圈论讨论班 - Lambda 演算 (1)

SOnion snowonionlee@gmail.com

2018-11-17 00:26 啊 Overleaf 上默认按零时区时区来的, 如何 fix 东八区?

0 Contents TODO 生成目录

1 从数学表达式到 λ 表达式

.

作为开场, 本节的内容是 λ 演算在历史上被发明的动机、无类型 λ 演算的语法、以及它的非形式化的语义 (也就是 "直观理解"). 以便引出后续小节的 $\alpha\beta\gamma$ 等价规约变换什么的.

本节的许多内容可以在 `王盛颐_让我们谈谈lambda演算.pdf` 中找到, 因此主要记录讨论班内容提纲 (若有余力, 会区分计划内容和实际内容). 本文中, 代换、β 规约等符号也使用王盛颐 pdf 里的记号 ($[N/x]M$ 表示把 λ 项 M 中自由出现的变量 x 给替换成 λ 项 N 所得到的 λ 项; $\triangleright_{1\beta}$ 是一步 β 规约,$\triangleright_\beta$ 是有限步 β 规约).

1.1 无类型 λ 演算的文法

.

λ 演算所处理的形式语言的句子被称为 λ 项. 首先我们从 (形式上) 最简单的无类型 λ 演算入手, 它的 λ 项有三种形式: $M ::= x \mid \lambda x.M \mid M_1M_2$.

第一种 x 叫变量, 有无穷多的变量 (是可数无穷?), 通常用小写英文字母表示. 第二种 $\lambda x.M$ 叫 λ 抽象. 第三种 M_1M_2 叫 λ 应用. 第二、三种里 x 是变量, M, M_1, M_2 都是 λ 项.

"λ 表达式" 是 "λ 项" 的同义词.

1.2 λ 抽象: 以 $x^3 + 2y$ 为例

.

动机: 定义函数.

$f_m(x) = x^3 + 2y$, $g_m(y) = x^3 + 2y$, $h_m(x, y) = x^3 + 2y$ 不一样的.

$f := \lambda x.x^3 + 2y$, $g := \lambda y.x^3 + 2y$, $h := \lambda x.\lambda y.x^3 + 2y$. 其中 $:=$ 是元语言的记号, 不妨称为 "绑定"(Haskell 的叫法), 把某个 λ 项绑定到某名字上.

约定: λx 约束的 "辖域" 尽可能大. 如 $\lambda x.\lambda y.MN =_{def} \lambda x.(\lambda y.(MN))$ 而非 $\lambda x.(\lambda y.M)N$ 也非 $(\lambda x.\lambda y.M)N$.

讨论班记录: 这个约定引发多次麻烦. 如 (1) 验证 $succ0$ 等于 1 时; (2) 写 ω 组合子的定义时; (3) 讲 Church 1958 年定理 (β 规约终止性判则), 定义 head position 时.

+ J: 有没有唯一可读性定理?

+ 我认为同时要考虑抽象和应用这两种优先级是挺折腾人的.

+ 差点中途使用 (abs x M) 和 (app M1 M2), 这很垠.

不严格地讨论一下 f, g, h 的类型.

$x \in X, y \in Y, x^3 + 2y \in Z$ 则 $h : X \to (Y \to Z)$. 对比数学常用的 $h_m : (X \times Y) \to Z$.

柯里化 (currying) 及其逆映射:

$currying : h_m \mapsto h$, $uncurrying : h \mapsto h_m$.

对集合 A, B, 范畴论里常把 $A \to B$ 的函数写作 B^A(这同时也有避免混淆 "从对象 A 到对象 B 的态射" 和 "从集合 A 到集合 B 的函数" 之功效吧?). 用一下后者符号.

$currying : Z^{X\times Y} \to (Z^Y)^X$. 用范畴语言来看, $X \times Y$ 是两个对象的 product, Z^Y 是 exponential object, currying 是一个 natural isomorphism. (TODO 此处 natural 何解?)

1.3 λ 应用

.

动机: 函数求值、"参数代入".

约定: λ 应用是左结合的. 如 $MNP =_{def} (MN)P$.

变量在 λ 项里的自由出现和约束出现. 类比一阶逻辑的量词 $\forall x, \exists x$.

定义: 不含自由变量的 λ 项叫组合子.

1.3.1 代入的安全性

.

规定元语言记号 $[N/x]M$, 表示把 λ 项 M 中自由出现的变量 x 给替换成 λ 项 N 所得到的 λ 项.

规定: 只有当这个代入安全时才可以这么写.

1.3.2 $x^3 + 2y$ 这东西是 M_1M_2 的形式吗?

.

它可以是. 转换方式不唯一.

add (cube x) (mul 2 y)

add (pow x 3) (mul 2 y)

add cube mul pow 2 3 这些是 λ 演算形式语言里的变量, 还是元语言里的被绑定了 λ 项的名字?

我说不好. 但如果是后者, 可以做到. 一种转换方式是邱奇编码 (Church's Encoding)(特定到自然数及其运算则称为邱奇数):

$1 := \lambda f.\lambda x.fx$

$2 := \lambda f.\lambda x.f(f\ x)$ 即应用两次

$0 := \lambda f.\lambda x.x$ 即应用零次

...

$succ :=$

验证 $succ$ 0"做代入后""等于"1.

加法、乘法、幂

2 α, β, γ × 等价、规约、变换

所谓 "演算" 就是字符串变换规则嘛.

2.1 α 变换、α 等价

约束变量换名.

不唯一.

2.2 β 规约

.

β 可约式 (β-redex).

对 λ 项 P 做一步 β 规约: 把 P 中出现的 β 可约式 $(\lambda x.M)N$ 的一次出现给替换成 $[N/x]M$.

举例

(1) 每步唯一且能终止的

(2) 会不变短的

2.2.1 β 范式 (β normal form)

.

不含 β 可约式的 λ 项.

prop:

2.2.2 β 规约的第 (3) 种情形: 有多种选择

Thm(Church-Rosser)
term rewriting system; confluence
Thm
Thm (1958 Church)
正则序应用序
Haskell 的求值规则 = 正则序 + 记忆化
Thm 不可判定

2.3 η 规约、η 等价

M 与 $\lambda x.Mx$.

3 简单类型 λ 演算

.
主要参考: Basic Proof Theory 第二版 (A.S.Troelstra 等). 1.2 节 Simple type theories.
讨论班记录: 应听众呼声, 这里先把 `https://en.wikipedia.org/wiki/Lambda_cube`画出来了. 本来想晚点画作为结尾. 解释三个方向.

3.1 简单类型

基本类型 $A, B, \cdots$; 函数类型 $T_1 \to T_2, \cdots$.

3.2 简单类型 λ 演算 (记作 $\lambda_\to$) 的 λ 项

3.3 $\alpha, \beta, \gamma \times$ 等价、规约、变换

与无类型的定义相同.

3.4 演算规则

“该演算的内定理是可规约关系和等价关系”.
这里我自己也没弄清楚. TODO

4 自然演绎、柯里-霍华德对应

4.1 自然演绎

.
公理系统、矢列演算、自然演绎是三大类演算系统 (就是把语义扔一边去只关心语形证明的系统).
(`https://sonion.xyz/logic/proof-theory/yu/2017/`)
自然演绎里做推导的三种操作: 假设、使用 $(\to I)$ 之外的规则、使用 $(\to I)$ 规则并关闭假设.
演示: 证 $\alpha \to (\beta \to \alpha)$, $\alpha \to \alpha$,$(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$.

4.2 柯里-霍华德对应

这里说这个对应的 “一个分支”.

4.3 更多的类型

.

参考 Awodey 书 2.5 Examples of products

在简单类型 λ 演算的基础上加一种 type operator(具体地, 是 type constructor): 积类型 (product type). 在编程语言 (Haskell) 里对应二元组.

相当于走到了 lambda cube 的 $\lambda\underline{\omega}$. 当然 $\lambda\underline{\omega}$ 也有不同丰富程度的类型系统. 比如可以再加和类型 (sum type). 在 Haskell 里对应一个类型的多个 data constructor.

CH 对应到合取.

5 Haskell 演示

i,k,s 组合子的类型.

ω 组合子只能在无类型 l 演算中定义. 在 Haskell 里定义不了.

我: Haskell 有类型构造器, 也有多态, 所以类型系统至少是 $\lambda\omega$ 的, 即 $\lambda\underline{\omega}$"加上"$\lambda 2$. 但为什么维基百科说 Haskell 的类型系统 (HM) 是 λ_2(即 System F) 的削弱版, 而不说是 $\lambda\omega$ 的削弱版?...

预告: Y 组合子, 一种不动点组合子, 用于在 "匿名的" λ 演算里实现函数的递归定义 (没名字怎么 call 自己嘛!).

瞎鸡儿分享一番

LaTeX 符号手写识别 `http://detexify.kirelabs.org/classify.html`

E.X.2. 文字下脚料

.

明天大概是 80 分钟的与范畴无直接关系的 lambda 演算 +Haskell+ 柯里霍华德对应, 和 30 分钟的一点点范畴论 (product 那块儿)

1. 无类型 lambda 演算

   许多内容可以在 `王盛颐_让我们谈谈lambda演算.pdf` 中找到.

   1.1. "它用于表示函数" 的想法.

      i. 啊啊啊

2. 简单类型 lambda 演算

   约为 Awodey 书 2.5 Examples of products 的前半, 即 p43-45、pdf60-62.

```
1.1. Simply typed lambda calculus 的记号;
1.2. alpha 等价、beta 规约、eta 变换与 currying/uncurrying.
1.2.1. 几个组合子（不含自由变量的 lambda term）
1.2.2. 运行 Haskell 程序展示上述概念.

2. 柯里-霍华德对应
（约为 Awodey 书 2.5 Examples of products 的后半，即 p46、pdf63）
2.1 自然演绎简介
2.2. （极小？直觉主义？）命题逻辑的{蕴含,合取}片段的自然演绎 与 simply typed lambda calculus 的对应.
2.2.1. 运行 Haskell 程序——特别是利用 GHCi 能够显示 lambda term 的类型的能力——展示柯霍对应.

3.
（约为 Awodey 书 6.6 -calculus（p135 起）和 6.7 Variable sets 的内容，及其必要的前置知识）
TODO. 草稿:

(chapter 6.6.)
```

1）cartesian closed poset with finite joins（即海廷代数）与直觉主义命题逻辑的对应（同构？）
2）上述对应，是 CCC 与 lambda calculus 的对应，在 poset 的特殊情况
3）Def：一个 lambda 演算的_理论_ L.
4）Def：_built from 一个理论 L 的笛卡尔闭范畴_ C(L).
5）Def(6.16.)：L 在范畴 C 里的 _模型_. （lambda 演算的指称语义（之一？））
6）Prop(6.17.)：lambda 演算的理论 L 的两个性质

(chapter 6.7.)
1）We conclude with a special kind of CCC related to the so-called Kripke models of logic, namely ca
一次肯定讲不到这儿吧(F .）先不想了……