Datos: Especificación de forma recursiva

$X \mid T$

Inductiva

$2 \in S$

$$\frac{n \in S}{n+2 \in S}$$

$2, 4, 6, 8, \ldots \infty$

$8 \to 6 \to 4 \to (2)$

par

$0 \in \mathbb{N}$

$$\frac{n \in \mathbb{N}}{n+1 \in \mathbb{N}}$$

$'() \in S$

$$\frac{l \in S, n \in \mathbb{N}}{(n \mid l) \in S}$$

lista

$'() \in S$

$$\frac{s \in S_y, l \in S, r \in S}{(s \mid r)}$$

$(s \mid r)$

$(x\ (y\ ()\ ())\ (x\ (t\ 0\ ())\ (y\ 0\ ())))$

```
(define arbol? (lambda (arb)
    (cond
        [(null? arb) #T]
        [(and
            (symbol? (car arb))
            (arbol? (cadr arb))
            (arbol? (caddr arb)))
        ]))
```

s ← (symbol? (car arb))
l ← (arbol? (cadr arb))
r ← (arbol? (caddr arb)))

```
(define arb->list (lambda (arb)
    (cond
        [(null? arb) '()]
        [(append
            (list (symbol? (car arb)))
            (arb->list (cadr arb))
            (arb->list (caddr arb)))
        ]))
```

s ← (list (symbol? (car arb)))
l ← (arb->list (cadr arb))
r ← (arb->list (caddr arb)))

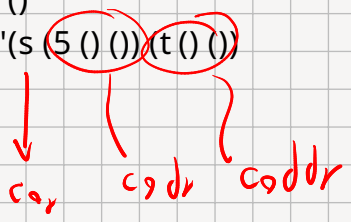
Especifición mediante gramáticas

<list-s> ::= <symbol>*

<list-s> ::= '()
::= <symbol> <list-s>

<arbol-b> ::= '()
::= <symbol> <arbol-b> <arbol-b>
::= <int> <arbol-b> <arbol-b>
::= <list-s> <arbol-b> arbol-b>
::= <symbol> <int>
<arbol-b><arbol-b>

'()
'(s (5 () ()) (t () ())




Alcance y ligadura de variables

- Alcance estatico / Block scope

```
def pepito():
    a = 8
pepito()
print(a)
```

Error

```
if(a<=3){
    int k = 8;
}else{ }
System.out.println(k);
```

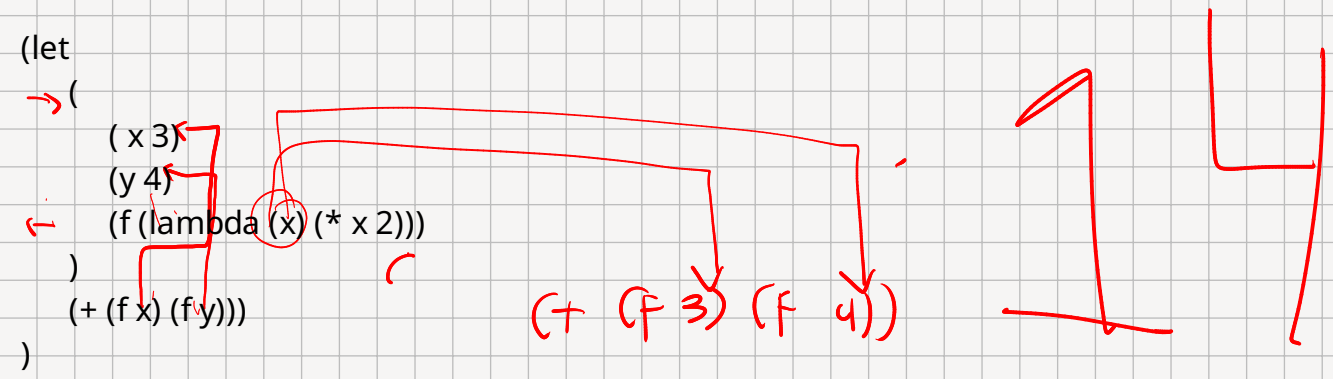
```
function [x,y] perrito(a,b):
        s = 9

perrito(2,3)
print(s)
```

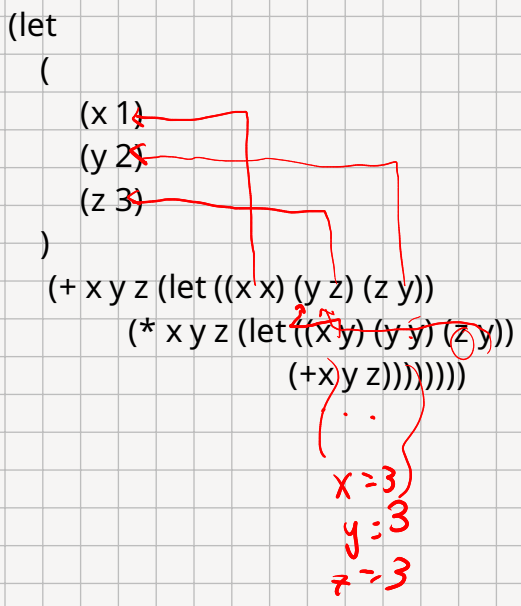
Octave/Matlab

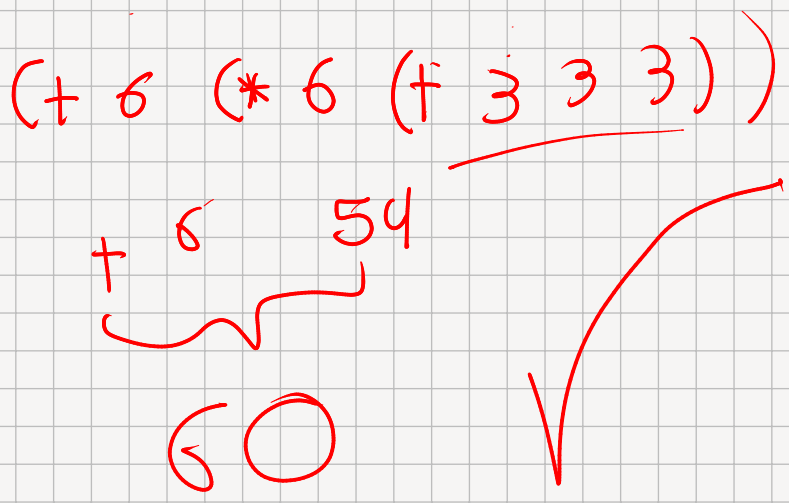
- Alcance dinámico

```
def pepito():
    global a
    a = 3

print(a) //3

def carlitos():
    global a
    a = 4

print(a) //4
```

@a = 9
a = 9






```
(let ((x 6)(y 7))
  (*
   (let ((y 8))
     (+
      (let ((x 6) (y x))
        (+ x
           (let ((y 3) (x y)) (+ x (+ 2 y)))
           )
        )
      y)
     )
   (let ((x 4)) (- y x))
   )
  )
```

Ocurre libre o ocurre ligado

<lc-exp> ::= <symbol>
::= lambda ( <symbol> ) <lc-exp>
::= <lc-exp> <lc-exp>

1
2
3

( (lambda (x) x) x)

Ocurre libre

1) Si el id es igual al simbol

2) Si id del lambda es diferente al simbolo Y ocurre libre en la expresión interna.

3) Si ocurre libre en alguna de las dos