

Evaluación semanal 3

Santiago González Tamariz

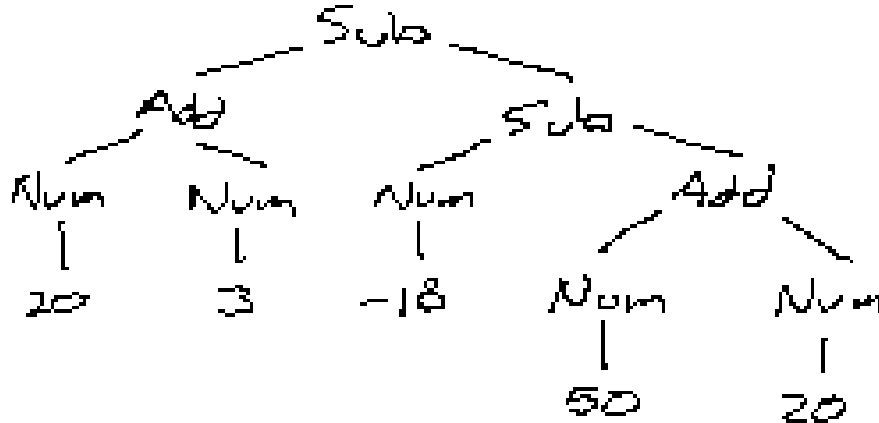
Lenguajes de Programación

1. Dadas las siguientes expresiones en sintaxis concreta de nuestro lenguaje **MiniLisp**

- Obtener su sintaxis abstracta
- Evaluarlas usando las reglas de semántica natural
- Evaluarlas usando las reglas de semántica estructural

(a) `(- (+ 20 3) (- -18 (+ 50 20)))`

Sintaxis abstracta



`Sub(Add(Num(20) Num(3)) Sub(Num(-18) Add(Num(50) Num(20))))`

Semántica natural

$$\begin{array}{c}
 \frac{\text{Num}(20) \Rightarrow \text{Num}(20) \quad \text{Num}(3) \Rightarrow \text{Num}(3)}{\text{Add}(\text{Num}(20) \text{ Num}(3)) \Rightarrow \text{Num}(23)} \quad \frac{\text{Num}(-18) \Rightarrow \text{Num}(-18) \quad \frac{\text{Num}(50) \Rightarrow \text{Num}(50) \quad \text{Num}(20) \Rightarrow \text{Num}(20)}{\text{Add}(\text{Num}(50) \text{ Num}(20)) \Rightarrow \text{Num}(70)}}{\text{Sub}(\text{Num}(-18) \text{ Add}(\text{Num}(50) \text{ Num}(20))) \Rightarrow \text{Num}(-88)} \\
 \hline
 \text{Sub}(\text{Add}(\text{Num}(20) \text{ Num}(3)) \text{ Sub}(\text{Num}(-18) \text{ Add}(\text{Num}(50) \text{ Num}(20)))) \Rightarrow \text{Num}(111)
 \end{array}$$

(b) `(not (+ 1 (- 3 (+ -8 1))))`

(c) `(not (not (+ 3 5)))`

2. Extender la batería de operaciones de **MiniLisp**

En los tres casos, deberás usar la notación formal que vimos en clase

- Dar la gramática libre de contexto modificada (en notación EBNF) añadiendo las nuevas construcciones del lenguaje
- Modificar las reglas de sintaxis abstracta para considerar los nuevos constructores
- Extender las reglas de semántica natural y estructural

(a) Especificar un nuevo constructor `*` para la multiplicación binaria de expresiones aritméticas. Por ejemplo

`> (* 20 2)`

40

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

`> (/ 20 2)`

10

`> (/ 10 0)`

error: División entre cero

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

```
> (add1 10)
11
```

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

```
> (sub1 10)
9
```

- (e) 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:

```
> (sqrt 81)
9
> (sqrt -2)
error: Raíz negativa
```