Lambda Calculus

국민대학교 컴퓨터공학부
강 승 식

Brief history

- Achim Jung, “A short introduction to the Lambda Calculus”, available at http://www.cs.bham.ac.uk/~axj/pub/papers/lambda-calculus.pdf

Alonzo Church, 14.6.1903–11.8.1995

- λ-calculus, 1932
  - Alonzo Church(1903-1995)
  - *Functional* foundation for Mathematics
- Re-discovered by Computer Science
  - McCarthy, Strachey, Landin, and Scott in the 1960s

Expressions in λ-calculus

- Expression are written in *prefix* form
  - No infix or postfix operators

- Function and argument are simply written next to each other
  - *without brackets* around the argument

    + x 3

    * x x

    + (sin x) 4

Functions in λ-calculus

- Example function
  - f(x) = 3x

- λ-calculus
  - λx . * 3 x
  - λ says that the variable *x* is not part of an expression, but the *formal parameter* of the function declaration

Functions in λ-calculus

- In Pascal,

```
function   f(   x    :   int) :   int   begin   f :=   3 * x    end;
    |           |                        |               |
    λ           x                        .             * 3 x
```

- In Lisp,
  - (lambda (x) (* 3 x))
  - Function definition with no function name

Free and bound variables

- In a term $\lambda\, x\ .\ M$, the scope of x is M
  - x is bound in M
  - Variables that are not bound are free
- Example: $(\lambda x\ .\ (\lambda y\ .\ (x\ (z\ y))\ )\ )\ y$
  - z is free
  - The last y is free
  - The x and remaining y are bound

Function application

- Application is simply juxtaposition
- A function F that is defined by (λx . * 3 x)
    - (λx . * 3 x) 4
    - F 4

$$
\begin{array}{lcll}
M & ::= & \lambda\, x.\, M & \text{function} \\
  & \mid & M\ M & \text{function application} \\
  & \mid & x & \text{variable}
\end{array}
$$

Function of a function

- Body of a function consists of another function
- A function N that is defined by λy . (λx . * y x)
  - It returns another function, i.e., N 3 behaves like F

- Function with two arguments y and x
  - λ y . λ x . * y x
  - λ y x . * y x
- "N 3 4" is the same as "(N 3) 4"

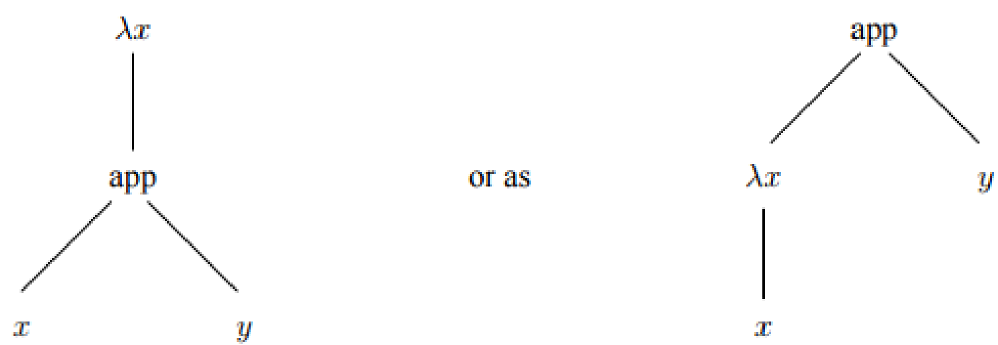
  $\lambda\, x\, y\, z.\, M$ is a shorthand for $\lambda\, x.\, (\lambda\, y.\, (\lambda\, z.\, M))$

  $M_1\, M_2\, M_3$ is a shorthand for $(M_1\, M_2)\, M_3$

.이 없으면(MMM) 좌결합 있으면 우결합

λ-term: official definition

- $M ::= c \mid x \mid M\ M \mid \lambda x\ .\ M$
  - c is any constants 1,2,3,… or arithmetic operators +,*.
  - x is any of possible variables
- The grammar is ambiguous for "λx . x y"



- It should be interpreted as "λx . (x y)" not "(λx . x) y"

엠비규어스 하지않게 다시 써줘야함

Unambiguous definition

- "λx . x y" → "λx . (x y)" or "(λx . x) y"
- The conventional interpretation can be enforced

<term>	::=	<atom> | <app> | <fun>
<atom>	::=	<head−atom> | (<app>)
<head−atom>	::=	x | c | (<fun>) 괄호 = 함수정의
<app>	::=	<head−atom> <atom> | <app> <atom>
<fun>	::=	$\lambda x.$<term>

함수정의

Reduction or β-reduction

- Textual replacement of a formal parameter by the actual parameter
  - (λx . * 3 x) 4 $\dashrightarrow_\beta$ * 3 4 　베타 리덕션 - 더이상 베타리듀스를 할수있을때까지
  - (λy . y 5) (λx . * 3 x) $\dashrightarrow_\beta$ (λx . * 3 x) 5 $\dashrightarrow_\beta$ * 3 5

- A term reaches a form where no further reductions are possible, except
  - (λx . x x) (λx . x x) → reduces to itself

자기자신으로만드면 끝나지 않음

β -reduction

The main reduction rule in the λ-calculus is *β-reduction*, which is just function application.

$$(\lambda\, x.\ M)\ N \quad \longrightarrow_\beta \quad [x \mapsto N]M$$

m에 있는 모든 x를 n으로 대체 하는것이 베타 리덕션이다.

The notation $[x \mapsto N]M$ means:

M, *with all free occurrences of x replaced by N.*

Restriction: N should not have any free variables which are bound in M.

Example:

$$(\lambda\, x.\ (\lambda\, y.\ (x\, y)))\ (\lambda\, y.\ y) \quad \longrightarrow_\beta \quad \lambda\, y.\ (\lambda y.y)\ y$$

An expression that cannot be β-reduced any further has been reduced to *normal form*.

Evaluation strategies

We have the β-rule, but if we have a complex expression, where should we apply it first?

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

Two popular strategies:

- **normal-order**: Reduce the outermost "redex" first.

$$[x \mapsto (\lambda x.\, x)(\lambda y.\, z)](\lambda y.\, y\, x\, x) \longrightarrow_\beta$$
$$\lambda y.\, y\, ((\lambda x.\, x)(\lambda y.\, z))\, ((\lambda x.\, x)(\lambda y.\, z))$$

- **applicative-order**: Arguments to a function evaluated first, from left to right.

$$(\lambda x.\, \lambda y.\, y\, x\, x)\, ([x \mapsto (\lambda y.\, z)]x) \longrightarrow_\beta (\lambda x.\, \lambda y.\, y\, x\, x)\, ((\lambda y.\, z))$$

Exercises

1. Translate the following Java expressions into λ-calculus notation:

   (a) `sin(x+3)`

   (b) `length(y)+z`

   (c) `public static int quot(double x, double n)`
   `{ return (int)(x/n); }`

(λ y x . sin y) (+ x 3)
λ x . sin (+ x 3)

(λ x y z . + x z) (length y)
(λ y z . + (length y) z)

(λ y z . + (length y) z) (2 3)
(λ z . + (length 2) z) 3
+ (length 2) 3

2. Draw the syntax trees for the following λ-terms:

   (a) $\lambda xy.x$

   (b) $\lambda xyz.xyz$

   (c) $(\lambda x.xx)(\lambda x.xx)$

3. Reduce to normal form:

   (a) $(\lambda x. + \ x\ 3)4$

   (b) $(\lambda fx.f(fx))\ (\lambda y. *\ y\ 2)\ 5$

4. Let T be the λ-term $\lambda x.xxx$. Perform some β-reductions on TT. What do you observe?

5. Let S be the term $\lambda xyz.(xz)(yz)$ and K the term $\lambda xy.x$. Reduce SKK to normal form. (Hint: This can be messy if you are not careful. Keep the abbreviations S and K around as long as you can and replace them with their corresponding λ-terms only if you need to. It becomes very easy then.)

6. Let Z be the λ-term $\lambda zx.x(zzx)$ and let Y be ZZ. By performing a few β-reductions, show that YM will be a fixpoint of M for any term M, i.e., we have $YM =_\beta M(YM)$.

7. Suppose a symbol of the λ-calculus alphabet is always 5mm wide. Write down a pure λ-term (i.e., without constants) with length less than 20cm having a normal form with length at least $10^{10^{10}}$ light-years. (A light-year is about 10^{13} kilometers.)