Auxiliar Once - CC41A
Lenguajes de Programación
IMutación

Oscar E. A. Callaú
oalvarez@dcc.uchile.cl

Santiago - Chile, 3/Nov/2008

1. Store (cont.)

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
     ...
     [if0 (test then else)
         (if (num-zero? (interp test env store))
             (interp then env store)
             (interp else env store))) ...))
```

Recordemos que como ahora podemos inducir cambios de estado en nuestro lenguaje, éstas son expresiones válidas y por lo tanto pueden ser usados en cualquier parte de nuestro programa. En particular dentro del `test`, pueden provocar cambios de estado; luego tal cual está el intérpete no se *traspasa* esa información a las ramificaciones del `if0`.

```
{with {b {newbox 0}}
      {if0 {seqn {setbox b 5} {openbox b}}
           1
           {openbox b}}} -> 0! y queremos 5!.
```

Como muestra el anterior ejemplo, la implementación es incorrecta, modifiquela para que cumpla con su propósito.

2. Variables Mutables (cont.)

De las siguientes expresiones, justifique los resultados que obtienen al ser evaluadas, detallando cada paso del `Env` y el `Store`:

1. ```
   {with {init {fun {a}
                    {setbox a 0}}}
         {with {z 5}
               {with {x {box z}}
                     {seqn
                      {init x}
                      {set z 10}
                      {unbox x}}}}}
   ```

   Como vimos en clases anteriores el resultado de la evaluación es: 0. Justifique esta respuesta con diagramas.

2. ```
   {with {setter {refun {x}
                        {set x 41}}}
         {with {y 10}
               {with {a {newbox y}}
                     {seqn
                      {setter y}
                      {+ {openbox a} y}}}}}
   ```

   Como vimos en clases anteriores el resultado de la evaluación es: 51 (consideramos que el paso `.. {with {a {newbox y}} ..}` la subexpresión `y` da como resultado su valor evaluado como `(numV ..)`). Justifique esta respuesta con diagramas.

3. (nuevo)

   ```
   {with {setter-box! {fun {b}
                           {set-box! b 41}}}
         {with {setter-var! {fun {v}
                                 {set! v 5}}}
               {with {x 3}
                     {with {b {box x}}
                           {seqn {setter-box! b}
                                 {setter-var! x}
                                 {+ {openbox b} x}}}}}}
   ```