Introduction to Lambda Calculus

Topic 1: Some history

Leibniz had as ideal the following.

1. Create a ‘universal language’ in which all possible problems can be stated.
2. Find a decision method to solve all the problems stated in the universal language.

Leibniz的第一个愿望可以用罗素，弗雷格他们的一阶形式逻辑的集合论来fulfill。第二个问题导致了人们有关哲学的思考，人类可以解决世界上能被this universal language表示的所有问题嘛？答案是不能。这个问题叫做Entscheidungsproblem(可判定性问题)。1936年，这个问题独立被Alonzo Church(阿隆佐·邱奇)和Alan Turing(阿兰.图灵)所证明。解决这个问题首先需要define what kind of problem is COMPUTABLE。他们引用了两种不同的model。

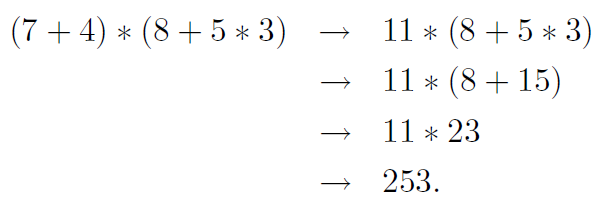
1. Alan Turing, as we know, invented a class of machines called “Turing Machine” and define the notion of computable function via these machines.
2. Church invented a formal system called the “lambda calculus” and define the notion of computable function via this system.

尽管后来Turing证明出两种系统在定义可计算函数上是一样强的，但是这却导致了两种不同的语言的出现。”Imperative programming languages” 命令式编程语言，like C++, Pascal, etc；”Functional programming languages” 函数式编程语言, like Lisp, Scheme, etc. 前者处理方法是基于图灵机的对于a sequence of statements的处理方法。后者是基于由Lambda Calculus理论而产生的Reduction Machine来处理方法。

下面我就给大家简要介绍一下大家不太熟悉的lambda calculus的基本定义和概念。

Topic 2: Basic introduction

Reduction

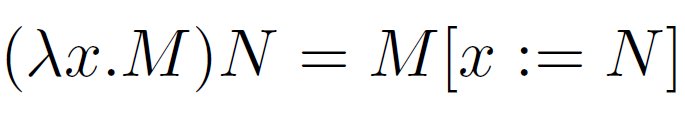


Application & Abstraction------ two basic operation of lambda calculus

Application : FA 表示函数F作用在输入数据A上。Free-type

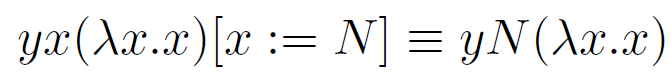
Abstraction: λx. x+2 抽象（要说明的是，这里的加号在没有前提定义时是不合法的）

抽象出一个函数后就可以应用了。

 表示用N来替换M中自由出现的x，进而产生计算结果。

Free and bound variables

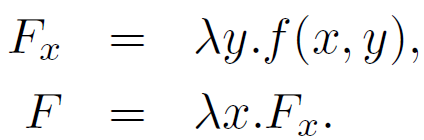
替换只能发生在free variable上。如，

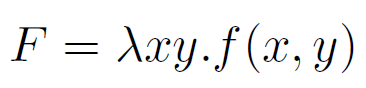


对于bound variable， (alpha-conversion)

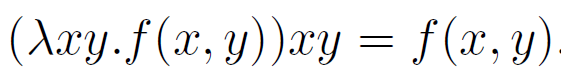
Ex. λf.λx.fx 分析变量的自由出现和绑定情况。

Functions of more arguments



So, 

当你想应用这个抽象出的函数时，你会这么做去得到f(x,y)，对于特定的x,y。



这里需要提到lambda calculus的两个conventions。

1. Application uses left association.



1. Abstraction uses right association.



其实这也是非常自然的。

Topic3: More knowledge about lambda-calculus

下面我们用形式化的语言来介绍lambda演算。

Formalized Definition of lambda-terms (lambda项)

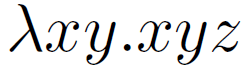
The set of lambda-terms(notation ) is built up from an infinite set of variables  using application and abstraction.

1. 
2. 
3. 

Formalized Definition of Free Variables

The set of free variables of M(M is in), notation FV(M), is defined inductively as follows.

1. 
2. 
3. 

Ex. 

Some Rules

Alpha-conversion alpha-变换

*λx.M=λy.M[x/y]*

其实就是试图形式化的来说明，对于bound variable，variable本身不是重要的。

比如，。

Beta-reduction beta-归约

*(λx.M)N  M[x/N]*

用N来替换M中出现的自由出现的x。要求N中的自由变元在M中也保持自由出现。

Ex. (y.y a)((x. x)(z.(u.u) z))  (y.y a)(z.(u.u) z)

 (y.y a)(z. z) (z. z)a a

以上就是对lambda calculus的一些基本介绍，有兴趣的同学可以去查看其他的书籍来深入学习。

Bibliography:

1. Introduction to Lambda Calculus, Henk Barendregt, Erik Barendsen.
2. 数理逻辑与集合论 第二版 清华大学出版社
3. http://en.wikipedia.org/wiki/Lambda\_calculus