Primer examen parcial - Fundamentos de lenguajes de programación - Duración: 2.5 horas

Carlos Andres Delgado S, Msc [*]

28 de Enero 2020

1. (40 puntos) Se tiene un tipo de dato que consiste en una lista de parejas cuyo primer elemento es par y el segundo es múltiplo de 3. Este tipo de de dato tiene la siguiente gramática:

```
<lista-par> ::= (lista-vacia) '()
            ::= (lista-no-vacia) <pareja> <lista-par>

<pareja>  ::= (pareja) <multiplo-2> <multiplo-3>
```

   Una pareja es una lista que tiene dos elementos únicamente.

   De acuerdo a esto indique:

   a) (15 puntos) Especificación inductiva (5 pts) y escriba la función in-S? (10 pts).

   b) (25 puntos) Deseamos implementar la función suma (10 pts), la cual recibe una lista de parejas, suma los elementos de cada pareja primero con primero y segundo con segundo, retornando una pareja como resultado.

   Diseñe las funciones constructoras y observadores usando los nombres especificados en la gramática. Con estos implemente la función suma, usted es libre de usar representación basada en listas o procedimientos.

   - Constructores: lista-vacia, lista-no-vacia, pareja (5 pts)
   - Predicados: lista-vacia?, lista-no-vacia?, pareja? (5 pts)
   - Extractores lista-par->primero, lista-par->resto, pareja->primero, pareja->segundo (5pts)

   Ejemplo:

```
(suma (lista-no-vacia (pareja 2 9) (lista-no-vacia (pareja 4 3) (
   lista-no-vacia (pareja 8 21) (lista-vacia)))))
;;Retorna
(pareja 14 33)
```

   Para este caso no se preocupe por verificar que el primer elemento de la pareja es par y el segundo múltiplo de 3.

[*] carlos.andres.delgado@correounivalle.edu.co

2. (30 puntos) Dada la siguiente expresión en lenguaje visto en el curso, suponiendo como ambiente inicial el vacío

```
let
    f = proc(a b) *(2,+(b,a))
    g = proc(c d) *(c,+(d,2))
    x = 5
    in
       let
         h = proc(m n) +(m,(f m n))
         i = proc(o p) +(o,(g (g o p) p))
         y = (f x (g x x))
         in
           letrec
              j(q,r) = if q then +(r,(j -(q,1) +(r,2))) else (h (i x
                 y) y)
           in
              (j x y)
```

El resultado de la expresión es 101455, dibuje los ambientes con sus respectivos valores denotados.

3. (30 puntos) Se desea agregar al interpretador la sentencia switch que tiene la siguiente gramática

```
<expression> ::= "switch" <identifier>
                 "(" ( "case" <expression> ":" <expression> )*  ")"
```

   a) (5 puntos) Agregue el caso a la expresión gramatical

```
(define grammar-simple-interpreter
    '(
    ...
    (expression (....) switch-exp)
    ...
    )
)
```

   b) (25 puntos) Agregue el caso a la función eval-expression y genere el código necesario para que funcione la estructura

```
(define eval-expression
  (lambda (exp env)
    (cases expression exp
    ...
    (switch-exp (...) ...)
    ...
    )
  )
)
```

Importante: Explique que hizo en cada uno de los casos.