Teoría de categorías y cálculo lambda

Mario Román García

11 de junio de 2018

Grado en Ingeniería Informática y Matemáticas - Universidad de Granada

Outline

Cálculo lambda: definición

El cálculo lambda es un sistema formal dado por ecuaciones sobre términos lambda. Abstracción y aplicación son nociones primitivas.

$$\mathsf{Expr} := \begin{cases} x, & \text{variables, de un conjunto numerable,} \\ \mathsf{Expr}\ \mathsf{Expr}, & \text{aplicación de funciones,} \\ \lambda x.\mathsf{Expr}, & \text{abstracción sobre una variable.} \end{cases}$$

Consideramos

- α-equivalencia, invariancia a renombramientos $(\lambda x.M[x]) \equiv (\lambda y.M[y])$;
- β-reducciones, aplicación de funciones $(\lambda x.M)\ N \longrightarrow_\beta M_{[N/x]}$;
- η-reducciones, extensionalidad $(\lambda x.f\ x) \equiv f$.

Alonzo Church. “A set of postulates for the foundation of logic”. En: Annals of mathematics (1932), págs. 346-366.

H.P. Barendregt. The lambda calculus: its syntax and semantics. Studies in logic and the foundations of mathematics. North-Holland, 1984. ISBN: 9780444867483.

Cálculo lambda: lenguaje de programación

Buscamos formas normales invariantes a reducciones. Existen términos que divergen como $\Omega = (\lambda x.(xx))(\lambda x.(xx))$.

Teorema (Church-Rosser)

La reducción es confluente.

$$\begin{tikzcd}[column sep=small] & M \arrow[dl, ->>] \arrow[dr, ->>] & \\ N \arrow[dr, dashed, ->>] & & P \arrow[dl, dashed, ->>] \\ & Z & \end{tikzcd}$$

Si existe la forma normal, es única.

Hay expresiones que se reducirán o no dependiendo del orden de evaluación, como $(\lambda x.\lambda y.y)\ \Omega\ (\lambda x.x)$.

Teorema (Evaluación a izquierda)

Si existe una forma normal del término lambda, la estrategia que reduce a cada paso la aplicación más a la izquierda la encuentra.

La reducción da una forma de cálculo Turing-completa.

Robert Pollack. "Polishing Up the Tait-Martin-Löf Proof of the Church-Rosser Theorem". En: Proc. De Wintermöte, Chalmers University (1995).

Ryo Kashima. "A Proof of the Standardization Theorem in Lambda-Calculus". En: Tokyo Institute of Technology (2000).

Mikrokosmos: implementación

Usamos Haskell para escribir un intérprete de cálculo lambda. Los algoritmos inductivos y puros se traducen directamente. Usamos la librería de combinadores monádicos `parsec`.

```haskell
-- | Substitutes an index for a lambda expression
subs :: Integer -> Exp -> Exp -> Exp
subs n p (Lambda e) = Lambda (subs (n+1) (incrementFreeVars 0 p) e)
subs n p (App f g)  = App (subs n p f) (subs n p g)
subs n p (Var m)
  | n == m    = p         -- The lambda is replaced directly
  | n <  m    = Var (m-1) -- A more exterior lambda decreases a number
  | otherwise = Var m     -- An unrelated variable remains untouched
```

Usamos índices de DeBruijn, una representación abstracta interna del ámbito de una variable. La expresión,

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

se reescribe como $\lambda\ (1\ \lambda\ (2\ 1))$.

Daan Leijen. Parsec, a fast combinator parser. Inf. téc. 35. Department of Computer Science, University of Utrecht (RUU), 2001.

N.G. de Bruijn. “Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem”. En: Indagationes Mathematicae (Proceedings) 75.5 (1972), págs. 381 -392.

Mikrokosmos: intérprete

```
mario@kosmos  ~  mikrokosmos
Welcome to the Mikrokosmos Lambda Interpreter!
Version 0.7.0. GNU General Public License Version 3.

mikro> :load std
Loading /home/mario/.mikrokosmos/logic.mkr...
Loading /home/mario/.mikrokosmos/nat.mkr...
Loading /home/mario/.mikrokosmos/basic.mkr...
Loading /home/mario/.mikrokosmos/ski.mkr...
Loading /home/mario/.mikrokosmos/datastructures.mkr...
Loading /home/mario/.mikrokosmos/fixpoint.mkr...
Loading /home/mario/.mikrokosmos/types.mkr...
Loading /home/mario/.mikrokosmos/std.mkr...
mikro> :verbose on
verbose: on
mikro> mult 3 2
(mult 3) 2
((λλλ((4 (3 2)) 1) λλ(2 (2 (2 1)))) λλ(2 (2 1)))
(λλλ((λλ(2 (2 (2 1))) (3 2)) 1) λλ(2 (2 1)))
λλ((λλ(2 (2 (2 1))) (λλ(2 (2 1)) 2)) 1)
λλ(λ((λλ(2 (2 1)) 3) ((λλ(2 (2 1)) 3) ((λλ(2 (2 1)) 3) 1))) 1)
λλ((λλ(2 (2 1)) 2) ((λλ(2 (2 1)) 2) ((λλ(2 (2 1)) 2) 1)))
λλ(λ(3 (3 1)) (λ(3 (3 1)) (λ(3 (3 1)) 1)))
λλ(2 (2 (λ(3 (3 1)) (λ(3 (3 1)) 1))))
λλ(2 (2 (2 (2 (λ(3 (3 1)) 1)))))
```

Mikrokosmos: SKI

Implementación

Interacción

Uso

Entorno de programación

Programando en el cálculo lambda sin tipos

Cálculo lambda con tipos

Argumentos diagonales

Conclusiones

¡Muchas gracias!