# Rewriting and termination in lambda calculus

Silvia Ghilezan

University of Novi Sad
Mathematical Institute SASA
Serbia

Lecture 2

Oregon Programming Language Summer School
Eugene, June 2022

# Roadmap

- Rewriting basics
- Lambda calculus
- Confluence: Church-Rosser property, local confluence
- Normalisation: strong normalisation, normalisation
- Strategies: leftmost-outermost strategy, perpetual strategies
- Simple types in lambda calculus: strong normalization
- Intersection types in lambda calculus: complete characterisations of normalisations

Higher-order rewriting systems - Lambda calculus

Simply lambda calculus

# $\alpha$-conversion

The formalisation of the principal that the name of the bound variable is irrelevant

## Definition

$\alpha$-reduction:

$$\lambda x.M \longrightarrow_\alpha \lambda y.M[x := y], \quad y \notin FV(M)$$

**Proposition** $\twoheadrightarrow_\alpha$ is an equivalence relation, notation $=_\alpha$

## Proof.

Symmetry, the interesting case □

## Example

$\lambda x.fx =_\alpha \lambda y.fy$

“A rose by any other name would smell as sweet"

William Shakespeare, “Romeo and Juliet"

# $\beta$-reduction

The formalisation of function evaluation

$$(\lambda x.M)N \longrightarrow_\beta M[x := N]$$

- $M[x := N]$ represents an evaluation of the function $M$ with $N$ being the value of the parameter $x$.
- $(\lambda x.M)N$ is a redex and $M[x := N]$ is a contractum
- $\beta$-conversion is the symmetric closure of $\twoheadrightarrow_\beta$ is an equivalence (with $\alpha$-reduction), notation $\equiv_\beta$

- Barendregt's variable convention: If a term contains a free variable which would become bound after *beta*-reduction, that variable should be renamed.
- Renaming could be done also by using De Bruijn name free notation.

## Example

$$(\lambda x.x^2 + 1)5 \longrightarrow_\beta 5^2 + 1 \to 26$$

# $\eta$-conversion

The formalisation of extensionality

## Definition

$\eta$-reduction:

$$\lambda x.(Mx) \longrightarrow_\eta M, \quad x \notin FV(M)$$

- This rule identifies two functions that always produce equal results if taking equal arguments.

## Example

$$\lambda x.\mathbf{succ}x \longrightarrow_\eta \mathbf{succ}$$

$$(\lambda x.\mathbf{succ}x)2 \longrightarrow_\beta \mathbf{succ}2 \qquad \mathbf{succ}2$$

# Combinatory logic

Language

$$M ::= x \mid \mathbf{S} \mid \mathbf{K} \mid \mathbf{I} \mid MM$$

Rewrite rules

$$\begin{array}{lcl} \mathbf{K}MN & \longrightarrow & M \\ \mathbf{S}MNP & \longrightarrow & MP(NP) \\ \mathbf{I}M & \longrightarrow & M \end{array}$$

Identity $\mathbf{I} = \mathbf{SKK}$

Lambda calculus and combinatory logic

$$\mathbf{I} = \lambda x.x \quad \mathbf{K} = \lambda x.y.x \quad \mathbf{S} = \lambda x.y.z.xz(yz)$$

Translation

- from CL to lambda calculus is a unique map
- from lambda calculus to CL is not unique, depends on how abstraction is defined

# Expressiveness

In the mid 1930s

- **(Curry)** Equivalence of $\lambda$-calculus and Combinatory Logic.
- **(Kleene)** Equivalence of $\lambda$-calculus and recursive functions.
- **(Turing)** Equivalence of $\lambda$-calculus and Turing machines.

# Recall: ARS normal forms

- $n \in A$ is a normal form if there is no $s$ such that $n \longrightarrow s$
- $t \in A$ has a normal form if $t \twoheadrightarrow n$ and $n$ is a normal form, we say that $n$ is a NF of $t$

**Notation**: $\twoheadrightarrow$ will denote $\twoheadrightarrow_\beta \cup \longrightarrow_\alpha$

# Running example: 2. $\beta$-normal forms

| | |
|---|---|
| $xyzx$ | normal form NF |
| $\mathbf{I} = \lambda x.x$ | normal form NF |
| $\mathbf{K} = \lambda xy.x$ | normal form NF |
| $\mathbf{S} = \lambda xyz.xz(yz)$ | normal form NF |
| $\mathbf{KI}(\mathbf{KII})$ | strongly normalizing SN |
| $\mathbf{KI}\Omega$ | normalizing N |
| $\mathbf{Y} = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ | head normalizing HN (solvable) |
| $\Omega = \Delta\Delta = (\lambda x.xx)(\lambda x.xx)$ | unsolvable |

$\mathbf{KI}(\mathbf{KII}) \to \mathbf{KII} \to \mathbf{I}$

$\mathbf{KI}(\mathbf{KII}) \to \mathbf{I}$

$\mathbf{KI}\Omega \to \mathbf{I}$

# Properties - Confluence

## Theorem (Church-Rosser theorem)

*If $M \twoheadrightarrow N$ and $M \twoheadrightarrow P$, then there exists $S$ such that $N \twoheadrightarrow S$ and $P \twoheadrightarrow S$*

The proof is deep and involved.

## Corollary

- *If $M \twoheadrightarrow N$ and $M \twoheadrightarrow P$, then $N = P$*
- *Every lambda term has at most one normal form (uniqueness of NF)*
- *The order of the applied reductions is arbitrary and always leads to the same result*
- *Reductions can be executed in parallel (parallel computing)*

Proof. □

The general form of a lambda term

- $\lambda x_1 \ldots . x_n.((\lambda y.P)Q)Q_1 \ldots Q_k, \quad n \geq 0, \ k \geq 0$
- $\lambda x_1 \ldots . x_n.xQ_1 \ldots Q_k, \quad n \geq 0, \ k \geq 0$

Head redex (head reduction $\longrightarrow_h$ and $\twoheadrightarrow_h$)

$$(\lambda y.P)Q$$

$$\lambda x_1 \ldots . x_n.((\lambda y.P)Q)Q_1 \ldots Q_k, \quad n \geq 0$$

Internal redex is not a head redex (internal reduction $\longrightarrow_i$ and $\twoheadrightarrow_i$)

Head normal form

$$\lambda x_1 \ldots . x_n.xQ_1 \ldots Q_k, \quad n \geq 0, \ k \geq 0$$

Normal form is a HNF

$$\lambda x_1 \ldots . x_n.xQ_1 \ldots Q_k, \quad Q_i \in NF, \ i \leq k$$

# Standardization

**Example**

$$\begin{array}{llll}
\lambda x.((\lambda z.zz)(\mathbf{I}(\mathbf{I}x))) & \longrightarrow_h & \lambda x.(\mathbf{I}(\mathbf{I}x))(\mathbf{I}(\mathbf{I}x)) & \\
 & \longrightarrow_h & \lambda x.(\mathbf{I}x)(\mathbf{I}(\mathbf{I}x)) & \\
 & \longrightarrow_h & \lambda x.x(\mathbf{I}(\mathbf{I}x)) & \text{no more } \longrightarrow_h \\
 & \longrightarrow_i & \lambda x.x(\mathbf{I}x) & \\
 & \longrightarrow_i & \lambda x.xx & \text{standard } (\twoheadrightarrow_s)
\end{array}$$

$$\begin{array}{llll}
\lambda x.((\lambda z.zz)(\mathbf{I}(\mathbf{I}x))) & \twoheadrightarrow_h & \lambda x.x(\mathbf{I}(\mathbf{I}x)) & \text{no more } \longrightarrow_h \\
 & \longrightarrow_i & \lambda x.x(\mathbf{I}x) & \\
 & \longrightarrow_i & \lambda x.xx & \text{not standard}
\end{array}$$

**Theorem (Standardization)**

- *If $P \twoheadrightarrow S$ then $P \twoheadrightarrow_h M \twoheadrightarrow_i S$ for some $M$*
- *If $P \twoheadrightarrow S$ then $P \twoheadrightarrow_s S$*

# Strategies

- perpetual
- normalization
- call-by-name, call-by-value, call-by-need ...
- optimal
- ...

# Perpetual strategies

A strategy $\longrightarrow_e \subseteq \longrightarrow$ is **perpetual** if for every $M \in \Lambda$ which is not SN (has an infinite reduction), there is an infinite reduction path $M \longrightarrow_e M_1 \longrightarrow_e M_2 \longrightarrow_e \ldots$

An effective perpetual strategy is $M_i \longrightarrow_p M_{i+1}$ where $M_{i+1} = F_\infty(M_i)$

$$F_\infty(M) = \begin{cases} M & \text{if } M \text{ is a nf} \\ \text{otherwise,} & M \equiv C[R], \text{ if } R \equiv (\lambda x.P)Q \text{ is the leftmost redex} \\ \begin{cases} C[P[x := Q]] & \text{if } R \text{ is I-redex} \\ \text{otherwise} & \text{if } R \text{ is K-redex} \\ \begin{cases} C[P] & \text{if } Q \text{ is NF} \\ C[(\lambda x.P)F_\infty(Q)] & \text{if } Q \text{ is not NF} \end{cases} \end{cases} \end{cases}$$

## Example

$\mathbf{KI}\Omega \twoheadrightarrow \mathbf{I}$

$\mathbf{KI}\Omega \rightarrow \mathbf{KI}\Omega \rightarrow \ldots \rightarrow \mathbf{KI}\Omega \rightarrow \mathbf{I}$ stop

$\mathbf{KI}\Omega \rightarrow \mathbf{KI}\Omega \rightarrow \ldots \rightarrow \mathbf{KI}\Omega \rightarrow \ldots$ infinite loop

$\mathbf{KI}\Omega \rightarrow_p (\lambda y.\mathbf{I})\Omega \rightarrow_p (\lambda y.\mathbf{I})\Omega \rightarrow_p \ldots$ perpetual strategy

## Theorem

- *If $M$ is not SN then $M = M_0 \longrightarrow_p M_1 \longrightarrow_p M_2 \longrightarrow_e \ldots$, where $M_{i+1} = F_\infty(M_i)$, $i = 0, 1$*

# S combinator

$SSS(SSS)(SSS) = AAA$[1] has an infinite head reduction[2]

$A \equiv SSS$

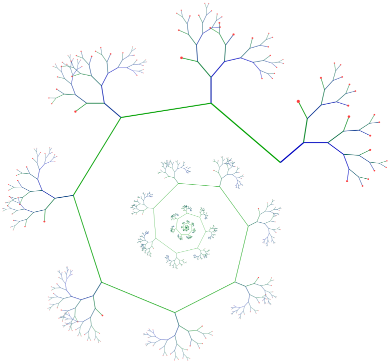


**Fig. 5.** *Limit of the infinite head reduction of AAA.*

[1] Dance of the Starlings, Henk Barendregt, Jörg Endrullis, Jan Willem Klop, and Johannes Waldmann, Raymond Smullyan on self reference, Springer, 2017

[2] Written an a wall in Barendregt's house in the 1970s

# Y combinator - Fixed point theorems

$$\begin{array}{llll} \mathbf{Y} & \equiv & \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)) & \\ & \rightarrow & \lambda f.f((\lambda x.f(xx))(\lambda x.f(xx))) & \text{HNF} \\ & \rightarrow & \lambda f.ff((\lambda x.f(xx))(\lambda x.f(xx))) & \text{HNF} \end{array}$$

- **Fixedpoint theorem: Y** is a fixed point combinator such that for all $F \in \Lambda$
$$F(\mathbf{Y}F) = \mathbf{Y}F,$$
- These properties enable the representation of the recursive functions in $\lambda$-calculus.

# Normalization strategies

A strategy $\longrightarrow_n \subseteq \longrightarrow$ is **normalizing** if for every $M \in \Lambda$ which is normalizing (has a NF $M_{nf}$) $M \longrightarrow_n M_1 \longrightarrow_n \ldots \longrightarrow_n M_{nf}$

## Definition

- A lambda term is in the *normal form* if it does not contain any redex:

$$\lambda x_1 x_2 ... x_m . y N_1 ... N_k, \quad N_i \in NF \quad m, k \geq 0$$

An effective normalising strategy is the leftmost reduction

$$C[(\lambda x.P)Q] \longrightarrow_l C[P[x := Q]], \text{ where} (\lambda x.P)Q \text{ is the most left redex in } C$$

# Normalization theorem

**Theorem**

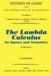
*M is normalizing (has a NF $M_{nf}$), then $M \twoheadrightarrow_l M_{nf}$*

**Proof.**

$$\begin{array}{lcll} M \text{ has NF } M_{nf} & \Rightarrow & M \twoheadrightarrow M_{nf} & \\ & \Rightarrow & M \twoheadrightarrow_s M_{nf} & \text{by the Standardization theorem} \\ & \Rightarrow & M \twoheadrightarrow_h P \twoheadrightarrow_i M_{nf} & \end{array}$$

Head reductions are leftmost reductions, and the internal reductions of the standard reduction are ordered from left to right. □

# References


H.P. Barendregt 

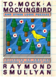

Lambda Calculus: Its syntax and Semantics.

*North Holland, 1984.*

F. Cardone, J. R. Hindley

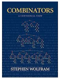
History of Lambda-calculus and Combinatory Logic

Handbook of the History of Logic. Volume 5. Logic from Russell to Church Elsevier, 2009, pp. 723-817 (*online 2006*)

R. Smullyan. 


To Mock a Mockingbird.

*Alfred A. Knopf, New York, 1985.*

S. Wolfram 


Combinators: A Centennial View

2021

Higher-order rewriting systems - Lambda calculus

## Simply lambda calculus

# Motivation

- “Disadvantages" of the untyped $\lambda$-calculus:
  - infinite computation - there exist $\lambda$-terms without a normal form
  - meaningless applications - it is allowed to create terms like **sin log**
- Types are syntactical objects that can be assigned to $\lambda$-terms.
  - Reasoning with types present in the early work of Church on untyped lambda calculus.
- two typing paradigms:
  - *à la* Curry - implicit type assignment (*lambda calculus with types*);
  - *à la* Church - explicit type assignment (*typed lambda calculus*).

# Simply typed $\lambda$-calculus - syntax of types

## Definition

- The alphabet consists of
  - $\mathrm{V} = \{\alpha, \beta, \gamma, \alpha_1, \ldots\}$, a countable set of type variables
  - $\rightarrow$, a type forming operator
  - ), ( auxiliary symbols
- The language is the set of types **T** defined as follows
  - If $\alpha \in \mathrm{V}$ then $\alpha \in$ **T**
  - If $\sigma, \tau \in$ **T** then $(\sigma \rightarrow \tau) \in$ **T**.
- The abstract grammar that generates the language

$$\sigma \ ::= \ \alpha \mid \sigma \rightarrow \sigma$$

Conventions for minimizing the number of the parentheses:

- $\sigma_1 \rightarrow \sigma_2 \rightarrow \sigma_3$ stands for $(\sigma_1 \rightarrow (\sigma_2 \rightarrow \sigma_3))$

$$M : \sigma$$

Definition

- **Type assignment** is an expression of the form $M : \sigma$, where $M$ is a $\lambda$-term and $\sigma$ is a type
- **Declaration** $x : \sigma$ is a type assignment in which the term is a variable
- **Basis (environment)** $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ is a set of declarations in which all term variables are different

# $\lambda \rightarrow$ - the type system

- Axiom

$(Ax)$ $$\frac{}{\Gamma, x : \sigma \vdash x : \sigma}$$

- Rules

$(\rightarrow_{elim})$ $$\frac{\Gamma \vdash M : \sigma \rightarrow \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

$(\rightarrow_{intr})$ $$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \rightarrow \tau}$$

# Running example: 3. types

| $M$ | Type |
|---|---|
| $xyz$ | $x : \sigma \to \tau \to \rho, y : \sigma, z : \tau \vdash xyz : \rho$ |
| $\lambda x.zx$ | $z : \sigma \to \rho \vdash \lambda x.zx : \sigma \to \rho$ |
| $\mathbf{I} = \lambda x.x$ | $\sigma \to \sigma$ |
| $\mathbf{K} = \lambda xy.x$ | $\sigma \to \rho \to \sigma$ |
| $\mathbf{S} = \lambda xyz.xz(yz)$ | $\sigma \to \rho \to \tau \to (\sigma \to \tau) \to (\sigma \to \rho)$ |
| $\Delta = \lambda x.xx$ | NO |
| $\mathbf{Y} = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ | NO |
| $\Omega = \Delta\Delta = (\lambda x.xx)(\lambda x.xx)$ | NO |

Typability (type inference): given $M$

$$M : ?$$

Inhabitation: given $\sigma$

$$? : \sigma$$

Type checking: given $M$ and $\sigma$

$$(M : \sigma)?$$

# The logical meaning of $\to$

Intuitionistic logic - Natural deduction, Gentzen 1930s

- Axiom

$(Ax)$ $$\frac{}{\Gamma, \sigma \vdash \sigma}$$

- Rules

$(\to_{elim})$ $$\frac{\Gamma \vdash \sigma \to \tau \qquad \Gamma \vdash \sigma}{\Gamma \vdash \tau}$$

$(\to_{intr})$ $$\frac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \to \tau}$$

# Inhabitation

Intuitionistic logic - Natural deduction, Gentzen 1930s

- Axiom

$(Ax)$ $$\frac{}{\Gamma, x : \sigma \vdash x : \sigma}$$

- Rules

$(\to_{elim})$ $$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

$(\to_{intr})$ $$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}$$

# Curry-Howard correspondence

## Intuitionistic logic vs computation

$$\vdash \sigma \quad \Leftrightarrow \vdash M : \sigma$$

A formula is provable in *LI* if and only if it is inhabited in $\lambda \rightarrow$.

- 1950s Curry
- 1968 (1980) Howard formulae-as-types
- 1970s Lambek - CCC Cartesian Closed Categories
- 1970s de Bruijn AUTOMATH

| | | |
|---|---|---|
| formulae | –as– | types |
| proofs | – as – | terms |
| proofs | –as– | programs |
| proof normalisation | –as– | term reduction |

- BHK - Brouwer, Heyting, Kolmogorov interpretation of logical connectives is formalized by the Curry-Howard correspondence

- **Subject reduction, type preservation under reduction**

  If $M \longrightarrow P$ and $M : \sigma$, then $P : \sigma$.

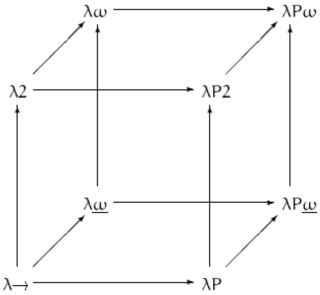
  - Broader context: evaluation of terms (expressions, programs, processes) does not cause the type change.
  - type soundness
  - type safety = progress and preservation

- **Strong normalization**

  If $M : \sigma$, then $M$ is strongly normalizing.

  - Tait 1967
  - reducibility method (reducibility candidates, logical relations)
  - arithmetic proofs

# Lambda cube

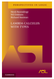


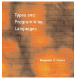

### Theorem

*M is typable* $\Longrightarrow$ *M strongly normalizing.*

- More type systems: intersection types, recursive types

# References


H.P. Barendregt, W. Dekkers, R. Statman. 
Lambda Calculus with Types.
*Cambridge University Press 2013.*

B. C. Pierece. 

Types and programming languages.
*MIT Press 2002.*

# There is no perfect world

In $\lambda \rightarrow$

$$\lambda x.xx : \textit{NO}$$