Cálculo Lambda

Alan Reyes-Figueroa
Teoría de la Computación **(Aula 27) 15.noviembre.2023**

Reglas

Funciones anónimas

Ejemplos

Cálculo Lambda

- El Cálculo λ es un sistema formal en lógica matemática para expresar el cálculo basado en la abstracción de funciones y la aplicación mediante el enlace y la sustitución de variables.
- Es un modelo universal de computación que se puede usar para simular cualquier máquina de Turing. Introducido por Alonzo Church (1930s).

Cálculo Lambda

- Consiste en construir términos lambda y realizar operaciones de reducción sobre ellos.
- Reglas:

1) **Variable**: (asignar variables) **v:=E**
2) **Abstracción** (definición de función): **λx.E**
   (E es un término lambda)
   (E = expresión)
3) **Aplicación**: $\mathbf{E_1\ E_2}$
   (aplicamos la función E_1 al argumento E_2)

Cálculo Lambda

Junto con dos reglas de reducción:

- Conversión α (*α-Conversion*):

$$(\lambda x\ .\ M[x]) \rightarrow (\lambda y\ .\ M[y])$$

cambia de nombre las variables ligadas en la expresión. Se utiliza para evitar colisiones de nombres.

- Reducción β (*β-Reduction*):

$$((\lambda x\ .\ M)\ E) \rightarrow (M[x:=E])$$

reemplazando las variables vinculadas con la expresión del argumento en el cuerpo de la abstracción.

Ejemplo: Cálculo Lambda

- El cálculo lambda puro no posee funciones built-in. Evaluamos la siguiente expresión:

  (+ (* 5 6) (* 8 3))

- Aquí, no podemos comenzar con aplicar '+' ya que éste sólo opera sobre números. Hay dos expresiones reducibles: (* 5 6) y (* 8 3).
  - **(+ (* 5 6) (* 8 3))**
  - **(+ 30 (* 8 3))**
  - **(+ 30 24)**
  - **= 54**

Ejemplo: Cálculo Lambda

Reducción β:

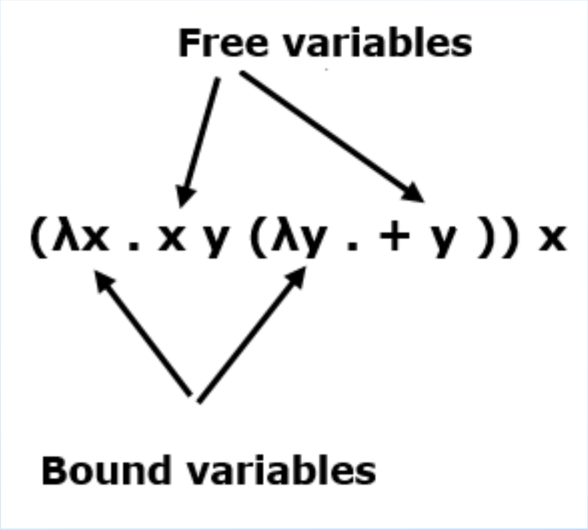
- Es necesario una regla de reducción que permita manejar los lambdas

- **f := (λx . * 2 x)** **“f(x) = 2*x”**

- **(λx . * 2 x) 4** **“f(4)”**
- **(* 2 4)**
- **= 8**

Variables libres y vinculadas

- En una expresión, cada apariencia de una variable es "libre" (a λ) o "ligada" (a λ).
- Al aplicar la reducción β a la exp. **(λx . E) y**, reemplazamos cada x que ocurre libre en E con y.



Funciones anidadas

- $r := \lambda x \,.\, \text{sqrt}\ x$ **"$r(x) = \sqrt{x}$"**
- $\text{sqrt} := \lambda x \,.\, \inf \{y\colon y \geq 0,\ y^2 \geq x\}$

- $t := (\lambda x \,.\, (+\ x\ 1))$ **"$t(x) = x + 1$"**

- **Hallar una expresión lambda para r(t(x))**
- $(\lambda x \,.\, \text{sqrt}\ (\lambda x \,.\, (+\ x\ 1))\ x)$
- $(\lambda x \,.\, \text{sqrt}\ (\lambda y \,.\, (+\ y\ 1))\ x)$

Ejemplo

- $g := (\lambda x \,.\, (-\ x\ 1))$ **"$g(x) = x - 1$"**
- $f := (\lambda x \,.\, +\ g\ x\ 3))$ **"$f(x) = g(x) + 3$"**

- $\boldsymbol{(\lambda x \,.\, +\ (\lambda x \,.\, (-\ x\ 1))\ x\ 3)\ 9}$
- $\boldsymbol{(\lambda x \,.\, +\ (\lambda y \,.\, (-\ y\ 1))\ x\ 3)\ 9}$ Reducción α
- $\boldsymbol{+\ (\lambda y \,.\ (-\ y\ 1))\ 9\ 3}$ Reducción β
- $\boldsymbol{+\ (-\ 9\ 1)\ 3}$ Reducción β
- $\boldsymbol{+\ 8\ 3}$
- $\boldsymbol{=\ 11}$

Funciones anónimas

- Funciones lambda = funciones anónimas.
- Igual que una función de Python normal (def), pero se puede definir sin un nombre.
- Las funciones anónimas se definen con la palabra clave lambda.

- Están restrictas a una sola línea de expresión.
- Pueden tomar múltiples parámetros como en las funciones regulares.

Funciones lambda en Python

- lambda parameters : expression
- Ejemplo: x = lambda a : a + 10
  x(5)

x := λa . + a 10

- Ejemplo: **s := λa . λb . + a b**
- s = lambda a, b: a+b
- s = lambda a: lambda b: a+b

Pueden aceptar funciones lambda como parámetros

¿Para qué se usan?

- Reducen el número de líneas de código en comparación con la función normal de Python definida con def (simpleza).
- Se usan cuando se necesita una función temporalmente durante un período corto de tiempo, a menudo para usarse dentro de otra función, (*e.g.* filter, map, reduce).
- Podemos definir una función y llamarla inmediatamente al final de la definición.

Diferencia con def

- squares = lambda x: x*x
- print('Using lambda: ', squares(5))

- def squares_def (x):
        return x*x
- print('Using def: ', squares_def(5))

Funciones anidadas

- my_function = lambda a : lambda b: a * b
- doubler = my_function(2)
- tripler = my_function(3)
- print(doubler(2023))
  print(tripler(2023))

Aritmética en el Cálculo λ

- Existen varias formas posibles de definir los números naturales en el cálculo lambda, pero los más comunes son los **números de Church**, definidos de la siguiente manera:

- $0 := \lambda f\ .\ \lambda x\ .\ x$
- $1 := \lambda f\ .\ \lambda x\ .\ f\ x$
- $2 := \lambda f\ .\ \lambda x\ .\ f\ (f\ x)$
- $3 := \lambda f\ .\ \lambda x\ .\ f\ (f\ (f\ x))$

Aritmética en el Cálculo λ

- 0 := λf . λx . x
- **cero = lambda f: lambda x: x**
- **"cero(f, x) = x"   (aplica 0-veces f a x)**
- 1 := λf . λx . f x
- **uno = lambda f: lambda x: f(x)**
- **"uno(f, x) = f(x)"   (aplica 1-vez f a x)**
- 2 := λf . λx . f (f x)
- **dos = lambda f: lambda x: f (f x)**
- **"dos(f, x) = f(f(x))"   (aplica 2-veces f a x)**

Aritmética en el Cálculo λ

- La función sucesor es
- **S(n, f, x) := λn . λf . λx . (f (n (f x)))**

donde

- n un número, es una función que toma una función f y devuelve otra función que toma x, y aplica f sobre x, n veces.
- El número sucesor es aplicar f al resultado de esto; es decir, aplicar f un total de n+1 veces.