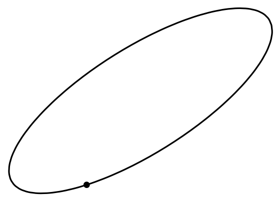
一点儿综合同伦论

A Bit of Synthetic Homotopy Theory

kang

July 10, 2023

$$\left[\!\left[\begin{array}{l}\mathsf{data}\ \mathsf{S}^1 : \mathsf{Type}\ \mathsf{where} \\ \quad \mathsf{base} : \mathsf{S}^1 \\ \quad \mathsf{loop} : \mathsf{base} = \mathsf{base}\end{array}\right]\!\right] =$$



目录

1 What is... 综合同伦论?

同伦论 (homotopoy theory) 是拓扑学的主要分支, 而拓扑学是一门研究形状的学科，更准确地说，它探究的是在连续变形下仍然保持不变的几何性质的学科. 为了进行严谨的数学表述，数学家们首先创立了点集拓扑，也称为一般拓扑学. 事实上，点集拓扑最初是为了研究分析学而创造的，但后来成为了几何学的基石之一. 基于此，我们可以定义我们所研究的几何对象，即拓扑空间. 以及一种被称为同胚的等价关系, 这种等价的概念不受连续形变的影响.

数学家们常说，拓扑学的基本课题是研究在同胚下保持不变的特征，也就是所谓的拓扑不变量. 长期以来，最重要的拓扑不变量都是通过应用代数学的方法获得的代数对象，比如一些群或环，由此产生了代数拓扑的概念. 有趣的是，这些不变量往往在一种更一般的等价关系——同伦等价下仍能保持不变. 随着研究的深入，数学家们开始意识到同伦在拓扑学中具有更基础的地位. 于是同伦论诞生了，它转而研究空间在同伦等价下的不变性质. 虽然严格来说，同伦论应该被看作是代数拓扑的一个分支，但如今人们常常将两者看作是同义词.

虽然拓扑学总被认为是非常抽象的理论，但其中许多几何上的直观概念，即使对于缺乏训练的普通人而言也是容易理解的. 然而，严格地定义空间及其不变量是一项困难的任务. 更加艰巨的挑战在于理解空间之间的同伦等价关系，并建立一套完整的技术体系来进行构造和证明.

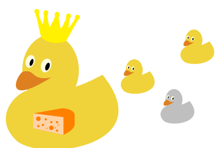
2 类型与空间

2.1 类型的语义（可以）是空间!

如果有人信誓旦旦地向你保证，同伦类型论可以用来研究同伦论，那个人究竟在说些什么? 简而言之，同伦类型论是一门语言，它可以以空间的同伦论作为语义, 就像中文可以以现实世界中的事物作为语义. 换句话说，同伦类型论中的任何语句都可以被视为关于空间的陈述，或者是同伦论中的理论表达.

为了阐明这一点，首先我们需要理解什么是语言、什么又是语义. 语言是由一系列符号的组合构成的. 然而，我们不能随意地将这些符号组合在一起，而必须遵循一定的语法规则. 在形式语言中，这些组合规则被称为句法 **句法** (syntax). 我们必须严格依照句法规则来构建表达式. 当我们描述类型论时，我们是在描述可以使用的符号以及组合这些符号的句法. 而当我们使用类型论时，我们是在按照句法规则书写各种不同长度的语句.

当然，除了逻辑学家和语言学家，很少有人仅仅为了一门语言的句法而去学习它. 我们更感兴趣的是语言能够表达什么，也就是语言的**语义** (semantics). 然而，语言和语义经常会被混淆. 举个例子：在中文中，我们使用词语 鸭鸭 来代表鸭子这种可爱的动物. 然而，一个单词（作为一串符号）和它所指代的事物（也就是它的含义或语义）实际上是不同的概念. 为了清晰地区分它们，我们使用类型论学家的括号 ⟦ – ⟧ 来扩住一种语言中的语句，以表示它所指代的含义. 在刚刚的例子中，单词 鸭鸭 是中文中的一个表达式，而 ⟦ 鸭鸭 ⟧ 则表示这个词所指代的动物:

⟦ 鸭鸭 ⟧ = 

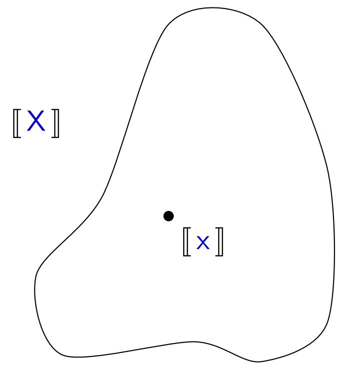
让我们回到类型论与同伦论. 类型论是一门形式语言，而在这门语言中，最基本的句式如下:

$$x : X$$

这同样是最重要的句式，我们在使用类型论时几乎只会构造出这一种句子. 对于这类句子, 我们会说 X 是一个**类型** (type), 而 x 是类型 X 的一个**项** (term). 需要强调的是，术语“类型”和“项”很大程度上仅仅是语言学的概念, 或者说句法的一部分. 这类似于在分析中文句子时:

鸭鸭在游泳。

我们会称 $\textsf{鸭鸭}$ 是主语, 而 $\textsf{游泳}$ 是谓语. 这种纯粹的语法概念可能会使你感到困惑. 幸运的是，为了理解一门语言，人们没有必要完全从形式的规则出发，而可以从它们所表达的含义出发，或者说从语义入手. 就像主语往往指向现实世界中的事物，而谓语则代表它们的动作或行为一样，我们认为**每个类型都指向一个几何空间，而该类型的项则指代该空间中的一个点**:



一旦我们理解了这一点，我们就能够理解为什么类型论可以用来表述同伦论. 这其实很简单，不是吗? 很快你就会了解到，实际上**类型论所有的概念和规则都可以被视作是"连续的"**. 我们将这种**将类型对应为空间，将关于类型的陈述解读为关于空间同伦性质的陈述**的诠释称为**空间语义**. 本讲义会始终使用空间语义来理解类型论与同伦论之间的关系.

如果我们需要表达 X 是一个类型，可以这样写:

$$\mathsf{X} : \mathsf{Type}$$

这里 Type 是**所有类型的类型**, 通常被称为**类型宇宙** (type universe). 而作为类型宇宙的项，X 自然就是一个类型了. 换句话说，一个类型可以看成是某个总体类型的项. 那么 Type 指的是什么空间呢? 用先前的术语来说，我们在问其语义 $[\![\mathsf{Type}]\!]$ 是什么. 答案并不令人意外——它指代的是**所有空间的空间**, 被称为**宇宙** (universe) 或者**对象分类器** (object classifier), 我们用 $\mathcal{U}$ 来表示它:

$$[\![\mathsf{Type}]\!] = \mathcal{U}$$

要理解所有空间的空间是什么，需要花一些功夫. 这个概念使我们能够将一个空间本身看作是某个总体空间中的一个点. 同时，由于我们可以在空间中自由连续地移动其中的点，因此这个总体空间允许我们对一个空间进行连续的变化. 这种变化就是所谓的同伦. 我们稍后会更详细地讨论这个概念，但现在你只需要知道这样的想法是有意义的.

小心 ⚠ 2.1. 你可能已经注意到，讨论由所有空间组成的空间是一件危险的事情. 就像讨论所有集合组成的集合一样，这些概念都会引发所谓的罗素悖论. 实际上，并不存在由所有空间组成的空间，我们只能选择其中一部分作为总体. 这一部分被称为**小空间** *(small space)*，而其余的则被称为**大空间** *(large space)*. 然而，在数学实践中，数学家们并不关心这个问题, 这是因为从技术上讲，考虑所有空间和仅考虑小空间并没有太大的区别，所以我们也不会过多强调这一点.

2.2 类型论的句法

因为我们会使用类型论的语言来研究同伦论，所以有必要先讨论一下类型论这种语言本身. 在类型论中，完整的句子被称为**判断** (judgement), 判断和它们的规则一般以**自然演绎** (natural deduction) 的方式被写下来. 我们使用的类型论里只有三种类型的判断:

$$\Gamma\ \text{context} \qquad\qquad \Gamma \vdash \mathsf{x} : \mathsf{X} \qquad\qquad \Gamma \vdash \mathsf{x} = \mathsf{y}$$

这三种判断从左至右分别是:

"Γ 是一个合法的语境."

"在语境 Γ 下，x 是类型 X 的项."

"在语境 Γ 下，项 x 与 y 依判断相等."

在上一节中，我们已经提到了中间这种类型的判断. 然而, 你肯定会注意到这里多了一个叫做语境的概念. 让我们来解释一下这些术语和判断的含义. **语境** (context), 又叫**上下文**, 一般用大写的希腊字母来表示. 语境指的是我们在写下一句话时所有需要用到的自由变量, 或者说前提. 语境是按照下面的规则归纳构造的:

$$\frac{}{\cdot\ \text{context}} \qquad\qquad \frac{\Gamma \vdash \mathsf{X} : \mathsf{Type}}{\Gamma\ (x : \mathsf{X})\ \text{context}}$$

符号 $\cdot$ 指**空语境** (empty context), 在这个语境下的表达式不包含任何变量. 要求 Γ 是合法语境的判断在这里被省略了, 但实际上我们总是要求任何写出的 Γ 都必须是合法的. 这些规则意味着, 语境可以以列表形式被写下来，这被称为 **telescope**:

$$(x_1 : \mathsf{X}_1)\ (x_2 : \mathsf{X}_1)\ \ldots\ (x_n : \mathsf{X}_1)$$

在 telescope 中，每一个 X_i 都是类型. 在这个语境下写出的表达式就可以使用这些 **bound variable** x_i.

当我们在真正使用类型论而不是研究它们的句法时，我们往往不会明确地指出语境. 这和我们日常使用中文是一样的，没有人在说一句话之前会把这句话的一切前提都列举一遍，这些背景条件往往是靠人们自行从上下文里推断. 而如果你在使用基于类型论的程序语言或者 proof assistant，表达式的语境则会由计算机根据语法自动推导出来.

判断相等 (judgemental equality), 或者称为**定义相等** (definitional equality), 是类型论句法中所包含的一种等价关系. 在通常的语言中，语句是由遵循一些规则的字符串. 然而，在类型论中，更好的理解方式将语句视为字符串的**等价类**, 其中需要商掉的关系就是判断相等. 换句话说，彼此间判断相等的表达式就是"同一个"表达式. 这是类型论和通常语言的一个区别.

在类型论中，我们要求所有对表达式的操作和演绎都必须保持判断相等，尽管很多时候我们不会明确地写出这些判断相等的规则.

对于类型论中的表达式，**代换** (substitution) 是非常重要的操作. 所谓的代换就是把一个表达式里中的某个自由变量全部替换为另一个表达式, 表示为

$$\{...\}[\mathsf{a}/x]$$

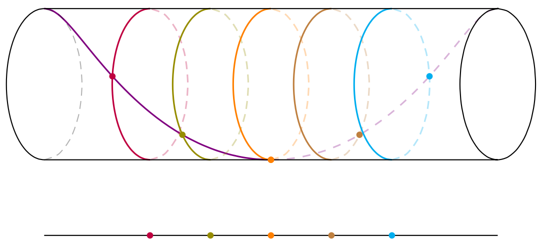
这里的 $\{...\}$ 是一个包含自由变量 x 的表达式. 我们会把其中的 x 全部替换为表达式 a. 下面的 (自然语言中的) 例子可以这一点:

$$\text{“}x\text{ 真的很可爱”}[\text{鸭鸭}/x] = \text{“鸭鸭真的很可爱”}$$

最后我们有两类重要的规则，它们分别是代换规则和弱化(weakening) 规则. 代换规则指的是: **如果在语境 Γ 下的的表达式中的自由变量能够被全部代换成语境 Δ 下的表达式，那么代换后的就是语境 Δ 下合法的表达式**. 而弱化规则指的是: **在某个语境 Γ 下的的表达式，仍然是将 Γ 做扩充之后得到的语境下的表达式**. 同时，所有表达式间的判断相等都被这些操作所保持. 这些不涉及具体类型的、约束并规范如何对判断进行演绎的规则被称为**结构规则** (structural rule).

2.3 纤维化

在这一节中，我们来解释类型论中的判断语句是如何对应着关于空间的陈述的. 我们需要几何学和拓扑学中最基本且最重要的概念之一——**纤维化** (fibration). 下图展示了一个纤维化，还有它的一个涂成紫色的**截面** (section). 这个纤维化的的**底空间** (base space) 是下方的直线, 其所有**纤维** (fiber) 都是圆周, 而**全空间** (total space) 是圆柱的表面. 我们在底空间上选择了五个点，并分别为**这些点处的纤维**以及**截面对应的值**粉刷上了五种不同的颜色. 如果你把横向摆放的圆柱面切割为无数个纵向的圆周，就可以得到这个纤维化, 而截面则是围绕圆柱面旋转一周形成的螺旋线.



让我们更仔细地解释这些术语, 你可以对照上面的图加以理解. 在纤维化中，我们连续地为**底空间**上的每个点指定一个空间，这个空间称为该点处的**纤维**. 一个纤维化所有的截面可以组成一个空间，称为**截面空间** (section space) (并没有在图中画出来). 纤维化的一个**截面**就是为底空间的每个点连续地选取其纤维中的一个点. 此外，我们可以沿着底空间将所有纤维合并，得到的空间称为**全空间**. 很快你就会发现，虽然用语言描述纤维化相当繁琐，但这个概念其实非常直观.

在类型论中，语境 Γ 也可以被理解为空间. 如果将 Γ 展开成一个 telescope:

$$\Gamma \;=\; (x_1 : \mathsf{X}_1)\,(x_2 : \mathsf{X}_1)\;\ldots\;(x_n : \mathsf{X}_1)$$

我们就可以把 $[\![\Gamma]\!]$ 视为由 n 元组组成的空间:

$$(x_1,\, x_2,\, \ldots\,,\, x_n)$$

而语境中的表达式可以被认为是被这个空间所参数化、连续分布的语句. 虽然这听起来可能有些抽象，但只要我们讨论的事物具有一定的连续性，即使是日常生活中的语言也会表现出这种性质:

$(t$: 时间$)$ $(x$: 在t时的湖面$)$ $\vdash$ 鸭鸭在t时游到了湖面上的x处。

就像我们之前强调的那样，类型论中的所有概念和规则都尊重连续性. 因此, 类型论中所有符合句法的语句都类似于上面的句子，关于它们的语境是“连续的”. 当我们选择了一个语境 Γ 时，就相当于选择了一个底空间 $[\![\Gamma]\!]$, 而一切基于 Γ 的表达式都与底空间 $[\![\Gamma]\!]$ 上的纤维化相关联. 事实上，类型论中的判断都可以用这种方式解读. 考虑下面的判断:

$$\Gamma \vdash \mathsf{X} : \mathsf{Type}$$

基于我们的空间语义, $[\![\mathsf{X}]\!]$ 可以被解读为在 $[\![\Gamma]\!]$ 上连续分布的一族空间. 换句话说，它就是 $[\![\Gamma]\!]$ 上的一个纤维丛. 如果我们进行代换 (假定这个代换合法):

$$\mathsf{X}[\mathsf{a}/x] : \mathsf{Type}$$

那么实际上，这个判断就是在断言 $[\![\mathsf{X}[\mathsf{a}/x]]\!]$ 是纤维化 $[\![\mathsf{X}]\!]$ 在点 $[\![\mathsf{a}]\!]$ 处的纤维. 类似的，对于以下判断:

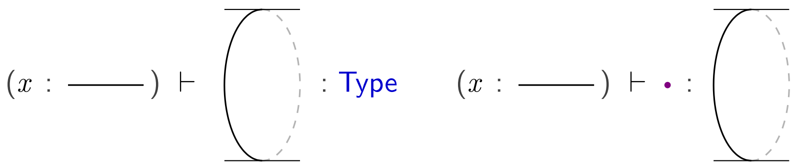
$$\Gamma \vdash \mathsf{x} : \mathsf{X}$$

它可以被解读为在 $[\![\Gamma]\!]$ 上连续地选择 $[\![\mathsf{X}]\!]$ 中的点. 换句话说，它对应着纤维丛 $[\![\mathsf{X}]\!]$ 在 $[\![\Gamma]\!]$ 上的一个截面. 同样的，判断

$$\mathsf{x}[\mathsf{a}/x] : \mathsf{X}[\mathsf{a}/x]$$

可以被理解为取截面 $[\![\mathsf{x}]\!]$ 在点 $[\![\mathsf{a}]\!]$ 处的值 (该值在纤维 $[\![\mathsf{X}[\mathsf{a}/x]]\!]$ 中).

例 2.2. *在本节开头的图中，纤维化和截面可以使用两个判断来描述:*



而底空间上的五个点可以用以下方式描述:

$$\cdot \vdash \bullet\ \bullet\ \bullet\ \bullet\ \bullet : \text{———}$$

现在，让我们思考一下，如果我们将上述两个判断中的表达式分别用这五个项进行代换，会得到什么结果呢?

除了这些判断外，还有一类判断我们尚未提及，那就是判断相等. 事实上, 判断相等的表达式非常类似于通常语言中的同义词. 例如 鸭鸭 和 duck,

虽然它们由不同的字符构成，但是其所指的动物没有任何差别. 同样地，在类型论中，**所有判断相等的表达式所对应的空间中的事物都是完全相同的**. 因此，当我们考虑空间语义时，判断相等并不会产生实质性的影响. 它更多地在句法层面扮演角色，而不是在语义层面发挥作用.

2.4 类型规则

在前面的部分中，我们类型论中进行表达和推理的基本要素和模式, 以及这些基本的规则如何与几何学概念相关联. 然而，作为一种“类型的理论”, 我们却还没有真的写出哪怕一个非平凡的类型[1]. 构造和表达类型以及项的规则叫做**类型规则** (typing rule). 每种类型都对应着一系列类型规则. 尽管严格地定义什么叫做类型规则很困难，但它们通常遵循一定的模式甚至套路. 类型规则主要有以下几类:

1. **类型构造规则** (type formation rule) 告诉我们如何构造类型. 通常，我们会使用现有的项和类型来组合生成新的类型表达式.
2. **项引入规则** (term introduction rule) 告诉我们如何编写特定类型的项. 这通常包括该类型本身固有的项，以及如何通过组合现有的项和类型生成新的项.
3. **项消去规则** (term elimination rule) 告诉我们如何使用特定类型的项. 通常，这些规则允许我们编写到其他类型的函数.
4. **计算规则** (computation rule) 或者有时称为 β **规则** (β-rule), 告诉我们当把消去规则应用于引入规则所引入的项时会得到什么.
5. 当我们对一个项使用消去规则时，会得到许多其他结果. **唯一性原理** (uniqueness principle) 或者有时称为 η **规则** (η-rule) 告诉我们，如果再次使用引入规则，从这些结果重新构造出一个项时，得到的项与最初的项之间是什么关系. 许多类型并没有唯一性原理，但该原理在理论上具有重要价值.

注 2.3. 很多时候，类型构造规则可以被视为类型宇宙 Type 的项引入规则.

当然，这些泛泛而谈恐怕很难帮助你真的理解类型规则的含义. 在接下来的章节中，我们会逐步引入新的类型并给出它们的类型规则. 您很快就会熟悉这些规则的概念.

---

[1] 类型宇宙 Type: “?”

2.5 Π 类型与 Σ 类型

在给定语境 $\Gamma\,(x : \mathsf{X})$ 下的类型 $\mathsf{Y} : \mathsf{Type}$ 时, 我们可以基于此构造两种不同的 Γ 下的类型. 他们分别是 **Π 类型** (Π-type)，一般写作

$$\Pi\,(x : \mathsf{X})\;\mathsf{Y} \qquad \text{或者} \qquad (x : \mathsf{X}) \to \mathsf{Y}$$

和 **Σ 类型** (Σ-type)，一般写作

$$\Sigma\,(x : \mathsf{X})\;\mathsf{Y} \qquad \text{或者} \qquad (x : \mathsf{X}) \times \mathsf{Y}$$

这其实就是两种类型的 type formation rule. 在前面的章节中，我们解释了为什么 $[\![\mathsf{Y}]\!]$ 可以被理解为纤维丛, 而这两种类型则分别对应着纤维丛 $[\![\mathsf{Y}]\!]$ 的截面空间和全空间.

Π 类型是类型论中最重要[2] 的类型之一. Π 类型的项被称为**函数** (function)，其语义对应于纤维丛的截面，这可以从它的 term introduction rule 中得出. 给定语境 $\Gamma\,(x : \mathsf{X})$ 下的项

$$\texttt{\{...\}} : \mathsf{Y}$$

我们可以得到语境 Γ 下类型为 $(x : \mathsf{X}) \to \mathsf{Y}$ 的项

$$\mathsf{f} \;=\; \lambda\,x \;\mapsto \texttt{\{...\}}$$

这个句法规则也被称为 **λ-抽象** (λ-abstraction). 在许多情况下，我们会使用**声明** (declaration) 的写法，这也是在编程语言和 proof assistant 中常见的做法. 我们首先为要定义的项指定一个**名字** (name) 并标注其类型:

$$\mathsf{f} : (x : \mathsf{X}) \to \mathsf{Y}$$

然后在后面给出它的定义:

$$\mathsf{f} \;=\; \lambda\,x \;\mapsto \texttt{\{...\}} \qquad \text{或者} \qquad \mathsf{f}\;x \;=\; \texttt{\{...\}}$$

根据空间语义，λ-抽象意味着空间 $[\![(x : \mathsf{X}) \to \mathsf{Y}]\!]$ 中的点恰好就是纤维丛 $[\![\mathsf{Y}]\!]$ 的截面. 因此，Π 类型所对应的空间是就是**所有截面组成的空间**, 换句话说——就是截面空间.

[2]“连 Π 类型都没有，那还能算类型论吗?”

另一方面,Σ 类型的 term introduction rule 如下: 给定语境 Γ 下的项

$$\mathsf{a} : \mathsf{X}$$
$$\mathsf{b} : \mathsf{Y}\,[\,\mathsf{a}/x\,]$$

我们可以得到相同语境下的项

$$(\mathsf{a}\,,\,\mathsf{b}) : \Sigma\,(x\,:\,\mathsf{X})\;\mathsf{Y}$$

这个句法规则有时也被称为**配对** (pairing). 空间 $[\![\Sigma\,(x\,:\,\mathsf{X})\;\mathsf{Y}]\!]$ 中的点其实就是一对点——底空间中的一个点和该点处纤维中的一个点. 而这正好就是纤维丛 $[\![\mathsf{Y}]\!]$ 全空间中的点. 因此, Σ 类型对应的是纤维丛的全空间.

这两个类型的剩余类型规则如下. Π 类型的 elimination rule 被称为 **应用** (application). 给定函数 $\mathsf{f} : (x\,:\,\mathsf{X}) \to \mathsf{Y}$ 和项 $\mathsf{a} : \mathsf{X}$, 我们可以将 f 应用到 a 上得到:

$$\mathsf{f}\;\mathsf{a} : \mathsf{Y}\,[\,\mathsf{a}/x\,]$$

这对应着在底空间中某一点处取截面的值. Π 类型的 computation rule 是说, 将 λ-抽象定义的函数应用到某个表达式上等同于做一个代换:

$$(\lambda\;x \mapsto \{\ldots\})\;\mathsf{a}\;=\;\{\ldots\}\,[\,\mathsf{a}/x\,]$$

而 Π 类型的 uniqueness principle 则是说, 在将一个函数应用在一个自由变量之后再进行 λ-抽象时, 得到的函数等于原函数:

$$(\lambda\;x \mapsto \mathsf{f}\;x)\;=\;\mathsf{f}$$

Σ 类型的 elimination rule 通常称为**投影** (projection). 给定 $\Sigma\,(x\,:\,\mathsf{X})\;\mathsf{Y}$ 中的项 p, 我们可以得到两个项:

$$\mathsf{p}\,.1 : \mathsf{X}$$
$$\mathsf{p}\,.2 : \mathsf{Y}\,[\,(\mathsf{p}\,.1)/x\,]$$

这对应着分别取全空间点在底空间和纤维上的两个分量. Σ 类型的 computation rule 是说, 取 pairing 的投影就是取其定义时的两个项:

$$(\mathsf{a}\,,\,\mathsf{b})\;.1\;=\;\mathsf{a}$$
$$(\mathsf{a}\,,\,\mathsf{b})\;.2\;=\;\mathsf{b}$$

而 Σ 类型的 uniqueness principle 则是说，将一个项的两个投影重新 pair 起来，得到的结果就是最初的项:

$$(\mathsf{p}\,.1\,,\,\mathsf{p}\,.2)\;=\;\mathsf{p}$$

当我们的纤维丛是**平凡丛** (trivial bundle)——也就是说 Y 的表达式里不包含自由变量 x 时，相应的 Π 类型和 Σ 类型被称为**函数类型** (function type) 和**乘积类型** (product type), 分别记作:

$$\mathsf{X} \to \mathsf{Y} \qquad\qquad \mathsf{X} \times \mathsf{Y}$$

它们分别对应于**函数空间** (function space) 和空间的**笛卡尔积** (cartesian product):

$$\mathrm{Map}([\![\mathsf{X}]\!],[\![\mathsf{Y}]\!]) \qquad\qquad [\![\mathsf{X}]\!] \times [\![\mathsf{Y}]\!]$$

值得注意的是，类型

$$\mathsf{X} \to \mathsf{Type}$$

中的项就是 X 上的"纤维化". 因此这个类型就是 X 上所有"纤维化"组成的类型, 而它的语义可以被看作是 $[\![\mathsf{X}]\!]$ 上所有纤维化组成的空间.

2.6 空间的逻辑学

在更传统的对类型论的解读中，类型语言通常被视为一种逻辑上的形式语言，其涉及到命题、逻辑推理和演绎. 如果从这种角度看待类型论，尤其是使用类型论来表达数学理论时，有一个基本的哲学，你可能已经了解到或者将在本讲义中了解到:

类型即命题

换句话说，逻辑学中的基本概念"命题"和类型论中的基本概念"类型"是同一的. 尽管这个原则非常简单，但如果你没有从小就接触类型论并用它来理解数学，一开始可能会感到十分困惑.

在数学训练中，理解什么是"命题"以及什么是"命题"的"证明"是非常重要的环节. 这也是令许多数学教师非常头疼的教学任务. 在传统的数学基础之下，"命题"和"数学对象"本质上处在两个不同的世界之中. "命题"陈述了"数学对象"的某些性质，这件事不可能反过来，我们也不会声称"证明"了一个"数学对象". 当然，我们可以证明某些"数学对象"的存在性或者其他什么性质. 但请注意，这里证明的实际上都是"命题".

然而，在类型论中，"命题"和"数学对象"在语法上属于同一类事物，它们都是我们所说的"类型". 而"证明"一个"命题"和为一类"数学对象"构造实例是同一件事，它们都是为一个类型编写一个项. 换句话说，逻辑学意义上的推理和演绎与类型论中的构造成为了统一的概念.

让我们来举个例子. 整数 $\mathbb{Z}$ 是一个类型:

$$\mathbb{Z} : \mathsf{Type}$$

众所周知，整数有加法运算 $+$. 断言"整数的加法是交换的"的命题也是一个类型 (我们将其命名为 Comm):

$$\begin{aligned}\mathsf{Comm} &: \mathsf{Type}\\ \mathsf{Comm} &= (m\ n : \mathbb{Z}) \to m + n = n + m\end{aligned}$$

构造类型 $\mathbb{Z}$ 的项就是给出一个整数，这可以非常简单:

$$0 : \mathbb{Z}$$

也可以非常复杂:

$$\begin{aligned}\text{超难的数} &: \mathbb{Z}\\ \text{超难的数} &= \text{“用 20000 页高深数学写出的表达式”}\end{aligned}$$

而构造类型Comm的项则是在证明“整数的加法是交换的”这个命题. 当然，这个证明可以很简单，也可以故意写得晦涩难懂.

$$\begin{aligned}\mathsf{proof} &: \mathsf{Comm}\\ \mathsf{proof} &= \text{“整数加法交换性的证明”}\end{aligned}$$

无论乍看上去如何，“类型即命题”是类型论中非常自然的现象. 一旦你熟悉了类型论的语言，也必定会理解甚至欣赏这件事.

空间语义和这种逻辑学解读是完全相容的. 在人们意识到类型可以被解读为空间之前，类型偶尔会被当作集合——也就是离散的空间. 在这种情况下，如果我们把一个类型解读为一个命题，那么该类型本身就可以被看成是**相应命题所有证明的集合**. 于是，或多或少，“命题”与“数学对象”之间的天堑被打通了. 而现在，我们的类型具有某种的几何连续结构，因而我们完全可以说这个类型对应**相应命题所有证明的空间**. 这暗示着，逻辑学或语言学实际上具有几何学的结构. 这也是为什么人们有时将同伦类型论称为**空间的逻辑学**[3] (the logic of space) 的原因. 下面的表格列出了一部分我们曾讨论过的，逻辑学、类型论和空间语义术语间的相互转换:

表 1: 术语转换表

逻辑学	**类型论**	**拓扑学**
命题 X	类型 X	空间 X
证明 X 为真	构造 X 的项	给出 X 的点
以 x 为变量的谓词 P	语境 $(x : \mathsf{X})$ 下的类型 P	以 X 为底空间的纤维化 P
全称量词 ∀	Π 类型	截面空间
存在量词 ∃	Σ 类型	全空间
全称命题	函数	截面
存在命题	配对	全空间中的点
对每个 x, P(x) 为真	语境 $(x : \mathsf{X})$ 下的项 $\mathsf{p} : \mathsf{X}$	对每个底空间中的点 $x \in X$, 连续地给出纤维 P_x 上的点
存在 x 使得 P(x) 为真	一个项 $\mathsf{a} : \mathsf{X}$ 和 一个项 $\mathsf{b} : \mathsf{P}[\mathsf{a}/x]$	底空间上的一个点 $a \in X$ 和 其纤维上的一个点 $b \in P_a$

[3]Isn't it interesting? 我觉得这真是太有意思了.

3 立方体

3.1 区间与道路

同伦是描述连续移动或变换的概念. 最简单的同伦是点的同伦，简单到我们甚至不会提到同伦这个词, 而称其为**道路** (path). 在空间中，道路指的是一个点从一处连续移动到另一处所形成的轨迹. 通常情况下，我们将道路定义为从**区间** (interval) $\mathrm{I} = [0, 1]$ 到空间 X 的连续映射:

$$p : \mathrm{I} \to X$$

尽管在这个定义中涉及一些技术细节，但实际上它非常直观. 可以想象一下，一个点从区间 I 的左端点 0 连续移动到右端点 1，并且这个点的运动在映射 p 下保持不间断，其最终在空间 X 中便形成一条道路. 另一种理解方式是，用两只手分别握住一条绳子的两端，然后随意摆弄绳子，这条绳子就是我们所处的三维空间中的一条道路.

注 3.1. *不过事实上只要有点的同伦，利用类型论的规则，我们就可以表达所有的同伦. 因为我们只要能把我们想要考虑的事物的类型写出来，这个类型中的道路其实就代表着这些事物的连续变化，或者说同伦关系了.*

这个概念乍看上去十分平凡，但它是整个同伦论的起点，并且可以引出非常复杂和深刻的思想. 不过在进一步深入之前，让我们先考虑如何在类型论的语言中重现这个概念. 实际上，类型论中有不止一种表达道路的方式，在这里我们选择一种与几何本身非常接近的方法. 我们直接引入一个区间类型 $\mathbb{I}$ 来指代区间 I, 同时要求 $\mathbb{I}$ 的句法规则能够非常简单. 这意味着区间的精细结构 (例如在分析学或点集拓扑中的性质) 将被完全抛弃，只有少部分能够描述同伦论的组合结构会被保留. 这也意味着能够对区间类型 $\mathbb{I}$ 做的事情要比真正的区间空间少得多 (但已经足够研究整个同伦论了!).

我们使用的 $\mathbb{I}$ 被称为 **de Morgan 区间类型**，它拥有两个特殊的项:

$$0 : \mathbb{I} \qquad 1 : \mathbb{I}$$

并且对于任意 $i \ j : \mathbb{I}$，我们可以进行三种运算:

$$\sim i : \mathbb{I} \qquad i \wedge j : \mathbb{I} \qquad i \vee j : \mathbb{I}$$

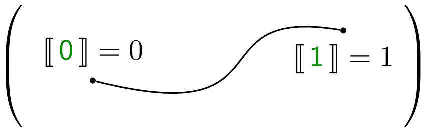
此外，这两个项 0、1 以及三种运算 $\sim$、$\wedge$ 和 $\vee$ 还需要一起满足 **de Morgan 代数的运算律**, 使得区间类型 $\mathbb{I}$ 构成一个 **de Morgan 代数** (可以看成是一

种特殊的**格** (lattice)), 这也是它名字的来历. 以上其实就是我们的区间类型 $\mathbb{I}$ 所拥有的全部结构了.

注 3.2. *这里再次强调, 作为类型的区间 $\mathbb{I}$ 和作为空间的区间 I 是不一样的. 前者是句法的一部分, 而后者是作为语义的数学对象.*

从语义的角度来说, 区间类型自然对应着区间空间, 而它的的两个特殊项分别代表区间的两个端点:

$$[\![\mathbb{I}]\!] \quad = \quad$$



$$= \quad \mathrm{I}$$

它所具备的三种运算则分别对应于颠倒区间 (或者说左右翻转)、以及将正方形压扁成区间的两种操作. 更准确地说, 对任意 $i\ j : \mathbb{I}$, 它们分别指代:

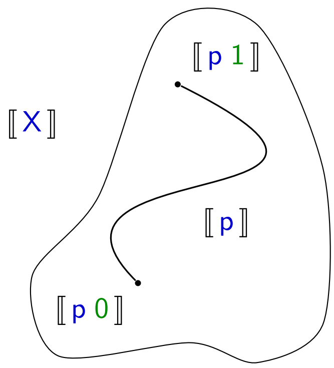
$$\begin{aligned} [\![\sim i]\!] &= 1 - [\![i]\!] \\ [\![i \wedge j]\!] &= \min\ [\![i]\!]\ [\![j]\!] \\ [\![i \vee j]\!] &= \max\ [\![i]\!]\ [\![j]\!] \end{aligned}$$

在数学中, 这里的两种二元运算被称为**连接** (connection), 它们是用立方体研究同伦论时所必需的基本操作.

有了区间类型 $\mathbb{I}$, 类型论中的道路就可以定义为从 $\mathbb{I}$ 出发的函数:

$$\mathsf{p} : \mathbb{I} \to \mathsf{X}$$

它的语义自然就是空间中的道路:



注 3.3. *区间类型应该被赋予哪些结构并没有标准答案, 目前至少有两种方案——de Morgan 立方类型论 和 Cartesian 立方类型论. 我们的区间类型就是前者所采用的.*

在大多数情况下，我们只会关注固定端点的道路. 给定类型 $X : \mathsf{Type}$ 的两个项 $x\ y : X$, 我们可以构造连接 x 和 y 的**道路类型** (path type)，其写作

$$\mathsf{Path}_X\ x\ y \qquad \text{或者} \qquad x =_X y$$

如果不会造成混淆，我们会省略下标 X. 道路类型的项就是使得下面判断相等成立的道路 p (term introduction):

$$\mathsf{p}\ 0 = x \qquad\qquad \mathsf{p}\ 1 = y$$

它的语义自然就是连接 $[\![\mathsf{x}]\!]$ 和 $[\![\mathsf{y}]\!]$ 的道路所组成的空间. 你可能会好奇，为什么我们会使用相等符号 $=$ 来表示道路类型. 这是因为我们可以在类型论中定义**相等类型** (identity type), 它由一个被称为 **J 规则** (J rule) 的公理所刻画，而道路类型正满足这个规则. 在下一节中，我们将更详细地讨论恒等类型的概念.

$$\begin{aligned}\mathsf{J} : \{X : \mathsf{Type}\}\ \{x\ y : X\}\ (P : X \to \mathsf{Type})\\ (p : x = y) \to P\ x \to P\ y\end{aligned}$$

注 3.4. *这里的 J 规则其实只是一个特例，但它已经能够解释相等的概念. 完整的版本将在下一节给出.*

对于每个项 $x : X$，都存在一条常值道路:

$$\mathsf{refl}_x : \mathsf{Path}_X\ x\ x$$

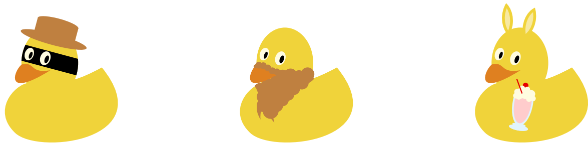
同样，如果不会造成混淆，我们会省略下标 x. 它的定义就是“常值”的道路:

$$\mathsf{refl}_x\ i = x$$

常值道路的涵义，从几何的角度看，它表示始终停留在一个点处的“道路”. 而从相等的角度看，这对应于一个普遍的原理——**任何事物都等于它自己**. 用更专业的术语来说，这被称为**自反性** (reflexivity).

在这里，我们给出的道路类型实际上是一种特殊的扩张类型，它也是以此方式定义的. 关于这个问题，我们在后面的章节中再进行讨论.

3.2 相等类型



如果它看起来像鸭子、游泳像鸭子、叫声像鸭子，那么它可能就是只鸭子。

——似乎是美国谚语

两个事物何时是相等的？这是一个哲学问题．哲学家们对此有各种不同的观点，其中之一被称为**不可分者同一性原理** (identity of indiscernibles). 后来，Per Martin-Löf 将这个思想引入了类型论，并将其命名为 **J 规则**．不可分者同一性原理的陈述是：两个事物 x 和 y 相等，当且仅当

对任何性质 P，只要 x 满足 P，那么 y 也满足 P.

通过令性质 P 为“和 x 相等”，我们可以直接证明这个原理中“仅当”的部分．你可以从上节最后 J 的类型中看出，J 规则就是这个原理中“当”的部分．总的来说，相等类型的定义是：如果存在这样一种关系，只要 x 和 y 满足它，就没有任何东西能够真正区分 x 和 y，那么这个关系其实就是“x 与 y 相等”!

有趣的是，这些概念已经暗示了我们的类型论语言正在描述同伦论——两点的同伦和两点的相等是一回事！这正是从同伦论视角看待事物的方式．在继续深入之前，让我们更详细地探讨一下 J 规则．

J 规则 上一节最后给出的迷你青春版 J 规则已经或多或少地对应于不可分者同一性原理，但它还不足以完整地刻画一个类型．完全体的 J 规则实际上就是相等类型的类型规则．

相等类型的 formation rule 是：对于任意类型 $X : \mathsf{Type}$ 和其中的两个项 $x\ y : X$，我们有类型:

$$x =_X y$$

其项被称为相等，或者**命题相等** (propositional equality)．相等类型的 introduction rule 很简单，就是对于任意 $x : X$ 有一个特殊的项:

$$\mathsf{refl} : x =_X x$$

而相等类型的 elimination rule 是如下的 J. 很多时候当人们提起 J 规则时，其实是专指这个 elimination rule.

$$
\begin{aligned}
&\mathsf{J} : \{X : \mathsf{Type}\}\ \{x : X\} \\
&\quad (P : (y : \mathsf{X}) \to x = y \to \mathsf{Type})\ (d : P\ x\ \mathsf{refl}) \\
&\quad \{y : X\}\ (p : x = y) \to P\ y\ p
\end{aligned}
$$

它和上节中简化版的区别是考虑了整个“相等类型族”上的类型族，而不是只有 X 上的类型族. 最后 J 还需要满足一个 β 规则, 也就是相等类型的 computation rule:

$$\mathsf{J}\beta : \mathsf{J}\ \_\ d\ \mathsf{refl}\ =\ d$$

注 3.5. *相等类型一般没有 uniqueness principle. 因为它的唯一性原理等价于公理* ***K***(Axiom K), *而这个公理和 univalence 是不相容的.*

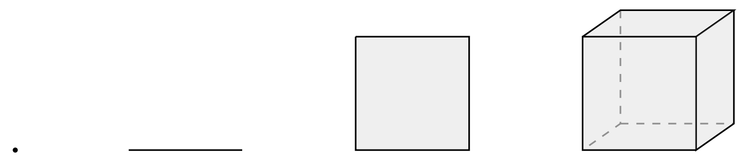
注意到，在一般的类型规则中，我们使用的是判断相等而不是这里的命题相等来叙述 computation rule, 这是有区别的. 为了区分这两种相等类型，我们将使用判断相等进行定义的称为**严格相等类型** (strict identity type). 有时候, 我们会用下面的写法来强调这一点:

$$\mathsf{Id}_X\ x\ y$$

而使用命题相等进行定义的的被称为**弱相等类型** (weak identity type). 它们在表达能力上其实没有太大区别，但前者可能更简洁一些. 正如上一节最后提到的，我们的道路类型就是一种弱相等类型. 之后，我们会使用 transport 运算来证明道路类型满足 J 规则.

3.3 立方体与高阶同伦

一个 n 维的**立方体** (cube) 可以被认为是 n 维欧几里得空间 $\mathbb{R}^n$ 中，所有坐标取值在单位区间 $x_i \in [0,1]$ 中的点所组成的子空间. 在低维情形下，立方体是我们熟悉的物体: 0 维时是点，1 维时是区间，2 维时是正方形而 3 维时就是人们常说的"立方体".



在第一节中，我们为自己的语言引入了区间类型 $\mathbb{I}$, 从而获得了表达道路的能力. 任何包含参数 $i : \mathbb{I}$ 的表达式可以被看成是在一条道路上连续分布的语句. 但事实上，我们可以为我们的语境引入任意多个区间类型的自由变量:

$$i_1 \ \dots \ i_n : \mathbb{I}$$

当我们写出的表达式依赖于这 n 个变量时，可以认为这个表达式实际上被一个 n 维的立方体所参数化. 换句话说，由于类型论的基本规则，只需要一个区间类型，我们实际上就可以表述任意维度的立方体了.

同样，区间类型所拥有的项和运算也都对立方体有意义. 这使得我们可以对立方体进行操作，表达一个立方体的任意面，并定义立方体之间复杂的映射. 特别地，我们可以定义很多**退化** (degenerate) 的立方体.

在类型 X 中，我们称如下的映射 p 是 X 中的 n 维立方体:

$$\mathsf{p} : (i_1 \ \dots \ i_n : \mathbb{I}) \to \mathsf{X}$$

道路其实就是 X 中的 1 维立方体. 很明显，这个概念是对道路的推广, 有时也被称为**高阶道路**或者**高维道路**. 值得注意的是，高阶道路可以自然地被看成是 $\underbrace{\text{道路的道路的} \dots \text{道路}}_{n\ \text{个“道路”}}$. 类似的，就像上一节讨论的那样，道路可以被理解为**同伦** (连续的变化) 或者 **相等** (事物的不可区分性). 同样的，高阶道路也可以被理解为 $\underbrace{\text{同伦的同伦的} \dots \text{同伦}}_{n\ \text{个“同伦”}}$ 或者 $\underbrace{\text{相等的相等的} \dots \text{相等}}_{n\ \text{个“相等”}}$. 因此，它们也被称为**高阶同伦**或者**高阶相等**.

从这个角度来看，立方体之间的映射实际上对应于对高阶同伦或者高阶相等的操作. 在通常的 (没有区间类型的)HoTT 中，高阶同伦是一种隐含的

概念，在某种意义上只能以间接的方式处理. 而我们拥有表达立方体的能力，也就可以明确地表示它们之间的关系.

不过，目前我们的语法规则只允许我们表达立方体之间的映射，而没有提供构造和使用它们的方法. 为了扩展我们语言的表达能力，我们还需要进一步引入以下概念：

1. 通过**余纤维化**、以及相应的**部分元素**和**扩张类型**的概念，来表示立方体的子复形，并处理定义在不同子复形上的元素及其关系；

2. 通过 fill **运算**和它的各种变体，来把现有的立方体组合起来构造新的立方体；

3. 通过 Glue **类型**，来用类型间的同伦等价以及同伦等价间的高阶同伦，构造类型宇宙 Type 中的立方体, 即类型的立方体. 这可以看作是用立方体表达 univalence 的方法.

实际上，这些就是关于立方体的全部句法规则了. 虽然它们加在一起并不算很多，也不是非常复杂，但这些句法规则允许我们做出相当深刻的同伦论论证. 在接下来的几节中，我们将详细解释这些内容.

3.4 子复形

一个立方体的**面** (face) 指的是其边界上的那些维度更低的子立方体. 需要强调的是，在我们的用法中，面不仅仅是余一维的子立方体，还包括顶点、连接顶点的线段、线段围成的方形等等 ...

术语**复形** (complex) 一般指的是一种能够被非常规整地表示成某些非常简单的图形组合的几何体. **立方复形** (cubical complex) 则是由立方体沿着它们的面拼接而成的空间. 在我们的讨论中，我们将只使用一个特例——通过将立方体的一些面并起来得到的子复形.

正如之前提到的那样, 当我们的语境 Γ 中有 n 个类型为区间 $i : \mathbb{I}$ 的变量时, 我们可以认为我们的语境本身包含一个 n 维的立方体. 为了能够更深入地表达同伦论的概念，我们必须有能力将我们的语境限制在这个立方体的某个子复形上, 这意味着需要引入新的句法.

让我们仔细解释一下表达“语境立方体”子复形的句法规则——**余纤维化** (cofibration). 余纤维化 ϕ 是一类表达式，它们与“语境立方体”所有的子复形一一对应:

$$\phi : \mathsf{Cof}$$

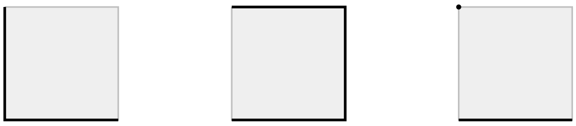
如何写出一个真正的余纤维化呢? 首先，在句法中有两个特殊的余纤维化，分别对应于空集和整个立方体:

$\mathbf{0} : \mathsf{Cof}$ $\quad$ $\mathbf{1} : \mathsf{Cof}$

其次，对于语境中每个区间类型的自由变量 $i : \mathbb{I}$，我们可以写出对应于坐标 $[\![i]\!]$ 等于 0 和 1 时的那一对面的余纤维化:

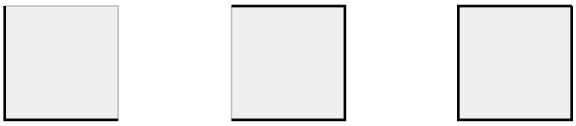
$(i = 0) : \mathsf{Cof}$ $\quad$ $(i = 1) : \mathsf{Cof}$

由此,“语境立方体”中所有的余一维面都可以被表示出来. 接下来，我们注意到立方体的所有低维面实际上可以写成余一维面的交集. 因此我们引入一个“交”运算，对于任意余纤维化 $\phi\ \psi : \mathsf{Cof}$, 它可以写成:



ϕ ψ $\phi \cap \psi$: Cof

方便起见，很多时候我们会省略运算符而直接写 $\phi\psi$ 来代表“交”运算. 最后，一个立方体的子复形就是其面的并集, 因此我们再引入一个“并”运算, 对于任意余纤维化 $\phi\ \psi$: Cof, 它可以写成:



ϕ ψ $\phi \cup \psi$: Cof

借助于这些基本的余纤维化和运算，我们的类型论获得了表达“语境立方体”任意子复形的能力. 值得注意的是，这些代数结构使得余纤维化 Cof 构成一个**格**. 它也可以看成是由子复形和它们之间包含关系所形成的格.

注 3.6. *在同伦论中，***余纤维化** *(cofibration) 是一个相当糟糕的名字, 它其实和纤维化是完全不一样的概念. 这个术语最初是用来描述“性质好的子空间”, 或者说“性质好的单映射”的. 因为它的“好性质”与“性质好的满映射”——也就是纤维化 (fibration) 的“好性质”相对偶，所以才有了这样的名字.*

我们可以将余纤维化加入语境，从而把语句限制在对应的子复形上:

$$\Gamma\,,\,\phi$$

需要注意的是，这里的 ϕ 不能被当作自由变量来使用，相反它约束了合法表达式的形式，这和把 (ϕ : Cof) 加入语境是完全不一样的. 当同时加入数个余纤维化时，我们的语境就被限制在了这些子复形的交上. 为了体现这一点，我们的句法中包含了如下的判断相等:

$$\Gamma\,,\,\phi_1\,,\,\ldots\,,\,\phi_n \;=\; \Gamma\,,\,\phi_1 \cap \ldots \cap \phi_n$$

余纤维化中的区间类型变量也可以被代换, 代换运算会将区间上的 de Morgan 代数结构对应到余纤维化上的代数结构. 具体的规则读者可以自己从语义上推导出来.

到目前为止，我们仍然没有给出在子复形上构造语句的方法. 这将在下一节中解释.

3.5 部分元素与扩张类型

在一个给定的“语境子复形”ϕ 上,我们该怎么写出一个属于类型 $X : \mathsf{Type}$ 的项呢？这样的项被称为 X 在 ϕ 上的**部分元素** (partial element)，记作:

$$\texttt{\{...\}} : \mathsf{Partial}\ \phi\ X$$

部分元素是通过模式匹配来定义的. 简单地说，当我们的余纤维化是一系列余纤维化的并时:

$$\phi\ =\ \phi_1 \cup \ldots \cup \phi_n$$

我们就可以通过下面的写法引入一个在 ϕ 上的部分元素:

$$\lambda \begin{Bmatrix} \phi_1 & \mapsto & \mathsf{x_1} \\ \ldots & & \\ \phi_n & \mapsto & \mathsf{x_n} \end{Bmatrix}$$

这里的 $\mathsf{x_i}$ 是 X 分别在每个 ϕ_i 上的部分元素. 它们需要满足拼图一样的关系——在对应子复形的交集上它们必须是相同的元素 (这样才能拼成一个整体!). 更准确地说，我们要求在每个 $\phi_i \cap \phi_j$ 上有判断相等:

$$\mathsf{x_i}\ =\ \mathsf{x_j}$$

现在，只要我们知道在基本的纤维化及其交上如何构造部分元素，就足以涵盖所有的情形下的构造了. 在空子集 $\mathbf{0}$ 上只有一种元素，表示没有任何内容:

$$\lambda\ ()\ : \mathsf{Partial}\ \mathbf{0}\ X$$

而在整个立方体 $\mathbf{1}$ 上，X 的部分元素就是 X 本身的项，换句话说:

$$X\ =\ \mathsf{Partial}\ \mathbf{1}\ X$$

最后，在 ϕ 对应余一维的面交时，我们的语境仍然是一个立方体，只不过维度变低了. 此时 X 的部分元素依然是 X 的项，但是所有出现在 ϕ 表达式中的区间类型变量都不再是自由的，而会受限于 ϕ 对它们的约束.

例 3.7. 假设语境里有自由变量

$$i\ j\ k : \mathbb{I}$$

然后我们将其限制到如下的子复形上

$$\phi\ =\ (i\ =\ 0) \cap (k\ =\ 1)$$

这样，在试图编写一个部分元素时

$$\{...\} : \mathsf{Partial}\ \phi\ X$$

我们能使用的自由变量就只有 j. 如果表达式 $\{...\}$ 里出现了 i 或 k, 那么它们就会看成是已经被代换为了 0 或 1.

为了处理立方体, 我们还需要另一个非常重要的概念. 对于任意类型 $X : \mathsf{Type}$ 和其在某个子复形 ϕ 上的部分元素 $x : \mathsf{Partial}\ \phi\ X$, 我们可以定义**扩张类型** (extension type):

$$X\ \{\ \phi \mapsto x\ \}$$

扩张类型的项就是 X 的项，但是这些项限制在 ϕ 上后必须与 x 判断相等. 换句话说，这些项就是将 x(作为定义在 ϕ 子复形上的元素) 扩展到整个语境上所得到的元素.

Γ $\quad$ Γ, ϕ $\quad$ $x : \mathsf{Partial}\ \phi\ X$ $\quad$ $\overline{x} : X\{\phi \mapsto x\}$

在我们的语言中，之前提到的固定端点的道路类型，其实是使用扩张类型来严格定义的.

定义 3.8. 给定类型的道路

$$X : \mathbb{I} \to \mathsf{Type}$$

以及它两端的项

$$a : X\ 0 \qquad b : X\ 1$$

在 X 之上连接 a 和 b 的道路类型是:

$$\mathsf{PathP}\ X\ a\ b\ =\ (i : \mathbb{I}) \to X\ i \left\{ \begin{array}{l} (i = 0) \mapsto a \\ (i = 1) \mapsto b \end{array} \right\}$$

作为特例，给定类型 $X : \mathsf{Type}$ 和项 $a\ b : X$, 连接 a 和 b 的道路类型是:

$$\mathsf{Path}_X\ a\ b\ =\ \mathsf{PathP}\ (\lambda\ i \mapsto X)\ a\ b$$

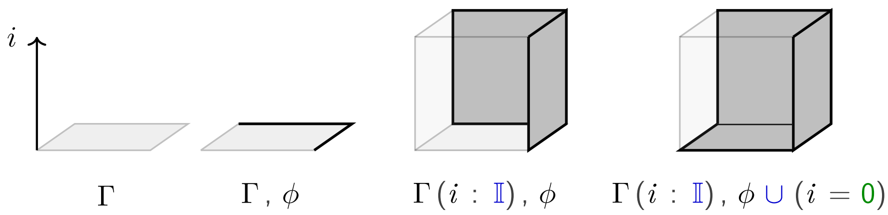
部分元素和扩张类型是立方类型论的基本概念，自始至终我们都会不断使用它们.

3.6 HELP

当一位拓扑学家需要帮助时, **同伦扩张与提升性质** (homotopy extension and lifting property) 就会现身. 在类型论的语言中，HELP 对应着一个名为 fill 的运算.

$$
\begin{aligned}
\mathsf{fill} : \{\phi : \mathsf{Cof}\}\ (X : \mathbb{I} \to \mathsf{Type}) \\
(x : (i : \mathbb{I}) \to \mathsf{Partial}\ (\phi \cup (i = 0))\ (X\ i)) \\
(i : \mathbb{I}) \to X\ i\ \{\ \phi \cup (i = 0) \mapsto x\ i\ \}
\end{aligned}
$$

这究竟是什么意思呢? 让作者我来解释一下. 从类型的角度来看，fill 的作用是将立方体某些特定形状子复形上定义的元素扩展到整个立方体上. 当子复形是立方体本身时 (即 $\phi = \mathbf{1}$)，这个操作当然是平凡的 (没有什么需要扩张的). 在其他情形下，我们可以将其视为从立方体的边界的一部分向内部用元素填充整个立方体，因此这个运算得名 fill. 这些非平凡情形恰好对应着不包含面 $(i = 1)$ 的子复形，你可以将其想象成缺了盖子的箱子. 实际上，正是为了表达这个条件，fill 的类型才不得不写得这么复杂.



让我们更具体地解释一下. 当使用 fill 时, 我们会处于一个形如 $\Gamma\ (i : \mathbb{I})$ 的语境中. 换句话说，我们会选择一个特定的维度 i. 我们需要提供在整个语境上分布的类型 X, 以及不包含自由变量 i 的余纤维化 ϕ (也就是说 ϕ 其实定义在 Γ 上). 如果我们按照上图的方式，将维度 i 竖直放置，那么在语境 $\Gamma\ (i : \mathbb{I})$ 下, 与 ϕ 对应的子复形看起来就像几道竖直的"**墙**". 相应地，作为初始条件，部分元素 x 定义在"**墙和地板的并**"上. 最后，我们可以使用 fill 进行填充，从而获得遍布整个立方体的元素. 当然，填充的结果 fill X x 限制回"墙和地板"上时仍然是 x.

对于空间而言，这种填充操作总是可行的，这个性质就是 HELP(同伦扩张与提升性质). 正如之前解释的，立方体可以被看作是高维度的同伦，这种填充实际上是将一个部分同伦扩张成为更大的同伦. 从另一个角度来看，我们的 X 对应一个纤维化, 此时填充可以被看作是将底空间上的立方体"提升"到纤维化上成为一个截面. 因而我们将其称为"同伦扩张与提升性质".

在任何涉及同伦概念的地方，HELP 都会扮演重要的角色. 在更富有代数或组合色彩的场合下，比如对于我们的立方体而言，这个性质又被称为满足 **Kan 条件** (Kan condition). 需要注意的是，不是每种余纤维化上的元素都可以被无条件地扩展到整个立方体上，能够达成这一点的余纤维化被称为**平凡余纤维化** (acyclic cofibration). 因此我们可以认为，运算 fill 表明了形如 $\phi \cup (i = 0)$ 的余纤维化是一种平凡余纤维化.

注 3.9. *事实上，由 fill 给出的 Kan 条件要比通常同伦论中的条件更强. 首先，通常的 Kan 条件仅要求填充的存在性，而 fill 给出了一个确定的结果. 其次，运算 fill 的结果可以与任意语境间的代换操作交换. 前面这个更强的条件被称为代数 **Kan** 条件* (algebraic Kan condition)*，而后面这个性质在类型论中被称为 **uniformity**.*

作为 fill 的第一个应用，我们来证明一个有趣且重要的事实——只要存在 fill 运算，这样的运算就是唯一的.

定理 3.10 (fill 的唯一性)**.** *任取两个部分元素 x 的扩张:*

$$a\ b : (i : \mathbb{I}) \to X\ i\ \{\ \phi \cup (i = 0)\ \mapsto x\ i\ \}$$

它们都是相等的:

$$a = b$$

特别地，它们都等于:

$$\mathsf{fill}\ X\ x$$

证明. 连接 a 和 b 的道路 p 可以这么定义:

$$\mathsf{p}\ j\ =\ \mathsf{fill}\ X\ \lambda\ i \mapsto \lambda\ \left\{ \begin{array}{l} (j = 0) \mapsto a \\ (j = 1) \mapsto b \\ \phi \cup (i = 0)\ \mapsto\ x\ i \end{array} \right\}$$

□

正如这个证明所展现的，运算 fill 允许我们通过现有的立方体 (作为边界条件) 构造新的立方体. 你可能会感觉到，这个运算很像是立方体的“term introduction rule”. 这种直觉基本上是正确的. 更有意思的是，您将会看到，这同时也是立方体的“term elimination rule”，并且可以用来建立某些弱版

本[4] 的“computation rule”和“uniqueness principle”. 这是因为一方面，我们几乎唯一能够使用立方体的方法，就是把它们输入到 fill 中来制造新的立方体！另一方面，在我们的语言中，相等就是道路，而立方体间的相等就是以这些立方体为边界的高维立方体. 因此，就像刚刚的证明中那样，我们会使用 fill 来构造这种高维立方体，从而建立某些立方体之间的相等关系.

注 3.11. *这段话看上去十分暴论，但它其实只是忽略了一种 (非常重要的) 情形：在类型宇宙* Type *中，还有一种构造立方体的方法——*Glue *类型，这是我们后面章节的内容.*

在实际应用中，我们经常使用填充操作的特殊情况——立方体的**复合** (composition) 运算. 这个运算被记做 comp，实际上它就是 fill 的终点:

$$\mathsf{comp}\ X\ x = \mathsf{fill}\ X\ x\ 1$$

如果将 fill 比喻为从墙与地板建造出一栋完整的房子，那么 comp 就是在最后单独把天花板拆下来. 和 fill 一样, 运算 comp 以及其它变体也具有关于自身的唯一性. 这个事实在应用中经常被使用.

不过，最常用的还不是 comp 运算本身，而是其进一步的两个特例——hcomp 和 transport. 它们是接下来两节的主角, 你将会在那里看到更多例子、解释以及应用.

注 3.12. *我们把* fill *作为最基本的运算来构造出其它的运算，这种顺序符合经典同伦论的精神. 然而，在 de Morgan 立方类型论中, 我们通常反过来使用* comp *来定义* fill*. 而* comp *又常常是被* hcomp *和* transp *所定义的. 这里的* transp *是* comp *的另一个特例.*

在 Cartesian 立方类型论中对应于 comp*、*hcomp *和* transp *的运算有不同的名字，它们分别被称为* com*、*hcom *和* coe*.*

[4]这些规则通常会断言一些判断相等. 而“弱版本”的规则使用的是命题相等，或者说道路.

3.7 在类型内复合

当类型 X 关于“竖直”维度 i 是常值时，运算 comp 就会化为所谓的 hcomp, 这时参数 X 往往被省略:

$$\mathsf{hcomp}\ \{X\}\ x \;=\; \mathsf{comp}\ (\lambda\ i \mapsto X)\ x$$

换句话说，运算 comp 就是在一个固定类型的内部复合立方体.

3.8 在纤维间穿梭

当余纤维化 ϕ 为空时, 运算 comp 就会化为所谓的 transport:

$$\begin{aligned}&\mathsf{transport} : (X : \mathbb{I} \to \mathsf{Type}) \to X\ 0 \to X\ 1\\&\mathsf{transport}\ X\ x = \mathsf{comp}\ X\ (\lambda\ i \mapsto \lambda\ \{(i = 0) \to x\})\end{aligned}$$

换句话说，运算 transport 没有任何关于子复形的限制，它的用处就是把一个元素从道路的这一头“运送”到道路的另一头.

我们首先来陈述一下 transport 的唯一性. 这个性质将在之后多次被用到. 这个唯一性可以直接作为 fill 唯一性的推论得到. 证明中需要使用的，在 X 上连接 x 和 transport X x 的道路，我们会称其为 transportFill.

定理 3.13 (transport 的唯一性)**.** *给定一个项 $x : X\ 0$, 如果存在两条 X 上的道路:*

$$p\ q : (i : \mathbb{I}) \to X\ i$$

使得它们的起点都判断相等于 x:

$$p\ 0 = x = q\ 0$$

那么它们的终点相等 (可以被道路连接):

$$p\ 1 = q\ 1$$

特别地，它们都等于:

$$\mathsf{transport}\ X\ x$$

作为唯一性的第一个应用，我们可以计算沿着常值道路进行 transport 的结果——等于什么都没做. 这只需要使用 x 处的常值道路即可证明. 这个结果同样会经常用到.

定理 3.14. *对任意类型 X : Type 和项 $x : X$，将 x 沿着常值道路进行 transport 得到的结果等于它自己.*

$$\mathsf{transportRefl} : \mathsf{transport}\ \mathsf{refl}\ x = x$$

注 3.15. *如果在一个立方类型论中，上面这个相等是句法规则中的判断相等，那么我们称这个类型论满足* ***regularity***. *不过至今仍不知道是否有能够在满足 regularity 的同时，仍能保持非常优异的元性质的类型论.*

现在我们就可以利用 $\mathsf{transport}$ 证明道路类型满足之前章节中提到的 J 规则了. 这样也就证明了道路类型就是相等类型.

定理 3.16. *道路类型满足 J 规则.*

证明.

$$\mathsf{J}\ P\ d\ p \;=\; \mathsf{transport}\ (\lambda\ i \mapsto P\ \_\ (\lambda\ j \mapsto p\ (i \wedge j)))\ d$$

$$\mathsf{J}\beta \;=\; \mathsf{transportRefl}$$

□

你能看到...

接下来我们对每一种类型构造计算 $\mathsf{transport}$ 的值.

定理 3.17 (Π 类型)**.** *给定类型的道路 X 及其上的类型道路 Y:*

$$X : \mathbb{I} \to \mathsf{Type} \qquad Y : (i : \mathbb{I}) \to X\ i \to \mathsf{Type}$$

以及左端点处的函数:

$$f : (x : X\ 0) \to Y\ 0\ x$$

如果我们计算:

$$\_ \;=\; \mathsf{transport}\ (\lambda\ i \mapsto (x : X\ i) \to Y\ i\ x)\ f\ x$$

其值就 (命题) 相等于:

$$\_\ x \;=\; \mathsf{transport}\ (\lambda\ i \mapsto Y\ i\ (\mathsf{transportFill}\ X^{\text{-}1}\ x\ (\sim\ i)))\ (f\ (\mathsf{transport}\ X^{\text{-}1}\ x))$$

定理 3.18 (Σ 类型)**.** *给定类型道路 X 及其上的类型道路 Y:*

$$X : \mathbb{I} \to \mathsf{Type} \qquad Y : (i : \mathbb{I}) \to X\ i \to \mathsf{Type}$$

以及左端点处 Σ 类型的项:

$$(a\ ,\ b) : \Sigma\,(x : X\ 0)\ Y\ 0\ x$$

如果我们计算:

$$\_ \;=\; \mathsf{transport}\ (\lambda\ i \mapsto \Sigma\,(x : X\ i)\ Y\ i\ a)\ (a\ ,\ b)$$

其值就 (命题) 相等于:

$$\_\ .1 = \mathsf{transport}\ X\ a$$

$$\_\ .2 = \mathsf{transport}\ (\lambda\ i \mapsto Y\ i\ (\mathsf{transportFill}\ X\ x\ i))\ b$$

定理 3.19 (道路类型)**.** 给定类型道路 X 及其上的两条道路 a b:

$$X : \mathbb{I} \to \mathsf{Type} \qquad a\ b : (i : \mathbb{I}) \to X\ i$$

以及左端点处的一条道路:

$$p : a\ 0 = b\ 0$$

如果我们计算:

$$\_ \;=\; \mathsf{transport}\ (\lambda\ i \mapsto a\ i = b\ i)\ p$$

其值就 (命题) 相等于:

$$\_\ j \;=\; \mathsf{comp}\ X\ \lambda\ i \mapsto \lambda \left\{ \begin{array}{l} (j = 0) \mapsto \mathsf{transportFill}\ X\ a\ i \\ (j = 1) \mapsto \mathsf{transportFill}\ X\ b\ i \\ (i = 0) \mapsto p\ j \end{array} \right\}$$

注 3.20. *YeahYeahYeah!*

3.9 Univalence 公理

立方类型论引入了一种名为 Glue 的新类型, 你可以将其视为 Type 的 cube constructor. 它允许我们从类型的同伦等价和现有的立方体构造出新的立方体. Glue 的引入使得 univalence 成为一个定理.

$$\begin{aligned}\mathsf{Glue} : \{\phi : \mathsf{Cof}\}\ \{X : \mathsf{Type}\}\ \{T : \mathsf{Partial}\ \phi\ \mathsf{Type}\}\\ (f : \mathsf{Partial}\ \phi\ (T \simeq X)) \to \mathsf{Type}\ \{\ \phi \mapsto T\ \}\end{aligned}$$

为了更清晰地表示 Glue 类型的参数，我们通常将其写为:

$$\mathsf{Glue}\ \{\ \phi \mapsto (T\ ,\ f\ )\ \}\ X\ =\ \mathsf{Glue}\ \{\phi\}\ \{X\}\ \{T\}\ f$$

从另一个角度来说,Glue 本身也是一个 type constructor. 它的 term introduction rule 与 term elimination rule 分别由一个 constructor 和一个 destructor 给出:

$$\mathsf{glue} : (t : \mathsf{Partial}\ \phi\ T) \to X\ \{\ \phi \mapsto f\ t\ \} \to (\mathsf{Glue}\ \{\ \phi \mapsto \_\ \}\ X)\ \{\ \phi \mapsto t\ \}$$

$$\mathsf{unglue} : \mathsf{Glue}\ \{\ \phi \mapsto \_\ \}\ X \to X$$

类似的，我们会把 glue 写成

$$\mathsf{glue}\ \{\ \phi \mapsto t\ \}\ x\ =\ \mathsf{glue}\ t\ x$$

Glue 类型的 computation rule 与 uniqueness principle 如下, 你可以认为这两个规则就是在表达 glue 和 unglue 是一对互逆的函数:

$$\begin{aligned}\mathsf{unglue}\ (\mathsf{glue}\ \{\ \phi \mapsto t\ \}\ x)\ &=\ x\\ \mathsf{glue}\ \{\ \phi \mapsto t\ \}\ (\mathsf{unglue}\ u)\ &=\ u\end{aligned}$$

正如本节开头提到的，我们可以使用 Glue 类型来证明 univalence.

$$\mathsf{ua} : \{X\ Y : \mathsf{Type}\} \to X \simeq Y \to X = Y$$

首先，我们可以这样定义 ua:

$$\mathsf{ua}\ \{X\}\ \{Y\}\ f\ i\ =\ \mathsf{Glue}\ \left\{\begin{array}{l}(i = 0) \mapsto (X\ ,\ f\ )\\ (i = 1) \mapsto (Y\ ,\ \mathsf{id}_Y)\end{array}\right\}\ Y$$

接下来，我们需要证明 β 规则:

$$\mathsf{ua}\beta : \mathsf{transport}\ (\mathsf{ua}\ f)\ =\ f$$

定理 3.21. *Univalence 在我们的类型论中成立.*

证明. 证明 $\mathsf{ua}\beta$ 相当于证明对任意 $x : X$ 有

$$\mathsf{transport}\ (\mathsf{ua}\ f)\ x\ =\ f\ x$$

由 transport 的唯一性，我们只需要构造一条在 $\mathsf{ua}\ f$ 上方连接 x 和 $f\ x$ 的道路即可. 这条道路可以使用 glue 得到:

$$\lambda\ i \mapsto \mathsf{glue}\ \left\{ \begin{array}{l} (i\ =\ 0)\ \mapsto\ x \\ (i\ =\ 1)\ \mapsto\ f\ x \end{array} \right\}\ (f\ x)$$

□

定理 3.22. *给定 $X : \mathsf{Type}$, 我们有可缩性:*

$$\mathsf{isContr}\ (\Sigma\,(T\ :\ \mathsf{Type})\ (T \simeq X))$$

证明. 事实上，在某种意义上 Glue 类型等价于这个命题成立. 我们来为这个 Σ 类型构造一个 contr 运算.

$$\mathsf{contr}\ \{\phi\}\ (\,T\ ,\ f\,)\ =\ \mathsf{Glue}\ \{\ \phi \mapsto (\,T\ ,\ f\,)\ \}\ X$$

To Be Continued. . .

□

4 简单的同伦论 I

4.1 映射的同伦

就像先前多次提到的那样，**同伦** (homotopy) 最初的也是使用最广泛的含义是指连续映射间连续变化. 在点集拓扑中，对于拓扑空间 X、Y 间的两个连续映射 f 和 g，如果存在一个连续映射 $h : X \times \mathrm{I} \to Y$ 满足边界条件

$$h(x,0) = f(x) \qquad h(x,1) = g(x)$$

这样的 h 就被称为 f 和 g 间的同伦. 当拓扑空间 X 和 Y 的性质足够好时, 我们可以得到两个同胚的空间:

$$\mathrm{Map}(\mathrm{I}, \mathrm{Map}(X, Y)) \ \simeq \ \mathrm{Map}(X, \mathrm{Map}(\mathrm{I}, Y))$$

这两个空间中的点都可以被看成是连续函数 $X \times \mathrm{I} \to Y$. 直观上来看，前者就是 **$X$ 到 Y 的函数空间中的道路**，而后者可以被理解为**对 X 中的每一点连续地指定 Y 中的一条道路**, 当然，在考虑给定函数之间的同伦时, 这二者在区间的两端必须回到函数 f 和 g. 这两个空间中满足这种端点约束的子空间也是相互同胚的，因此它们都可以被视为由映射 f 和 g 间的同伦组成的空间.

将这两种空间中的定义反过来，我们可以得到两种同伦概念在类型论中的表述. 给定类型 $X\ Y$: Type 以及函数 $f\ g : X \to Y$, 前一种定义自然地对应于函数类型中的道路类型，其项就是**函数间的道路**:

$$f = g$$

而后一种定义对应于下面的类型，其项被称为**函数间的同伦**:

$$(x : X) \to f\ x = g\ x$$

幸运的是，和空间的情形一样，这两个定义是等价的.

定理 4.1. *函数间的道路和函数间的同伦一一对应.*

证明. 根据定义，函数间的道路和函数间的同伦分别就是下面两个类型中满足相应边界条件的项:

$$\mathbb{I} \to (X \to Y) \qquad\qquad X \to (\mathbb{I} \to Y)$$

通过交换两个变量的顺序，这两个类型的项是一一对应的:

$$\mathsf{h}\ i\ x \iff \mathsf{h}\ x\ i$$

这个对应保持边界条件不变，因此我们得到了所需的一一对应的. □

注 4.2. *类似的性质对更一般的 Π 类型也成立.*

我们刚刚证明了类型论中的**函数外延性** (function extensionality)，即函数间的道路和函数间的同伦两个定义的一致性. 因而，在使用同伦的概念时，我们无需明确区分这两种定义. 但如果必要, 我们会用 funExt 来记这个等价.

尽管我们对函数外延性的证明很简单，并且与空间中的情形非常类似，但实际上它强烈依赖于我们的语言中具有一个区间类型 $\mathbb{I}$、并且利用 $\mathbb{I}$ 来定义道路的事实. 在更一般的类型论中，我们使用由 J 规则刻画的相等类型来进行这些定义, 此时函数外延性就不一定成立了. 然而，由于函数外延性在空间中的对应陈述成立 (这就是本节开头在讨论的事情)，并且我们的兴趣主要集中在空间语义上，因此我们不会深入探讨函数外延性的概念。

4.2 同构

对于任意类型 $X\ Y$: Type 我们有一个运算将同构变成等价并且保持映射本身:

$$\mathsf{isoToEquiv} : \mathsf{Iso}\ X\ Y \to X \simeq Y$$

5 简单的同伦论 II

5.1 可缩空间

我们说一个空间是可缩的，如果这个空间中存在一个点，使得整个空间可以通过道路收缩到这个点上.

定义 5.1. *给定类型 X : Type, 如果存在 X 的一个的点, 使得从该点到 X 的任意点都存在一条道路, 我们就称 X 是可缩的.*

$$\mathsf{isContr}\ X \ = \ \Sigma\,(a\ :\ X)\ ((x\ :\ X) \to (a = x))$$

这个定义在最原始的 HoTT 中也是有意义的. 然而，一旦我们的语言中有立方体的概念，就可以给出一个 (乍看上去) 更强的定义, 这个定义使用起来更加灵活且具有更大的威力.

定理 5.2. *类型 X : Type 可缩当且仅当可以定义如下的 contr 运算:*

$$\mathsf{contr} : \{\phi\ :\ \mathsf{Cof}\}\ (x\ :\ \mathsf{Partial}\ \phi\ X) \to X\,\{\ \phi \mapsto x\ \}$$

注 5.3. *定理中的 contr 运算可以这么理解: 对于任何立方体, 类型 X 定义在这个立方体任意部分上的元素，都可以被无条件地扩张到整个立方体上.*

证明. 首先，我们使用运算 contr 来证明 X 的可缩性. 我们构造如下的项:

$$\mathsf{p} : \mathsf{isContr}\ X$$

它的两个分量都是 contr 的直接应用. 首先，注意到当我们取 ϕ 为空子集时，运算 contr 会给出 X 的一个项. 我们就取这个项作为收缩的中心:

$$\mathsf{p}\ .1 \ = \ \mathsf{contr}\ \lambda\ ()$$

随后，我们利用 contr 来将其连接到任意的项 $x : X$:

$$\mathsf{p}\ .2\ x\ i \ = \ \mathsf{contr}\ \lambda \left\{ \begin{array}{l} (i = 0) \mapsto \mathsf{p}\ .1 \\ (i = 1) \mapsto x \end{array} \right\}$$

反之，假设 X 可缩:

$$(a\ ,\ q) : \mathsf{isContr}\ X$$

我们来构造一个 contr 运算. 对于任意的部分元素 $x : \mathsf{Partial}\ \phi\ X$, 我们首先利用收缩 q 把 x 变成一个常值的元素, 然后将常值元素进行扩张 (扩张成定义域更大的常值元素即可), 最后再使用 hcomp 将收缩逆转.

$$\mathsf{contr}\ \{\phi\}\ x\ =\ \mathsf{hcomp}\ \lambda\ i \mapsto \lambda \begin{Bmatrix} \phi \mapsto q\ x\ (\sim i) \\ (i = 0) \mapsto a \end{Bmatrix}$$

□

利用这个性质，我们可以轻松地证明可缩性天生就是一个命题.

定理 5.4. *类型 isContr X 是命题.*

证明. 对于任意的两个项 $u_0\ u_1 : \mathsf{isContr}\ X$, 我们假设它们形如:

$$\begin{aligned} u_0 &= a_0 , q_0 \\ u_1 &= a_1 , q_1 \end{aligned}$$

现在，我们来构造这两个项之间的道路:

$$\mathsf{path} : u_0 = u_1$$

我们可以使用模式匹配来定义道路的两个分量. 注意，我们已经假设了 X 的可缩性 (u_0 和 u_1 在我们当前的语境中可用)，因此我们可以自由地使用 contr 运算.

首先，contr 可以直接连接 a_0 和 a_1:

$$\mathsf{path}\ i\ .1\ =\ \mathsf{contr}\ \lambda \begin{Bmatrix} (i = 0) \mapsto a_0 \\ (i = 1) \mapsto a_1 \end{Bmatrix}$$

接下来需要构造这条道路之上的 q_0 和 q_1 间的道路. 按定义，我们需要对每个 $x : X$ 构造一个给定边界的正方形，这同样可以用 contr 做到:

$$\mathsf{path}\ i\ .2\ x\ j\ =\ \mathsf{contr}\ \lambda \begin{Bmatrix} (i = 0) \mapsto q_0\ j \\ (i = 1) \mapsto q_1\ j \\ (j = 0) \mapsto \mathsf{path}\ i\ .1 \\ (j = 1) \mapsto x \end{Bmatrix}$$

□

定理 5.5. *类型 X : Type 可缩当且仅当存在 X 的项 $a : X$ 使得如下的 extend 运算可以被定义:*

$$\text{extend} : \{Y : X \to \text{Type}\} \to Y\ a \to (x : X) \to P\ x$$

定理 5.6. *对类型 X : Type,以下性质是等价的:*

1. *类型 X 可缩;*
2. *类型 X 是是一个命题,并且存在一个点 $a : X$;*
3. *类型 X 等价于单位类型 $\mathbb{1}$.*

5.2 同伦纤维

5.3 截断

6 S^1

6.1 连通性

让我们使用类型论证明下面这个非常基本的性质.

定理 6.1. *在类型 S^1 中，我们可以 merely 用道路连接 base 和任意点.*

$$(x : \mathsf{S}^1) \to \| \mathsf{base} = x \|$$

注 6.2. *这个性质比连通性 isConnected 更强.*

证明. 我们直接进行模式匹配来构造这样一个项：

$$\mathsf{proof} : (x : \mathsf{S}^1) \to \| \mathsf{base} = x \|$$

对于基点 base 的情形, 最简单的选择当然是常值道路:

$$\mathsf{proof}\ \mathsf{base} = |\,\mathsf{refl}\,|$$

接下来，在处理 loop i 的情形时，需要一点点 (但非常常见的) 技巧. 对于每个 $i : \mathbb{I}$, 一个想当然的选择是道路

$$\lambda\, j \mapsto \mathsf{loop}\ (i \wedge j)$$

遗憾的是，当 $i = 1$ 时，这条道路成为了 loop 而不是 refl, 与先前在 base 处的定义不一致. 不过请注意，这里的构造不是在道路空间本身，而是在道路空间的命题截断中进行的. 因此我们可以使用 squash 来连接不同的项, 然后再配合 hcomp 修正错开的端点即可:

$$\mathsf{proof}\ (\mathsf{loop}\ i) = \mathsf{hcomp}\ \lambda\, k \mapsto \lambda \left\{ \begin{array}{l} (i = 0) \mapsto |\,\mathsf{refl}\,| \\ (i = 1) \mapsto \mathsf{squash}\ |\,\mathsf{loop}\,|\ |\,\mathsf{refl}\,|\ k \\ (k = 0) \mapsto |\,\lambda\, j \mapsto \mathsf{loop}\ (i \wedge j)\,| \end{array} \right\}$$

$\square$

推论 6.3. *类型 S^1 是连通的.*

$$\mathsf{isConnected}\ \mathsf{S}^1$$

6.2 环路空间

环路 (loop) 就是两端连接相同点的道路，这个点叫做环路的**基点** (base). **环路空间** (loop space) 指的是一个空间中某点处所有环路构成的空间. 在这一节中，我们使用类型论来研究 S^1 的环路空间.

$$\Omega\mathsf{S}^1 = \Omega(\mathsf{S}^1, \mathsf{base}) = \mathsf{base} = \mathsf{base}$$

定理 6.4. *类型 S^1 在基点 base 处的环路类型 $\Omega\mathsf{S}^1$ 等于整数类型 $\mathbb{Z}$.*

$$\Omega\mathsf{S}^1 = \mathbb{Z}$$

对于单个的道路类型，我们可以基于 HELP 来构造许多道路 (term introduction)，但却不能直接使用道路 (term elimination). 这意味着，虽然定义到道路类型的函数相对容易，但是定义从道路类型出发的函数却十分困难. 即使我们知道道路类型的具体模型应该是什么，也很难证明它们的确是等价的.

为了解决这个问题，我们可以采用一种称为 encode-decode 的方法. 这种方法的核心思想是，虽然很难对单个的道路类型进行消去操作 (elimination)，但整个**路径丛** (path fibration) 却具有天然的消去规则，即 J 规则. 因此，我们不再局限于单个道路类型的模型，转而构造整个路径丛的模型，并试图证明它们作为丛的等价性.

证明梗概. 首先，我们需要构造路径丛的模型，通常被称为 Code:

$$\mathsf{Code} : \mathsf{S}^1 \to \mathsf{Type}$$

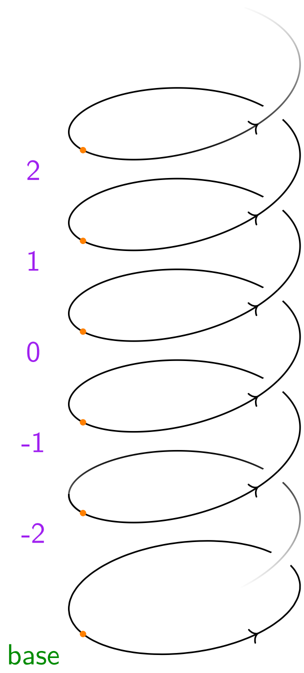
接下来，我们定义 Code 和路径丛之间的一对函数，通常被称为 encode 和 decode:

$$\begin{aligned} \mathsf{encode} &: (x : \mathsf{S}^1) \to \mathsf{base} = x \to \mathsf{Code}\ x \\ \mathsf{decode} &: (x : \mathsf{S}^1) \to \mathsf{Code}\ x \to \mathsf{base} = x \end{aligned}$$

最后，我们证明它们互为逆函数即可.

构造 Code 我们可以使用模式匹配来定义类型族 Code:

$$\begin{aligned} \mathsf{Code}\ \ \mathsf{base} &= \mathbb{Z} \\ \mathsf{Code}\ (\mathsf{loop}\ i) &= \mathsf{ua}\ \mathsf{succ}\ \ i \end{aligned}$$



构造 encode 正如之前提到的，我们可以轻松地通过 J 来定义 encode:

$$\text{encode}\ \_\ p \ = \ \text{J Code}\ p\ 0$$

基点 base 处的 encode 函数通常被称为**卷绕数** (winding number):

$$\text{winding} : \Omega \mathsf{S}^1 \to \mathbb{Z}$$
$$\text{winding} \ = \ \text{encode loop}$$

实际上，这个函数相当于在计算一条环路绕着圆周旋转了几圈. 这个性质可以通过 univalence 的 β 规则 uaβ 来证明.

$$\text{winding}\ \underbrace{(\text{loop} \cdot \ ...\ \cdot \text{loop})}_{\text{n 个 loop 的复合}} \ = \ \text{n}$$

注 6.5. *在这里，绕 0 圈应该理解被为常值道路 refl. 而绕负数圈则是沿着反方向的 loop^{-1} 旋转.*

构造 decode 函数 decode 的定义需要更具体的构造了. 首先，我们定义基点 base 处的 decode:

$$\text{looping} : \mathbb{Z} \to \Omega \mathsf{S}^1$$

直观上，looping 的定义是满足下面等式的函数. 换句话说，它将整数 n 映射到沿着 loop 绕 n 圈的环路.

$$\mathsf{looping}\ \mathsf{n} = \underbrace{\mathsf{loop} \cdot \ldots \cdot \mathsf{loop}}_{\mathsf{n}\ \text{个}\ \mathsf{loop}\ \text{的复合}}$$

注意到根据前面的陈述，winding 和 looping 应该是一对互逆的函数，因此下面的等式成立:

$$\mathsf{winding}\ (\mathsf{looping}\ \mathsf{n}) = \mathsf{n}$$

一旦完成 looping 的定义，我们就可以进行模式匹配了:

$$\begin{aligned} \mathsf{decode}\ \ \mathsf{base}\ \ \ \ &= \mathsf{looping} \\ \mathsf{decode}\ (\mathsf{loop}\ i) &= \{\ldots\} \end{aligned}$$

需要注意的是，我们还没有给出 looping 的严格定义，也没有补全模式匹配中 {...} 具体的表达式，以及证明函数的互逆性. 我们把需要用到的构造和性质概括在下面的引理中:

引理 6.6. *我们可以完成下面的任务:*

1. *构造满足上面描述的 looping 函数;*

2. *证明对每个 $n : \mathbb{Z}$ 有:*

$$\mathsf{winding}\ (\mathsf{looping}\ n) = n$$

3. *补全 decode 的模式匹配.*

我们之所以这样做，是为了提供清晰的直观理解，而避免陷入繁琐的细节中. 这些构造依赖于我们如何定义整数类型 $\mathbb{Z}$，以及在我们的类型论中包含多少判断相等. 如果我们将 $\mathbb{Z}$ 定义为归纳类型，那么上面关于 looping 的陈述可以很容易地被严格化. 而引理中的第 1 点和第 2 但可以通过简单的归纳法建立。

现在我们来解释引理中的第 3 点. 一种理解这个问题的方法是注意到，补全 decode 的模式匹配等价于找到这样的道路:

$$\mathsf{transport}\ (\lambda\ i \mapsto\ \ \mathsf{Code}\,(\mathsf{loop}\ i) \to \mathsf{base} = \mathsf{loop}\ i\ )\ \mathsf{looping}\ \ = \ \ \mathsf{looping}$$

换句话说，我们需要证明沿着 loop 将 looping 函数 transport 一周后得到的函数仍然等于它自身. 利用 Π 类型和道路类型的 transport 规则，以及

univalence 的 β 规则, 左侧的函数可以被更具体地表述实际上，左侧的函数可以写成:

$$\lambda\ n \mapsto \mathsf{looping}\ (\mathsf{pred}\ n) \cdot \mathsf{loop}$$

根据我们之前对 looping 的描述，我们可以证明:

$$\mathsf{looping}\ (\mathsf{pred}\ n)\ =\ \mathsf{looping}\ n \cdot \mathsf{loop}^{-1}$$

现在使用道路的乘法结构即可证明我们所需的相等性.

证明互逆性 接下来我们需要验证 encode 和 decode 是一对互逆的函数. 验证其中一侧的互逆性非常容易，只需要直接使用 J 即可:

$$\begin{aligned}&\_\ :\ \{x\ :\ \mathsf{S}^1\}\ (p\ :\ \mathsf{base} = x) \to \mathsf{decode}\ \_\ (\mathsf{encode}\ \_\ p) = p\\ &\_\ =\ \mathsf{J}\ \_\ \{\ldots\}\end{aligned}$$

这里的 {...} 应该填写一条道路:

$$\mathsf{looping}\ (\mathsf{J}\ \_\ \mathsf{refl}\ 0)\ =\ \mathsf{refl}$$

这并不困难，我们首先利用 Jβ, 将左侧化简为 looping 0. 根据 looping 的定义，我们知道这就是常值道路，

接下来我们处理另一侧. 事实上，我们的定理只用到了在 base 处的互逆性，而这就是引理6.6的第 3 条性质.

一般情形可以被视为推论. 我们可以利用 encode 在 base 处是等价，以及 S^1 的连通性来推出 encode 在整个 S^1 上都是等价:

$$(x\ :\ \mathsf{S}^1) \to \mathsf{isEquiv}\,(\mathsf{encode}\ x)$$

因为等价的单侧逆就是双侧逆，使用 decode 是 encode 的单侧逆即可完成全部证明.

□

注 6.7. 这个定理的语义是，空间 S^1 的环路空间同伦等价于整数的集合 $\mathbb{Z}$.

$$\Omega S^1 \simeq \mathbb{Z}$$

6.3 同伦群

几乎在每一门代数拓扑课程中, 计算 S^1 的基本群都是第一个真正意义上的非平凡定理.

定理 6.8. *类型* S^1 *是群胚.*

$$\mathsf{isGroupoid}\ \mathsf{S}^1$$