Lambda Calculus

Lambda Calculus
- Some Facts
- Definition
  - Free and Bound Variables
  - Substitutions
- Arithmetic
  - Addition
  - Multiplication

Some Facts

1. The λ calculus consists of a single transformation rule (variable substitution) and a single function definition scheme.
2. The λ calculus is universal that any computable function can be expressed and evaluated using this formalism.
3. The λ calculus is equivalent to Turing Machine. But the λ calculus cares more about the transformation rules and does not care about the actual machine implementation.

Definition

Here we give some definition used in λ calculus.

Name (Variable): A name is an identifier which can be any of the letters $a, b, c, \ldots$.

Expression: The central concept in λ calculus. An expression is defined recursively as follows:

> [expression] := [name] | [function] | [application]
>
> [function] := λ [name].[expression]
>
> [application] := [expression][expression]

Note: The expression can be surrounded with parenthesis with same rule in basic math.

Function: Function is defined in this form:

> [function] := λ [name].[expression]

The name after the λ is the identifier of the argument of this function. The expression after the point is called the body of the definition.

Function can be applied to expressions. Here is an example:

$$(\lambda x.\, x)y = [y/x]x = y \tag{1}$$

$[y/x]$ is used to indicate that all occurrences of x are substituted by y in the expression to the right.

Remark: The names of the arguments in function definitions do not carry any meaning, they just "placeholders". Therefore:

$$(\lambda z.\, z) \equiv (\lambda y.\, y) \equiv (\lambda t.\, t) \equiv (\lambda u.\, u) \equiv \cdots \quad (2)$$

Free and Bound Variables

In λ calculus all names are local to definitions.

Bound variables: In the function $\lambda x.\, x$ we say that x is bound because its occurrence in the body of the definition is preceded by λx.

Free variables: A name not preceded by a λ is called a free variable.

For example:

In the expression

$$(\lambda x.\, xy) \quad (3)$$

the variable x is bound and y is free.

In the expression

$$(\lambda x.\, x)(\lambda y.\, yx) \quad (4)$$

the x in the body of the first expression is bound to the first λ. The y in the body of the second expression is bound to the second λ and x is free. **The x in the second expression is independent of the x in the first expression.**

Formally Definition of Free Variable: We say a variable [name] is free in an expression if one of the following three cases holds:

- [name] is free in [name]
- [name] is free in λ[name1].[expression] if [name] $\neq$ [name1] and [name] free in [expression].
- [name] is free in $E_1 E_2$ if [name] is free in E_1 or if it is free in E_2.

Formally Definition of Bound Variable: A variable [name] is bound if one of two cases holds:

- [name] is bound in λ[name1].[expression] if [name] = [name1] or [name] is bound in [expression]
- [name] is bound in $E_1 E_2$ if [name] is bound in E_1 or if it is bound in E_2.

Note: The same identifier can occur free and bound in the same expression. For example: In the expression

$$(\lambda x.\, xy)(\lambda y.\, y) \quad (5)$$

The first y is free in the left subexpression but bound in the right subexpression. It occurs therefore free as well as bound in the whole expression. (interesting...)

Substitutions

> The more confusing part of standard λ calculus is that we do not give names to functions, i.e. functions are anonymous. Any time we want to apply a function, we write the whole function definition and then precede to evaluate it.

To simplify the notation, we will give function a name. For example, the identity function $(\lambda x.\, x)$ can be denoted with I. We have:

$$\begin{aligned} II &= (\lambda x.\, x)(\lambda x.\, x) = (\lambda x.\, x)(\lambda y.\, y) \\ &= [\lambda y.\, y/x]x = \lambda y.\, y = I \end{aligned}$$

Rename of mixed bound and free variable

Here is a problem: We should be very careful when performing substitutions to avoid mixing up free occurrences of an identifier with bound ones. For example, in the expression

$$(\lambda x.\, (\lambda y.\, xy))y \tag{6}$$

If we do substitution directly, we will get:

$$\lambda y.\, yy \tag{7}$$

Here we mix up these two independent variable y. Simple by renaming the bound y to t we obtain:

$$(\lambda x.\, (\lambda t.\, xt))t = \lambda t.\, yt \tag{8}$$

Therefore, if the function $\lambda x.\, [\text{expression}]$ is applied to E, we substitute all **free** occurrences of x in [expression] with E. If the substitution would bring a free variable of E in an expression where the variable with same name occurs bound, we rename the bound one before performing the substitution.

> *example*
>
> Note that two x bounded by different λx is different. In substitution we only do substitution for the outer x.
>
> $$\begin{aligned} (\lambda x.\, (\lambda y.\, (x(\lambda x.\, xy))))y &=_{\text{rename bound y}} (\lambda x.\, (\lambda t.\, (x(\lambda x.\, xt))))y \\ &= [y/x](\lambda t.\, (x(\lambda x.\, xt))) \\ &=_{\text{note two different x}} (\lambda t.\, (t(\lambda x.\, xt))) \end{aligned}$$

Arithmetic

Numbers can be defined in lambda calculus starting from zero. We define zero as

$$\lambda s.\, (\lambda z.\, z) \tag{9}$$

Or we can abbreviate such expressions with more than one argument as:

$$s(z) = \lambda sz.\, z \tag{10}$$

Then all natural numbers can be defined as

$$
\begin{aligned}
1 &= \lambda sz.\, s(z) \\
2 &= \lambda sz.\, s(s(z)) \\
3 &= \lambda sz.\, s(s(s(z))) \\
&\dots
\end{aligned}
$$

Then the successor function can be defined as

$$S = \lambda wyx.\, y(wyx) \tag{11}$$

The successor function applied to our representation for zero yields

$$
\begin{aligned}
S0 &= (\lambda wyx.\, y(wyx))(\lambda sz.\, z) \\
&= (\lambda yx.\, y((\lambda sz.\, z)yx)) \\
&= \lambda yx.\, y((\lambda z.\, z)x) \\
&= \lambda yx.\, y(x) = 1
\end{aligned}
$$

Similarly we have

$$
\begin{aligned}
S1 &= (\lambda wyx.\, y(wyx))(\lambda sz.\, s(z)) \\
&= (\lambda yx.\, y((\lambda sz.\, s(z))yx)) \\
&= \lambda yx.\, y((\lambda z.\, y(z))x) \\
&= \lambda yx.\, y(y(x)) = 2
\end{aligned}
$$

Addition

If we want to add say n and m, we just apply the successor function n times to m

Let us try the following in order to compute $2+3$:

$$
\begin{aligned}
2S3 &= (\lambda sz.\, s(sz))(\lambda wyx.\, y(wyx))(\lambda uv.\, u(u(uv))) \\
&= (\lambda sz.\, s(sz))S3 \\
&= (\lambda z.\, S(Sz))3 \\
&= S(S3) \\
&= 5
\end{aligned}
$$

> *Note.* 对于丘奇数 n，本身的含义是传入一个函数 s 和一个变量 z，将 z 递归地作用在此 z 上 n 次，这一点在上面的 $2S3$ 上有非常直观的体现，即对于丘奇数2，令$s = S$ 和 $z = 3$， 有
>
> $$2S3 = \lambda sz.\, s(s(z))S3 = S(S(3)) = 5 \tag{12}$$

Multiplication

The multiplication of two numbers x and y can be computed using the following function:

$$\lambda xyz.\, x(yz) \tag{13}$$

The product of 2 by 2 is then:

$$

```
(\lambda xyz.x(yz))22 = (\lambda z.2(2z)) = 4
```

$$

$$(\lambda xyz.\, x(yz))22 = (\lambda z.2(2z)) = 4 \tag{1}$$
$$(\lambda xyz.\, x(yz))22 = (\lambda z.2(2z)) = 4 \tag{14}$$