Lenguajes de Programación
Auxiliar N°3

Auxiliares: Elizabeth L. Deniz

29/03/2019

1. Para cada uno de los siguientes programas en Racket, indique si es un programa válido. Si lo es, indique el valor que debería retornar al ejecutarse.

   (a)
   ```
   (let
     ([x 3])
    (let
        ([y x])
      y))
   ```

   (b)
   ```
   (let
       ([x 3])
     (let ([x x])
       x))
   ```

   (c)
   ```
   (let
       ([x (let
               ([y 3])
             y)])
     x)
   ```

   (d)
   ```
   (let
       ([x 3])
     (+ x (let
              ([x 5])
            x)))
   ```

2. Defina la función `(free-vars expr)` que retorna la lista de variables libres en la expresión `expr`.

3. Conside el lenguaje WAE(`with` with arithmetic expressions) definido en clases y la siguiente implementación de la función subst:

```
(define (subst expr sub-id val)
  (match expr
    [(num n) expr]
    [(add l r) (add (subst l sub-id val)
                    (subst r sub-id val))]
    [(with bound-id named-expr body)
     (if (symbol=? bound-id sub-id)
         expr
         (with bound-id named-expr
               (subst body sub-id val)))]
    [(id x) (if (symbol=? x sub-id) val expr)]))
```

   Explique porque se cae en los siguientes casos y arregle la función:

   - `{with {x 5} {with {y x} y}}`
   - `{with {x 5} {with {x x} x}}`

4. Urilizando la función `subst` arreglada en la pregunta anterior, describa cada paso de substitución y evaluación para el siguiente programa:

```
{with {x 5}
  {with {y {+ 1 x}}
    {with {z {+ y x}}
      {with {w z}
        {+ {+ 5 y} {+ z w}}}}}}
```

5. Considere la siguiente función `interp`:

```
(define (interp expr)
  (match expr
    [(num n) n]
    [(add l r) (+ (interp l) (interp r))]
    [(sub l r) (- (interp l) (interp r))]
    [(id x) (error "free identifier" x)]
    [(with bid ne body)
     (interp (subst bid (interp ne) body))]))
```

   ¿Qué régimen de evaluación implementa este intérprete? ¿Qué habría que cambiar para pasar de un régimen al otro?

6. Considere el lenguaje WAE-L y WAE-E, donde la única diferencia entre ellos es que en el primero la substitución es *lazy* y en el segundo es *eager*. La gramática de ambos lenguajes es:

```
(deftype WAE-*
  (num n)
  (add l r)
  (with id named-expr body)
  (id s))
```

(a) Para cada lenguaje, muestre paso a paso la evaluación del siguiente programa.

```
{with {x {+ 5 2}}
  {with {y {+ 3 x}}
    {+ y y}}}
```

(b) En el ejemplo anterior, la evaluación *eager* llega en menos pasos a la respuesta que la evaluación *lazy*. Demuestre que en todo caso esto es cierto o muestre un contraejemplo.

(c) Genere un test que indique si su función de substitución es *eager* o *lazy*.

7. Implemente la función `obfuscate` que cambia los identificadores de un programa WAE a nombres aleatorios (use `gensym` para generar los nombres). La transformación debe respetar el alcance léxico. Ejemplo:

```
> (obfuscate (parse '{with {x 1} x}))
(with 'g108 (num 1) (id 'g108))

> (obfuscate (parse '{with {x 1} {with {y x} {+ x y}}}))
(with 'g109 (num 1) (with 'g110 (id 'g109) (add (id 'g109) (id 'g110))))
```