Lenguajes de Programación

Práctica 3 - Generando código ejecutable

Semestre 2017-1

Fecha de inicio: 18 de septiembre de 2016

Fecha de término: 28 de septiembre de 2016

Instrucciones

- Resolver los siguientes ejercicios de manera clara y ordenada.
- Se deben modificar y entregar los archivos `FWAE.rkt`, `parser.rkt` e `interp.rkt`, más el archivo ejecutable del lenguaje.
- Todas las funciones deben ir acompañadas de al menos 5 pruebas unitarias en un archivo por separado `test.rkt`.
- La entrega es en equipos de máximo 3 integrantes. Seguir los lineamientos especificados en: `http://sites.ciencias.unam.mx/ldp171/formato`.
- Los puntos extra son individuales y sólo se puede escoger uno. El ensayo debe de realizarse **a mano y entregarse a más tardar el 28 de septiembre en la hora de laboratorio**. El punto extra de programación se envía por correo en un archivo `<no.cuenta>_P03.rkt` y un archivo `<no.cuenta>_P03.pdf` (no es necesario comprimirlos) enviar con el asunto `[LDP-EXP3]`. Incluir el nombre completo del autor en el cuerpo del correo.

Descripción general

La práctica consiste en modificar la versión vista en clase del lenguaje `FWAE`.

- **Functions** El intérprete debe evaluar correctamente funciones anónimas y sus aplicaciones. A diferencia del lenguaje visto en clase, esta versión extiende la definición de funciones para que acepten **más de un parámetro**.
- **With** El intérprete debe evaluar correctamente asignaciones locales. A diferencia del lenguaje visto en clase, esta versión extiende la definición de asignaciones locales para que acepten **más de un identificador**.
- **Arithmetic Expression** El intérprete debe evaluar correctamente expresiones aritméticas como son números y operaciones binarias sobre los mismos.

La gramática `FWAE` en `BNF` extendido es la siguiente:

```
<FWAE> ::= <num>
         | <id>
         | {<binop> <FWAE> <FWAE>}
         | {with {{<id> <FWAE>}+} <FWAE>}
         | {fun {<id>+} <FWAE>}
```

```
          | {<FWAE> <FWAE>+}

<num> ::= ... | -2 | - 1 | 0 | 1 | 2 | ...

<id> ::= foo | bar | a | b | x | y | p | ...

<binop> ::= + | - | * | / | % | min | max | pow
```

Algunos ejemplos de expresiones de esta gramática son:

```
1729
{+ 1 2}
{pow 2 3}
{with {{a 2} {b 3}} {+ a b}}
{fun {x y} {pow x y}}
{{fun {x y} {pow x y}} 2 3}
```

Ejercicios

1. **Sintaxis abstracta**

   Anexo a este PDF se encuentra un archivo `FWAE.rkt` que contiene la definición del tipo para construir el árbol de sintaxis abstracta correspondiente. Con el uso de listas modificar el tipo de dato para que acepte asignaciones locales y funciones **multiparamétricas**.

   Hint: Para las funciones usar una lista de símbolos y para las asignaciones locales definir un tipo de dato «`Binding`» que almacene el nombre del identificador (`name`, un simbolo) y el valor asociado (`value`, de tipo `FWAE`).

2. **Análisis sintáctico**

   Anexo a este PDF se encuentra un archivo `parser.rkt` que contiene la definición de una función `(parse expr)` que toma una expresión en sintaxis concreta y regresa su representación en sintaxis abstracta. Modificar esta función para que procese el tipo de dato generado en el ejercicio 1. No olvidar usar la notación `quote` al ejecutar la función.

3. **Intérprete**

   Anexo a este PDF se encuentra un archivo `interp.rkt` que contiene la definición de una función `(interp expr)` que toma un árbol de sintaxis abstracta y regresa su significado. Modificar esta función para que procese el tipo de dato generado en el ejercicio 1. Para la evaluación de funciones y asignaciones locales **usar el algoritmo de sustitución**.

4. **Ejecutable**

   En este ejercicio se debe escribir un intérprete que interactúe con el usuario. Seguir los siguientes pasos para obtener el archivo ejecutable:

a) Una vez terminado el ejercicio 3, escoger un nombre para el lenguaje diseñado, puede ser el nombre del equipo por ejemplo.

b) Usando las características imperativas de Racket, escribir una pequeña función que pida datos al usuario mediante el uso de `read`. Además se debe elegir un nuevo prompt para saber que se está usando el nuevo lenguaje. Se sugiere usar la siguiente función, sin embargo el equipo es libre de dar su propia implementación:

```
;; Función encargada de ejecutar el interprete para que el usuario
;; interactúe con el lenguaje. Para diferenciar el prompt de Racket del
;; nuestro, usamos "(λ)". Aprovechamos los aspectos imperativos del
;; lenguaje.
(define (ejecuta)
   (begin
      (display "(λ) ")
      (define x (read))
      (if (equal? x '{exit})
         (display "")
         (begin
            (display (interp (parse x)))
            (display "\n")
            (ejecuta)))))

;; Llamada a función encargada de iniciar la ejecución del interprete
(display "Bienvenido a FWAE v1.0.\n")
(ejecuta)
```

c) Una vez hechas todas las pruebas y de estar seguros de que la función anterior funciona correctamente y que el intérprete arroja resultados correctos, seleccionar la opción «Crear ejecutable» del menú «Racket».

d) Cambiar el nombre del archivo por el del nombre del lenguaje, seleccionar un ejecutable «Stand-alone» que use como base Racket y dar clic en [`Crear`].

e) Una vez creado el archivo ejecutable, abrir el directorio donde se creo en una terminal y ejecutarlo mediante "`./`", por ejemplo si su lenguaje se llama «ElMejorLenguajeDelMundo» escribir en la terminal: `./ElMejorLenguajeDelmundo`.

Puntos extra

- (1 pto.) De acuerdo al lenguaje `FWAE` diseñado en esta práctica, escribir un ensayo de al menos dos cuartillas donde se expliquen de manera clara y ordenada las etapas para obtener código ejecutable, en cada etapa se deben usar como ejemplos las gramáticas, funciones, algoritmos

y demás técnicas usadas en esta práctica. El ensayo debe contener título, introducción, desarrollo, conclusiones, bibliografía y debe estar debidamente citado.

- (1 pto.) Modificar el lenguaje `FWAE`, para que en lugar de operaciones binarias, acepte operaciones n-arias, por ejemplo, podría aceptar una suma `{+ 1 2 3 4 5}`. Proponer además, al menos 3 operadores nuevos. Justificar el diseño e implementación en un archivo PDF por separado.