PARADIGMAS DE PROGRAMACIÓN

2006

CALCULO LAMBDA

CALCULO LAMBDA

El **cálculo lambda** fue desarrollado por **Alonso Church** en la década del 30 con el objeto de dar una **teoría general de las funciones**.

El cálculo lambda ha sido empleado como fundamento conceptual de los lenguajes de programación, aportando:

- **una sintaxis básica**
- **una semántica para el concepto de función como proceso de transformación de argumentos en resultados**
- **un medio para definir primitivas de programación**

SINTAXIS DEL CALCULO LAMBDA

La sintaxis del Cálculo Lambda (CL) es la siguiente:

<término> ::= <variable> |

λ<variable> . <término> |

(<término> <término>)

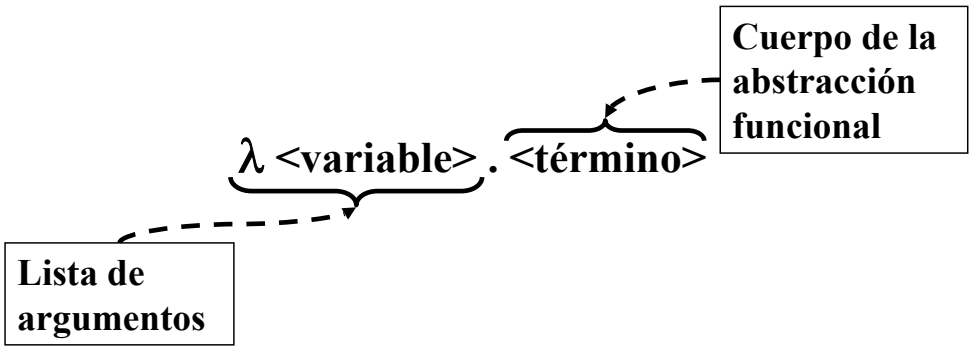
En esta sintaxis no existe el concepto de

<nombre> o <constante>

¿Qué implica esto?

El formalismo **no tendrá primitivas**, no nos permitirá emplear funciones con el concepto de módulos abstractos.

En esta nueva sintaxis la **abstracción funcional** es:



La sintaxis propuesta utiliza la representación de Funciones con un sólo argumento.

Una Función que requiera más de un argumento se representa de la sgte. forma:

$$\lambda\, x_1.\ \lambda x_2.\ \ldots.\ \lambda x_n.\ M$$

CALCULO LAMBDA

Tiene por objeto explicitar el concepto que representa el empleo de funciones como medio de transformación de argumentos en resultados.

A este formalismo lo denominamos CÁLCULO, ya que el mismo empleará

- **un conjunto de axiomas, y**
- **reglas de inferencia (de la misma forma que lo utilizan los sistemas formales)**

para representar el medio de transformación mencionado.

REGLAS DEFINIDAS EN EL CL

De acuerdo a la sintaxis propuesta, una **aplicación funcional** tendrá el siguiente formato

(M N)

la cual producirá un **resultado**, como consecuencia de la correspondiente **regla del cálculo**,

R (resultado)

La consistencia del sistema requiere que

la aplicación funcional y **el resultado**

puedan ser interpretadas como

expresiones equivalentes

(M N) ≈ R

es decir, que representan el mismo valor.

El **cálculo del resultado** de una **aplicación funcional** será obtenido mediante la **generación de expresiones equivalentes** por **aplicación de las reglas del cálculo λ** que definiremos a continuación.

La relación de equivalencia entre expresiones del cálculo λ tendrá las propiedades:

reflexividad	**M = M**
simetría	**M = N ⇒ N = M**
transitividad	**M = N ∧ N = P ⇒ M = P**

más las siguientes, que resultan lo suficientemente intuitivas

M = N ⇒ (M P) = (N P)

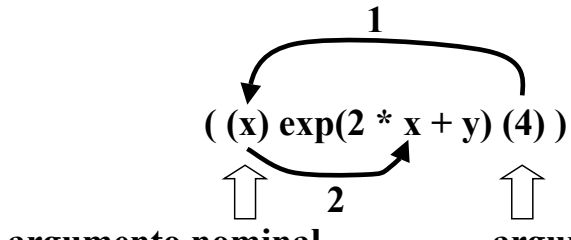
M = N ⇒ (P M) = (P N)

M = N ⇒ λx . N = λx . M

La regla del cálculo λ que permite generar expresiones que satisfagan la relación de equivalencia se denomina

Regla Beta

La Regla Beta establece como sustituir en el cuerpo de la abstracción funcional cada ocurrencia de la variable que hace de argumento nominal por el argumento efectivo de la aplicación funcional correspondiente.



La regla Beta especifica los pasos 1 y 2

Esta idea es recogida por la Regla Beta expresada de la siguiente forma

$$(\lambda x .M\ \ N) = [N/x]\ M$$

la parte derecha de la expresión se interpreta como:

"el término que se obtiene de introducir N en lugar de x, toda vez que x ocurre libre en M"

OCURRENCIA DE UN TERMINO

Definición. Ocurrencia de P en Q

Clausura

P ocurre en P

Inducción

Si P ocurre en M o N $\Rightarrow$ P ocurre en (M N)

Si P ocurre en M o P es igual a x $\Rightarrow$ P ocurre en λx.M

Por ejemplo, dado en el siguiente término:

((x y) λx.(x y))

(x y) ocurre dos veces

x ocurre tres veces

y ocurre dos veces

Los conceptos que necesitamos a continuación son los de ocurrencia libre y ligada de un variable.

Definición. Una ocurrencia de la **variable x** en un **término P es ligada** sí y solo sí, **x** ocurre en un **subtérmino de P** de la forma λ**x.M**

Si se aplica esta definición en el siguiente término

$(\lambda$y.(y y) λx.(x y))

x e **y** *ocurren ligadas* ya que el término contiene los subtérminos

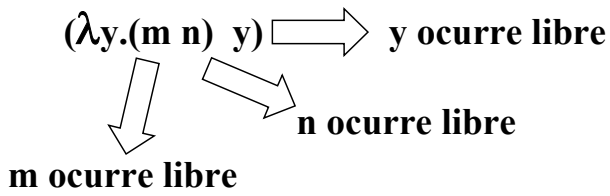
λy.(y y) → y ocurre ligada (3 veces)

λx.(x y) → x ocurre ligada (2 veces)

Definición. La variable **x** ocurre libre en en término **N**, solamente sí:

1. **N = x**
2. **N = λz.M** tal que **x ≠ z,** y **x** ocurre libre en **M**
3. **N = (P Q)** donde **x** ocurre libre en **M** y/o en **P**

Definición. Si **x ocurre libre** al menos un vez en un término **P**, entonces **x es una variable libre de P**.



y es una variable libre del término

OCURRENCIAS DE TERMINOS, OCURRENCIAS LIBRES Y LIGADAS DE VARIABLES

Dado el siguiente término del Cálculo λ:

$$((\lambda y.(x\ y)\ z)\ \lambda x.(x\ \lambda x.(x\ y)))$$

Tenemos que:

(x y) ocurre 2 veces

x ocurre 5 veces, 4 veces ligada y 1 vez libre

y ocurre 3 veces, 2 veces ligada y 1 libre

z ocurre una vez libre

Por lo tanto:

variables libres: x, y, z

variables ligadas: x, y

REGLA BETA: DEFINICIÓN DE SUSTITUCIÓN

$$[N/x]\ M$$

En el caso en que M sea la variable a sustituir, la misma se realiza de la siguiente forma:

1 - [N/x] x :=: N

Ejemplos:

$$[\lambda y.(m\ n)\ /\ y]\ y\ :=:\ \lambda y.(m\ n)$$

$$[(m\ n)\ /\ t]\ t\ :=:\ (m\ n)$$

DEFINICIÓN DE SUSTITUCIÓN

[N/x] M

En el caso en que M sea una variable, pero diferente de la sustituida, la misma se realiza de la siguiente forma:

2 - [N/x] y :=: y ; y ≠ x

Ejemplos:

[λy.(m n) / y] l :=: l

[(m n) / t] r :=: r

DEFINICIÓN DE SUSTITUCIÓN

[N/x] M

En el caso en que M sea una aplicación funcional, la misma se realiza de la siguiente forma:

3 - [N/x] (P Q) :=: ([N/x]P [N/x]Q)

Ejemplos:

[λy.(m n) / y] **(y l) :=: ([λy.(m n) / y]y** [λy.(m n) / y] **l)**

[x / m] **(λy.(m n) m) :=: ([x / m]λy.(m n)** [x / m]**m)**

DEFINICIÓN DE SUSTITUCIÓN

[N/x] M

En el caso en que M sea una abstracción funcional, la misma se realiza de la siguiente forma:

4 - [N/x] λx.M :=: λx.M

Ejemplos:

[λy.(m n) / x] λx.(t x) :=: λx.(t x)

[x / m] **λm.(m n) :=: λm.(m n)**

DEFINICIÓN DE SUSTITUCIÓN

En el caso en que M sea una abstracción funcional, la misma se realiza de la siguiente forma:

5 - [N/x] λy.M :=: λy. [N/x]M

¿Siempre podría reducir así?

((λx. λy. (y x) M) N) — **Por regla Beta y def. 5, 3, 2 y 1**

(λy. (y M) N) — **Por regla Beta y def. 3, 1 y 2**

(N M)

En particular si M :=: y y N :=: x

Por regla Beta y def. 5, 3, 2, 1

((λx. λy. (y x) y) x) **= (x y)**

(λy. (y y) x)

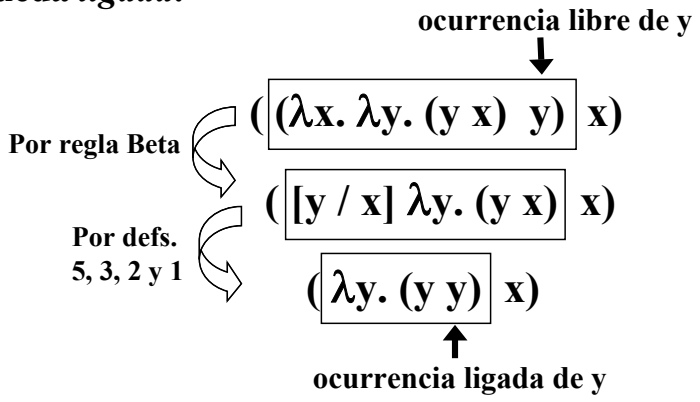
Por regla Beta y def. 3, 2, 1

(x x) ⟹ **¿ x = y ?**

Por transitividad

En el ejemplo anterior x e y denotan términos genéricos ⇨ ~~**los términos cualesquiera del cálculo lambda son iguales**~~ **Incorrecto!**

El problema surge pues, al evaluar la primera aplicación funcional, la *variable libre* 'y' que es argumento efectivo queda *ligada*.

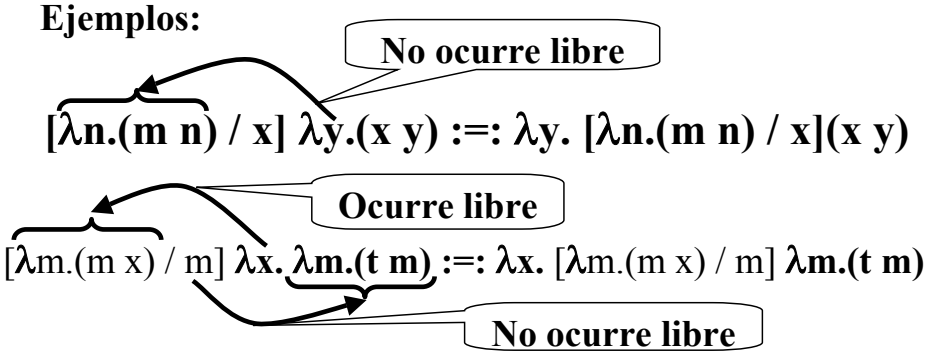


DEFINICIÓN DE SUSTITUCIÓN

En el caso en que M sea una abstracción funcional, la misma se realiza de la siguiente forma:

5 - [N/x] λy.M :=: λy. [N/x]M
si y "no ocurre libre" en N
o
si x "no ocurre libre" en M

Ejemplos:



DEFINICIÓN DE SUSTITUCIÓN

[N/x] M

En el caso en que M sea una abstracción funcional, la misma se realiza de la siguiente forma:

6 - [N/x] λy.M :=: λz. [N/x] [z/y] M

si y "ocurre libre" en N

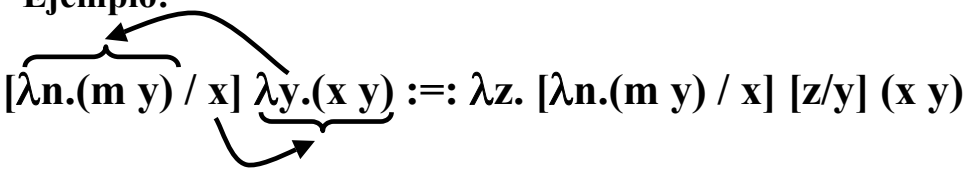
y

si x "ocurre libre" en M

y

si z "no ocurre libre" en N o M

Ejemplo:



REDUCCION DE EXPRESIONES Y FORMA NORMAL

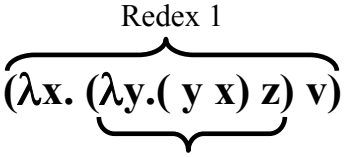
Con el enfoque funcional la evaluación de una expresión se entiende como la aplicación de una función a determinados argumentos efectivos.

En el Cálculo Lambda esto se denomina REDEX.

Definición:

Un REDEX es un término de la siguiente forma

(λx . M N)



Redex 1

M = (λy.(y x) z)
N = v

Redex 2

M =(y x)
N = z

Definición:

Una FORMA NORMAL es un término que no contiene ningún REDEX

Si puede deducirse de las reglas del cálculo que P :=: Q y si Q está en forma normal, se dice que Q ES FORMA NORMAL DE P

La forma normal de un término es única

Si un término P puede reducirse a dos términos M y N entonces existe un término T al cual M y N pueden, a su vez, reducirse

No todos los métodos de reducción garantizan encontrar la forma normal de un término

ESTRATEGIAS DE REDUCCIÓN

Estrategia de evaluación de <u>ORDEN NORMAL</u>: consiste en reducir siempre primero el *redex* de más a la izquierda (aquel cuyo λ aparece más a la izquierda)

La estrategia de evaluación de orden normal aplicada a un término que tiene forma normal termina por encontrarla en un número finito de reducciones

Estrategia de evaluación de <u>ORDEN APLICATIVO</u>: consiste en reducir primero los dos términos del redex antes que la aplicación que el denota sea reducida.

Redex 1

Orden Normal

$(\lambda x.\ (\lambda y.(y\ x)\ z)\ v)$

Orden Aplicativo

Redex 2

1. Reduzco Redex 1

$[v/x]\ (\lambda y.(y\ x)\ z) :=: (\lambda y.(y\ v)\ z)$

por reglas 3,5,3,...

2. Reduzco Redex 2

$[z/y]\ (y\ v) :=: (z\ v)$

por reglas 3,2 y 1

1. Reduzco Redex 2

$[z/y]\ (y\ x) :=: (z\ x)$

por reglas 3,2 y 1

2. Reduzco Redex 1

$(\lambda x.(\ z\ x)\ v)$

por reglas 3,2 y 1

$[v/x]\ (z\ x) :=: (z\ v)$

En este ejemplo se llega a la forma normal con ambas estrategias

Redex 1

$(\lambda u.v\ (\lambda x.((x\ x)\ y)\ \lambda x.((x\ x)\ y)))$

Redex 2

Orden Normal

1. Reduzco Redex 1

$[(\lambda x.((x\ x)\ y)\ (\lambda x.((x\ x)\ y))/u]\ v :=: v$

Orden Aplicativo

1. Reduzco Redex 2 **$(\lambda x.((x\ x)\ y)\ \lambda x.((x\ x)\ y))$** I

M N

$[\lambda x.((x\ x)\ y)/x]\ ((x\ x)\ y) :=: ((\lambda x.((x\ x)\ y)\ \lambda x.((x\ x)\ y))\ y)$ II

Redex 3

Reeemplazo II en la aplicación original

(λu.v ((λx.((x x) y) λx.((x x) y)) y)) (III)

Redex 3

reduzco

[λx.((x x) y)/x] ((x x) y):=: ((λx.((x x) y) λx.((x x) y)) y)

Reeemplazo el resultado en III

(λu.v (((λx.((x x) y) λx.((x x) y)) y) y))

Redex 4

Se puede volver a reducir y se llegará a un resultado similar al anterior.

Para este ejemplo, utilizando el orden aplicativo no es posible llegar a la forma normal. Si en cambio se puede llegar con una estrategia de orden normal