Lambda Calculus

Andrés Sicard-Ramírez

Universidad EAFIT

Semester 2010-2

Introduction

Bibliography

- Textbook: Hindley, J. Roger and Seldin, Jonathan P. [2008]. Lambda-Calculus and Combinators. An Introduction. Cambridge University Press.
- Barendregt, Henk and Barendsen, Erik [2000]. Introduction to Lambda Calculus. Revisited edition.
- Barendregt, H. P. [1981] [2004]. The Lambda Calculus. Its Syntax and Semantics. Revised edition, 6th impression. Vol. 103. Studies in Logic and the Foundations of Mathematics. Elsevier.
- Paulson, Lawrence C. [2000]. Foundations of Functional Programming. Lecture notes. URL: http://www.cl.cam.ac.uk/~lp15/ [visited on 10/06/2020].

Lambda Calculus

What is the Lambda Calculus?

Invented by Alonzo Church (around 1930s).

- The goal was to use it in the foundation of mathematics. Intended for studying functions and recursion.
- Computability model.
- Model of untyped functional programming languages.

Introduction

- λ-calculus is a collection of several formal systems
- λ-notation
  - Anonymous functions
  - Currying

Introduction

Definition (λ-terms)

The set of λ-terms is inductively defined by

$$
\begin{aligned}
v \in V \Rightarrow v \in \lambda\text{-terms} & \quad \text{(atom)} \\
c \in C \Rightarrow c \in \lambda\text{-terms} & \quad \text{(atom)} \\
M, N \in \lambda\text{-terms} \Rightarrow (MN) \in \lambda\text{-terms} & \quad \text{(application)} \\
M \in \lambda\text{-terms}, x \in V \Rightarrow (\lambda x.M) \in \lambda\text{-terms} & \quad \text{(abstraction)}
\end{aligned}
$$

where V/C is a set of variables/constants.

Introduction

Conventions and syntactic sugar

- $M \equiv N$ means the syntactic identity
- Application associates to the left
  $M N_1 N_2 \dots N_k$ means $(\dots((M N_1) N_2)\dots N_k)$
- Application has higher precedence
  $\lambda x.PQ$ means $(\lambda x.(PQ))$
- $\lambda x_1 x_2 \dots x_n.M$ means $(\lambda x_1.(\lambda x_2.(\dots(\lambda x_n.M)\dots)))$

Example

$(\lambda xyz.xz(yz))uvw \equiv ((((\lambda x.(\lambda y.(\lambda z.((xz)(yz)))))u)v)w)$.

Term-Structure and Substitution

Substitution ($[N/x]M$)

The result of substituting N for every free occurrence of x in M, and changing bound variables to avoid clashes.

$$
\begin{aligned}
[N/x]x &\equiv N; && && (1)\\
[N/x]a &\equiv a, && \text{for all atoms } a \not\equiv x; && (2)\\
[N/x](PQ) &\equiv ([N/x]P)([N/x]Q); && && (3)\\
[N/x](\lambda x.P) &\equiv \lambda x.P; && && (4)\\
[N/x](\lambda y.P) &\equiv \lambda y.P, && y \not\equiv x, x \notin \mathrm{FV}(P); && (5)\\
[N/x](\lambda y.P) &\equiv \lambda y.[N/x]P, && y \not\equiv x, x \in \mathrm{FV}(P), y \notin \mathrm{FV}(N); && (6)\\
[N/x](\lambda y.P) &\equiv \lambda z.[N/x][z/y]P, && y \not\equiv x, x \in \mathrm{FV}(P), y \in \mathrm{FV}(N); && (7)
\end{aligned}
$$

where in the last equation, z is chosen to be a variable $\notin \mathrm{FV}(NP)$.

Term-Structure and Substitution

Example

$[(\lambda y.vy)/x](y(\lambda v.xv)) \equiv y(\lambda z.(\lambda y.vy)z)$ (with $z \not\equiv v, y, x$).

Term-Structure and Substitution

α-conversion or changed of bound variables

Replace $\lambda x.M$ by $\lambda y.[y/x]M$ ($y \notin \mathrm{FV}(M)$).

α-congruence ($P \equiv_\alpha Q$)

P is changed to Q by a finite (perhaps empty) series of α-conversions.

Example

Whiteboard.

Theorem

The relation $\equiv_\alpha$ is an equivalence relation.

Beta-Reduction

β-contraction ($\cdot \rhd_{1\beta} \cdot$)

$(\lambda x.M)N$: β-redex

$[N/x]M$: contractum

$(\lambda x.M)N \rhd_{1\beta} [N/x]M$

$P \rhd_{1\beta} Q$: Replace an occurrence of $(\lambda x.M)N$ in P by $[N/x]M$.

Example

Whiteboard.

Beta-Reduction

β-reduction $(P \triangleright_\beta Q)$

P is changed to Q by a finite (perhaps empty) series of β-contractions and α-conversions.

Example

$(\lambda x.(\lambda y.yx)z)v \triangleright_\beta zv$.

Beta-Reduction

β-normal form

A term which contains no β-redex.

β-nf: The set of all β-normal forms.

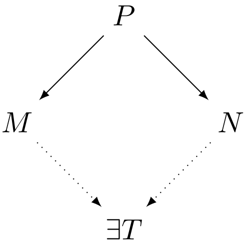
Example

Whiteboard.

Beta-Reduction

Theorem (The Church-Rosser theorem for $\triangleright_\beta$ (the diamond property))

$$\frac{P \triangleright_\beta M \qquad P \triangleright_\beta N}{\exists T. M \triangleright_\beta T \wedge N \triangleright_\beta T}$$



Corollary

If P has a β-normal form, it is unique modulo $\equiv_\alpha$; that is, if P has β-normal forms M and N, then $M \equiv_\alpha N$.

Proof

Whiteboard.

Beta-Equality

β-equality or β-convertibility ($P =_\beta Q$)

Exist $P_0, \ldots, P_n$ such that

- $P_0 \equiv P$
- $P_n \equiv Q$
- $(\forall i \leq n-1)(P_i \rhd_{1\beta} P_{i+1} \quad \vee \quad P_{i+1} \rhd_{1\beta} P_i \quad \vee \quad P_i \equiv_\alpha P_{i+1})$

Theorem (Church-Rosser theorem for $=_\beta$)

$$\frac{P =_\beta Q}{\exists T. P \rhd_\beta T \wedge Q \rhd_\beta T}$$

Proof

Whiteboard.

Beta-Equality

Corollary

If $P, Q \in \beta$-nf and $P =_\beta Q$, then $P \equiv_\alpha Q$.

Corollary

The relation $=_\beta$ is non-trivial (not all terms are β-convertible to each other).

Proof

Whiteboard.

Fixed-Point Combinators

Idea

For every term F there is a term X such

$$FX =_\beta X.$$

The term X is a **fixed-point** of F.

Fixed-Point Combinators

Theorem

$\forall F \exists X . FX =_\beta X$.

Fixed-Point Combinators

Theorem

$\forall F \exists X . FX =_\beta X$.

Proof.

Let $W \equiv \lambda x.F(xx)$, and let $X \equiv WW$. Then

$$\begin{aligned} X &\equiv (\lambda x.F(xx))W \\ &=_\beta F(WW) \\ &\equiv FX \end{aligned}$$

∎

Fixed-Point Combinators

Fixed-point combinator

A fixed-point combinator is any combinator Y such that $\mathsf{Y}F =_\beta F(\mathsf{Y}F)$, for all terms F.

Theorem (Turing)

The term $\mathsf{Y} \equiv UU$, where $U \equiv \lambda ux.x(uux)$ is a fixed-point combinator.

Proof

Whiteboard.

Theorem (Curry and Rosenbloom)

The term $\mathsf{Y} \equiv \lambda f.VV$, where $V \equiv \lambda x.f(xx)$ is a fixed-point combinator.

Proof

Whiteboard.

Fixed-Point Combinators

Corollary

For every term Z and $n \geq 0$, the equation

$$xy_1 \ldots y_n = Z$$

can be solved for x. That is, there is a term X such that

$$Xy_1 \ldots y_n =_\beta [X/x]Z.$$

Proof

$X \equiv \mathsf{Y}(\lambda xy_1 \ldots y_n.Z)$ (whiteboard).

Leftmost Reduction

Idea

Proving that a given term has no normal form.

Definition

A **contraction** in X is an order triple $\langle X, R, Y \rangle$ where R is an redex in X and Y is the result of contracting R in X.

Notation

A contraction $\langle X, R, Y \rangle$ is denoted by $X \triangleright_R Y$.

Leftmost Reduction

Example

Two contractions in $(\lambda x.(\lambda y.yx)z)v$.

(i) $(\lambda x.(\lambda y.yx)z)v \triangleright_R (\lambda y.yv)z,$ where $R \equiv (\lambda x.(\lambda y.yx)z)v$.

(ii) $(\lambda x.(\lambda y.yx)z)v \triangleright_R (\lambda x.zx)v,$ where $R \equiv (\lambda y.yx)z$.

Leftmost Reduction

Definition

A **reduction** ρ is a finite or infinite sequence of contractions separated by α-conversions

$$X_1 \triangleright_{R_1} Y_1 \equiv_\alpha X_2 \triangleright_{R_2} \dots$$

Question

Given an initial term X, there is some way of choosing a reduction that will terminate if X has a normal form?

Leftmost Reduction

Definition

A redex is **outermost** (or **maximal**) iff it is not contained in any other redex.

Definition

A (outermost) redex is the **leftmost outermost redex** (or **leftmost maximal redex**) iff it is the leftmost of the outermost redexes.

Definition

A reduction has **maximal length** iff either it is infinite or its last term contains no redexes.

Leftmost Reduction

Definition

The **leftmost reduction** (or **normal reduction**) of a term X_1 is a reduction

$$X_1 \triangleright_{R_1} X_2 \triangleright_{R_2} X_3 \triangleright_{R_3} \dots$$

where

(i) Every R_i is the leftmost outermost redex of X_i.

(ii) The reduction has maximal length.

Leftmost Reduction

Example

The leftmost reduction for $(\lambda y.a)\Omega$, where $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$.

$$(\lambda y.a)\Omega \triangleright_\beta a.$$

Leftmost Reduction

Example

The leftmost reduction for $X(YZ)$, where $X \equiv \lambda x.xx$, $Y \equiv \lambda y.yy$ and $Z \equiv \lambda z.zz$.

$$\begin{aligned}\underline{X(YZ)} &\rhd_\beta (\underline{YZ})(YZ) \\ &\rhd_\beta (\underline{ZZ})(YZ) \\ &\vdots\end{aligned}$$

Leftmost Reduction

Theorem (Standardization theorem (or leftmost reduction theorem))

If a term X has a normal form X^*, then the leftmost reduction of X is finite and ends at X^*.

Lambda Calculus and Inconsistencies

Lambda Calculus and Inconsistencies

Paradoxes

- Curry's paradox (λ-calculus + logic)
- Rusell's paradox (λ-calculus + set theory)

Curry's Paradox

Introduction

Informally, Curry's paradox is obtained in a deductive theory formed by λ-calculus + logic formulated by Church [1932, 1933].

Notation

In our presentation of Curry paradox equality means β-equality, that is, $A = B := A =_\beta B$.

Theorem (Curry's paradox)

Any proposition is probable in Church's theory

Curry's Paradox

Proof (Rosser [1984, p. 340])

Suppose we have two familiar logical principles:

$$\vdash P \supset P \tag{8}$$

$$\vdash (P \supset (P \supset Q)) \supset (P \supset Q) \tag{9}$$

together with modus ponens (if P and $P \supset Q$, then Q).

Let A be an arbitrary proposition. We construct a X such that

$$\vdash X = X \supset A \tag{10}$$

To do this, we take $F = \lambda x.x \supset A$ in the fixed-point theorem. By (8), we get

$$\vdash X \supset X.$$

Continued on next slide

Curry's Paradox

Proof (continuation).

Applying (10) to the second Φ gives

$$\vdash X \supset (X \supset A).$$

By (9) and modus ponens, we get

$$\vdash X \supset A.$$

By (10) reversed, we get

$$\vdash X.$$

By modus ponens and the last two formulas, we get

$$\vdash A.$$

∎

Curry's Paradox

Church's theory

Adding to the set of λ-terms a constant $\supset$, the sub-theory from Church's theory required for proving Curry's paradox is defined by the following inference rules [Barendregt 2014], where Γ is a set of λ-terms:

$$\frac{}{\Gamma, A \vdash A}\,\text{hyp} \quad (\text{if } A \in \Gamma)$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\,{\supset}\text{I} \qquad \frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}\,{\supset}\text{E}$$

$$\frac{\Gamma \vdash A \qquad A = B}{\Gamma \vdash B}\,\text{subst}$$

Curry's Paradox

Proof (Barendregt [2014])

Using the previous inference rules, we prove Curry's paradox. Let A be an arbitrary proposition and let $X = X \supset A$ by the fixed-point theorem.

Initially, we prove $\vdash X \supset A$.

$$\dfrac{\dfrac{\dfrac{X \vdash X \qquad X = X \supset A}{X \vdash X \supset A}\,\text{subst} \qquad X \vdash X}{X \vdash A}\,\supset\text{E}}{\vdash X \supset A}\,\supset\text{I}$$

And then we prove $\vdash A$.

$$\dfrac{\vdash X \supset A \qquad \dfrac{\vdash X \supset A \qquad X \supset A = X}{\vdash X}\,\text{subst}}{\vdash A}\,\supset\text{E}$$

Rusell's Paradox

See [Paulson 2000, § 4.6].

Encoding Data in the Lambda Calculus

Encoding Data in the Lambda Calculus

Remark

From [Paulson 2000, Ch. 3].

Booleans

$$\begin{aligned} \mathsf{true} &\equiv \lambda xy.x \\ \mathsf{false} &\equiv \lambda xy.y \\ \mathsf{if} &\equiv \lambda pxy.pxy \end{aligned}$$

where

$$\begin{aligned} \mathsf{if}\,\mathsf{true}\,M\ N &=_\beta M \\ \mathsf{if}\,\mathsf{false}\,M\ N &=_\beta N \end{aligned}$$

Encoding Data in the Lambda Calculus

Ordered pairs

$$\mathsf{pair} \equiv \lambda xyf.fxy$$
$$\mathsf{fst} \equiv \lambda p.p\,\mathsf{true}$$
$$\mathsf{snd} = \lambda p.p\,\mathsf{false}$$

where

$$\mathsf{fst}\,(\mathsf{pair}\,M\;N) =_\beta M$$
$$\mathsf{snd}\,(\mathsf{pair}\,M\;N) =_\beta N$$

Encoding Data in the Lambda Calculus

Natural numbers

Notation:

$$X^n Y \equiv \underbrace{X(X(\ldots(X}_{n\ \text{`}X\text{'s}} Y)\ldots)) \quad \text{if } n \geq 1,$$

$$X^0 Y \equiv Y.$$

The Church numerals:

$$\overline{n} \equiv \lambda f x. f^n x$$

Encoding Data in the Lambda Calculus

Some operations

$$\begin{aligned} \mathsf{add} &\equiv \lambda mnfx.mf(nfx) \\ \mathsf{mult} &\equiv \lambda mnfx.m(nf)x \\ \mathsf{isZero} &\equiv \lambda n.n(\lambda x.\mathsf{false})\ \mathsf{true} \end{aligned}$$

where

$$\begin{aligned} \mathsf{add}\,\overline{m}\,\overline{n} &=_\beta \overline{m+n} \\ \mathsf{mult}\,\overline{m}\,\overline{n} &=_\beta \overline{m \times n} \end{aligned}$$

$$\begin{aligned} \mathsf{isZero}\,\overline{0} &=_\beta \mathsf{true} \\ \mathsf{isZero}\,\overline{n+1} &=_\beta \mathsf{false} \end{aligned}$$

Recursion Using Fixed-Points

Example

Let Y be a fixed-point combinator. An informally example using the factorial function [Peyton Jones 1987].

$$\text{fac} \equiv \lambda n.\text{if}\, n = 0 \,\text{then}\, 1 \,\text{else}\, n * \text{fac}\,(n-1)$$

$$\text{fac} \equiv \lambda n.(\ldots \text{fac} \ldots)$$

$$\text{fac} \equiv (\lambda f n.(\ldots f \ldots))\, \text{fac}$$

$$h \equiv \lambda f n.(\ldots f \ldots) \quad \text{-- not recursive!}$$

$$\text{fac} \equiv h\, \text{fac} \quad \text{-- fac is a fixed-point of } h\text{!}$$

$$\text{fac} \equiv \mathsf{Y}\, h$$

Recursion Using Fixed-Points

Example (cont.)

$$
\begin{aligned}
\mathsf{fac}\,1 &\equiv \mathsf{Y}\,h\,1 \\
&=_\beta h(\mathsf{Y}\,h)\,1 \\
&\equiv (\lambda f n.(\ldots f \ldots))(\mathsf{Y}\,h)\,1 \\
&\rhd_\beta \mathsf{if}\,1 = 0\,\mathsf{then}\,1\,\mathsf{else}\,1 * (\mathsf{Y}\,h\,0) \\
&\rhd_\beta 1 * (\mathsf{Y}\,h\,0) \\
&=_\beta 1 * (h(\mathsf{Y}\,h)\,0) \\
&\equiv 1 * ((\lambda f n.(\ldots f \ldots))(\mathsf{Y}\,h)0) \\
&\rhd_\beta 1 * (\mathsf{if}\,0 = 0\,\mathsf{then}\,1\,\mathsf{else}\,1 * (\mathsf{Y}\,h\,(-1))) \\
&\rhd_\beta 1 * 1 \\
&\rhd_\beta 1
\end{aligned}
$$

Representing the Computable Functions

Representability

Let φ be a partial function $\varphi : \mathbb{N}^n \to \mathbb{N}$. A term X represents φ iff

$$\varphi(m_1, \dots, m_n) = p \Rightarrow X\overline{m_1} \dots \overline{m_n} =_\beta \overline{p},$$
$$\varphi(m_1, \dots, m_n) \text{ does not exits} \Rightarrow X\overline{m_1} \dots \overline{m_n} \text{ has no nf.}$$

Example

The successor function $\mathrm{succ}(n) = n + 1$ is represented by

$$\mathsf{succ} \equiv \lambda n f x.f(n f x)$$

Theorem (Representation of Turing-computable functions)

In λ-calculus every Turing-computable function can be represented by a combinator.

Undecidability

Gödel numbering

$$\begin{aligned} \# : \lambda\text{-terms} &\to \mathbb{N} \\ \#x_i &= 2^i \\ \#(\lambda x_i.M) &= 3^i 5^{\#M} \\ \#(MN) &= 7^{\#M} 11^{\#N} \end{aligned}$$

Notation: $\ulcorner M \urcorner = \overline{\#M}$

Theorem (Double fixed-point theorem)

$\forall F \exists X. F \ulcorner X \urcorner =_\beta X.$

Proof

Whiteboard.

Undecidability

Theorem (Rice's theorem for the λ-calculus)

Let $A \subset \lambda$-terms such as A is non-trivial (i.e. $A \neq \emptyset$, $A \neq \lambda$-terms). Suppose that A is closed under $=_\beta$ (i.e. $M \in A, M =_\beta N \Rightarrow N \in A$). Then A is no recursive, that is, $\#A = \{\#M \mid M \in A\}$ is not recursive.

Proof

Whiteboard (see [Barendregt (1990) 1992]).

Theorem

The set $\mathrm{NF} = \{M \mid M \text{ has a normal form}\}$ is not recursive.

Proof.

The set NF is not trivial and it is closed under $=_\beta$. ∎

ISWIM

ISWIM: Lambda Calculus as a Programming Language

- ISWIM: If you See What I Mean
- Landin [1966]

ISWIM Features

Remark

This section is from [Paulson 2000, Ch. 3].

ISWIM Features

Remark

This section is from [Paulson 2000, Ch. 3].

Simple declaration

$$\mathsf{let}\, x = M \,\mathsf{in}\, N \quad \equiv \quad (\lambda x.N)M$$

Example

- $\mathsf{let}\, n = \overline{0} \,\mathsf{in}\, \mathsf{succ}\, n$
- $\mathsf{let}\, m = \overline{0} \,\mathsf{in}\, (\mathsf{let}\, n = \overline{1} \,\mathsf{in}\, \mathsf{add}\, m\, n)$

ISWIM Features

Function declaration

$\textsf{let}\ f x_1 \ldots x_k = M\ \textsf{in}\ N \quad \equiv \quad (\lambda f.N)(\lambda x_1 \ldots x_k.M)$

Example

$\textsf{let succ}\ n = \lambda f x.f(n f x)\ \textsf{in succ}\ \overline{0}$

ISWIM Features

Recursive declaration

$$\textsf{letrec}\ f x_1 \ldots x_k = M\ \textsf{in}\ N \quad \equiv \quad (\lambda f.N)(\textsf{Y}(\lambda f x_1 \ldots x_k.M))$$

Example

$$\textsf{letrec}\ \mathrm{fac}\ n = \textsf{if}\ (n == 0)\, 1\, (n * \mathrm{fac}(n-1))\ \textsf{in}\ \mathrm{fac}\ 0$$

ISWIM Features

Pairs

(M, N) : pair constructor

fst, snd : projections

$\text{let}\ \lambda(x, y).E \quad \equiv \quad \lambda z.(\lambda xy.E)(\text{fst}\ z)(\text{snd}\ z)$

Example

$\text{let}\ (x, y) = (\overline{2}, \overline{3})\ \text{in add}\ x\ y$

Formal Theories

The Formal Theory $\lambda\beta$ of β-Equality

Formulas

$M = N$, where $M, N \in \lambda$-terms.

Axiom-schemes

$$(\alpha) \quad \lambda x.M = \lambda y.[y/x]M \quad \text{if } y \in \mathrm{FV}(M),$$
$$(\beta) \quad (\lambda x.M)N = [N/x]M,$$
$$(\rho) \quad M = M.$$

The Formal Theory $\lambda\beta$ of β-Equality

Rules of inference

$$\frac{M = M'}{NM = NM'}\,(\mu) \qquad \frac{M = M'}{\lambda x.M = \lambda x.M'}\,(\xi) \qquad \frac{M = N}{N = M}\,(\sigma)$$

$$\frac{M = M'}{MN = M'N}\,(\nu) \qquad \frac{M = N \qquad N = P}{M = P}\,(\tau)$$

The Formal Theory $\lambda\beta$ of β-Equality

Notation

If there is a deduction of B from the assumptions $A_1, \ldots, A_n$ in $\lambda\beta$ is denoted by

$$\lambda\beta, A_1, \ldots, A_n \vdash B.$$

Notation

If the formula B is a theorem in $\lambda\beta$ is denoted by

$$\lambda\beta \vdash B.$$

Remark

$\lambda\beta$ is an equational theory and it is a logic-free theory (there are not logical constants in its formulae).

The Formal Theory $\lambda\beta$ of β-Equality

Example

Let M and N be two closed terms, then $\lambda\beta \vdash (\lambda xy.x)MN = M$.

$$\dfrac{\dfrac{(\lambda x.(\lambda y.x))M = [M/x]\lambda y.x \equiv \lambda y.M}{(\lambda x.(\lambda y.x))MN = (\lambda y.M)N}\,(\nu) \qquad (\lambda y.M)N = [N/y]M \equiv M}{(\lambda x.(\lambda y.x))MN = M}\,(\tau)$$

The Formal Theory $\lambda\beta$ of β-Equality

Theorem

$$M =_\beta N \Leftrightarrow \lambda\beta \vdash M = N.$$

The Formal Theory $\lambda\beta$ of β-Reduction

Similar to the formal theory of β-equality, but:

(i) Formulas: $M \triangleright_\beta N$.

(ii) To change '$=$' by '$\triangleright_\beta$'.

(iii) Remove the rule (σ).

Theorem

$$M \triangleright_\beta N \Leftrightarrow \lambda\beta \vdash M \triangleright_\beta N.$$

Remark

Formal theories for combinatory logic.

Remark

$\lambda\beta$ is not a first-order theory.

References

References

Barendregt, H. P. [1981] (2004). The Lambda Calculus. Its Syntax and Semantics. Revised edition, 6th impression. Vol. 103. Studies in Logic and the Foundations of Mathematics. Elsevier (cit. on p. 3).

Barendregt, Henk [1990] (1992). Functional Programming and Lambda Calculus. In: Handbook of Theoretical Computer Science. Volume B. Formal Models and Semantics. Ed. by van Leeuwen, J. Second impression. MIT Press. Chap. 7. DOI: 10.1016/B978-0-444-88074-1.50012-3 (cit. on p. 48).

— (2014). The Impact of the Lambda Calculus. (Slides). URL: http://www.cs.ru.nl/~henk/CT271014.pdf (visited on 12/06/2019) (cit. on pp. 36, 37).

Barendregt, Henk and Barendsen, Erik (2000). Introduction to Lambda Calculus. Revisited edition (cit. on p. 3).

Church, Alonzo (1932). A Set of Postulates for the Foundation of Logic. Annals of Mathematics 33.2, pp. 346–366. DOI: 10.2307/1968337 (cit. on p. 33).

— (1933). A Set of Postulates for the Foundation of Logic (Second Paper). Annals of Mathematics 34.4, pp. 839–864. DOI: 10.2307/1968702 (cit. on p. 33).

References

Hindley, J. Roger and Seldin, Jonathan P. (2008). Lambda-Calculus and Combinators. An Introduction. Cambridge University Press (cit. on p. 3).

Landin, P. J. (1966). The Next 700 Programming Languages. Communications of the ACM 9.3, pp. 157–166. DOI: 10.1145/365230.365257 (cit. on p. 50).

Paulson, Lawrence C. (2000). Foundations of Functional Programming. Lecture notes. URL: http://www.cl.cam.ac.uk/~lp15/ (visited on 10/06/2020) (cit. on pp. 3, 38, 40, 51, 52).

Peyton Jones, Simon L. (1987). The Implementation of Functional Programming Languages. Series in Computer Sciences. Prentice-Hall International (cit. on p. 44).

Rosser, J. Barkley (1984). Highlights of the History of Lambda-Calculus. Annals of the History of Computing 6.4, pp. 337–349. DOI: 10.1109/MAHC.1984.10040 (cit. on p. 34).