# Ordinals and Typed Lambda Calculus 

Numeral Systems in the Lambda Calculus

Andrés Sicard-Ramírez

Universidad EAFIT
Semester 2018-2

## Church Encoding

## Booleans

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

## Church Encoding

## Definition

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

$$
\begin{aligned}
F^{0} M & :=M, \\
F^{n+1} M & :=F\left(F^{n} M\right) .
\end{aligned}
$$

## Church Encoding

## Definition

The Church numerals, denoted by $\mathrm{c}_{n}$ with $n \in \mathbb{N}$, encode the natural numbers.
One definition:

$$
\mathrm{c}_{n}:=\lambda f x . f^{n} x
$$

Other definition:

$$
\begin{aligned}
c_{0} & :=\lambda f x . x \\
\text { succ } & :=\lambda n f x . f(n f x) \\
c_{n+1} & :=\operatorname{succ}_{n}
\end{aligned}
$$

## Church Encoding

## Example

$\beta$-nf's for some numerals.

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

Whiteboard.

## Church Encoding

Addition, multiplication and exponentiation
Encoding for the arithmetic operations.

$$
\begin{aligned}
\text { add } & :=\lambda m n f x . m f(n f x), \text { where } & & \text { add } \mathrm{c}_{m} \mathrm{c}_{n}={ }_{\beta} \mathrm{c}_{m+n} . \\
\text { mult } & :=\lambda m n f x \cdot m(n f) x & & \\
& \equiv \lambda m n f . m(n f), \text { where } & & \text { mult } \mathrm{c}_{m} \mathrm{c}_{n}={ }_{\beta} \mathrm{c}_{m \times n} . \\
& & & \operatorname{exp~}_{m} \mathrm{c}_{n}={ }_{\beta} \mathrm{c}_{m^{n}} .
\end{aligned}
$$

## Church Encoding

Testing for zero
A combinator for testing for zero is defined by

$$
\text { isZero }:=\lambda n . n(\lambda x . f a l s e) \text { true, }
$$

where

$$
\begin{aligned}
& \text { isZero } \mathrm{c}_{0} \quad={ }_{\beta} \text { true, } \\
& \text { isZero } \mathrm{c}_{n+1}={ }_{\beta} \text { false. }
\end{aligned}
$$

## Church Encoding

## Predecessor

A predecessor combinator is defined by

$$
\text { pred }:=\lambda n f x . n(\lambda g h . h(g f))(\lambda u \cdot x)(\lambda u \cdot u),
$$

where

$$
\begin{aligned}
& \text { pred } \mathrm{c}_{0}={ }_{\beta} \mathrm{c}_{0}, \\
& \text { pred } \mathrm{c}_{n+1}={ }_{\beta} \mathrm{c}_{n} .
\end{aligned}
$$

## Church Encoding

## Recursion

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

$$
\begin{aligned}
& f \quad: \mathbb{N} \rightarrow A \\
& f 0 \quad=a \\
& f(\mathrm{~S} n)=g n(f n)
\end{aligned}
$$

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

## Remark

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

## Church Encoding

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

$$
\begin{array}{ll}
\text { rec } & :(\mathbb{N} \rightarrow A \rightarrow A) \rightarrow A \rightarrow \mathbb{N} \rightarrow A \\
\operatorname{rec} f a 0 & =a \\
\operatorname{rec} f a(\mathrm{~S} n) & =f n(\operatorname{rec} f a n)
\end{array}
$$

[^0]
## Church Encoding

## Recursor

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

$$
\begin{aligned}
\mathrm{h} & :=\lambda r f a n .(\text { isZero } n) a(f(\operatorname{pred} n)(r f a(\operatorname{pred} n)))), \\
\mathrm{rec} & :=\mathrm{fixh},
\end{aligned}
$$

where

$$
\begin{aligned}
& \operatorname{rec} f a \mathrm{c}_{0}={ }_{\beta} a, \\
& \operatorname{rec} f a \mathrm{c}_{n+1}={ }_{\beta} f \mathrm{c}_{n}\left(\operatorname{rec} f a \mathrm{c}_{n}\right) .
\end{aligned}
$$

## Church Encoding

## Recursor

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

$$
\begin{aligned}
\mathrm{h} & :=\lambda r f a n .(\text { isZero } n) a(f(\operatorname{pred} n)(r f a(\operatorname{pred} n)))), \\
\mathrm{rec} & :=\mathrm{fixh},
\end{aligned}
$$

where

$$
\begin{aligned}
& \operatorname{rec} f a \mathrm{c}_{0}={ }_{\beta} a, \\
& \operatorname{rec} f a \mathrm{c}_{n+1}={ }_{\beta} f \mathrm{c}_{n}\left(\operatorname{rec} f a \mathrm{c}_{n}\right) .
\end{aligned}
$$

## Question

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

## Church Encoding

## Example

A combinator for the factorial function

$$
\begin{array}{ll}
\text { fac } \quad: \mathbb{N} \rightarrow \mathbb{N} \\
\text { fac } 0 \quad & =1 \\
\text { fac }(\operatorname{S} n) & =\mathrm{S} n \times \operatorname{fac} n
\end{array}
$$

is defined by

$$
\mathrm{fac}:=\operatorname{rec}(\lambda x y . \text { mult }(\operatorname{succ} x) y) \mathrm{c}_{1} .
$$

## Church Encoding

## Definition

A number-theoretic function is a function whose signature is

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

## Church Encoding

## Definition

Let $\varphi$ be a partial number-theoretic function $\varphi: \mathbb{N}^{k} \rightarrow \mathbb{N}$. The function $\varphi$ is $\boldsymbol{\lambda}$-definable respect to the Church encoding iff there exists a $\lambda$-term $F$ such that for all $n_{1}, \ldots, n_{k} \in \mathbb{N}$,

$$
\begin{aligned}
\varphi\left(n_{1}, \ldots, n_{k}\right)=a & \Rightarrow F \mathrm{c}_{n_{1}} \ldots \mathrm{c}_{n_{k}}={ }_{\beta} \mathrm{c}_{a} \\
\varphi\left(n_{1}, \ldots, n_{k}\right) \text { does not exits } & \Rightarrow F \mathrm{c}_{n_{1}} \ldots \mathrm{c}_{n_{k}} \text { has no } \beta \text {-nf. }
\end{aligned}
$$

[^1]
## Church Encoding

## Definition

Let $\varphi$ be a partial number-theoretic function $\varphi: \mathbb{N}^{k} \rightarrow \mathbb{N}$. The function $\varphi$ is $\boldsymbol{\lambda}$-definable respect to the Church encoding iff there exists a $\lambda$-term $F$ such that for all $n_{1}, \ldots, n_{k} \in \mathbb{N}$,

$$
\begin{aligned}
\varphi\left(n_{1}, \ldots, n_{k}\right)=a & \Rightarrow F \mathrm{c}_{n_{1}} \ldots \mathrm{c}_{n_{k}}={ }_{\beta} \mathrm{c}_{a} \\
\varphi\left(n_{1}, \ldots, n_{k}\right) \text { does not exits } & \Rightarrow F \mathrm{c}_{n_{1}} \ldots \mathrm{c}_{n_{k}} \text { has no } \beta \text {-nf. }
\end{aligned}
$$

## Theorem

The Turing-computable functions are $\lambda$-definable respect to the Church encoding.*

[^2]
## Numeral Systems

## Definition

A numeral system is a sequence of combinators

$$
\mathrm{d}=\mathrm{d}_{0}, \mathrm{~d}_{1}, \ldots
$$

and combinators trued, false $_{d}$, succ $_{d}$ and isZero ${ }_{d}$ such that for all $n \in \mathbb{N}$ [Barendregt (1981) 2004, Definition 6.4.1],

$$
\begin{aligned}
\text { succ }_{\mathrm{d}} \mathrm{~d}_{n} & ={ }_{\beta} \mathrm{d}_{n+1}, \\
\text { isZero }_{\mathrm{d}} \mathrm{~d}_{0} & ={ }_{\beta} \text { true }_{\mathrm{d}}, \\
\text { isZero }_{\mathrm{d}} \mathrm{~d}_{n+1} & ={ }_{\beta} \text { false }_{\mathrm{d}} .
\end{aligned}
$$

## Numeral Systems

## Definition

A numeral system is a sequence of combinators

$$
\mathrm{d}=\mathrm{d}_{0}, \mathrm{~d}_{1}, \ldots
$$

and combinators true ${ }_{d}$, false $_{d}$, succ $_{d}$ and isZerod such that for all $n \in \mathbb{N}$ [Barendregt (1981) 2004, Definition 6.4.1],

$$
\begin{aligned}
\text { succ }_{\mathrm{d}} \mathrm{~d}_{n} & ={ }_{\beta} \mathrm{d}_{n+1}, \\
\text { isZero }_{\mathrm{d}} \mathrm{~d}_{0} & ={ }_{\beta} \text { true }_{\mathrm{d}}, \\
\text { isZero }_{\mathrm{d}} \mathrm{~d}_{n+1} & ={ }_{\beta} \text { false }_{\mathrm{d}} .
\end{aligned}
$$

## Example

The Church numerals $c=c_{0}, c_{1}, \ldots$ and the combinators true, false, succ and isZero are a numeral system.

## Numeral Systems

## Notation

We shall denote by $d$ a numeral system $d_{0}, d_{1}, \ldots$, true $_{d}$, false ${ }_{d}$, succ $_{d}$ and isZero ${ }_{d}$.

## Numeral Systems

## Notation

We shall denote by $d$ a numeral system $d_{0}, d_{1}, \ldots$, true $_{d}$, false ${ }_{d}$, succ $_{d}$ and isZerod .
Lambda definable functions on a numeral system
Let $d$ be a numeral system and let $\varphi$ be a partial number-theoretic function $\varphi: \mathbb{N}^{k} \rightarrow \mathbb{N}$. The function $\varphi$ is $\boldsymbol{\lambda}$-definable respect to the numeral system $d$ iff there exists a $\lambda$-term $F$ such that for all $n_{1}, \ldots, n_{k} \in \mathbb{N}$,

$$
\begin{aligned}
\varphi\left(n_{1}, \ldots, n_{k}\right)=a & \Rightarrow F \mathrm{~d}_{n_{1}} \ldots \mathrm{~d}_{n_{k}}={ }_{\beta} \mathrm{d}_{a} \\
\varphi\left(n_{1}, \ldots, n_{k}\right) \text { does not exits } & \Rightarrow F \mathrm{~d}_{n_{1}} \ldots \mathrm{~d}_{n_{k}} \text { has no } \beta \text {-nf. }
\end{aligned}
$$

## Numeral Systems

## Definition

A numeral system $d$ is adequate iff all the Turing-computable functions are $\lambda$-definable with respect to d [Barendregt (1981) 2004, Definition 6.4.2ii].

## Numeral Systems

## Definition

A numeral system d is adequate iff all the Turing-computable functions are $\lambda$-definable with respect to d [Barendregt (1981) 2004, Definition 6.4.2ii].

Example
The Church numeral system is adequate.

## Numeral Systems

## Theorem

A numeral system $d$ is adequate iff there exists a combinator pred $_{\mathrm{d}}$ such that for all $n \in$ $\mathbb{N}$ [Barendregt (1981) 2004, Theorem 6.4.3],

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

## Numeral Systems

## Theorem

A numeral system d is adequate iff there exists a combinator pred $_{\mathrm{d}}$ such that for all $n \in$ $\mathbb{N}$ [Barendregt (1981) 2004, Theorem 6.4.3],

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

## Remark

There are various adequate numeral systems in the literature. See, e.g. Barendregt [(1981) 2004], Goldberg [2000] and Jansen [2013].

## 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 pp. 15-18, 21-24).
Goldberg, Mayer (2000). An Adequate and Efficient Left-Associated Binary Numeral System in the $\lambda$-Calculus. Journal of Functional Programming 10.6, pp. 607-623. DOI: 10.1017/S095679680000380 (cit. on pp. 23, 24).
Hindley, J. Roger and Seldin, Jonathan P. (2008). Lambda-Calculus and Combinators. An Introduction. Cambridge University Press (cit. on pp. 15, 16).
Jansen, Jan Martin (2013). Programming in the $\lambda$-Calculus: From Church to Scott and Back. In: The Beauty of Functional Code. Ed. by Achten, Peter and Koopman, Pieter. Vol. 8106. Lecture Notes in Computer Science. Springer, pp. 168-180. DoI: 10.1007/978-3-642-40355-2_12 (cit. on pp. 23, 24).


[^0]:    *A higher-order function which allow define recursive functions.

[^1]:    *See, e.g. [Barendregt (1981) 2004, Corollary 6.4.6] and [Hindley and Seldin 2008, Theorem 4.23].

[^2]:    *See, e.g. [Barendregt (1981) 2004, Corollary 6.4.6] and [Hindley and Seldin 2008, Theorem 4.23].

