Lenguajes de Programación

Tarea 2

José Ethan Ortega González

11 de octubre de 2021

1. Se quieren representar árboles binarios cuyos únicos nodos etiquetados (con elementos de cualquier conjunto) son las hojas. Para ello se utiliza la siguiente definición recursiva de árboles:

```
(define (any? a) #t)

(define-type Arbol
  [hoja (a any?)]
  [mkt (t1 Arbol?) (t2 Arbol?)])
```

   (a) Definir las funciones recursivas nh, nni que calculan el número de hojas y el número de nodos internos (los que no son hojas) en un árbol respectivamente.

   (b) Demostrar que `(nh t) = (+ (nni t) 1)`

Solución:

(a) Las funciones son las siguientes:

```
(define (nh arbol)
  (match arbol
    [(hoja _) 1]
    [(mkt i d) (+ (nh i) (nh d))]))

(define (nni arbol)
  (match arbol
    [(hoja _) 0]
    [(mkt i d) (add1 (+ (nni i) (nni d)))]))
```

(b) *Demostración.* Lo demostraremos por inducción sobre `t`

- **Base de inducción:** `t` = `(hoja a)`. Tenemos lo siguiente:

  `(nh (hoja a)) = 1 = (+ 0 1) = (+ (nni t) 1)`

  Por lo que queda demostrada la base de inducción.
- **Hipótesis de inducción:** Supongamos que se cumple para t_1 y t_2.
- **Paso inductivo:** Tenemos que

  `(nh t) = (+ (nh i) (nh d))`

  Pero por hipótesis de inducción tenemos lo que sigue:

  `(+ (nh i) (nh d)) = (+ (+ (nni i) 1) (+ (nni d) 1))`

Y esto, por propiedades de la suma, es igual a:

```
(+ (+ (nni i) 1) (+ (nni d) 1)) = (+ (nni i) (nni d) 2)
```

Por lo tanto,

```
(nh t) = (+ (nni i) (nni d) 2)
```

Ahora, tenemos que:

```
(+ (nni t) 1) = (+ (add1 (+ (nni i) (nni d))) 1)
```

Y esto, es igual a:

```
(+ (add1 (+ (nni i) (nni d))) 1) = (+ (nni i) (nni d) 2)
```

Por lo tanto,

```
(+ (nni t) 1) = (+ (nni i) (nni d) 2)
```

Por ultimo, concluimos que:

```
(nh t) = (+ (nni t) 1)
```

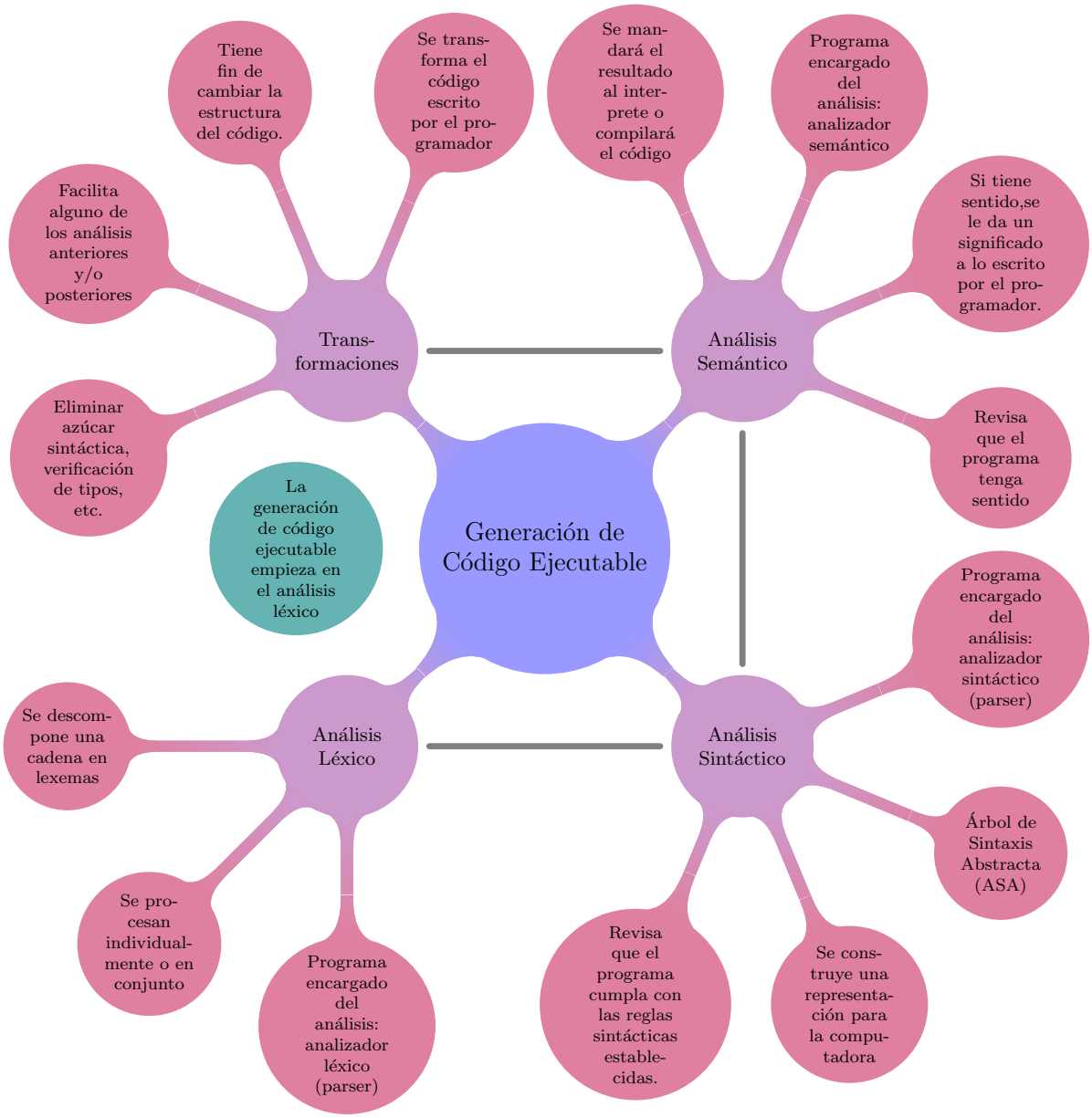
Por lo que queda demostrado.

□

2. Dibuja un mapa mental que muestre las fases de generación código ejecutable, sus principales características y elementos involucrados.

Solución:

El mapa mental es el siguiente:



3. Dadas las siguientes expresiones de WAE en sintaxis concreta, da su respectiva representación en sintaxis abstracta por medio de los Árboles de Sintaxis Abstracta correspondientes. En caso de no poder generar el árbol, justificar.
   (a) `{+ 18 { - 15 {+ 40 5}}}`
   (b) `{+ { - 15 {+ 40}}}`
   (c)
```
{with {a 2}
      {with {b {+ a a}}
         {+ a {- b 5}}}}
```

Solución:

(a)
```
(add (num 18) (sub (num 15) (add (num 40) (num 5))))
```

(b) No se puede general el árbol, porque falta un argumento en la suma donde solo está el 40 y en la suma principal.

(c)
```
(with (id 'a) (num 2)
      (with (id 'b) (add (id 'a) (id 'a))
         (add (id 'a) (sub (id 'b) (num 5)))))
```

4. Currifica cada uno de los siguientes términos:
   (a) $\lambda abc.abc$
   (b) $\lambda abc.\lambda cde.acbdce$
   (c) $(\lambda x.(\lambda xy.y)(\lambda zw.w))(\lambda uv.v)$

Solución:

(a) $\lambda a.\lambda b.\lambda c.abc$

(b) $\lambda a.\lambda b.\lambda c.(\lambda c.\lambda d.\lambda e.acbdce)$

(c) $(\lambda x.(\lambda x.\lambda y.y)(\lambda z.\lambda w.w))(\lambda u.\lambda v.v)$

5. Para cada uno de los siguientes términos, aplica α-conversiones para obtener términos donde todas las variables de ligado sean distintas.
   (a) $\lambda x.\lambda y.(\lambda x.y\lambda y.x)$
   (b) $\lambda x.(x(\lambda y.(\lambda x.xy)x))$
   (c) $\lambda a.(\lambda b.a\lambda b(\lambda a.ab))$

Solución:

(a) Tenemos lo siguiente:

$$\begin{aligned}\lambda x.\lambda y.(\lambda x.y\lambda y.x) &\equiv_\alpha \lambda x.\lambda y.(\lambda x.y\lambda y.x)[y := w]\\ &\equiv_\alpha \lambda x.\lambda w.(\lambda x.w.\lambda y.x)[x := z]\\ &\equiv_\alpha \lambda x.\lambda w.(\lambda z.w.\lambda y.z)\end{aligned}$$

(b) Tenemos lo siguiente:

$$\begin{aligned}\lambda x.(x(\lambda y.(\lambda x.xy)x)) &\equiv_\alpha \lambda x.(x(\lambda y.(\lambda x.xy)x))[x := w]\\ &\equiv_\alpha \lambda x.(x(\lambda y.(\lambda w.wy)x))\end{aligned}$$

(c) Tenemos lo siguiente:

$$\begin{aligned}\lambda a.(\lambda b.a\lambda b(\lambda a.ab)) &\equiv_\alpha \lambda a.(\lambda b.a\lambda b(\lambda a.ab))[b := c]\\ &\equiv_\alpha \lambda a.(\lambda b.a\lambda c(\lambda a.ac))[a := d]\\ &\equiv_\alpha \lambda a.(\lambda b.a\lambda c(\lambda d.dc))\end{aligned}$$