A Brief Introduction to λ-calculas

by 丁尧尧

How to express a function ?

Usually we define a function like this:

$$f(x, y) = x - y$$

$$g(x) = e^x$$

Then use it like this:

$$f(5, 1) = 5 - 1 = 4$$

$$g(0) = e^0 = 1$$

Is the name of a function so important ?

We can define the two functions: $f(x, y) = x - y$ and $g(x) = e^x$ like this:

$$(x, y \to x - y)$$

$$(x \to e^x)$$

and use them like this:

$$(x, y \to x - y)(5, 1) = 5 - 1 = 4$$

$$(x \to e^x)(1) = e$$

Is the ability to define functions with more than one parament necessary?

For function:

$$f(x, y) = x - y$$

We can also define a function like this:

$$(x \to (y \to x - y))$$

The function map x to another function ,which maps y to x - y.
(Now we can see the power to write the function itself as the name of it)

We call $(x \to x - 1)$ an **anonymous function**.

And the method that using

$$(x \to (y \to x - y))$$

to replace the function

$$f(x, y) = x - y$$

is called **Currying**

Let's see the formal definition of λ-calculus

Definition(λ-terms)

Assume there is a sequence of expressions $v_0, v_{00}, v_{000}, \ldots$ called **variables**. The set of expression called λ**-terms** is defined as follows:

- all variables are λ-terms (called **atoms**);
- if M and N are any λ-terms, then (MN) is a λ-term (called an **Application**);
- if M is any λ-term and x is any variable, then $(\lambda x.M)$ is a λ-term (called an **Abstraction**).

Examples of λ-term

If x, y, z are any distinc variables, the following are λ-terms:

1. $(\lambda v_0.(v_0 v_{00}))$
2. $(\lambda x.(xy))$
3. $(x(\lambda x.(\lambda x.x)))$
4. $((\lambda y.y)(\lambda x.(xy)))$
5. $(\lambda x.(yz))$

For simplicity, we omit some unnecessary parentheses.

Original expressions	Shorter expressions
$(\lambda x.(\lambda y.(((yx)a)b)))$	$\lambda x.\lambda y.yxab$
$(((\lambda x.(\lambda y.(yx)))a)b)$	$(\lambda x.\lambda y.yx)ab$
$(\lambda x.(\lambda y.((ab)(\lambda z.z))))$	$\lambda x.\lambda y.ab\lambda z.z$

Definition(Free variable, FV)

For any λ-term P, we can define its free variables $FV(P)$ such as:

- $FV(x) = \{x\}$
- $FV(MN) = FV(M) \cup FV(N)$
- $FV(\lambda x.M) = FV(M) - \{x\}$

Examples of free variables

- $FV(\lambda x.\lambda y.xyab) = \{a, b\}$
- $FV(abcd) = \{a, b, c, d\}$
- $FV(xy\lambda y.\lambda x.x) = \{x, y\}$

Definition(Substitution)

For any M, N, x, define $[N/x]M$ as follows:

1. $[N/x]x \equiv N$
2. $[N/x]y \equiv y$
3. $[N/x](PQ) \equiv ([N/x]P[N/x]Q)$
4. $[N/x](\lambda x.P) \equiv \lambda x.P$
5. $[N/x](\lambda y.P) \equiv \lambda y.P \quad (x \notin FV(P))$
6. $[N/x](\lambda y.P) \equiv \lambda y.[N/x]P$
   $(x \in FV(P)$ and $y \notin FV(N))$

7. $[N/x](\lambda y.P) \equiv \lambda[N/x][z/y]P$
$(x \in FV(P)$ and $y \in FV(N))$

Examples of substitution

- $[(uv)/x](\lambda y.x(\lambda w.vwx))$
$\equiv \lambda y.(uv)(\lambda w.vw(uv))$
- $[(\lambda y.vy)/x](y(\lambda v.xv))$
$\equiv y(\lambda w.(\lambda y.vy)w)$

Definition(α-conversion)

Let a λ-term P contain an occurrence of $\lambda x.M$ and let $y \notin FV(M)$. The act of replacing

$$\lambda x.M$$

by

$$\lambda y.[y/x]M$$

is called an α**-conversion.** If P can convert to Q within finite α-conversion, we say P α-convert to Q, noted as $P \equiv_\alpha Q$.

Examples of α-conversion

- $\lambda x.x \equiv \lambda y.y$
- $\lambda x.\lambda y.x(xy) \equiv_\alpha \lambda u.\lambda v.u(uv)$

Definition(β-reducing)

Any term of form $(\lambda x.M)N$ is called a β*-redex* and the corresponding term $[N/x]M$ is called its *contractum*. We call the act that replace the β-redex by its contractum in P a β-contract, noted as

$$P \triangleright_{1\beta} Q$$

If P can β-contracts or α-converses to Q within finite steps, we say P can β-reduce to Q and noted:

$$P \triangleright_{\beta} Q$$

Examples of β-reducing

- $(\lambda x.x(xy))N \quad \triangleright_\beta \quad N(Ny)$
- $(\lambda x.y)N \quad \triangleright_\beta \quad y$
- $(\lambda x.(\lambda y.yx)z)v \quad \triangleright_\beta \quad (\lambda y.yv)z \quad \triangleright_\beta \quad zv$
- $(\lambda x.xx)(\lambda x.xx) \quad \triangleright_\beta \quad (\lambda x.xx)(\lambda x.xx)$
- $(\lambda x.xxy)(\lambda x.xxy) \quad \triangleright_\beta \quad (\lambda x.xxy)(\lambda x.xxy)y$

Definition(β-normal form)

A λ-term Q which contains no β-redexes is called a β**-normal form**. If P can β-reduces to a β-normal form Q, we say Q is a β-normal form of P.

Examples of β-normal form

- zv is a β-normal form of $(\lambda x.(\lambda y.yx)z)v$
- Let $L \equiv (\lambda x.xxy)(\lambda x.xxy)$ and we have
  $$L \quad \rhd \quad Ly \quad \rhd \quad Lyy \quad \rhd \quad \ldots$$
  So L has no β-normal form.
- Let $P \equiv (\lambda u.v)L$.

  1. $P \quad \rhd_\beta \quad v$
  2. $$P \quad \rhd_\beta \quad (\lambda u.v)Ly \quad \rhd_\beta \quad (\lambda u.v)Lyy \quad \rhd_\beta \quad \cdots$$

There are some great results

Fact 1

The relation $\equiv_\alpha$ is an equivalence relation.

Fact 2(Church-Rosser Theorem)

If $P \triangleright_\beta M$ and $P \triangleright_\beta N$, then exist a λ-term T such:

$$M \triangleright_\beta T \quad and \quad N \triangleright_\beta T$$

Fact 3

If we always β-reduce the leftest-outest β-redex and this process can't stop, any order of reduction will not stop.

Why can we say the "computing ability" of β-calculus is equal to the turing machine ?

Code number and basic arithmetic

Name	λ-terms
ZERO	$\lambda f.\lambda x.x$
SUCC	$\lambda n.\lambda f.\lambda x.f\,(n\ f\ x)$
PLUS	$\lambda m.\lambda n.m\,\mathrm{SUCC}\,n$
MULT	$\lambda m.\lambda n.\lambda f.m\,(n\ f)$
POW	$\lambda b.\lambda e.e\,b$
PRED	$\lambda n.\lambda f.\lambda x.n\,(\lambda g.\lambda h.h\,(g\ f))\,(\lambda u.x)\,(\lambda u.u)$
SUB	$\lambda m.\lambda n.n\,\mathrm{PRED}\,m$

Code boolean and basic logic

Name	λ-terms
TRUE	$\lambda x.\lambda y.x$
FALSE	$\lambda x.\lambda y.y$
AND	$\lambda p.\lambda q.pqp$
OR	$\lambda p.\lambda q.ppq$
NOT	$\lambda p.\lambda a.\lambda b.pba$
IF	$\lambda p.\lambda a.\lambda b.pab$

Combination of number and boolean

Name	λ-terms
ISZERO	$\lambda n.n(\lambda x.\mathrm{FALSE})\mathrm{TRUE}$
LEQ	$\lambda m.\lambda n.\mathrm{ISZERO}(\mathrm{SUB}mn)$
EQ	$\lambda m.\lambda n.\mathrm{AND}(\mathrm{LEQ}mn)(\mathrm{LEQ}nm)$

Code repetition (recursion)

The fixed points:

$$Y \equiv (\lambda g.(\lambda x.g(xx))\lambda x.g(xx))$$

Y follows:

$$YF =_\beta F(YF)$$

Example

Now we are going to create the factorial function as example:

$$f(n) = n(n-1)(n-2)\cdots$$

An intuitive idea is:

```
FACT = \n. If (ISZERO n) ONE (MULT n (FACT (PRED n)))
```

But we can't define such term beacuse it's self-recurisive.

Example

The right answer is:

```
FACT2 = \f. \n. IF (ISZERO n) ONE (MULT n (f (PRED n)))
FACT = Y FACT2
```

Example

Let's see what happend when we call "FACT THREE":

```
FACT THREE
= Y FACT2 THREE
= FACT2(Y FACT2) THREE
= IF(ISZERO THREE) ONE (MULT THREE(Y FACT2 TWO))
= MULT THREE(Y FACT2 TWO)
= ...
= MULT THREE (MULT TWO (MULT ONE ONE))
```

Reference

- Lambda-Calculus and Combinators,an Introduction,
  by J. ROGER HINDLEY and JONATHAN P. SELDIN
- 让我们谈谈 λ 演算
  by 王盛颐
- Lambda calculus - Wikipedia

Slide is made by Marp(a markdown editor)

Any question ?