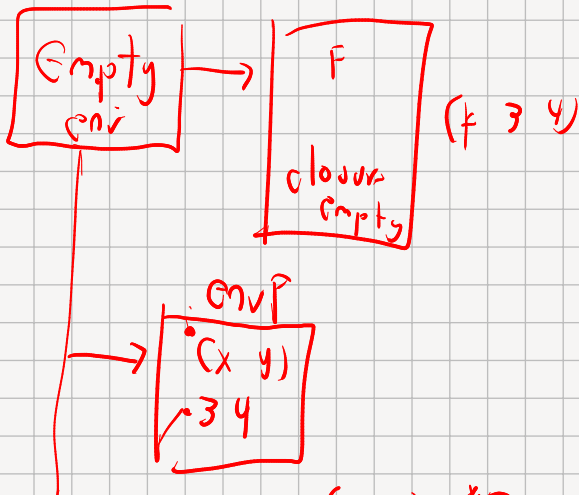


Procedimientos recursivos

let

f = proc(x,y) if >(x,0) then +(y,(f -(x,1) y) else y
else
(f 3 4)

Error



>(x,0) #T
+(4, (f 2))
↓
f, 1/g

letrec

(<identificador> "(" (<identificador>, >)* ")" =
<expresion>)*

in

<expresion>

letrec

f(a,b) = if >(a,b) then +(a, (f -(a,1) b)) else b

g(a,b) = if >(a,b) then *(a,(g -(a,1) b)) else b

in

+((f 2 3) (g 2 3))

- 1) Introducir en la gramática el caso del letrec
- 2) Introducir un nuevo tipo de ambiente ambiente-extendido-recursivo que es un caso ESPECIAL que solo contiene PROCEDIMIENTOS
- 3) Cambiar el apply-env de tal manera que si un proc está dentro de un ambiente extendido debo generar una nueva clausura (va a guardar el ambiente extendido recursivo)

<<

>>

let

$$b = \text{proc}(x,y) + (x,y,z)$$

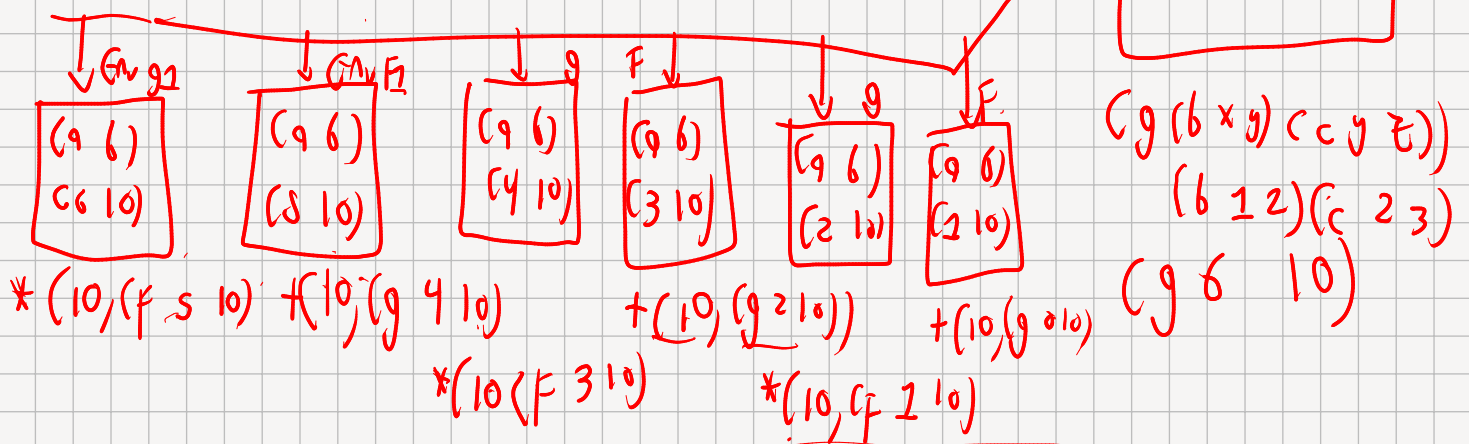
```
c = proc(a,b) let x = +(a,b) in +(x,y,z)
```

in

letrec

$$f(a,b) = \text{if } a > 0 \text{ then } +(b, (g(a,1) b)) \text{ else } b$$
$$g(a,b) = \text{if } a > 0 \text{ then } (b, f(a-1, b)) \text{ else } b$$

in

$$(g(b \times y)(c y z))$$

$$*(10, + (10, * (10, + (10, * (10, + (10, 10))))))$$

21100

Considere el siguiente ambiente inicial

(x,y,z,f,g)

(4,5,6

closure (a,b) let a=+(a,b) b=*(b,a) in +(a,b)

closure (x,y,z) if >(x,+(y,z)) then +(x,y) else *(y,z)

let

a = +((f x y), (g x y z))

b = let k = proc(a,b) +(a,b) in (k x y)

in

letrec

h(x,y) = if >(x,0) then +((f a b), (g -(x,1) y)) else y

g(m,n) = if >(x,0) then +(a, (f -(x,1) n)) else n

in

(h 6 a)