猫猫类型论

千里冰封
https://live.bilibili.com/937724

前面部分的前置知识

- Typed lambda calculus
  - 至少李姐 substitution、context、unit type、product type 等概念
- 最好熟悉范畴、态射等基本概念的定义
  - 这样可以跳一部分例子
  - 但没有的话 也不是不可以
- 最好对代数有一定了解
  - 这样距离群同态的时候就可以听懂了，拓扑空 间与连续函数同理
  - 但没有的话 也不是不可以

什么是类型论

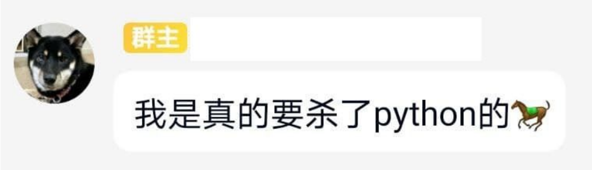
- 就是关于编程语言中的「类型」的代数理论
- 类型论的研究就是对所有静态类型编程语言的研究
- 类型论是一种结构证明论，它讨论语法（编程语言的语法）和模型（编程语言的语义）

2017年07月06日

如何掌握所有的程序语言

类型论有什么用

- 帮助我们河狸地设计编程语言
- 可以把数学工具用在编程语言的研究上
- 抽象的大锤：范畴论
  - 祖与占写的知乎回答 👇



范畴论相关的 topic 有 @parker liu 的 2-category 和 @老废物千里冰封 的 <在范畴论里解释类型论>, 虽然我都没看得很懂, 但是我大受震撼.gif. 但我觉得介绍 motivation 方面冰封相对于 Parker 来说做得更好, 用范畴论 “modeling” 类型论的 motivation 就是很数学家的套路: 一旦我们能把类型论套路进范畴论, 我们就可以继续挥舞抽象~~废话~~的大锤了! (IIRC 现在越来越多数学领域被范畴化, 范畴论算是通用建模语言?). 当然凭我浅薄的理解水平在有限的时间里是 get 不到什么有意思的东西的.

为什么要用范畴论

- 范畴论抽象了代数理论
  - 类型论也是一种代数理论
- 近年的类型论研究中有很多都基于范畴模型
  - Cartmell, Jacobs, Dybjer, Pitts, Hoffman, Curien, etc.
  - Lumsdaine, Coquand, Shulman, Voevodsky, etc.
  - 同伦类型论的发展中，范畴模型是不可或缺的一 环
- 因为猫猫很可爱(帕瓦也很可爱！)

自我介绍和讲义

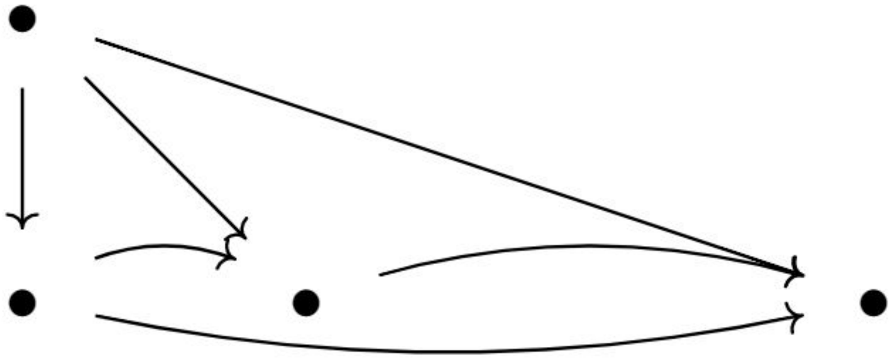
- 我是千里冰封，是个伞兵，会点编程和编程语言理论，正在学习范畴论
- 讲义地址：https://personal.psu.edu/yqz5714/cwa.pdf 目前正在持续更新
  - 欢迎 dalao 纠错
  - 感谢 Alice 聚聚纠错。Alice，我的超人。

关于上次函数式编程大会

- 之前就这个话题讲过一次，但是讲得 hin 失败，给的定义有很多是错的，这次我已经悉数纠正了，希望听众也能注意一下我可能的错误
- 上次讲义里用 TikZ 画的猫猫👇

什么是范畴

- 对象和态射，交换图，函子，自然变换
- 举例



符号约定

Notation 1.2. We introduce some notational conventions in category theory.

- We write $A \in \mathcal{C}$ to say that "A is an object in the category $\mathcal{C}$", and $f \in \mathcal{C}(A, B)$ to say that "f is a morphism in the category $\mathcal{C}$ from A towards B". In the literature, the former is also written as $A \in \mathrm{Ob}(\mathcal{C})$ and the latter is also written as $f \in \mathrm{Hom}_{\mathcal{C}}(A, B)$.
- We write $\mathrm{Ob}(\mathcal{C})$ for the set of objects in $\mathcal{C}$.
- We write $\mathrm{id}_A \in \mathcal{C}(A, A)$ for the identity morphism, $dom(f)$ and $cod(f)$ for the domain projection and the codomain projection of morphisms. We also say a morphism f is a morphism *from* $dom(f)$ and *towards* $cod(f)$.
- We write $f \circ g \in \mathcal{C}(dom(g), cod(f))$ for composition of morphisms.

积对象

- 「万有性质」

Definition 2.11 (Product). For a category $\mathcal{C}$ and $A, B \in \mathcal{C}$, the *product object* of A and B, denoted $(A \times B) \in \mathcal{C}$, is an object in $\mathcal{C}$ equipped with two morphisms $\pi_1 \in \mathcal{C}(A \times B, A)$ and $\pi_2 \in \mathcal{C}(A \times B, B)$ such that for every object $D \in \mathcal{C}$ with two morphisms $f_1 \in \mathcal{C}(D, A)$ and $f_2 \in \mathcal{C}(D, B)$, there is a unique morphism in $\mathcal{C}(D, A \times B)$ that commutes the following diagram:

$$\begin{tikzcd}
 & D \arrow[dl, "f_1"'] \arrow[d, dashed] \arrow[dr, "f_2"] & \\
A & A \times B \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & B
\end{tikzcd}$$

拉回

$$\begin{tikzcd}
E \arrow[r, dashed] \arrow[rr, bend left, "b'"] \arrow[dr, "a'"'] & D \arrow[r, "b"'] \arrow[d, "a"] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d, "g"] \\
& A \arrow[r, "f"'] & C
\end{tikzcd}$$

Lemma 2.19. *In a category $\mathcal{C}$ with all pullbacks (definition 2.16) and a terminal object (definition 2.9) $1 \in \mathcal{C}$, for every $A, B \in \mathcal{C}$ we have $A \times_1 B = A \times B$.*

Proof. We prove by constructing both commutative squares:

$$\begin{tikzcd}
A \times_1 B \arrow[r, dashed] \arrow[d, dashed] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d, "1"] \\
A \arrow[r, "1"'] & 1
\end{tikzcd}
\qquad
\begin{tikzcd}
A \times B \arrow[r, dashed] \arrow[d, dashed] & B \\
A &
\end{tikzcd}$$

类型论长啥样

- 语境：一组类型，用希腊字母 Γ 表示，里面包含的是带类型的变量绑定
- 我们还需要如下推理规则形式：

$$\Gamma \vdash A \text{ type}$$

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type}}{\Gamma \vdash u : A} \qquad \frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type} \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : A}{\Gamma \vdash u = v : A}$$

这其实是「依值」类型论

- 在更简单的类型论中（比如 Java 的类型系统），我们不在一个语境中讨论类型。语法合法的类型就是好的。
  - 不像表达式，表达式我 们从来都是在 语境中讨论的。语法合法的表达式，比如 $(\lambda x.\ x) + 1$，不一定是好的。
- 依值类型论中，我们可以描述更多的类型，因为在描述类型的时候我们可以用到更丰富的信息

开始思考：类型论应该对应什么样的范畴

- 从这个规则入手
  - 如果 Γ 是一个长度为一的语境，那么 u 就是一个和两个类型关联起来的值
  - 复合：给定两个值
    - `x: Int |- toString(x) : String`
    - `x: String |- length(x): Int`
    - 这俩可以互相复合，和态射复合很像
  - 能不能把这个规则推广到 Γ 长度不为一的情况呢？

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \ \mathrm{type}}{\Gamma \vdash u : A}$$

解决方法：一次讨论一组值

- 注意符号的使用

$$\frac{\Gamma \vdash \qquad \Delta \vdash}{\Gamma \vdash \sigma : \Delta}$$

It reads "in the context Γ, the *i*-th term in σ have the *i*-th type in Δ, presupposing $\Gamma \vdash, \Delta \vdash$". It is assumed that σ and Δ have the same length.

- Consider a substitution object $\Gamma \vdash \sigma : \Delta$ (a judgment already described in remark 2.6): it is a fact that "terms in σ can refer to bindings in Γ", and we say "σ *instantiates* the context Δ".
- Consider a term $\Delta \vdash u : A$, both u and A can refer to bindings in Δ.

We can turn u into another term, typed in the context Γ by applying the substitution σ since the open references to Δ can be replaced by the terms in σ, which can refer to bindings in Γ. This process can be described by the following deduction:

$$\frac{\dfrac{\Gamma \vdash \qquad \Delta \vdash}{\Gamma \vdash \sigma : \Delta} \qquad \dfrac{\Delta \vdash A \text{ type}}{\Delta \vdash u : A}}{\Gamma \vdash A\sigma \text{ type} \qquad \text{and} \qquad \Gamma \vdash u\sigma : A\sigma}$$

Similarly, for contexts $\Gamma, \Gamma', \Gamma''$ and substitutions objects σ, γ, there is:

$$\frac{\dfrac{\Gamma \vdash \qquad \Gamma' \vdash}{\Gamma \vdash \sigma : \Gamma'} \qquad \dfrac{\Gamma' \vdash \qquad \Gamma'' \vdash}{\Gamma' \vdash \gamma : \Gamma''}}{\Gamma \vdash \gamma\sigma : \Gamma''}$$

剩下的问题

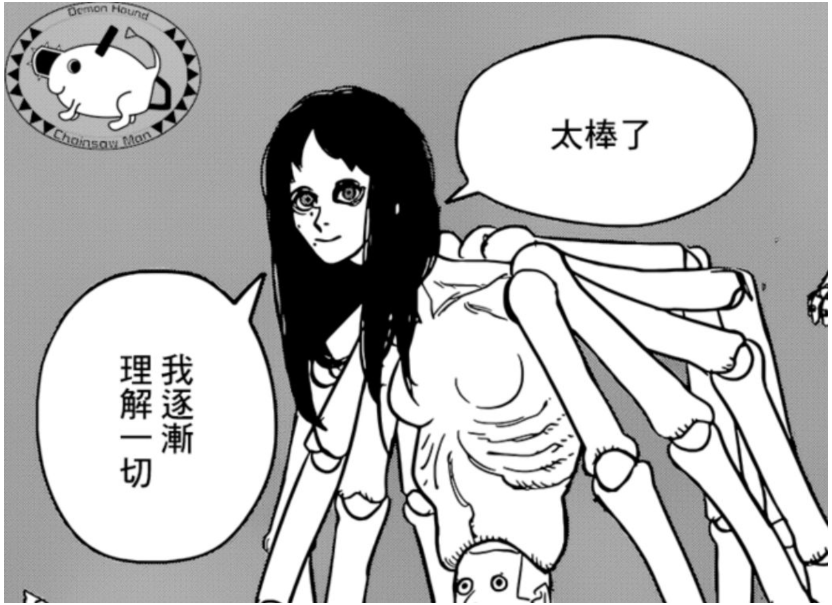
- 如何讨论单个类型？
  - 思想：

$$\boxed{\Gamma \vdash A \ \mathrm{type}} \ \text{as} \ \boxed{\Gamma, A \vdash}$$

  - 定义一个操作 π 用来把一个类型所在的语境抽出来
  - 每个类型对应一个这样的 π 操作
- 如何讨论单个值？
  - 就是 π 的逆

我们现在有一个范畴了

- 每个类型论里的构造，都「应该」一个对应的范畴里的构造
  - 那么我们可以在范畴论里讨论类型论了



对一个类型进行一组替换

$$
\begin{tikzcd}
\Delta, \sigma^* A \arrow[r, "\sigma^A"] \arrow[d, "\pi_{\sigma^* A}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \Gamma, A \arrow[d, "\pi_A"] \\
\Delta \arrow[r, "\sigma"'] & \Gamma
\end{tikzcd}
$$

终结对象就是 unit type

$$\frac{}{\vdash \top\ \text{type}} \qquad \frac{}{\vdash \star : \top} \qquad \frac{\Gamma \vdash \qquad \Gamma \vdash u : \top}{\Gamma \vdash u = \star : \top}$$

$$\begin{tikzcd}
1, [\![\top]\!] \arrow[rr, equal, "\pi_1"] \arrow[d, dashed] \arrow[dr, phantom, "\lrcorner", very near start] & & 1 \arrow[ll, dashed, bend right, "{[\![\star]\!]}"'] \arrow[d] \\
\Delta, A \arrow[rr, "\pi_A"] & & \Delta
\end{tikzcd}$$

积类型是啥？

积类型就是如下类型

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma \vdash (A \times B) \text{ type}}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash (A \times B) \text{ type} \qquad \Gamma \vdash u : A \times B}{\Gamma \vdash u.1 : A \qquad \text{and} \qquad \Gamma \vdash u.2 : B}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : B}{\Gamma \vdash \langle u, v \rangle : A \times B}$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type} \qquad \Gamma \vdash u : A \qquad \Gamma \vdash v : B}{\Gamma \vdash \langle u, v \rangle.1 = u : A \qquad \text{and} \qquad \Gamma \vdash \langle u, v \rangle.2 = v : B}$$

我们搞一个这样的交换图

$$
\begin{tikzcd}
\llbracket \Gamma \rrbracket, \llbracket A \times B \rrbracket \arrow[r, dashed] \arrow[d, dashed] \arrow[dr, phantom, "\lrcorner", very near start] & \llbracket \Gamma \rrbracket, \llbracket A \rrbracket \\
\llbracket \Gamma \rrbracket, \llbracket B \rrbracket & \llbracket \Gamma \rrbracket \arrow[l, "\llbracket v \rrbracket"'] \arrow[u, "\llbracket u \rrbracket"']
\end{tikzcd}
$$

积类型就是一个拉回

$$
\begin{tikzcd}
{[\![\Gamma]\!], [\![A \times B]\!]} \arrow[r, dashed] \arrow[d, dashed] \arrow[dr, phantom, very near start, "\lrcorner"] \arrow[dr, "\pi_{[\![A \times B]\!]}"] & {[\![\Gamma]\!], [\![B]\!]} \arrow[d, "\pi_{[\![B]\!]}"] \\
{[\![\Gamma]\!], [\![A]\!]} \arrow[r, "\pi_{[\![A]\!]}"] & {[\![\Gamma]\!]}
\end{tikzcd}
$$

$$
\begin{tikzcd}
{[\![\Gamma]\!], [\![A \times B]\!]} \arrow[r, dashed, "\mathrm{elim}_2"] \arrow[d, dashed, "\mathrm{elim}_1"'] \arrow[dr, phantom, very near start, "\lrcorner"] & {[\![\Gamma]\!], [\![A]\!]} \\
{[\![\Gamma]\!], [\![B]\!]} & {[\![\Gamma]\!]} \arrow[l, "{[\![v]\!]}"] \arrow[u, "{[\![u]\!]}"'] \arrow[ul, "{[\![\langle u,v \rangle]\!]}"]
\end{tikzcd}
$$

发现新定理

$$
\begin{tikzcd}
\llbracket \Gamma \rrbracket, \llbracket A \times B \rrbracket \arrow[rr, dashed, "\mathrm{elim}_2"] \arrow[d, dashed, "\mathrm{elim}_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & & \llbracket \Gamma \rrbracket, \llbracket A \rrbracket \\
\llbracket \Gamma \rrbracket, \llbracket B \rrbracket & & \llbracket \Gamma \rrbracket \arrow[ll, "\llbracket t.2 \rrbracket"] \arrow[u, "\llbracket t.1 \rrbracket"'] \arrow[ull, "\llbracket t \rrbracket"']
\end{tikzcd}
$$

$$
\frac{\Gamma \vdash (A \times B)\ \mathit{type} \qquad \Gamma \vdash t : A \times B}{\Gamma \vdash t = \langle t.1, t.2 \rangle : A \times B}
$$

所以在范畴论里面解释类型论有什么用

- 我们可以用范畴论里已有的构造去解释一些类型
- 然后可以发现一些类型论里面证明不出来的性质
- 我们可以在范畴论里面证明类型论的**元性质**！

等号类型是啥？

我这里使用「外延相等类型」

- 我背叛了马丁洛夫
- 这个相等类型很少见
  - 我仅在 Nuprl 等奇怪的软件里见过
  - 因为它会破坏类型检查的可判定性
  - 但我们是猫猫人，猫猫不需要可判定性

Definition 6.8 (Id). We extend definition 4.1 with the following type:

$$\frac{\Gamma \vdash \qquad \Gamma \vdash A\ \text{type} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash (\mathrm{Id}_A\ a\ b)\ \text{type}} \qquad \frac{\Gamma \vdash A\ \text{type} \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : \mathrm{Id}_A\ a\ a}$$

$$\frac{\Gamma \vdash A\ \text{type} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma \vdash p : \mathrm{Id}_A\ a\ b}{\Gamma \vdash a = b : A}$$

This type is known as the *extensional equality type*, in contrast to the *intensional equality type*. We will only talk about the extensional one for convenience. ▲

首先你需要知道什么是单态射

Definition 6.1 (Monic). We say a morphism $f \in \mathcal{C}(A, B)$ to be a *monomorphism*, a *mono*, or *monic* if for every $C \in \mathcal{C}$ and $g_1, g_2 \in \mathcal{C}(C, A)$ such that:

$$f \circ g_1 = f \circ g_2 \implies g_1 = g_2$$

Holds. A monomorphism is also known as a *left-cancallative* morphism. The above equality is visualized below:

$$\begin{tikzcd} C \arrow[r, shift left, "g_1"] \arrow[r, shift right, "g_2"'] & A \arrow[r, "f"] & B \end{tikzcd}$$

Demonstration 6.2 (Injection). In **Set**, a morphism is monic (definition 6.1) if and only if it is an injective function.

单态射的基本性质

Lemma 6.4 (MonoIso). *If for a mono (definitions 6.1 and 6.3) f, there exist a morphism g such that $f \circ g = id_{cod(f)}$, then f is an isomorphism.*

Proof. $f \circ g \circ f = \mathrm{id}_{cod(f)} \circ f = f = f \circ \mathrm{id}_{dom(f)}$ and by the definition of mono and the associativity of composition $g \circ f = \mathrm{id}_{dom(f)}$. Thus f an isomorphism. □

Remark 6.5. There is an intuitive justification of lemma 6.4: in **Set**, invertible and injective (see demonstration 6.2) functions are isomorphisms. ✐

Lemma 6.6 (IsoMono). *If for an isomorphism f, there exist morphisms g_1, g_2 such that $g_1 \circ f = g_2 \circ f$, then $g_1 = g_2$.*

Proof. $g_1 = g_1 \circ f \circ f^{-1} = g_2 \circ f \circ f^{-1} = g_2$. Note that this lemma is converse to lemma 6.4. □

「等化子」

Definition 6.10 (Equalizer)**.** In a category $\mathcal{C}$ and for $f, g \in \mathcal{C}(X, Y)$ (we refer to pairs of morphisms like this as *parallel morphisms*), there might be some objects $E \in \mathcal{C}$ together with morphisms of form $e \in \mathcal{C}(E, X)$ commuting the following diagram (so that $f \circ e = g \circ e \in \mathcal{C}(E, Y)$):

$$\begin{tikzcd} E \arrow[r, "e"] & X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y \end{tikzcd}$$

We take the terminal object of the full subcategory (definitions 3.31 and 3.32) of $\mathcal{C}_{/X}$ commuting the above diagram which consists of morphisms like e, and refer to the object E (denoted $\mathrm{Eq}(f, g)$) and the morphism e (denoted $\mathrm{eq}(f, g)$) as the *equalizer* of f and g. We bring these notations into the diagram above:

$$\begin{tikzcd} \mathrm{Eq}(f, g) \arrow[r, dashed, "\mathrm{eq}(f,g)"] & X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y \end{tikzcd}$$

In case each parallel morphisms in $\mathcal{C}$ has an equalizer, we say that $\mathcal{C}$ *has all equalizers*. The definition of equalizers is from [EH63]. ▲

相等类型就是等化子，等式证明就是等化子的逆

Definition 6.14. We interpret the extensional identity type in a contextual category (definition 3.48) as an equalizer (definition 6.10). The formation of the *context* Γ *extended by* $\mathrm{Id}_A\ a\ b$ (denoted $(\llbracket\Gamma\rrbracket, \llbracket\mathrm{Id}_A\ a\ b\rrbracket) \in \mathcal{C}$) is by taking the equalizer $\mathrm{Eq}(\llbracket a\rrbracket, \llbracket b\rrbracket)$, where the display map (definition 3.20) is given by the equalizer: $\pi_{\llbracket\mathrm{Id}_A\ a\ b\rrbracket} = \mathrm{eq}(\llbracket a\rrbracket, \llbracket b\rrbracket)$:

$$\llbracket\Gamma\rrbracket, \llbracket A\rrbracket \underset{\llbracket b\rrbracket}{\overset{\llbracket a\rrbracket}{\leftleftarrows}} \llbracket\Gamma\rrbracket \xleftarrow{\mathrm{eq}(\llbracket a\rrbracket,\llbracket b\rrbracket)} \llbracket\Gamma\rrbracket, \llbracket\mathrm{Id}_A\ a\ b\rrbracket$$

By lemma 6.13, $\mathrm{eq}(\llbracket a\rrbracket, \llbracket a\rrbracket) = \mathrm{id}_{\llbracket\Gamma\rrbracket}$. So, we define $\llbracket\mathsf{refl}_a\rrbracket = \mathrm{id}_{\llbracket\Gamma\rrbracket}$:

$$\llbracket\Gamma\rrbracket, \llbracket A\rrbracket \xleftarrow{\llbracket a\rrbracket} \llbracket\Gamma\rrbracket \overset{\llbracket\mathsf{refl}_a\rrbracket}{=\!=} \llbracket\Gamma\rrbracket, \llbracket\mathrm{Id}_A\ a\ a\rrbracket$$

In case we have a morphism f corresponding to an instance of the identity type, it is an inverse to the equalizer ($\mathrm{eq}(\llbracket a\rrbracket, \llbracket b\rrbracket) \circ f = \mathrm{id}_{\llbracket\Gamma\rrbracket, \llbracket\mathrm{Id}_A\ a\ b\rrbracket}$), and by lemma 6.4 f is an isomorphism. Then, by lemma 6.6 $\llbracket a\rrbracket = \llbracket b\rrbracket$ holds, and that makes sense of the elimination rule:

$$\begin{tikzcd} \llbracket\Gamma\rrbracket, \llbracket A\rrbracket \arrow[r, shift right=3, "\pi_{\llbracket A\rrbracket}"] & \llbracket\Gamma\rrbracket \arrow[l, shift left=1, "\llbracket a\rrbracket"'] \arrow[l, shift right=5, "\llbracket b\rrbracket"] \arrow[r, dashed, bend right, "f"'] & \llbracket\Gamma\rrbracket, \llbracket\mathrm{Id}_A\ a\ b\rrbracket \arrow[l, dashed, bend right, "\mathrm{eq}(\llbracket a\rrbracket,\llbracket b\rrbracket)"'] \end{tikzcd}$$

$\blacktriangle$

我们得到了什么新的定理？

- 等化子的逆是唯一的，所以类型论中外延等号类型的实例也是唯一的！
  - 热 (HoTT) 人 震 怒

Lemma 6.15 (UIP). *The interpretation in definition 6.14 gives rise to the following "uniqueness of the identity proof" rule:*

$$\frac{\Gamma \vdash A\ type \qquad \Gamma \vdash a : A \qquad \Gamma \vdash p : \mathrm{Id}_A\ a\ a}{\Gamma \vdash p = refl_a : \mathrm{Id}_A\ a\ a}$$

不仅如此，我们还能证这个

- 而且还能提供两种不同的证明视角！

Theorema 6.16 (Uniqueness). *The interpretation in definition 6.14 gives rise to the following "uniqueness rule" of the identity type:*

$$\frac{\begin{array}{ccc} \Gamma \vdash A \ \mathit{type} & \Gamma \vdash a : A & \Gamma \vdash b : A \\ \multicolumn{3}{c}{\Gamma \vdash p : \mathrm{Id}_A \ a \ b \qquad \Gamma \vdash q : \mathrm{Id}_A \ a \ b} \end{array}}{\Gamma \vdash p = q : \mathrm{Id}_A \ a \ b}$$

Proof. (Type theory perspective) By equality reflection, the existence of p and q implies $a = b$, so p and q are also instances of $\mathrm{Id}_A \ a \ a$, and by lemma 6.15 they are both equal to refl_a. □

Proof. (Categorical perspective) $[\![p]\!] \circ \mathrm{eq}([\![a]\!], [\![b]\!]) = [\![q]\!] \circ \mathrm{eq}([\![a]\!], [\![b]\!])$, and then $[\![p]\!] = [\![q]\!]$ since $\mathrm{eq}([\![a]\!], [\![b]\!])$ is monic. □

只拘泥于类型论本身，往往会陷入意想不到的境地

- 除非超越类型论

还有哪些类型可以解释？

- 范畴论里面有一个引理，基本上所有类型论里的构造都可以直接搞进去
  - 但是我倾向于使用范畴论里本来就存在的构造来解 释类型论里的构造
  - 这样的话发现新的现象和定理就更容易，和其他 领域结合起来就更方便
- 讲义里面写了函数类型、依值积类型、命题宇宙类型
  - 没时间讲了

但是这样……

- 我们学到的还是不够
  - 我们只是机械地弄了个范畴出来
  - 然后让它和类型论一一对应
- 抽象的大锤不是这么玩的
- 我们能不能把整个范畴重新定义出来
  - 先纯范畴论地定义一个范畴
  - 然后用它解释类型论里面的东西
  - 然后其他满足这些结构的范畴就可以被套上去
  - 这样类型这个概念本身都可以和其他数学 对象关联起来

后面部分的前置知识

- 对 Martin-Löf 类型论有一定熟悉程度
  - 李姐 identity type 和 inductive type
  - 不行的话，有 Coq 背景也可以，但是多少有些拉跨
- 最好有 HoTT 或者代拓的背景，但是没有也不是不可以听（
- 看得懂少量 Haskell 或者 ML 风格的代码

什么是准层？

Definition 3.1 (Presheaf)**.** For a category $\mathcal{C}$, functor $T : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ is called a *presheaf*. Sometimes we also say that functor $T' : \mathcal{C}^{\mathrm{op}} \to \mathcal{V}$ to be a $\mathcal{V}$-valued presheaf. ▲

有哪些准层？

Definition 3.3 (HomFunctor)**.** For a category $\mathcal{C}$ and an object $B \in \mathcal{C}$, the *hom functor* $\mathcal{C}(-, B) : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ sends each object $A \in \mathcal{C}$ to the hom set $\mathcal{C}(A, B)$. The morphism $h \in \mathcal{C}(X, Y)$ is sent to the function who sends $g \in \mathcal{C}(Y, B)$ to $g \circ h \in \mathcal{C}(X, B)$. A poor visualization:

$$
\begin{tikzcd}
A_0 \arrow[r, bend left, "h_0"] \arrow[r, "h_1"] \arrow[r, bend right, "h_2"'] & B & A_1 \arrow[l, bend right, "g_1"'] \arrow[l, bend left, "g_0"]
\end{tikzcd}
$$

The hom functor sends A_0 to the set $\{h_0, h_1, h_2\}$ and A_1 to the set $\{g_0, g_1\}$. Similarly, we also say $\mathcal{C}(B, -) : \mathcal{C} \to \mathbf{Set}$ to be a hom functor.

Demonstration 3.4. Hom functors (definition 3.3) are presheaves (definition 3.1). For a category $\mathcal{C}$, a presheaf $T : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ is called a *representable presheaf* if it is naturally isomorphic to $\mathcal{C}(-, B)$.

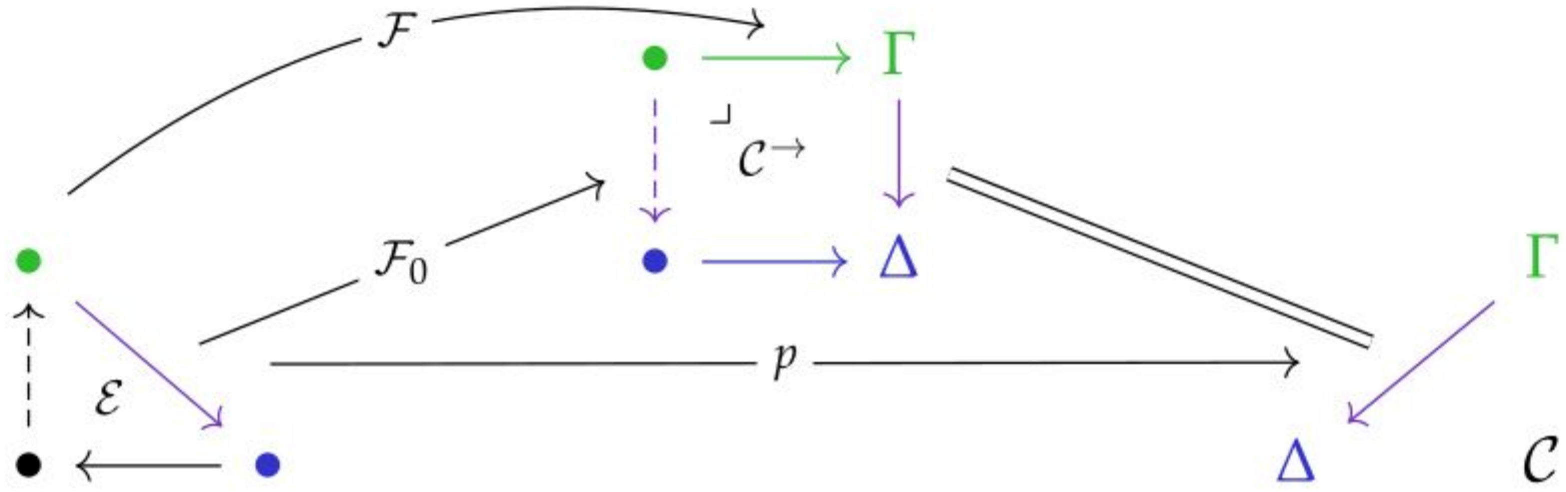
一个稍显生搬硬套的定义

Definition 5.6 (Pitts' TypeCat). A *type category* consists of:

(1) A category $\mathcal{C}$, in analog to the category of contexts (definition 2.25).
(2) A presheaf $\mathrm{Ty} : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$. We refer to $\mathrm{Ty}(\Gamma)$ as the set of *dependent types indexed by* Γ.
(3) A *context extension* operation, constructing $(\Gamma, A) \in \mathcal{C}$ from $A \in \mathrm{Ty}(\Gamma)$.
(4) A *projection morphism* $\pi_A \in \mathcal{C}((\Gamma, A), \Gamma)$, the inverse of context extension.

直接弄一个范畴论的构造

- 我就省略不提了，不过大概是下面这样的



我们能做什么？

- 我们把函子 p 定义一个代数拓扑常用构造——纤维化
  - 这事是 Bart Jacobs 干的，在他之前是 Cartmell 的 CwA
- 然后 C 换成某个拓扑空间的范畴，比如 Top
- 那么我们实际上就把类型解释为了拓扑空间
  - 值就是空间里的点，函数就是连续函数
  - 我们可以把拓扑空间里面的定理搬进类型论！

听众中如果有 POPL 2022 的审稿人
请离开直播间

如果你开着 slides 也请不要往后翻了

Martin-Löf 的等号类型

- 没有 equality reflection，而是通过「Id A x y 的实例可以把类型 P x 的实例转为 P y 的实例」这种思想去定义相等
- 这个类型可以在拓扑空间的模型里面被解释成 path space

$$\frac{\Gamma \vdash A : \mathcal{U} \qquad \Gamma \vdash x : A \qquad \Gamma \vdash y : A \qquad \Gamma \vdash P : (y' : A) \to \mathrm{Id}\ A\ x\ y' \to \mathcal{U} \qquad \Gamma \vdash p : P\ x\ \mathrm{refl}_x \qquad \Gamma \vdash q : \mathrm{Id}\ A\ x\ y}{\Gamma \vdash \mathrm{J}\ P\ p\ q : P\ y\ q}$$

什么是 path space

- 就是一个有无数个点的空间，大概是一条可以随意弯曲的曲线段
  - 假设 A 是个空间，x 和 y 是里面的点，那么它们之间的曲线段构成的空间就是 Id A x y
- 如果这个线段的开头和结尾是同一个点，那么这个线段被成为一个自环

- 如果一个空间，它包含一个点和一个自环，那么这个空间被称作圆圈
  - 图中的 ≡ 就是 Id 的中缀语法

```
data 𝕊¹ : Type₀ where
  base : 𝕊¹
  loop : base ≡ base
```

我们怎么把这些类型搞进类型论呢

- 设计新的类型论：允许给归纳类型定义这样的构造器（其中 I 类型表示一个线段，left 和 right 分别是它的两个实例，表示两个端点）

$$
\begin{aligned}
&\mathsf{data}\ \mathbb{S}^1 : \mathcal{U} \\
&\mid \mathsf{base} \\
&\mid \mathsf{loop}\ \mathbb{I}\ \mathsf{with} \left\{ \begin{array}{l} \mathsf{left} \Rightarrow \mathsf{base} \\ \mathsf{right} \Rightarrow \mathsf{base} \end{array} \right\}
\end{aligned}
$$

加了这些功能，然后能干嘛？

- 我们发现这样的语言特性还可以定义更多新的类型

```
data ℕ : 𝒰
  | zero
  | suc ℕ
```

```
data ℤ : 𝒰
  | pos ℕ
  | neg ℕ with zero ⇒ pos zero
```

这些新的语言特性还可以再次被解释到范畴里！

- 下图是一个 pushout。如果我们的类型论对应的范畴允许这样的 pushout 的存在，那么我们又得到了一个新的、结构更多的范畴

$$
\begin{tikzcd}
1 \arrow[r, "\llbracket \mathrm{zero} \rrbracket"] \arrow[d, "\llbracket \mathrm{zero} \rrbracket"'] & \llbracket \mathrm{N} \rrbracket \arrow[d, "\llbracket \mathrm{pos} \rrbracket"] \\
\llbracket \mathrm{N} \rrbracket \arrow[r, "\widetilde{\llbracket \mathrm{neg} \rrbracket}"'] & \llbracket \mathrm{Z} \rrbracket \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}
$$

有好多 paper 可以发

Aya 编程语言开发组诚邀您的加入

以上所述语言特性均已实现，目前需要能做编程语言工具链的小伙伴和能设计类型论的小伙伴。我们在不断地实现对编程语言工具链和类型系统的一些巧妙的想法。

我们始终使用最新版本的 Java 工具链进行开发，确保跟进上游最新进度，你将使用一个半函数式的编程语言