

Dato  $\rightarrow$  Conjunto posibles valores

Integer  $\rightarrow \{-2^{31}, 2^{31}-1\}$

string  $\rightarrow \{ "a" \dots \} \infty$

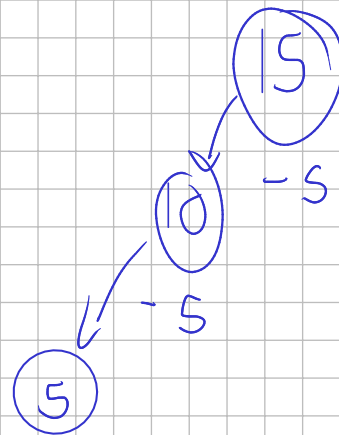
bool  $\rightarrow \{ 0, 1 \}$

Inductive

$5 \in S$

$x \in S$

$x+5 \in S$



$Lrsta \quad num$   
 $para$   
 $( ) \in S$

$2 \in P$

1) Define un elemento base

$l \in S, n \in P$   


---

 $(n \ l) \in S$

$\rightarrow cons$

$x \in P$   


---

 $x+2 \in P$

2) Define una regla recursiva  
+ -

$(i, j, k)$   


---

 $(\underline{i+1}, \underline{j+1}, \underline{k+1})$

# Ligadura y alcance de variables

## Ocurre libre u ocurre ligada

$$\begin{aligned} \langle \text{c-exp} \rangle ::= & (\text{var-exp}) \quad \langle \text{simbolo} \rangle \\ & (\text{lambda-exp}) \quad \text{lambda} (\langle \text{id} \rangle) \quad \langle \text{c-exp} \rangle \\ & (\text{app-exp}) \quad ( \underbrace{\langle \text{c-exp} \rangle}_{\text{rator}} \quad \underbrace{\langle \text{c-exp} \rangle}_{\text{rand}} ) \end{aligned}$$

1. (var-exp) Ocurre libre
2. (lambda-exp) Distinta de  $\langle \text{id} \rangle$
3. (app-exp) (or (ocurre libre en rator)  
(ocurre libre en rand))

(lambda (x)  
 ((x y) (lambda (y) y)))

1. x ¿Ocurre libre o ligada? --> Ligada
2. y ... --> Libre

# Alcance de variables

```
(let  
  (  
    (x 5) (y 6) (p 7)  
  )  
  (let  
    (  
      (x (+ x y)) (y 9) (p (+ y 1))  
    )  
      (+ x y p)  
    )  
  )
```

Handwritten annotations in red:

- A red arrow points from the `p` in the inner `let` block to the `p` in the outer `let` block.
- A red circle highlights the inner `let` block.
- A red circle highlights the expression `(p (+ y 1))`.
- The number `11` is written above `(x (+ x y))`.
- The number `7` is written above `(p (+ y 1))`.
- The number `27` is written below `(+ x y p)`.
- The number `11` is written below `(+ x y p)`.

let  
in

Integer  $\rightarrow$  [ 000 ... 110 ]

32 bits

+ - % \* / ^ 9 bits ...

Implementación

Interfaz

(6) = 16

( \_ \_ \_ )

( )  $\leftarrow$  (1)

( 1 2 3 )

$1 \times 16^0 + 2 \times 16^1 + 3 \times 16^2$

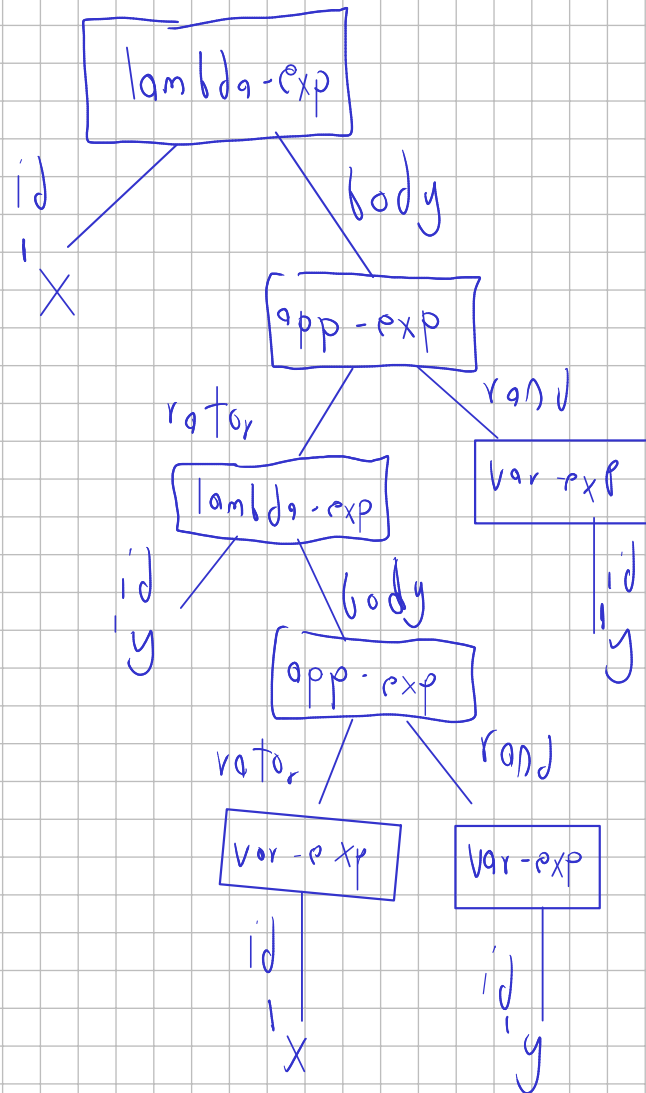
(15)  $\rightarrow$  (0 4)

Lists  $\leftarrow$  '(...)

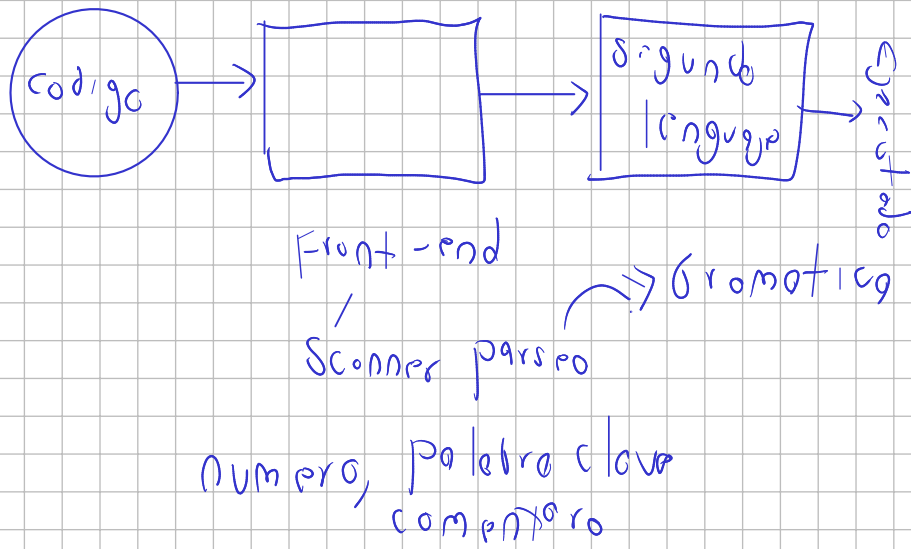
Procedimientos  $\leftarrow$  #

Datatypes  $\rightarrow$  define-datatype

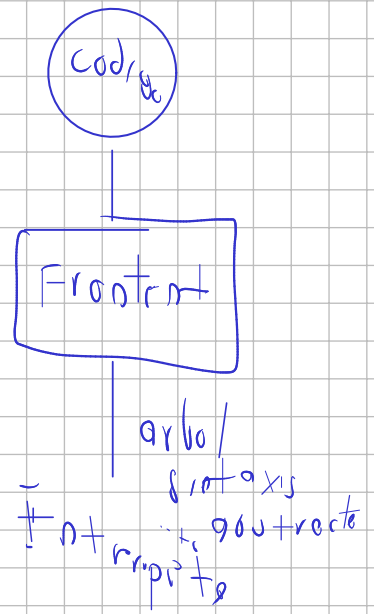
```
(define exp1  
  (lambda-exp 'x (app-exp (lambda-exp 'y (app-exp (var-exp 'x)  
    (var-exp 'y)))  
    (var-exp 'y))))
```



# Compilación



# Interpretación



## Lenguajes interpretados

Especificación **lexica** → unidades significativas

↓  
**Gramática**

**BNF** → **arbol sintaxis abstracto**

**tokens** (tipo, coding, long)

**SLLGEN**