Lambda Calculus

Andrés Sicard-Ramírez

Universidad EAFIT

Semester 2020-1

Administrative Information

Course web page

http://www1.eafit.edu.co/asr/courses/lambda-calculus/

Exams, programming labs, course's repository, etc.

See course web page.

Textbook

Barendregt, H. P. [1984] [2004]. The Lambda Calculus. Its Syntax and Semantics. Revised edition, 6th impression. Vol. 103. Studies in Logic and the Foundations of Mathematics. Elsevier.

Conventions

The numbers assigned to examples, exercises, figures, pages, propositions and theorems correspond to the numbers in the textbook.

What is the Lambda Calculus?

Alonzo Church (1903 – 1995)*

*Figures sources: History of computers, Wikipedia and MacTutor History of Mathematics.

What is the Lambda Calculus?

From p 3:

'The lambda calculus is a type free theory about functions as rules, rather than as graphs. "Functions as rules" is the old fashioned notion of function and refers to the process of going from argument to value, a process coded by a definition.'

'The lambda calculus regards functions again as rules in order to stress their computational aspects.'

'The functions as rules are considered in full generality.'

'The objects of study are at the same time function and argument.'

'In particular a function can be applied to itself. For the usual notion of function in mathematics (as in Zermelo-Fraenkel set theory), this is impossible.'

What is the Lambda Calculus?

Three aspects of the lambda calculs

- Foundations of mathematics
- Computations
- Pure lambda calculus

See § 1.1.

Primitive Operations: Application and Abstraction

Application

Application of the function M to argument N is denoted by MN (juxtaposition).

Primitive Operations: Application and Abstraction

Application

Application of the function M to argument N is denoted by MN (juxtaposition).

Abstraction

'If M is any formula containing the variable x, then $\lambda x[M]$ is a symbol for the function whose values are those given by the formula.' [Church 1932, p. 352]

Currying

Currying

‘Adopting a device due to Schönfinkel, we treat a function of two variables as a function of one variable whose values are functions of one variable, and a function of three or more variables similarly.’ [Church 1932, p. 352]

Such device is called **currying** after Haskell Curry.

(continued on next slide)

Currying

Currying (continuation)

Let $g : X \times Y \to Z$ be a function of two variables. We can define two functions f_x and f:

$$f_x : Y \to Z \qquad\qquad f : X \to (Y \to Z)$$
$$f_x = \lambda y.g\,(x, y), \qquad\qquad f = \lambda x.f_x.$$

Then

$$(f\;x)\;y = f_x\;y = g\,(x, y).$$

That is, the function of two variables

$$g : X \times Y \to Z$$

is represented as the higher-order function

$$f : X \to (Y \to Z).$$

Conversion

Introduction

- ‘The principal object of study in the λ-calculus is the set of λ-terms modulo convertibility.’ (p. 22).
- The relation of convertibility is a relation of equivalence on λ-terms.
- The relation of convertibility will be generated from a formal theory called the $\boldsymbol{\lambda}$ theory.

The $\boldsymbol{\lambda}$ Theory

Terms and formulae

The terms of the $\boldsymbol{\lambda}$ theory are the λ-terms and its formulae are $M = N$, where M, N are λ-terms.

The $\boldsymbol{\lambda}$ Theory

Terms and formulae

The terms of the $\boldsymbol{\lambda}$ theory are the λ-terms and its formulae are $M = N$, where M, N are λ-terms.

Remark

Our textbook [Barendregt 2004] formalised in the $\boldsymbol{\lambda}$ theory a binary relation using the symbol for equality '$=$' maybe following [Curry and Feys 1958, § 3.D.3]. Church used the infix name 'conv' instead (see, e.g. [Church 1951]).

The λ Theory

Axioms and inference rules

β-conversion

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

Equality axiom and rules

$$\frac{}{M = M} \qquad \frac{M = N}{N = M} \qquad \frac{M = N \quad N = L}{M = L}$$

Compatibility rules

$$\frac{M = N}{ML = NL} \qquad \frac{M = N}{LM = LN} \qquad \frac{M = N}{\lambda x.M = \lambda x.N} \text{ rule } \xi$$

The $\boldsymbol{\lambda}$ Theory

Notation

If $M = N$ is a theorem in $\boldsymbol{\lambda}$ it is denoted by $\boldsymbol{\lambda} \vdash M = N$. We shall also use the notation $M = N$.

The $\boldsymbol{\lambda}$ Theory

Notation

If $M = N$ is a theorem in $\boldsymbol{\lambda}$ it is denoted by $\boldsymbol{\lambda} \vdash M = N$. We shall also use the notation $M = N$.

Definition

Two λ-terms M and N are **convertible** iff $\boldsymbol{\lambda} \vdash M = N$.

The $\boldsymbol{\lambda}$ Theory

Notation

If $M = N$ is a theorem in $\boldsymbol{\lambda}$ it is denoted by $\boldsymbol{\lambda} \vdash M = N$. We shall also use the notation $M = N$.

Definition

Two λ-terms M and N are **convertible** iff $\boldsymbol{\lambda} \vdash M = N$.

Remark

The $\boldsymbol{\lambda}$ theory is

- an equational theory,
- logic-free, i.e. there are not logical constants in its formulae.

The $\boldsymbol{\lambda}$ Theory

Theorem (fixed-point theorem)

$\forall F \, \exists X \;\; FX = X.$

Combinators

Theorem (Some combinators)

$$
\begin{aligned}
\mathsf{B} &\equiv \lambda fgx.f(gx) & \mathsf{B}MNL &= M(NL) & &\text{(composition)}\\
\mathsf{B}' &\equiv \lambda fgx.g(fx) & \mathsf{B}'MNL &= N(ML) & &\text{(reversed composition)}\\
\mathsf{I} &\equiv \lambda x.x & \mathsf{I}M &= M & &\text{(identity)}\\
\mathsf{K} &\equiv \lambda xy.x & \mathsf{K}MN &= M & &\text{(projection)}\\
\mathsf{K}_* &\equiv \lambda xy.y & \mathsf{K}_*MN &= N & &\text{(projection)}\\
\mathsf{S} &\equiv \lambda fgx.fx(gx) & \mathsf{S}MNL &= ML(NL) & &\text{(stronger composition)}\\
\mathsf{W} &\equiv \lambda fx.fxx & \mathsf{W}MN &= MNN & &\text{(doubling)}
\end{aligned}
$$

Lambda Terms are "Black Boxes"

Theorem

i)

$$\neg\exists F\,\forall M\,\forall N\ F(MN) = M, \quad \text{(Exercise 2.4.6)}$$
$$\neg\exists F\,\forall M\,\forall N\ F(MN) = N.$$

Lambda Terms are "Black Boxes"

Theorem

i)

$$\neg\exists F\,\forall M\,\forall N\; F(MN) = M, \quad \text{(Exercise 2.4.6)}$$
$$\neg\exists F\,\forall M\,\forall N\; F(MN) = N.$$

ii) There is no λ-term F such that for all $M \in \Lambda$,

$$FM = \begin{cases} 1, & \text{if } M \text{ is a variable;} \\ 2, & \text{if } M \text{ is an application;} \\ 3, & \text{if } M \text{ is a } \lambda\text{-abstraction.} \end{cases}$$

(continued on next slide)

Lambda Terms are "Black Boxes"

Theorem (continuation)

iii) There is no λ-term F such that for all $M \in \Lambda$,

$$FM = \begin{cases} \text{true}, & \text{if } M \text{ is in } \beta\text{-normal form;} \\ \text{false}, & \text{otherwise;} \end{cases}$$

Lambda Terms are "Black Boxes"

Theorem (continuation)

iii) There is no λ-term F such that for all $M \in \Lambda$,

$$FM = \begin{cases} \text{true}, & \text{if } M \text{ is in } \beta\text{-normal form;} \\ \text{false}, & \text{otherwise;} \end{cases}$$

iv) There is no λ-term F such that for all $M \in \Lambda$

$$FM = \mathsf{n},$$

where n is the number of λ-abstractions in M.

Exercises

- Exercises 2.4.1–2.4.4, except 2.4.1 (iv).
- Proposition 2.1.19 and Exercises 2.4.5 and 2.4.15.
- Exercises 2.4.6, 2.4.7 and 2.4.9.
- Parts ii), iii) and iv) (at least one) from λ-terms are "black boxes", Exercises 2.4.10 (i) and 2.4.10 (ii).

Reduction

The Binary Relation β

Definition

The binary relation β on Λ is defined by

$$\beta = \{\, ((\lambda x.M)N, M[x := N]) \mid M, N \in \Lambda \,\}.$$

Compatible Relations

Definition

A binary relation $\boldsymbol{R}$ on Λ is **compatible** iff (Definition 3.1.1 (i))

$$\begin{aligned}(M, N) \in \boldsymbol{R} \Rightarrow & (LM, LN) \in \boldsymbol{R}, \\ & (ML, NL) \in \boldsymbol{R}, \\ & \text{and } (\lambda x.M, \lambda x.N) \in \boldsymbol{R}.\end{aligned}$$

One Step Beta Reduction

Definition

The binary relation **one step β-reduction** on Λ, denoted by $\rightarrow_\beta$, is the compatible closure of β.

The $\rightarrow_\beta$ relation can be inductively defined by (Definition 3.1.5):

$$\frac{(M, N) \in \beta}{M \rightarrow_\beta N}$$

$$\frac{M \rightarrow_\beta N}{LM \rightarrow_\beta LN} \qquad \frac{M \rightarrow_\beta N}{ML \rightarrow_\beta NL} \qquad \frac{M \rightarrow_\beta N}{\lambda x.M \rightarrow_\beta \lambda x.N}$$

Beta Reduction

Definition

The binary relation β**-reduction** on Λ, denoted by $\twoheadrightarrow_\beta$, is the reflexive and transitive closure of $\rightarrow_\beta$.

The $\twoheadrightarrow_\beta$ relation can be inductively defined by (Definition 3.1.5):

$$\frac{M \rightarrow_\beta N}{M \twoheadrightarrow_\beta N}$$

$$\frac{}{M \twoheadrightarrow_\beta M} \qquad \frac{M \twoheadrightarrow_\beta N \qquad N \twoheadrightarrow_\beta L}{M \twoheadrightarrow_\beta L}$$

Beta Equality or Beta Convertibility

Definition

The binary relation β-**equality** (or β-**convertibility**) on Λ, denoted by $=_\beta$, is the equivalence relation generated by $\twoheadrightarrow_\beta$.

The $=_\beta$ relation can be inductively defined by (Definition 3.1.5):

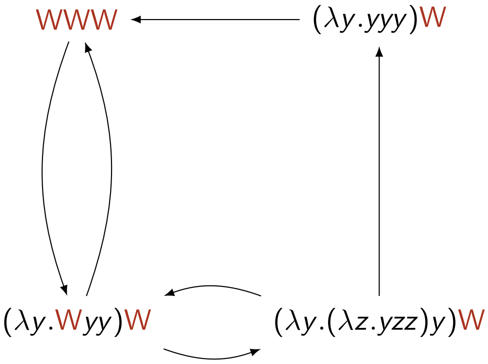
$$\frac{M \twoheadrightarrow_\beta N}{M =_\beta N}$$

$$\frac{M =_\beta N}{N =_\beta M} \qquad \frac{M =_\beta N \qquad N =_\beta L}{M =_\beta L}$$

Reduccion Graphs

Example (3.1.21 (iv))

$G_\beta(\mathsf{WWW})$ where $\mathsf{W} \equiv \lambda xy.xyy$.



Beta Reduction and The $\boldsymbol{\lambda}$ Theory

Theorem (Proposition 3.2.1)

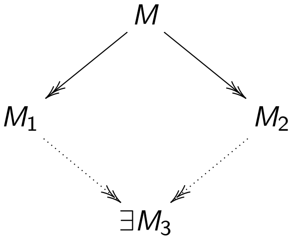
$$M =_\beta N \Leftrightarrow \boldsymbol{\lambda} \vdash M = N.$$

Church-Rosser Theorem for Beta Reduction

Theorem (Theorem 3.2.8 (i))

Church-Rosser theorem (the diamond property) for $\twoheadrightarrow_\beta$:

$$\frac{M \twoheadrightarrow_\beta M_1 \qquad M \twoheadrightarrow_\beta M_2}{\exists M_3\ (M_1 \twoheadrightarrow_\beta M_3 \wedge M_2 \twoheadrightarrow_\beta M_3)}$$



Exercises

- Exercises 3.5.1 (i)–(iii), Exercises 3.5.2 (i)–(iii) and Proposition 3.2.1 ($\Rightarrow$).
- Exercise 3.5.7, Fact 3.1.23 (ii) ($\Rightarrow$) and Exercise 3.5.9.

Classical Lamba Calculus

Introduction

Definition

A **numeric function** (or **number-theoretic function**) is a function

$$f : \mathbb{N}^p \to \mathbb{N}, \text{ for some } p \in \mathbb{N}.$$

Introduction

Example

Numeric functions.

$$
\begin{aligned}
\mathrm{Z}(n) &= 0 && \text{(\textbf{zero} function)} \\
\mathrm{S}^{+}(n) &= n + 1 && \text{(\textbf{successor} function)} \\
\mathrm{U}_i^p(n_1, \ldots, n_p) &= n_i, \quad 0 < i \leq p && \text{(\textbf{projection} functions)} \\
\mathrm{Id}(n) &= n && \text{(\textbf{identity} function)} \\
\mathrm{C}_k^p(n_1, \ldots, n_p) &= k && \text{(\textbf{constant} functions)} \\
m + n & && \text{(\textbf{addition} function)} \\
m \cdot n & && \text{(\textbf{multiplication} function)} \\
m^n & && \text{(\textbf{exponentiation} function)} \\
n! & && \text{(\textbf{factorial} function)}
\end{aligned}
$$

Introduction

Example

Numeric functions.

$$\mathrm{Pred}(n) = \begin{cases} 0, & \text{if } n = 0; \\ n - 1, & \text{otherwise}; \end{cases}$$

(**predecessor** function)

$$m \mathbin{\dot{-}} n = \begin{cases} m - n, & \text{if } m \geq n; \\ 0, & \text{otherwise}; \end{cases}$$

(**truncated subtraction** function)

$$|m - n| = \begin{cases} m \mathbin{\dot{-}} n, & \text{if } m \geq n; \\ n \mathbin{\dot{-}} m, & \text{otherwise}; \end{cases}$$

(**absolute difference** function)

Introduction

Example

Numeric functions.

$$\mathrm{Sg}(n) = \begin{cases} 0, & \text{if } n = 0; \\ 1, & \text{otherwise}; \end{cases}$$

(**signum** function)

$$\overline{\mathrm{Sg}}(n) = \begin{cases} 1, & \text{if } n = 0; \\ 0, & \text{otherwise}. \end{cases}$$

(**inverse signum** function)

$$\begin{aligned} \mathrm{Ack}(0, n) &= n + 1 \\ \mathrm{Ack}(m + 1, 0) &= \mathrm{Ack}(m, 1) \\ \mathrm{Ack}(m + 1, n + 1) &= \mathrm{Ack}(m, \mathrm{Ack}(m + 1, n)) \end{aligned}$$

(**Ackermann** function)

Introduction

Theorem

The following sets of functions are coextensive:

i) the numeric functions λ-definables,

ii) the numeric functions computable by a Turing machine and

iii) the recursive functions.

Primitive Recursive Functions

Definition

The **initial functions** are the functions

$$\mathrm{Z}(n) = 0 \qquad \text{(\textbf{zero} function)}$$

$$\mathrm{S}^{+}(n) = n + 1 \qquad \text{(\textbf{successor} function)}$$

$$\mathrm{U}_i^p(n_1, \ldots, n_p) = n_i, \quad 0 < i \leq p \qquad \text{(\textbf{projection} functions)}$$

Primitive Recursive Functions

Definition

Let $\mathfrak{C}$ be a class of numeric functions. The class $\mathfrak{C}$ is **closed under composition** iff

i) $g : \mathbb{N}^m \to \mathbb{N} \in \mathfrak{C}$ and

ii) $h_1, \ldots, h_m : \mathbb{N}^n \to \mathbb{N} \in \mathfrak{C}$,

imply

$$f(\vec{n}) = g(h_1(\vec{n}), \ldots, h_m(\vec{n})) \in \mathfrak{C}.$$

Primitive Recursive Functions

Definition

Let $\mathfrak{C}$ be a class of numeric functions. The class $\mathfrak{C}$ is **closed under primitive recursion** iff

i) $g : \mathbb{N}^n \to \mathbb{N} \in \mathfrak{C}$ and

ii) $h : \mathbb{N}^{n+2} \to \mathbb{N} \in \mathfrak{C}$,

imply

$$\begin{aligned} f &: \mathbb{N}^{n+1} \to \mathbb{N} \in \mathfrak{C} \\ f(0, \vec{n}) &= g(\vec{n}), \\ f(k+1, \vec{n}) &= h(f(k, \vec{n}), k, \vec{n}), \end{aligned}$$

Primitive Recursive Functions

Definition

The class $\mathfrak{PR}$ of **primitive recursive functions** is the smallest class of numeric functions including the initial functions and closed under composition and primitive recursion.

Primitive Recursive Functions

Definition

The class $\mathfrak{PR}$ of **primitive recursive functions** is the smallest class of numeric functions including the initial functions and closed under composition and primitive recursion.

Remark

Some textbooks which introduced the $\mathfrak{PR}$ are [Boolos, Burges and Jeffrey 2007; Davis 1982; Gómez Marín and Sicard Ramírez 2002; Kleene 1974; Mendelson 2015].

Primitive Recursive Functions

Example

All the numeric functions in the previous examples except the Ackerman functions are primitive recursive functions.

Recursive Functions

Definition

Let $\mathfrak{C}$ be a class of numeric functions. The class $\mathfrak{C}$ is **closed under minimalisation** iff

i) $g : \mathbb{N}^{n+1} \to \mathbb{N} \in \mathfrak{C}$ and

ii) $(\forall \vec{n})(\exists m)(g(\vec{n}, m) = 0)$,

imply

$$f : \mathbb{N}^n \to \mathbb{N}$$
$$f(\vec{n}) = \mu m \, [g(\vec{n}, m) = 0], \in \mathfrak{C}.$$

Recursive Functions

Definition

The class $\mathfrak{R}$ of **(total) recursive functions** is the smallest class of numeric functions including the initial functions and closed under composition, primitive recursion and minimalisation.

Recursive Functions

Definition

The class $\mathfrak{R}$ of **(total) recursive functions** is the smallest class of numeric functions including the initial functions and closed under composition, primitive recursion and minimalisation.

Theorem

The Ackermann function is a recursive primitive function.

Exercises

- Exercise 6.8.6.

References

Barendregt, H. P. [1984] (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 pp. 2, 12, 13).

Boolos, George S., Burges, John P. and Jeffrey, Richard C. [1974] (2007). Computability and Logic. 5th ed. Cambridge University Press (cit. on pp. 44, 45).

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 pp. 6–8).

— [1941] (1951). The Calculi of Lambda-Conversion. Second printing. Princenton University Press (cit. on pp. 12, 13).

Curry, Haskell B. and Feys, Robert (1958). Combinatory Logic. Vol. I. North-Holland Publishing Company (cit. on pp. 12, 13).

Davis, Martin (1982). Computability and Unsolvability. Dover Publications (cit. on pp. 44, 45).

Gómez Marín, Raúl and Sicard Ramírez, Andrés [2001] (2002). Informática Teórica: Elementos propedeúticos. 1era. reimpresión. Fondo Editorial Universidad EAFIT. URL: http://www1.eafit.edu.co/asr/pubs/informatica-teorica.html (cit. on pp. 44, 45).

References

- Kleene, Stephen Cole [1952] (1974). Introduction to Metamathematics. Seventh reprint. North-Holland (cit. on pp. 44, 45).
- Mendelson, Elliott [1964] (2015). Introduction to Mathematical Logic. 6th ed. CRC Press (cit. on pp. 44, 45).