Compiladores 2020-1
Facultad de Ciencias UNAM
Práctica 4: Análisis Sintáctico

Lourdes del Carmen González Huesca Alejandra Krystel Coloapa Díaz
Javier Enríquez Mendoza

Integrantes del equipo

Ángeles Martínez Ángela Janín	314201009
García Landa Valeria	314033008
Martínez Monroy Emily	314212391
Rebollar Pérez Ailyn	314322164

Desarrollo de los Ejercicios

Ejercicio 1
Para este ejercicio primero se creó un lenguaje intermedio L7 para poder eliminar los let y letrec con muchos parámetros y se volvieron a agregar pero con un sólo parámetro para tipo, expresión y variable.

```
(define-language L7
  (extends L6)
  (Expr (e body)
        (- (let ([x* t* e*] ...) body)
           (letrec ([x* t* e*] ...) body))
        (+ (let ([x t e] body)
           (let () body)
           (letrec ([x t e]) body)
           (letrec () body)))))
```

Ejercicio 2
Para este ejercicio no se creó un lenguaje nuevo. Lo que se hizo fue cazar las expresiones let. Hacemos la comparación de t que es el tipo de la variabl y si es de tipo Lambda se cambia por un letrec y el cuerpo se queda tal cual. Pero en caso de que sea del tipo Boolen, Int, List o Char se deja la expresión tal cual del let. Y al final se hace un catamorfismo en el cuerpo del let puesto que existe la posibilidad de tener let's anidados.

Ejercicio 3
Para éste ejercicio lo que hicimos fue hacer una función auxiliar que se llama *(new name)* la cual hace los nombres únicos para cada una de las variables para declarar lambdas y para ésta hicimos un contador global que se llama *count*, así podemos asignar "fooz un número con el contador.
Ahora para el proceso lo que hacemos es cazar las expresiones del tipo lambda, entonces construimos una nueva expresión **letfun** con el nuevo nombre de la lambda que da la función auxiliar, la lambda que cazamos y mandamos llamar la lambda para que no se pierda el sentido, entonces así letfun funciona como una aplicación de función.
Y como agregamos la expresión letfun, el lenguaje se extiende a L8 y queda de la siguiente manera:

```
(define-language L8
  (extends L7)
  (Expr (e body)
        (+ (letfun ([x t e]) body))))
```

Ejercicio 4
Para este ejercicio no se hizo un enguaje nuevo ya que sólo se verifica la aridad de las primitivas.
Para esto se definieron dos funciones auxiliares en las cuales se verifica en qué caso de primitivas cae. Para las aritméticas tenemos $+ \ - \ * \ /$ y en el caso de la lista tenemos *car cdr length*. Después se hizo el proceso el cuál matchea la expresión que corresponde a las primitivas y se hacen dos comparaciones.
Para la primera se pregunta que si pr tiene longitud 2 puesto que las operaciones aritméticas tienen dicha aridad, y si es correcto se regresa la expresión, en otro caso, se manda un error. Lo mismo pasa con la lista sólo que en este caso cambia la longitud ya que aquí es longitud 1.

Ejercicio 5
Para éste ejercicio no se definió un nuevo lenguaje pero lo que hicimos fue hacer una función auxiliar llamada FV donde nos encargamos de encontrar las variables libres en cada expresión del lenguaje cuidando no hacer variables libres a las ligadas en let, letrec, lambdas, etc. y devolver una lista con ellas para que en el proceso la mandemos llamar y verificar si ésta lista es vacía, si lo es, es porque no hubo variables libres y regresamos la misma expresión pero si no es vacía, entonces mandamos un error indicando a la primer variable libre que encontramos.