Universidad Nacional Autónoma de México
Facultad de Ciencias
Lenguajes de Programación

Práctica 8

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: 4 de mayo de 2018
Fecha de término: 16 de mayo de 2018
Semestre 2018-2

Objetivo

Extender el lenguaje de programación implementado en la Práctica 7 para que manipule cajas e instrucciones imperativas mediante la técnica *Store Passing Style*.

Antecedentes

En clase y en las sesiones previas de laboratorio se revisó la técnica *Store Passing Style* y se implementó un pequeño intérprete para una versión simplificada del lenguaje de programación `BERCFWAE/L` que implementa agrega cajas e instrucciones imperativas. Se recomienda revisar dicha actividad junto con el libro de texto del curso [3].

Repositorio

El material necesario para completar esta práctica se encuentra en el repositorio de *GitHub Classroom* del curso: `https://classroom.github.com/g/Z0iATEYB`.

Desarrollo de la práctica

La gramática en EBNF para las expresiones del lenguaje `BERCFWBAEL/L` (*Boxes*, *Exceptions*, *Recursive*, *Conditionals*, *Functions*, *With*, *Booleans*, *Arithmetic Expressions Lists and Lazy)* que se implementa en esta práctica es la siguiente:

```
<expr> ::= <id>
         | <num>
         | <bool>
         | <list>
         | {<op> <expr>+}
         | {if <expr> <expr> <expr>}
         | {cond {<expr> <expr>}+ {else <expr>}}
         | {with {{<id> <expr>+}
         | {with* {{<id> <expr>}+} <expr>}
         | {rec {{<id> <expr>}+} <expr>}
         | {fun {<id>*} <expr>}
         | {<expr> <expr>*}
         | {thorws <id>}
         | {try/catch {{<id> <expr>}+} <expr>}
         | {newbox <expr>}
         | {openbox <expr>}
         | {setbox <expr> <expr>}
         | {seqn <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

<list> ::= empty
         | {list <expr>+}

<op> ::= + | - | * | / | % | min | max | pow | sqrt | inc | dec
       | < | <= | = | /= | > | >= | zero?
       | not | and | or
       | head | tail | append | empty?
```

En equipos de **tres integrantes** se deben de completar cada una de las funciones faltantes de los archivos correspondientes hasta que logren pasar todas las pruebas unitarias que se incluyen en el archivo `pruebas_practica8.rkt` y se ejecute correctamente el archivo `pratica8.rkt`[1].

Ejercicio 8.1 (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 `BERCFWBAEL/L` incluido en el archivo `grammars.rkt`.

```
;; parse: s-expression -> BERCFWBAEL/L
(define (parse sexp) ...)
```

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

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

Ejercicio 8.2 (1 pt.) Completar el cuerpo de la función `(desugar expr)` del archivo `desugar.rkt` el cual elimina el azúcar sintáctica[2] de las expresiones de BERCFWBAEL/L, es decir, las convierte en expresiones del TDA BERCFBAEL/L incluido en el archivo `grammars.rkt`[3].

```
;; desugar: BERCFWBAEL/L -> BERCFBAEL/L
(define (desugar expr) ...)
```

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

Ejercicio 8.3 (8 pts.) Completar el cuerpo de la función `(interp expr env store)` del archivo `interp.rkt` que realiza el análisis semántico correspondiente, es decir, evalúa expresiones de BERCFBAEL/L.

Además de las especificaciones que se tomaron en cuenta para resolver la Práctica 7, se deben considerar ahora, los siguientes puntos para implementar la función `interp`:

- Al crear una caja se debe crear un nuevo registro en la estructura `Store`, para generar los índices de dicha estructura, se provee de la función `next-location` que genera secuencias de números enteros. Se debe regresar una caja con la dirección en memoria del valor correspondiente.

  ```
  > (interp (desugar (parse '{newbox 1729})) (mtSub) (mtSto))
  (boxV 0)
  ```

- Al abrir una caja se debe buscar el índice correspondiente en la estructura `Store`, mediante la función `lookup-sto` que busca el último valor agregado en la estructura.

  ```
  > (interp (desugar (parse '{openbox {newbox 1729}})) (mtSub) (mtSto))
  (numV 1729)
  ```

- Al modificar una caja se debe crear un nuevo registro en la estructura Store con la misma dirección en memoria de la caja que se está mutando. No se debe eliminar el registro anterior, pues la función lookup-sto busca el último valor agregado en la estructura. Se debe regresar el nuevo valor de la caja modificada.

  ```
  > (interp (desugar (parse '{setbox {newbox 1729} 1835})) (mtSub) (mtSto))
  (numV 1835)
  ```

[2]Tipo de sintaxis que hace que un programa sea más "dulce" o fácil de escribir.
[3]Para esta práctica no se agregan nuevas expresiones endulzadas.

- La primitiva `seqn` permite ejecutar instrucciones una después de otra. Esta versión de `seqn` permite múltiples instrucciones, debe ejecutar todas, pero sólo debe regresar el resultado de la última instrucción. Su funcionalidad es equivalente a la de la primitiva `begin` de Racket.

```
> (define expr
   '{with {{a {newbox 1729}}}
      {seqn
         {setbox a 1835}
         {setbox a 405}
         {openbox a}}})
> (interp (desugar (parse expr)) (mtSub) (mtSto))
(numV 405)
```

```
;; interp: BERCFBAEL/L Env Store -> BERCFBAEL/L-Value
 (define (interp expr env store) ...)
```

```
> (interp (op + (list (num 1) (num 2))) (mtSub) (mtSto))
(numV 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.