Facultad de Ciencias, UNAM
Lenguajes de Programación
Tarea 5

Rubí Rojas Tania Michelle

07 de diciembre de 2020

1. Evalúa la siguiente expresión usando el tipo de alcance y régimen de evaluación que se indica. Es necesario incluir el ambiente final en forma de pila en cada caso.

```
{with {a {+ 2 2}}
   {with {b {+ a a}}
      {with {foo {fun {x} {- x b}}
         {with {a {- 2 2}}
            {with {b {- a a}}
               {foo -3}}}}}}}
```

   a) Alcance estático y evaluación glotona
   Solución: La expresión que debemos evaluar es `{foo -3}`, por lo que

$$\begin{aligned}
\texttt{\{foo -3\}} &= \texttt{\{\{fun \{x\} \{- x b\}\} -3\}} \\
&= \texttt{\{- (-3) b\}} \\
&= \texttt{\{- (-3) 8\}} \\
&= -11
\end{aligned}$$

b	0
a	0
foo	[closureV: x,{-x b}, env-ant:((x -3),(b 8),(a 4))]
b	8
a	4

Tabla 1: Ambiente final

   b) Alcance dinámico y evaluación glotona
   Solución: La expresión que debemos evaluar es `{foo -3}`, por lo que

$$\begin{aligned}
\texttt{\{foo -3\}} &= \texttt{\{\{fun \{x\} \{- x b\}\} -3\}} \\
&= \texttt{\{- (-3) b\}} \\
&= \texttt{\{- (-3) 0\}} \\
&= \texttt{-3}
\end{aligned}$$

x	-3
b	0
a	0
foo	{fun {x} {- x b}}
b	8
a	4

Tabla 2: Ambiente final

c) Alcance estático y evaluación perezosa
SOLUCIÓN: La expresión que debemos evaluar es `{foo -3}`, por lo que

```
{foo -3} = {{fun {x} {- x b}} -3}
         = {- (-3) b}
         = {- (-3) {+ a a}}
         = {- (-3) {+ {+ 2 2} {+ 2 2}}}
         = {- (-3) {+ 4 4}}
         = {- (-3) 8}
         = -11
```

b	{- a a}
a	{- 2 2}
foo	[closureV: x,{-x b}, env-ant:((x -3),(b {+ a a}), (a {+ 2 2}))]
b	{+ a a}
a	{+ 2 2}

Tabla 3: Ambiente final

d) Alcance dinámico y evaluación perezosa
SOLUCIÓN: La expresión que debemos evaluar es `{foo -3}`, por lo que

```
{foo -3} = {{fun {x} {- x b}} -3}
         = {- (-3) b}
         = {- (-3) {- a a}}
         = {- (-3) {- {- 2 2} {- 2 2}}}
         = {- (-3) {- 0 0}}
         = {- (-3) 0}
         = -3
```

x	-3
b	{- a a}
a	{- 2 2}
foo	{fun {x} {- x b}}
b	{+ a a}
a	{+ 2 2}

Tabla 4: Ambiente final

2. Dada la siguiente función:

```
(define (goo l)
    (if (empty? l)
        empty
        (append (car l) (goo (cdr l)))))
```

a) Explica qué hace y dale un nombre mnemotécnico.
SOLUCIÓN: Por cómo está definida la función `goo`, ésta debe recibir una lista de listas; por lo que `goo` hará la concatenación de las listas de la lista l, es decir, regresa una lista con todos los elementos de las listas de la lista l de acuerdo a su órden de aparición en su respectiva lista. Así, un nombre mnemotécnico para esta función podría ser `concatena-listas-de-lista`. Por lo tanto, nuestra función queda de la siguiente forma:

```
(define (concatena-listas-de-lista l)
    (if (empty? l)
        empty
        (append (car l) (concatena-listas-de-lista (cdr l)))))
```

b) Muestra los registros generados cuando es llamada con el argumento `'((1 2) (3 4) (4 6))`. ¿Cuántos registros son generados? ¿Cuántos son ocupados a la vez?

SOLUCIÓN: Veamos que se generan 8 registros de activación (4 de entrada y 4 de salida), y ningún registro es ocupado a la vez.

Ingresa `(concatena-listas-de-lista '((1 2) (3 4) (4 6)))`

```
(append '(1 2) (concatena-listas-de-lista '((3 4) (4 6))))
                              ...
                    (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                    '((1 2) (3 4) (4 6))
                  concatena-listas-de-lista
```

Ingresa `(concatena-listas-de-lista '((3 4) (4 6)))`

```
   (append '(3 4) (concatena-listas-de-lista '((4 6))))
                              ...
                    (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                       '((3 4) (4 6))
                  concatena-listas-de-lista
```

```
(append '(1 2) (concatena-listas-de-lista '((3 4) (4 6))))
                              ...
                    (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                    '((1 2) (3 4) (4 6))
                  concatena-listas-de-lista
```

Ingresa `(concatena-listas-de-lista '((4 6)))`

```
     (append '(4 6) (concatena-listas-de-lista '()))
                              ...
                    (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                          '((4 6))
                  concatena-listas-de-lista
```

```
   (append '(3 4) (concatena-listas-de-lista '((4 6))))
                              ...
                    (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                       '((3 4) (4 6))
                  concatena-listas-de-lista
```

```
(append '(1 2) (concatena-listas-de-lista '((3 4) (4 6))))
                              ...
                    (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                    '((1 2) (3 4) (4 6))
                  concatena-listas-de-lista
```

Ingresa `(concatena-listas-de-lista '())`

```
                          '()
                          ...
                  (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                          '()
                concatena-listas-de-lista
```

```
     (append '(4 6) (concatena-listas-de-lista '()))
                          ...
                  (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                        '((4 6))
                concatena-listas-de-lista
```

```
   (append '(3 4) (concatena-listas-de-lista '((4 6))))
                          ...
                  (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                     '((3 4) (4 6))
                concatena-listas-de-lista
```

```
(append '(1 2) (concatena-listas-de-lista '((3 4) (4 6))))
                          ...
                  (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                  '((1 2) (3 4) (4 6))
                concatena-listas-de-lista
```

Sale `(concatena-listas-de-lista '())`

```
                   (append '(4 6) '())
                          ...
                  (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                        '((4 6))
                concatena-listas-de-lista
```

```
   (append '(3 4) (concatena-listas-de-lista '((4 6))))
                          ...
                  (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                     '((3 4) (4 6))
                concatena-listas-de-lista
```

```
(append '(1 2) (concatena-listas-de-lista '((3 4) (4 6))))
                          ...
                  (if (empty? l) empty
  (append (car l) (concatena-listas-de-lista (cdr l))))
                  '((1 2) (3 4) (4 6))
                concatena-listas-de-lista
```

Sale `(concatena-listas-de-lista '((4 6)))`

```
(append '(3 4) '(4 6))
...
(if (empty? l) empty
(append (car l) (concatena-listas-de-lista (cdr l))))
'((3 4) (4 6))
concatena-listas-de-lista
```

```
(append '(1 2) (concatena-listas-de-lista '((3 4) (4 6))))
...
(if (empty? l) empty
(append (car l) (concatena-listas-de-lista (cdr l))))
'((1 2) (3 4) (4 6))
concatena-listas-de-lista
```

Sale `(concatena-listas-de-lista '((3 4) (4 6)))`

```
(append '(1 2) '(3 4 4 6))
...
(if (empty? l) empty
(append (car l) (concatena-listas-de-lista (cdr l))))
'((1 2) (3 4) (4 6))
concatena-listas-de-lista
```

Sale `(concatena-listas-de-lista '((1 2) (3 4) (4 6)))`

```
'(1 2 3 4 4 6)
```

c) Optimiza la función usando la técnica de recursión de cola.
SOLUCIÓN:

```
(define (concatena-listas-de-lista l)
  (concatena-listas-de-lista-tail l '()))

(define (concatena-listas-de-lista-tail l acc)
  (if (empty? l)
      acc
      (concatena-listas-de-lista-tail (cdr l) (append acc (car l)))))
```

d) Muestra los registros generados por la función del inciso anterior con el argumento `'((1 2) (3 4) (4 6))`. ¿Cuántos registros son generados? ¿Cuántos son ocupados a la vez?
SOLUCIÓN:

3. Dada la siguiente función

```
(define (foo n)
    (if (< n 10)
        n
        (+ (modulo n 10) (foo (quotient n 10)))))
```

a) Explica qué hace y dale un nombre mnemotécnico.

b) Muestra los registros generados cuando es llamada con el argumento 1729. ¿Cuántos registros son generados?

c) Optimiza la función usando la técnica de recursión de cola.

d) Muestra los registros generados por la función del inciso anterior con el argumento 1729. ¿Cuántos registros son generados? ¿Cuántos son ocupados a la vez?

4. Dada la siguiente función

```
(define (hoo n l)
    (if (zero? n)
        l
        (hoo (sub1 n) (cdr l))))
```

a) Explica qué hace y dale un nombre mnemotécnico.
SOLUCIÓN:

b) Muestra los registros generados cuando es llamada con los argumentos 3 y `'(1 2 3 4)`. ¿Cuántos registros son generados? ¿Cuántos son ocupados a la vez?

c) Optimiza la función usando la técnica de recursión de cola.
SOLUCIÓN:

```
(define (drop n l)
  (drop-tail n l l))

(define (drop-tail n l acc)
  (if (zero? n)
      l
      (drop-tail (sub1 n) (cdr l) acc)))
```

d) Muestra los registros generados por la función del inciso anterior con los argumentos 3 y `'(1 2 3 4)`. ¿Cuántos registros son generados? ¿Cuántos son ocupados a la vez?

5. Evalúa la siguiente expresión usando el intérprete para cajas visto en clase. Debes usar alcance estático y evaluación glotona. Mostrar el ambiente (stack) y memoria (heap) finales.

```
{with {a newbox 17}
   {with {b {newbox 29}}
      {with {foo {fun {} {setbox a {openbox a}}}}
         {with foo {fun {} {setbox b {openbox a}}}}
            {seqn {foo}
               {+ {openbox a} {openbox b}}}}}}
```

   Solución: La expresión que debemos evaluar es `{seqn {foo} {+ {openbox a} {openbox b}}}`, por lo que

6. Dada la definición de la función `next-location` que genera nuevas direcciones de memoria, vista en clase, modifícala para que no tenga efectos secundarios, es decir, que no dependa de ninguna variable externa.

   Hint: Modifícala usando la técnica Store Passing Style

```
(define last-location -1) ;; al inicio del intérprete

;; next-location: number
(define (next-location)
   (begin
      (set! last-location (add1 last-location))
      last-location))
```