tarea-07

Mario Alejandro Castro Lerma

Noviembre de 2023

1 Extiende el lenguaje agregando un nuevo operador minus que toma un argumento n y regresa -n. Por ejemplo, el valor de minus(-(minus(5),9)) debe ser 14.

Analisis léxica: minus = minus
Sintaxis Concreta: Expression → minus(Expression)
Sintaxis Abstracta: (minus-exp exp1)
Especificación Semantica:

$$\frac{\mathcal{E}(exp, \rho) = val1}{\mathcal{E}(minus(exp), \rho) = -val1}$$

2 Extiende el lenguaje agregando operadores para la suma, multiplicación y cociente de enteros.

2.1 Suma

Especificación léxica: suma = +
Sintaxis Concreta: Expression → +(Expression, Expression)
Sintaxis Abstracta: (plus-exp exp1 exp2)
Especificación Semántica:

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2 \;\; (val1 + val2) = val3}{\mathcal{E}(+(exp1, exp2), \rho) = val3}$$

2.2 Multiplicación

Especificación léxica: multiplicación = *
Sintaxis Concreta: Expression → *(Expression, Expression)
Sintaxis Abstracta: (mult-exp exp1 exp2)
Especificación Semántica:

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2 \;\; (val1 * val2) = val3}{\mathcal{E}(*(exp1, exp2), \rho) = val3}$$

2.3 División

Especificación léxica: división = /
Sintaxis Concreta: Expression → /(Expression, Expression)
Sintaxis Abstracta: (div-exp exp1 exp2)
Especificación Semántica:

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2 \;\; \mathcal{E}(val2, \rho) \neq 0 \;\; (val1/val2) = val3}{\mathcal{E}(/(exp1, exp2), \rho) = val3}$$

3 Agrega un predicado de igualdad numérica equal? y predicados de orden greater? y less? al conjunto de operaciones del lenguaje LET.

3.1 equal?

Especificación léxica: equal? = equal?
Sintaxis Concreta: Expression → equal?(Expression, Expression)
Sintaxis Abstracta: (equal?-exp exp1 exp2)
Especificación Semántica:

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2 \;\; val1 = val2}{\mathcal{E}(equal?(exp1, exp2), \rho) = True}$$

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2 \;\; val1 \neq val2}{\mathcal{E}(equal?(exp1, exp2), \rho) = False}$$

3.2 greater?

Especificación léxica: greater? = greater?
Sintaxis Concreta: Expression → greater?(Expression, Expression)
Sintaxis Abstracta: (greater?-exp exp1 exp2)
Especificación Semántica:

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2 \;\; val1 > val2}{\mathcal{E}(greater?(exp1, exp2), \rho) = True}$$

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2 \;\; val1 < val2}{\mathcal{E}(greater?(exp1, exp2), \rho) = False}$$

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2 \;\; val1 = val2}{\mathcal{E}(greater?(exp1, exp2), \rho) = False}$$

3.3 less?

Especificación léxica: less? = less?
Sintaxis Concreta: Expression → less?(Expression, Expression)
Sintaxis Abstracta: (less?-exp exp1 exp2)
Especificación Semántica:

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2 \;\; val1 < val2}{\mathcal{E}(less?(exp1, exp2), \rho) = True}$$

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2 \;\; val1 > val2}{\mathcal{E}(less?(exp1, exp2), \rho) = False}$$

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2 \;\; val1 = val2}{\mathcal{E}(less?(exp1, exp2), \rho) = False}$$

4 Agrega operaciones de procesamiento de listas al lenguaje, incluyendo cons , car , cdr , null? y emptylist. Una lista debe poder contener cualquier valor expresado, incluyendo otra lista.

4.1 cons

Especificación léxica: cons = cons
Sintaxis Concreta: Expression → cons(Expression, Expression)

Sintaxis Abstracta: (cons-exp exp1 exp2)
Especificación Semántica:

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(exp2, \rho) = val2}{\mathcal{E}(cons(exp1, exp2), \rho) = Pair(val1, val2)}$$

4.2 car

Especificación léxica: car = car
Sintaxis Concreta: Expression → car(Expression)
Sintaxis Abstracta: (car-exp exp1)
Especificación Semántica:

$$\frac{\mathcal{E}(exp1, \rho) = Pair(val1, val2)}{\mathcal{E}(car(exp1), \rho) = val1}$$

4.3 cdr

Especificación léxica: cdr = cdr
Sintaxis Concreta: Expression → cdr(Expression)
Sintaxis Abstracta: (cdr-exp exp1)
Especificación Semántica:

$$\frac{\mathcal{E}(exp1, \rho) = Pair(val1, val2)}{\mathcal{E}(cdr(exp1), \rho) = val2}$$

4.4 null?

Especificación léxica: null? = null?
Sintaxis Concreta: Expression → null?(Expression)
Sintaxis Abstracta: (null?-exp exp1)
Especificación Semántica:

$$\frac{\mathcal{E}(exp1, \rho) = emptylist}{\mathcal{E}(null?(exp1), \rho) = True}$$

$$\frac{\mathcal{E}(exp1, \rho) \neq emptylist}{\mathcal{E}(null?(exp1), \rho) = False}$$

4.5 emptylist

Especificación léxica: emptylist = emptylist
Sintaxis Concreta: Expression → emptylist
Sintaxis Abstracta: (emptylist-exp)
Especificación Semántica:

$$\overline{\mathcal{E}(emptylist, \rho) = emptylist}$$

5 Agrega una operación list al lenguaje. Esta operación debe tomar cualquier cantidad de argumentos y regresar un valor expresado de la lista de sus valores.

Especificación léxica: list = list
Sintaxis Concreta:
Expression → list(Exps)
Exps → ϵ
$\|(Exps1)$
Exps1 → Expression
$\|(Expression, Exps1)$
Sintaxis Abstracta: (list-exp exps)
Especificación Semántica:

$$\frac{(null?(exps)) = True}{\mathcal{E}(list(exps), \rho) = emptylist}$$

$$\frac{\mathcal{E}(exp1, \rho) = val1 \;\; \mathcal{E}(list(exps...), \rho) = val2}{\mathcal{E}(list(exp1, exps...), \rho) = Pair(val1, val2)}$$

7 Incorpora al lenguaje expresiones cond usando Expression → cond $\{Expression => Expression\}$* end

Especificación léxica: cond = cond , end = end, => = =>
Sintaxis Concreta:
Expression → cond CondExps
CondExps → *end*

$\|CondExps1$
CondExps1 $\rightarrow$ $\{Expression \ => \ Expression\}$ CondExps1
$\|$ $\{Expression \ => \ Expression\}$
Sintaxis Abstracta: (cond-exp CondExps end)
Especificación Semántica:

$$\frac{\mathcal{E}(exp1, \rho) = True \ \ \mathcal{E}(exp2, \rho) = val2}{\mathcal{E}(cond(exp1 => exp2... \ end), \rho) = val2}$$

$$\frac{\mathcal{E}(exp1, \rho) = False \ \ \mathcal{E}(cond(exp3 => exp4... \ end), \rho) = val}{\mathcal{E}(cond(exp1 => exp2, exp3 => exp4... \ end), \rho) = val}$$

$$\frac{}{\mathcal{E}(cond(end), \rho) = \mathcal{E}(+(zero?(1), zero?(0)), \rho)}$$

8 Cambia los valores del lenguaje para que los enteros sean los únicos valores expresados. Modifica if para que el valor de 0 sea tratado como falso y todos los otros valores sean tratados como verdaderos. Modifica los predicados de manera consistente.

Especificación léxica: No hay cambios
Sintaxis Concreta: No hay cambios
Sintaxis Abstracta: No hay cambios
Especificación Semántica:
Para if:

$$\frac{\mathcal{E}(exp1, \rho) \neq 0 \ \ \mathcal{E}(exp2, \rho) = val2}{\mathcal{E}(if \ exp1 \ then \ exp2 \ else \ exp3), \rho) = val2}$$

$$\frac{\mathcal{E}(exp1, \rho) = 0 \ \ \mathcal{E}(exp3, \rho) = val3}{\mathcal{E}(if \ exp1 \ then \ exp2 \ else \ exp3), \rho) = val3}$$

Para zero?:

$$\frac{\mathcal{E}(exp1, \rho) = 0}{\mathcal{E}(zero?(exp1)), \rho) = 1}$$

$$\frac{\mathcal{E}(exp1, \rho) \neq 0}{\mathcal{E}(zero?(exp1)), \rho) = 0}$$

9 Como una alternativa al ejercicio anterior, agrega una nueva categoría sintáctica Bool-exp de expresiones booleanas al lenguaje. ¿En dónde terminan estando los predicados del ejercicio 3 con este cambio?

Especificación léxica: No hay cambios
Sintaxis Concreta:
Expression → ***if*** Bool-exp ***then*** Expression ***else*** Expression
Bool-exp → zero?(Expression)
Bool-exp → true
Bool-exp → false
Sintaxis Abstracta:
(if-exp bool-exp exp2 exp3)
(zero?-bool-exp exp1)
(bool-exp *bool*)
Especificación Semántica:
Para if:

$$\frac{\mathcal{B}(bool-exp,\rho)=true \;\; \mathcal{E}(exp2,\rho)=val2}{\mathcal{E}(if\ bool-exp\ then\ exp2\ else\ exp3),\rho)=val2}$$

$$\frac{\mathcal{B}(bool-exp,\rho)=false \;\; \mathcal{E}(exp3,\rho)=val3}{\mathcal{E}(if\ bool-exp\ then\ exp2\ else\ exp3),\rho)=val3}$$

Para zero?:

$$\frac{\mathcal{E}(exp1,\rho)=0}{\mathcal{B}(zero?(exp1)),\rho)=true}$$

$$\frac{\mathcal{E}(exp1,\rho)\neq 0}{\mathcal{B}(zero?(exp1)),\rho)=false}$$

9.1 Con los predicados del ejercicio 3:

Especificación léxica: No hay cambios
Sintaxis Concreta:
Bool-exp → equal?(Expression, Expression)
Bool-exp → greater?(Expression, Expression)
Bool-exp → less?(Expression, Expression)
Sintaxis Abstracta:
(equal?-bool-exp exp1 exp2)
(greater?-bool-exp exp1 exp2)
(less?-bool-exp exp1 exp2)

Especificación Semántica:
Para equal:

$$\frac{\mathcal{E}(exp1,\rho) = val1 \;\; \mathcal{E}(exp2,\rho) = val2 \;\; val1 = val2}{\mathcal{B}(equal?(exp1, exp2),\rho) = true}$$

$$\frac{\mathcal{E}(exp1,\rho) = val1 \;\; \mathcal{E}(exp2,\rho) = val2 \;\; val1 \neq val2}{\mathcal{B}(equal?(exp1, exp2),\rho) = False}$$

Para greater?:

$$\frac{\mathcal{E}(exp1,\rho) = val1 \;\; \mathcal{E}(exp2,\rho) = val2 \;\; val1 > val2}{\mathcal{B}(greater?(exp1, exp2),\rho) = true}$$

$$\frac{\mathcal{E}(exp1,\rho) = val1 \;\; \mathcal{E}(exp2,\rho) = val2 \;\; val1 < val2}{\mathcal{B}(greater?(exp1, exp2),\rho) = false}$$

$$\frac{\mathcal{E}(exp1,\rho) = val1 \;\; \mathcal{E}(exp2,\rho) = val2 \;\; val1 = val2}{\mathcal{B}(greater?(exp1, exp2),\rho) = false}$$

Para less?:

$$\frac{\mathcal{E}(exp1,\rho) = val1 \;\; \mathcal{E}(exp2,\rho) = val2 \;\; val1 < val2}{\mathcal{B}(less?(exp1, exp2),\rho) = true}$$

$$\frac{\mathcal{E}(exp1,\rho) = val1 \;\; \mathcal{E}(exp2,\rho) = val2 \;\; val1 > val2}{\mathcal{B}(less?(exp1, exp2),\rho) = false}$$

$$\frac{\mathcal{E}(exp1,\rho) = val1 \;\; \mathcal{E}(exp2,\rho) = val2 \;\; val1 = val2}{\mathcal{B}(less?(exp1, exp2),\rho) = false}$$

10 Modifica la implementación del intérprete agregando una nueva operación print que toma un argumento, lo imprime, y regresa el entero 1. ¿Por qué esta operación no es expresable en nuestro método de especificación formal?

Con la forma que hemos estado definiendo nuestras operaciones, siempre regresamos un valor, pero no hemos realizado en ninguna de estas una interacción con la consola o algo parecido, ya que lo máximo que nos permite la especificación formal es el especificar valores y como se utilizan o interactúan estos, pero finalmente se entrega un valor. Especificar interacciones con el usuario van mas allá de las capacidades de la especificación formal.

11 Extiende el lenguaje para que las expresiones let puedan vincular una cantidad arbitraria de variables usando Expression → let $\{Identifier = Expression\}$* in Expression

Especificación léxica: No hay cambios
Sintaxis Concreta:
Expression → let LetExps in Expression
LetExps → ϵ
$\|LetExps1$
LetExps1 → $Identifier\ =\ Expression$ LetExps1
$\|\ Identifier\ =\ Expression$
Sintaxis Abstracta: (let-exp letExps body)
Especificación Semántica:

$$\frac{}{\mathcal{E}(let\ in\ body, \rho) = \mathcal{E}(body, \rho)}$$

$$\frac{\mathcal{E}(exp1, \rho) = val1}{\mathcal{E}(let\ x = exp1\ in\ body, \rho) = \mathcal{E}(body, [x : val1]\rho)}$$

$$\frac{\mathcal{E}(exp1, \rho) = val1, \mathcal{E}(exp2, \rho) = val2, ...}{\mathcal{E}(let\ x_1 = exp1, x_2 = exp2, ...\ in\ body, \rho) = \mathcal{E}(body, [x : val1, x_2 : val2, ...]\rho)}$$

12 Extiende el lenguaje con una expresión let* que funciona como en Racket.

Especificación léxica: let* = let*
Sintaxis Concreta:
Expression → let* LetExps in Expression
LetExps → ϵ
$\|LetExps1$
LetExps1 → $Identifier\ =\ Expression$ LetExps1
$\|\ Identifier\ =\ Expression$
Sintaxis Abstracta: (let*-exp letExps body)
Especificación Semántica:

$$\frac{}{\mathcal{E}(let\ in\ body, \rho) = \mathcal{E}(body, \rho)}$$

$$\frac{\mathcal{E}(exp1, \rho) = val1, x_1 = val1\ \ \mathcal{E}(exp2, [x_1 : val1]\rho) = val2, x_2 = val2\ \ \mathcal{E}(exp3, [x_1 : val1, x_2 : val2]\rho) = val3...}{\mathcal{E}(let\ x_1 = exp1, x_2 = \mathcal{E}(exp2, [x : val1]\rho), ...\ in\ body, \rho) = \mathcal{E}(body, [x : val1, x_2 : val2, ...]\rho)}$$

13 Agrega una expresión al lenguaje de acuerdo a la siguiente regla: Expression → unpack $\{Identifier\}$* = Expression in Expression.

Especificación léxica: unpack = unpack
Sintaxis Concreta:
Expression → unpack UnpackIds = Expression in Expression
UnpackIds → ϵ
$\|UnpackIds1$
UnpackIds1 → $Identifier$ UnpackIds1
$\|$ $Identifier$
Sintaxis Abstracta: (unpack-exp UnpackIds Expression body)
Especificación Semántica:

$$\frac{\mathcal{E}(body, \rho) = val1}{\mathcal{E}(unpack\ emptylist = emptylist\ in\ body, \rho) = val1}$$

$$\frac{\mathcal{E}(exp1, \rho) = val1,\ \mathcal{E}(exp2, \rho) = val2, ...\ \ \mathcal{E}(body, [x_1 : val1, x_2 : val2, ...]\rho) = val}{\mathcal{E}(unpack\ x_1\ x_2... = list(exp1, exp2, ...)\ in\ body) = val}$$

$$\frac{\mathcal{E}(null?(exp1)) = true}{\mathcal{E}(unpack\ \ x\ y\ z... = exp1\ in\ body, \rho) = \mathcal{E}(+(zero?(1), zero?(0)), \rho)}$$