

Lenguajes de Programación (CC41A) - Primavera 2009

Clase Auxiliar 7

Profesor: Tomás Barros

Auxiliar: Víctor Ramiro

1. Describa la evaluación del siguiente programa en el lenguaje VBCFAE detallando en cada paso qué pasa con el Environment y con el Store. ¿Cuál es el resultado obtenido?

Sea claro al describir la evolución Environment/Store. Para ello **DEBE** usar diagramas.

```
{with {{x {box 100}}
      {y 20}}
  {with {{f {fun {z} {+ z {+ {unbox x} y}}}}}
    {seqn
      {set-box! x 200}
      {set! y 30}
      {with {{y 5}}
        {f 1}}}}}
```

2. Considere el siguiente intérprete de un lenguaje con estructuras de datos mutables (box), escrito usando *store-passing style* :

```
(define (interp expr env store)
  (type-case Expr expr
    ...
    [app (fun-expr arg-expr)
      (type-case Value*Store (interp fun-expr env store)
        [v*s (fun-value fun-store)
          (local ([arg-value (interp arg-expr env fun-tore)]
                  [define new-loc (next-location fun-store)])
            (interp (closureV-body fun-value)
                    (aSub (closureV-param fun-value)
                          new-loc
                          (closureV-env fun-value))
                    (aSto new-loc
                          arg-value
                          fun-store))) .)])
```

- a) ¿Qué problema tiene este intérprete?
 - b) Escriba un programa que ilustre que la semántica del **app** es errónea.
 - c) Modifique el intérprete para que funcione adecuadamente.
3. En clases vimos un tipo de dato abstracto contador, que solamente permitía incrementar el contador. Definimos, usando una *representación procedural* de un contador, las funciones **new-counter** y **inc** tal que:

```

> (define c (new-counter))
> (inc c)
1
> (inc c)
2
> (inc c)
3
...

```

- a) Esta forma de proceder tiene la limitante de que no es posible hacer más de una sola cosa con una función: solo se puede aplicar. Entonces, ¿cómo podemos representar un contador en forma procedural, que permita a la vez incrementar y decrementar?

```

> (define c (new-counter))
> (inc c)
1
> (inc c)
2
> (dec c)
1
> (dec c)
0

```

Modifique su definición de `new-counter` y de `inc`, de modo de acomodar la nueva funcionalidad, y defina `dec`.

Hint: el problema consiste en hacer que aplicar una función pueda tener distintos efectos (como incrementar o decrementar el estado de la función). Recuerde que es posible definir funciones en un scope anidado.

- b) Usando la misma técnica, de una representación procedural del tipo de dato abstracto pila siguiendo la siguiente especificación:
- `(new-stack)`: retorna una nueva pila vacía.
 - `(push st val)`: agrega `val` en la pila `st`.
 - `(pop st)`: saca el elemento encima de la pila, y lo retorna.
 - `(peek st)`: retorna el valor encima de la pila `st`, sin modificarla.

4. Un poco más de lambda cálculo.

En esta pregunta van a ver como codificar los booleanos y condicionales. El λ -cálculo es un lenguaje que solo cuenta con identificadores x , funciones (abstracciones) $\lambda x.e$, y aplicaciones $(e\ e)$.

Además, la reducción de expresiones se hace usando la siguiente regla (llamada “reducción β ”):

$$((\lambda x.e_1)\ e_2) \longrightarrow_{\beta} e_1[x \leftarrow e_2]$$

(donde $e_1[x \leftarrow e_2]$ denota la substitución, en e_1 , de todas las ocurrencias libres de x por e_2).

- a) La reducción β especifica un régimen de evaluación temprana o perezosa? ¿por qué?
- b) La codificación de **true** y **false** es muy simple:

$$\begin{aligned}\mathbf{true} &= \lambda x.\lambda y.x \\ \mathbf{false} &= \lambda x.\lambda y.y\end{aligned}$$

Defina **if** de tal manera que respete la semántica esperada:

$$\begin{aligned}\mathbf{if\ true\ } e_1\ e_2 &= e_1 \\ \mathbf{if\ false\ } e_1\ e_2 &= e_2\end{aligned}$$

Ilustre que su definición de **if** es correcta detallando los pasos de evaluación para ambas ecuaciones.

- c) Análogo al punto anterior. Defina **and** y **or** siguiendo la semántica booleana.
- d) ¿Sus definiciones podrían ser directamente transpuestas a un lenguaje como Scheme? ¿Haskell? ¿por qué?