λ-numerals

Are the Numbers same as the Numerals?

Roman, Arabic and Binary Numerals

Alphabet

Roman	Arabic	Binary
$\{I, V, X, L, C, D, M\}$	$\{0, 1, \cdots, 9\}$	$\{0, 1\}$

Roman, Arabic and Other Numeral : Examples

Roman	Arabic	Binary
	0	0
I	1	1
IV	4	100
XLII	42	101010
LXXII	72	1001000
XCIII	93	1011101
CCCXC	390	110000110
CMXLIX	949	11 1011 0101

A Good System of Numerals

- The **alphabet** Σ of the **system of numerals**.
- Some strings over Σ are **valid numerals**. There is a set of **formation rules**.
- **Arithmetic opeartions** can be performed **with ease** with the system of numerals.
- Compare the **Roman** and the **Arabic** systems for **addition** and **multiplcation** operations.

Inductive Definition of Binary Numerals

Let the alphabet be $\mathbf{\Sigma} = \{\mathbf{0}, \mathbf{1}\}$ and let the set of **binary numerals** be $\mathcal{B}$.

- **0** and **1** are **binary numerals**.
- If **n** be a **binary numeral**, then so is **n0** and **n1**.
- Nothing else is a **binary numeral**.

$$\mathcal{B} = \{\mathbf{0}, \mathbf{1}, \mathbf{00}, \mathbf{01}, \mathbf{10}, \mathbf{11}, \mathbf{000}, \mathbf{001}, \mathbf{010}, \cdots\}$$

Truth Values, Conditionals and Ordered Pair

- **true**: $\lambda \mathbf{xy.x}$, we call it **T**.
- **false**: $\lambda \mathbf{xy.y}$, we call it **F**.
- **if B then u else v**: **Euv**
- **Ordered Pair** $(\mathbf{u}, \mathbf{v})$: $\lambda \mathbf{x.xuv}$.

Why are they defined in this way?

Truth Values and Conditionals

- If we substitute **true** (λ**xy.x**) in place of **E** of the **conditional** **Euv**, it is evaluated to **u**.

$$(\lambda \mathbf{xy.x})\mathbf{uv} \rightarrow_\beta (\lambda \mathbf{y.u})\mathbf{v} \rightarrow_\beta \mathbf{u}$$

- Similarly if we substitute **false** (λ**xy.y**) for **E** we get **v**.

Ordered Pair

- **First Projection**: $\lambda\mathbf{x.xT}$, let us call it $\mathbf{P_0}$.
- **Second Projection**: $\lambda\mathbf{x.xF}$, let us call it $\mathbf{P_1}$.

$$
\begin{aligned}
P_0(u, v) &= (\lambda x.xT)(\lambda x.xuv) \\
&\rightarrow_\beta (\lambda x.xuv)T \\
&\rightarrow_\beta Tuv \\
&= u
\end{aligned}
$$

- Similar is the **second projection**.

Fixed-Point Combinator

A **λ-term** F is called a **fixed-point combinator** if for any λ-term u, $\mathbf{Fu} = \mathbf{u}(\mathbf{Fu})$ i.e. Fu is a **fixed-point** of u.

There is a fixed-point combinator

Curry Fixed-Point Combinator

The following **fixed-point combinator** is due to **Curry**[a].

Let $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ and **u** be any λ-term.

$$\begin{aligned} Yu &= (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx))) \\ &\rightarrow_\beta (\lambda x.u(xx))(\lambda x.u(xx)) \\ &\rightarrow_\beta u((\lambda x.u(xx))(\lambda x.u(xx))) \\ &= u(Yu) \end{aligned}$$

[a]Huskell B. Curry was the founder of **combinatory logic**, a version of **λ-calculus**. The functional programming language **Huskell** is named after him.

Turing Fixed-Point Combinator

The following **fixed-point combinator** is due to **Turing**[a].

Let $\Theta = (\lambda xy.y(xxy))(\lambda xy.y(xxy))$ and **u** be any λ-term.

$$
\begin{aligned}
\Theta u &= (\lambda xy.y(xxy))(\lambda xy.y(xxy))u \\
&\rightarrow_\beta (\lambda y.y((\lambda xy.y(xxy))(\lambda xy.y(xxy))y))u \\
&\rightarrow_\beta u((\lambda xy.y(xxy))(\lambda xy.y(xxy))u) \\
&= u(\Theta u)
\end{aligned}
$$

[a] Alen Turing of **Turing Machine**.

Fool's Fixed-Point Combinator

The following **fixed-point combinator** is due to a **fool**.

Let $A = (\lambda fol.l(fool))$. Then a **fool's** combinator is $F = AAA$. Let **u** be a λ-term.

$$
\begin{aligned}
Fu &= AAAu \\
&= (\lambda fol.l(fool))(\lambda fol.l(fool))(\lambda fol.l(fool))u \\
&\rightarrow_{\beta}^{*} u(AAAu) \\
&= u(Fu)
\end{aligned}
$$

There are Infinitely Many

Fixed Point Combinators

Examples

Let us apply Θ on different **closed** λ-terms (a λ-term without any **free variable**).

- $\Theta I = I(\Theta I) = \Theta I = \cdots$ - **a nonterminating computation.**
- $\Theta K = K(\Theta K) = \lambda y.\Theta K$ - **a function that will return a nonterminating computation.**
- $\Theta K_* = K_*(\Theta K_*) = \lambda y.y$ - **will return the identity function.**

Barendregt Numerals

Decimal (n)	Barendregt Numeral $(\lceil n \rceil)$
$\mathbf{0}$	$\lceil \mathbf{0} \rceil = \mathbf{I} = \lambda \mathbf{x}.\mathbf{x}$
$\mathbf{n+1}$	$\lceil \mathbf{n+1} \rceil = (\mathbf{F}, \lceil \mathbf{n} \rceil) = \lambda \mathbf{x}.\mathbf{x}\mathbf{F}\lceil \mathbf{n} \rceil$

Essential Functions

- **isZero:** Evaluates to **T** if applied to $\lceil \mathbf{0} \rceil$ and evaluates to **T** if applied to $\lceil \mathbf{n} \rceil$, $n > 0$. We choose $\lambda x.xT$.

$$
\begin{aligned}
(\lambda x.xT)\lceil 0 \rceil &= (\lambda x.xT)\lambda x.x \\
&\rightarrow_\beta (\lambda x.x)T \\
&\rightarrow_\beta T
\end{aligned}
$$

Essential Functions

$$
\begin{aligned}
(\lambda x.xT)\lceil n \rceil \quad &= \quad (\lambda x.xT)\lambda x.xF\lceil n-1 \rceil, \ \textbf{if } n > 0 \\
&\rightarrow_\beta \quad (\lambda x.xF\lceil n-1 \rceil)T \\
&\rightarrow_\beta \quad TF\lceil n-1 \rceil \\
&\rightarrow_\beta \quad F
\end{aligned}
$$

Essential Functions

- **Successor:**

$$\lceil \mathbf{n} \rceil \rightarrow^*_\beta \lceil \mathbf{n+1} \rceil$$

Consider $\boldsymbol{\lambda}\mathbf{yx.xFx}$

$$\begin{aligned}(\lambda yx.xFy)\lceil n \rceil &\rightarrow_\beta \lambda x.xF\lceil n \rceil \\ &= \lceil n+1 \rceil\end{aligned}$$

Let us call the **successor** to be **S**.

Essential Functions

- **Predecessor:**

$$\lceil n \rceil \rightarrow_{\beta}^{*} \begin{cases} \lceil 0 \rceil & \textbf{if } n = 0, \\ \lceil n-1 \rceil & \textbf{if } n > 0. \end{cases}$$

Consider $\lambda \mathbf{x}.(\mathbf{isZero\ x})(\lambda \mathbf{x}.\mathbf{x})(\mathbf{xF})$.

$$\begin{aligned} & (\lambda x.(\textbf{isZero } x)(\lambda x.x)(xF))\lceil 0 \rceil \\ \rightarrow_{\beta}\ & (\textbf{isZero } \lceil 0 \rceil)(\lambda x.x)(\lceil 0 \rceil F) \\ \rightarrow_{\beta}\ & T(\lambda x.x)(\lceil 0 \rceil F) \rightarrow_{\beta}^{*} (\lambda x.x) = \lceil 0 \rceil \end{aligned}$$

Essential Functions

$$
\begin{array}{ll}
 & (\lambda x.(\textbf{isZero}\ x)(\lambda x.x)(xF))\lceil n \rceil \ n > 0 \\
\rightarrow_\beta & (\textbf{isZero}\ \lceil n \rceil)(\lambda x.x)(\lceil n \rceil F) \\
\rightarrow_\beta & F(\lambda x.x)(\lceil n \rceil F) \\
\rightarrow_\beta & \lceil n \rceil F \\
= & (\lambda x.xF\lceil n-1 \rceil)F \\
\rightarrow_\beta & FF\lceil n-1 \rceil \\
\rightarrow_\beta^* & FF\lceil n-1 \rceil
\end{array}
$$

Let us call the **predecessor** to be **P**.

Let Us Add

The Inductive definition of **sum** of two natural numbers is as follows:

$$\mathbf{sum}\ m\ n = \begin{cases} m & \textbf{if}\ n = 0, \\ S\ (\mathbf{sum}\ m\ (P\ n)) & \textbf{if}\ n > 0. \end{cases}$$

Functional F

We define the functional F in the following way,

$$F = \lambda f.\lambda m.\lambda n. \begin{cases} m & \text{if } n = 0, \\ S\ (f\ m\ (P\ n)) & \text{if } n > 0. \end{cases}$$

Functional F

$$
\begin{aligned}
& \mathbf{F\ sum} \\
= \quad & \left(\lambda f.\lambda m.\lambda n. \begin{cases} m & \text{if } n = 0, \\ S\ (f\ m\ (P\ n)) & \text{if } n > 0. \end{cases} \right) \text{sum} \\
\rightarrow_\beta \quad & \lambda m.\lambda n. \begin{cases} m & \text{if } n = 0, \\ S\ (\text{sum}\ m\ (P\ n)) & \text{if } n > 0. \end{cases} \\
= \quad & \mathbf{sum}
\end{aligned}
$$

sum is a **fixed-point** of F.

Functional F

$$F = \lambda fmn.(Zn)\ m\ (S\ (f\ m\ (P\ n)))$$

$$Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

$(Y\ \ F)$ **should be sum**

sum ⌈3⌉ ⌈2⌉

$$
\begin{aligned}
Y\ F\ \lceil 3 \rceil\ \lceil 2 \rceil \ &\to_\beta^* \ F\ (Y\ F)\ \lceil 3 \rceil\ \lceil 2 \rceil \\
&\to_\beta^* \ (\lambda fmn.(Zn)\ m\ (S\ (f\ m\ (P\ n))))\\
&\qquad (Y\ F)\ \lceil 3 \rceil\ \lceil 2 \rceil \\
&\to_\beta^* \ (Z\ \lceil 2 \rceil)\ \lceil 3 \rceil\ (S\ ((Y\ F)\ \lceil 3 \rceil\ (P\ \lceil 2 \rceil))) \\
&\to_\beta^* \ S\ ((Y\ F)\ \lceil 3 \rceil\ (P\ \lceil 2 \rceil)) \\
&\to_\beta^* \ S\ ((Y\ F)\ \lceil 3 \rceil\ \lceil 1 \rceil) \\
&\to_\beta^* \ S\ (F\ (Y\ F)\ \lceil 3 \rceil\ \lceil 1 \rceil)
\end{aligned}
$$

sum $\lceil 3 \rceil$ $\lceil 2 \rceil$

$$
\begin{aligned}
\rightarrow^*_\beta &\quad S\ (S\ ((Y\ F)\ \lceil 3 \rceil\ (P\ \lceil 1 \rceil)))\\
\rightarrow^*_\beta &\quad S\ (S\ ((Y\ F)\ \lceil 3 \rceil\ \lceil 0 \rceil))\\
\rightarrow^*_\beta &\quad S\ (S\ ((\lambda fmn.(Zn)\ m\ (S\ (f\ m\ (P\ n)))\\
&\quad\ (Y\ F)\ \lceil 3 \rceil\ \lceil 0 \rceil))\\
\rightarrow^*_\beta &\quad S\ (S\ ((Z\ \lceil 0 \rceil)\ \lceil 3 \rceil\ \ (S\ ((Y\ F)\ \lceil 3 \rceil\ (P\ \lceil 0 \rceil\ )))))\\
\rightarrow^*_\beta &\quad S\ (S\ (\lceil 3 \rceil)\\
\rightarrow^*_\beta &\quad \lceil 5 \rceil
\end{aligned}
$$

Multiplication

Multiplication can be defined as follows.

$$\text{mult } m \; n = \begin{cases} 0 & \text{if } n = 0, \\ \text{sum } m \; (\text{mult } m \; (P \; n)) & \text{if } n > 0. \end{cases}$$

mult is **fixed point** of

$$\lambda \mathbf{f}.\lambda \mathbf{m}.\lambda \mathbf{n}.\ (\mathbf{Zn})\ \lceil \mathbf{0} \rceil\ (\mathbf{sum\ m\ (f\ m\ (P\ n))})$$

Exponentiation : m^n

Multiplication can be defined as follows.

$$\exp\ m\ n = \begin{cases} 1 & \text{if } n = 0, \\ \text{mult } m\ (\exp\ m\ (P\ n)) & \text{if } n > 0. \end{cases}$$

exp is **fixed point** of

$$\lambda \mathbf{f}.\lambda \mathbf{m}.\lambda \mathbf{n}.\ (\mathbf{Zn})\ \lceil \mathbf{1} \rceil\ (\mathbf{mult\ m}\ (\mathbf{f\ m}\ (\mathbf{P\ n})))$$

Apply on Pair

We want a λ-term that will take two functions **f** and **g**, and an ordered pair $(\mathbf{a}, \mathbf{b})$, then will form the ordered pair $(\mathbf{f\ a}, \mathbf{g\ b})$. Let us call it **apply on pair (appP)**

$$\text{appP} \;=\; \lambda f.\lambda g.\lambda p.\lambda x.x\ (f\ (P_0\ p))\ (g\ (P_1\ p))$$

An Example

$$
\begin{aligned}
\text{appP } f\ g\ (u, v) &= \text{appP } f\ g\ \lambda x.xuv \\
&= (\lambda f.\lambda g.\lambda p.\lambda x.x\ (f\ (P_0\ p))\ (g\ (P_1\ p))) \\
&\quad\ f\ g\ (\lambda x.xuv) \\
&\rightarrow_\beta^* \lambda x.x\ (f\ (P_0\ (\lambda x.xuv))) \\
&\quad\ (g\ (P_1\ (\lambda x.xuv))) \\
&\rightarrow_\beta^* \lambda x.x\ (f\ u)\ (g\ v)
\end{aligned}
$$

n Application of f

We want a λ-term that will take **three arguments**, **f**, **n** and **x** and will apply **f**, **n** times on **x**.

$$\mathbf{app\ f\ n\ x} = \begin{cases} x & \textbf{if } n = 0, \\ f\ (\mathbf{app}\ f\ (P\ n)\ x) & \textbf{if } n > 0. \end{cases}$$

app is **fixed point** of

$$\lambda \mathbf{g}.\lambda \mathbf{f}.\lambda \mathbf{n}.\lambda \mathbf{x}.\ (\mathbf{Zn})\ \mathbf{x}\ \mathbf{f}\ (\mathbf{g}\ (\mathbf{P}\ \mathbf{n})\ \mathbf{x}))$$

Review

- We are convinced (I hope) that many useful functions are λ-definable.
- But we shall not prove that the **λ-definable** functions are exactly the **recursive** functions i.e. the functions that can be computed (in principle) by any digital computer.

Church Numerals

Decimal (n)	**Church Numeral** ($\lfloor n \rfloor$)
0	$\lfloor \mathbf{0} \rfloor = \mathbf{I} = \lambda \mathbf{fx.x}$
n	$\lfloor \mathbf{n} \rfloor = \lambda \mathbf{fx}. \overbrace{f\ (f\ (\cdots (f}^{\mathbf{n}}\ \mathbf{x}) \cdots))$

Find out isZero, successor and predecessor functions for Church numerals.