

# Lenguajes de Programación. Tarea01

Abrego Álvarez Jonathan  
Ascencio Espindola Jorge Eduardo  
Moreno de la Rosa Alan

September 21, 2015

## 1 Problema I

Hemos visto en clase que la definición de sustitución resulta en una operación ineficiente: en el peor caso es de orden cuadrático en relación al tamaño del programa (considerando el tamaño del programa como el número de nodos en el árbol de sintaxis abstracta). También se vio la alternativa de diferir la sustitución por medio ambientes. Sin embargo, implementar un ambiente usando un stack no parece ser mucho mas eficiente. Responde las siguientes preguntas.

- Provee un esquema para un programa que ilustre la no-linealidad de la implementación de ambientes basada en un stack. Explica brevemente porque su ejecución en tiempo no es lineal con respecto al tamaño de su entrada.

```
{with {x 5}
  {with {y 6}
    { with{z 5}
      {with{w 4}
        (+ x (+ y (+ z w))))}}}}
```

La ejecución de esta expresión no es lineal debido a que por cada asignación de variable, esta se tiene que buscar en el ambiente, y como buscar en el ambiente cuesta  $O(n)$  y esta operación se se debe hacer con respecto al número de variables, que es  $n$ , entonces la complejidad es  $O(n^2)$

- Describe una estructura de datos para un ambiente que un interprete de FWAE pueda usar para mejorar su complejidad

Una estructura de datos que puede ocupar el intérprete para que la complejidad mejore es un Diccionario, mejor conocido como "Tabla hash".

Al meter cada uno de los valores en el ambiente se crean instancias llave-valor y la complejidad de acceso cada uno de los elementos del ambiente es de  $O(1)$  en lugar de que fuera  $O(n)$  buscando una por una cada actualización.

- Muestra como usaría el intérprete esta nueva estructura de datos.

```

(define (interp expr hash);; en lugar de env, el nombre hash de hash-table
  (type-case FAE expr
    [num (n) (numV n)]
    [add (l r) (num+ (interp l hash) (interp r hash))]
    [sub (l r) (num- (interp l hash) (interp r hash))]
    [id (v) (hash-getVal v hash)] ;;Cambie esto. El Hash-getVal es O(1)
    [fun (bound-id bound-body)
      (closureV bound-id bound-body hash)]
    [app (fun-expr arg-expr)
      (local ([define fun-val (interp fun-expr hash)])
        (if (closureV? fun-val)
            (interp (closureV-body fun-val)
                    (aSub (closureV-param fun-val)
                          (interp arg-expr hash)
                          (closureV-env fun-val)))
            (error 'interp (string-append (~a fun-expr) " expression is not a func

```

- Indica cual es la nueva complejidad del interprete (análisis del peor caso) y de forma informal pero rigurosa pruébalo.

La función lookup, que era la que checaba si estaba o no en el ambiente, ahora es reemplazada por la función que se encarga de extraer del hash el valor del id. Y como este es de acceso  $O(1)$  y como cada id (si se consideran  $n$  id's) se tiene que buscar el valor del id por cada id, por lo tanto la complejidad en el peor caso sería  $O(n)$ .

## 2 Problema II

Dada la siguiente expresión de FWAE:

```

{with {x 4}
  {with {f {fun {y} {+ x y}}}
    {with {x 5}
      {f 10}}}}

```

debe evaluar a (num 14) usando alcance estático, mientras que usando alcance dinámico se obtendría (num 15), Ahora Ben un agudo pero excéntrico estudiante dice que podemos seguir usando alcance dinámico mientras tomemos el valor mas viejo de x en el ambiente en vez del nuevo y para este ejemplo el tiene razón.

- ¿ Lo que dice Ben esta bien en general? si es el caso justificalo.
- Si Ben esta equivocado entonces da un programa de contraejemplo y explica por que la estrategia de evaluación de Ben podría producir una respuesta incorrecta

Sí esta bien en general ya que se recomienda que los nombres de las variables no se repitan para evitar confusiones o modificaciones en como manejar los ambientes del interp y obtener el valor deseado, por lo que las variables se

declararían una sola vez así no habría problemas al buscar su valor. Pero hay muchas expresiones que no tienen problemas al tener el mismo nombre de las variables osea que ese general sí tendría problemas si se evalua con el algoritmo de Ben y los nombres de las variables se repiten.

```
{with {x 2}
  {with {x 3}
    {+ x {with {foo {fun {y} {+ 5 y}}}
      {with {x 5}
        {foo x}}}}}}}
```

En este caso podemos ver que el programa no tiene problemas al evaluarse en el interp, devuelve (num 13), pero si lo hacemos con el algoritmo de Ben nos da (num 9)

### 3 Problema III

Dada la siguiente expresión de FWAE con with multi-parametrico:

```
{with {{x 5} {adder {fun {x} {fun {y} {+ x y}}}} {z 3}}
  {with {{y 10} {add5 {adder x}}}
    {add5 {with {{x {+ 10 z}} {y {add5 0}}}
      {+ {+ y x} z}}}}}
```

- Da la forma Bruijn de la expresión anterior
- Realiza la corrida de esta expresión, es decir escribe explícitamente cada una de las llamadas tanto para subst y interp, escribiendo además los resultados parciales en sintaxis concreta.

```
{with {5 {fun {x} {fun {y} {+ x y}}} 3}
  {with {10 {<:0 1> <:0 0>}}
    {<:0 1> {with {{+ 10 <:1 2>} {<:0 1> 0}}
      {+ {+ <:0 1> <:0 0>} <:2 2>}}}}}
```

;;Primero hacemos el parse

```
{with {{x 5} {adder {fun {x} {fun {y} {+ x y}}}} {z 3}}
  {with {{y 10} {add5 {adder x}}}
    {add5 {with {{x {+ 10 z}} {y {add5 0}}}
      {+ {+ y x} z}}}}}
```

```
{parse {with {{x 5} {adder {fun {x} {fun {y} {+ x y}}}} {z 3}}
  {with {{y 10} {add5 {adder x}}}
    {add5 {with {{x {+ 10 z}} {y {add5 0}}}
      {+ {+ y x} z}}}}}
```

```
{with {x adder z}
  {parse {5 {fun {x} {fun {y} {+ x y}}} 3}}
  {parse {with {{y 10} {add5 {adder x}}}
    {add5 {with {{x {+ 10 z}} {y {add5 0}}}
      {+ {+ y x} z}}}}}
```

```

{with {x adder z}
  {{num 5} {fun x {parse {fun {y} {+ x y}}}} {num 3}}
  {with {y add5}
    {parse {10 {adder x}}}
    {parse {add5 {with {{x {+ 10 z}} {y {add5 0}}}
      {+ {+ y x} z}}}}}}

{with {x adder z}
  {{num 5} {fun x {fun y {parse {+ x y}}}} {num 3}}
  {with {y add5}
    {{num 10} {app {parse adder} {parse x}}}
    {app {parse add5} {parse {with {{x {+ 10 z}} {y {add5 0}}}
      {+ {+ y x} z}}}}}}

{with {x adder z}
  {{num 5} {fun x {fun y {add {parse x} {parse y}}}} {num 3}}
  {with {y add5}
    {{num 10} {app {id adder} {id x}}}
    {app {id add5} {with {x y}
      {parse {{+ 10 z} {add5 0}}}
      {parse {+ {+ y x} z}}}}}}

{with {x adder z}
  {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}
  {with {y add5}
    {{num 10} {app {id adder} {id x}}}
    {app {id add5} {with {x y}
      {{add {parse 10} {parse z}} {app {parse add5} {parse 0}}}
      {add {parse {+ y x}} {parse z}}}}}}

{with {x adder z}
  {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}
  {with {y add5}
    {{num 10} {app {id adder} {id x}}}
    {app {id add5} {with {x y}
      {{add {num 10} {id z}} {app {id add5} {num 0}}}
      {add {add {parse y} {parse x} {id z}}}}}}

{with {x adder z}
  {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}
  {with {y add5}
    {{num 10} {app {id adder} {id x}}}
    {app {id add5} {with {x y}
      {{add {num 10} {id z}} {app {id add5} {num 0}}}
      {add {add {id y} {id x} {id z}}}}}}

;;Aquí termina el parse

```

```
;;Iniciamos el interp
{interp {with {x adder z}
  {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}
  {with {y add5}
    {num 10} {app {id adder} {id x}}
    {app {id add5} {with {x y}
      {{add {num 10} {id z}} {app {id add5} {num 0}}}
      {add {add {id y} {id x}} {id z}}}}}}}
```

```
{interp {subst {with {y add5}
  {{num 10} {app {id adder} {id x}}
  {app {id add5} {with {x y}
    {{add {num 10} {id z}} {app {id add5} {num 0}}}
    {add {add {id y} {id x}} {id z}}}}}}
  {x adder z}
  {interp {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}}
```

;Hacemos el primer subinterp

```
{interp {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}
->{{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}
```

;Hacemos el subst

```
{subst {with {y add5}
  {{num 10} {app {id adder} {id x}}
  {app {id add5} {with {x y}
    {{add {num 10} {id z}} {app {id add5} {num 0}}}
    {add {add {id y} {id x}} {id z}}}}}}
  {x adder z}
  {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}

{with {y add5}
  {subst {{num 10} {app {id adder} {id x}}
    {x adder z}
    {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}
  {subst {app {id add5} {with {x y}
    {{add {num 10} {id z}} {app {id add5} {num 0}}}
    {add {add {id y} {id x}} {id z}}}}
    {x adder z}
    {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}}
```

```

{with {y add5}
  {{num 10} {app {subst {id adder}
    {x adder z}
    {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}}
    {subst {id x}
      {x adder z}
      {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}}}
  {app {subst {id add5}
    {x adder z}
    {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}
    {subst {with {x y}
      {{add {num 10} {id z}} {app {id add5} {num 0}}}
      {add {add {id y} {id x}} {id z}}}
      {x adder z}
      {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}}}

{with {y add5}
  {{num 10} {app {fun x {fun y {add {id x} {id y}}}} {num 5}}}
  {app {id add5}
    {with {x y}
      {subst {{add {num 10} {id z}} {app {id add5} {num 0}}}
        {x adder z}
        {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}
      {subst {add {add {id y} {id x}} {id z}}
        {x adder z}
        {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}}}

{with {y add5}
  {{num 10} {app {fun x {fun y {add {id x} {id y}}}} {num 5}}}
  {app {id add5}
    {with {x y}
      {{add {subst {num 10}
        {x adder z}
        {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}
        {subst {id z}
          {x adder z}
          {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}}}
      {app {subst {id add5}
        {x adder z}
        {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}
        {subst {num 0}
          {x adder z}
          {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}}}
      {add {subst {add {id y} {id x}}
        {x adder z}
        {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}
        {subst {id z}
          {x adder z}
          {{num 5} {fun x {fun y {add {id x} {id y}}}} {num 3}}}}}}}

```

```

{with {y add5}
  {{num 10} {app {fun x {fun y {add {id x} {id y}}}} {num 5}}}}
  {app {id add5} {with {x y}
    {{add {num 10} {num 3}} {app {id add5} {num 0}}}}
    {add {add {subst {id y}
      {x adder z}
      {{num 5} {fun x {fun y {add {id x} {id y}}}} {
        {subst {id x}
          {x adder z}
          {{num 5} {fun x {fun y {add {id x} {id y}}}} {
            {num 3}}}}}}

```

```

-> {with {y add5}
  {{num 10} {app {fun x {fun y {add {id x} {id y}}}} {num 5}}}}
  {app {id add5} {with {x y}
    {{add {num 10} {num 3}} {app {id add5} {num 0}}}}
    {add {add {subst {id y} {id x}} {num 3}}}}}}

```

```

;Regresando al interp
{interp {with {y add5}
  {{num 10} {app {fun x {fun y {add {id x} {id y}}}} {num 5}}}}
  {app {id add5} {with {x y}
    {{add {num 10} {num 3}} {app {id add5} {num 0}}}}
    {add {add {subst {id y} {id x}} {num 3}}}}}}

{interp {subst {app {id add5}
  {with {x y}
    {{add {num 10} {num 3}} {app {id add5} {num 0}}}}
    {add {add {subst {id y} {id x}} {num 3}}}}}}
  {y add5}
  {interp {{num 10} {app {fun x {fun y {add {id x} {id y}}}} {num 5}}}}}}

```

```

;Hacemos el segundo sub-interp
{interp {{num 10} {app {fun x {fun y {add {id x} {id y}}}} {num 5}}}}

{{num 10}
  (local ([define fun-val (interp {fun x {fun y {add {id x} {id y}}}})])
    (interp (subst (fun-body fun-val)
      (fun-param fun-val)
      (interp (num 5)))))}

```

```

;Resolvemos (interp {fun x {fun y {add {id x} {id y}}}})
-> {fun x {fun y {add {id x} {id y}}}}
;Sustituimos

```

```

{{num 10}
  (local ([define fun-val (interp {fun x {fun y {add {id x} {id y}}}})])
    (interp (subst (fun-body {fun x {fun y {add {id x} {id y}}})
                     (fun-param {fun x {fun y {add {id x} {id y}}})
                     {num 5}))))}

```

```

{{num 10}
  (interp (subst {fun y {add {id x} {id y}}} (x) {num 5})))}

```

```

;Resolvemos (subst {fun y {add {id x} {id y}}} (x) {num 5})
(subst {fun y {add {id x} {id y}}} (x) {num 5})

```

```

{fun y {subst {add {id x} {id y}}} (x) {num 5}}}

```

```

{fun y {add {subst {id x}(x) {num 5}} {subst {id y} (x) {num 5}}}}

```

```

-> {fun y {add {num 5} {id y}}}
;Sustituimos

```

```

{{num 10} (interp {fun y {add {num 5} {id y}}})}
-> {{num 10} {fun y {add {num 5} {id y}}}}

```

```

;Hacemos el segundo subst

```

```

{subst {app {id add5}
  {with {x y}
    {{add {num 10} {num 3}} {app {id add5} {num 0}}}
    {add {add {subst {id y} {id x}}} {num 3}}}}
  {y add5}
  {{num 10} {fun y {add {num 5} {id y}}}}}

```

```

{app {subst {id add5}
  {y add5}
  {{num 10} {fun y {add {num 5} {id y}}}}}
  {subst {with {x y}
    {{add {num 10} {num 3}} {app {id add5} {num 0}}}
    {add {add {subst {id y} {id x}}} {num 3}}
    {y add5}
    {{num 10} {fun y {add {num 5} {id y}}}}}
  }}

```

```

{app {fun y {add {num 5} {id y}}}
  {with {x y}
    {subst {{add {num 10} {num 3}} {app {id add5} {num 0}}}
      {y add5}
      {{num 10} {fun y {add {num 5} {id y}}}}}
    {subst {add {add {subst {id y} {id x}}} {num 3}}
      {y add5}
      {{num 10} {fun y {add {num 5} {id y}}}}}}}

```



```

{app {fun y {add {num 5} {id y}}}
  {with {x y}
    {{add {subst {num 10}
      {y add5}
      {{num 10} {fun y {add {num 5} {id y}}}}}
    {subst {num 3}
      {y add5}
      {{num 10} {fun y {add {num 5} {id y}}}}}
    {app {subst {id add5}
      {y add5}
      {{num 10} {fun y {add {num 5} {id y}}}}}
    {subst {num 0}
      {y add5}
      {{num 10} {fun y {add {num 5} {id y}}}}}
    {add {add {subst {id y}
      {y add5}
      {{num 10} {fun y {add {num 5} {id y}}}}}
      {subst {id x}
        {y add5}
        {{num 10} {fun y {add {num 5} {id y}}}}}
      {subst {num 3}
        {y add5}
        {{num 10} {fun y {add {num 5} {id y}}}}}
      {{num 10} {fun y {add {num 5} {id y}}}}}}}

->{app {fun y {add {num 5} {id y}}}
  {with {x y}
    {{add {num 10} {num 3}} {app {fun y {add {num 5} {id y}}} {num 0}}}
    {add {add {id y} {id x}} {num 3}}}}

;Regresando al interp
{interp {app {fun y {add {num 5} {id y}}}
  {with {x y}
    {{add {num 10} {num 3}} {app {fun y {add {num 5} {id y}}} {num 0}}}
    {add {add {id y} {id x}} {num 3}}}}}

(local ([define fun-val (interp {fun y {add {num 5} {id y}})])
  (interp (subst (fun-body fun-val)
    (fun-param fun-val)
    (interp {with {x y}
      {{add {num 10} {num 3}} {app {fun y {add {num 5} {id y}}} {num 0}}}
      {add {add {id y} {id x}} {num 3}}}})))

;Resolviendo el tercer sub-interp
(interp {with {x y}
  {{add {num 10} {num 3}} {app {fun y {add {num 5} {id y}}} {num 0}}}
  {add {add {id y} {id x}} {num 3}}})

(interp {subst {add {add {id y} {id x}} {num 3}}
  {x y}
  {{add {num 10} {num 3}} {app {fun y {add {num 5} {id y}}} {num 0}}})

```

```
;Resolviendo {subst {add {add {id y} {id x}} {num 3}} {x y} {{add {num 10} {num 3}} {a
{subst {add {add {id y} {id x}} {num 3}}
{x y}
{{add {num 10} {num 3}} {app {fun y {add {num 5} {id y}}} {num 0}}}}
```

```
{add {subst {add {id y} {id x}}
{x y}
{{add {num 10} {num 3}} {app {fun y {add {num 5} {id y}}} {num 0}}}}
{subst {num 3}
{x y}
{{add {num 10} {num 3}} {app {fun y {add {num 5} {id y}}} {num 0}}}}}
```

```
{add {add {subst {id y}
{x y}
{{add {num 10} {num 3}} {app {fun y {add {num 5} {id y}}} {num 0}}}}
{subst {id x}
{x y}
{{add {num 10} {num 3}} {app {fun y {add {num 5} {id y}}} {num 0}}}}
{num 3}}
```

```
->{add {add {app {fun y {add {num 5} {id y}}} {num 0}} {add {num 10} {num 3}}}
{num 3}}
;Sustituimos
```

```
(interp {add {add {app {fun y {add {num 5} {id y}}} {num 0}} {add {num 10} {num 3}}} {num
(add-numbers (interp {add {app {fun y {add {num 5} {id y}}} {num 0}} {add {num 10} {num 3}
(add-numbers (add-numbers (interp {app {fun y {add {num 5} {id y}}} {num 0})) (interp {ad
(add-numbers (add-numbers (interp {app {fun y {add {num 5} {id y}}} {num 0})) (add-number
```

```
; (interp {app {fun y {add {num 5} {id y}}} {num 0}})
(interp {app {fun y {add {num 5} {id y}}} {num 0}})

(local ([define fun-val (interp {fun y {add {num 5} {id y}})])
(interp (subst (fun-body fun-val)
(fun-param fun-val)
(interp {num 0}))))
(interp (subst ({add {num 5} {id y}})
(y)
(interp {num 0})))

(interp (subst ({add {num 5} {id y}})
(y)
{num 0}))
```

```

;Resolviendo sub-interp
{add {subst {num 5}
          (y)
          {num 0}}
  {subst {id y}
        (y)
        {num 0}}}}

->{add {num 5} {num 0}}
;Fin
(interp {add {num 5} {num 0}})

(add-numbers (interp {num 5}) (interp {num 0}))

-> (add-numbers {num 5} {num 0})
;Fin
-> (add-numbers (add-numbers (add-numbers {num 5} {num 0}) (add-numbers {num 10} {num 3})))

```