Lenguajes de Programación, 2023-1
Práctica 03: Variables y ligado

Manuel Soto Romero Tania Michelle Rubí Rojas Silvia Díaz Gómez

Fecha de entrega: 30 de septiembre de 2022

Objetivos

- Implementar un intérprete para un lenguaje de programación que incluye asignaciones locales a partir de su especificación formal.
- Entender el concepto de azúcar sintáctica y aplicarlo en el proceso de generación de código ejecutable para facilitar la implementación del intérprete.

Especificación del lenguaje

En esta práctica agregaremos a MINILISP asignaciones locales.

Sintaxis concreta

```
<expr> ::= <id>
         | <num>
         | <bool>
         | (<op> <expr>+)
         | (let ([<id> <expr>]+) <expr>)
         | (let* ([<id> <expr>]+) <expr>)

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

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

<bool> ::= #t | #f

<op> ::= + | - | * | / | add1 | sub1 |
       | < | > | = | not | or | and
```

Sintaxis abstracta endulzada

El lenguaje tiene una sintaxis abstracta endulzada que se encarga de traducir las expresiones en sintaxis concreta a una abstracta que sea un mapeo directo.

$$\frac{\texttt{i}\ \texttt{String}}{\texttt{idS(i)}\ ASA} \qquad \frac{\texttt{n} \in \mathbb{Z}}{\texttt{numS(n)}\ ASA} \qquad \frac{\texttt{n} \in \mathbb{B}}{\texttt{booleanS(b)}\ ASA}$$

$$\frac{\texttt{f:String} \quad \texttt{args:}[ASA]}{\texttt{opS(f,args)}\ ASA} \qquad \frac{\texttt{bindings:[(String,ASA)]} \quad \texttt{b:}ASA}{\texttt{letS(bindings,b)}\ ASA}$$

$$\frac{\texttt{bindings:[(String,ASA)]} \quad \texttt{b:}ASA}{\texttt{letS*(bindings,b)}\ ASA}$$

Sintaxis abstracta desendulzada

Una vez obtenida la sintaxis abstracta endulzada, se procede a detectar expresiones que puedan ser equivalentes para evitar hacer un doble análisis en la especificación de la semántica.

En este caso, tenemos un par de expresiones con azúcar sintáctica:

- **Operaciones *n*-arias.** Las operaciones pueden traducirse a expresiones de aridad 1 o 2 respectivamente.
  - Operaciones unarias: Al sólo recibir un parámetro sólo basta con detectar a este tipo de operaciones: {
  - Operaciones binarias: `(f x1 x2 x3 ... xn)` se traduce en `(f x1 (f x2 ... xn ...))` con `f` alguna de las siguientes operaciones {`+`, `-`, `*`, `/`, `<`, `>`, `=`, `or`, `and`}. Es decir, aplica de dos en dos los operandos de acuerdo a la operación.
- **Asignaciones locales:** Nuestras expresiones `let*` son una versión endulzada de expresiones `let` anidadas. Por ejemplo, la expresión:

```
(let* ([a 2][b 3])
        (+ a b))
```

Es equivalente a:

```
(let ([a 2])
   (let ([b 3])
      (+ a b)))
```

Dicho esto, se presenta la sintaxis abstracta desendulzada de nuestro lenguaje. Observa que hemos simplificado y eliminado algunas reglas.

$$\frac{\texttt{i}\ \texttt{String}}{\texttt{id(i)}\ ASA} \qquad \frac{\texttt{n} \in \mathbb{B}}{\texttt{boolean(b)}\ ASA} \qquad \frac{\texttt{f}:\texttt{String} \quad \texttt{i}\ ASA \quad \texttt{d}\ ASA}{\texttt{binop(f,i,d)}\ ASA}$$

$$\frac{\texttt{n} \in \mathbb{Z}}{\texttt{num(n)}\ ASA} \qquad \frac{\texttt{f}:\texttt{String} \quad \texttt{arg}\ ASA}{\texttt{uop(f,arg)}\ ASA} \qquad \frac{\texttt{bindings}:\texttt{[(String,ASA)]} \quad \texttt{b}:ASA}{\texttt{let(bindings,b)}\ ASA}$$

Semántica

Usaremos una semántica operacional de paso grande mediante la sintaxis abstracta del lenguaje.

Números

Los números se interpretan a la representación de números correspondiente al lenguaje anfitrión.

$$\frac{}{\texttt{num}(n) \Rightarrow \hat{n}}$$

Booleanos

Los booleanos se interpretan a la representación de booleanos correspondiente al lenguaje anfitrión.

$$\frac{}{\texttt{boolean}(b) \Rightarrow \hat{b}}$$

Operaciones unarias

Las operaciones unarias se interpretan mediante la operación correspondiente. El comportamiento de las operaciones es el habitual.

$$\frac{\texttt{elige}(f) = g \quad \texttt{a} \Rightarrow a_v}{\texttt{uop}(f, a) \Rightarrow g\ \hat{a}_v}$$

Operaciones binarias

Las operaciones unarias se interpretan mediante la operación correspondiente. El comportamiento de las operaciones es el habitual.

$$\frac{\texttt{elige}(f) = g \quad i \Rightarrow i_v \quad d \Rightarrow i_v}{\texttt{binop}(f, i, d) \Rightarrow g\ \hat{i_v}\ d_v}$$

Asignaciones locales

Las operaciones unarias se interpretan mediante la operación correspondiente sustitución. El comportamiento se define como la interpretación del cuerpo luego de sustituir el identificador por el valor correspondiente.

$$\frac{\texttt{c} \Rightarrow \hat{c_v}}{\texttt{let}(empty, c) \Rightarrow \hat{c_v}}$$

$$\frac{\texttt{v} \Rightarrow num(\hat{v_v}) \quad \texttt{let(bs,c[i := } \hat{v_v}\texttt{])} \Rightarrow \hat{c_v}}{\texttt{let}((i, v) : bs, c) \Rightarrow \hat{c_v}}$$

Por ejemplo para:

$$\frac{num(2) \Rightarrow num(2) \quad \dfrac{\overline{num(2) \Rightarrow \hat{2}}}{\texttt{let}([], num(2)) \Rightarrow \hat{2}}}{\texttt{let}([(\text{"}a\text{"}, num(2)], id(\text{"}a\text{"})) \Rightarrow \hat{2}}$$

Ejercicios

Adjunto a este archivo PDF se encuentran una serie de archivos con secciones que debes completar para dar vida a tu intérprete. Descárgalos y realiza lo siguiente:

Ejercicio 1: Análisis léxico y sintáctico (25 pts.)

Con el apoyo de HAPPY define el proceso de análisis léxico con el nombre `lexer` y análisis sintáctico con el nombre `parser` para el lenguaje. El archivo `Grammars.y` contiene un esqueleto con estas definiciones tu misión es completar:

1. La sección de gramáticas ubicada después de los símbolos `%%`, ahí encontrarás el siguiente comentario:
   `/* DEFINE AQUÍ TUS GRAMÁTICAS PARA EL PARSER. */`
2. La función `lexer`.

Pon especial atención a los tipos `Token` y `ASAS` definidos en el archivo.

Ejercicio 2: Azúcar sintáctica (40 pts.)

Completa la función `desugar` contenida en el archivo `Desugar.hs`. La función debe implementarse de acuerdo con la definición de la eliminación de azúcar sintáctica dada en este documento.

Pon especial atención al tipos `ASA` definido en el archivo.

Ejercicio 3: Análisis semántico (25 pts.)

Completa la función `interp` contenida en el archivo `Interp.hs`. La función debe implementarse de acuerdo con la definición de las reglas de la semántica operacional dada en este documento. Recuerda implementar la sustitución.

La salida debe ser algo de tipo `Value` con el fin de homogeneizar resultados.

Ejercicio 4: Pruebas (10 pts.)

Una vez terminados los ejercicios anteriores, ejecuta el archivo `Practica03.hs` y prueba tus implementaciones. **Este archivo no se debe modificar.**