Universidad Nacional Autónoma de México
Facultad de Ciencias
Lenguajes de Programación

Práctica 5

Karla Ramírez Pulido
karla@ciencias.unam.mx

J. Ricardo Rodríguez Abreu
ricardo_rodab@ciencias.unam.mx

Manuel Soto Romero
manu@ciencias.unam.mx

Fecha de inicio: 6 de abril de 2018
Fecha de término: 13 de abril de 2018
Semestre 2018-2

Objetivo

Agregar condicionales al lenguaje implementado en la Práctica 4 y repasar el concepto de evaluación perezosa y su implementación mediante cerraduras de expresión[1] y la identificación de puntos estrictos.

Antecedentes

En las sesiones previas de laboratorio se revisó el concepto de evaluación perezosa y se realizó una actividad para implementar un pequeño intérprete para una versión simplificada del lenguaje de programación `CFWAE` que hace uso de cerraduras de expresión. Se recomienda revisar dicha actividad junto con las Notas 4 del curso.

Repositorio

El material necesario para completar esta práctica se encuentra en el repositorio de *GitHub Classroom* del curso: `https://classroom.github.com/g/MJn_Vq0V`.

Desarrollo de la práctica

La gramática en EBNF para las expresiones del lenguaje `CFWBAE` (*Conditionals, Functions, With, Booleans and Arithmetic Expressions)* que se implementa en esta práctica es la siguiente:

[1] *Expression closure.*

```
<expr> ::= <id>
         | <num>
         | <bool>
         | {<op> <expr>+}
         | {if <expr> <expr> <expr>}
         | {cond {<expr> <expr>}+ {else <expr>}}
         | {with {{<id> <expr>+}
         | {with* {{<id> <expr>}+} <expr>}
         | {fun {<id>*} <expr>}
         | {<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 | ...

<bool> ::= true | false

 <op> ::= + | - | * | / | % | min | max | pow | sqrt
        | < | <= | = | /= | > | >=
        | not | and | or
```

En equipos de **tres integrantes** se deben completar las funciones faltantes de los archivos `parser.rkt`, `desugar.rkt` e `interp.rkt` hasta que logren pasar todas las pruebas unitarias que se incluyen en el archivo `pruebas_practica5.rkt` y se ejecute correctamente el archivo `pratica5.rkt`[2].

Ejercicio 5.1 (1 pt.) Completar el cuerpo de la función `(parse sexp)` del archivo `parser.rkt` que realiza el análisis sintáctico correspondiente, es decir, construye expresiones del TDA `CFWBAE` incluido en el archivo `grammars.rkt`.

```
;; parse: s-expression -> CFWBAE
(define (parse sexp) ...)
```

```
> (parse '{+ 1 2})
(opS + (list (numS 1) (numS 2))
```

Ejercicio 5.2 (2 pts.) Completar el cuerpo de la función `(desugar expr)` del archivo `desugar.rkt` que elimina azúcar sintáctica[3] de las expresiones de `CFWBAE`, es decir, las convierte en expresiones del TDA `CFBAE` incluido en el archivo `grammars.rkt`.

Además de las expresiones endulzaddas de la Práctica 4, en esta práctica, se agregan las siguientes expresiones con azúcar sintáctica:

[2]Para tener derecho a calificación, los archivos deben ejecutarse sin errores.

[3]Tipo de sintaxis que hace que un programa sea más "dulce" o fácil de escribir.

- `cond` este tipo de expresiones son una versión endulzada de expresiones `if` anidadas. Por ejemplo:

```
{cond
     {{< 2 3} 1}
     {{> 10 2} 2}
     {else 3}}
```

  se transforma en

```
{if {< 2 3}
     1
     {if {> 10 2}
         2
         3}}
```

```
;; desugar: CFWBAE -> CFBAE
(define (desugar expr) ...)
```

```
> (desugar (parse '{with {{a 3}} {+ a 4}}))
(app (fun '(a) (op + (id 'a) (num 4))) (list (num 3)))
```

Ejercicio 5.3 (7 pts.) Completar el cuerpo de la función `(interp expr env)` del archivo `interp.rkt` que realiza el análisis semántico correspondiente, es decir, evalúa expresiones de `CFBAE`. Para evaluar los puntos estrictos del lenguaje se ebe hacer uso de la función `strict`.

Además de las especificaciones que se tomaron en cuenta para resolver la Práctica 4, se deben considerar ahora, los siguientes puntos para implementar la función `interp`:

- En las operaciones n-arias, se encuentra un punto estricto. Se debe evaluar cada uno de los operandos para poder realizar la operación correspondiente. Ejemplo:

```
> (interp (desugar (parse '{* {+ 1 2} {+ 3 4} 5})) (mtSub))
(numV 50)
```

- Las expresiones `if` deben evaluar su condición y regresar el valor correspondiente en caso de que la condición se evalúe a verdadero o falso, según sea el caso.

  En este tipo de expresión se encuentra un punto estricto, por lo que se debe evaluar la condición para regresar el valor correspondiente. Ejemplo:

```
> (interp (desugar (parse '{if true true false})) (mtSub))
(boolV #t)
```

- En las expresiones `cond` se debe evaluar cada condición y regresar el valor correspondiente en caso de que alguna sea verdadera. En caso contrario, se debe continuar con la evaluación del resto de condiciones o ejecutar el caso `else` si ninguna condición fue verdadera.

  Al ser azúcar sintáctica de expresiones `if`, cada condición de este tipo de expresión también es un punto estricto. Ejemplo:

```
> (interp (desugar (parse '{cond {true 1} {true 2} {else 3}})) (mtSub))
(numV 1)
```

- En la aplicación a función se encuentra un punto estricto, en el cual se debe evaluar a función para poder aplicarla con los argumentos correspondientes.

  Este tipo de expresión debe agregar los argumentos correspondiente al ambiente sin evaluar mediante el constructor `exprV`. Por ejemplo, en la siguiente llamada el ambiente de evaluación es: `(aSub 'a (exprV (op + (list (num 2) (num 3))) (mtSub)) (mtSub))`.

```
> (interp (desugar (parse '{{fun {x} {+ x 2}} {+ 2 3}})) (mtSub))
(numV 7)
```

```
;; interp: CFBAE -> CFBAE-Value
 (define (interp expr env) ...)
```

```
> (interp (op + (list (num 1) (num 2))))
3
```

Referencias

Algunas referencias de consulta:

[1] Karla Ramírez, Manuel Soto, *Notas de laboratorio del curso de Lenguajes de Programación*, Semestre 2018-2, Facultad de Ciencias, UNAM. Disponibles en: [http://lenguajesfc.com/notas.html].

[2] Rodrigo Ruiz Murgía, *Manual de prácticas para la asignatura de Lenguajes de Programación*, Reporte de actividad docente, Facultad de Ciencias, 2016.

[3] Shriram Krishnamurthi, *Programming Languages: Application and Interpretation*, Primera edición, Brown University, 2007.