我们研究程序设计语言的概念和特征的主要目的就是对某种适当的详细程度上透过语法的表面去理解程序的语义。我们将使用类型化的lambda演算框架来研究程序设计语言的各种概念,lambda演算是程序设计语言中函数定义和命名约定的基本机制。

这一章将给出一个简单的能对可计算函数进行编程的示例语言, 称之为PCF， 我们将讨论它的语法, 操作语义和程序设计能力。

该语言基于类型化的lambda演算， 以自然数和布尔值(真值)作为基本类型。 PCF可以构造值的对偶(pair)和定义递归函数。

接下来研究纯类型化的lambda演算, 然后给PCF添加栈和树等数据类型。 由于PCF的类型系统并不如我们期望的那样灵活, 我们只考虑带有多态函数(polymorphism)和类型说明的系统。

首先注意一点的是每一个λ项都不具有名字, 在lambda演算中, 一个函数的定义域可以对形参给出的相应类型而给出(指的是该函数可以操作的数),下面我们来看一个简单的例子。

$$\lambda x{:}nat.x$$

上面这个函数就是自然数的等同函数, 记法x:nat指出该函数的定义域是nat, 也是自然数类型， 由于在x:nat的情况下, 函数的值也是x, 所以值域也是nat。

理解λ项的一种方式是通过与其他数方法相比较, 还可以使用$x{:}nat \mapsto x$来描述任何x:nat到其自身的函数，在程序语言中定义等同函数的一种更加通用的方式是写成:

$$Id(x{:}nat){=}x$$

注意这种记法将迫使我们对每一个想写的函数起一个名字, 而使用λ记数的方法可以使得我们直接定义一个函数。

λ抽象的一个重要方面就是λ本身就是一个约束算子, α等价指的是两个函数只是参数名称不同, 如果一个变量仅仅出现在函数的内部, 那么这个函数就是约束的。

在λ记法中, 我们使用下面这样的方式来表示一个函数的作用: λx.x 1,

第一个约定是λ的作用是左结合的, 第二个约定是左边的括号可以被解释成尽可能无限的大, 换句话说, 表达式λx:σ的后面添加一个左括号。而相应的右边括号可以远远的放到只要在其中可以形成语法合适的表达式即可。例如, λx:σ.MN 应该理解为λx:σ.(MN), 而不是(λx:σ.M)N， 同样的λx:σ.λy:τ.MN是加上括号的表达式λx:σ.(λy:τ.(MN))