MATHEMATICS FOR INFORMATION SCIENCE

NO.6 LAMBDA CALCULUS

Tatsuya Hagino

hagino@sfc.keio.ac.jp

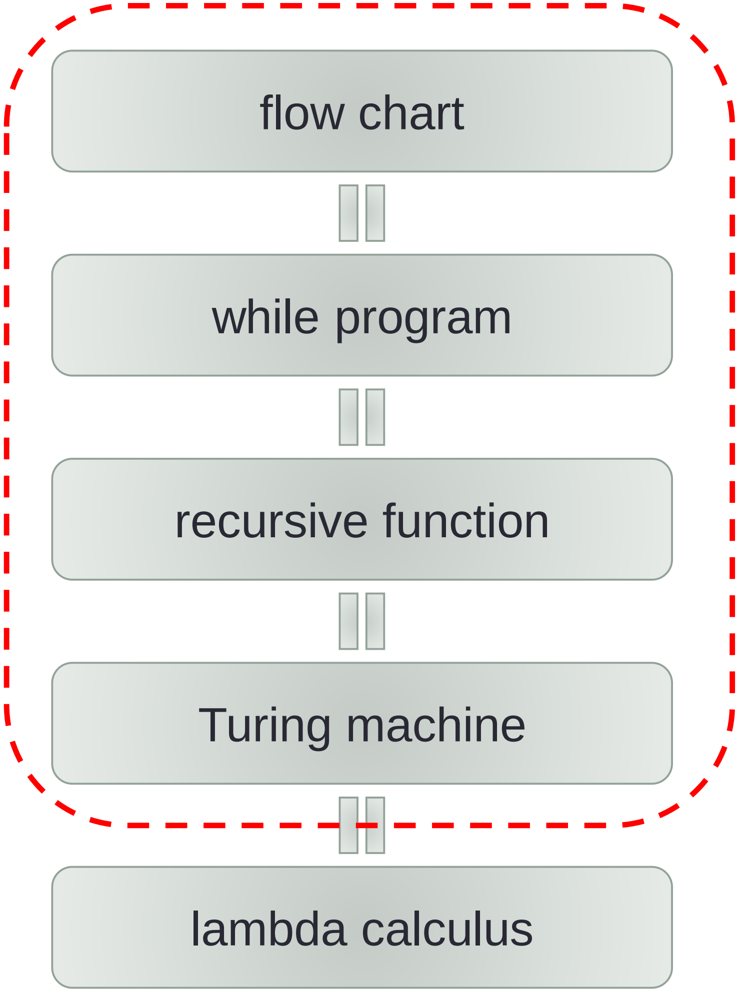
Slides URL

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

Currying

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

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

Lambda Expression

- **Definition:** λ expressions are defined as follows:
  (1) Variables $x, y, z, x_1, x_2, y', \ldots$ are λ expressions.
  (2) For a λ expression M and a variable x,
  $$(\lambda x. M)$$
  is a λ expression (*Function Abstraction*).
  (3) For a λ expression M and N,
  $$(MN)$$
  is a λ expression (*Function Application*)

- **Example:** Let x, y, f be variables.
  - $\left(\lambda x. (f (f\ x))\right)$
  - $\left((\lambda f. (f\ x))(\lambda y. y)\right)$

- Abbreviated Notation
  - $\lambda x_1 x_2 \cdots x_n. M \equiv \left(\lambda x_1. \left(\lambda x_2. (\cdots (\lambda x_n. M) \cdots)\right)\right)$
  - $M_1 M_2 M_3 \cdots M_n \equiv \left((\cdots ((M_1 M_2) M_3) \cdots) M_n\right)$

Bound and Free Variables

- *Bound variable*
  - For $\lambda x.M$, a variable x in M is bound by λx.
  - $\lambda x.(yx)$
  - $\lambda x.(\lambda y.xyz)$
  - $(\lambda x.yx)(\lambda y.yx)$
  - $\lambda xy.x(\lambda y.xy)$
- *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.
  - $\lambda x.x \xrightarrow{\alpha} \lambda y.y$
  - $\lambda x.(y(\lambda x.yx)x) \xrightarrow{\alpha} \lambda x.(y(\lambda z.yz)x)$
- α conversion can be used to separate free variables and bound variables.
- Assignment $M[x \coloneqq N]$
  - Replace free variable x in M by N.
  - $((\lambda x.y\ x)\ y)[y \coloneqq (\lambda z.z)] \equiv (\lambda x.(\lambda z.z)x)(\lambda z.z)$
  - do not change variable bind relationship
  - $((\lambda x.y\ x)\ y)[y \coloneqq x] \not\equiv (\lambda x.x\ x)\ x$
- Formal Definition of assignment:
  - $x[x \coloneqq N] \equiv N$
  - $y[x \coloneqq N] \equiv y$
  - $(\lambda y.M)[x \coloneqq N] \equiv \lambda y.(M[x \coloneqq N])$
    (where $y \notin FV(N)$)
  - $(\lambda x.M)[x \coloneqq N] \equiv \lambda x.M$
  - $(M\ M')[x \coloneqq N] \equiv (M[x \coloneqq N]\ M'[x \coloneqq N])$
- α変換:
  - $\lambda x.M \xrightarrow{\alpha} \lambda y.(M[x \coloneqq y])$

Beta Reduction (Conversion)

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

$$(\lambda x. M)N \overset{\beta}{\rightarrow} M[x \coloneqq N]$$

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

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

- **Example:**
  - $(\lambda x. x)y \overset{\beta}{\rightarrow}$
  - $(\lambda x. x\ y)(\lambda x. x) \overset{\beta}{\rightarrow}$
  - $(\lambda x\ y. x\ y)(\lambda x. y) \overset{\beta}{\rightarrow}$

Beta Reduction Sequence

- $P \xrightarrow{\alpha\beta} Q$
  - $P \xrightarrow{\alpha} Q \quad \text{or} \quad P \xrightarrow{\beta} Q$
- $P \overset{\alpha\beta}{\Longrightarrow} Q$
  - $P \equiv P_1 \xrightarrow{\alpha\beta} P_2 \xrightarrow{\alpha\beta} P_3 \xrightarrow{\alpha\beta} \cdots \xrightarrow{\alpha\beta} P_n \equiv Q$
- $P \overset{\alpha\beta}{\longleftrightarrow} Q$
  - $P \xrightarrow{\alpha\beta} Q \quad \text{or} \quad Q \xrightarrow{\alpha\beta} P$
- $P \overset{\alpha\beta}{\Longleftrightarrow} Q$
  - $P \equiv P_1 \overset{\alpha\beta}{\longleftrightarrow} P_2 \overset{\alpha\beta}{\longleftrightarrow} P_3 \overset{\alpha\beta}{\longleftrightarrow} \cdots \overset{\alpha\beta}{\longleftrightarrow} P_n \equiv Q$

Leftmost and Rightmost Beta Reduction

- Leftmost β reduction
  - Do reduction on the leftmost β redex.
  - $(\lambda x\ y.\ y)\big((\lambda y.\ y\ x)(\lambda x.\ x)\big) \xrightarrow{\beta}$
- Rightmost β reduction
  - Do reduction on the rightmost β redex.
  - $(\lambda x\ y.\ y)\big((\lambda y.\ y\ x)(\lambda x.\ x)\big) \xrightarrow{\beta}$
- A λ expression is in *normal form* when there is no β 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
  - λ expressions combining variables, *S* and *K* with function application

- For any λ expression M, there exists an SK expression X such that $X \overset{\alpha\beta}{\Rightarrow} M$.
  - For an SK expression X and a variable x, let define an SK expression $\Lambda x. X$ as follows:
    - $\Lambda x. x \equiv S\ K\ K$
    - $\Lambda x. y \equiv K\ y$
    - $\Lambda x. S \equiv K\ S$
    - $\Lambda x. K \equiv K\ K$
    - $\Lambda x. X_1\ X_2 \equiv S(\Lambda x. X_1)(\Lambda x. X_2)$
  - $\Lambda x. X \overset{\alpha\beta}{\Rightarrow} \lambda x. X$
  - Replace λ in M with Λ.

- Example:
  - Find an SK expression for $M \equiv \lambda xy. xy$
  - $\Lambda x. (\Lambda y. x\ y) \equiv$

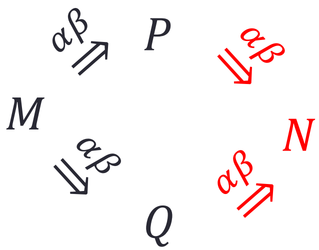
Special Lambda Expressions

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

Church-Rosser Theorem

- Theorem: If $M \overset{\alpha\beta}{\Rightarrow} P$ and $M \overset{\alpha\beta}{\Rightarrow} Q$, then there exists N such that $P \overset{\alpha\beta}{\Rightarrow} N$ and $Q \overset{\alpha\beta}{\Rightarrow} N$.
  - The proof is difficult.
  - $(\lambda x.\, x\ x)\big((\lambda y.\, y)\ z\big) \overset{\beta}{\rightarrow}$

    $\downarrow \beta$



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

Summary

- Lambda expression
- Conversion and reduction
  - α conversion
  - β reduction
- SK expression
- Normal form
- Church-Rosser theorem

Homework 6

- Convert back the following SK expression to a lambda expression.

$$S\left(S(KS)\left(S(KK)(SKK)\right)\right)\left(S(KK)(SKK)\right)$$

where

$$S \equiv \lambda x\ y\ z.\ x\ z\ (y\ z)$$

$$K \equiv \lambda x\ y.\ x$$