The λ-calculus - the smallest, most elegant, most powerful* programming language

Chris Stephenson†

May 13, 2006

† Computer Science Department,
Istanbul Bilgi University

*At least as powerful as any other

Dreamers and Hackers

A Little Bit of History

- Leibniz had a dream
- Hilbert laid a challenge
- Frege had a plan
- Russell found a contradiction
- Gödel shot everything down in flames
- Church, Kleene and Turing invented Computer Science

Who did what and when

- Hilbert asked if a formal system for maths was decidable in 1900.
- Gödel proved that no consistent formal system that extends Principia Mathematica is complete in 1931
- Church extended Gödel's result to show that no system that meets Gödel's requirements can be decided by a "computer" (the λ-calculus) in 1936.
- Turing proved the existence of the halting problem for his idea of a computer in 1936.
- Turing proved that the λ-calculus and what Church now called the "Turing Machine" were computationally equivalent in 1937.
- This led to the "Church-Turing thesis"

The syntax of the λ-calculus

$$T \rightarrow v$$

$$T \rightarrow (\lambda\, v\, T)$$

$$T \rightarrow (T\, T)$$

The syntax of the λ-calculus

$$T \to v$$

$$T \to (\lambda\, v\, T)$$

$$T \to (T\, T)$$

This definition contains less characters than …

“This page is intentionally left blank”

The semantics of the λ-calculus

β reduction is the only permitted transformation of statements in the λ-calculus.
It corresponds to the idea of applying a function to its parameters.

The semantics of the λ-calculus

The Free Variables of a term M, $FV(M)$ are defined as follows:

$$M = x \Rightarrow FV(M) = \{x\} \tag{1}$$

$$M = (M_1\ M_2) \Rightarrow FV(M) = FV(M_1) \cup FV(M_2) \tag{2}$$

$$M = (\lambda\, x\ M_1) \Rightarrow FV(M) = FV(M_1) - \{x\} \tag{3}$$

The semantics of the λ-calculus

Now we need to define β reduction itself, in terms of substitution:

$$L = (\lambda\, x\, M) \Rightarrow (L\, N) \longrightarrow_{\beta} M[x := N]$$

Where $M[x := N]$ is the result of substituting N for x in M

The semantics of the λ-calculus

Finally we need to define the trickiest of all, how to perform substitution.

$$M = x \Rightarrow M[x := N] = N$$

$$M = y, y \neq x \Rightarrow M[x := N] = y$$

$$M = (M_1\ M_2) \Rightarrow M[x := N] = (M_1[x := N]\ M_2[x := N])$$

$$M = (\lambda\ x\ M_1) \Rightarrow M[x := N] = M$$

The tricky case, the one that has tripped up many language designers, is …

$$M = (\lambda\ y\ M_1), y \neq x$$

There are two cases

$$\text{If } x \notin FV(M_1) \cup y \notin FV(N), M[x := N] = (\lambda\ y\ M_1[x := N])$$

$$\text{Otherwise} M[x := N] = (\lambda\ z\ M_1[y := z][x := N]),$$

where z is the first variable name in some sequence$\{v_0, v_1, \ldots\} \notin M_1 \notin N$

The semantics of the λ-calculus

Now let's play a little …

Lessons for today

- Church-Rosser - order independence
- Shönfinkel - partial closures. Paul Graham's fortune.
- Integers that are integers
- Simplicity plus extendability. Less language = more power
- "To much syntactic sugar leads to cancer of the semicolon"

References

A webpage
http://cs.bilgi.edu.tr/pages/courses/year_3/comp_314/

Wikipedia is good.

This talk will, if it gets past Ali Nesin, be an article in Matematik Dünyası

Barendregt has an online short guide.

H.F. Barendregt. **The Lambda Calculus, Its Syntax and Semantics** 1981.