Lenguajes de Programación
Tarea 1

Dondiego Pacheco Ricardo Daniel
Mendoza Ponce Miguel Angel
Serrato Solano Victor Manuel

21 de septiembre de 2015
Facultad de Ciencias UNAM

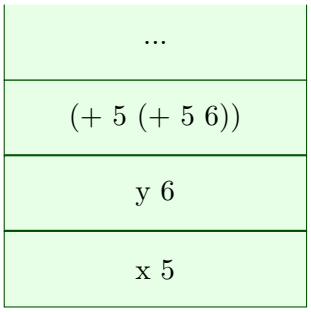
1. Problema I

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 implementacion 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}
            (+ x (+ x y))}

Para el ambiente:
env0 = ()
env1 = ((x 5))
env2 = ((y 6) (x 5))
```



Cuando realizamos la busqueda en el stack para la llamada de la funcion:

```
(+ x (+ x y))
```

entonces hacemos tres veces lookup (busqueda en el stack) en primera x es el primer elemento en el stack y recorremos todo el stack para encontrarlo (n) eso se hace dos veces pues aparece x en dos ocasiones y y aparece una vez y en este caso es el primer elemento $(n-1)$.

Entonces:
$O(((2n)+(n-1))(n)) = O((3n-1)(n)) = O(3n^2-n) = O(n^2)$

Si esto sucede en todos los programas al hacer la busqueda en el peor de los caso la sustitución no será lineal entonces será $O(n^2)$.

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

Podemos utilizar la estructura de datos "Tabla Hash". Esta estructura es tiempo constante pues al proveer una buena funcion para almacenar nuestros datos y buscarlos y asegurar la complejidad.

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

Para poder utilizar nuestra Tabla Hash vamos a ver cada env en (expression env) como una tabla hash en donde cada una tendra nuestro contenido de ambiente:
env n = ((varn valn), ... (var1 val1))
En la implementacion todo sera igual, como estaba en FWAE, pero para la aplicación de la función será:

```
[app (fun-expr arg-expr)
     (local ([define fun-val (interp fun-expr env)])  <-- por default en env0
        (if (closureV? fun-val)
            (interp (closureV-body fun-val)
                (aSub (closureV-param fun-val
                      (interp arg-expr env
                      (closureV-env fun-val)))
            (error 'interp (string-append
            (~a fun-expr) " expression is not a function"))))
```

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

La complejidad del interprete es $O(n)$ pues el obtener los elementos y agregar requiere $O(1)$, pero esto lo hacemos n veces (en el peor caso) por eso la complejidad.

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 alance 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.

Evaluación de alcance estático.

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

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

{subst {with {x 5}
    {f 10}}}}}

 Aplicando sustitución y evaluación
{f {fun {y} {+ x y}}
{f {fun {y} {+ 4 y}}
{f {fun {10} {+ 4 10}}
{f {fun {10} {14}}
=(num 14)
```

Evaluación de alcance dinamico.

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

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

{subst {with {x 5}
    {f 10}}}}}

 Aplicando sustitución y evaluación
{f {fun {y} {+ x y}}
{f {fun {y} {+ 5 y}}
{f {fun {10} {+ 5 10}}
{f {fun {10} {15}}
=(num 15)
```

- ¿Lo que dice Ben esta bien en general? si es el caso justificalo.

Lo que Ben dice esta mal pues en este ejemplo aunque esta bien porque $x = 4$ y es el valor mas viejo. Para un caso general esta incorrecto

- 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.

```
{with {x 1}
    {with {x 2}
        {with {x 3}
            {with {x 4}
                {with {f {fun {y} {+ x y}}}
                    {with {x 5}
                        {with {x 6}
                            {f 7}}}}}}}
```

En este caso el valor mas viejo de x es 1, en alcance estatico el valor es 7 y en dinamico es 5, entonces lo que Ben dice esta mal.

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

```
{with {{5} {adder {fun {x} {fun {y} {+ x y}}}} {3}}
    {with {{10} {add5 {adder <0 0>}}}
        {add5 {with {{+ 10 <1 2>} {y {add5 0}}}
            {+ {+ <0 1> <0 0> } <2 2>}}}}}
```

- 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.

```
Primero hay que aplicar el Parser

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

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

{with {{(id x) (num 5)} {adder {fun {parse {x}} {fun {parse{y}} parse{+ x y}}}}} {(id
    {with {{(id y) (num 10)} {add5 parse{adder x}}}
          {add5 {with parse{{x {+ 10 z}} {y {add5 0}}}
                  parse{+ {+ y x} z}}}}}

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

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

{with {{(id x) (num 5)} {adder {fun {(id x)} {fun {(id y)} {add (id x) (id y)}}}}} {(
    {with {{(id y) (num 10)} {add5 {adder (id x)}}}
          {add5 {with {(id x) {add (num 10) (id z)}} {(id y) {add5 (id 0)}}}
                  {add {add (id y) (id x)} {(id z)}}}}}


Aplicamos interp y subst

{interp {with {{(id x) (num 5)} {adder {fun {(id x)} {fun {(id y)} {add (id x) (id y)
    {with {{(id y) (num 10)} {add5 {adder (id x)}}}
         {add5 {with {{(id x) {add (num 10) (id z)}} {(id y) {add5 (num 0)}}}
              {add {add (id y) (id x)} (id z)}}}}}}
```

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

{interp {with {{(id x) (num 5)} {adder {subst {fun {(id x)} {fun {(id y)} {add (id x)
    {subst {with {{(id y) (num 10)} {add5 {adder (id x)}}}
         {add5 {with {{(id x) {add (num 10) (id z)}} {(id y) {add5 (num 0)}}}
              {add {add (id y) (id x)} (id z)}}}}}}}}}

{interp {with {{(id x) (num 5)} {adder {fun {(id x)} {fun {(id y)} {add (num 5) (id y
    {subst {with {{(id y) (num 10)} {add5 {adder (id x)}}}
         {add5 {with {{(id x) {add (num 10) (id z)}} {(id y) {add5 (num 0)}}}
              {add {add (id y) (id x)} (id z)}}}}}}}}}

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