



**“El saber de mis hijos  
hará mi grandeza”**

UNIVERSIDAD DE SONORA

TAREA 7

## **Lenguajes de Programación**

*Gutierrez Navarro Gustavo*

November 27, 2023

## 1 Preguntas:

1. Extiende el lenguaje agregando un nuevo operador minus que toma un argumento  $n$  y regresa  $-n$ .

### Especificacion Lexica

minus = minus

### Especificacion Sintactica

#### Concreta

$Expression \rightarrow \mathbf{minus}(Expression)$

#### Abstracta

(minus-exp  $exp1$ )

### Especificacion Semantica

$$\frac{\mathcal{E}(-(0, exp_1), \rho) = x}{\mathcal{E}(\mathbf{minus}(exp_1), \rho) = x}$$

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

### Especificacion Lexica

suma = +  
multiplicacion = \*  
cociente(enteros) = /

### Especificacion Sintactica

#### Concreta

$Expression \rightarrow + (Expression, Expression)$

$Expression \rightarrow - (Expression, Expression)$

$Expression \rightarrow / (Expression, Expression)$

#### Abstracta

(sum-exp  $exp1$   $exp2$ )

(mult-exp  $exp1$   $exp2$ )

(div-exp  $exp1$   $exp2$ )

### Especificacion Semantica

$$\frac{\mathcal{E}(exp_1, \rho)=x, \mathcal{E}(exp_2, \rho)=y}{\mathcal{E}(+(exp_1, exp_2), \rho)=x+y} \quad \frac{\mathcal{E}(exp_1, \rho)=x, \mathcal{E}(exp_2, \rho)=y}{\mathcal{E}(* (exp_1, exp_2), \rho)=x*y}$$

$$\frac{\mathcal{E}(exp_2, \rho)=0}{\mathcal{E}(/ (exp_1, exp_2), \rho)=error} \quad \frac{\mathcal{E}(exp_1, \rho)=x, \mathcal{E}(exp_2, \rho)=y}{\mathcal{E}(/ (exp_1, exp_2), \rho)=x/y}$$

### 3. Agrega un predicado de igualdad numérica *equal?* y predicados de orden *greater?* y *less?*.

#### Especificacion Lexica

equal? = equal?  
 > = greater?  
 < = less?

#### Especificacion Sintactica

Concreta

*Expression* → **equal?** (*Expression*, *Expression*)  
*Expression* → **greater?** (*Expression*, *Expression*)  
*Expression* → **less?** (*Expression*, *Expression*)

Abstracta

(equal?-exp *exp1 exp2*)  
 (greater?-exp *exp1 exp2*)  
 (less?-exp *exp1 exp2*)

#### Especificacion Semantica

$$\frac{\mathcal{E}(\mathcal{E}(exp_1, \rho) - \mathcal{E}(exp_2, \rho))\rho = 0}{\mathcal{E}(\text{equal?}(exp_1, exp_2), \rho) = \#t} \quad \frac{\mathcal{E}(\mathcal{E}(exp_1, \rho) - \mathcal{E}(exp_2, \rho))\rho \neq 0}{\mathcal{E}(\text{equal?}(exp_1, exp_2), \rho) = \#f}$$

$$\frac{\mathcal{E}(\mathcal{E}(exp_1, \rho) - \mathcal{E}(exp_2, \rho))\rho > 0}{\mathcal{E}(\text{greater?}(exp_1, exp_2), \rho) = \#t} \quad \frac{\mathcal{E}(\mathcal{E}(exp_1, \rho) - \mathcal{E}(exp_2, \rho))\rho \leq 0}{\mathcal{E}(\text{greater?}(exp_1, exp_2), \rho) = \#f}$$

$$\frac{\mathcal{E}(\mathcal{E}(exp_1, \rho) - \mathcal{E}(exp_2, \rho))\rho < 0}{\mathcal{E}(\text{less?}(exp_1, exp_2), \rho) = \#t} \quad \frac{\mathcal{E}(\mathcal{E}(exp_1, \rho) - \mathcal{E}(exp_2, \rho))\rho \geq 0}{\mathcal{E}(\text{less?}(exp_1, exp_2), \rho) = \#f}$$

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.

#### Especificacion Lexica

```
cons = cons
car = car
cdr = cdr
null? = null?
emptylist = emptylist
```

#### Especificacion Sintactica

##### Concreta

```
Expression → cons (Expression, Expression)
Expression → car (Expression)
Expression → cdr (Expression)
Expression → null? (Expression)
Expression → emptylist ()
```

##### Abstracta

```
(cons-exp exp1 exp2)
(car-exp exp1)
(cdr-exp exp1)
(null?-exp exp1)
(emptylist-exp exp1)
```

#### Especificacion Semantica

```
(value-of (cons exp1 exp2)) = (pair-val (value-of exp1) (value-of exp2))
(value-of (car (cons exp1 exp2))) = (value-of exp1)
(value-of (cdr (cons exp1 exp2))) = (value-of exp2)
(value-of (null? exp1)) = (if (= exp1 emptylist) #t #f)
(value-of emptylist) = null
```

## 5. Agrega una operación *list* al lenguaje

### Especificacion Lexica

list = list

### Especificacion Sintactica

Concreta

$Expression \rightarrow \mathbf{list}(Expressions)$

$Expressions \rightarrow \epsilon$

$Expressions \rightarrow Expression1$

$Expressions1 \rightarrow Expression$

$Expressions1 \rightarrow Expression, Expressions1$

Abstracta

(list-exp *exps*)

### Especificacion Semantica

$$\frac{\mathcal{E}((\text{null? } \text{exps}), \rho) = \#t}{\mathcal{E}((\text{list-exp } \text{exps}), \rho) = \text{emptylist}}$$
$$\frac{\mathcal{E}(\text{exp}_0, \rho) = \text{val}_1, \mathcal{E}(\text{list}(\text{exp}_1, \dots), \rho) = \text{val}_2}{\mathcal{E}(\text{list}(\text{exp}_0, \text{exp}_1, \dots), \rho) = \text{pair}(\text{val}_1, \text{val}_2)}$$

## 7. Incorpora al lenguaje expresiones *cond* en base a la siguiente gramatica

$Expression \rightarrow \mathbf{cond}\{Expression \Rightarrow Expression\}^* \mathbf{end}$

### Especificacion Lexica

cond = cond

end = end

=> = =>

### Especificacion Sintactica

Concreta

$Expression \rightarrow \mathbf{cond}\{Expression \Rightarrow Expression\}^* \mathbf{end}$

Abstracta

(cond-exp *exps*)

### Especificacion Semantica

$$\frac{\mathcal{E}(exp_1, \rho) = \#t, \mathcal{E}(exp_2, \rho) = x}{\mathcal{E}(\text{cond}(exp_1, exp_2, exp, \dots), \rho) = x} \quad \frac{\mathcal{E}(exp_1, \rho) = \#f, \mathcal{E}(\text{cond}(exp, \dots)), \rho) = x}{\mathcal{E}(\text{cond}(exp_1, exp_2, exp, \dots), \rho) = x}$$
$$\overline{\mathcal{E}(\text{cond}(), \rho) = error}$$

8. Cambia los valores del lenguaje para que los enteros sean los únicos valores expresados con todos los cambios que esto implica.

### Especificacion Lexica

Sin cambios.

### Especificacion Sintactica

Concreta

Sin cambios.

Abstracta

Sin cambios.

### Especificacion Semantica

$$\frac{\mathcal{E}(exp_1, \rho) = x}{\mathcal{E}(\text{zero?}(exp_1, \rho) = x)} \quad \frac{\mathcal{E}(exp_1, \rho) = 0, \mathcal{E}(exp_2, \rho) = x}{\mathcal{E}(\text{if}(exp_1, exp_2, exp_3), \rho) = x} \quad \frac{\mathcal{E}(exp_1, \rho) \neq 0, \mathcal{E}(exp_3, \rho) = y}{\mathcal{E}(\text{if}(exp_1, exp_2, exp_3), \rho) = y}$$

9. Agrega una nueva categoría sintáctica *Bool* – *exp* de expresiones booleanas al lenguaje. Cambia la producción para expresiones condicionales para que sea

*Expression*  $\rightarrow$  **if** *Bool* – *exp* **then** *Expression* **else** *Expression*

### Especificacion Lexica

Sin cambios.

## Especificacion Sintactica

### Concreta

$Expression \rightarrow \text{if } Bool - exp \text{ then } Expression \text{ else } Expression$

$Bool - exp \rightarrow \#t$

$Bool - exp \rightarrow \#f$

$Bool - exp \rightarrow \text{zero?}(Expression)$

$Bool - exp \rightarrow \text{equal?}(Expression, Expression)$

$Bool - exp \rightarrow \text{greater?}(Expression, Expression)$

$Bool - exp \rightarrow \text{less?}(Expression, Expression)$

### Abstracta

$(\text{if } boolexp \ exp1 \ exp2)$

$(\text{bool-exp } bool) \rightarrow (\text{bool puede ser } \#t \text{ o } \#f)$

$(\text{zero?-boolexp } exp1)$

$(\text{equal?-boolexp } exp1, exp2)$

$(\text{greater?-boolexp } exp1, exp2)$

$(\text{less?-boolexp } exp1, exp2)$

## Especificacion Semantica

$(\text{value-of } (\text{bool-exp } b), \rho) = (\text{bool-val } b)$

$$\frac{\mathcal{E}(boolexp, \rho) = \#t \quad \mathcal{E}(exp1, \rho) = x}{\mathcal{E}(\text{if}(boolexp, exp1, exp2), \rho) = x} \quad \frac{\mathcal{E}(boolexp, \rho) = \#f \quad \mathcal{E}(exp2, \rho) = y}{\mathcal{E}(\text{if}(boolexp, exp1, exp2), \rho) = y}$$

Los predicados del ejercicio 3 se quedarian igual.

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 ex-presable en nuestro método de especificación formal?**

Uno de nuestros puntos débiles al especificar de manera formal es la limitación para conectarnos con el entorno externo.

Observando más de cerca, todos los procesos comparten una característica fundamental: reciben alguna entrada y generan una salida, a veces en espacios diferentes, pero su operación es esencialmente similar.

La función *print* puede ser especificada hasta cierto punto; sin embargo, el concepto de imprimir no existe en este contexto, lo que hace imposible expresar la acción de imprimir. El proceso de la especificación formal ter-

mina con el evaluador que devuelve un valor pero nosotros metemos de por medio una impresion del valor.

11. **Extiende el lenguaje para que las expresiones *let* puedan vincular una cantidad arbitraria de variables, usando la producción,**

$$Expression \rightarrow \mathbf{let} \{Identifier = Expression\}^* \mathbf{in} Expression$$

#### Especificacion Lexica

Sin cambios.

#### Especificacion Sintactica

##### Concreta

$$Expression \rightarrow \mathbf{let} \{Identifier = Expression\}^* \mathbf{in} Expression$$

$$Iexprs \rightarrow \epsilon$$

$$Iexprs \rightarrow IExprs1$$

$$Iexprs1 \rightarrow Identifier Expression$$

$$Iexprs1 \rightarrow Identifier Expression Iexprs1$$

##### Abstracta

$$(\mathbf{let}\text{-exp } Iexprs \text{ body})$$

#### Especificacion Semantica

$$\frac{\mathcal{E}(exp_1, \rho) = val_1 \quad \mathcal{E}(exp, \rho) = val \quad \dots}{\mathcal{E}(\mathbf{let}((id_1 \ exp_1 \ id \ exp \ \dots) \ body), \rho) = \mathcal{E}(\mathbf{let}((\ ) \ body), [id_1:val_1, id:val, \dots] \rho)}$$

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

12. **Extiende el lenguaje con una expresión *let\**.**

#### Especificacion Lexica

$$\mathbf{Let}^* = \mathbf{Let}^*$$

#### Especificacion Sintactica

##### Concreta

$$Expression \rightarrow \mathbf{let}^* \{Identifier = Expression\}^* \mathbf{in} Expression$$

*Iexprs* y *Iexprs1* se reutilizan.



Abstracta  
 (let-exp\* *Iexp* *body*)

$$\frac{\mathcal{E}(exp_1, \rho) = val_1}{\mathcal{E}(\text{let}^*((id_1 \ exp_1 \ id \ exp \ \dots) \ body), \rho) = \mathcal{E}(\text{let}^*((id \ exp \ \dots) \ body), [id_1 : val_1] \rho)}$$

$$\frac{\mathcal{E}(body, \rho) = x}{\mathcal{E}(\text{let}^*((id) \ body), \rho) = x}$$

### 13. Agrega una expresión al lenguaje de acuerdo a la siguiente regla,

*Expression*  $\rightarrow$  **unpack**{*Identifier*}\* = *Expression* **in** *Expression*

tal que *unpack* x y z = *lst* in ... vincula x , y y z a los elementos de *lst* si *lst* es una lista con exactamente tres elementos, reportando un error en otro caso.

#### Especificacion Lexica

unpack = unpack.

#### Especificacion Sintactica

Concreta  
*Expression*  $\rightarrow$  **unpack**{*Identifier*}\* = *Expression* **in** *Expression*  
*Ids*  $\rightarrow \epsilon$   
*Ids*  $\rightarrow Ids1$   
*Ids1*  $\rightarrow Identifier$   
*Ids1*  $\rightarrow Identifier \ Ids1$

Abstracta  
 (unpack-exp *Ids* *expl* *body*)      *expl*, *body* = *Expression*

#### Especificacion Semantica

$$\frac{\mathcal{E}(exp_1, \rho) = pair(val_1, rest)}{\mathcal{E}(\text{unpack}((id_1 \ id \ \dots) \ exp_1 \ body), \rho) = \mathcal{H}(\text{list}(id_1 \ id \ \dots), pair(val_1, rest), body, \rho)}$$

$$\frac{\mathcal{E}(exp_1, \rho) = val_1 \quad \mathcal{E}(body, \rho) = val}{\mathcal{E}(\text{unpack}((id) \ exp_1 \ body), \rho) = val} \quad \mathcal{H}(\text{list}(x_1 \ x \ \dots), pair(val_0, rest), body, \rho) = \mathcal{H}(\text{list}(x \ \dots), rest, body, [x_1 = val_0] \rho)$$

$$\frac{\mathcal{E}(body, \rho) = val_1}{\mathcal{H}(\text{emptylist}, \text{emptylist}, body, \rho) = val_1} \quad \frac{val_1 \neq \text{emptylist}}{\mathcal{H}(\text{list}(), val_1, body, \rho) = \text{error}}$$

$$\frac{val_1 = \text{emptylist}}{\mathcal{H}(\text{list}(x_1 \ x \ \dots), val_1, body, \rho) = \text{error}}$$