Curso: Lenguajes de programación
Practica 3: Generación de código ejecutable - Análisis sintáctico.

Karla Ramírez Pulido Alan Alexis Martínez Lopez José Ricardo Desales Santos

José Eliseo Ortíz Montaño

Fecha de inicio: Miercoles 22 de Marzo del 2023.
Fecha de entrega: Miercoles 29 de Marzo del 2023.

Objetivo

Reforzar los conceptos relacionados con los tipos de análisis por los que pasa el código fuente para generar código ejecutable, así como implementar un analizador sintáctico para el lenguaje `WBAE`.

Estructura

La práctica está conformada por los siguientes archivos:

- `grammars.rkt`; en este archvo se encuentra la definición de la gramática del lenguaje `WAE`[1] que se utilizará para el desarrollo de esta práctica, la cual, en EBNF[2] es la siguiente:

```
<expr> ::= <id>
         | <num>
         | {<op> <expr>+}
         | {with {{<id> <expr>}+} <expr>}
         | {with* {{<id> <expr>}+} <expr>}

   <id> ::= a | ... | z | A | Z | aa | ab | ab | ... | aaa | ...
          (Cualquier combinación de caracteres alfanuméricos
           con al menos uno alfabético)

<num>   ::= ... | -1 | 0 | 1 | 2 | ...
<op>    ::= + | - | * | / | modulo | min | max | expt | sqrt | sub1 | add1
            | < | <= | = | > | >= | zero? | num?
```

y para esto, se definen el siguiente par de ADT [3]:

```
;; Definición del tipo Binding
(define-type Binding
    [binding (id symbol?) (value WAE?)])
```

[1] *With and Arithmetic Expressions*
[2] Extended Backus-Naur Form, o en español *Forma Backus-Naur Extendida.*
[3] Trabajo colaborativo realizado en la ayudantía.

```
;; Definición del tipo WAE
(define-type WAE
    [id    (i symbol?)]
    [num   (n number?)]
    [op    (f procedure?) (args (listof WAE?))]
    [with  (bindings (listof binding?)) (body WAE?)]
    [with* (bindings (listof binding?)) (body WAE?)])
```

- `parser.rkt`; en este archivo se encuentra la definición de la función `(parse sexp)`. Dicha función realiza el análisis sintáctico correspondiente a la `sexp` pasada como parámetro, contruyendo expresiones del TDA `WBAE`[4] que se encuentra definido en el archivo `grammmars.rkt`.
- `test.rkt`; aquí se encuentran definidas un conjunto de pruebas unitarias para probar el trabajo realizando en la práctica. En el intérprete, se deberá dar click en el botón de *Run* para ejecutar dichas pruebas. El pasar todas las pruebas satisfactoriamente no garantiza obtener la máxima calificación en la práctica, ya que esto depende también de la manera en cómo se desarrolle esta (trabajo colaborativo, buenas prácticas de programación, etc...).

Nota: es importante guardar todos los archivos dentro de un mismo directorio.

Ejercicios

1. **(1 pt)** Extender la definición de la gramática del lenguaje `WAE`, para así convertirlo en el lenguaje `WBAE`[5] especificado por la siguiente gramática:

```
<expr> ::= <id>
         | <num>
         | <bool>
         | <char>
         | <string>
         | <list>
         | {<op> <expr>+}
         | {with {{<id> <expr>}+} <expr>}
         | {with* {{<id> <expr>}+} <expr>}

   <id> ::= a | ... | z | A | Z | aa | ab | ab | ... | aaa | ...
          (Cualquier combinación de caracteres alfanuméricos
           con al menos uno alfabético)

<bool>  ::= true | false
<char>  ::= 'a' | 'b' | 'c' | ...
<string>::= "a" | "aa" | "ab" | ...
<list>  ::= empty | {lst <expr>*}
<num>   ::= ... | -1 | 0 | 1 | 2 | ...
<op>    ::= + | - | * | / | modulo | min | max | expt | sqrt | sub1 | add1
              | < | <= | = | > | >= | not | and | or | zero?
              | num? | bool? | char? | string? | list?
              | car | cdr   | length | empty? | string-length
```

[4] Definición que tendrás que completar como parte del ejercicio 1 de esta prática.
[5] *With Boolean and Arithmetic Expressions.*

Se deberán desarrollar los siguientes puntos:

- Definir el tipo correspondiente a la expresión de la gramática `<bool>` dentro de la definción del ADT `WBAE`[6], el cual debe tener como identificador `bool` y un único parámetro `b` de tipo booleano[7].
- De manera similar, definir el tipo correspondiente a la expresión de la gramática `<char>`, la cual debe tener como identificador `chaR` y un único parámetro `c` de tipo `char`[8].
- Análogamente, definir el tipo correspondiente a la expresión de la gramática `<string>`, la cual debe tener como identificador `strinG` y un único parámetro `s` de tipo `string`[9]
- Análogamente, definir el tipo correspondiente a la expresión de la gramática `<list>`, de manera que su identificador sea `lst`, y como único parámetro tenga una lista, llamada `l`, de elementos de tipo `WBAE`.

de tal manera que la definición del lenguaje `WBAE` sea similar a lo mostrado a continuación:

```
;; Definición del tipo WBAE
(define-type WBAE
    [id    (i symbol?)]
    [num   (n number?)]
    [op    (f procedure?) (args (listof WBAE?))]
    ...
    ...
    ;; Aquí se encuentran las definiciones anteriormente especificadas
    ;; que deberán ser agregadas.
    ...
    ...
    [with  (bindings (listof binding?)) (body WBAE?)]
    [with* (bindings (listof binding?)) (body WBAE?)])
```

Para este ejercicio, es posible eliminar la definición del ADT `WAE`, comentar dicha definición o modificarla acorde a la nueva especificación. Es importante observar que el ADT `Binding` necesita de un ligero ajuste.

2. **(0.25 pts)** Definir las funciones `anD` y `oR`. Por ahora, no recibirán parámetros y el cuerpo de las funciones debe ser `void`[10]. Estas definiciones son importantes, ya que permiten definir a las funciones `and` y `or` como procedimientos, ya que no se encuentran definidas como tales en Racket.

   Si en el interprete se intenta hacer algo como lo mostrado a continuación, se obtiene el siguiente error:

```
    >and
    and: bad syntax in: and
```

   por lo que no es posible realizar el parse de `and` y `or` como procedimientos. Por ello, en el diseño del lenguaje `WBAE` se opta por definir las funciones `anD` y `oR`[11] de manera que podamos tener dichos operadores booleanos como procedimientos dentro de WBAE.

```
> oR
#<procedure:oR>

> anD
#<procedure:anD>
```

[6] Observa que se tiene que cambiar el nombre del ADT en su definición.
[7] Nativo de Racket.
[8] Nativo de Racket.
[9] Nativo de Racket.
[10] Literalmente, ese debe ser el cuerpo de las funciones.
[11] Son un *wrapper* para las primitivas del lenguaje.

Es importante aclarar que, la semántica de las funciones `anD` y `oR` es trabajo de prácticas posteriores.

3. **(8.75 pts)** Completar el cuerpo de la función `(parse sexp)` del archivo `parser.rkt`; la cual, como ya se mencionó anteriormente, realiza el análisis sintáctico correspondiente a la `sexp` pasada como parámetro.
   Toma en cuenta las siguientes observaciones:

   - Para los predicados, las funciones `sub1`, `add1`, `not`, `length`, `car`, `cdr` y `string-length` la aridad debe ser 1.
   - Para las funciones `modulo` y `expt`, la aridad debe ser 2.
   - Para el resto de las funciones, la aridad es arbitraria, pero mayor a 1.
   - En todos los casos anteriores, deberás arrojar un mensaje de error en caso de que la aridad sea incorrecta; especificando la cantidad de argumentos dados y la cantidad de argumentos esperados.
   - Se debe verificar la estructura de los `bindings` contruidos resultado de aplicar la función `parse` a las expresiones `with` y `with*`, esto es, un binding similar a `[x y 3]` debe dar como error el mensaje de error

     `"Binding «binding en cuestión» malformado."`

```
;; parse : s-expression -> WBAE
(define (parse sexp)
    ;; Aquí va su código
    ...
)

>(parse 12345)
(num 12345)

>(parse 'apple)
(id 'apple)

>(parse true)
(bool true)

>(parse 'A')
(chaR 'A')

>(parse "Hoy comeré rico")
(stringG "Hoy comeré rico")

;; Observa que en el proceso de parsing no estamos verificando la semantica, en este caso,
;; los argumentos de las funciones pueden causar problemas al momento de su evaluación,
;; pero sintácticamente son correctos.

>(parse '{+ 1 2})
(op + (list (num 1) (num 2))

>(parse '{+ 1})
error "El procedimiento espera al menos 2 argumentos.
       Número de argumentos dados: 1."

>(parse '(or true false true))
(op oR (list (bool true) (bool false) (bool true)))

>(parse '(and true '{+ 1 2}))
(op anD (list (bool true) (op + (list (num 1) (num 2)))))
```

```
>(parse '(false 1 'hello 'B' "Tengo hambre"))
(lst (list (bool false) (num 1) (id 'hello) (chaR 'B') (stringG "Tengo hambre")))

;; El parsing en las expresiones with y with* es exactamente igual (salvo por la '*');
;; la diferencia entre ambas funciones es a nivel semántico.

>(parse '{with ([x 2] [y 3]) (+ x y)})
(with (list (binding 'x (num 2)) (binding 'y (num 3))) (op + (list (id 'x) (id 'y))))

>(parse '{with ([x 2 2] [y 3]) (+ x y)})
error "Binding "[x 2 2]" mal formado."

>(parse '{with ([x 2] [3 y]) (+ x y)})
error "Binding "[3 y]" mal formado."

>(parse '{with* ([x 2] [y 3]) ( + x y)})
(with* (list (binding 'x (num 2)) (binding 'y (num 3))) (op + (list (id 'x) (id 'y))))
```

Entrega

Acerca de la entrega de la práctica:

- La realización y entrega de la práctica deberá realizarse en equipos de a lo más 4 personas.[12]
- Su entrega debe consistir en los archivos `grammars.rkt` y `parser.rkt`, los cuales se proporcionan junto con este archivo. Recuerden agregar, en cada uno de estos, los datos de los integrantes del equipo.
- El uso de funciones auxiliares está totalmente permitido, sin embargo, deben ser declaradas justo después de la función principal que hace uno de ellas.
- Las funciones deben incluir comentarios, tanto de documentación como del proceso de desarrollo. Estos deben ser claros, concisos y descriptivos.
- Queda estrictamente prohibido utilizar funciones primitivas del lenguaje que resuelvan directamente los ejercicios.[13]
- Se deberá subir la versión final de su práctica al apartado del classroom correspondiente antes de la fecha límite. Esto sólo debe realizarlo un integrante del equipo; el resto deberá marcar como entregada la actividad y en un comentario privado especificar quienes son los miembros del equipo.
- Para la administración de la práctica en GitHub, selecciona el siguiente enlace.

[12] Cualquier situación con respecto a este punto será tratada de acuerdo a las particularidades del caso. Para esto, acercarse al ayudante del rubro del laboratorio a la brevedad.

[13] Cualquier duda con respecto a este punto, por favor manda un correo para atenderla.