mystack1 'top)                 ⟹ b
```

Modifique la función `make-stack` para que acepte un tipo de mensaje `pop2` de tal forma que `(mystack1 'pop2 n)` remueve el `n`-ésimo elemento de la pila `mystack1` empezando desde el tope. Así por ejemplo `(mystack1 pop2 1)` remueve el elemento tope de la pila `mystack1`. Usted deberá escribir el código completo de la función `make-stack` el cual incluya el código para `pop2`.

2. Escriba en Scheme una functión `maximo` que cálcule el máximo valor de una lista. Por ejemplo:

   `(maximo '(2 7 5 4))` $\Longrightarrow$ `7`
   `(maximo '(9 3 0 8))` $\Longrightarrow$ `9`

   (Nota: No se podrá utilizar la función stándard `max` de Scheme)