Universidad Nacional Autónoma de México
Facultad de Ciencias
Lenguajes de Programación
2025-1
Tarea 03

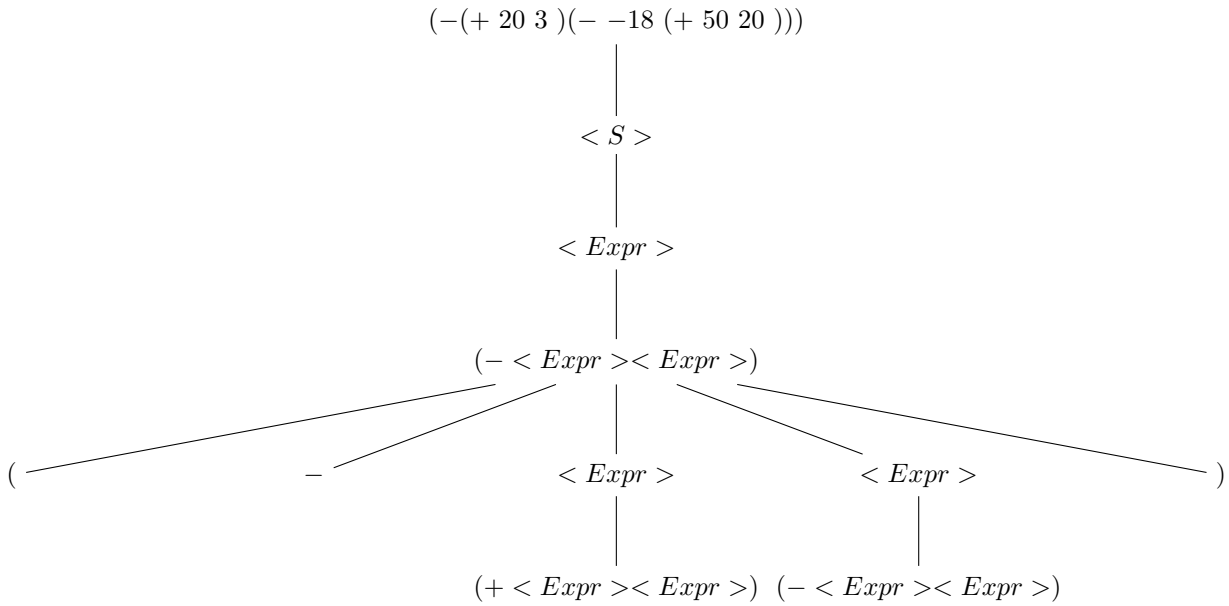
Victoria Morales Ricardo Maximiliano
Sánchez Estrada Alejandro
Suárez Ortiz Joshua Daniel

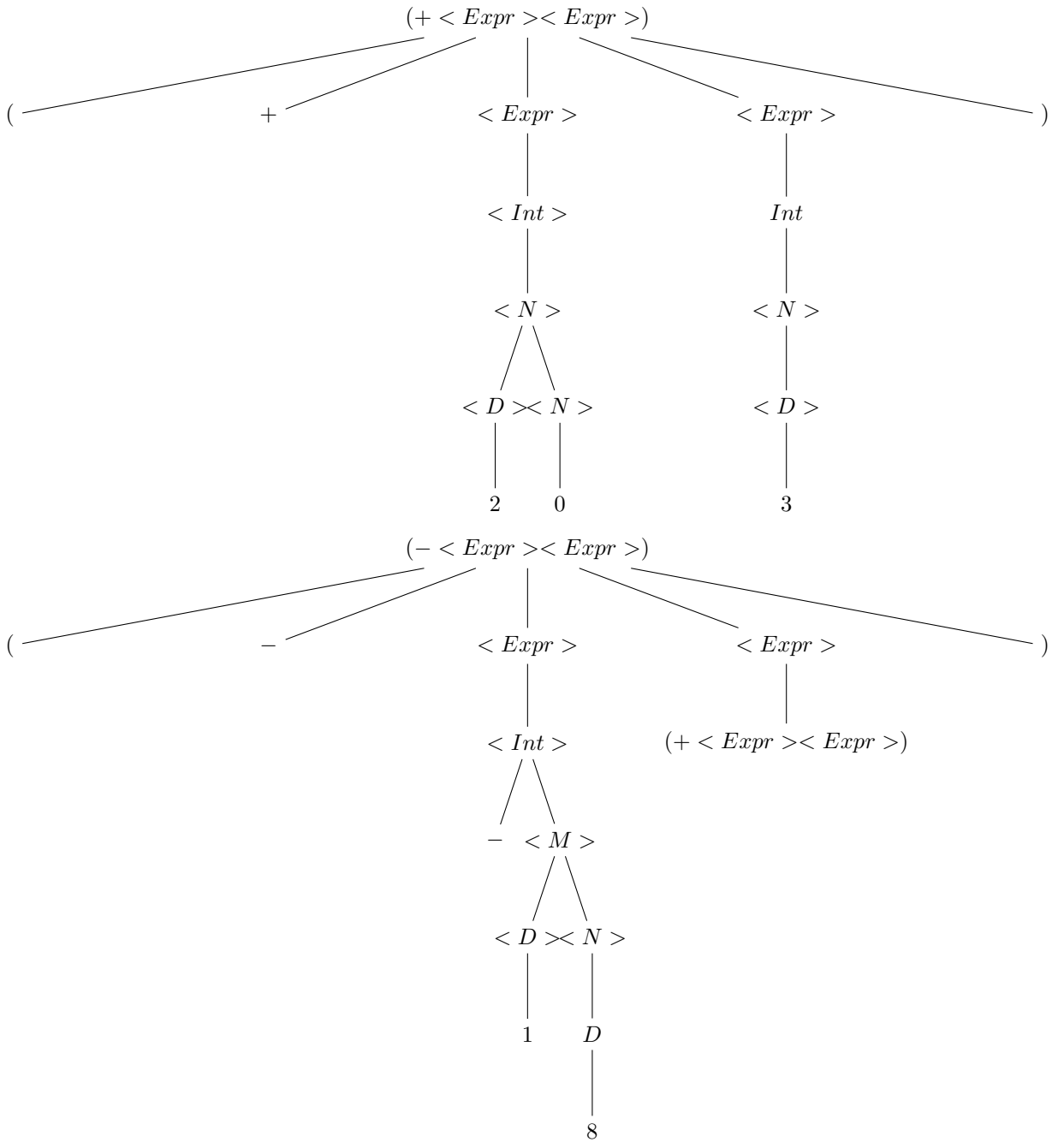
Ejercicios

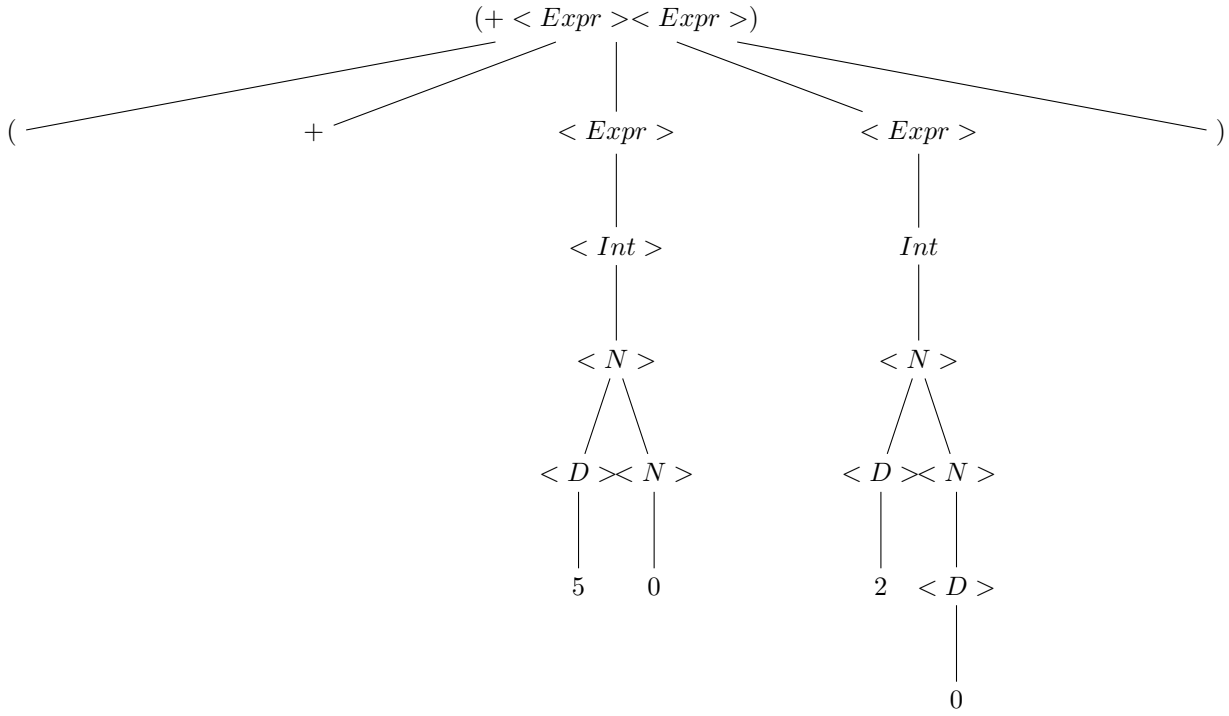
1 Dadas las siguientes expresiones en sintaxis concreta de nuestro lenguaje **MiniLisp** (a) obtener su sintaxis abstracta, (b) evaluarlas usando las reglas de semántica natural y (c) evaluarlas usando las reglas de semántica estructural. Todas las reglas las podrán consultar en la [Nota de Clase 6] (https://lambdasspace.github.io/LDP/notas/ldpn06.pdf) y la [Nota de Clase 7] (https://lambdasspace.github.io/LDP/notas/ldpn07.pdf).

- ‘$(-(+20\ 3)(--18(+50\ 20)))$‘
- ‘$(not(+1(-3(+-8\ 1))))$‘
- ‘$(not(not(+3\ 5)))$‘

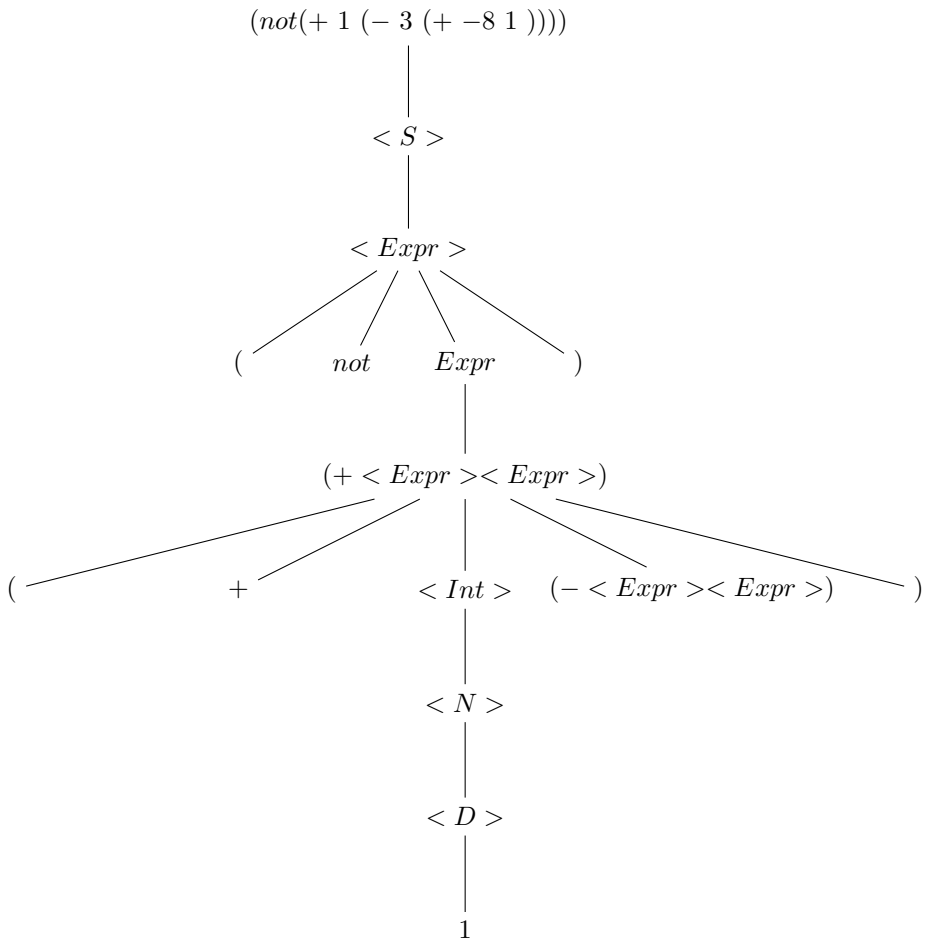
1.-$(-(+\ 20\ 3\ )(-\ -18\ (+\ 50\ 20\ )))$

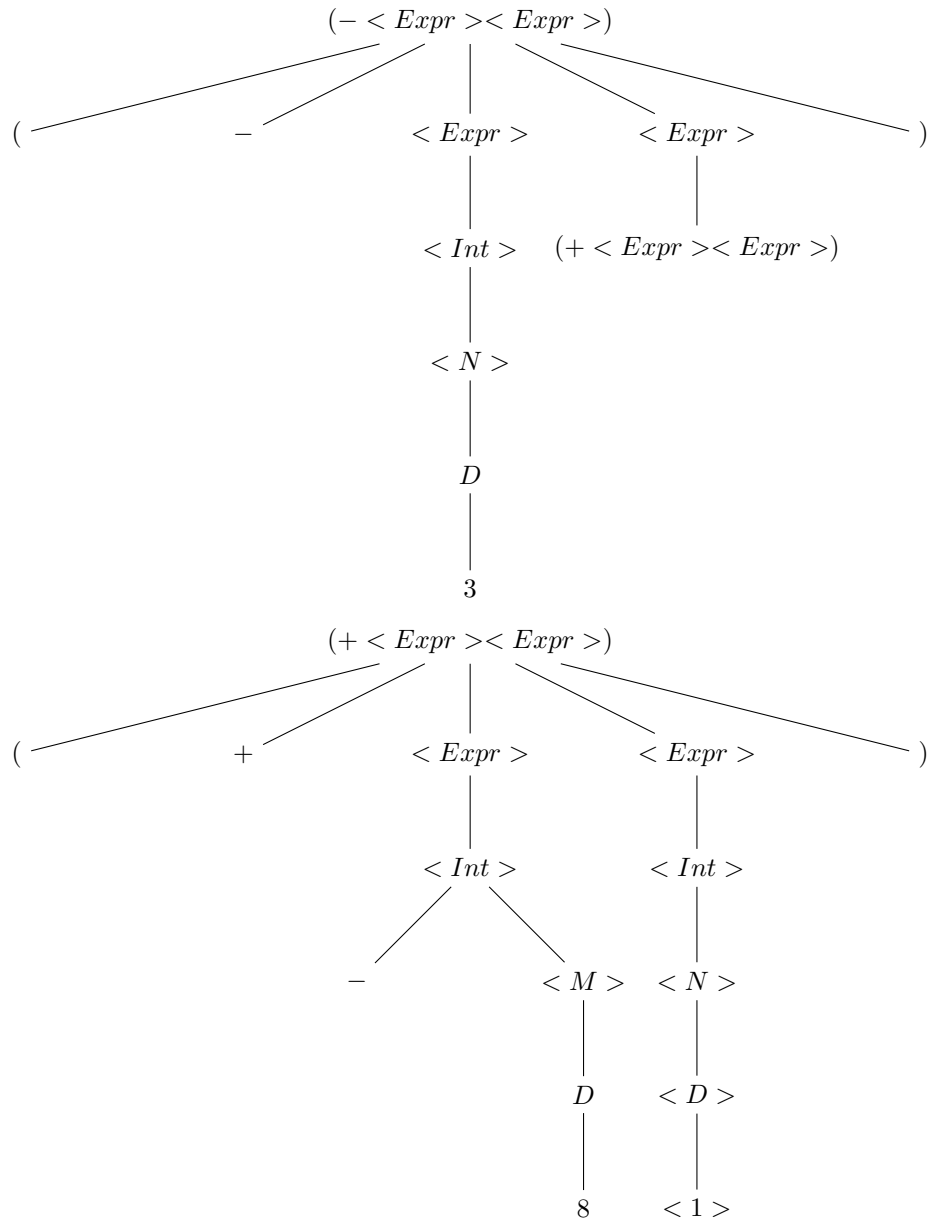



(+ < Expr >< Expr >)
(
+
< Expr >
< Expr >
)
< Int >
Int
< N >
< N >
< D >< N >
< D >
2
0
3
(− < Expr >< Expr >)
(
−
< Expr >
< Expr >
)
< Int >
(+ < Expr >< Expr >)
− < M >
< D >< N >
1
D
8

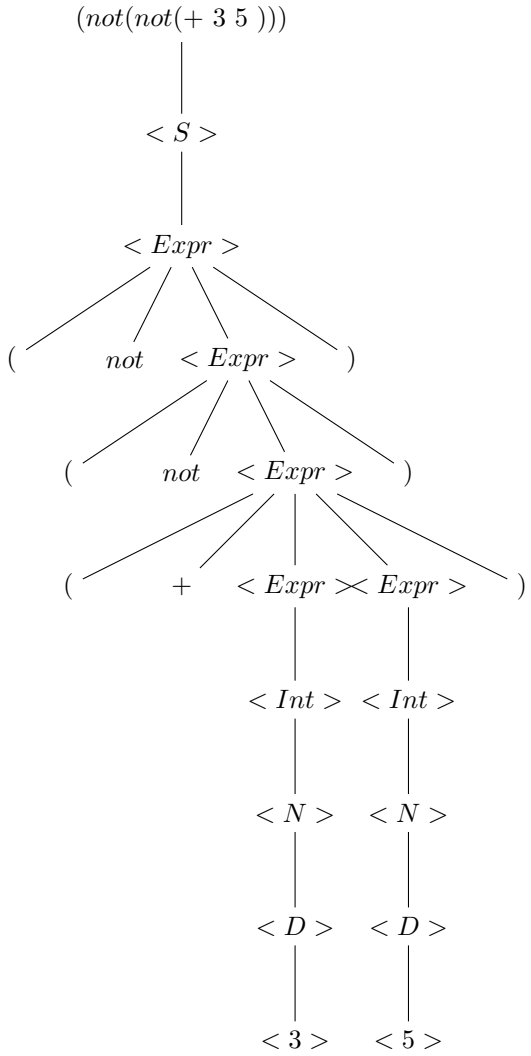

(+ < Expr >< Expr >)
(
+
< Expr >
< Expr >
)
< Int >
Int
< N >
< N >
< D >< N >
< D >< N >
5
0
2
< D >
0

2.-(*not*(+ 1 (− 3 (+ −8 1))))


(not(+ 1 (− 3 (+ −8 1))))
< S >
< Expr >
(
not
Expr
)
(+ < Expr >< Expr >)
(
+
< Int >
(− < Expr >< Expr >)
)
< N >
< D >
1


(− < Expr >< Expr >)
(
−
< Expr >
< Expr >
)
< Int >
(+ < Expr >< Expr >)
< N >
D
3
(+ < Expr >< Expr >)
(
+
< Expr >
< Expr >
)
< Int >
< Int >
−
< M >
< N >
D
< D >
8
< 1 >

3.-(*not*(*not*(+ 3 5)))



2 Como segundo ejercicio deberán extener la batería de operaciones de **MiniLisp**, para ello deberán (a) dar la gramática libre de contexto modificada (en notación EBNF) añadiendo las nuevas construcciones del lenguaje, (b) modificar las reglas de sintaxis abstracta para considerar los nuevos constructores y finalmente (c) extender las reglas de semántica natural y estructural. En los tres casos, deberás usar la notación formal que vimos en clase.

(a) Gramática Libre de Contexto (EBNF)

<S> ::= <Expr>

<Expr> ::= <Int>
| <Bool>
| (+ <Expr> <Expr>)
| (- <Expr> <Expr>)
| (* <Expr> <Expr>)
| (/ <Expr> <Expr>)
| (add1 <Expr>)
| (sub1 <Expr>)
| (sqrt <Expr>)
| (not <Expr>)

<Int> ::= <N>
| -<M>

<D> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<N> ::= 0 | <D>{<N>}

<M> ::= <D>{<N>}

<Bool> ::= # t| # f

(b)

Regla para mul:

$$\frac{e_1 \quad \text{ASA} \qquad e_2 \quad \text{ASA}}{\text{Mul}(e_1, e_2) \quad \text{ASA}}$$

Regla para add1:

$$\frac{e \quad \text{ASA}}{\text{Add1 ASA}}$$

Regla para div:

$$\frac{e_1 \quad \text{ASA} \qquad e_2 \quad \text{ASA}}{\text{Div}(e_1, e_2) \quad \text{ASA}}$$

Regla para sub1:

$$\frac{e_1 \quad \text{ASA} \qquad e_2 \quad \text{ASA}}{\text{Sub1}(e_1, e_2) \quad \text{ASA}}$$

Regla para sqrt:

$$\frac{e_1 \quad \text{ASA} \qquad e_2 \quad \text{ASA}}{\text{Sqrt}(e_1, e_2) \quad \text{ASA}}$$

(c)

- Especificar un nuevo constructor '*' para la multiplicación binaria de expresiones aritméticas. Por ejemplo:

"'lisp > (* 20 2) 40 "'

$$\frac{e_1 \Rightarrow \text{Num}(n_1) \quad e_2 \Rightarrow \text{Num}(n_2)}{\text{Mul}(e_1, e_2) \Rightarrow \text{Num}(n_1 * n_2)}$$

Mul(Num(20) Num(2)) = Num (20 * 2) = Num(40)

- Especificar un nuevo constructor '/' para la división binaria de expresiones aritméticas. Consideren que no se pueden realizar divisiones entre cero. Por ejemplo:

"'lisp > (/ 20 2) 10 > (/ 10 0) error: División entre cero "'

$$\frac{e_1 \Rightarrow \text{Num}(n_1) \quad e_2 \Rightarrow \text{Num}(n_2) \quad n_2 \neq 0}{\text{Div}(e_1, e_2) \Rightarrow \text{Num}(n_1/n_2)}$$

(Div (Num 20) (Num 2)) = Num 10

- Especificar un nuevo constructor 'add1' que dada una expresión, incrementa en uno su valor. Por ejemplo:

"'lisp > (add1 10) 11 "'

$$\frac{e \Rightarrow \text{Num}(n)}{\text{Add1}(e) \Rightarrow \text{Num}(n+1)}$$

(Add1 (Num 10)) = Num 11

- Especificar un nuevo constructor ‘sub1‘ que dada una expresión, decrementa en uno su valor. Por ejemplo:

“‘lisp > (sub1 10) 9 “‘

Semantica Natural:

$$\frac{i \Rightarrow \text{Num}(d)}{\text{Sub1}(i) \Rightarrow \text{Num}(d-1)}$$

Semantica Estructural:

$$\frac{i \longrightarrow d}{\text{Sub1}(i) \longrightarrow \text{Sub1}(d)}$$

$$\frac{}{\text{Sub1}(Num(i)) \longrightarrow \text{Num}(i-1)}$$

- Especificar un nuevo constructor ‘sqrt‘ que dada una expresión, obtiene la raíz cuadrada de dicha expresión. Consideren que no se pueden calcular raíces cuadradas de números negativos. Por ejemplo:

“‘lisp > (sqrt 81) 9 > (sqrt -2) error: Raíz negativa “‘

Semantica Natural:

$$\frac{i \Rightarrow \text{Num}(d)}{\text{Sqrt}(i) \Rightarrow \text{Num}(\sqrt{d})}$$

Semantica Estructural:

$$\frac{i \longrightarrow d}{\text{Sqrt}(i) \longrightarrow \text{Sqrt}(d)}$$

$$\frac{i \geq 0}{\text{Sqrt}(Num(i)) \longrightarrow \text{Num}(\sqrt{i})}$$

$$\frac{i < 0}{\text{Sqrt}(Num(i)) \longrightarrow \text{Error: Raiz negativa}}$$