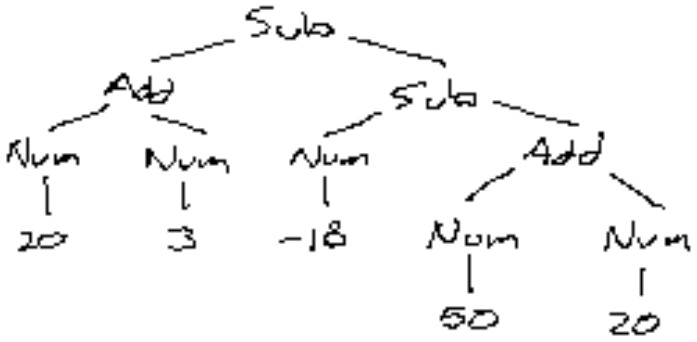
Evaluación semanal 3

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

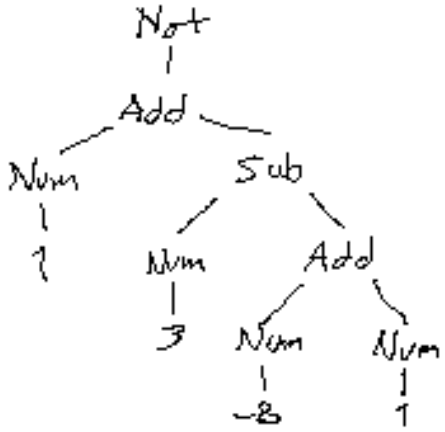
$$\dfrac{\dfrac{\dfrac{}{\texttt{Num(20)} \Rightarrow \texttt{Num(20)}} \quad \dfrac{}{\texttt{Num(3)} \Rightarrow \texttt{Num(3)}}}{\texttt{Add(Num(20) Num(3))} \Rightarrow \texttt{Num(23)}} \quad \dfrac{\dfrac{}{\texttt{Num(-18)} \Rightarrow \texttt{Num(-18)}} \quad \dfrac{\dfrac{}{\texttt{Num(50)} \Rightarrow \texttt{Num(50)}} \quad \dfrac{}{\texttt{Num(20)} \Rightarrow \texttt{Num(20)}}}{\texttt{Add(Num(50) Num(20))} \Rightarrow \texttt{Num(70)}}}{\texttt{Sub(Num(-18) Add(Num(50) Num(20)))} \Rightarrow \texttt{Num(-88)}}}{\texttt{Sub(Add(Num(20) Num(3)) Sub(Num(-18) Add(Num(50) Num(20))))} \Rightarrow \texttt{Num(111)}}$$

   Semántica estructural

   `Sub( Add( Num(20) Num(3) ) Sub( Num(-18) Add( Num(50) Num(20) ) ) )` $\rightarrow$
   `Sub( Num(23) Sub( Num(-18) Add( Num(50) Num(20) ) ) )` $\rightarrow$
   `Sub( Num(23) Sub( Num(-18) Num(70) ) )` $\rightarrow$
   `Sub( Num(23) Num(-88) )` $\rightarrow$
   `Num(111)` $\rightarrow$ `Num(111)`

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

   Sintaxis abstracta



   `Not( Add( Num(1) Sub( Num(3) Add( Num(-8) Num(1) ) ) ) )`

Semántica natural

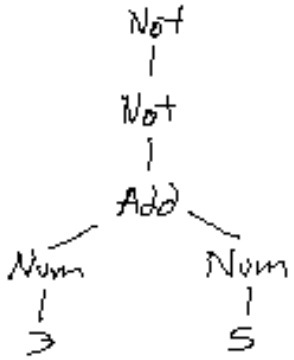
$$\dfrac{\dfrac{\dfrac{}{\texttt{Num(1)} \Rightarrow \texttt{Num(1)}} \quad \dfrac{\dfrac{\dfrac{}{\texttt{Num(3)} \Rightarrow \texttt{Num(3)}} \quad \dfrac{\dfrac{}{\texttt{Num(-8)} \Rightarrow \texttt{Num(-8)}} \quad \dfrac{}{\texttt{Num(1)} \Rightarrow \texttt{Num(1)}}}{\texttt{Add(Num(-8) Num(1))} \Rightarrow \texttt{Num(-7)}}}{\texttt{Sub(Num(3) Add(Num(-8) Num(1)))} \Rightarrow \texttt{Num(10)}}}{\texttt{Add(Num(1) Sub(Num(3) Add(Num(-8) Num(1))))} \Rightarrow \texttt{Num(11)}}}{\texttt{Not(Add(Num(1) Sub(Num(3) Add(Num(-8) Num(1)))))} \Rightarrow \texttt{Bool(False)}}$$

Semántica estructural

```
Not( Add( Num(1) Sub( Num(3) Add( Num(-8) Num(1) ) ) ) ) →
Not( Add( Num(1) Sub( Num(3) Num(-7) ) ) ) →
Not( Add( Num(1) Num(10) ) ) →
Not( Num(11) ) →
Bool(True) → Bool(True)
```

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

Sintaxis abstracta




```
Not( Not( Add( Num(3) Num(5) ) ) )
```

Semántica natural

$$\dfrac{\dfrac{\dfrac{}{\texttt{Num(3)} \Rightarrow \texttt{Num(3)}} \quad \dfrac{}{\texttt{Num(5)} \Rightarrow \texttt{Num(5)}}}{\texttt{Add(Num(3) Num(5))} \Rightarrow \texttt{Num(8)}}}{\texttt{Not(Add(Num(3) Num(5)))} \Rightarrow \texttt{Bool(False)}}}{\texttt{Not(Not(Add(Num(3) Num(5))))} \Rightarrow \texttt{Bool(True)}}$$

Semántica estructural

```
Not( Not( Add( Num(3) Num(5) ) ) ) →
Not( Not( Num(8) ) ) →
Not( Bool(False) ) →
Bool(True) → Bool(True)
```

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
```

Nueva gramática

$$
\begin{aligned}
\langle Expr\rangle ::= & \langle Bool\rangle \mid \langle Num\rangle \\
& \mid (+\ \langle Expr\rangle\ \ \langle Expr\rangle) \\
& \mid (-\ \langle Expr\rangle\ \ \langle Expr\rangle) \\
& \mid (*\ \langle Expr\rangle\ \ \langle Expr\rangle) \\
& \mid (/\ \langle Expr\rangle\ \ \langle Expr\rangle) \\
& \mid (add1\ \langle Expr\rangle) \\
& \mid (sub1\ \langle Expr\rangle) \\
& \mid (sqrt\ \langle Expr\rangle) \\
\langle Bool\rangle ::= & \#t \mid \#f \\
\langle Int\rangle ::= & \langle N\rangle \mid\ -\langle M\rangle \\
\langle D\rangle ::= & 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 7 \mid 8 \mid 9 \\
\langle N\rangle ::= & 0 \mid \langle D\rangle\{\langle N\rangle\} \\
\langle M\rangle ::= & \langle D\rangle\{\langle N\rangle\}
\end{aligned}
$$

Nueva sintaxis abstracta
Añadimos lo siguiente a nuestra sintaxis abstracta

$$\frac{\texttt{i:ASA} \qquad \texttt{d:ASA}}{\texttt{Mult(i, d):ASA}}$$

$$\frac{\texttt{i:ASA} \qquad \texttt{d:ASA}}{\texttt{Div(i, d):ASA}}$$

$$\frac{\texttt{c:ASA}}{\texttt{Add1(c):ASA}}$$

$$\frac{\texttt{c:ASA}}{\texttt{Sub1(c):ASA}}$$

$$\frac{\texttt{c:ASA}}{\texttt{Sqrt(c):ASA}}$$

Nueva semántica natural
Añadimos lo siguiente a nuestra semántica natural

$$\frac{\texttt{i} \Rightarrow \texttt{Num(i')} \qquad \texttt{d} \Rightarrow \texttt{Num(d')}}{\texttt{Mult(i, d)} \Rightarrow \texttt{Num(i' * d')}}$$

$$\frac{\texttt{i} \Rightarrow \texttt{Num(i')} \qquad \texttt{d} \Rightarrow \texttt{Num(0)}}{\texttt{Div(i, d)} \Rightarrow \texttt{error: División entre 0}}$$

$$\frac{\texttt{i} \Rightarrow \texttt{Num(i')} \qquad \texttt{d} \Rightarrow \texttt{Num(d')}}{\texttt{Div(i, d)} \Rightarrow \texttt{Num(i' / d')}}$$

$$\frac{\texttt{c} \Rightarrow \texttt{Num(c')}}{\texttt{Add1(c)} \Rightarrow \texttt{Num(c' + 1)}}$$

$$\frac{\texttt{c} \Rightarrow \texttt{Num(c')}}{\texttt{Sub1(c)} \Rightarrow \texttt{Num(c' - 1)}}$$

$$\frac{\texttt{c} \Rightarrow \texttt{Num(c')} \qquad \texttt{c' < 0}}{\texttt{Sqrt(c)} \Rightarrow \texttt{error: Raíz negativa}}$$

$$\frac{\texttt{c} \Rightarrow \texttt{Num(c')}}{\texttt{Sqrt(c)} \Rightarrow \texttt{Num}(\sqrt{\texttt{c'}})}$$

Nueva semántica estructural
Añadimos lo siguiente a nuestra semántica estructural

$$\frac{\texttt{i} \to \texttt{i'}}{\texttt{Mult(i, d)} \to \texttt{Mult(i', d)}}$$

$$\frac{\texttt{d} \to \texttt{d'}}{\texttt{Mult(Num(i), d)} \to \texttt{Mult(Num(i), d')}}$$

$$\frac{-}{\texttt{Mult(Num(i), Num(d))} \to \texttt{Num(i * d)}}$$

$$\frac{\texttt{i} \to \texttt{i'}}{\texttt{Div(i, d)} \to \texttt{Div(i', d)}}$$

$$\frac{\texttt{d} \to \texttt{d'}}{\texttt{Div(Num(i), d)} \to \texttt{Div(Num(i), d')}}$$

$$\frac{-}{\texttt{Div(Num(i), Num(0))} \to \texttt{error: División entre 0}}$$

$$\frac{-}{\texttt{Div(Num(i), Num(d))} \to \texttt{Num(i / d)}}$$

$$\frac{\texttt{c} \to \texttt{c'}}{\texttt{Add1(c)} \to \texttt{Add1(c')}}$$

$$\frac{-}{\texttt{Add1(Num(c))} \to \texttt{Num(c + 1)}}$$

$$\frac{\texttt{c} \to \texttt{c'}}{\texttt{Sub1(c)} \to \texttt{Sub1(c')}}$$

$$\frac{-}{\texttt{Sub1(Num(c))} \to \texttt{Num(c - 1)}}$$

$$\frac{\texttt{c} \to \texttt{c'}}{\texttt{Sqrt(c)} \to \texttt{Sqrt(c')}}$$

$$\frac{\texttt{c < 0}}{\texttt{Sqrt(Num(c))} \to \texttt{error: Raíz negativa}}$$

$$\frac{-}{\texttt{Sqrt(Num(c))} \to \texttt{Num}(\sqrt{\texttt{c}})}$$