Auxiliar Cuatro - CC41A
Lenguajes de Programación
Substituci'on y Funciones (1er Parte)

Oscar E. A. Callaú
oalvarez@dcc.uchile.cl

Santiago - Chile, 6/Abr/2008

1. Substituación Explícita, C1 I-2007

Considere la siguiente implementación de la función de substitución `subst` del lenguaje WAE visto en clases:

```
;; subst: WAE symbol WAE -> WAE
(define (subst expr sub-id val)
  (type-case WAE expr
    (num (n) expr)
    (add (l r) (add (subst l sub-id val)
                    (subst r sub-id val)))
    (sub (l r) (sub (subst l sub-id val)
                    (subst r sub-id val)))
    (with (bound-id named-expr bound-body)
          (if (symbol=? bound-id sub-id)
              expr
              (with bound-id
                    named-expr
                    (subst bound-body sub-id val))))
    (id (v) (if (symbol=? v sub-id) val expr))))
```

Con esta función de substitución, el calculador se cae en las dos expresiones siguientes:

1. `{with {x 5} {with {y x} y}}`
2. `{with {x 5} {with {x x} x}}`

Para cada caso explique porqué se cae el calculador, y luego de una versión corregida de `subst`.

2. Más Substitución

En que casos y porque falla esta implementación de substitución

```
(define (subst expr sub-id val)
  (type-case WAE expr
             [num (n) expr]
             [add (l r) (add (subst l sub-id val) (subst r sub-id val))]
             [sub (l r) (sub (subst l sub-id val) (subst r sub-id val))]
             [with (bound-id named-expr bound-body)
                   (if (symbol=? bound-id sub-id)
                       (with bound-id
                             named-expr
                             (subst bound-body sub-id val))
                       (with bound-id
                             (subst named-expr sub-id val)
                             (subst bound-body sub-id val)))]
             [id (v) (if (symbol=? v sub-id) val expr)]))
```

3. Cálculo Lambda

Según la siguiente definición de λ-cálculo, en Gramática Concreta:

```
(define-type Lambda-Calculus
  (Id (id symbol?))
  (Lambda (parameter symbol?) (body Lambda-Calculus?))
  (App (function Lambda-Calculus) (argument Lambda-Calculus)))
```

implemente la función:

eqLambda , el cual toma dos expresiones λ como parámetros y retorna si ellas son **exactamente** iguales.

4. De Bruijn, C1 II-2008

Al definir la operación de substitución, uno de los problemas principales fue especificar bien lo que pasa cuando se usa el mismo identificador varias veces. Resulta que es posible deshacerse de los nombres, simplemente. Nicolas De Bruijn propusó reemplazar nombres por números, más bien, *índices*, que indican la profundidad de una asociación (binding).[1]

Por ejemplo, en vez de escribir: `{with {x 5}{+ x x}}`, podemos escribir: `{with 5 {+ <0> <0>}}`.

Por convención el alcance (scope) corriente es `0`, entonces `x` se reemplazó por el índice `<0>`. Con esta representación, basta saber que la presencia de un `with` indica que entramos en un nuevo alcance.

Así mismo, la siguiente expresión:

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

[1] Esa representación es usada por (casi) todos los compiladores.

se convierte en:

```
{with 5
  {with 4
    {+ <1> <0>}}}
```

1. Escriba la conversión de la siguiente expresión usando índices de De Bruijn:

   ```
   {with {x 5}
     {with {y {+ x 2}}
       {+ x
          {with {y {- y 1}}
            {+ y x}}}}}
   ```

2. Extiende la gramtica del lenguaje `WAE` para acomodar expresiones con ndices de De Bruijn.

3. Implemente la función `toDeBruijn :: WAE ->WAE` que transforma una expresión con `with` normales a una expresión con `with` modificado e índices De Bruijn.

5. Scope, C1 I-2007

Considere un lenguaje donde la ejecuci ón del siguiente programa retorna el valor 5:

```
(define (f p) n)
(local ((define n 5))
(f 10)) --> retorna 5
```

- Como caracterizar ía tal lenguaje?
- En que difiere de lo que esperaría un programador Java?
- Que modificaci ón har ía su interprete para volver a la semántica esperada?

6. F1WAE + Recursión, C2 I-2007

En un intérprete con funciones de primer orden (tal como el F1WAE), Se puede soportar definiciones recursivas? Considera que añadir nuevos elementos a su gramática solucione este problema? Explique.

Propuesto. Implemente el lenguaje F1CWAE, cuya base es F1WAE incluyendo todos los cambios necesarios que permiten definir la función `factorial`.