

Lambda calculus

Functional models of computation

Ivan Trepakov

NSU Sys.Pro

History

- 1928 — Hilbert's *Entscheidungsproblem* ¹
 - Is there an *algorithm* for deciding whether a proposition in first-order logic is true or false?
- Replacement for set theory as foundation of mathematics
 - 1930 — Combinatory logic (*Curry, Schönfinkel*)
 - 1932 — λ -calculus (*Church*)
 - 1935 — Kleene-Rosser paradox
- Effective computability
 - 1935 — Untyped λ -calculus (*Church, Kleene, Rosser*)
 - 1936 — Turing machine
 - 1936 — Church-Turing thesis
- 1936 — Undecidability of first-order logic
 - Halting problem of Turing machine
 - Equivalence of λ -terms

¹German for “decision problem”

History

- 1928 — Hilbert's *Entscheidungsproblem*¹
 - Is there an *algorithm* for deciding whether a proposition in first-order logic is true or false?
- Replacement for set theory as foundation of mathematics
 - 1930 — Combinatory logic (*Curry, Schönfinkel*)
 - 1932 — λ -calculus (*Church*)
 - 1935 — Kleene-Rosser paradox
- Effective computability
 - 1935 — Untyped λ -calculus (*Church, Kleene, Rosser*)
 - 1936 — Turing machine
 - 1936 — Church-Turing thesis
- 1936 — Undecidability of first-order logic
 - Halting problem of Turing machine
 - Equivalence of λ -terms

David Hilbert

- Haskell Curry
- Wilhelm Ackermann
- John von Neumann
- Ernst Zermelo
- ...

Alonzo Church

- Stephen Cole Kleene
- J. Barkley Rosser
- Alan Turing
- Dana Scott
- Michael O. Rabin
- ...

¹German for “decision problem”

Untyped lambda calculus

Syntax

$$term ::= \underbrace{var}_{\text{Variable}} \mid \underbrace{(term \ term)}_{\text{Application}} \mid \underbrace{(\lambda var. term)}_{\text{Abstraction}}$$

Examples

$$\lambda x. x \quad (\lambda x. xx)(\lambda y. yy) \quad \lambda f. \lambda x. f(fx)$$

Conventions

- Application is left associative
 $abc = (ab)c$
- Abstraction is right associative
 $\lambda x. \lambda y. x = \lambda x. (\lambda y. x)$
- Consecutive abstractions can be combined
 $\lambda x. \lambda y. x = \lambda xy. x$

Untyped lambda calculus

Syntax

$$term ::= \underbrace{var}_{\text{Variable}} \mid \underbrace{(term \ term)}_{\text{Application}} \mid \underbrace{(\lambda var. term)}_{\text{Abstraction}}$$

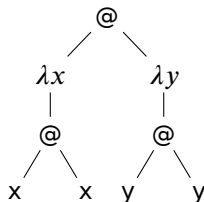
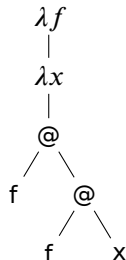
Examples

 $\lambda x. x \quad (\lambda x. xx)(\lambda y. yy) \quad \lambda f. \lambda x. f(fx)$

Conventions

- Application is left associative
 $abc = (ab)c$
- Abstraction is right associative
 $\lambda x. \lambda y. x = \lambda x. (\lambda y. x)$
- Consecutive abstractions can be combined
 $\lambda x. \lambda y. x = \lambda xy. x$

Tree representation

$$\begin{array}{c} \lambda x. x \\ | \\ \lambda x \\ | \\ x \end{array}$$
 $(\lambda x. xx)(\lambda y. yy)$  $\lambda f. \lambda x. f(fx)$ 

Free and bound variables

Free variables $FV(t)$

Variable: $FV(x) = \{x\}$

Application: $FV(MN) = FV(M) \cup FV(N)$

Abstraction: $FV(\lambda x. M) = FV(M) \setminus \{x\}$

Bound variables $BV(t)$

Variable: $BV(x) = \emptyset$

Application: $BV(MN) = BV(M) \cup BV(N)$

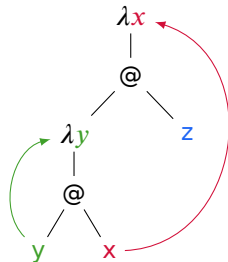
Abstraction: $BV(\lambda x. M) = BV(M) \cup \{x\}$

Closed terms

Term t is called **closed** or **combinator** if $FV(t) = \emptyset$

Example

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



Substitution

Substitution $t_{[v:=S]}$

$$x_{[v:=S]} = \begin{cases} S & v = x \\ x & v \neq x \end{cases}$$

$$(MN)_{[v:=S]} = (M_{[v:=S]} N_{[v:=S]})$$

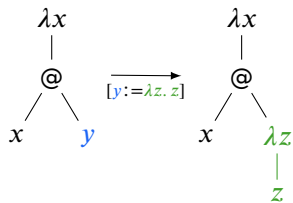
$$(\lambda x. M)_{[v:=S]} = \begin{cases} \lambda x. M & v = x \\ \lambda x. M_{[v:=S]} & v \neq x \end{cases}$$

Safe substitution

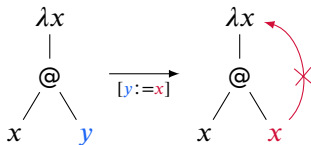
Substitution $t_{[v:=S]}$ is **safe** if $BV(t) \cap FV(S) = \emptyset$

Example

$$(\lambda x. x y)_{[y:=\lambda z. z]} = \lambda x. x(\lambda z. z)$$

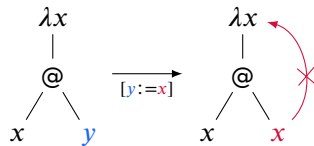


$$(\lambda x. x y)_{[y:=x]} = \lambda x. x x$$



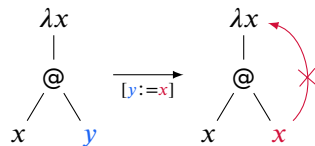
Problem

$$\lambda x. x y_{[y:=x]} = \lambda x. x x$$



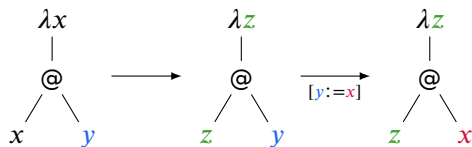
Problem

$$\lambda x. x y_{[y:=x]} = \lambda x. x x$$



Solution

$$(\lambda x. x y)_{[y:=x]} \equiv (\lambda z. z y)_{[y:=x]} = \lambda z. z x$$



Renaming

α -equivalence

$$\lambda x. M \equiv_{\alpha} \lambda y. M_{[x:=y]} \quad \text{if } x \notin FV(M)$$

$$\lambda x. M \equiv_{\alpha} \lambda x. N \quad \text{if } M \equiv_{\alpha} N$$

$$MP \equiv_{\alpha} NP \quad \text{if } M \equiv_{\alpha} N$$

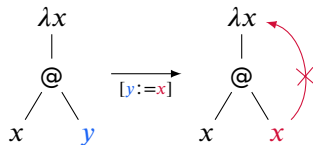
$$PM \equiv_{\alpha} PN \quad \text{if } M \equiv_{\alpha} N$$

Conventions

- λ -terms are considered **identical** up to α -equivalence
- Appropriate renaming happens **implicitly** if required during substitution

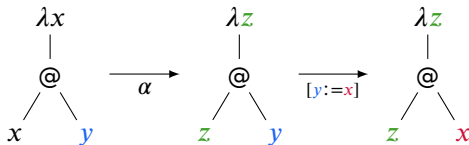
Problem

$$\lambda x. xy_{[y:=x]} = \lambda x. xx$$



Solution

$$(\lambda x. xy)_{[y:=x]} \equiv_{\alpha} (\lambda z. zy)_{[y:=x]} = \lambda z. zx$$



Definitions

- Subterm of form $(\lambda x. M)N$ is called β -redex
- Redex $(\lambda x. M)N$ can be reduced to $M_{[x:=N]}$
- $M \rightarrow_{\beta} M'$ denotes single β -reduction
- $M \twoheadrightarrow_{\beta} M'$ denotes several β -reductions
- $M \leftrightarrow_{\beta} M'$ denotes β -conversion as smallest equivalence relation containing \rightarrow_{β}

β -reduction

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

$$\lambda x. M \rightarrow_{\beta} \lambda x. N \quad \text{if } M \rightarrow_{\beta} N$$

$$MP \rightarrow_{\beta} NP \quad \text{if } M \rightarrow_{\beta} N$$

$$PM \rightarrow_{\beta} PN \quad \text{if } M \rightarrow_{\beta} N$$

Definitions

- Subterm of form $(\lambda x. M)N$ is called β -redex
- Redex $(\lambda x. M)N$ can be reduced to $M_{[x:=N]}$
- $M \rightarrow_{\beta} M'$ denotes single β -reduction
- $M \Rightarrow_{\beta} M'$ denotes several β -reductions
- $M \leftrightarrow_{\beta} M'$ denotes β -conversion as smallest equivalence relation containing \rightarrow_{β}

β -reduction

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

$$\lambda x. M \rightarrow_{\beta} \lambda x. N \quad \text{if } M \rightarrow_{\beta} N$$

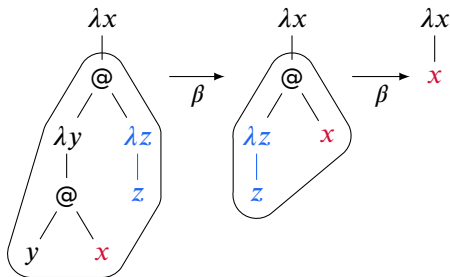
$$MP \rightarrow_{\beta} NP \quad \text{if } M \rightarrow_{\beta} N$$

$$PM \rightarrow_{\beta} PN \quad \text{if } M \rightarrow_{\beta} N$$

Example

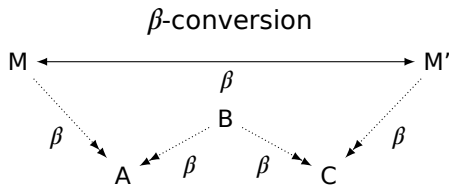
$$(\lambda x. (\lambda y. yx)(\lambda z. z)) \rightarrow_{\beta}$$

$$(\lambda x. (\lambda z. z)x) \rightarrow_{\beta} \lambda x. x$$



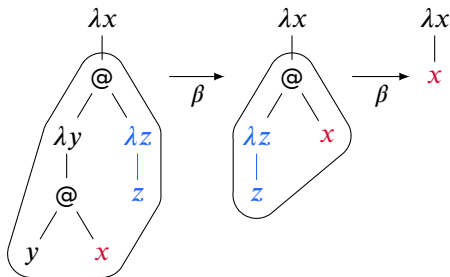
Definitions

- Subterm of form $(\lambda x. M)N$ is called **β -redex**
- Redex $(\lambda x. M)N$ can be **reduced** to $M_{[x:=N]}$
- $M \rightarrow_{\beta} M'$ denotes single **β -reduction**
- $M \Rightarrow_{\beta} M'$ denotes several β -reductions
- $M \leftrightarrow_{\beta} M'$ denotes **β -conversion** as smallest equivalence relation containing \rightarrow_{β}



Example

$$(\lambda x. (\lambda y. yx)(\lambda z. z)) \rightarrow_{\beta} (\lambda x. (\lambda z. z)x) \rightarrow_{\beta} \lambda x. x$$



Reduction order

Normal order

- Term without any redex is in β -normal form
- Reduction sequence that always reduces leftmost outermost redex is called normal order reduction

Example

$$\underline{(\lambda z. ((\lambda x. xz)f) ((\lambda y. yz)g))a} \rightarrow_{\beta}$$

$$\underline{((\lambda x. xa)f) ((\lambda y. ya)g)} \rightarrow_{\beta}$$

$$(fa) \underline{((\lambda y. ya)g)} \rightarrow_{\beta} (fa) (ga)$$

Normal order

- Term without any redex is in β -normal form
- Reduction sequence that always reduces leftmost outermost redex is called **normal order reduction**

Totality

Example

$$\underline{(\lambda z. ((\lambda x. xz) f) ((\lambda y. yz) g)) a} \rightarrow_{\beta}$$

$$\underline{((\lambda x. xa) f) ((\lambda y. ya) g)} \rightarrow_{\beta}$$

$$(fa) \underline{((\lambda y. ya) g)} \rightarrow_{\beta} (fa) (ga)$$

Reduction order

Normal order

- Term without any redex is in **β -normal form**
- Reduction sequence that always reduces leftmost outermost redex is called **normal order reduction**

Example

$$\underline{(\lambda z. ((\lambda x. xz) f) ((\lambda y. yz) g)) a} \rightarrow_{\beta}$$

$$\underline{((\lambda x. xa) f) ((\lambda y. ya) g)} \rightarrow_{\beta}$$

$$(fa) ((\lambda y. ya) g) \rightarrow_{\beta} (fa) (ga)$$

Totality

$$\Omega = \underline{(\lambda x. xx)(\lambda x. xx)} \rightarrow_{\beta}$$

$$(xx)_{[x := (\lambda x. xx)]} \rightarrow_{\beta}$$

$$\underline{(\lambda x. xx)(\lambda x. xx)} \rightarrow_{\beta} \dots$$

Reduction order

Normal order

- Term without any redex is in **β -normal form**
- Reduction sequence that always reduces leftmost outermost redex is called **normal order reduction**

Example

$$\underline{(\lambda z. ((\lambda x. xz) f) ((\lambda y. yz) g)) a} \rightarrow_{\beta}$$

$$\underline{((\lambda x. xa) f) ((\lambda y. ya) g)} \rightarrow_{\beta}$$

$$(fa) ((\lambda y. ya) g) \rightarrow_{\beta} (fa) (ga)$$

Totality

$$\Omega = \underline{(\lambda x. xx)(\lambda x. xx)} \rightarrow_{\beta}$$

$$(xx)_{[x := (\lambda x. xx)]} \rightarrow_{\beta}$$

$$\underline{(\lambda x. xx)(\lambda x. xx)} \rightarrow_{\beta} \dots$$

Ω has **no β -normal form**

Reduction order

Normal order

- Term without any redex is in **β -normal form**
- Reduction sequence that always reduces leftmost outermost redex is called **normal order reduction**

Example

$$\underline{(\lambda z. ((\lambda x. xz) f) ((\lambda y. yz) g)) a} \rightarrow_{\beta}$$

$$\underline{((\lambda x. xa) f) ((\lambda y. ya) g)} \rightarrow_{\beta}$$

$$(fa) \underline{((\lambda y. ya) g)} \rightarrow_{\beta} (fa) (ga)$$

Totality

$$\Omega = \underline{(\lambda x. xx)(\lambda x. xx)} \rightarrow_{\beta}$$

$$(xx)_{[x := (\lambda x. xx)]} \rightarrow_{\beta}$$

$$\underline{(\lambda x. xx)(\lambda x. xx)} \rightarrow_{\beta} \dots$$

Ω has **no β -normal form**

Strictness

Reduction order

Normal order

- Term without any redex is in **β -normal form**
- Reduction sequence that always reduces leftmost outermost redex is called **normal order reduction**

Example

$$\underline{(\lambda z. ((\lambda x. xz)f) ((\lambda y. yz)g))a} \rightarrow_{\beta}$$

$$\underline{((\lambda x. xa)f) ((\lambda y. ya)g)} \rightarrow_{\beta}$$

$$(fa) \underline{((\lambda y. ya)g)} \rightarrow_{\beta} (fa) (ga)$$

Totality

$$\Omega = \underline{(\lambda x. xx)(\lambda x. xx)} \rightarrow_{\beta}$$

$$(xx)_{[x:= (\lambda x. xx)]} \rightarrow_{\beta}$$

$$\underline{(\lambda x. xx)(\lambda x. xx)} \rightarrow_{\beta} \dots$$

Ω has **no β -normal form**

Strictness

$$\underline{(\lambda x. y)\Omega} \rightarrow_{\beta} y$$

Reduction order

Normal order

- Term without any redex is in **β -normal form**
- Reduction sequence that always reduces leftmost outermost redex is called **normal order reduction**

Example

$$\underline{(\lambda z. ((\lambda x. xz) f) ((\lambda y. yz) g)) a} \rightarrow_{\beta}$$

$$\underline{((\lambda x. xa) f)} \underline{((\lambda y. ya) g)} \rightarrow_{\beta}$$

$$(fa) \underline{((\lambda y. ya) g)} \rightarrow_{\beta} (fa) (ga)$$

Totality

$$\Omega = \underline{(\lambda x. xx)(\lambda x. xx)} \rightarrow_{\beta}$$

$$(xx)_{[x := (\lambda x. xx)]} \rightarrow_{\beta}$$

$$\underline{(\lambda x. xx)(\lambda x. xx)} \rightarrow_{\beta} \dots$$

Ω has **no β -normal form**

Strictness

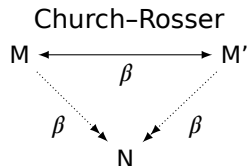
$$\underline{(\lambda x. y)\Omega} \rightarrow_{\beta} y$$

Normal order reduction models
non-strict or **lazy** evaluation

Reduction order

Church-Rosser theorem

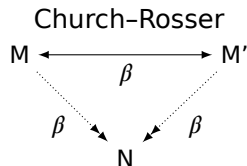
If $M \leftrightarrow_{\beta} M'$ then there exists N such that $M \twoheadrightarrow_{\beta} N$
and $M' \twoheadrightarrow_{\beta} N$



Reduction order

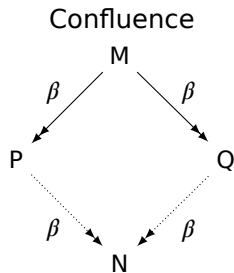
Church-Rosser theorem

If $M \leftrightarrow_{\beta} M'$ then there exists N such that $M \twoheadrightarrow_{\beta} N$ and $M' \twoheadrightarrow_{\beta} N$



Confluence property

If $M \twoheadrightarrow_{\beta} P$ and $M \twoheadrightarrow_{\beta} Q$ then there exists N such that $P \twoheadrightarrow_{\beta} N$ and $Q \twoheadrightarrow_{\beta} N$



Reduction order

Church-Rosser theorem

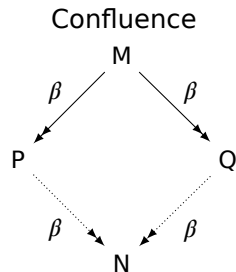
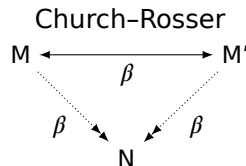
If $M \leftrightarrow_{\beta} M'$ then there exists N such that $M \twoheadrightarrow_{\beta} N$ and $M' \twoheadrightarrow_{\beta} N$

Confluence property

If $M \twoheadrightarrow_{\beta} P$ and $M \twoheadrightarrow_{\beta} Q$ then there exists N such that $P \twoheadrightarrow_{\beta} N$ and $Q \twoheadrightarrow_{\beta} N$

Corollary

β -normal form is **unique** up to α -equivalence



Reduction order

Church-Rosser theorem

If $M \leftrightarrow_{\beta} M'$ then there exists N such that $M \twoheadrightarrow_{\beta} N$ and $M' \twoheadrightarrow_{\beta} N$

Confluence property

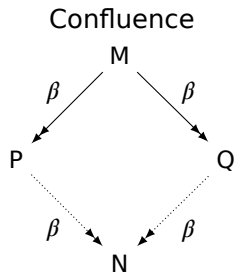
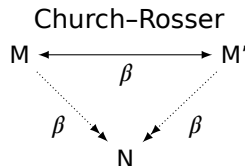
If $M \twoheadrightarrow_{\beta} P$ and $M \twoheadrightarrow_{\beta} Q$ then there exists N such that $P \twoheadrightarrow_{\beta} N$ and $Q \twoheadrightarrow_{\beta} N$

Corollary

β -normal form is **unique** up to α -equivalence

Normal order theorem

If $M \twoheadrightarrow_{\beta} N$ and N is in β -normal form then there exists **normal order reduction** sequence from M to N



Fixed-point combinator

Curry's Y -combinator

$$Y = \lambda f. (\lambda x. f(xx)) (\lambda x. f(xx))$$

Turing's Θ -combinator

$$\Theta = (\lambda xy. x(xxy)) (\lambda xy. x(xxy))$$

Undecidability

Church numerals

Relation to folds

Algebraic data types

Predecessor

η -conversion

$$(\lambda x. Mx) \underset{\eta}{\longleftrightarrow} M \text{ if } x \notin FV(M)$$

Q&A