

# 拟态语言技术手册

类型论

之恪提案

2025 年 11 月 18 日

## 目录

<b>1 无类型的 <math>\lambda</math> 演算</b>	<b>1</b>
1.1 变量 . . . . .	1
1.2 抽象 . . . . .	1
1.3 应用 . . . . .	1
1.4 具体示例 . . . . .	1
1.5 判断一个 $\lambda$ 表达式是否合法 . . . . .	2
1.6 $\lambda$ 达式的定义的展开 . . . . .	2
1.7 作用域和变量遮蔽 . . . . .	2
1.8 判断两个 $\lambda$ 表达式是否等价 . . . . .	2

## 1 无类型的 $\lambda$ 演算

$\lambda$  演算具有三种构造方式，变量、抽象、应用。

### 1.1 变量

变量，即  $x, y, z$  等，一般要求变量的名称以字母或下划线开头，其余部分可以使用字母、下划线和数字。

### 1.2 抽象

抽象，即  $\lambda x. M$ ，其中  $x$  为变量， $M$  为函数体。要求函数体不能为空，可以是其他的变量、抽象、应用。

### 1.3 应用

应用，即  $M N$ ，即将  $M$  应用于  $N$ 。

### 1.4 具体示例

定义： $TRUE = \lambda x. \lambda y. x$ ，这个抽象输入参数  $x$ ，返回另一个函数  $\lambda y. x$ （这个函数接受参数  $y$ ，但忽略它，直接返回  $x$ ）因此， $TRUE$  的行为可以概括为：无论参数  $y$  为何值，永远返回参数  $x$ （选择第一个参数）。

类似的，定义： $FALSE = \lambda x. \lambda y. y$ ，无论参数  $x$  为何值，永远返回参数  $y$ （选择第二个参数）。

我们可以尝试应用  $TRUE$ ，例如：

$$\begin{aligned} & TRUE\ a\ b \\ &= (\lambda x. \lambda y. x)\ a\ b \\ &= (\lambda y. a)\ b \\ &= a \end{aligned}$$

应用的过程中的每一步，可以大体描述为：若有  $\lambda x. M\ y$ ，则它等价于——一个去掉头部的  $\lambda x.$  和参数  $y$ ，将  $M$  中的同一作用域的所有  $x$  替换为  $y$  的  $\lambda$  表达式。

我们将使用“ $\lambda$ ”和“ $.$ ”包裹起来的变量称为绑定变量，例如  $\lambda x.$  中的  $x$  是一个绑定变量。将没有使用“ $\lambda$ ”和“ $.$ ”包裹起来的变量称为自由变量，例如  $a$  是一个自由变量。将一个  $\lambda$  表达式中的“ $($ ”和“ $)$ ”包括起来的内容称为该  $\lambda$  表达式的子表达式，它可以整体作为一个参数传递给绑定变量，也可以作为一个函数接收其他参数。

例子： $\lambda x. \lambda y. x\ y\ a\ b$ ，首先可以发现，第一个参数  $a$  将与  $x$  相绑定，因此去除  $\lambda x.$  和

$a$ ，并将  $\lambda y. x y b$  中的所有  $x$  替换为  $a$ ，即  $\lambda y. a y b$ 。同理，进行第二次替换，可以发现第二个参数  $b$  将与  $y$  相绑定，因此去除  $\lambda y.$  和  $b$ ，并将  $a y$  中的所有  $y$  替换为  $b$ ，就能得到结果  $a b$ 。

### 1.5 判断一个 $\lambda$ 表达式是否合法

判断一个  $\lambda$  表达式是否合法，可以关注以下几点特征：

- 表达式中的变量是否使用了规定的命名规则？
- 每个  $\lambda$  抽象的后方是否都有函数体？
- 绑定变量的声明是否符合语法？

### 1.6 $\lambda$ 达式的定义的展开

由于可以使用标识符定义指代  $\lambda$  表达式，例如  $TRUE = \lambda x. \lambda y. x$ 。因此，定义的展开可以理解为将定义的标识符替换为定义的  $\lambda$  表达式，但是替换时需要在左右分别加上左括号和右括号，表示这个子表达式是一个整体。

例如，定义： $AND = \lambda p. \lambda q. p q FALSE$ ，其等价于： $\lambda p. \lambda q. p q (\lambda x. \lambda y. y)$ 。

如果一个定义中包含其他定义，那么可以继续展开，直到只包含最基础的三种构造方式。

**注意：不允许两个定义形成递归的结构！** 例如一个最简单的递归定义： $A = B, B = A$ ，无论怎么展开，都无法将其展开为最基础的构造方式。并且会无限递归。

### 1.7 作用域和变量遮蔽

$\lambda$  表达式的变量名可以重复，但是可能会产生变量遮蔽。例如： $\lambda x. \lambda x. x$  这个表达式，外层的第一个绑定变量  $x$  被内层的第二个绑定变量  $x$  所遮蔽。导致应用时，第一个绑定变量将无法影响函数体中的  $x$ ，例如  $\lambda x. \lambda x. x a b = b$ 。

也可以认为外部的绑定变量与内部的绑定变量作用域不同，且内部绑定变量的作用域优先级更高。另外，绑定在定义时确定，例如  $x \lambda x.$ ，由于绑定变量  $x$  在自由变量  $x$  之后，因此它们不是同一个变量，第一个是未被绑定的自由变量。

### 1.8 判断两个 $\lambda$ 表达式是否等价

判断两个  $\lambda$  表达式是否等价，可以运用如下的方法：

- 两个  $\lambda$  表达式的形式必须相同，如果把绑定变量、自由变量、子表达式视为基础的元素，那么它们排列的顺序必须一样，个数也要相同。
- 两个  $\lambda$  表达式中的所有相对应的子表达式必须等价，这两个  $\lambda$  表达式才可能等价。

- 两个  $\lambda$  表达式中位置相对应的绑定变量的名称可以不同，但它们的作用必须相同，即都在相对应的位置出现。
- 两个  $\lambda$  表达式中相对应的，且没有在相同作用域绑定过的自由变量必须同名。