Ordinals and Typed Lambda Calculus Numeral Systems in the Lambda Calculus

Andrés Sicard-Ramírez

Universidad EAFIT

Semester 2018-2

(Last modification: 7th May 2025)

Booleans

We encoding the Booleans.

true := $\lambda x y. x$, where true $M N =_{\beta} M$. false := $\lambda x y. y$, where false $M N =_{\beta} N$.

Definition

For $n \in \mathbb{N}$ and $F, M \in \Delta$, we define

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

Definition

The **Church numerals**, denoted by c_n with $n \in \mathbb{N}$, encode the natural numbers.

• One definition

$$\mathsf{c}_n \coloneqq \lambda \, f \, x. \, f^n \, x,$$

• Other definition

$$\begin{split} \mathbf{c}_0 &\coloneqq \lambda \, f \, x. \, x, \\ \mathbf{succ} &\coloneqq \lambda \, n \, f \, x. \, f \, (n \, f \, x), \\ \mathbf{c}_{n+1} &\coloneqq \mathbf{succ} \, \mathbf{c}_n. \end{split}$$

Example

 β -nf's for some numerals.

$$\begin{split} \mathbf{c}_{0} &=_{\beta} \lambda \, f \, x. \, x, \\ \mathbf{c}_{1} &=_{\beta} \lambda \, f \, x. \, f \, x, \\ \mathbf{c}_{2} &=_{\beta} \lambda \, f \, x. \, f \, (f \, x), \\ \mathbf{c}_{3} &=_{\beta} \lambda \, f \, x. \, f \, (f \, (f \, x)) \end{split}$$

.

Whiteboard.

Addition, multiplication and exponentiation

Encoding for the arithmetic operations.

add := $\lambda m n f x. m f (n f x)$, where

add $c_m c_n =_{\beta} c_{m+n}$.

$$\begin{split} \mathsf{mult} &\coloneqq \lambda \, m \, n \, f \, x. \, m \, (n \, f) \, x \\ &\equiv \lambda \, m \, n \, f. \, m \, (n \, f), \text{ where} \end{split}$$

mult $c_m c_n =_{\beta} c_{m \times n}$.

 $\exp := \lambda m n f x. (n m) f x$, where

 $\exp c_m c_n =_{\beta} c_{m^n}$.

Testing for zero

A combinator for testing for zero is defined by

isZero $\coloneqq \lambda n. n (\lambda x. false)$ true,

where

 $\begin{aligned} &\text{isZero } \mathbf{c}_0 &=_\beta \text{true}, \\ &\text{isZero } \mathbf{c}_{n+1} =_\beta \text{false}. \end{aligned}$

Predecessor

A predecessor combinator is defined by

```
\mathsf{pred} \coloneqq \lambda \, n \, f \, x. \, n \, (\lambda \, g \, h. \, h \, (g \, f)) \, (\lambda u. x) \, (\lambda u. u),
```

where

 $\begin{aligned} & \operatorname{pred} \mathsf{c}_0 & =_\beta \mathsf{c}_0, \\ & \operatorname{pred} \mathsf{c}_{n+1} =_\beta \mathsf{c}_n. \end{aligned}$

Recursion

The factorial function is an example of recursive functions on natural numbers following the schemata

 $f : \mathbb{N} \to A$ f 0 = af (S n) = g n (f n)

where A is a set, $a \in A$, $g : \mathbb{N} \to A \to A$ and S is the successor function.

Remark

A reader familiarised with recursion theory will identify the primitive recursion schemata.

A recursor for natural numbers

Let A be a set. From the previous schemata for (primitive) recursive functions, we define a recursor.*

rec :
$$(\mathbb{N} \to A \to A) \to A \to \mathbb{N} \to A$$

rec $f a 0 = a$
rec $f a (S n) = f n (rec f a n)$

^{*}A higher-order function which allow define recursive functions. Numeral Systems in the Lambda Calculus

Recursor

Since rec is also a recursive function, we can define a recursor for numerals using fix.

$$\begin{split} \mathbf{h} &\coloneqq \lambda \, r \, f \, a \, n. \, (\mathsf{isZero} \, n) \, a \, (f \, (\mathsf{pred} \, n) \, (r \, f \, a \, (\mathsf{pred} \, n)))), \\ \mathsf{rec} &\coloneqq \mathsf{fix} \, \mathbf{h}, \end{split}$$

where

$$\operatorname{rec} f a c_{0} =_{\beta} a,$$
$$\operatorname{rec} f a c_{n+1} =_{\beta} f c_{n} (\operatorname{rec} f a c_{n}).$$

Recursor

Since rec is also a recursive function, we can define a recursor for numerals using fix.

$$\begin{split} \mathbf{h} \coloneqq \lambda \, r \, f \, a \, n. \, (\mathsf{isZero} \, n) \, a \, (f \, (\mathsf{pred} \, n) \, (r \, f \, a \, (\mathsf{pred} \, n)))), \\ \mathsf{rec} \coloneqq \mathsf{fix} \, \mathbf{h}, \end{split}$$

where

$$\operatorname{rec} f \, a \, \mathbf{c}_{\mathbf{0}} =_{\beta} a,$$
$$\operatorname{rec} f \, a \, \mathbf{c}_{n+1} =_{\beta} f \, \mathbf{c}_n \, (\operatorname{rec} f \, a \, \mathbf{c}_n).$$

Question

What is it necessary for defining rec? A fixed-point combinator, (implicit) 'Boolean' case analysis, a test for zero and a predecessor function.

Example

A combinator for the factorial function

fac $: \mathbb{N} \to \mathbb{N}$ fac 0 = 1fac $(S n) = S n \times fac n$

is defined by

 $fac \coloneqq rec (\lambda x y. mult (succ x) y) c_1.$

Definition

A number-theoretic function is a function whose signature is

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

Definition

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

$$\begin{split} \varphi(n_1,\ldots,n_k) &= a \Rightarrow F \operatorname{c}_{n_1} \ldots \operatorname{c}_{n_k} =_\beta \operatorname{c}_a, \\ \varphi(n_1,\ldots,n_k) \text{ does not exits} \Rightarrow F \operatorname{c}_{n_1} \ldots \operatorname{c}_{n_k} \text{ has no } \beta\text{-nf.} \end{split}$$

*See, e.g. [Bar2004, Corollary 6.4.6] and [HS2008, Theorem 4.23].

Definition

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

$$arphi(n_1, \dots, n_k) = a \Rightarrow F \, \mathsf{c}_{n_1} \dots \mathsf{c}_{n_k} =_{eta} \mathsf{c}_a,$$

 $arphi(n_1, \dots, n_k) \text{ does not exits} \Rightarrow F \, \mathsf{c}_{n_1} \dots \mathsf{c}_{n_k} \text{ has no } eta \text{-nf}$

Theorem

The Turing-computable functions are λ -definable respect to the Church encoding.*

Numeral Systems in the Lambda Calculus

^{*}See, e.g. [Bar2004, Corollary 6.4.6] and [HS2008, Theorem 4.23].

Definition

A numeral system is a sequence of combinators

 $\mathsf{d}=\mathsf{d}_0,\mathsf{d}_1,\ldots$

and combinators true_d, false_d, succ_d and isZero_d such that for all $n \in \mathbb{N}$ [Bar2004, Definition 6.4.1],

 $\begin{aligned} & \operatorname{succ}_{\mathsf{d}} \mathsf{d}_n & =_\beta \mathsf{d}_{n+1}, \\ & \operatorname{isZero}_{\mathsf{d}} \mathsf{d}_0 & =_\beta \operatorname{true}_{\mathsf{d}}, \\ & \operatorname{isZero}_{\mathsf{d}} \mathsf{d}_{n+1} =_\beta \operatorname{false}_{\mathsf{d}}. \end{aligned}$

Definition

A numeral system is a sequence of combinators

 $\mathsf{d}=\mathsf{d}_0,\mathsf{d}_1,\ldots$

and combinators true_d, false_d, succ_d and isZero_d such that for all $n \in \mathbb{N}$ [Bar2004, Definition 6.4.1],

 $\begin{aligned} & \operatorname{succ}_{\mathsf{d}} \mathsf{d}_n & =_\beta \mathsf{d}_{n+1}, \\ & \operatorname{isZero}_{\mathsf{d}} \mathsf{d}_0 & =_\beta \operatorname{true}_{\mathsf{d}}, \\ & \operatorname{isZero}_{\mathsf{d}} \mathsf{d}_{n+1} =_\beta \operatorname{false}_{\mathsf{d}}. \end{aligned}$

Example

The Church numerals $c=c_0,c_1,\ldots$ and the combinators true, false, succ and isZero are a numeral system.

Numeral Systems in the Lambda Calculus

Notation

We shall denote by d a numeral system $d_0, d_1, \ldots, true_d, false_d, succ_d$ and $isZero_d$.

Notation

We shall denote by d a numeral system $d_0, d_1, \ldots, true_d, false_d, succ_d$ and $isZero_d$.

Lambda definable functions on a numeral system

Let d be a numeral system and let φ be a partial number-theoretic function $\varphi : \mathbb{N}^k \to \mathbb{N}$. The function φ is λ -definable respect to the numeral system d iff there exists a λ -term F such that for all $n_1, \ldots, n_k \in \mathbb{N}$,

$$\varphi(n_1, \dots, n_k) = a \Rightarrow F \operatorname{d}_{n_1} \dots \operatorname{d}_{n_k} =_{\beta} \operatorname{d}_a,$$
$$\varphi(n_1, \dots, n_k) \text{ does not exits} \Rightarrow F \operatorname{d}_{n_1} \dots \operatorname{d}_{n_k} \text{ has no } \beta\text{-nf.}$$

Definition

A numeral system d is **adequate** iff all the Turing-computable functions are λ -definable with respect to d [Bar2004, Definition 6.4.2ii].

Definition

A numeral system d is **adequate** iff all the Turing-computable functions are λ -definable with respect to d [Bar2004, Definition 6.4.2ii].

Example

The Church numeral system is adequate.

Theorem

A numeral system d is adequate iff there exists a combinator $pred_d$ such that for all $n \in \mathbb{N}$ [Bar2004, Theorem 6.4.3],

 $\operatorname{pred}_{\operatorname{d}}\operatorname{d}_{n+1} =_{\beta} \operatorname{d}_{n}.$

Theorem

A numeral system d is adequate iff there exists a combinator $pred_d$ such that for all $n \in \mathbb{N}$ [Bar2004, Theorem 6.4.3],

 $\operatorname{pred}_{\operatorname{d}}\operatorname{d}_{n+1} =_{\beta} \operatorname{d}_{n}.$

Remark

There are various adequate numeral systems in the literature. See, e.g. Barendregt [Bar2004], Goldberg [Gol2000] and Jansen [Jan2013].

References

- [Bar2004] H. P. Barendregt. The Lambda Calculus. Its Syntax and Semantics. Revised edition, 6th impression. Vol. 103. Studies in Logic and the Foundations of Mathematics. Elsevier, 2004 (1981) (cit. on pp. 15–18, 21–24).
- [Gol2000] Mayer Goldberg. An Adequate and Efficient Left-Associated Binary Numeral System in the λ -Calculus. Journal of Functional Programming 10.6 (2000), pp. 607–623. DOI: 10.1017/S0956796800003804 (cit. on pp. 23, 24).
- [HS2008] J. Roger Hindley and Jonathan P. Seldin. Lambda-Calculus and Combinators. An Introduction. Cambridge University Press, 2008 (cit. on pp. 15, 16).
- [Jan2013] Jan Martin Jansen. Programming in the λ-Calculus: From Church to Scott and Back. In: The Beauty of Functional Code. Ed. by Peter Achten and Pieter Koopman. Vol. 8106. Lecture Notes in Computer Science. Springer, 2013, pp. 168–180. DOI: 10.1007/978-3-642-40355-2_12 (cit. on pp. 23, 24).