STLC 开始的类型论入门

∞-type Café 暑期学校讲座

Alias Qli

July 02, 2023

什么是类型论

狭义类型论: STLC, MLTT, HoTT 等. (一个类型论)

广义类型论: 这些类型论的汇总, 一个学科.

λ 演算的动机

函数定义: 解析式

$$f(x) := M$$

注意: 并不存在 “解析式不是本质的, 集合表示才是”. 类型论里面解析式就是本质的.

x 只在 M 里面有定义:

$$\frac{x \vdash M}{\lambda x.\, M}$$

λ 演算的动机

函数定义: 解析式

$$x \mapsto M$$

注意: 并不存在 “解析式不是本质的, 集合表示才是”. 类型论里面解析式就是本质的.

x 只在 M 里面有定义:

$$\frac{x \vdash M}{\lambda x.\, M}$$

λ 演算的动机

函数定义: 解析式

$$\lambda x.\, M$$

注意: 并不存在 “解析式不是本质的, 集合表示才是”. 类型论里面解析式就是本质的.

x 只在 M 里面有定义:

$$\frac{x \vdash M}{\lambda x.\, M}$$

函数应用: 代入

$$f(x) := M$$

$f(3)$:

$$f(3) = M[x \mapsto 3]$$

函数应用: 代入

$$f(x) := M$$

$$f(3): \qquad x = 3 \text{ 代入 } M$$

$$f(3) = M[x \mapsto 3]$$

函数应用: 代入

$$f(x) := M$$

$f(3)$: 将 M 中的 x 替换成 3

$$f(3) = M[x \mapsto 3]$$

简单类型 λ 演算

Simply Typed Lambda Calculus

简称 STLC

顾名思义, 简单

相继式演算

$$\frac{\text{Premise}_1 \quad \text{Premise}_2 \quad \dots \quad \text{Premise}_n}{\text{Conclusion}}(\text{Name})$$

$$\text{Judgement} \begin{cases} \text{Premise} \\ \text{Conclusion} \end{cases}$$

是关于类型论中对象的命题

类型

什么是类型?

引入判断 A type 表示 “A 是类型”.

首先取一个基本类型的集合 $\mathcal{B} = \{\top, \bot, \mathrm{Ans}\}$.

$$\frac{A \in \mathcal{B}}{A \text{ type}}\ \mathrm{Base}$$

$$\frac{A \text{ type} \quad B \text{ type}}{A \to B \text{ type}}\ \mathrm{Func}$$

$\top$: top
$\bot$: bottom

语境

语境表示 “哪些变量有定义, 类型是什么”.

引入判断 Γ ctx 表示 “Γ 是语境”.

$$\frac{}{\emptyset \text{ ctx}}$$

$$\frac{\Gamma \text{ ctx}}{\Gamma, x : A \text{ ctx}}$$

“变量 : 类型” 二元组组成的反向列表. $\emptyset$ 常常省略.

谓词 $x : A \in \Gamma$ 表示 “x 是 Γ 中类型为 A 的变量”.

项

项: 类型所描述的东西, 那些具有类型的东西.

讨论项不能脱离类型和语境: 引入判断 $\Gamma \vdash M : A$ 表示 “M 是语境 Γ 下类型为 A 的项”.

取一个常量的集合 $\mathcal{C} = \{\mathrm{tt}, \mathrm{yes}, \mathrm{no}\}$.

$$\frac{}{\Gamma \vdash \mathrm{tt} : \top}$$

$$\frac{}{\Gamma \vdash \mathrm{yes} : \mathrm{Ans}} \qquad \frac{}{\Gamma \vdash \mathrm{no} : \mathrm{Ans}}$$

真正重要的规则们:

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \mathrm{Var}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda(x : A).\, M : A \to B}\ \mathrm{Lam}$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash M N : B}\ \mathrm{App}$$

多元函数:

规定 $\to$ 右结合: $A \to B \to C = A \to (B \to C)$

函数应用左结合: $MNR = (MN)R$

$$\dfrac{\dfrac{f : A \to B \to C \quad a : A \quad b : B}{\dots}}{f\,a\,b : C}$$

$$f := \lambda x.\, \lambda y.\, \dots \;=\; \lambda x\ y.\, \dots$$

检查一个项的每一部分是用哪条规则构造出来的称为类型检查 (得名于计算机实现).

$$
\dfrac{\dfrac{\dfrac{\dfrac{f : A \to B \in f : A \to B, x : A}{f : A \to B, x : A \vdash f : A \to B}\,\text{Var} \qquad \dfrac{x : A \in f : A \to B, x : A}{f : A \to B, x : A \vdash x : A}\,\text{Var}}{f : A \to B, x : A \vdash f\ x : B}\,\text{App}}{f : A \to B \vdash \lambda x.\, f\ x : A \to B}\,\text{Lam}}{\emptyset \vdash \lambda f\ x.\, f\ x : (A \to B) \to A \to B}\,\text{Lam}
$$

替换

替换: 将项 M 中的某个变量 x 全部换成另一个项 N, 记作 $M[x \mapsto N]$.

注意: N 里面有个变量 M 正在用, 怎么办?

例子: $(\lambda x.\, y)[y \mapsto x] \stackrel{?}{=} \lambda x.\, x$, 能替换吗?

解决方法: 给 x 改名字: $\lambda x.\, y = \lambda z.\, y$, $(\lambda z.\, y)[y \mapsto x] = \lambda z.\, x$

给函数参数和它的所有使用改名字, 只要这样不造成新的重名: α 转换. 我们认为转换前后的项等价.

总可以通过有限步 α 转换使不同的变量两两不重名.

因此我们可以默认所有变量两两不重名: 替换可以直接换.

归约

归约就是化简. 归约的手段是两条规则, 归约的目标是既约形式.

$$\frac{x : A \vdash M : B \quad N : A}{(\lambda x.\, M)N \equiv M[x \mapsto N] : B}\ \beta\ \text{rule}$$

$$\frac{f : A \to B}{f \equiv \lambda x.\, f\ x : A \to B}\ \eta\ \text{rule}$$

例子:

$$
\begin{aligned}
& (\lambda(f : \mathrm{Ans} \to \mathrm{Ans}).\, f \ \mathrm{yes})(\lambda(x : \mathrm{Ans}).\, x) \\
\equiv\ & (f \ \mathrm{yes})[f \mapsto \lambda(x : \mathrm{Ans}).\, x] && (\beta \text{ reduction}) \\
\equiv\ & (\lambda(x : \mathrm{Ans}).\, x) \ \mathrm{yes} && (\text{substitution}) \\
\equiv\ & x[x \mapsto \mathrm{yes}] && (\beta \text{ reduction}) \\
\equiv\ & \mathrm{yes} && (\text{substitution})
\end{aligned}
$$

总结

- 类型论讨论的是**语境**中有**类型**的**项**
- 一个项是不是良类型的?

  尝试用项的规则构造它/类型检查
- 一个项的最简形式是什么?

  归约.

  方法? β/η 规则

STLC 的模型

模型, 也叫语义, 在计算机里面叫指称语义, 近似于解释器.

思想: 为理论中的每一个东西在模型中找一个对应 (映射, 解释), 使得原本成立的性质都成立 (模型中可以有多出来的性质).

在理论中成立的在每个模型中都成立; 在某个模型里不成立的在理论中不成立.

注意: 命题不成立和命题的否定成立是两个概念, 后者能推出前者.

STLC 的模型: 将类型, 语境, 项分别映射到模型, 使得 α β η 规则在模型里也成立. 惯例: 用同一个记号 $[\![-]\!]$ 表示这三个映射.

STLC 的集合模型: 将大家都解释到集合. 对各自的结构归纳定义.

类型:

$$\llbracket \bot \rrbracket = \emptyset$$
$$\llbracket \top \rrbracket = \{*\}$$
$$\llbracket \mathrm{Ans} \rrbracket = \{0, 1\}$$
$$\llbracket A \to B \rrbracket = \llbracket A \rrbracket \to \llbracket B \rrbracket$$

语境:

$$\llbracket \emptyset \rrbracket = \{*\}$$
$$\llbracket \Gamma, x : A \rrbracket = \llbracket \Gamma \rrbracket \times \llbracket A \rrbracket$$

项 $\Gamma \vdash M : A$ 解释到 $[\![\Gamma]\!] \to [\![A]\!]$ 的函数.

$$[\![\Gamma \vdash \mathrm{tt} : \top]\!] = - \mapsto *$$

$$[\![\Gamma \vdash \mathrm{yes} : \mathrm{Ans}]\!] = - \mapsto 1$$

$$[\![\Gamma \vdash \mathrm{no} : \mathrm{Ans}]\!] = - \mapsto 0$$

$$[\![\Gamma \vdash x : A]\!] = \pi_{i+1} \quad (x : A\text{是}\Gamma\text{中的第}i\text{项})$$

$$\frac{[\![\Gamma \vdash M : A \to B]\!] = m \quad [\![\Gamma \vdash N : A]\!] = n}{[\![\Gamma \vdash MN : B]\!] = \gamma \mapsto m(\gamma)(n(\gamma))}$$

$$\frac{[\![\Gamma, x : A \vdash M : B]\!] = m}{[\![\Gamma \vdash \lambda x.\, M : A \to B]\!] = \gamma \mapsto (x \mapsto m(\gamma, x))}$$

集合模型能证明一些简单的推论.

更复杂的模型能证明更复杂的推论!

模型可以照搬到代码里当解释器.

Bool 类型

Ans 类型的项没法使用: 不存在从 Ans 类型出发的非平凡的常函数.
缺少消去规则!

- 如何构造一个类型的项: 引入规则
- 如何使用一个类型的项: 消去规则

形成规则:

$$\frac{}{\mathbb{B}\ \text{type}}$$

引入规则:

$$\frac{}{\text{true} : \mathbb{B}} \qquad \frac{}{\text{false} : \mathbb{B}}$$

消去规则:

$$\frac{b : \mathbb{B} \quad c_t : C \quad c_f : C}{\text{elim}_{\mathbb{B}}(b, c_t, c_f) : C}$$

计算规则:

$$\frac{c_t : C \quad c_f : C}{\mathrm{elim}_{\mathbb{B}}(\mathrm{true}, c_t, c_f) \equiv c_t : C} \qquad \frac{c_t : C \quad c_f : C}{\mathrm{elim}_{\mathbb{B}}(\mathrm{false}, c_t, c_f) \equiv c_f : C}$$

计算规则描述了消去规则如何抵消引入规则, 反过来, 引入规则如何抵消消去规则呢?

唯一性规则:

$$\frac{b : \mathbb{B} \quad c : \mathbb{B} \to C}{\mathrm{elim}_{\mathbb{B}}(b, c\ \mathrm{true}, c\ \mathrm{false}) \equiv c\, b : C}$$

积类型

元组, 笛卡尔积

形成规则:

$$\frac{A \text{ type} \quad B \text{ type}}{A \times B \text{ type}}$$

引入规则:

$$\frac{a : A \quad b : B}{(a, b) : A \times B}$$

消去规则:

$$\frac{p : A \times B}{\text{fst}(p) : A} \quad \frac{p : A \times B}{\text{snd}(p) : B}$$

计算规则:

$$\frac{a : A \quad b : B}{\mathrm{fst}((a, b)) \equiv a : A} \qquad \frac{a : A \quad b : B}{\mathrm{snd}((a, b)) \equiv b : B}$$

唯一性规则:

$$\frac{p : A \times B}{p \equiv (\mathrm{fst}(p), \mathrm{snd}(p)) : A \times B}$$

余积类型

和类型, 具名联合. 积类型的对偶.

形成规则:

$$\frac{A\ \text{type} \quad B\ \text{type}}{A + B\ \text{type}}$$

引入规则:

$$\frac{a : A}{\text{inl}(a) : A + B} \qquad \frac{b : B}{\text{inr}(b) : A + B}$$

消去规则:

$$\frac{s : A + B \quad c_l : A \to C \quad c_r : B \to C}{\text{elim}_{A+B}(s, c_l, c_r) : C}$$

自然数类型

一个递归的类型!

引入规则:

$$\frac{}{\mathrm{zero} : \mathbb{N}} \qquad \frac{n : \mathbb{N}}{\mathrm{suc}(n) : \mathbb{N}}$$

消去规则:

$$\frac{n : \mathbb{N} \quad c_0 : C \quad c_s : \mathbb{N} \to C \to C}{\mathrm{elim}_{\mathbb{N}}(n, c_0, c_s) : C}$$

计算规则:

$$\frac{c_0 : C \quad c_s : \mathbb{N} \to C \to C}{\mathrm{elim}_{\mathbb{N}}(\mathrm{zero}, c_0, c_s) \equiv c_0 : C}$$

$$\frac{n : \mathbb{N} \quad c_0 : C \quad c_s : \mathbb{N} \to C \to C}{\mathrm{elim}_{\mathbb{N}}(\mathrm{suc}(n), c_0, c_s) \equiv c_s\, n\, \mathrm{elim}_{\mathbb{N}}(n, c_0, c_s) : C}$$

$$\mathrm{elim}_{\mathbb{N}}(n, c_0, c_s) \equiv \underbrace{c_s(n-1)(c_s(n-2)...(c_s\ \mathrm{zero}}_{n \text{ times}}\ c_0))$$

模式匹配

消去规则的另一种语法: 对引入规则分情况讨论

$$\begin{aligned}
\mathrm{elim}_{\mathbb{B}}(b, c_t, c_f) \qquad & \text{case } b \text{ of} \\
& \quad \text{true} \to c_t \\
& \quad \text{false} \to c_f \\
f\ \ \mathrm{fst}(p)\ \mathrm{snd}(p) \qquad & \text{case } p \text{ of} \\
& \quad (a, b) \to f\ a\ b \\
\mathrm{elim}_{A+B}(s, c_l, c_r) \qquad & \text{case } s \text{ of} \\
& \quad \mathrm{inl}(a) \to c_l\ a \\
& \quad \mathrm{inr}(b) \to c_r\ b
\end{aligned}$$

自然数的模式匹配

$$f := \lambda n.\, \mathrm{elim}_{\mathrm{Nat}}(n, c_0, c_s) \qquad \begin{aligned} f\ \ \mathrm{zero} \quad\ \ &:= c_0 \\ f\ \ \mathrm{suc}(m) &:= c_s\ m\ (f\ m) \end{aligned}$$

定义自然数加法试试看:

$$\begin{aligned} &\mathrm{add} : \mathbb{N} \to \mathbb{N} \to \mathbb{N} \\ &\mathrm{add}\ m\ n := \mathrm{elim}_{\mathbb{N}}(m, n, \lambda\_.\, \lambda n.\, \mathrm{suc}(n)) \end{aligned}$$

$$\begin{aligned} &\mathrm{add}\ \mathrm{zero} \qquad n := n \\ &\mathrm{add}\ \mathrm{suc}(m)\ n := \mathrm{suc}(\mathrm{add}\ m\ n) \end{aligned}$$

我们允许任意递归吗? 停机性检查

函数类型的 5 种规则

形成规则:

$$\frac{A \text{ type} \quad B \text{ type}}{A \to B \text{ type}}$$

引入规则:

$$\frac{x : A \vdash M : B}{\lambda x.\, M : A \to B}\ \text{Lam}$$

消去规则:

$$\frac{M : A \to B \quad N : A}{MN : B}\ \text{App}$$

计算规则:

$$\frac{x : A \vdash M : B \quad N : A}{(\lambda x.\, M)N \equiv M[x \mapsto N] : B}\ \beta$$

唯一性规则:

$$\frac{f : A \to B}{\lambda x.\, f\ x \equiv f : A \to B}\ \eta$$

这些规则就是我们对于函数的要求.

特殊的类型

太简单了, 没有完整的 4 种规则

$\top$ 只有引入规则和计算规则:

$$\frac{}{\mathrm{tt} : \top} \qquad \frac{e : \top}{e \equiv \mathrm{tt} : \top}$$

$\bot$ 只有消去规则和唯一性规则:

$$\frac{e : \bot}{\mathrm{elim}_{\bot}(e) : C} \qquad \frac{e : \bot \quad c : C}{\mathrm{elim}_{\bot}(e) \equiv c : C}$$

Curry–Howard 同构

直觉主义自然演绎: 不包含排中律的命题逻辑系统.

记 Γ 为一个命题的有限集合, 称为假设 (或公理). 定义仅在下列情况中, 命题 P 可由 Γ 推出, 记为 $\Gamma \vdash P$:

- P 是 Γ 中的一个命题: 引入假设或使用公理

$$\frac{P \in \Gamma}{\Gamma \vdash P}\ \mathrm{Ax}$$

- P 具有 $A \to B$ 的形式, 并且 $\Gamma, A \vdash B$: 演绎定理

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \to I$$

- 存在 A 使得有 $\Gamma \vdash A \to P$ 和 $\Gamma \vdash A$: 肯定前件

$$\frac{\Gamma \vdash A \to P \quad \Gamma \vdash A}{\Gamma \vdash P} \to E$$

这东西怎么和 Var, Lam, App 这么像? 不是偶然

$$\boxed{\frac{P \in \Gamma}{\Gamma \vdash P}\ \mathrm{Ax}}$$

- 函数类型对应蕴含
- 类型对应命题
- 项对应证明

类型论就对应逻辑系统. 这就是 CH 同构.

证明 $(P \to Q) \to (Q \to R) \to P \to R$:

$$\text{proof} : (P \to Q) \to (Q \to R) \to P \to R$$

$$\text{proof} := \lambda f\ g\ x.\ g\ (f\ x)$$

这个对应能推得更广:

- $\top$ 对应真命题, tt 是它的证明.
- $\bot$ 对应假命题, $\mathrm{elim}_\bot$ 是谎言爆炸原理.
- $A \times B$ 对应 $A \wedge B$: 如果命题 A 有证明 a, 命题 B 有证明 b, 那么 $A \wedge B$ 有证明 (a, b).
- $A + B$ 对应 $A \vee B$: 如果命题 A 有证明 a, 那么 $A \vee B$ 有证明 $\mathrm{inl}(a)$; 如果命题 B 有证明 b, 那么 $A \vee B$ 有证明 $\mathrm{inr}(b)$.

“P 的否定” 就是 “P 能推出假”.

$A \to B \to A + B$ 有两个项, 对应两个证明?

是的

更高级的类型论里面能够精确表达 “命题” 的概念

直觉主义逻辑

直觉主义逻辑讲究构造

还能做数学吗? 能

Godel 嵌入

总结

- 研究类型论性质的方法: 构造模型
- 更多类型: Bool 类型, 积类型, 余积类型, 自然数类型
- 形成规则, 引入规则, 消去规则 (模式匹配), 计算规则, 唯一性规则
- CH 同构, 直觉主义