Ordinals and Typed Lambda Calculus

Definable Ordinals in the Simply Typed Lambda Calculus

Andrés Sicard-Ramírez

Universidad EAFIT

Semester 2018-2

Introduction

Alonzo Church (1903 – 1995)*

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

Introduction

A FORMULATION OF THE SIMPLE THEORY OF TYPES

ALONZO CHURCH

The purpose of the present paper is to give a formulation of the simple theory of types[1] which incorporates certain features of the calculus of λ-conversion.[2] A complete incorporation of the calculus of λ-conversion into the theory of types is impossible if we require that λx and juxtaposition shall retain their respective meanings as an abstraction operator and as denoting the application of function to argument. But the present partial incorporation has certain advantages from the point of view of type theory and is offered as being of interest on this basis (whatever may be thought of the finally satisfactory character of the theory of types as a foundation for logic and mathematics).

For features of the formulation which are not immediately connected with the incorporation of λ-conversion, we are heavily indebted to Whitehead and Russell,[3] Hilbert and Ackermann,[4] Hilbert and Bernays,[5] and to forerunners of these, as the reader familiar with the works in question will recognize.

1. The hierarchy of types. The class of *type symbols* is described by the rules that ι and o are each type symbols and that if α and β are type symbols then $(\alpha\beta)$ is a type symbol: it is the least class of symbols which contains the symbols ι and o and is closed under the operation of forming the symbol $(\alpha\beta)$ from the symbols α and β.

As exemplified in the statement just made, we shall use the Greek letters α, β, γ to represent variable or undetermined type symbols. We shall abbreviate type symbols by omission of parentheses with the convention that association is to the left—so that, for instance, $o\iota$ will be an abbreviation for $(o\iota)$, $\iota\iota$ for $((\iota\iota)\iota)$, $\iota(\iota\iota)$ for $((\iota\iota)(\iota\iota))$, etc. Moreover, we shall use α' as an abbreviation for $((\alpha\alpha)(\alpha\alpha))$, α'' as an abbreviation for $((\alpha'\alpha')(\alpha'\alpha'))$, etc.

The type symbols enter our formal theory only as subscripts upon variables and constants. In the interpretation of the theory it is intended that the

Received March 23, 1940.

Simply Typed Lambda Calculus

Convention

The simply typed λ-calculus will be denoted by $\lambda^{\to}$.

Simply Typed Lambda Calculus

Convention

The simply typed λ-calculus will be denoted by $\lambda^{\to}$.

Definition

Let V a denumerable set of variables. The set of **$\boldsymbol{\lambda}$-terms**, denoted by Λ, is inductively defined by

$$x \in V \Rightarrow x \in \Lambda \qquad \text{(variable)}$$
$$M, N \in \Lambda \Rightarrow (M\,N) \in \Lambda \qquad \text{(application)}$$
$$M \in \Lambda, x \in V \Rightarrow (\lambda x.M) \in \Lambda \qquad (\lambda\text{-abstraction})$$

Types

Definition

Let B a set of base types (i.e. constant types). The **set of types** of $\lambda^{\to}$, denoted by $\mathbb{T}$, is inductively defined by

$$b \in B \Rightarrow b \in \mathbb{T} \qquad \text{(constant types)}$$
$$\sigma, \tau \in \mathbb{T} \Rightarrow (\sigma \to \tau) \in \mathbb{T} \qquad \text{(function types)}$$

Remark

Usually, the set of types is defined by the abstract grammar

$$\begin{aligned} \tau ::= & \; b \in B \\ & \mid \tau \to \tau \end{aligned}$$

Types

Convention

Arbitrary types will be denoted by σ and τ, with or without number-subscripts.

Types

Convention

Arbitrary types will be denoted by σ and τ, with or without number-subscripts.

Example

The simple theory of types of Church [1940] only included two base types:

ι: the type of individuals and
o: the type of propositions.

Type-Assignments

Definition

A **formula** or **type-assignment** in $\lambda^{\to}$ is of the form

$$M : \sigma,$$

which means that λ-term M has type σ.

Type-Assignments

Example

The simple theory of types of Church [1940] included the following type-assignments:

$N : o \to o$	(negation)
$A : o \to o \to o$	(disjunction)
$\Pi : (\sigma \to o) \to o$	(universal quantifier)
$0_{\sigma'} : \sigma' \to \sigma'$	(zero)
$S_{\sigma'} : (\sigma' \to \sigma') \to \sigma' \to \sigma'$	(successor)

where σ' denotes $(\sigma \to \sigma) \to (\sigma \to \sigma)$.

Type-Contexts

Definition

A **type-context** (or **basis**) Γ is a finite set of type-assignments

$$\Gamma = \{ x_1 : \sigma_1, \ldots, x_n : \sigma_n \}$$

in which there is no a term-variable with more than one assignment.

Remark

Note that a type-context only includes typed term-variables.

Typing Relation

Definition

The **typing relation** $\Gamma \vdash M : \sigma$ means that λ-term M has type σ in the context Γ.

Typing Rules

Rules of type-assignment

The typing rules for $\lambda^{\to}$ (relative to a context Γ) are given by*

$$\Gamma, x : \sigma \vdash x : \sigma \qquad \text{(axiom)}$$

$$\frac{\Gamma \vdash M : (\sigma \to \tau) \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash (M\,N) : \tau} \qquad (\to\text{-elimination})$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x.M) : (\sigma \to \tau)} \qquad (\to\text{-introduction})$$

*See, e.g. [Barendregt 1992; Hindley (1997) 2008].

Typing Rules

Example

To prove that $\lambda x.x$ has type $\sigma \to \sigma$, for all $\sigma \in \mathbb{T}$.

Typing Rules

Example

To prove that $\lambda x.x$ has type $\sigma \to \sigma$, for all $\sigma \in \mathbb{T}$.

Proof

$$\frac{x : \sigma \vdash x : \sigma}{\vdash \lambda x.x : \sigma \to \sigma} \to \mathsf{I}$$

Typing Rules

Example

To prove that $\lambda\, x\, y.\, x$ has type $\sigma \to (\tau \to \sigma)$, for all $\sigma, \tau, \in \mathbb{T}$.

Proof

$$\dfrac{\dfrac{x : \sigma, y : \tau \vdash x : \sigma}{x : \sigma \vdash \lambda y.x : \tau \to \sigma}\to\mathsf{I}}{\vdash \lambda\, x\, y.\, x : \sigma \to (\tau \to \sigma)}\to\mathsf{I}$$

Typing Rules

Example

The composition operator $\lambda\, f\, g\, x.\, f\, (g\, x)$ has type $(\sigma_2 \to \sigma_3) \to (\sigma_1 \to \sigma_2) \to \sigma_1 \to \sigma_3$, for all $\sigma_1, \sigma_2, \sigma_3 \in \mathbb{T}$.

Proof

Let $\Gamma = \{\, f : \sigma_2 \to \sigma_3, g : \sigma_1 \to \sigma_2, x : \sigma_1 \,\}$. Then

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash f : \sigma_2 \to \sigma_3 \qquad \dfrac{\Gamma \vdash g : \sigma_1 \to \sigma_2 \qquad \Gamma \vdash x : \sigma_1}{\Gamma \vdash g\, x : \sigma_2}\,{\to}\mathsf{E}}{\Gamma \vdash f\, (g\, x) : \sigma_3}\,{\to}\mathsf{E}}{f : \sigma_2 \to \sigma_3, g : \sigma_1 \to \sigma_2 \vdash \lambda\, x.\, f\, (g\, x) : \sigma_1 \to \sigma_3}\,{\to}\mathsf{I}}{f : \sigma_2 \to \sigma_3 \vdash \lambda\, g\, x.\, f\, (g\, x) : (\sigma_1 \to \sigma_2) \to (\sigma_1 \to \sigma_3)}\,{\to}\mathsf{I}}{\vdash \lambda\, f\, g\, x.\, f(\, g\, x) : (\sigma_2 \to \sigma_3) \to (\sigma_1 \to \sigma_2) \to (\sigma_1 \to \sigma_3)}\,{\to}\mathsf{I}$$

Typing Rules

Example

Let $\Gamma = \{y : \sigma\}$ be a basis. To prove that $\Gamma \vdash (\lambda x.x)\, y : \sigma$.

Proof

$$\dfrac{\dfrac{\Gamma, x : \sigma \vdash x : \sigma}{\Gamma \vdash \lambda x.x : \sigma \to \sigma}\,{\to}\mathsf{I} \qquad \Gamma \vdash y : \sigma}{\Gamma \vdash (\lambda x.x)\, y : \sigma}\,{\to}\mathsf{E}$$

Typing Rules

Example

The self-application xx has no a type assigned in $\lambda^{\to}$.

*See, e.g. [Barendregt, Dekkers and Statman 2013, Proposition 2.4.24].

Typing Rules

Example

The self-application xx has no a type assigned in $\lambda^{\to}$.

Example

The fixed-point combinator $\mathsf{Y} \coloneqq \lambda\, f.\, (\lambda x\,.\, f\,(x\,x))\,(\lambda\, x.\, f\,(x\,x))$ does not have a type in $\lambda^{\to}$.*

*See, e.g. [Barendregt, Dekkers and Statman 2013, Proposition 2.4.24].

Typing Rules

Example

The self-application xx has no a type assigned in $\lambda^{\to}$.

Example

The fixed-point combinator $\mathsf{Y} \coloneqq \lambda f.\,(\lambda x.\,f\,(x\,x))\,(\lambda x.\,f\,(x\,x))$ does not have a type in $\lambda^{\to}$.*

Example

The λ-term $\Omega \coloneqq (\lambda x.\,x\,x)\,(\lambda x.\,x\,x)$ does not have a type in $\lambda^{\to}$ (because it has no a β-normal form).

*See, e.g. [Barendregt, Dekkers and Statman 2013, Proposition 2.4.24].

Definable Number-Theoretic Functions

Remark

Recall that a number-theoretic function is a function whose signature is

$$\mathbb{N}^k \to \mathbb{N}, \text{ with } k \in \mathbb{N}.$$

Definable Number-Theoretic Functions

Example

The following functions are number-theoretic functions:

$$\mathrm{z}(x) = 0 \qquad \text{(**zero** function)}$$

$$\mathrm{s}(x) = x + 1 \qquad \text{(**successor** function)}$$

$$\mathrm{p}^n_k(x_1, \ldots, x_n) = x_k \qquad \text{(**n-ary projection** functions)}$$

$$\mathrm{k}_n(x) = n \qquad \text{(**n-constant** functions)}$$

$$\mathrm{discr}(x, y, z) = \begin{cases} y, & \text{if } x \neq 0 \\ z, & \text{if } x = 0 \end{cases} \qquad \text{(**discriminator** function)}$$

Definable Number-Theoretic Functions

Definition

Let's work with the $\lambda^{\to}$ with only one base type, denoted type o. The **type of natural numbers** is defined by

$$\mathsf{Nat} \coloneqq (o \to o) \to (o \to o).$$

Definable Number-Theoretic Functions

Example

The λ-term $\mathsf{c}_0 \coloneqq \lambda\, f\, x.\, x$ has type Nat.

$$\dfrac{\dfrac{f : o \to o, x : o \vdash x : o}{f : o \to o \vdash \lambda x.x : o \to o}\,{\to}\mathsf{I}}{\vdash \lambda\, f\, x.\, x : (o \to o) \to o \to o}\,{\to}\mathsf{I}$$

Definable Number-Theoretic Functions

Example

The λ-term succ $:= \lambda\, n\, f\, x.\, f\,(n\, f\, x)$ has type Nat $\to$ Nat.

Let $\Gamma = \{\, n : \mathsf{Nat}, f : o \to o, x : o \,\}$. Then

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash n : \mathsf{Nat} \qquad \Gamma \vdash f : o \to o}{\Gamma \vdash n\, f : o \to o}\to\mathsf{E} \qquad \Gamma \vdash x : o}{\Gamma \vdash n\, f\, x : o}\to\mathsf{E}}{\Gamma \vdash f\,(n\, f\, x) : o}\to\mathsf{E}}{n : \mathsf{Nat}, f : o \to o \vdash \lambda x.\, f\,(n\, f\, x) : o \to o}\to\mathsf{I}}{n : \mathsf{Nat} \vdash \lambda\, f\, x.\, f\,(n\, f\, x) : \mathsf{Nat}}\to\mathsf{I}}{\vdash \lambda\, n\, f\, x.\, f\,(n\, f\, x) : \mathsf{Nat} \to \mathsf{Nat}}\to\mathsf{I}$$

Definable Number-Theoretic Functions

Example

$$
\begin{aligned}
\mathsf{c}_{n+1} &: \mathsf{Nat} && \coloneqq \mathsf{succ}\,\mathsf{c}_n, \\
\mathsf{add} &: \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat} && \coloneqq \lambda\, m\, n\, f\, x.\, m\, f\, (n\, f\, x), \\
\mathsf{mult} &: \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat} && \coloneqq \lambda\, m\, n\, f.\, m\, (n\, f).
\end{aligned}
$$

Definable Number-Theoretic Functions

Definition

Let $\varphi : \mathbb{N}^k \to \mathbb{N}$ be a number-theoretic function. The function φ is **definable respect to** Nat in the $\lambda^{\to}$ iff there exists a λ-term $F : \mathsf{Nat}^n \to \mathsf{Nat}$ such that for all $n_1, \ldots, n_k \in \mathbb{N}$,

$$\varphi(n_1, \ldots, n_k) = a \Rightarrow F\, \mathsf{c}_{n_1} \ldots \mathsf{c}_{n_k} =_\beta \mathsf{c}_a.$$

Definable Number-Theoretic Functions

Definition

The class of the so-called **extended polynomials** is the smallest class of number-theoretic functions including the constant functions k_0 and k_1, the projection functions, addition, multiplication, the discrimination function and closed under composition.*

*See, e.g. [Statman 1979; Danner 1999]. A different but equivalent definition is given in [Troelstra and Schwichtenberg (1996) 2000].

Definable Number-Theoretic Functions

Theorem

If φ is an extended polynomial, then φ is definable (respect to the type Nat) in the $\lambda^{\rightarrow}$.

*The theorem was proved by Schwichtenberg [1976] and stated by Statman [1979] as pointed out by Danner [1999].

Definable Number-Theoretic Functions

Theorem

If φ is an extended polynomial, then φ is definable (respect to the type Nat) in the $\lambda^{\rightarrow}$.

Theorem

If a number-theoretic function φ is definable (respect to the type Nat) in the $\lambda^{\rightarrow}$, then the function φ is an extended polynomial.*

*The theorem was proved by Schwichtenberg [1976] and stated by Statman [1979] as pointed out by Danner [1999].

Definable Number-Theoretic Functions

Remark

A function

$$\mathsf{pred} : \mathsf{Nat} \to \mathsf{Nat}$$
$$\mathsf{pred}\, \mathsf{c}_{n+1} =_\beta \mathsf{c}_n.$$

cannot be defined in the $\lambda^{\to}$.*

*See, e.g. [Barendregt, Dekkers and Statman 2013, Proposition 2.4.22].

Definable Number-Theoretic Functions

Remark

Let $P = (o \to o) \to (o \to o) \to o$. The type of the predecessor function defined in the untyped λ-calculus is

$$\begin{aligned} &\mathsf{pred} : (P \to P) \to o \to o \to o \\ &\mathsf{pred} \coloneqq \lambda\, n\, f\, x.\, n\, (\lambda\, g\, h.\, h\, (g\, f))\, (\lambda u.x)\, (\lambda u.u). \end{aligned}$$

Definable Number-Theoretic Functions

Remark

Let $P = (o \to o) \to (o \to o) \to o$. The type of the predecessor function defined in the untyped λ-calculus is

$$\begin{aligned} &\mathsf{pred} : (P \to P) \to o \to o \to o \\ &\mathsf{pred} \coloneqq \lambda\, n\, f\, x.\, n\, (\lambda\, g\, h.\, h\, (g\, f))\, (\lambda u.x)\, (\lambda u.u). \end{aligned}$$

Remark

There are not functions defined by (primitive) recursion in the $\lambda^{\to}$.

Definable Number-Theoretic Functions

Remark

Troelstra and Schwichtenberg [(1996) 2000, p. 21–22] point out that

> *However, if we permit ourselves the use of Church numerals of different types, and in particular liberalize the notion of representation of a function by permitting numerals of different types for the input and the output, we can represent more than extended polynomials.*

Definable Number-Theoretic Functions

Example

Let Nat_A be the type of natural numbers of type A

$$\mathsf{Nat}_A \coloneqq (A \to A) \to (A \to A).$$

and let $\overline{n}_A$ be a numeral of type Nat_A. We can define exponentiation by

$$\mathsf{exp} : \mathsf{Nat}_A \to \mathsf{Nat}_{A\to A} \to \mathsf{Nat}_A \coloneqq \lambda\, m\, n\, f\, x.\, (n\, m)\, f\, x,$$

where

$$\mathsf{exp}\, \overline{m}_A\, \overline{n}_{A\to A} =_\beta \overline{m^n}_A\, (n > 0).$$

Definable Ordinals

Abstractions on the representation of natural numbers

- No abstraction

  We can define two constants

  $$\mathsf{Z} : o,$$
  $$\mathsf{S} : o \to o,$$

  so natural numbers are represented by terms of the form

  $$\mathsf{S}(\ldots(\mathsf{S}\,\mathsf{Z})\ldots).$$

  Using this representation the definable number-theoretic functions are those definable by iterating successor (e.g. projection and constant functions) [Danner 1999].

(continued on next slide)

Definable Ordinals

Abstractions on the representation of natural numbers (continuation)

- Abstraction on zero

  By abstracting on zero, natural numbers are represented by terms of the form

  $$\lambda z.\, \mathsf{S}\,(\ldots(\mathsf{S}\,z)\ldots).$$

  Using this representation the definable number-theoretic functions are those definable by iterating successor and addition [Danner 1999].

(continued on next slide)

Definable Ordinals

Abstractions on the representation of natural numbers (continuation)

- Abstraction on zero and successor

  By abstracting on zero and successor, natural numbers are represented by the Church numerals

  $$\lambda\, s\, z.\, s\,(\ldots(s\, z)\ldots).$$

  And we know which are the definable number-theoretic functions using this representation (i.e. the extended polynomials).

Definable Ordinals

Representations of ordinals below ω^ω

From the possibles abstractions for representing natural numbers, Danner [1999] defines four representations (called canonical, additive, arithmetic and intensional) for ordinals below ω^ω.

Definable Ordinals

Notation

For $n \in \mathbb{N}$, we define

$$M^0\, N \coloneqq N,$$
$$M^{n+1}\, N \coloneqq M\, (M^n\, N).$$

Definable Ordinals

An ω-fold iterator (informally)

We introduced a constant L which 'represents' an ω-fold iterator.

Example

- The λ-term $\mathsf{L}\,\mathsf{S}$ represents the function $\cdot + w$ (the limit of iterating successor).
- The λ-term $\mathsf{L}^2\,\mathsf{S}$ represents the function $\cdot + w^2$ (the limit of iterating $\cdot + w$).
- The λ-term $\mathsf{L}^n\,\mathsf{S}$ represents the function $\cdot + w^n$.
- The λ-term $\mathsf{S}\,(\mathsf{S}\,(\mathsf{L}\,\mathsf{S}\,(\mathsf{L}^2\,\mathsf{S}\,\mathsf{Z})))$ represents the ordinal $\omega^2 + w + 2$.

Definable Ordinals

Definition

The **type of ordinal numbers** is defined by

$$\mathsf{ON} \coloneqq (o \to o) \to o \to o.$$

Definable Ordinals

Canonical representations

Let L be of type ON and let α be an ordinal below ω^ω

$$\alpha = \omega^{n_r} + \cdots + \omega^{n_0},$$

where $n_r \geq n_{r-1} \geq \cdots \geq n_0$ and $n_i \in \mathbb{N}$.

The **canonical representation** for α, denoted by $\bar{\alpha}$, is defined by

$$\begin{aligned} \alpha^* &:= (\mathsf{L}^{n_0}\, s)\,((\mathsf{L}^{n_1}\, s)\,(\ldots((\mathsf{L}^{n_r}\, s)\, z)\ldots)), \\ \bar{\alpha} &:= \lambda s.\lambda z.\alpha^*, \end{aligned}$$

where $s : \mathsf{ON} \to \mathsf{ON}$ and $z : \mathsf{ON}$.

Definable Ordinal Functions

Definition

An ordinal function $\varphi(\alpha_1, \ldots, \alpha_n)$ is **canonically, additively, arithmetically or intensionally definable respect to** ON in the $\lambda^{\rightarrow}$ iff there exits a closed λ-term $F : \mathsf{ON}^n \rightarrow \mathsf{ON}$ such that

$$\varphi(\alpha_1, \ldots, \alpha_n) = \beta \Rightarrow F A_1 \ \ldots \ A_n \ \beta\eta\text{-reduces to } B,$$

where $A_1, \ldots, A_n$ and B are the canonical, additive, arithmetic or intensional representations for $\alpha_1, \ldots, \alpha_n$ and β, respectively [Danner 1999, p. 191].

Definable Ordinal Functions

Notation

Let A be a set of ordinal functions. The smallest class of ordinal functions below ω^ω including the constant functions, the projection functions, the set A and closed under composition is denoted by $[A]$.

Definable Ordinal Functions

Theorem (Danner [1999], Theorem 4.11)

The ordinal functions canonically definable are exactly the functions

$$[\alpha \mapsto \alpha + 1, \alpha \mapsto \omega\alpha].$$

Definable Ordinal Functions

Theorem (Danner [1999], Theorem 4.11)

The ordinal functions canonically definable are exactly the functions

$$[\alpha \mapsto \alpha + 1, \alpha \mapsto \omega\alpha].$$

Theorem (Danner [1999], Theorem 4.15)

The ordinal functions additively definable are exactly the functions

$$[+, \alpha \mapsto \omega\alpha].$$

Definable Ordinal Functions

Theorem (Danner [1999], Theorem 4.11)

The ordinal functions canonically definable are exactly the functions

$$[\alpha \mapsto \alpha + 1, \alpha \mapsto \omega\alpha].$$

Theorem (Danner [1999], Theorem 4.15)

The ordinal functions additively definable are exactly the functions

$$[+, \alpha \mapsto \omega\alpha].$$

Theorem (Danner [1999], Theorem 4.21)

The ordinal functions arithmetically definable are exactly the functions

$$[+, \times].$$

Definable Ordinal Functions

Definition

The **weak discriminator** function on ordinals is the function defined by [Danner 1999, p. 198]

$$\mathrm{wdiscr}(\alpha, \beta, \delta) \coloneqq \begin{cases} \beta, & \text{if } \alpha > 0; \\ \delta, & \text{if } a = 0. \end{cases}$$

Definable Ordinal Functions

Definition

The **weak discriminator** function on ordinals is the function defined by [Danner 1999, p. 198]

$$\mathrm{wdiscr}(\alpha, \beta, \delta) \coloneqq \begin{cases} \beta, & \text{if } \alpha > 0; \\ \delta, & \text{if } a = 0. \end{cases}$$

Theorem (Danner [1999], Theorem 4.24)

The ordinal functions intensionally definable are exactly the functions

$$[\,+, \times, \mathrm{wdiscr}\,].$$

References


Barendregt, Henk (1992). Lambda Calculi with Types. In: Handbook of Logic in Computer Science. Ed. by Abramsky, S., Gabbay, Dov M. and Maibaum, T. S. E. Vol. 2. Clarendon Press, pp. 117–309 (cit. on p. 13).


Barendregt, Henk, Dekkers, Wil and Statman, Richard (2013). Lambda Calculus with Types. Cambridge University Press. DOI: 10.1017/CBO9781139032636 (cit. on pp. 19–21, 32).


Church, Alonzo (1940). A Formulation of the Simple Theory of Types. The Journal of Symbolic Logic 5.2, pp. 55–68. DOI: 10.2307/2266170 (cit. on pp. 7, 8, 10).


Danner, N. (1999). Ordinals and Ordinal Functions Representable in the Simply Typed Lambda Calculs. Annals of Pure and Applied Logic 97.1-3, pp. 179–201. DOI: 10.1016/S0168-0072(98)00046-3 (cit. on pp. 29–31, 37, 38, 40, 45, 47–51).


Hindley, J. Roger [1997] (2008). Basic Simple Type Theory. Digitally printed version with corrections. Vol. 42. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (cit. on p. 13).

References


Schwichtenberg, H. (1976). Functions Definable in the Simply-Typed Lambda Calculus. Translation by R. J. Irwin of *Definierbare Funktionen im λ-Kalkül mit Typen*. Arch. Math. Logic, 1976, vol 17(3–4), pp. 113–114. URL: http://www.cis.syr.edu/~royer/htc96/fdst.ps (cit. on pp. 30, 31).


Statman, Richard (1979). The Typed λ-Calculus Is Not Elementary Recursive. Theoretical Computer Science 9.1, pp. 73–81. DOI: 10.1016/0304-3975(79)90007-0 (cit. on pp. 29–31).


Troelstra, A. S. and Schwichtenberg, H. [1996] (2000). Basic Proof Theory. 2nd ed. Vol. 44. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press. DOI: 10.1017/CBO9781139168717 (cit. on pp. 29, 35).