

MATHEMATICS FOR INFORMATION SCIENCE  
NO.6 LAMBDA CALCULUS

---

Tatsuya Hagino

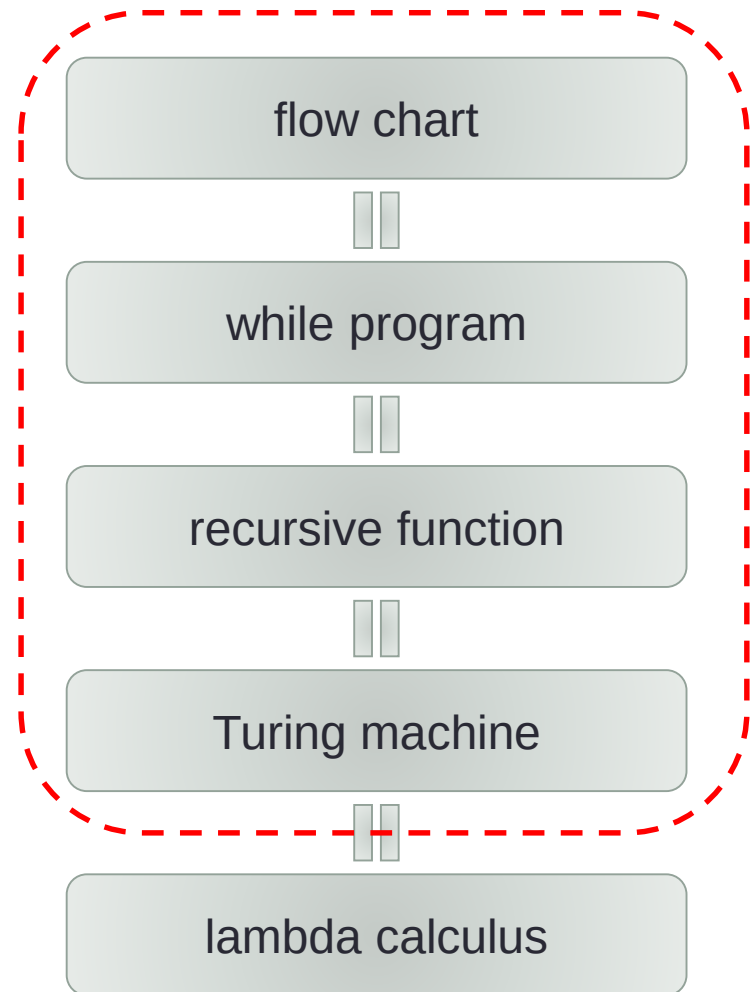
hagino@sfc.keio.ac.jp

Slides URL

<https://vu5.sfc.keio.ac.jp/slide/>

# So far

- Computation
  - flow chart program
  - while program
  - recursive function
    - primitive recursive function
    - minimization operator
  - Turing machine
- Undecidable Problems
  - Halting problem
  - Totality problem
  - Post correspondence problem



# Lambda Notation

- Function abstraction

- $f(x) = x + x \times x$

- $f = \lambda x. x + x \times x$

- Function application

- $f(2) = 2 + 2 \times 2 = 6$

- $(\lambda x. x + x \times x)(2) = 2 + 2 \times 2 = 6$

- Higher order function

- $twice(f) = \lambda x. f(f(x))$

- $twice = \lambda f. (\lambda x. f(f(x)))$

- $twice(\lambda x. x + x \times x) =$

# Currying

- Function with multiple arguments

- $f(x, y) = x \times (y + 5)$
- $f: N \times N \rightarrow N$

- Currying

- $f^*(x)(y) = x \times (y + 5)$
- $f^*(x) = \lambda y. (x \times (y + 5))$
- $f^* = \lambda x. (\lambda y. (x \times (y + 5)))$
- $f^*: N \rightarrow (N \rightarrow N)$
- $f(x, y) = f^*(x)(y)$

# Lambda Expression

- **Definition:**  $\lambda$  expressions are defined as follows:

(1) Variables  $x, y, z, x_1, x_2, y', \dots$  are  $\lambda$  expressions.

(2) For a  $\lambda$  expression  $M$  and a variable  $x$ ,

$$(\lambda x. M)$$

is a  $\lambda$  expression (*Function Abstraction*).

(3) For a  $\lambda$  expression  $M$  and  $N$ ,

$$(MN)$$

is a  $\lambda$  expression (*Function Application*)

- **Example:** Let  $x, y, f$  be variables.

- $(\lambda x. (f(f\ x)))$

- $((\lambda f. (f\ x))(\lambda y. y))$

- Abbreviated Notation

- $\lambda x_1 x_2 \cdots x_n. M \equiv (\lambda x_1. (\lambda x_2. (\cdots (\lambda x_n. M) \cdots)))$

- $M_1 M_2 M_3 \cdots M_n \equiv ((\cdots ((M_1 M_2) M_3) \cdots) M_n)$

# Bound and Free Variables

- *Bound variable*

- For  $\lambda x. M$ , a variable  $x$  in  $M$  is **bound** by  $\lambda x$ .
- $\lambda x. (yx)$
- $\lambda x. (\lambda y. xyz)$
- $(\lambda x. yx)(\lambda y. yx)$
- $\lambda xy. x(\lambda y. xy)$

- *Free variable*

- Variables which are not bound.
- $FV(x) = \{x\}$
- $FV(\lambda x. M) = FV(M) \setminus \{x\}$
- $FV(MN) = FV(M) \cup FV(N)$

- *Closed term*

- Expressions without free variables.
- $FV(M) = \emptyset$

# Alpha Conversion

- Meaning does not change by replacing bound variables.
  - $\lambda x. x \xrightarrow{\alpha} \lambda y. y$
  - $\lambda x. (y(\lambda x. yx)x) \xrightarrow{\alpha} \lambda x. (y(\lambda z. yz)x)$
- $\alpha$  conversion can be used to separate free variables and bound variables.
- **Assignment**  $M[x := N]$ 
  - Replace free variable  $x$  in  $M$  by  $N$ .
  - $((\lambda x. y x) y)[y := (\lambda z. z)] \equiv (\lambda x. (\lambda z. z)x)(\lambda z. z)$
  - do not change variable bind relationship
  - $((\lambda x. y x) y)[y := x] \not\equiv (\lambda x. x x) x$
- **Formal Definition of assignment:**
  - $x[x := N] \equiv N$
  - $y[x := N] \equiv y$
  - $(\lambda y. M)[x := N] \equiv \lambda y. (M[x := N])$   
(where  $y \notin FV(N)$ )
  - $(\lambda x. M)[x := N] \equiv \lambda x. M$
  - $(M M')[x := N] \equiv (M[x := N] M'[x := N])$
- **$\alpha$  变换:**
  - $\lambda x. M \xrightarrow{\alpha} \lambda y. (M[x := y])$

# Beta Reduction (Conversion)

- **Definition:** Replace  $\beta$  redex  $(\lambda x. M)N$  with  $M[x := N]$

$$(\lambda x. M)N \xrightarrow{\beta} M[x := N]$$

Do not bind any free variables in  $N$  when assigning. In such a case, use  $\alpha$  conversion first.

- **Example:**

- $(\lambda x. x)y \xrightarrow{\beta}$
- $(\lambda x. x y)(\lambda x. x) \xrightarrow{\beta}$
- $(\lambda x y. x y)(\lambda x. y) \xrightarrow{\beta}$

- If  $M \xrightarrow{\beta} N$ ,
  - $\lambda x. M \xrightarrow{\beta} \lambda x. N$
  - $M M' \xrightarrow{\beta} N M'$
  - $M' M \xrightarrow{\beta} M' N$



# Beta Reduction Sequence

- $P \xrightarrow{\alpha\beta} Q$ 
  - $P \xrightarrow{\alpha} Q$  or  $P \xrightarrow{\beta} Q$
- $P \xRightarrow{\alpha\beta} Q$ 
  - $P \equiv P_1 \xrightarrow{\alpha\beta} P_2 \xrightarrow{\alpha\beta} P_3 \xrightarrow{\alpha\beta} \dots \xrightarrow{\alpha\beta} P_n \equiv Q$
- $P \xleftrightarrow{\alpha\beta} Q$ 
  - $P \xrightarrow{\alpha\beta} Q$  or  $Q \xrightarrow{\alpha\beta} P$
- $P \Leftrightarrow^{\alpha\beta} Q$ 
  - $P \equiv P_1 \xleftrightarrow{\alpha\beta} P_2 \xleftrightarrow{\alpha\beta} P_3 \xleftrightarrow{\alpha\beta} \dots \xleftrightarrow{\alpha\beta} P_n \equiv Q$

# Leftmost and Rightmost Beta Reduction

- Leftmost  $\beta$  reduction
  - Do reduction on the leftmost  $\beta$  redex.
  - $(\lambda x y. y)((\lambda y. y x)(\lambda x. x)) \xrightarrow{\beta}$
- Rightmost  $\beta$  reduction
  - Do reduction on the rightmost  $\beta$  redex.
  - $(\lambda x y. y)((\lambda y. y x)(\lambda x. x)) \xrightarrow{\beta}$
- A  $\lambda$  expression is in *normal form* when there is no  $\beta$  redex.
  - $\lambda x y. x y$
  - $\lambda x. x x$
  - $x y$

# Basic Lambda Expressions

- $I \equiv \lambda x. x$ 
  - $I M \xrightarrow{\beta}$
- $K \equiv \lambda x y. x$ 
  - $K M N \xrightarrow{\beta}$
- $S \equiv \lambda x y z. x z (y z)$ 
  - $S P Q R \xrightarrow{\beta}$
  - $S(\lambda x. M)(\lambda x. N) \xrightarrow{\beta}$
  - $S K K \xrightarrow{\beta}$

# SK Expressions

- **SK Expressions**
  - $\lambda$  expressions combining variables,  $S$  and  $K$  with function application
- For any  $\lambda$  expression  $M$ , there exists an SK expression  $X$  such that  $X \xRightarrow{\alpha\beta} M$ .
  - For an SK expression  $X$  and a variable  $x$ , let define an SK expression  $\Lambda x.X$  as follows:
    - $\Lambda x.x \equiv S K K$
    - $\Lambda x.y \equiv K y$
    - $\Lambda x.S \equiv K S$
    - $\Lambda x.K \equiv K K$
    - $\Lambda x.X_1 X_2 \equiv S(\Lambda x.X_1)(\Lambda x.X_2)$
  - $\Lambda x.X \xRightarrow{\alpha\beta} \lambda x.X$
  - Replace  $\lambda$  in  $M$  with  $\Lambda$ .
- Example:
  - Find an SK expression for  $M \equiv \lambda xy.xy$
  - $\Lambda x.(\Lambda y.x y) \equiv$

# Special Lambda Expressions

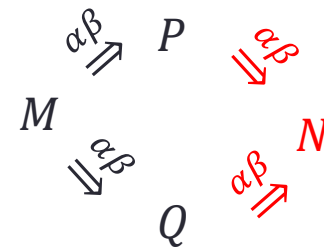
- $Z \equiv (\lambda x. x x)(\lambda x. x x)$ 
  - $(\lambda x. x x)(\lambda x. x x) \xrightarrow{\beta}$
- $Y \equiv \lambda y. (\lambda x. y(x x))(\lambda x. y(x x))$ 
  - Curry's **fixed point operator**
  - $Y M \xrightarrow{\alpha\beta} M(Y M)$
  - $Y M \xrightarrow{\beta}$
  - $Y(\lambda x. x) \xrightarrow{\beta}$
- $Y' \equiv (\lambda x y. y(x x y))(\lambda x y. y(x x y))$ 
  - Turing's **fixed point operator**
  - $Y' M \xrightarrow{\alpha\beta} M(Y' M)$

# Church-Rosser Theorem

- Theorem: If  $M \xRightarrow{\alpha\beta} P$  and  $M \xRightarrow{\alpha\beta} Q$ , then there exists  $N$  such that  $P \xRightarrow{\alpha\beta} N$  and  $Q \xRightarrow{\alpha\beta} N$ .

- The proof is difficult.

- $(\lambda x. x x)((\lambda y. y) z) \xrightarrow{\beta}$   
 $\downarrow \beta$



- This theorem assures that the normal form does not depend on the selection of  $\beta$  redex in a  $\lambda$  expression.
  - It is known the leftmost  $\beta$  reduction gives the normal form if it exists.

# Summary

- Lambda expression
- Conversion and reduction
  - $\alpha$  conversion
  - $\beta$  reduction
- SK expression
- Normal form
- Church-Rosser theorem