Universidad Nacional Autónoma de México
Facultad de Ciencias

Lenguajes de Programación

Karla Ramírez Pulido

Alcance y Funciones

Nueva implementación

De Sustitución a Ambientes

Mejorando la complejidad del algoritmo de sustitución

```
{with {x   3}
   {with {y   4}
      {with {z   5}
         {+  x  {+  y  z}}}} }
```

```
= {with {y   4}
      {with {z   5}
         {+   3 {+   y   z}}} }
= {with {z   5}
         {+  3  {+   4   z}} }
= {+   3  {+   4   5} }
```

```
{with { x  3 }
 {with { y  4 }
  {with { z  5 }
   {+ x { + y  z}}}}}
```

O(n)

+

```
{with { y  4 }
 {with { z  5 }
  {+ 3 { + y  z}}}}
```

O(n-1)

+

```
{with { z  5 }
 {+ 3 { + 4  z}}}
```

O(n-2)

+

```
{+ 3 { +  4  5}}
```

O(n-3)

Algoritmo de Sustitución

¿Cuántas variables puede haber en un programa?

n variables

¿Cuántas sustituciones habría?

$n + (n-1) + (n-2) + (n-3) + \ldots + (n-m) = O(n^2)$

¿De qué orden es el algoritmo de sustitución?

$O(n^2)$

- **¿Por qué?**
- **¿Se podría mejorar?**

Cambiar la implementación:

AMBIENTES

Ambientes

Ambientes: representación con LISTAS

Ambiente vacío= ( )

Ambiente extendido 1 = ((var_1 val_1))

Ambiente extendido 2 = ((var_2 val_2) (var_1 val_1))

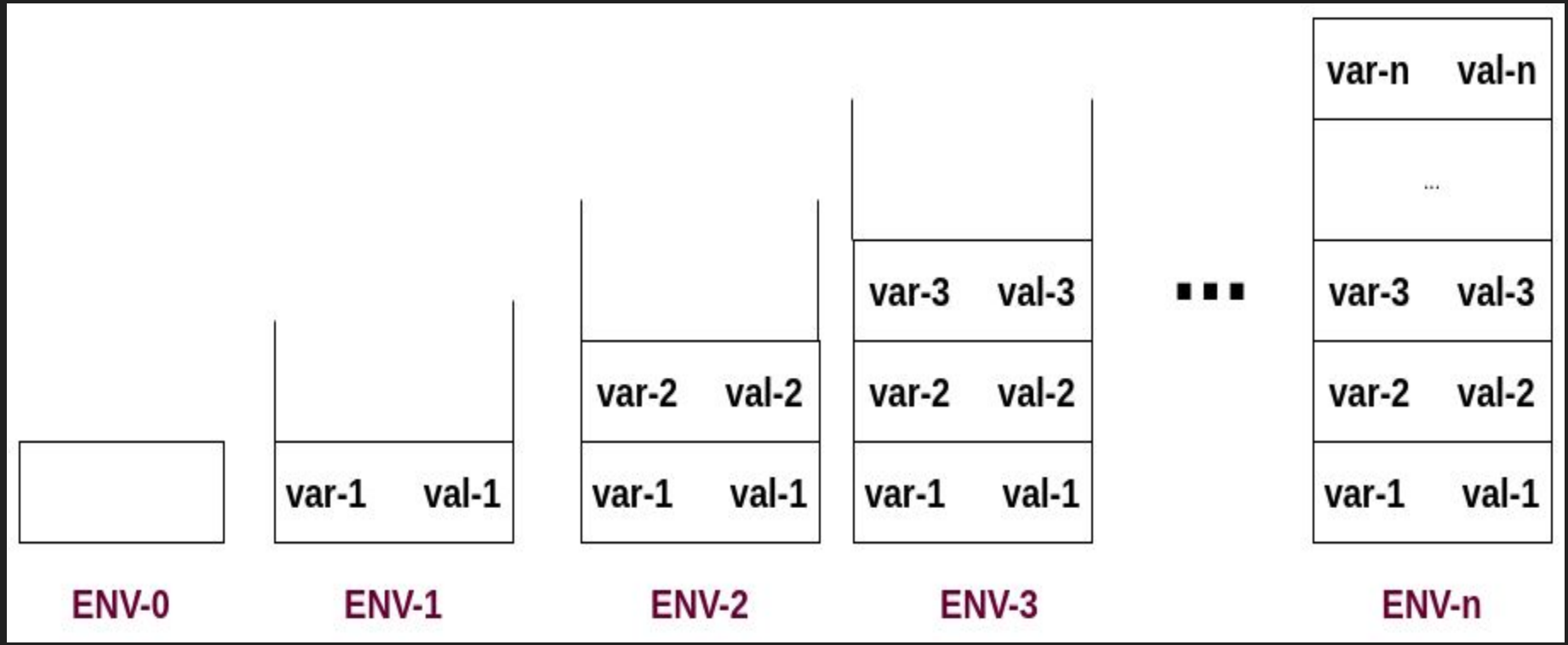
Ambiente extendido 3 = ((var_3 val_3) (var_2 val_2) (var_1 val_1))

…

Ambiente extendido n = ((var_n val_n) … (var_3 val_3) (var_2 val_2)

(var_1 val_1))

Representació de PILA



¿Cuándo vamos a usar el ambiente?

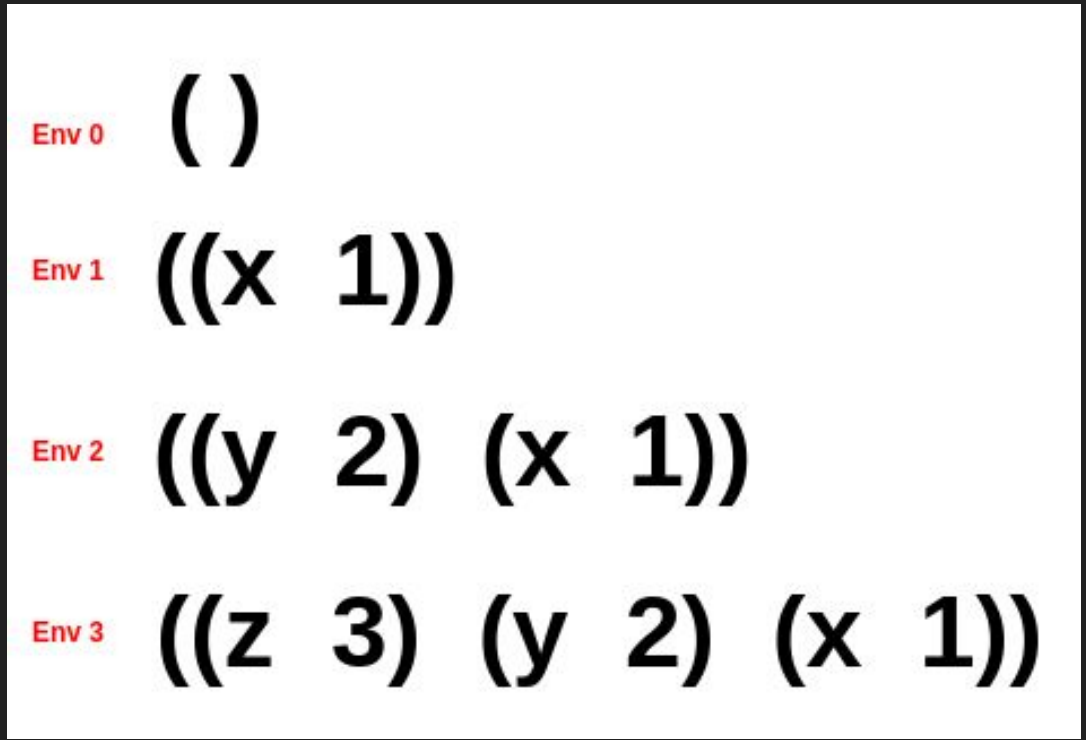
Para ir agregando las asignaciones de las variables de ligado con sus valores.

Ejemplo 1:

```
{with {x  1}
  {with {y  2}
    {with {z  3}
      {+  x  {+  y  z} } } } }
```

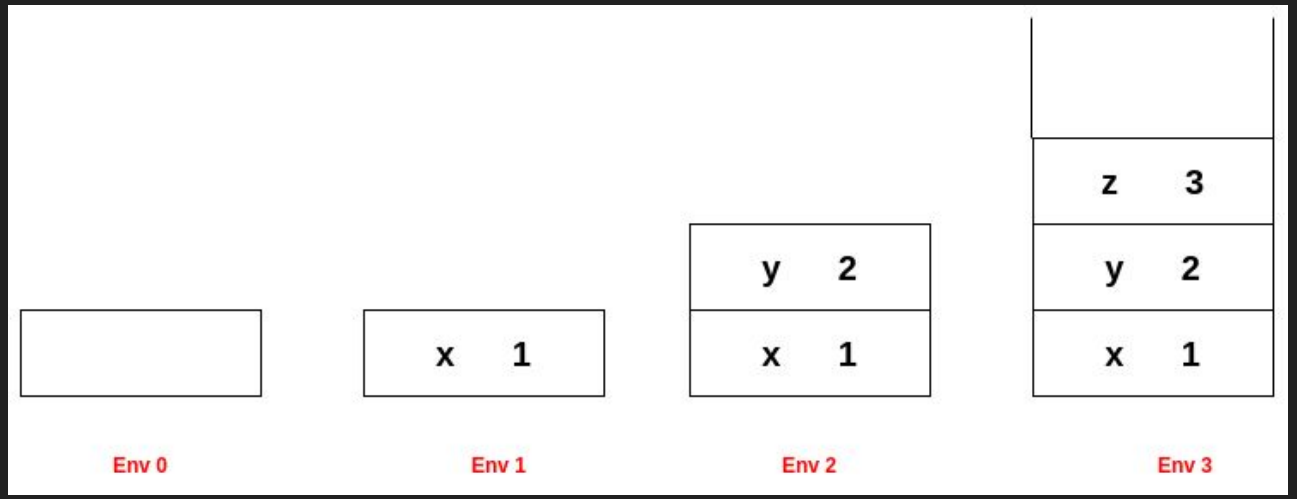
Ejemplo 1. Ambiente con LISTAS

```
{with {x  1}
  {with {y  2}
    {with {z  3}
      {+  x  {+  y  z}}}}}
```



Ejemplo 1. Ambiente con PILAS

```
{with {x  1}
  {with {y  2}
    {with {z  3}
      {+  x  {+  y  z}}}}}
```

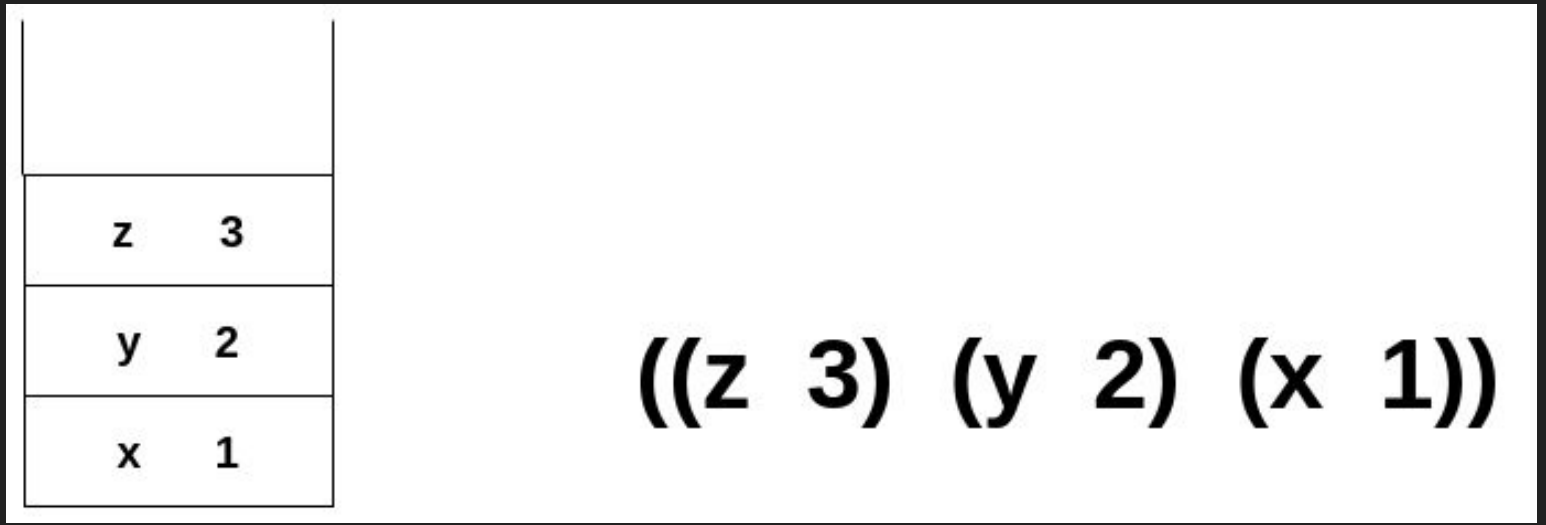


Ambiente final

```
{with {x  1}
  {with {y  2}
    {with {z  3}
      {+  x  {+  y  z} } } } }
```

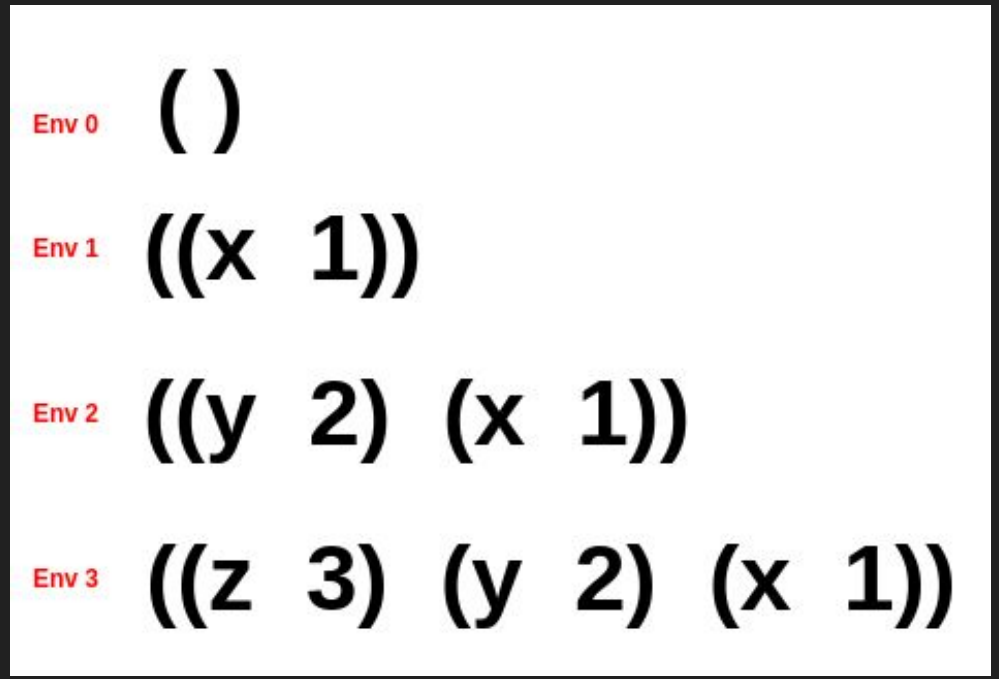
PILA

LISTAS



Ejemplo 2.

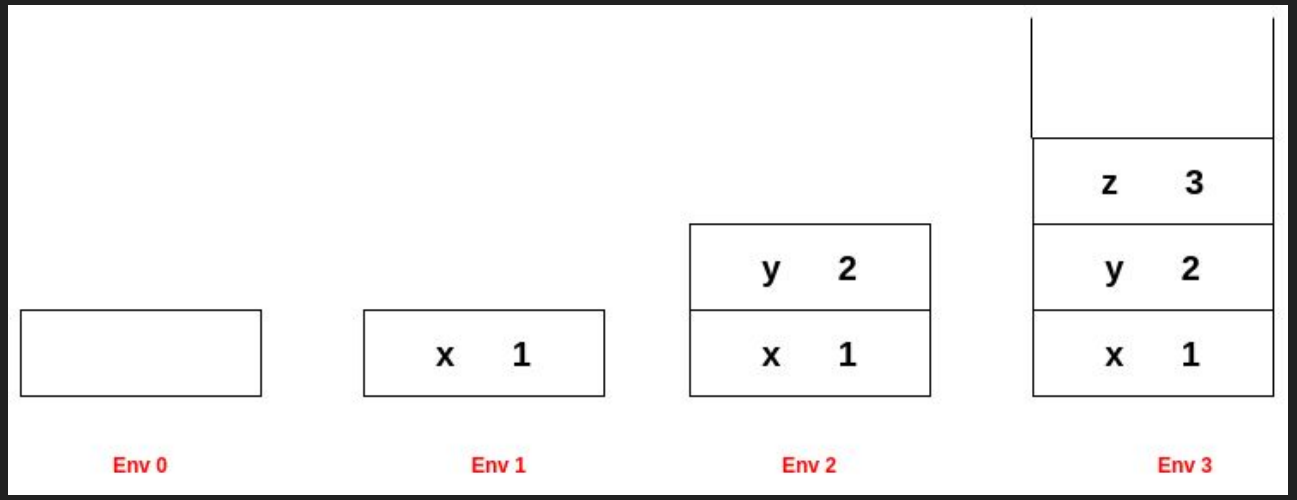
```
{with {x  1}
   {with {y  2}
      {with {z  3}
         {+  z  {+  z  z}} }}}
```



Representación con LISTAS

Ejemplo 2.

```
{with {x  1}
   {with {y  2}
      {with {z  3}
         {+  z  {+  z  z}} }}}
```



Representación con PILAS

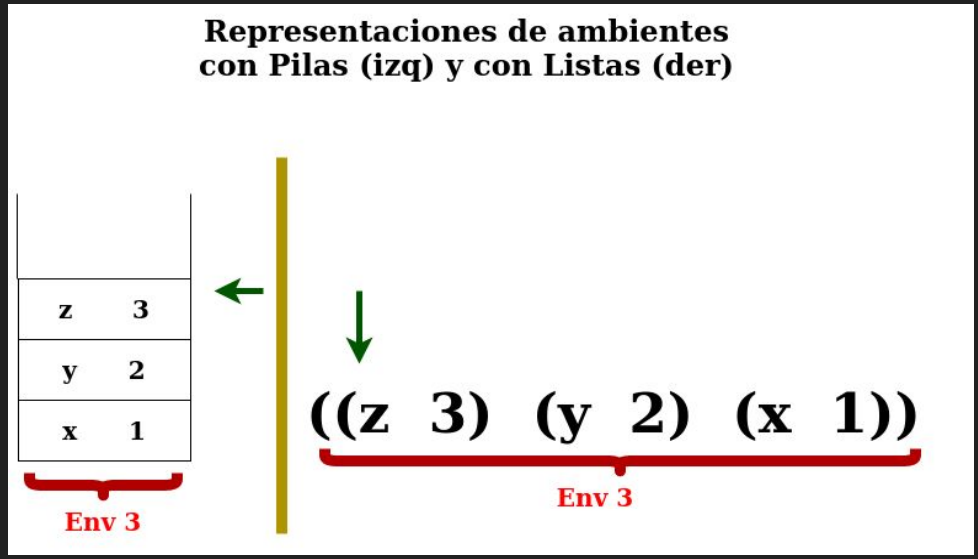
Ambientes

Creamos el ambiente

En la expresión (body-with) a evaluar:

A. Sustituir variables por valores y
B. Evaluar la expresión

```
{+ z {+ z z}}
```



Ambientes

{+ z {+ z z}}

=

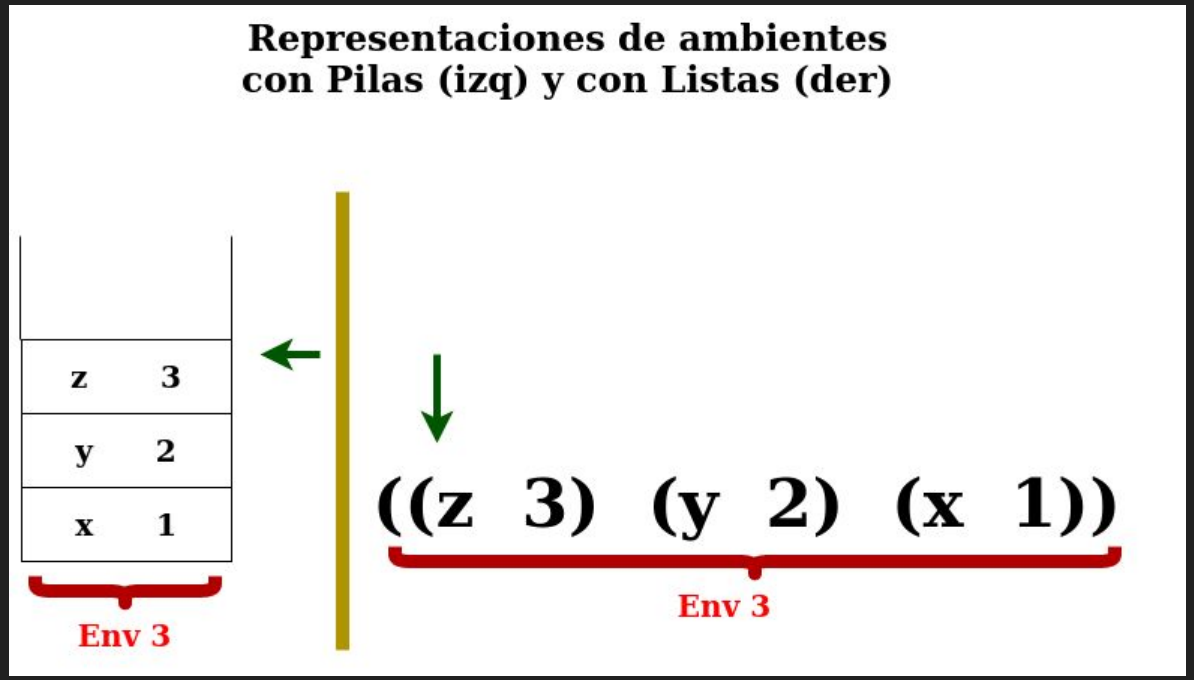
{+ 3 {+ z z}}

=

{+ 3 {+ 3 z}}

=

{+ 3 {+ 3 3}} = {+ 3 6} = 9



En cada una de las variables buscas en el ambiente su nombre, si lo es, sustituyes pero cuando terminas esa búsqueda ahí termina el recorrido de ese algoritmo, e incias otro con la nueva instancia.

Algoritmo de sustitución

```
{with {x  1}
  {with {y  2}
    {with {z  3}
      {+  z  {+  z  z}} }}}
```

[Sustitución x:= 1 en Expr]

Se hace el recorrido en Expr aunque la sustitución es vacía

```
{with {y  2}
  {with {z  3}
    {+  z  {+  z  z}}} }
```

[Sustitución y:= 2 en Expr]

Se hace el recorrido en Expr aunque la sustitución es vacía

Algoritmo de sustitución

{with {z 3}

{+ z {+ z z}} }

[Sustitución z:= 3 en Expr]

{+ z {+ z z}}

= {+ 3 {+ 3 3}}

Haces el recorrido sobre todas las subexpresiones y sustituyes solo en las que la variable ligada sea el mismo id que la var. de ligado.

Ejemplo 3. Ambientes

{with {x 1}

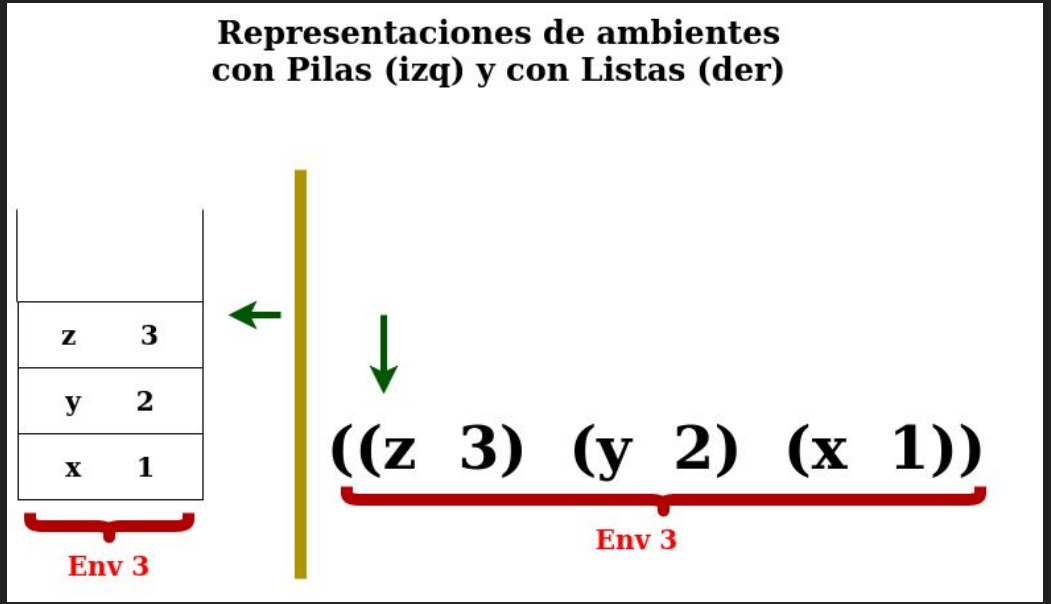
  {with {y 2}

    {with {z 3}

      {+ 7 {+ 5 5}} }}}

= {+ 7 {+ 5 5}}

= { + 7 10 } = 17



Ejemplo 3. Algoritmo de sustitución

{with {x 1}

{with {y 2}

{with {z 3}

{+ 7 {+ 5 5}} }}}

[Sustitución x:= 1 en Expr]

Se hace el recorrido en Expr aunque la sustitución es vacía

{with {y 2}

{with {z 3}

{+ 7 {+ 5 5}} }

[Sustitución y:= 2 en Expr]

Se hace el recorrido en Expr aunque la sustitución es vacía

Algoritmo de sustitución

{with {z 3}

{+ 7 {+ 5 5}} }

[Sustitución z:= 3 en Expr]

{+ 7 {+ 5 5}}

= {+ 7 10} = 17

Haces el recorrido sobre todas las subexpresiones y sustituyes solo en las que la variable ligada sea el mismo id que la var. de ligado.

¿Hubo sustituciones? NO pero el algoritmo si hizo el recorrido sobre todo el código por cada var. de ligado.

ALCANCE

Tipos de alcance:

- Estático
- Dinámico

Representación de ambientes

Alcance Dinámico

```
{with {x   3}
     {with {foo {fun {z}  {+  z   x}}}
          {with {x   7}
               {foo   1} }} }
```

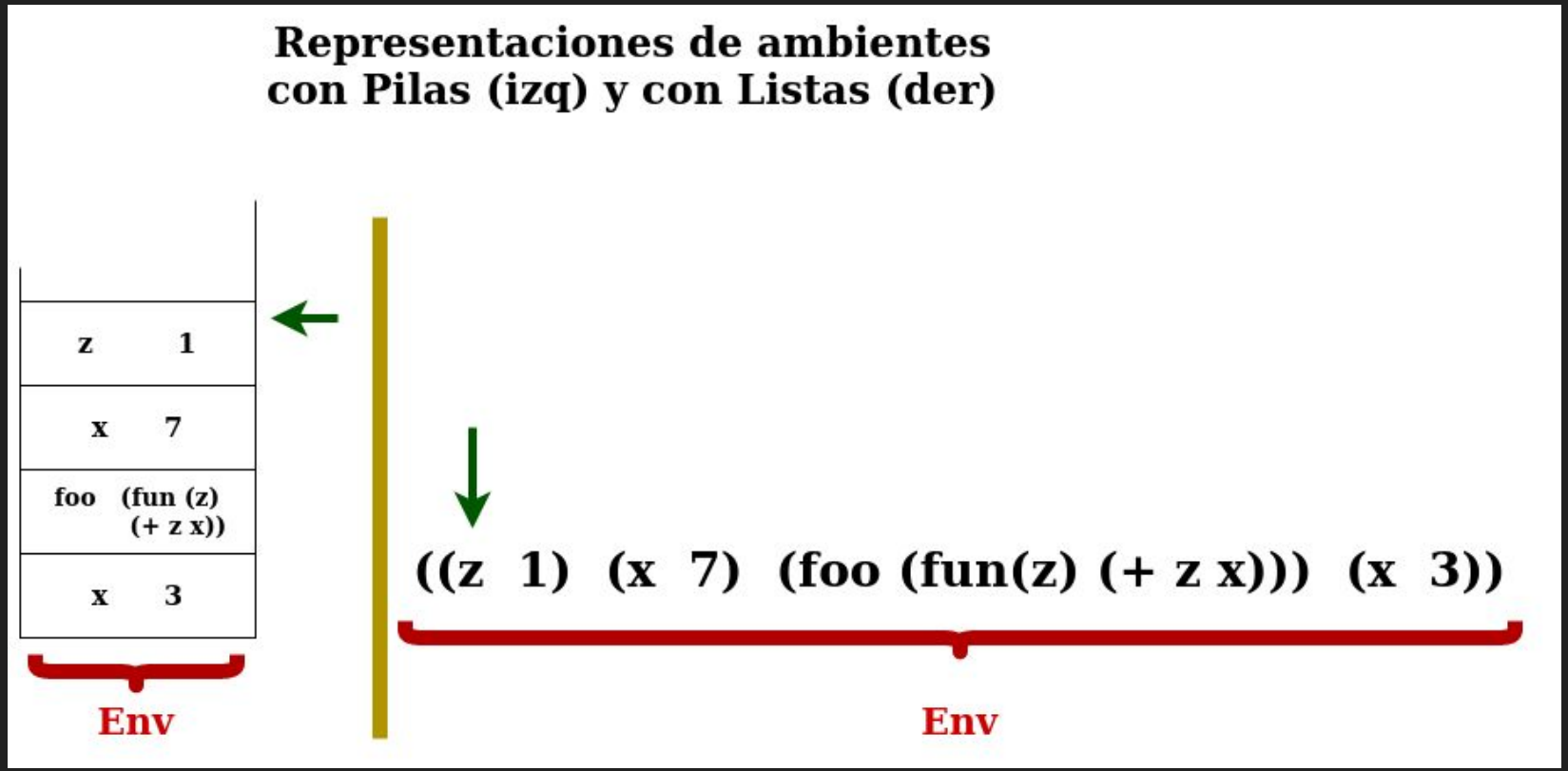
¿Qué tenemos que hacer?

1. Crear el ambiente
2. Evaluar la app de función

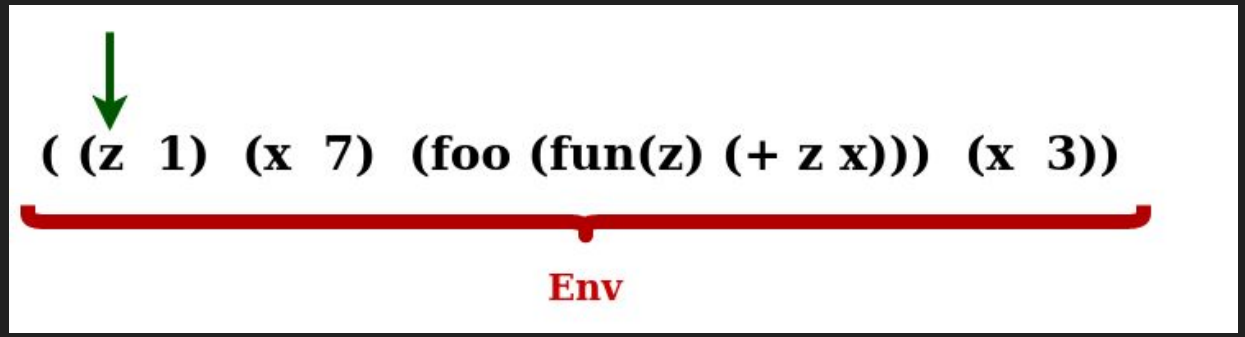
{foo   1}

(A)App: asignar parám. formal el parám. real. (B) sustituirlo en el cuerpo de función y (C) evaluarlo.

Crear el ambiente



Alcance Dinámico



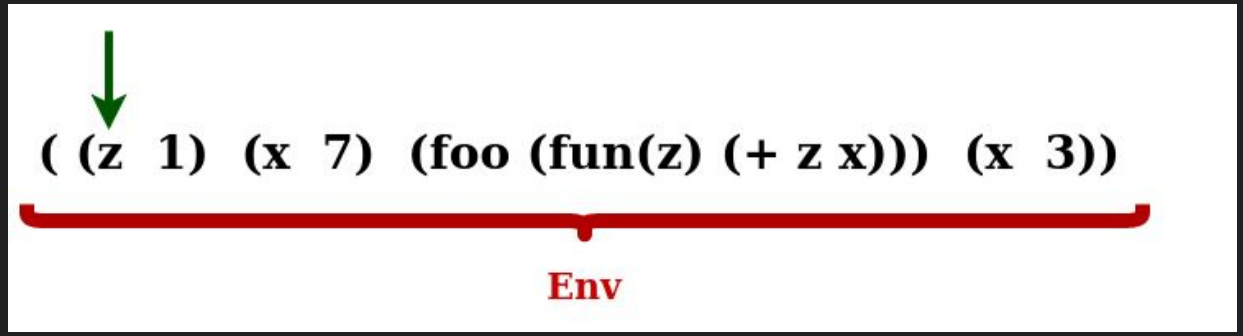
{foo 1}

= { {fun (z) {+ z x}} 1}

Asignamos [z := 1] y evaluamos el cuerpo de la función foo

⇒ {+ 1 x}

Alcance Dinámico



= {+ 1 x}

Busco en la lista o en la pila desde el inicio de éstas, y encuentro el mismo nombre de id. en el ambiente, entonces obtengo su valor y lo sustituyo en la expresión.

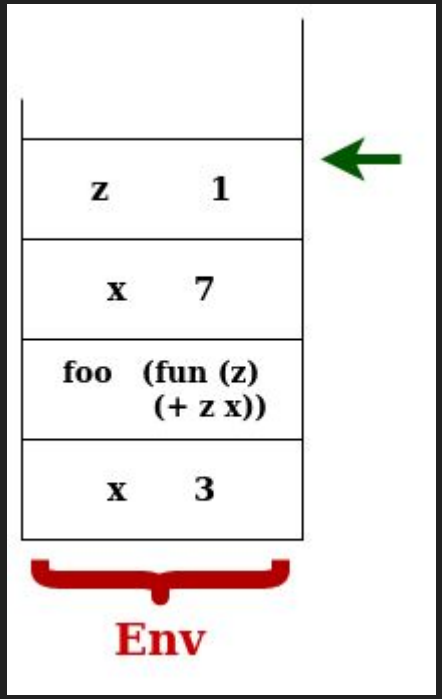
= {+ 1 7}

= 8

Alcance Dinámico

{foo 1}

= { {fun (z) {+ z x}}   1}



Asignamos [z := 1] y Sustituimos en el cuerpo de la función foo ⇒
{+ 1 x}

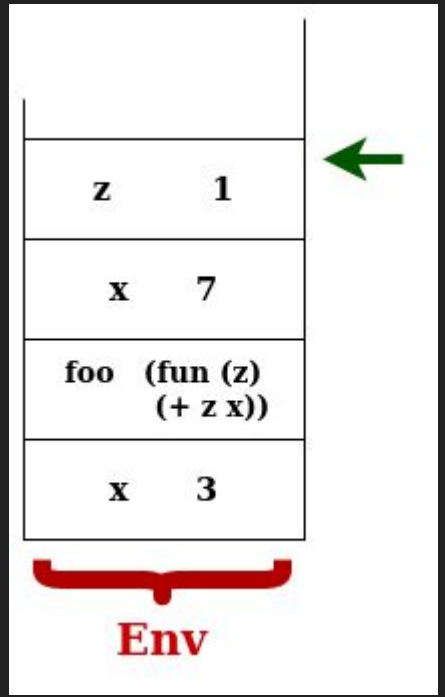
Alcance Dinámico

= {+ 1 x}

Busco en la lista o en la pila desde el inicio de éstas, y encuentro el mismo nombre de id. en el ambiente, entonces obtengo su valor y lo sustituyo en la expresión.

= {+ 1 7}

= 8



Alcance Estático

```
{with {x   3}
     {with {foo {fun {z}  {+  z   x}}}
          {with {x   7}
               {foo   1} }} }
```

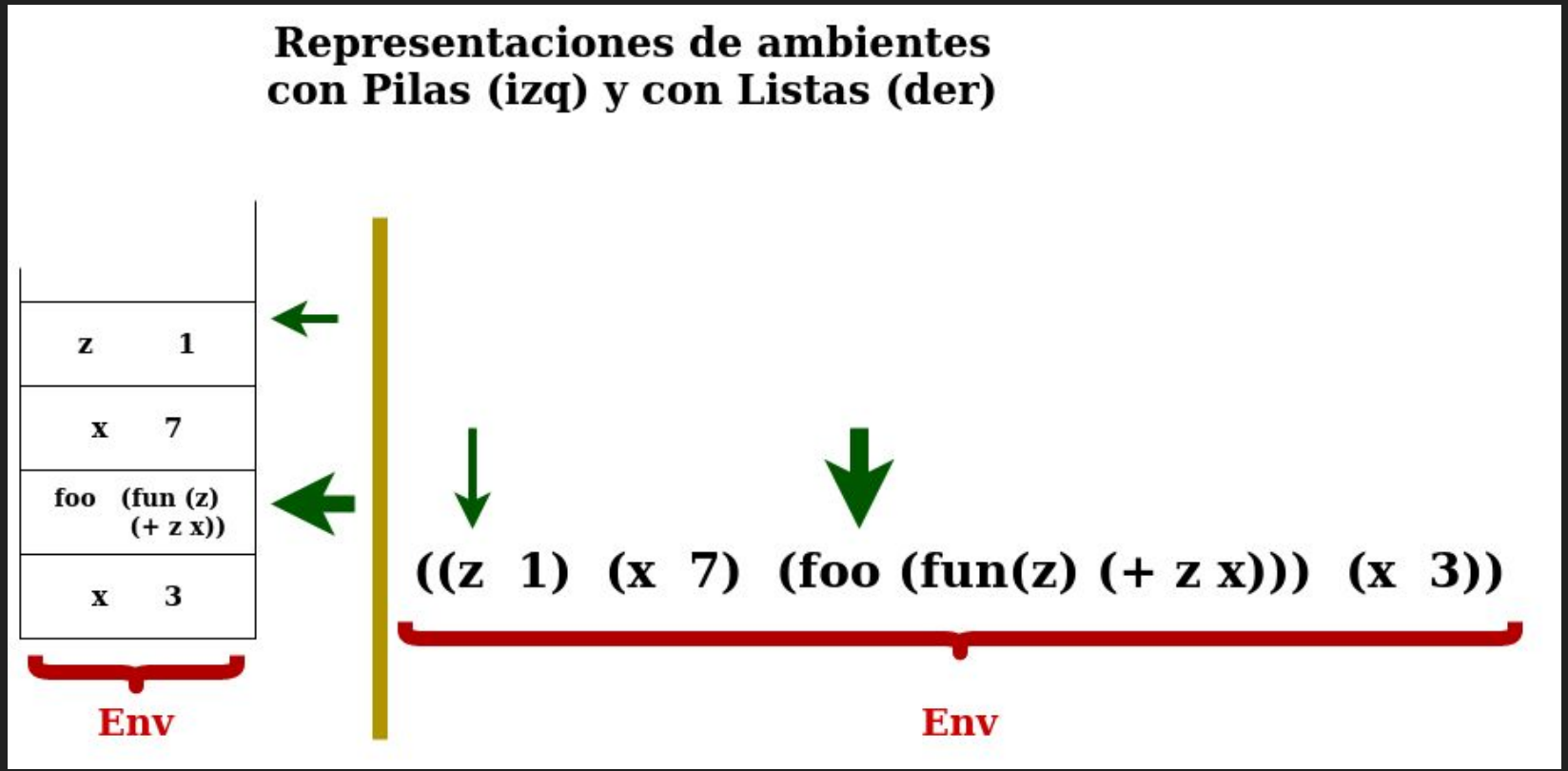
¿Qué tenemos que hacer?

1. Crear el ambiente
2. Evaluar la app de función

{foo   1}

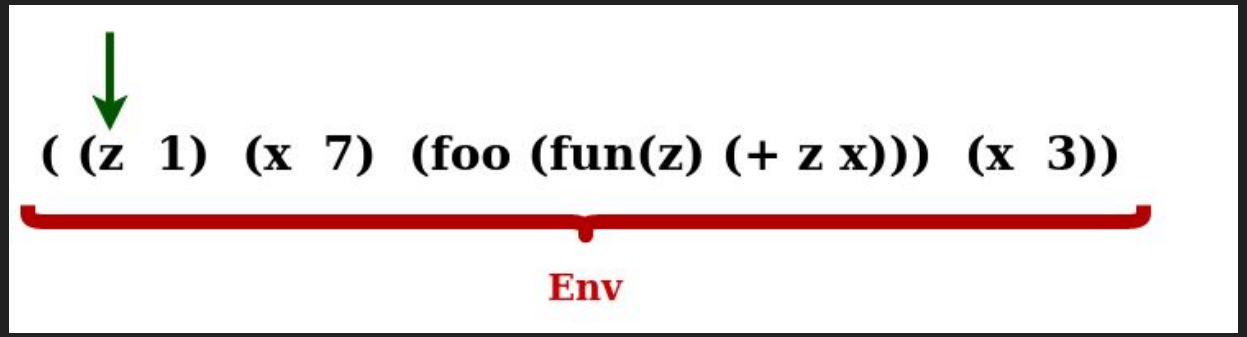
(A)App: asignar parám. formal el parám. real. (B) sustituirlo en el cuerpo de función y (C) evaluarlo.

Crear el ambiente



Alcance Estático

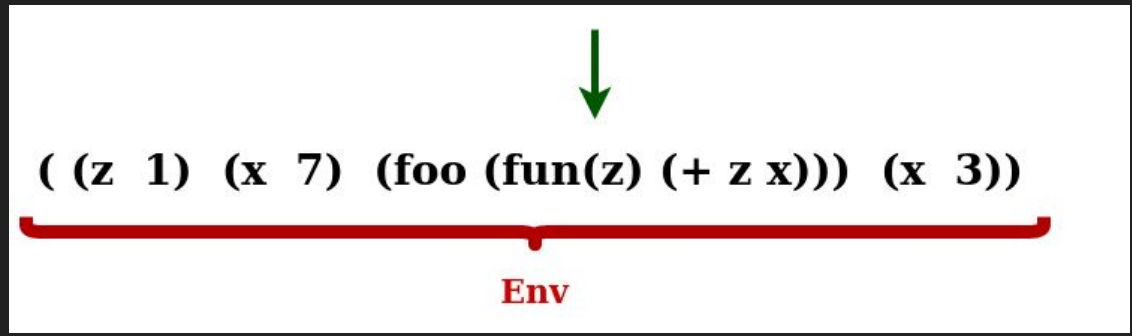
{foo 1}



= { {fun (z) {+ z x}} 1}

Asignamos [z := 1] y Sustituimos en el cuerpo de la función foo ⇒
{+ 1 x}

Alcance Estático



= {+ 1 x}

Busco en la lista o en la pila a partir del ambiente en donde está definida la función es decir ANTES de ésta, si la encuentro entonces obtengo su valor y la sustituyo en la expresión.

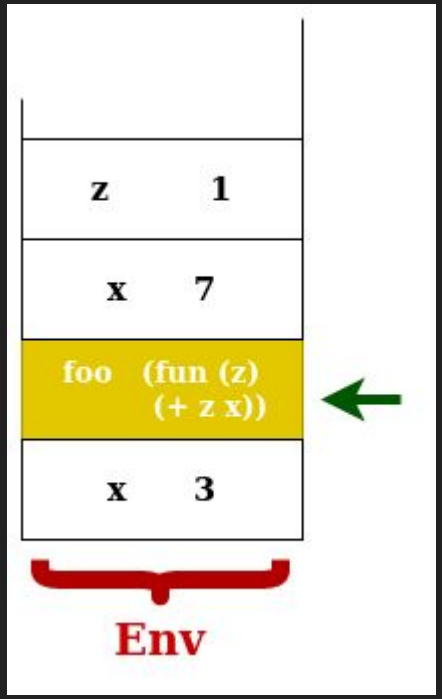
= {+ 1 3}

= 4

Alcance Estático

{foo 1}

= { {fun (z) {+ z x}} 1}



Asignamos [z := 1] y Sustituimos en el cuerpo de la función foo ⇒
{+ 1 x}

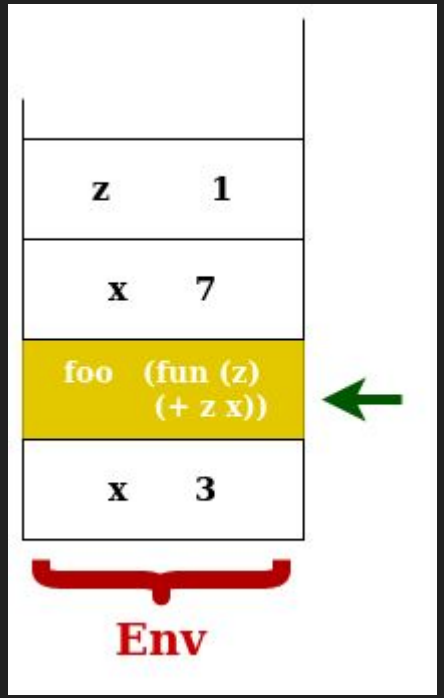
Alcance Estático

= {+ 1 x}

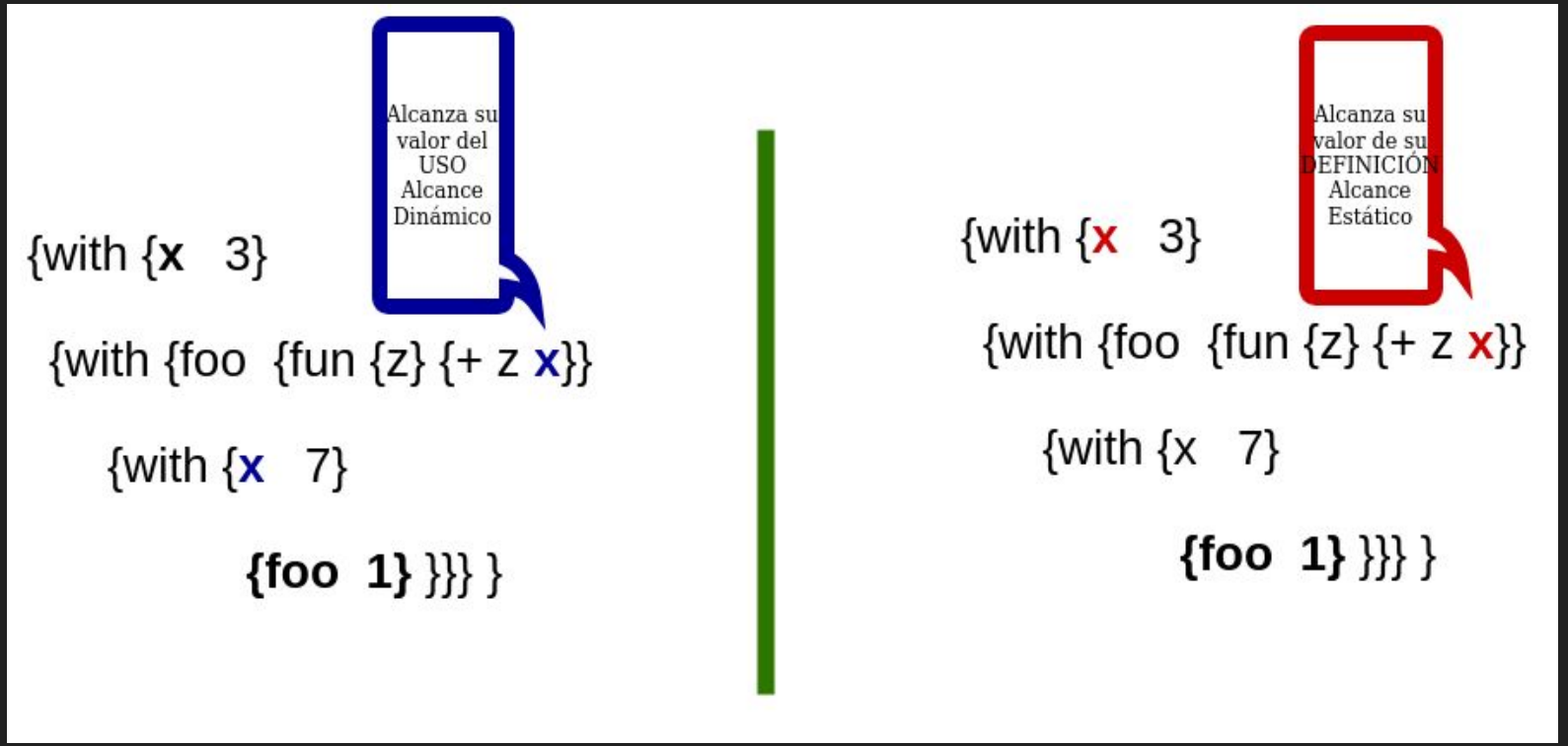
Busco en la lista o en la pila a partir del ambiente en donde está definida la función es decir ANTES de ésta, si la encuentro entonces obtengo su valor y la sustituyo en la expresión.

= {+ 1 3}

= 4



Alcance Dinámico VS Estático



Alcance Dinámico VS Estático

```
{with {x   3}
     {with {foo {fun {z}  {+  z   x}}}
          {with {x   7}
               {foo   1} }} }
= {{fun{z} {+ z x}}   1}
= {+ 1   x} = {+ 1  7} =  8
```

```
{with {x   3}
     {with {foo {fun {z}  {+  z   x}}}
          {with {x   7}
               {foo   1} }} }
= {{fun{z} {+ z x}}   1}
= {+  1   x} = {+  1   3} =  4
```

Decisión de diseño

¿Cuál alcance usarían para diseñar un lenguaje de programación?

¿Qué lenguajes usan alcance estático?

¿Qué lenguajes usan alcance dinámico?

¿Existirá algún lenguaje que tenga implementados AMBOS tipos de alcance?

Tareas Opcionales

Veamos Jamboard de la clase

https://jamboard.google.com/d/1OVcYJq8qXeYNyXkIx3f7yXlRXcHOKR_QOB2gxmAeZnA/viewer?f=4

FUNCIONES

Definición de función

Aplicaciones de función

Introducción a funciones

f (x) = x + 3;

f (x) = x + 3; f(5);

f = (λ(x). x+3)(5);

Extendiendo el lenguaje WAE a FWAE

<FWAE> ::= <num>

| {+ <FWAE> <FWAE>}

| {with {<id> <FWAE>} <FWAE>}

| <id>

| {fun {<id>} <FWAE>}

| {<FWAE> <FWAE>}

Constructor

```
(define-type  FWAE
    [num (n  number?)]
    [add  (lhs   FWAE?) (rhs   FWAE?)]
    [with  (name  symbol?) (named-expr   FWAE?) (body FWAE?)]
    [id (name   symbol?)]
    [fun (param   symbol?) (body  FWAE?)]
    [app  (fun-expr   FWAE?)  (arg-expr   FWAE?)])
```

Clasificación de Funciones

De primer orden: funciones que no pueden ser regresadas como valores.

Orden superior: funciones que pueden ser regresadas como valores.

De primera clase: funciones pueden ser pasadas como argumentos como valores, regresadas por funciones como valores, y almacenadas en estructuras de datos.

Funciones y Aplicaciones de funciones

1. {fun { x } x}
2. {fun { y } {+ x y}}

¿Qué regresa una función?

1. {{fun { x } {+ x 4}} 5}
2. {{fun { z } {- x z}} 5}

¿Qué regresa una aplicación de función?

Funciones en nuestra sintaxis

```
;;Función  identidad

(fun  '(id  x)   (id 'x))


;;Aplicación de función del argumento 5 a la función  identidad

((fun  '(id  x)   (id 'x))    5)
```

Al evaluar la expresión

```
{with {x   3}
      {fun { y }
           {+  x   y}}}
```

Nuestro intérprete regresaría:

```
(fun   'y   (add  (num 3)  (id 'y)) )
```

Azúcar sintáctica

{with {x 5} {+ x 3} }

f (x) = x + 3; f(5);

Reescribir otra expresión: más sencilla para los programadores.

Mantiene la funcionalidad.

Azúcar sintáctica

{with {var named} body}

{ {fun {var} body} named }

{with {x 3} {+ x x}}

= {+ 3 3} = 6

{ {fun {x} {+ x x}} 3 }

={+ 3 3} = 6

Ejemplo 4.

```
{with {x   3}
    {with {f  {fun {y} {+  x  y}}}
        {with {x   5}
            {f  4}}}}
```

1. Creamos el ambiente

   env0 = ( )

   env1 = ((x 3))

   env2 = ((f   (fun(y) (+ x y))) (x 3))

   env3 = ((x 5) (f   (fun(y) (+ x y))) (x 3))

   i.e.

   env0 = ( )

   env1 = (x   3) + env0

   env2 = (f   (fun(y) (+ x y))) + env1

   env3 = (x 5) + env-2

La expresión a evaluar es:

{f 4}

{{fun (y) (+ x y)} 4}

Paso de parámetros

(+ x 4)

Dependiendo del alcance

sustituimos el valor de x

env3=

((x 5) (f (fun(y) (+ x y))) (x 3))

Implica una nueva asignación

y:=4 y lo tenemos que ingresar al ambiente

((y 4) (x 5) (f (fun(y) (+ x y))) (x 3))

La expresión a evaluar es:

(+ x 4)

Alcance dinámico:

(+ 5 4) = 9

Alcance estático:

(+ 3 4) = 7

env3=

((x 5) (f (fun(y) (+ x y))) (x 3))

Implica una nueva asignación

y:=4 y lo tenemos que ingresar al ambiente

((y 4) (x 5) (f (fun(y) (+ x y))) (x 3))

Gracias