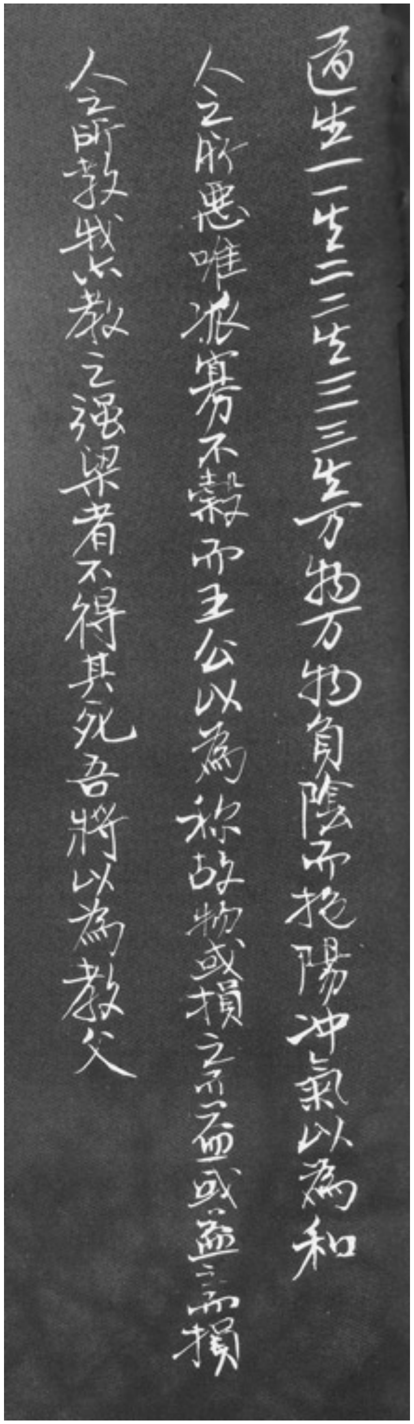


λ Calculus

--- Fun applications

Yan Huang

λ Calculus Formalism (Grammar)

Key words: **λ . ( ) terminals**

Defining λ-*term*:

term → *variable*	e.g., x	y
| (*term*)	e.g., (x)	(y)
| λ *variable* . *term*	e.g., $\lambda\, x\,.\, x$	
| *term term*	e.g., $(\lambda\, x\,.\, x)\, y$	

Humans can give meaning to those symbols to describe computations.

λ Calculus Formalism (Rules)

α-reduction (renaming)

$$\lambda y \,.\, M \longrightarrow_{\alpha} \lambda v \,.\, M\,[y \mapsto v]$$

where v does not occur in M.

β-reduction (substitution)

$$(\lambda x \,.\, M\,)\; T \longrightarrow_{\beta} M\,[x \mapsto T\,]$$

Replace all bounded x of M with T

What’s your favorite number?

What is 11?

eleven

elf

undici

十一

11

once

XI

одиннадцать

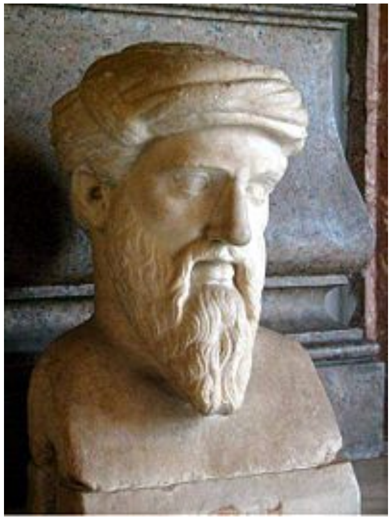
onze

أحد عش

イレブン

Numbers

- The natural numbers had their origins in the words used to count things
- Numbers as abstractions



pred (**succ** N) $\to_\beta$ N
succ (**pred** N) $\to_\beta$ N
pred (**0**) $\to_\beta$ **0**
succ (**zero**) $\to_\beta$ **1**

Defining Numbers

- **0** ≡ $\lambda f x.\ x$
- **1** ≡ $\lambda f x. f(x)$
- **2** ≡ $\lambda f x. f(f(x))$

In Church numerals, ***n*** is represented as a function that maps any function *f* to its *n*-fold composition.

Defining **succ** and **pred**

succ ≡ $\lambda\, n\, f\, x.\, f\, (n\, f\, x)$

pred ≡ $\lambda\, n\, f\, x.\; n\, (\lambda\, g\, h\, .\; h\, (g\, f))\, (\lambda\, u.x)\, (\lambda\, u.u)$

0 ≡ $\lambda\, f\, x.\; x$

succ 0 $\rightarrow_{\beta}$ **?**

Defining **succ** and **pred**

succ $\equiv \lambda\, n\, f\, x.\, f\,(n\, f\, x)$

pred $\equiv \lambda\, n\, f\, x.\, n\,(\lambda\, g\, h\, .\, h\,(g\, f\,))\,(\lambda\, u.x)\,(\lambda\, u.u)$

$$\textbf{pred}(n) = \begin{cases} 0 & \text{if } n = 0 \\ n-1 & \text{otherwise} \end{cases}$$

pred 0 $\rightarrow_\beta$

pred 1 $\rightarrow_\beta$

pred 0

pred 1

Making Decisions

- What does **decision** mean?
  - Choosing different ***strategies*** depending on the ***predicate***

$$\textbf{if T}\ M\ N \rightarrow M$$
$$\textbf{if F}\ M\ N \rightarrow N$$

- What does **True** (**T**) mean?
  - **True** is something that when used as the first operand of **if**, makes the value of the **if** the value of its second operand

Finding the Truth

if $\equiv \lambda\, p\; c\; a\,.\, p\; c\; a$

T $\equiv \lambda\, x\; y\,.\, x$

F $\equiv \lambda\, x\; y\,.\, y$

if T $M\; N$

$((\lambda\, p\; c\; a\,.\, p\; c\; a)\; (\lambda\, x\; y\,.\, x))\; M\; N$

$\rightarrow_{\beta} (\lambda\, c\; a\,.\, (\lambda\, x\; y\,.\, x)\; ca))\; M\; N$

$\rightarrow_{\beta} (\lambda\, c\; a\,.\, c)\; M\; N$

$\rightarrow_{\beta} (\lambda\, a\,.\, M\,))\; N \rightarrow_{\beta} M$

Try out reducing (**if F T F**) on your own now!

and and **or**?

- **and** ≡ $\lambda\, x\, y.(\textbf{if}\ x\ y\ \textbf{F})$

  $\to_\beta \lambda\, x\, y.((\lambda\, p\, c\, a\, .\, p\, c\, a)\ x\ y\ \textbf{F})$

  $\to_\beta \lambda\, x\, y.(x\ y\ \textbf{F})$

  $\to_\beta \lambda\, x\, y.(x\ y\ (\lambda\, u\, v\, .\, v))$

- **or** ≡ $\lambda\, x\, y.(\textbf{if}\ x\ \textbf{T}\ y)$

Defining List

List is *either*

(1) **null**; *or*

(2) a **pair** whose second element is a list.

How to define **null** and **pair** then?

null, null?, pair, first, rest

null? null $\rightarrow_\beta$ **T**

null? (pair $M\ N$ **)** $\rightarrow_\beta$ **F**

first (pair $M\ N$ **)** $\rightarrow_\beta$ M

rest (pair $M\ N$ **)** $\rightarrow_\beta$ N

null and **null?**

- **null** ≡ $\lambda x.$**T**
- **null?** ≡ $\lambda x.(x\ \lambda yz.$**F**$)$

- **null? null** $\rightarrow_\beta$ $(\lambda x.(x\ \lambda yz.$**F**$))\ (\lambda x.$ **T**$)$

  $\rightarrow_\beta$ $(\lambda x.$ **T**$)(\lambda yz.$**F**$)$

  $\rightarrow_\beta$ **T**

pair, first, rest

- A pair (a, b) = "**pair** a b" is represented as

  $\lambda z \, . \, z \, a \, b$
- **pair** ≡ $\lambda x \, y \, z \, . \, z \, x \, y$
- **first** ≡ $\lambda p \, . \, p$ **T**
- **rest** ≡ $\lambda p \, . \, p$ **F**
- **first** (**pair** $M \, N$)

$\rightarrow_\beta$ ($\lambda p \, . \, p$ **T**) (**pair** $M \, N$)

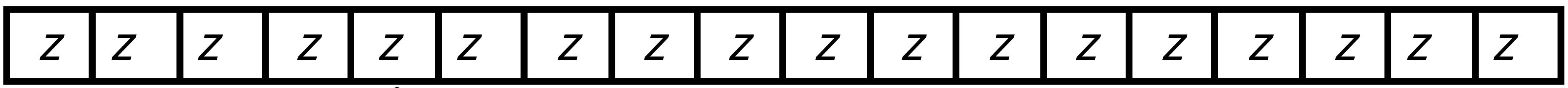
$\rightarrow_\beta$ (**pair** $M \, N$) **T** $\rightarrow_\beta$ ($\lambda z \, . \, z \, M \, N$) **T**

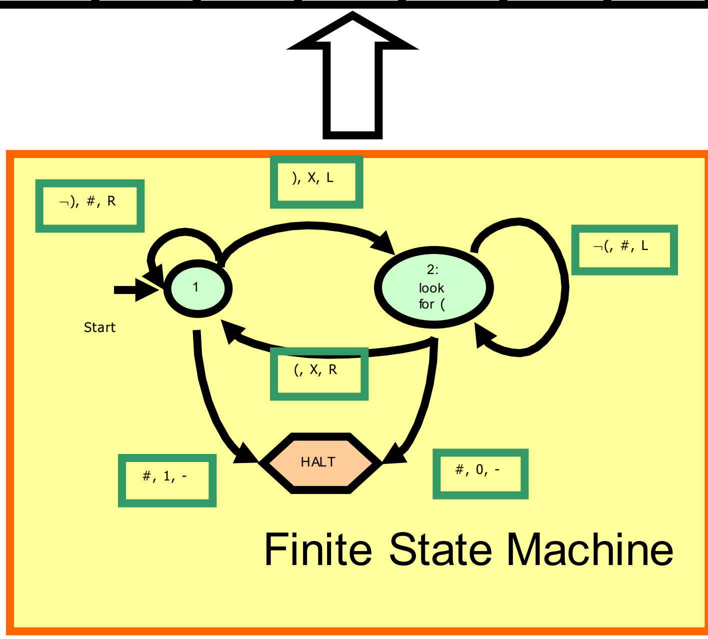
$\rightarrow_\beta$ **T** $M \, N$

$\rightarrow_\beta$ M

The Power of λ Calculus

Able to simulate Universal Turing Machine





Read/Write Infinite Tape

- ✔ **Mutable Lists**

Finite State Machine

- ✔ **Numbers**

Processing

- ✔ **Way to make decisions (if)**
- ✔ **Way to keep going**