Lenguajes de Programación

Práctica 3

Semestre 2022-1
Facultad de Ciencias, UNAM

Profesora	Karla Ramírez Pulido	**Fecha de inicio:** 03 de Noviembre de 2021
Ayud. Lab	Silvia Díaz Gómez	**Fecha de entrega:** 10 de Noviembre de 2021
Ayud. Lab	Fhernanda Montserrat Romo Olea	

Descripción

La práctica consiste en implementar un intérprete sencillo para el lenguaje **WAE**. Para esto, se debe completar el cuerpo de las funciones faltantes dentro de los archivos **grammars.rkt**, **parser.rkt** e **interp.rkt** hasta que pasen las pruebas unitarias incluidas en el archivo **test-practica3.rkt** y se ejecute correctamente el archivo **practica3.rkt**.

La gramática del lenguaje **WAE** se presenta a continuación:

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

<id> ::= a | b | c | ...

<num> ::= 1 | 2 | 3 | ...

<op> ::= + | - | * | / | modulo | expt | add1 | sub1
```

Ejercicios

1. (3.5 pts.) Completar el cuerpo de la función **(parse sexp)** dentro del archivo **parser.rkt** el cual recibe una expresión simbólica[1], realiza el análisis sintáctico correspondiente, esto es, construye un Árbol de Sintaxis Abstracta[2] (ASA). Para el análisis sintáctico de las operaciones aritméticas se debe hacer un mapeo entre los símbolos de función en sintaxis concreta y las funciones de Racket, utilizando la función **(elige s)** previamente definida.

[1] Del inglés, *s-expression*. Puede ser un número, un símbolo o una lista de expresiones simbólicas.
[2] Del inglés, *Abstract Syntax Tree (AST)*.

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

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

;; parse: s-expression → AST
(define (parse sexp) ...)
```

2. (3 pts.) Completar el cuerpo de la función **`(subst expr sub-id value)`** dentro del archivo **`interp.rkt`** la cual realiza la sustitución **`expr[sub-id := value]`** correspondiente, esto es, reemplaza cada presencia de la variable **`sub-id`** en la expresión **`expr`** por otra expresión **`value`**. Recuerda cuidar el alcance de las variables en las expresiones **`with`** y **`with*`**.

```
;; subst: AST symbol AST → AST
(define (subst expr sub-id value) ...)
```

3. (3.5 pts.) Completar el cuerpo de la función **`(interp expr)`** dentro del archivo **`interp.rkt`** que dada una expresión, regresa la evaluación correspondiente. Para la evaluación de expresiones **`with`** y **`with*`** es necesario usar la función **`subst`** definida en el inciso 2. Tomar en consideración:

   - **Identificadores**

     Se debe lanzar un error indicando que se trata de una variable libre.

     `(interp (parse 'foo)) => error: Variable libre`

   - **Números**

     Al ser un valor atómico, los números se evalúan así mismos.

     `(interp (parse 1729)) => 1729`

   - **Operaciones aritméticas**

     Se debe aplicar el operador correspondiente a la lista de operandos indicada.

```
;; Operaciones unarias
(interp (parse '{add1 18}))      => 19
(interp (parse '{sub1 35}))      => 34

;; Operaciones binarias
(interp (parse '{modulo 10 2})) => 0
(interp (parse '{expt 2 3}))     => 8

;; Operaciones n-arias
(interp (parse '{+ 1 2 3}))      => 6
(interp (parse '{- 3 2 1}))      => 0
(interp (parse '{* 1 2 3}))      => 6
(interp (parse '{/ 8 2 2}))      => 2
```

- **Asignaciones locales simples (`with`)**

  Dada la lista de parejas de identificadores con valores de la forma **`(binding id value)`**, se deben de sustituir en el cuerpo (**`body`**) correspondiente cada uno de los identificadores (**`id`**) por su valor (***`value`***).

```
(interp (parse '{with {{a 2} {b 3}} {+ a b}})) => 5
(interp (parse '{with {{a 2} {b {+ a a}}} b})) =>
   error: Variable libre
```

- **Asignaciones locales anidadas (`with*`)**

  Dada la lista de parejas de identificadores con valores de la forma **`(binding id value)`**, se deben de sustituir en el cuerpo (**`body`**) correspondiente cada uno de los identificadores (**`id`**) por su valor (***`value`***). En esta versión de **`with`**, se permite hacer referencia a identificadores definidos previamente (anidados).

```
(interp (parse '{with {{a 2} {b {+ a a}}} b})) => 4
```

```
;; interp: AST → number
(define (interp expr) ...)
```

Referencias

[1] Shriram Krishamurthi, *Programming Languages: Application and Interpretation,* First Edition, Brown University, 2007.