# 同伦类型论

数学的泛等基础

Univalent Foundations Program

中文翻译组

# 同伦类型论：数学的泛等基础

Homotopy Type Theory: Univalent Foundations of Mathematics

原作者：The Univalent Foundations Program

中文翻译组

# 译者序

本书是《Homotopy Type Theory: Univalent Foundations of Mathematics》的中文译本。

同伦类型论（Homotopy Type Theory，简称 HoTT）是 21 世纪数学基础研究的一个重要突破。它将同伦论的思想引入类型论，建立了一种全新的数学基础框架。核心的泛等公理（Univalence Axiom）断言：等价即相等，这一优雅的原理使得数学中大量显然的等同得以形式化。

本书是 2012-2013 年普林斯顿高等研究院数学泛等基础特别年活动的成果，由该领域的先驱学者们集体撰写。

翻译本书的目的是让更多中文读者能够接触和学习这一前沿领域。我们力求忠实于原文，同时使译文通顺易读。对于专业术语，我们参考了已有的中文文献，并在首次出现时标注英文原词。

由于译者水平有限，疏漏之处在所难免，恳请读者批评指正。

中文翻译组

# 目录


**译者序** **iii**

**导论** **1**

**第 1 部分　基础** **19**

**第 1 章　类型论** **21**

1.1 类型论 vs 集合论 . . . . . 21
1.2 函数类型 . . . . . 26
1.3 宇宙与族 . . . . . 29
1.4 依赖函数类型（Π-类型） . . . . . 30
1.5 积类型 . . . . . 33
1.6 依赖对类型（Σ-类型） . . . . . 38
1.7 余积类型 . . . . . 42
1.8 布尔类型 . . . . . 43
1.9 自然数 . . . . . 46
1.10 模式匹配与递归 . . . . . 50
1.11 命题即类型 . . . . . 52
1.12 恒等类型 . . . . . 59
1.12.1 路径归纳 . . . . . 61
1.12.2 路径归纳与基点路径归纳的等价性 . . . . . 65
1.12.3 不相等 . . . . . 68

**第 2 章 同伦类型论** **69**

2.1 类型是高阶群胚 . . . . . 73
2.2 函数是函子 . . . . . 84
2.3 类型族是纤维化 . . . . . 86
2.4 同伦与等价 . . . . . 88
2.5 类型构造子的高阶群胚结构 . . . . . 93
2.6 笛卡尔积类型 . . . . . 94
2.7 Σ-类型 . . . . . 98
2.8 单位类型 . . . . . 101
2.9 Π-类型与函数外延性公理 . . . . . 101
2.10 宇宙与泛等公理 . . . . . 105
2.11 恒等类型 . . . . . 107
2.12 余积 . . . . . 110
2.13 自然数 . . . . . 113
2.14 例子：结构的相等 . . . . . 115
2.14.1 提升等价 . . . . . 116
2.14.2 半群的相等 . . . . . 118
2.15 泛性质 . . . . . 119

**第 3 章 集合与逻辑** **123**

3.1 集合与 $n$-类型 . . . . . 123
3.2 命题即类型？ . . . . . 126
3.3 纯粹命题 . . . . . 129
3.4 经典逻辑与直觉主义逻辑 . . . . . 131
3.5 子集与命题调整 . . . . . 133
3.6 纯粹命题的逻辑 . . . . . 135
3.7 命题截断 . . . . . 136
3.8 选择公理 . . . . . 139
3.9 唯一选择原理 . . . . . 141
3.10 命题何时被截断？ . . . . . 142
3.11 可缩性 . . . . . 145

Notes . . . . . 148
Exercises . . . . . 149

**第 4 章 等价** **153**
4.1 拟逆 . . . . . 154
4.2 半伴随等价 . . . . . 156
4.3 双可逆映射 . . . . . 162
4.4 可缩纤维 . . . . . 163
4.5 关于等价的定义 . . . . . 164
4.6 满射与嵌入 . . . . . 165
4.7 等价的封闭性质 . . . . . 166
4.8 对象分类子 . . . . . 170
4.9 泛等性蕴含函数外延性 . . . . . 172
Notes . . . . . 175
Exercises . . . . . 175

**第 5 章 归纳** **179**
5.1 正类型简介 . . . . . 179
5.2 W-类型的唯一性 . . . . . 179
5.3 自然数 . . . . . 179
5.4 归纳原则的等价 . . . . . 179
5.5 函子化与自然性 . . . . . 179
5.6 高阶归纳类型的一般化 . . . . . 179
5.7 广义归纳定义 . . . . . 179
5.8 余归纳 . . . . . 179

**第 6 章 高阶归纳类型** **181**

**第 7 章 同伦 n-类型** **183**

**第 2 部分 数学** **185**

**第 8 章 同伦论** **187**

**第 9 章 范畴论** **189**

9.1 范畴与预范畴 . . . . . 189
9.2 函子与变换 . . . . . 189
9.3 伴随 . . . . . 189
9.4 等价 . . . . . 189
9.5 Yoneda 引理 . . . . . 189
9.6 严格范畴 . . . . . 189
9.7 dagger 范畴 . . . . . 189
9.8 Rezk 完备化 . . . . . 189
9.9 范畴的结构同一性原则 . . . . . 189

**第 10 章 集合论** **191**

10.1 -W-预拓扑 . . . . . 191
10.2 集合的范畴 . . . . . 191
10.3 基数 . . . . . 191
10.4 序数 . . . . . 191
10.5 累积层次 . . . . . 191

**第 11 章 实数** **193**

**附录 A 形式类型论** **197**

# 导论

**同伦类型论** (Homotopy type theory) 是数学的一个新分支，它以令人惊讶的方式结合了若干不同领域的诸多方面。它基于最近发现的**同伦论** (homotopy theory) 与**类型论** (type theory) 之间的联系。同伦论是代数拓扑和同调代数的延伸，与高阶范畴论有着联系；而类型论则是数理逻辑和理论计算机科学的一个分支。尽管这两者之间的联系目前正是密集研究的焦点，但越来越清楚的是，它们只是一个主题的开始，要完全理解这个主题还需要更多的时间和艰苦的工作。它涉及的主题看似遥远，如球面的同伦群、类型检查的算法，以及弱 $\infty$-群胚的定义。

同伦类型论也为数学的根基带来了新的思想。一方面，有 Voevodsky 精妙而优美的**泛等公理** (univalence axiom)。泛等公理特别蕴含了同构的结构可以被等同，这是数学家们在工作日里一直愉快使用的原则，尽管它与传统基础的官方教条不相容。另一方面，我们有**高阶归纳类型** (higher inductive types)，它们为同伦论的一些基本空间和构造提供了直接的逻辑描述：球面、柱面、截断、局部化等等。这两个思想在经典的集合论基础中都无法直接捕捉，但当它们在同伦类型论中结合时，它们允许一种全新的同伦类型的逻辑。

这暗示了一种数学基础的新概念，它具有内在的同伦内容，是数学对象的一种不变概念——并且有方便的机器实现，可以作为工作数学家的实用辅助。这就是**泛等基础** (Univalent Foundations) 纲领。本书旨在作为泛等基础基本原理的第一个系统性阐述，以及这种新推理风格的一系列例子——但不要求读者知道或学习任何形式逻辑，或使用任何计算机证明助手。

我们强调，同伦类型论是一个年轻的领域，泛等基础还在积极发展中。本书应被视为该领域某一部分在写作时的快照，而非一个完成大厦的精致

阐述。正如我们稍后将简要讨论的，同伦类型论还有许多方面尚未完全理解——有些甚至在这里没有涉及。最终的理论几乎肯定不会与本书中描述的理论完全一样，但它肯定至少会同样强大和有力；因此我们相信，泛等基础最终将成为集合论作为大多数数学家进行非形式化数学的隐含基础的可行替代方案。

## 类型论

类型论最初由 Bertrand Russell [Rus08] 发明，作为阻止当时正在研究的数学逻辑基础中悖论的手段。它在接下来的几十年中由许多人进一步发展，特别是 Church [Chu40, Chu41]，他将其与他的 $\lambda$-演算结合起来。尽管它通常不被视为经典数学的基础（集合论更为常见），但类型论仍有许多应用，特别是在计算机科学和程序设计语言理论中 [Pie02]。Per Martin-Löf [ML98, ML75, ML82, ML84] 等人发展了 Church 类型系统的预断修改，现在通常称为依赖的、构造的、直觉主义的，或简称为 *Martin-Löf 类型论*。这是我们在此考虑的系统的基础；它最初旨在作为构造数学形式化的严格框架。在下文中，我们通常会用类型论来特指这个系统及类似系统，尽管类型论作为一个学科要广泛得多（关于类型论的历史，见 [Som10, KLN04]）。

在类型论中，与集合论不同，对象使用**类型** (type) 的原始概念进行分类，类似于程序设计语言中使用的数据类型。这些精心构造的类型可用于表达被分类对象的详细规格说明，从而产生关于这些对象的推理原则。举一个非常简单的例子，乘积类型 $A \times B$ 的对象已知具有形式 $(a, b)$，因此人们自动知道如何构造它们以及如何分解它们。类似地，函数类型 $A \to B$ 的对象可以从由类型 $A$ 的对象参数化的类型 $B$ 的对象中获得，并且可以在类型 $A$ 的参数处求值。所有对象的这种严格可预测的行为（与集合论更自由的形成原则相反，允许非齐次集合）是类型论被广泛用于验证计算机程序正确性的一个方面。与类型构造相关的清晰推理原则也构成了现代**计算机证明助手** (computer proof assistants) 的基础，用于形式化数学和验证形式化证明的正确性。我们将在下文中回到类型论的这一方面。

然而，从数学的角度理解类型论的一个问题一直是**类型** (type) 的基本概念与**集合** (set) 的概念不同，其方式一直难以精确表述。我们相信，将类型视为空间而非奇怪的集合（可能不使用经典逻辑构造），从同伦论的角度

来看，是一个重大进步。特别是，它解决了理解类型的元素的相等性概念如何不同于集合的元素的相等性概念的问题。

在同伦论中，人们关注空间 及它们之间的连续映射，直到同伦为止。一对连续映射 $f : X \to Y$ 和 $g : X \to Y$ 之间的**同伦** (homotopy) 是满足 $H(x, 0) = f(x)$ 和 $H(x, 1) = g(x)$ 的连续映射 $H : X \times [0, 1] \to Y$。同伦 $H$ 可以被认为是 $f$ 到 $g$ 的连续变形。如果存在来回的连续映射，其复合同伦于相应的恒等映射，即它们在同伦意义下同构，则称空间 $X$ 和 $Y$ 是**同伦等价的** (homotopy equivalent)，记作 $X \simeq Y$。同伦等价的空间具有相同的代数不变量（例如，同调或基本群），并且被称为具有相同的**同伦型** (homotopy type)。

## 同伦类型论

同伦类型论 (HoTT) 从同伦的角度解释类型论。在同伦类型论中，我们将类型视为空间（如同伦论中所研究的）或高阶群胚，并将逻辑构造（如乘积 $A \times B$）视为这些空间上的同伦不变构造。通过这种方式，我们能够直接操作空间，而无需首先发展点集拓扑（或其任何组合替代，如单纯集理论）。为简要解释这一观点，首先考虑类型论的基本概念，即 **项** (term) $a$ 是**类型** (type) $A$，记作：

$$a : A.$$

这个表达式传统上被认为类似于：

$$a \text{ 是集合 } A \text{ 的元素。}$$

然而，在同伦类型论中，我们将其视为：

$$a \text{ 是空间 } A \text{ 的点。}$$

类似地，类型论中的每个函数 $f : A \to B$ 都被视为从空间 $A$ 到空间 $B$ 的连续映射。

我们应该强调，这些空间是纯粹从同伦角度而非拓扑角度处理的。例如，没有类型的开子集或类型元素序列的收敛的概念。我们只有同伦概念，如点之间的路径和路径之间的同伦，这在同伦论的其他模型（如单纯集）中也有意义。因此，更准确地说，我们将类型视为 $\infty$-*群胚*；这是同伦论的不

变对象的名称，可以由拓扑空间、单纯集或同伦论的任何其他模型来呈现。然而，有时使用空间和路径等拓扑词汇是方便的，只要我们记住其他拓扑概念不适用即可。

（使用短语**同伦型** (homotopy type) 来称呼这些对象也很诱人，暗示（作为类型论中的）类型从同伦角度看和从同伦论的角度看的空间的双重解释。后者与同伦型的经典含义作为空间在同伦等价下的等价类略有不同，尽管它确实保留了诸如这两个空间具有相同的同伦型等短语的含义。）

将类型解释为结构化对象而非集合的想法有着悠久的历史，并且已知可以澄清类型论的各种神秘方面。例如，将类型解释为层有助于解释类型论逻辑的直觉主义性质，而将它们解释为偏等价关系或域有助于解释其计算方面。它还意味着我们可以使用类型论推理来研究结构化对象，从而产生了丰富的范畴逻辑领域。同伦解释符合这一模式：它澄清了类型论中**同一性** (identity)（或相等性）的本质，并允许我们在同伦论的研究中使用类型论推理。

同伦解释的关键新思想是，同一类型 $A$ 的两个对象 $a, b : A$ 的逻辑同一性概念 $a = b$ 可以理解为空间 $A$ 中从点 $a$ 到点 $b$ 的路径 $p : a \rightsquigarrow b$ 的存在。这也意味着，如果两个函数 $f, g : A \to B$ 是同伦的，则它们可以被等同，因为同伦只是 $B$ 中路径 $p_x : f(x) \rightsquigarrow g(x)$ 的（连续）族，每个 $x : A$ 对应一条路径。在类型论中，对于每个类型 $A$，存在一个（以前有些神秘的）类型 $\mathsf{Id}_A$，表示 $A$ 的两个对象的同一性；在同伦类型论中，这恰好是从单位区间到 $A$ 的所有连续映射 $I \to A$ 的**路径空间** (path space) $A^I$。通过这种方式，项 $p : \mathsf{Id}_A(a, b)$ 表示 $A$ 中的路径 $p : a \rightsquigarrow b$。

同伦类型论的思想大约在 2006 年由 Awodey 和 Warren [AW09] 以及 Voevodsky [Voe06] 的独立工作中产生，但它受到 Hofmann 和 Streicher 早期的群胚解释 [HS98]的启发。事实上，高维范畴论（特别是弱 $\infty$-群胚理论）现在已知与同伦论密切相关，这是 Grothendieck 提出的，现在正被两类数学家密集研究。Awodey–Warren 和 Voevodsky 的原始语义模型使用了同伦论中的著名概念和技术，这些概念和技术现在也用于高阶范畴论，如 Quillen 模型范畴和 Kan 单纯集。

特别是，Voevodsky 构造了类型论在 Kan 单纯集中的解释，并认识到这种解释满足了他称为**泛等性** (univalence) 的另一个关键性质。这在类型

论中以前从未被考虑过（尽管 Church 的命题外延性原则被证明是它的一个非常特殊的情况，并且 Hofmann 和 Streicher 在宇宙外延性的名称下考虑过另一个特殊情况）。以新公理的形式将泛等性添加到类型论中具有深远的影响，其中许多是自然的、简化的和令人信服的。泛等公理还进一步加强了类型论的同伦观点，因为它在单纯模型和其他相关模型中成立，而在将类型视为集合的观点下失败。

## 泛等基础

非常简要地说，泛等公理的基本思想可以解释如下。在类型论中，可以有一个类型，其元素本身是类型；这样的类型称为**宇宙** (universe)，通常记为 $\mathcal{U}$。作为 $\mathcal{U}$ 的项的那些类型通常称为**小** (small) 类型。像任何类型一样，$\mathcal{U}$ 有一个同一性类型 $\mathsf{Id}_{\mathcal{U}}$，它表达小类型之间的同一性关系 $A = B$。将类型视为空间，$\mathcal{U}$ 是一个空间，其点是空间；要理解其同一性类型，我们必须问，$\mathcal{U}$ 中空间之间的路径 $p : A \rightsquigarrow B$ 是什么？泛等公理说，这样的路径对应于同伦等价 $A \simeq B$，（粗略地）如上所述。更精确地说，给定任何（小）类型 $A$ 和 $B$，除了 $A$ 与 $B$ 的同一性的原始类型 $\mathsf{Id}_{\mathcal{U}}(A, B)$ 之外，还有从 $A$ 到 $B$ 的等价的定义类型 $\mathsf{Equiv}(A, B)$。由于任何对象上的恒等映射是等价，因此存在规范映射，

$$\mathsf{Id}_{\mathcal{U}}(A, B) \to \mathsf{Equiv}(A, B).$$

泛等公理断言这个映射本身是等价。冒着过度简化的风险，我们可以简洁地陈述如下：

**泛等公理：** $(A = B) \simeq (A \simeq B)$。

换句话说，同一性等价于等价。特别是，人们可以说等价的类型是同一的。然而，这个短语有些误导，因为它可能听起来像一种骨架性条件，它将等价的概念坍缩为与同一性重合，而实际上泛等性是关于扩展同一性的概念，以便与（不变的）等价概念重合。

从同伦的角度来看，泛等性意味着具有相同同伦型的空间通过宇宙 $\mathcal{U}$ 中的路径连接，符合（小）空间的分类空间的直觉。然而，从逻辑的角度来

看，这是一个全新的思想：它说同构的东西可以被等同！数学家当然习惯于在实践中等同同构的结构，但他们通常通过记号滥用或其他一些非正式手段来这样做，知道所涉及的对象并不真正同一。但在这个新的基础方案中，这样的结构可以被正式等同，在逻辑意义上，涉及一个的每个性质或构造也适用于另一个。事实上，等同现在是明确的，性质和构造可以系统地沿着它传输。此外，进行这种等同的不同方式本身形成了一个结构，人们可以（并且应该!）考虑到这一点。

因此总的来说，对于宇宙 $\mathcal{U}$ 的点 $A$ 和 $B$（即小类型），泛等公理等同了以下三个概念：

- （逻辑）$A$ 和 $B$ 的同一性 $p : A = B$
- （拓扑）$\mathcal{U}$ 中从 $A$ 到 $B$ 的路径 $p : A \rightsquigarrow B$
- （同伦）$A$ 和 $B$ 之间的等价 $p : A \simeq B$。

### 高阶归纳类型

类型论的经典优势之一是其用于处理归纳定义结构的简单而有效的技术。最简单的非平凡归纳定义结构是自然数，它由零和后继函数归纳生成。从这个陈述可以算法地提取数学归纳原理，它刻画了自然数。更一般的归纳定义包括各种列表和良基树，每一个都由相应的归纳原理刻画。这包括某些程序设计语言中使用的大多数数据结构；因此类型论在关于后者的形式推理中的有用性。如果以非常一般的意义来理解，归纳定义还包括诸如不交并 $A+B$ 之类的例子，它可以被视为由两个注入 $A \to A+B$ 和 $B \to A+B$ 归纳地生成。在这种情况下，归纳原理是按情况分析证明，它刻画了不交并。

在同伦论中，考虑归纳定义的空间也很自然，它们不仅由点的集合生成，还由路径和更高路径的集合生成。经典地，这样的空间称为 *CW 复形*。例如，圆 $S^1$ 由单个点和从该点到自身的单个路径生成。类似地，2-球面 $S^2$ 由单个点 $b$ 和从 $b$ 处的常路径到自身的单个二维路径生成，而环面 $T^2$ 由单个点、从该点到自身的两条路径 $p$ 和 $q$，以及从 $p \cdot q$ 到 $q \cdot p$ 的二维路径生成。

通过使用同伦类型论中路径与同一性的等同，这种归纳定义的空间可

以在类型论中通过归纳原理刻画，完全类似于自然数和不交并等经典例子。由此产生的**高阶归纳类型** (higher inductive types) 提供了直接逻辑的方式来推理熟悉的空间，如球面，这（与泛等性结合）可以用于以纯形式的方式执行同伦论中的熟悉论证，如计算球面的同伦群。由此产生的证明是经典同伦论思想与经典类型论思想的结合，为两个学科都提供了新的见解。

此外，这只是冰山一角：同伦论中的许多抽象构造，如同伦余极限、悬挂、Postnikov 塔、局部化、完备化和谱化，也可以表示为高阶归纳类型。其中许多在经典上使用 Quillen 的小对象论证构造，它可以被视为无限 CW 复形呈现空间的有限算法描述方式，就像零和后继是自然数无限集的有限算法描述一样。通过小对象论证产生的空间是出了名的复杂和难以理解；类型论方法可能简单得多，通过直接访问适当的归纳原理绕过任何显式构造的需要。因此，泛等性和高阶归纳类型的结合暗示了同伦论实践中的某种革命的可能性。

### 泛等基础中的集合

我们声称泛等基础最终可以作为所有数学的基础，但到目前为止我们只讨论了同伦论。当然，有许多使用类型论而不使用新的同伦类型论特性来形式化数学的具体例子，例如最近在 Coq [GAA+13] 中形式化的 Feit–Thompson 奇阶定理。

但传统观点是，数学建立在集合论之上，从某种意义上说，所有数学对象和构造都可以编码到 Zermelo–Fraenkel 集合论 (ZF) 这样的理论中。然而，现在已经确立的是，对于集合论本身之外的大多数数学，ZF 中集合的复杂层次成员结构实际上是不必要的：一个更结构化的理论，如 Lawvere 的集合范畴的初等理论 [Law05]，就足够了。

在泛等基础中，基本对象是同伦类型而不是集合，但我们可以定义一类表现得像集合的类型。从同伦角度来看，这些可以被认为是每个连通分量都是可缩的空间，即那些同伦等价于离散空间的空间。这样的集合的范畴满足 Lawvere 的公理（或相关公理，取决于理论的细节）是一个定理。因此，可以在类似 ETCS 的理论中表示的任何数学（经验表明，这基本上是所有数学）同样可以在泛等基础中表示。

这支持了泛等基础至少与现有数学基础一样好的主张。在泛等基础中

工作的数学家可以以熟悉的方式从集合构建结构，更一般的同伦类型在基础背景中等待，直到需要它们。因此，本书中的大多数应用都被选择为泛等基础有新贡献的领域，使其与现有基础系统区分开来。

不出所料，同伦论和范畴论是其中两个，但也许不太明显的是，泛等基础即使在集合论和实分析等学科中也有新的和有趣的东西可以提供。例如，泛等公理允许我们等同同构的结构，而高阶归纳类型允许通过它们的泛性质直接描述对象。因此，我们通常可以避免诉诸任意选择的代表或超限迭代构造。事实上，甚至 ZF 集合论中的研究对象也可以在泛等基础的集合内部通过这样的归纳泛性质来刻画。

## 非形式类型论

经典数学家在面对学习类型论时经常遇到的一个困难是，它通常以完全或部分形式化的演绎系统呈现。这种风格对于证明论研究非常有用，但对于应用、非形式推理并不特别方便。它甚至对大多数工作数学家来说都不熟悉，即使是那些可能对数学基础感兴趣的人。本工作的一个目标是在泛等基础中发展一种非形式的数学风格，它既严格又精确，但也更接近日常数学的语言和表达风格。

在当今数学中，人们通常以一种原则上可以在初等集合论系统（如 ZFC）中形式化的方式构造和推理数学对象——至少给定足够的独创性和耐心。在大多数情况下，人们甚至不需要意识到这种可能性，因为它在很大程度上与证明完全严格的条件一致（从所有数学家通过教育和经验直观理解的意义上来说）。但是，人们确实需要学会对非形式集合论的几个方面保持谨慎：使用太大或太模糊而无法成为集合的集合；选择公理及其等价物；甚至（对于本科生）反证法；等等。采用新的基础系统（如同伦类型论）作为非形式推理的*隐含形式基础*将需要调整一些人的直觉和实践。本文旨在作为这种新数学的一个例子，它仍然是非形式的，但现在原则上可以在同伦类型论而不是 ZFC 中形式化，同样需要足够的独创性和耐心。

值得强调的是，在这个新系统中，这种形式化可以有真正的实际好处。类型论的形式系统适合计算机系统，并已在现有的证明助手中实现。证明助手是一种计算机程序，它引导用户构造完全形式的证明，只允许有效的推理步骤。它还提供一定程度的自动化，可以搜索库以查找现有定理，甚

至可以从由此产生的（构造性）证明中提取数值算法 。

我们相信，泛等基础纲领的这一方面使其与其他基础方法区分开来，可能为工作数学家提供新的实用性。事实上，基于较旧类型论的证明助手已经被用于形式化重要的数学证明，如四色定理 和 Feit–Thompson 定理。泛等基础的计算机实现目前正在进行中（就像理论本身一样）。然而，即使是其当前可用的实现（这些主要是对现有证明助手（如 Coq 和 Agda ）的小修改）也已经证明了它们的价值，不仅在已知证明的形式化方面，而且在发现新证明方面。事实上，本书中描述的许多证明实际上首先是以证明助手中的完全形式化形式完成的，现在才第一次被非形式化——这是形式数学和非形式数学之间通常关系的逆转。

可以想象，在不太遥远的将来，数学家将能够通过在泛等基础系统内工作（在证明助手中形式化）来验证自己论文的正确性，并且这样做将变得像在 TEX 中排版自己的论文一样自然。原则上，这对于任何其他基础系统都可以同样成立，但我们相信使用泛等基础更实际可行，正如本工作及其形式对应物所见证的那样。

## 构造性

经典基础和类型论之间最显著的区别之一是**证明相关性** (proof relevance) 的思想，根据这种思想，数学陈述甚至它们的证明都成为第一类数学对象。在类型论中，我们用类型表示数学陈述，这些类型可以同时被视为数学构造和数学断言，这种概念也称为**命题即类型** (propositions as types)。因此，我们可以将项 $a : A$ 同时视为类型 $A$ 的元素（或在同伦类型论中，空间 $A$ 的点），同时也是命题 $A$ 的证明。举个例子，假设我们有集合 $A$ 和 $B$（离散空间），考虑陈述 $A$ 同构于 $B$。在类型论中，这可以表示为：

$$\mathsf{Iso}(A,B) :\equiv \sum_{(f:A\to B)} \sum_{(g:B\to A)} \Big( \big(\textstyle\prod_{(x:A)} g(f(x)) = x\big) \times \big(\textstyle\prod_{(y:B)} f(g(y)) = y\big) \Big).$$

在这里将类型构造子 $\Sigma, \Pi, \times$ 分别读作存在、对所有和与，得到 $A$ 和 $B$ 是同构的的通常表述；另一方面，将它们读作和与积，得到 $A$ 和 $B$ 之间的所有同构的类型！要证明 $A$ 和 $B$ 是同构的，就要构造一个证明 $p : \mathsf{Iso}(A, B)$，

因此这与构造 $A$ 和 $B$ 之间的同构是一样的，即展示一对函数 $f, g$ 以及它们的复合是相应恒等映射的证明。后面的证明依次只是适当类型的同伦。通过这种方式，证明一个命题与构造某个特定类型的元素是一样的。特别地，证明形式为 $A$ 且 $B$ 的陈述就是证明 $A$ 和证明 $B$，即给出类型 $A \times B$ 的元素。证明 $A$ 蕴含 $B$ 就是找到 $A \to B$ 的元素，即从 $A$ 到 $B$ 的函数（确定从 $A$ 的证明到 $B$ 的证明的映射）。

命题即类型逻辑是灵活的，支持许多变体，例如仅使用类型的子类来表示命题。在同伦类型论中，有从这样一个事实产生的自然子类，即所有类型的系统，就像经典同伦论中的空间一样，根据它们的高阶同伦结构存在或坍缩的维度进行分层。特别地，Voevodsky 发现了**同伦 $n$-类型** (homotopy $n$-types) 的纯类型论定义，对应于在维度 $n$ 以上没有非平凡同伦信息的空间。（0-类型是前面提到的满足 Lawvere 公理的集合。）此外，使用高阶归纳类型，我们可以将类型普遍地截断为 $n$-类型；在经典同伦论中，这将是它的第 $n$ 个 Postnikov 截面。对于逻辑特别重要的是同伦 $(-1)$-类型的情况，我们称之为**纯命题** (mere propositions)。经典地，每个 $(-1)$-类型要么是空的要么是可缩的；我们将这些可能性分别解释为真值假和真。

使用所有类型作为命题会产生非常构造性的逻辑概念；更多关于此的信息，见 [Kol32, TvD88a, TvD88b]。例如，某物存在的每个证明都携带足够的信息来实际找到这样的对象；并且 $A$ 或 $B$ 成立的每个证明要么是 $A$ 成立的证明，要么是 $B$ 成立的证明。因此，从每个证明我们都可以自动提取算法；这在计算机编程的应用中可能非常有用。

然而，另一方面，这种逻辑确实与数学中对存在性证明的传统理解有所不同。特别是，它不能忠实地表示某些重要的经典推理原则，如选择公理和排中律。对于这些，我们需要使用 $(-1)$-截断逻辑，其中只有同伦 $(-1)$-类型表示命题。

更具体地说，一方面考虑**选择公理** (axiom of choice)：如果对于每个 $x : A$ 存在 $y : B$ 使得 $R(x, y)$，则存在函数 $f : A \to B$ 使得对所有 $x : A$ 我们有 $R(x, f(x))$。纯命题即类型概念的存在足够强大，使得这个陈述是可证明的——然而它没有通常选择公理的所有结果。但是，在 $(-1)$-截断逻辑中，这个陈述不是自动真的，而是一个强假设，具有与经典集合论中的对应物相同类型的结果。

另一方面，考虑**排中律** (law of excluded middle)：对所有 $A$，要么 $A$ 要么非 $A$。在纯命题即类型逻辑中解释这个会产生一个与泛等公理不一致的陈述。因为既然证明 $A$ 意味着展示它的一个元素，这个假设将给出从每个非空类型中统一选择一个元素的方法——一种 Hilbert 选择算子。泛等性意味着由这样的选择算子选择的 $A$ 的元素必须在 $A$ 的所有自等价下不变，因为这些被等同为自同一性，并且每个操作都必须尊重同一性；但显然某些类型有没有不动点的自同构，例如我们可以交换两元素类型的元素。然而，$(-1)$-截断的排中律虽然也不是自动真的，但可以与大多数经典数学中的相同结果一致地假设。

换句话说，虽然纯命题即类型逻辑在上述强算法意义上是构造性的，但默认的 $(-1)$-截断逻辑在不同的意义上是构造性的（即 Heyting 以直觉主义名义形式化的逻辑的意义）；对于后者，我们可以自由地添加选择公理和排中律，以获得可以称为经典的逻辑。因此，同伦类型论与构造性和经典逻辑概念兼容，以及更多其他概念。事实上，同伦视角揭示了经典逻辑和构造逻辑可以共存，作为不同系统的谱的端点，在两者之间有无限多的可能性（同伦 $n$-类型，其中 $-1 < n < \infty$）。我们可以说 $\mathsf{LEM}_n$ 和 $\mathsf{AC}_n$，其中 $\mathsf{AC}_\infty$ 是可证明的，$\mathsf{LEM}_\infty$ 与泛等性不一致，而 $\mathsf{AC}_{-1}$ 和 $\mathsf{LEM}_{-1}$ 是经典数学家熟悉的版本（因此在大多数情况下，当没有给出下标时，假设下标为 $(-1)$ 是适当的）。事实上，甚至可以有有用的系统，其中只有某些类型满足这些进一步的经典原则，而类型一般仍然是构造的。

值得强调的是，泛等基础不要求使用构造或直觉主义逻辑。依赖于排中律和选择公理的大多数经典数学都可以在泛等基础中执行，只需假设这两个原则成立（以其适当的 $(-1)$-截断形式）。然而，类型论确实鼓励在不必要时避免这些原则，原因有几个。

首先，每个数学家都知道，使用更少假设证明的定理更强大，因为它适用于更多例子。AC 和 LEM 的情况也不例外：类型论允许许多有趣的非标准模型，例如在层拓扑中，其中经典性原则如 AC 和 LEM 往往失败。同伦类型论在高阶拓扑中允许类似的模型，如 [TV02, Rez05, Lur09]中研究的那样。因此，如果我们避免使用这些原则，我们证明的定理将在所有这样的模型内部有效。

其次，类型论的另一个附加优点是其可计算特性。除了作为数学的基

础，类型论还是计算的形式理论，可以被视为强大的程序设计语言。从这个角度来看，系统的规则不能像集合论公理那样任意选择：它们之间必须有和谐，允许所有证明作为程序执行。我们还没有完全理解同伦类型论引入的新原则，如泛等性和高阶归纳类型，从这个角度来看，但基本轮廓正在浮现；例如，见 [LH12]。然而，长期以来已知的是，诸如 AC 和 LEM 之类的原则从根本上与可计算性对立，因为它们直言不讳地断言某些东西存在，而没有给出任何计算它们的方法。因此，避免它们对于维持类型论作为计算理论的特性是必要的。

幸运的是，构造性推理并不像看起来那么困难。在某些情况下，只需重新表述一些定义，定理就可以变得构造性，其证明也更优雅。此外，在泛等基础中，这似乎更经常发生。例如：

(i) 在集合论基础中，同伦论和范畴论的各个点需要选择公理来执行超限构造。但使用高阶归纳类型，我们可以直接和构造性地编码这些构造。特别是，第 8 章中的综合同伦论都不需要 LEM 或 AC 。

(ii) 在集合论基础中，陈述每个完全忠实且本质满射的函子是范畴的等价等价于选择公理。但使用泛等公理，它只是*真的*；见 第 9 章。

(iii) 在集合论中，需要各种迂回来获得规范地表示集合和良序集的同构类的基数和序数概念——可能涉及选择公理或基础公理。但使用泛等性和高阶归纳类型，我们可以通过截断宇宙直接获得这样的代表；见第 10 章。

(iv) 在集合论基础中，将实数定义为 Cauchy 序列的等价类需要排中律或（可数）选择公理才能表现良好。但使用高阶归纳类型，我们可以给出这个定义的一个版本，它表现良好并避免任何选择原则；见 第 11 章。

当然，这些简化也可以被视为证据，表明新方法最终不会真正被证明是构造性的。然而，我们再次强调，读者不必关心或担心构造性才能阅读本书。重点是，在上述所有例子中，我们给出的理论版本都有独立的优势，无论 LEM 和 AC 是否被假定可用。如果达到，构造性将是一个额外的奖励。

鉴于这种关于添加新原则（如泛等性、高阶归纳类型、AC 和 LEM ）的讨论，人们可能想知道由此产生的系统是否保持一致。（类型论相对于集合论的原始优点之一是它可以通过证明论手段被视为一致的）。与任何基础

系统一样，一致性是一个相对问题：相对于什么一致？简短的答案是，本书中考虑的所有构造和公理在 Kan 复形范畴中都有模型，这归功于 Voevodsky [KLV12]（对于高阶归纳类型见 [LS17]）。因此，已知它们相对于 ZFC（以及我们需要多少不可达基数 嵌套泛等宇宙）是一致的。给出这种一致性的更传统的类型论说明是正在进行的工作（见，例如， [LH12, BCH13]）。

我们在表 1中总结了类型论操作的不同观点。

| 类型 | 逻辑 | 集合 | 同伦 |
|---|---|---|---|
| $A$ | 命题 | 集合 | 空间 |
| $a : A$ | 证明 | 元素 | 点 |
| $B(x)$ | 谓词 | 集合族 | 纤维化 |
| $b(x) : B(x)$ | 条件证明 | 元素族 | 截面 |
| $\mathbf{0}, \mathbf{1}$ | $\bot, \top$ | $\emptyset, \{\emptyset\}$ | $\emptyset, *$ |
| $A + B$ | $A \vee B$ | 不交并 | 余积 |
| $A \times B$ | $A \wedge B$ | 序对集合 | 乘积空间 |
| $A \to B$ | $A \Rightarrow B$ | 函数集合 | 函数空间 |
| $\sum_{(x:A)} B(x)$ | $\exists_{x:A} B(x)$ | 不交和 | 全空间 |
| $\prod_{(x:A)} B(x)$ | $\forall_{x:A} B(x)$ | 乘积 | 截面空间 |
| $\mathsf{Id}_A$ | 相等 $=$ | $\{ (x, x) \mid x \in A \}$ | 路径空间 $A^I$ |

表 1: 比较类型论操作的不同观点

## 开放问题

对于那些有兴趣为这个新数学分支做出贡献的人来说，了解有许多有趣的开放问题可能会令人鼓舞。

也许其中最紧迫的是 Voevodsky 在 [Voe12] 中提出的泛等公理的构造性。类型论的基本系统遵循 Gentzen 自然演绎的结构。逻辑联结词由其引入规则定义，并具有由计算规则证明合理的消除规则。遵循这种模式，并使用 Tait 的可计算性方法（最初设计用于分析 Gödel 的辩证解释），人们

可以证明类型论的规范化性质。这反过来意味着重要性质，如类型检查的可判定性（一个关键性质，因为类型检查对应于证明检查，人们可以争辩说我们应该能够在看到证明时识别证明），以及所谓的正则性性质，即自然数类型的任何封闭项都归约为数字。当人们用公理（如函数外延性公理或泛等公理）扩展类型论时，这最后一个性质和引入/消除规则的统一结构就丢失了。Voevodsky 已经制定了一个与这个关于用泛等公理扩展的类型论的正则性问题相关的精确数学猜想：给定自然数类型的封闭项，是否总是可以找到一个数字和这个项等于这个数字的证明，其中这个相等的证明本身可以使用泛等公理？更一般地说，一个重要问题是是否可能提供泛等公理的构造性证明。如果添加其他同伦启发的构造（如高阶归纳类型），情况又如何？这些问题目前仍然开放，尽管目前正在开发试图找到答案的方法。

另一个基本问题是处理本质上是集合（即离散空间）的类型（如自然数）的困难，只包含平凡路径。目前，同伦类型论实际上只能刻画空间直到同伦等价，这意味着这些离散空间可能只是同伦等价于离散空间。类型论地说，这意味着有许多路径等于自反性，但不判断性地等于它（关于判断性地的含义，见 第 1.1 节）。虽然这种同伦不变性有优势，但这些无意义的同一性项确实给论证和构造带来了不必要的复杂性，因此拥有系统的方法来消除或坍缩它们会很方便。

一个更专业但同样重要的问题是同伦类型论与目前在高阶范畴论和同伦论交汇处发生的关于**高阶拓扑** (higher toposes) 的研究之间的关系。在熟悉这两个主题的人中，越来越相信它们密切相关。例如，泛等宇宙的概念应该与对象分类器的概念一致，而高阶归纳类型应该是局部可呈现性的初等反映。更一般地，同伦类型论应该是 $(\infty, 1)$-拓扑的内语言，正如直觉主义高阶逻辑是普通 1-拓扑的内语言一样。尽管有这种普遍共识，但细节仍有待解决——特别是，相干性和严格性问题仍有待解决——这样做无疑会导致对这两个概念的进一步洞察。

但到目前为止，要做的最大工作领域是在这个新系统中持续形式化日常数学。最近在形式化基础同伦论和范畴论的一些事实方面的成功令人鼓舞；其中一些在 第 8 章 和第 9 章 中描述。显然，然而，还有很多工作要做。

同伦类型论社区在 http://homotopytypetheory.org 维护网站和群

博客，以及讨论邮件列表。新人总是受欢迎的！

## 如何阅读本书

本书分为两个部分。第1部分，基础，发展同伦类型论的基本概念。这是在其上构建特定主题发展的数学基础，并且是理解泛等基础方法所必需的。对于程序员来说，这是库代码。由于泛等基础是一种新的和不同的数学，其基本概念需要一些时间来适应；因此 第1部分相当广泛。

第2部分，数学，由四章组成，这些章节建立在 第1部分的基本概念之上，以展示我们可以用泛等基础在四个不同的数学领域做的一些新事情：同伦论（第 8 章）、范畴论（第 9 章）、集合论（第 10 章）和实分析（第 11 章）。第2部分中的章节或多或少是相互独立的，尽管偶尔一个会使用另一个中证明的引理。

想要认真理解泛等基础并能够在其中工作的读者最终将不得不阅读并理解 第1部分的大部分内容。然而，只想了解泛等基础及其能做什么的读者在到达 第2部分中的主体之前必须阅读超过 200 页，这是可以理解的退缩。幸运的是，并非 第1部分的所有内容都是阅读 第2部分中章节所必需的。第2部分中的每一章都以其主题、泛等基础对它的贡献以及来自 第1部分的必要背景的简要概述开始，因此勇敢的读者可以立即转到他们喜欢的主题的相应章节。对于那些想要比这更深入地理解 第2部分中的一个或多个章节，但还没有准备好阅读所有 第1部分的人，我们在这里提供 第1部分的简要总结，并备注哪些部分对于 第2部分中的哪些章节是必要的。

第 1 章是关于类型论的基本概念，在任何同伦解释之前。熟悉 Martin-Löf 类型论的读者可以快速浏览它以了解我们使用的理论的细节。然而，没有类型论经验的读者需要阅读 第 1 章，因为类型论与集合论等其他基础之间有许多微妙的差异。

第 2 章介绍了类型论的同伦观点，以及支持这一观点的基本概念，并描述了 第 1 章中类型论的每个组件的同伦行为。它还介绍了*泛等公理*（第 2.10 节）——同伦类型论的两个基本创新之一。因此，它非常基础，我们鼓励每个人阅读它，特别是 第 2.1 节 至 第 2.4 节。

第 3 章描述了我们如何在同伦类型论中表示逻辑，以及它与经典逻辑以及构造和直觉主义逻辑的联系。这里我们定义了排中律、选择公理和命

题大小调整公理（尽管在大多数情况下，我们不需要在本书的其余部分假设这些），以及对于表示传统逻辑至关重要的命题截断。本章是 第 10 章 和第 11 章 的重要背景，对于 第 9 章不太重要，对于 第 8 章不太必要。

第 4 章 和第 5 章 详细研究两个特殊主题：等价（及相关概念）和广义归纳定义。虽然这些本身是重要的主题，并提供了对同伦类型论的更深入理解，但在大多数情况下，它们对于 第2部分不是必需的。只有 第 4 章中的几个引理在这里和那里使用，而 ?? 和第 5.1 节 和第 5.6 节 中的一般讨论有助于提供 第 6 章所需的直觉。第 5.6 节中讨论的广义归纳定义也在第 10 章 和第 11 章 的几个地方使用。

第 6 章介绍了同伦类型论的第二个基本创新——高阶归纳类型——并给出了许多例子。高阶归纳类型是 第 8 章中研究的主要对象，一些特定的高阶归纳类型在 第 10 章 和第 11 章 中发挥着重要作用。它们对于 第 9 章不太必要，尽管在 第 9.8 节中使用了一个例子。

最后，第 7 章讨论同伦 $n$-类型和相关概念，如 $n$-连通类型。这些概念对于 第 8 章很重要，但在 第2部分的其余部分不太重要，尽管某些引理的 $n = -1$ 情况在 第 10.1 节中使用。

这完成了 第1部分。如上所述，第2部分由四个基本无关的章节组成，每个章节描述泛等基础对特定主题的贡献。

在 第2部分的章节中，第 8 章（同伦论）也许是最激进的。泛等基础有一种非常不同的综合同伦论方法，其中同伦型是基本对象（即类型），而不是使用拓扑空间或其他一些集合论模型来构造。这使得经典代数拓扑定理的新证明风格成为可能，我们提供了一些样本，从 $\pi_1(\mathbb{S}^1) = \mathbb{Z}$ 到 Freudenthal 悬挂定理。

在 第 9 章（范畴论）中，我们发展了一些基本的（1-）范畴论，坚持泛等公理的原则，即相等即同构。这产生了令人愉快的效果，确保所有定义和构造在范畴等价下自动不变：事实上，等价的范畴是相等的，就像等价的类型是相等的一样。（它还与高阶范畴论和高阶拓扑理论有联系。）

第 10 章（集合论）研究泛等基础中的集合。集合范畴具有其通常的性质，因此为不需要同伦或高范畴结构的任何数学提供基础。我们还观察到，泛等性使基数和序数更加愉快，高阶归纳类型产生满足 Zermelo–Fraenkel 集合论通常公理的累积层次。

在 第 11 章（实数）中，我们总结了 Dedekind 实数的构造，然后观察到高阶归纳类型允许 Cauchy 实数的定义，避免了构造数学中的一些相关问题。然后我们勾画了 Conway 超实数的类似方法。

本书每一章都以注释部分结尾，其中收集了历史评论、文献参考和结果归属（尽可能）。我们还在每章结尾包含了习题，以帮助读者熟悉在泛等基础中进行数学。

最后，回想一下，本书是由大量人员作为大规模协作努力编写的。我们已尽最大努力在术语和符号方面实现一致性，并将数学置于逻辑流动的线性序列中，但很可能仍存在一些不完美之处。我们请求读者原谅任何此类不妥之处，并欢迎对下一版的改进建议。

# 第 1 部分

# 基础

# 第 1 章　类型论

## 1.1　类型论 vs 集合论

同伦类型论（除其他方面外）是数学的一种基础语言，即 Zermelo–Fraenkel 集合论的一种替代方案。然而，它在几个重要方面与集合论的行为不同，这需要一些时间来适应。仔细解释这些差异需要我们在此处比本书其余部分更加形式化。如导论所述，我们的目标是*非形式地*书写类型论；但对于习惯于集合论的数学家来说，开始时更精确可以帮助避免一些常见的误解和错误。

我们注意到，集合论基础有两个层次：一阶逻辑的演绎系统，以及在这个系统内部表述的特定理论的公理，如 ZFC。因此，集合论不仅仅是关于集合的，而是关于集合（第二层的对象）和命题（第一层的对象）之间的相互作用。

相比之下，类型论是它自己的演绎系统：它不需要在任何上层结构（如一阶逻辑）内部表述。类型论不是像集合论那样有两个基本概念——集合和命题，而是只有一个基本概念：*类型*。命题（我们可以证明、反驳、假设、否定等的陈述[1]）通过第 13 页表 1所示的对应关系与特定类型等同。因此，*证明定理*的数学活动被等同于*构造对象*的数学活动的一个特例——在这种情况下，是构造一个表示命题的类型的居民。

这引导我们到类型论和集合论之间的另一个区别，但要解释它，我们

[1]令人困惑的是，将命题一词与定理同义使用也是一种常见做法（可追溯到欧几里得）。我们将限制自己使用逻辑学家的用法，根据这种用法，*命题*是*可被证明*的陈述，而*定理*（或引理或推论）是*已被证明*的这样的陈述。因此 $0 = 1$ 及其否定 $\neg(0 = 1)$ 都是命题，但只有后者是定理。

必须先简要介绍一下演绎系统的一般概念。非形式地说，演绎系统是推导称为**判断**的事物的**规则**的集合。如果我们把演绎系统想象成一个形式游戏，那么判断就是我们通过遵循游戏规则所达到的局面。我们也可以把演绎系统想象成一种代数理论，在这种情况下，判断是元素（像群的元素），而演绎规则是运算（像群乘法）。从逻辑的角度来看，判断可以被认为是外部陈述，存在于元理论中，与理论本身的内部陈述相对。

在一阶逻辑的演绎系统中（集合论基于此），只有一种判断：给定命题有证明。也就是说，每个命题 $A$ 产生一个判断 $A$ 有证明，所有判断都是这种形式。一阶逻辑的规则，如从 $A$ 和 $B$ 推出 $A \wedge B$，实际上是证明构造的规则，它说的是：给定判断 $A$ 有证明和 $B$ 有证明，我们可以推导出 $A \wedge B$ 有证明。注意，判断 $A$ 有证明与命题 $A$ 本身存在于不同的层次，后者是理论的内部陈述。

类型论的基本判断，类似于 $A$ 有证明，写作 $a : A$，读作项 $a$ 具有类型 $A$，或更宽松地说 $a$ 是 $A$ 的元素（或者在同伦类型论中，$a$ 是 $A$ 的点）。当 $A$ 是表示命题的类型时，$a$ 可以被称为 $A$ 可证性的*见证*，或 $A$ 真值的*证据*（甚至是 $A$ 的*证明*，但我们将尽量避免这种容易混淆的术语）。在这种情况下，判断 $a : A$ 在类型论中可推导（对于某个 $a$）恰好当类似的判断 $A$ 有证明在一阶逻辑中可推导时（模去假设的公理和数学编码的差异，我们将在全书中讨论）。

另一方面，如果类型 $A$ 被更多地当作集合而非命题处理（尽管我们将看到，这种区别可能变得模糊），那么 $a : A$ 可以被视为类似于集合论陈述 $a \in A$。然而，有一个本质区别：$a : A$ 是一个*判断*，而 $a \in A$ 是一个*命题*。特别是，当在类型论内部工作时，我们不能做出诸如如果 $a : A$，那么不是 $b : B$ 这样的陈述，也不能反驳判断 $a : A$。

思考这一点的好方法是：在集合论中，成员关系是一种可能在两个预先存在的对象 $a$ 和 $A$ 之间成立或不成立的关系，而在类型论中，我们不能孤立地谈论一个元素 $a$：每个元素*本质上*是某个类型的元素，而且那个类型（一般而言）是唯一确定的。因此，当我们非形式地说设 $x$ 是自然数时，在集合论中这是设 $x$ 是一个东西并假设 $x \in \mathbb{N}$ 的简写，而在类型论中设 $x : \mathbb{N}$ 是一个原子陈述：我们不能在不指定其类型的情况下引入一个变量。

乍一看，这似乎是一个令人不适的限制，但它可以说更接近于设 $x$ 是

自然数的直觉数学含义。在实践中，似乎每当我们确实需要 $a \in A$ 是一个命题而非判断时，总是有一个环境集合 $B$，其中 $a$ 已知是 $B$ 的元素，而 $A$ 已知是 $B$ 的子集。这种情况在类型论中也很容易表示，只需取 $a$ 为类型 $B$ 的元素，$A$ 为 $B$ 上的谓词；参见 第 3.5 节。

类型论和集合论之间的最后一个区别是对相等性的处理。数学中熟悉的相等性概念是一个命题：例如，我们可以反驳一个等式或假设一个等式作为假设。由于在类型论中，命题是类型，这意味着相等性是一个类型：对于元素 $a, b : A$（即 $a : A$ 且 $b : A$），我们有一个类型 $a =_A b$。（当然，在同伦类型论中，这个相等性命题可能以不熟悉的方式行为：参见 第 1.12 节和第 2 章，以及本书的其余部分）。当 $a =_A b$ 有居民时，我们说 $a$ 和 $b$ 是**（命题地）相等的**。

然而，在类型论中还需要一个相等性判断，与判断 $x : A$ 存在于同一层次。这被称为**判断相等性** 或**定义相等性**，我们将其写作 $a \equiv b : A$ 或简写为 $a \equiv b$。把它理解为定义上相等是有帮助的。例如，如果我们通过方程 $f(x) = x^2$ 定义函数 $f : \mathbb{N} \to \mathbb{N}$，那么表达式 $f(3)$ 定义上等于 $3^2$。在理论内部，否定或假设一个定义相等性是没有意义的；我们不能说如果 $x$ 定义上等于 $y$，那么 $z$ 定义上不等于 $w$。两个表达式是否定义上相等只是展开定义的问题；特别是，它是算法上可判定的（尽管该算法必然是元理论的，而非理论内部的）。

随着类型论变得更加复杂，判断相等性可能比这更加微妙，但这是一个好的起点直觉。或者，如果我们把演绎系统看作代数理论，那么判断相等性就是该理论中的相等性，类似于群元素之间的相等性——唯一可能混淆的是，类型论的演绎系统内部还有一个对象（即类型 $a = b$），它在内部表现为相等性的概念。

我们想要判断相等性概念的原因是它可以控制另一种形式的判断 $a : A$。例如，假设我们给出了 $3^2 = 9$ 的证明，即我们对某个 $p$ 推导出了判断 $p : (3^2 = 9)$。那么同一个见证 $p$ 应该算作 $f(3) = 9$ 的证明，因为 $f(3)$ 定义上就是 $3^2$。表示这一点的最好方式是一条规则：给定判断 $a : A$ 和 $A \equiv B$，我们可以推导判断 $a : B$。

因此，对我们而言，类型论将是一个基于两种形式判断的演绎系统：

| 判断 | 含义 |
|---|---|
| $a : A$ | $a$ 是类型 $A$ 的对象 |
| $a \equiv b : A$ | $a$ 和 $b$ 是类型 $A$ 的定义上相等的对象 |

当引入定义相等性时，即定义一个东西等于另一个东西时，我们将使用符号 $:\equiv$。因此，上面函数 $f$ 的定义将写作 $f(x) :\equiv x^2$。

因为判断不能组合成更复杂的陈述，符号 $:$ 和 $\equiv$ 比任何其他东西结合得都更松散。[2] 因此，例如，$p : x = y$ 应该被解析为 $p : (x = y)$，这是有意义的，因为 $x = y$ 是一个类型，而不是 $(p : x) = y$，后者是没有意义的，因为 $p : x$ 是一个判断，不能等于任何东西。类似地，$A \equiv x = y$ 只能被解析为 $A \equiv (x = y)$，尽管在这种极端情况下，无论如何都应该加上括号以帮助理解。此外，稍后我们将陷入链接等式的常见记法——例如写 $a = b = c = d$ 表示 $a = b$ 且 $b = c$ 且 $c = d$，因此 $a = d$——我们也会在这种链中包含判断相等性。上下文通常足以使意图清晰。

这也许也是提到常见数学记法 $f : A \to B$ 的合适地方，该记法表达 $f$ 是从 $A$ 到 $B$ 的函数这一事实，可以被视为一个类型判断，因为我们使用 $A \to B$ 作为从 $A$ 到 $B$ 的函数类型的记法（这是类型论中的标准做法；参见 第 1.4 节）。

判断可以依赖于形如 $x : A$ 的假设，其中 $x$ 是变量 而 $A$ 是类型。例如，我们可以在假设 $m, n : \mathbb{N}$ 的情况下构造对象 $m + n : \mathbb{N}$。另一个例子是，假设 $A$ 是类型，$x, y : A$，以及 $p : x =_A y$，我们可以构造元素 $p^{-1} : y =_A x$。所有这些假设的集合称为**上下文**；从拓扑观点来看，它可以被认为是参数空间。事实上，技术上上下文必须是假设的有序列表，因为后面的假设可能依赖于前面的假设：假设 $x : A$ 只能在类型 $A$ 中出现的任何变量的假设之后做出。

如果假设 $x : A$ 中的类型 $A$ 表示一个命题，那么该假设是假说的类型论版本：我们假设命题 $A$ 成立。当类型被视为命题时，我们可以省略它们的证明的名称。因此，在上面的第二个例子中，我们可以说：假设 $x =_A y$，

[2] 在形式化类型论中，逗号和推导符号可以结合得更加松散。例如，$x : A, y : B \vdash c : C$ 被解析为 $((x : A), (y : B)) \vdash (c : C)$。然而，在本书中，我们在 第 A 章之前避免使用这种记法。

我们可以证明 $y =_A x$。然而，由于我们正在做证明相关的数学，我们将经常把证明作为对象来引用。例如，在上面的例子中，我们可能想要确立 $p^{-1}$ 连同传递性和自反性的证明表现得像一个群胚；参见 第 2 章。

注意，在假设这个词的这种含义下，我们可以假设一个命题相等性（通过假设变量 $p : x = y$），但我们不能假设一个判断相等性 $x \equiv y$，因为它不是可以有元素的类型。然而，我们可以做一些看起来像假设判断相等性的其他事情：如果我们有一个涉及变量 $x : A$ 的类型或元素，那么我们可以用任何特定元素 $a : A$ 代换 $x$ 以获得一个更具体的类型或元素。我们有时会使用诸如现在假设 $x \equiv a$ 这样的语言来指代这个代换过程，尽管它不是上面引入的技术意义上的假设。

同样，我们也不能证明一个判断相等性，因为它不是我们可以展示见证的类型。然而，我们有时会将判断相等性作为定理的一部分来陈述，例如存在 $f : A \to B$ 使得 $f(x) \equiv y$。这应该被视为做出两个独立的判断：首先我们对某个元素 $f$ 做出判断 $f : A \to B$，然后我们做出额外的判断 $f(x) \equiv y$。

在本章的其余部分，我们尝试给出类型论的非形式表述，足以满足本书的目的；我们在 第 A 章中给出更形式化的说明。除了一些相当明显的规则（如判断上相等的东西总是可以相互代换）之外，类型论的规则可以分组为*类型构造子*。每个类型构造子由构造类型的方式（可能使用先前构造的类型）组成，以及构造和行为规则用于该类型的元素。在大多数情况下，这些规则遵循相当可预测的模式，但我们不会在此处试图使其精确；然而参见 第 1.5 节的开头以及 第 5 章。

本章介绍的类型论的一个重要方面是，它完全由*规则*组成，没有任何*公理*。在用判断描述演绎系统时，*规则*允许我们从一组判断推导出另一个判断，而*公理*是我们一开始就给定的判断。如果我们把演绎系统想象成一个形式游戏，那么规则是游戏的规则，而公理是起始位置。如果我们把演绎系统想象成代数理论，那么规则是理论的运算，而公理是该理论某个特定自由模型的*生成元*。

在集合论中，唯一的规则是一阶逻辑的规则（如允许我们从 $A$ 有证明和 $B$ 有证明推导出 $A \wedge B$ 有证明的规则）：关于集合行为的所有信息都包含在公理中。相比之下，在类型论中，通常是*规则*包含所有信息，不需要公理。例如，在 第 1.5 节中，我们将看到有一条规则允许我们从 $a : A$ 和

$b : B$ 推导出判断 $(a, b) : A \times B$，而在集合论中，类似的陈述将是（配对公理的结果）。

仅使用规则来表述类型论的优势在于规则是程序性的。特别是，这个性质使类型论的良好计算性质（如正则性）成为可能（尽管它不自动保证）。然而，虽然这种风格对传统类型论有效，但我们还不理解如何以这种方式表述同伦类型论所需的一切。特别是，在 第 2.9 节 和第 2.10 节 和第 6 章中，我们将不得不通过引入额外的公理来增强本章介绍的类型论规则，特别是*泛等公理*。然而，在本章中，我们限于传统的基于规则的类型论。

## 1.2 函数类型

给定类型 $A$ 和 $B$，我们可以构造**函数**的类型 $A \to B$，其定义域为 $A$，陪域为 $B$。我们有时也将函数称为**映射**。与集合论不同，函数不是定义为函数关系；相反，它们是类型论中的原始概念。我们通过规定我们可以对函数做什么、如何构造它们以及它们诱导什么等式来解释函数类型。

给定函数 $f : A \to B$ 和定义域的元素 $a : A$，我们可以**应用** 函数以获得陪域 $B$ 的元素，记作 $f(a)$，称为 $f$ 在 $a$ 处的**值**。在类型论中，省略括号并简单地将 $f(a)$ 写作 $f\,a$ 是常见的，我们有时也会这样做。

但是我们如何构造 $A \to B$ 的元素呢？有两种等价的方式：要么通过直接定义，要么使用 $\lambda$-抽象。通过定义引入函数 意味着我们通过给它一个名字——比如 $f$——来引入一个函数，并通过给出方程

$$f(x) :\equiv \Phi \tag{1.2.1}$$

来定义 $f : A \to B$，其中 $x$ 是变量 而 $\Phi$ 是可能使用 $x$ 的表达式。为了使这有效，我们必须在假设 $x : A$ 的情况下检验 $\Phi : B$。

现在我们可以通过用 $a$ 替换 $\Phi$ 中的变量 $x$ 来计算 $f(a)$。作为例子，考虑由 $f(x) :\equiv x + x$ 定义的函数 $f : \mathbb{N} \to \mathbb{N}$。（我们将在 第 1.9 节中定义 $\mathbb{N}$ 和 $+$。）那么 $f(2)$ 判断上等于 $2 + 2$。

如果我们不想为函数引入名字，我们可以使用 **$\lambda$-抽象**。给定类型为 $B$ 的表达式 $\Phi$，它可能使用 $x : A$，如上所述，我们写 $\lambda(x : A).\,\Phi$ 来表示

由 (1.2.1) 定义的同一个函数。因此，我们有

$$(\lambda(x:A).\,\Phi) : A \to B.$$

对于前一段的例子，我们有类型判断

$$(\lambda(x:\mathbb{N}).\,x + x) : \mathbb{N} \to \mathbb{N}.$$

作为另一个例子，对于任何类型 $A$ 和 $B$ 以及任何元素 $y : B$，我们有一个**常值函数** $(\lambda(x:A).\,y) : A \to B$。

我们通常省略 $\lambda$-抽象中变量 $x$ 的类型，写成 $\lambda x.\,\Phi$，因为类型 $x : A$ 可以从函数 $\lambda x.\,\Phi$ 具有类型 $A \to B$ 的判断推断出来。按照惯例，变量绑定 $\lambda x.$ 的作用域 是表达式的整个剩余部分，除非用括号分隔。因此，例如，$\lambda x.\,x + x$ 应该被解析为 $\lambda x.\,(x + x)$，而不是 $(\lambda x.\,x) + x$（在这种情况下，后者无论如何都是类型错误的）。

另一种等价记法是

$$(x \mapsto \Phi) : A \to B.$$

我们有时也可以在表达式 $\Phi$ 中使用空白 – 代替变量，来表示隐式的 $\lambda$-抽象。例如，$g(x, –)$ 是写 $\lambda y.\,g(x, y)$ 的另一种方式。

现在 $\lambda$-抽象是一个函数，所以我们可以将它应用到参数 $a : A$。然后我们有以下**计算规则**[3]，这是一个定义等式：

$$(\lambda x.\,\Phi)(a) \equiv \Phi'$$

其中 $\Phi'$ 是将 $\Phi$ 中所有 $x$ 的出现替换为 $a$ 的表达式。继续上面的例子，我们有

$$(\lambda x.\,x + x)(2) \equiv 2 + 2.$$

注意，从任何函数 $f : A \to B$，我们可以构造一个 lambda 抽象函数 $\lambda x.\,f(x)$。由于这按定义是将 $f$ 应用到其参数的函数，我们认为它定义上等于 $f$：[4]

$$f \equiv (\lambda x.\,f(x)).$$

[3]这个等式的使用通常被称为 $\beta$-**变换** 或 $\beta$-**归约**。

[4]这个等式的使用通常被称为 $\eta$-**变换** 或 $\eta$-**展开**。

这个等式是**函数类型的唯一性原则**，因为它表明 $f$ 被其值唯一确定。

通过使用 λ-抽象，带有显式参数的定义引入函数可以归约为简单定义：即，我们可以将

$$f(x) :\equiv \Phi$$

对 $f : A \to B$ 的定义读作

$$f :\equiv \lambda x.\, \Phi.$$

当进行涉及变量的计算时，当用也涉及变量的表达式替换变量时，我们必须小心，因为我们想要保持表达式的绑定结构。所谓*绑定结构*是指由 λ、Π 和 Σ（后者我们很快会遇到）等绑定器在变量引入处和使用处之间生成的不可见链接。作为例子，考虑定义为

$$f(x) :\equiv \lambda y.\, x + y$$

的 $f : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$。现在如果我们在某处假设了 $y : \mathbb{N}$，那么 $f(y)$ 是什么？简单地在定义 $f(x)$ 的表达式 $\lambda y.\, x + y$ 中用 $y$ 替换 $x$ 是错误的，得到 $\lambda y.\, y + y$，因为这意味着 $y$ 被**捕获**了。以前，被代换的 $y$ 是指我们的假设，但现在它指的是 λ-抽象的参数。因此，这种朴素的代换会破坏绑定结构，允许我们执行语义上不健全的计算。

但在这个例子中 $f(y)$ 是什么？注意，约束（或哑）变量 如表达式 $\lambda y.\, x + y$ 中的 $y$ 只有局部含义，可以一致地用任何其他变量替换，同时保持绑定结构。事实上，$\lambda y.\, x + y$ 被声明为判断上等于[5] $\lambda z.\, x + z$。由此可得 $f(y)$ 判断上等于 $\lambda z.\, y + z$，这就回答了我们的问题。（除了 $z$，任何与 $y$ 不同的变量都可以使用，产生相等的结果。）

当然，这对任何数学家来说都应该是熟悉的：这与以下事实是同一现象：如果 $f(x) :\equiv \int_1^2 \frac{dt}{x-t}$，那么 $f(t)$ 不是 $\int_1^2 \frac{dt}{t-t}$ 而是 $\int_1^2 \frac{ds}{t-s}$。λ-抽象绑定哑变量的方式与积分完全相同。

我们已经看到了如何定义单变量函数。定义多变量函数的一种方式是使用笛卡尔积，这将在后面介绍；具有参数 $A$ 和 $B$ 且结果在 $C$ 中的函数将被赋予类型 $f : A \times B \to C$。然而，还有另一种避免使用积类型的选择，称为**柯里化**（以数学家 Haskell Curry 命名）。

[5] 这个等式的使用通常被称为 **α-变换**。

柯里化的思想是将两个输入 $a : A$ 和 $b : B$ 的函数表示为一个接受一个输入 $a : A$ 并返回另一个函数的函数，后者再接受第二个输入 $b : B$ 并返回结果。也就是说，我们认为双变量函数属于迭代函数类型，$f : A \to (B \to C)$。我们也可以不带括号地写成 $f : A \to B \to C$，默认右结合。然后给定 $a : A$ 和 $b : B$，我们可以将 $f$ 应用到 $a$，然后将结果应用到 $b$，得到 $f(a)(b) : C$。为了避免括号泛滥，我们允许自己将 $f(a)(b)$ 写作 $f(a, b)$，尽管没有涉及积类型。当完全省略函数参数周围的括号时，我们将 $f\,a\,b$ 写作 $(f\,a)\,b$，现在默认结合性是左结合的，使 $f$ 以正确的顺序应用到其参数。

我们带显式参数定义的记法扩展到这种情况：我们可以通过给出方程

$$f(x, y) :\equiv \Phi$$

来定义具名函数 $f : A \to B \to C$，其中 $\Phi : C$ 假设 $x : A$ 和 $y : B$。使用 $\lambda$-抽象这对应于

$$f :\equiv \lambda x.\, \lambda y.\, \Phi,$$

也可以写作

$$f :\equiv x \mapsto y \mapsto \Phi.$$

我们也可以通过写多个空白来隐式地对多个变量抽象，例如 $g(-, -)$ 意味着 $\lambda x.\, \lambda y.\, g(x, y)$。对三个或更多参数的函数进行柯里化是我们刚才描述的直接推广。

## 1.3 宇宙与族

到目前为止，我们一直非形式地使用表达式 $A$ 是类型。我们将通过引入**宇宙**来使这更加精确。宇宙是一个其元素是类型的类型。如同朴素集合论，我们可能希望有一个包含自身的所有类型的宇宙 $\mathcal{U}_\infty$（即满足 $\mathcal{U}_\infty : \mathcal{U}_\infty$）。然而，如同集合论，这是不健全的，即我们可以从它推导出每个类型，包括表示命题假的空类型（见 第 1.7 节），都是有居民的。例如，使用集合作为树的表示，我们可以直接编码 Russell 悖论 [Coq92]。

为了避免悖论，我们引入宇宙的层级

$$\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \cdots$$

其中每个宇宙 $\mathcal{U}_i$ 是下一个宇宙 $\mathcal{U}_{i+1}$ 的元素。此外，我们假设我们的宇宙是 **累积的**，即第 $i$ 个宇宙的所有元素也是第 $(i+1)$ 个宇宙的元素，即如果 $A : \mathcal{U}_i$ 则也有 $A : \mathcal{U}_{i+1}$。这很方便，但有一个稍微不愉快的后果，即元素不再有唯一的类型，并且在其他方面也有些棘手，这里不需要关心；见注记。

当我们说 $A$ 是类型时，我们的意思是它居住在某个宇宙 $\mathcal{U}_i$ 中。我们通常想要避免显式提及层级 $i$，只是假设层级可以以一致的方式分配；因此我们可以写 $A : \mathcal{U}$ 省略层级。这样我们甚至可以写 $\mathcal{U} : \mathcal{U}$，它可以读作 $\mathcal{U}_i : \mathcal{U}_{i+1}$，索引已被隐式省略。以这种风格写宇宙被称为 **典型歧义**。这很方便但有点危险，因为它允许我们写出看起来有效但重现自指悖论的证明。如果对一个论证是否正确有任何疑问，检查的方法是尝试一致地为其中出现的所有宇宙分配层级。当假设某个宇宙 $\mathcal{U}$ 时，我们可以将属于 $\mathcal{U}$ 的类型称为**小类型**。

为了对在给定类型 $A$ 上变化的类型的集合建模，我们使用陪域为宇宙的函数 $B : A \to \mathcal{U}$。这些函数被称为 **类型族**（或有时称为*依赖类型*）；它们对应于集合论中使用的集合族。

类型族的一个例子是有限集族 $\mathsf{Fin} : \mathbb{N} \to \mathcal{U}$，其中 $\mathsf{Fin}(n)$ 是恰好有 $n$ 个元素的类型。（我们还不能定义族 Fin——事实上，我们甚至还没有引入它的定义域 $\mathbb{N}$——但我们很快就能够做到；见 ??。）我们可以将 $\mathsf{Fin}(n)$ 的元素记为 $0_n, 1_n, \ldots, (n-1)_n$，带有下标以强调如果 $n$ 与 $m$ 不同则 $\mathsf{Fin}(n)$ 的元素与 $\mathsf{Fin}(m)$ 的元素不同，并且它们都不同于普通自然数（我们将在第 1.9 节中介绍）。

一个更平凡（但非常重要）的类型族例子是类型 $B : \mathcal{U}$ 的**常值**类型族，它当然是常值函数 $(\lambda(x : A).\, B) : A \to \mathcal{U}$。

作为非例子，在我们的类型论版本中，没有类型族 $\lambda(i : \mathbb{N}).\, \mathcal{U}_i$。事实上，没有足够大的宇宙作为它的陪域。此外，我们甚至不将宇宙 $\mathcal{U}_i$ 的索引 $i$ 与类型论的自然数 $\mathbb{N}$ 等同（后者将在 第 1.9 节中介绍）。

## 1.4 依赖函数类型（Π-类型）

在类型论中，我们经常使用函数类型的更一般版本，称为 **Π-类型**或**依赖函数类型**。Π-类型的元素是其陪域类型可以根据函数应用到的定义域元

素而变化的函数，称为**依赖函数**。名称 Π-类型的使用是因为这种类型也可以被视为在给定类型上的笛卡尔积。

给定类型 $A : \mathcal{U}$ 和族 $B : A \to \mathcal{U}$，我们可以构造依赖函数的类型 $\prod_{(x:A)} B(x) : \mathcal{U}$。这种类型有许多替代记法，如

$$\prod_{(x:A)} B(x) \qquad \prod_{(x:A)} B(x) \qquad \prod(x : A),\ B(x).$$

如果 $B$ 是常值族，那么依赖积类型就是普通函数类型：

$$\prod_{(x:A)} B \equiv (A \to B).$$

事实上，Π-类型的所有构造都是普通函数类型上相应构造的推广。

我们可以通过显式定义引入依赖函数：为了定义 $f : \prod_{(x:A)} B(x)$，其中 $f$ 是要定义的依赖函数的名称，我们需要一个可能涉及变量 $x : A$ 的表达式 $\Phi : B(x)$，我们写

$$f(x) :\equiv \Phi \qquad \text{对于 } x : A$$

或者，我们可以使用 **λ-抽象**

$$\lambda x.\, \Phi \ :\ \prod_{x:A} B(x). \tag{1.4.1}$$

与非依赖函数一样，我们可以将依赖函数 $f : \prod_{(x:A)} B(x)$ **应用**到参数 $a : A$ 以获得元素 $f(a) : B(a)$。等式与普通函数类型相同，即我们有以下计算规则：给定 $a : A$ 我们有 $f(a) \equiv \Phi'$ 以及 $(\lambda x.\, \Phi)(a) \equiv \Phi'$，其中 $\Phi'$ 是通过将 $\Phi$ 中所有 $x$ 的出现替换为 $a$（像往常一样避免变量捕获）得到的。类似地，对于任何 $f : \prod_{(x:A)} B(x)$，我们有唯一性原则 $f \equiv (\lambda x.\, f(x))$。

作为例子，回顾 第 1.3 节中有一个类型族 $\mathsf{Fin} : \mathbb{N} \to \mathcal{U}$，其值是标准有限集，元素为 $0_n, 1_n, \ldots, (n-1)_n : \mathsf{Fin}(n)$。然后有一个依赖函数 $\mathsf{fmax} : \prod_{(n:\mathbb{N})} \mathsf{Fin}(n+1)$ 返回每个非空有限类型的最大元素，$\mathsf{fmax}(n) :\equiv n_{n+1}$。与 Fin 本身一样，我们还不能定义 fmax，但很快就能够；见 ??。

另一类重要的依赖函数类型，我们现在可以定义，是在给定宇宙上**多态**的函数 。多态函数是以类型作为其参数之一，然后作用于该类型的元素（或从它构造的其他类型）的函数。一个例子是多态恒等函数 $\mathsf{id} : \prod_{(A:\mathcal{U})} A \to$

$A$，我们将其定义为 $\mathsf{id} :\equiv \lambda(A:\mathcal{U}).\,\lambda(x:A).\,x$。（像 λ-抽象一样，Π 自动在表达式的其余部分上定界，除非有分隔符；因此 $\mathsf{id} : \prod_{(A:\mathcal{U})} A \to A$ 意味着 $\mathsf{id} : \prod_{(A:\mathcal{U})}(A \to A)$。这个惯例虽然在数学中不常见，但在类型论中很常见。）

我们有时将依赖函数的某些参数写作下标。例如，我们可以等价地通过 $\mathsf{id}_A(x) :\equiv x$ 定义多态恒等函数。此外，如果参数可以从上下文推断，我们可以完全省略它。例如，如果 $a : A$，那么写 $\mathsf{id}(a)$ 是无歧义的，因为 id 必须意味着 $\mathsf{id}_A$ 才能应用到 $a$。

另一个不太平凡的多态函数例子是交换操作，它交换（柯里化的）双参数函数的参数顺序：

$$\mathsf{swap} : \prod_{(A:\mathcal{U})} \prod_{(B:\mathcal{U})} \prod_{(C:\mathcal{U})} (A \to B \to C) \to (B \to A \to C).$$

我们可以将其定义为

$$\mathsf{swap}(A, B, C, g) :\equiv \lambda b.\,\lambda a.\,g(a)(b).$$

我们也可以等价地将类型参数写作下标：

$$\mathsf{swap}_{A,B,C}(g)(b, a) :\equiv g(a, b).$$

注意，正如我们对普通函数所做的那样，我们使用柯里化来定义具有多个参数的依赖函数（如 swap）。然而，在依赖的情况下，第二个定义域可能依赖于第一个，而陪域可能依赖于两者。即，给定 $A : \mathcal{U}$ 和类型族 $B : A \to \mathcal{U}$ 以及 $C : \prod_{(x:A)} B(x) \to \mathcal{U}$，我们可以构造具有两个参数的函数的类型 $\prod_{(x:A)} \prod_{(y:B(x))} C(x, y)$。在 $B$ 是常值且等于 $A$ 的情况下，我们可以简化记法写作 $\prod_{(x,y:A)}$；例如，swap 的类型也可以写作

$$\mathsf{swap} : \prod_{A,B,C:\mathcal{U}} (A \to B \to C) \to (B \to A \to C).$$

最后，给定 $f : \prod_{(x:A)} \prod_{(y:B(x))} C(x, y)$ 和参数 $a : A$ 和 $b : B(a)$，我们有 $f(a)(b) : C(a, b)$，和以前一样，我们将其写作 $f(a, b) : C(a, b)$。

## 1.5 积类型

给定类型 $A, B : \mathcal{U}$ 我们引入类型 $A \times B : \mathcal{U}$，我们称之为它们的**笛卡尔积**。我们还引入一个零元积类型，称为**单位类型** $\mathbf{1} : \mathcal{U}$。我们打算让 $A \times B$ 的元素是序对 $(a, b) : A \times B$，其中 $a : A$ 且 $b : B$，而 $\mathbf{1}$ 的唯一元素是某个特定对象 $\star : \mathbf{1}$。然而，与集合论不同——在集合论中我们将有序对定义为特定的集合然后将它们全部收集到笛卡尔积中——在类型论中，有序对是原始概念，正如函数一样。

*Remark* 1.5.1. 在类型论中引入新种类的类型有一个一般模式。我们已经在第 1.2 节 和第 1.4 节[6]中看到了这个模式，所以值得强调一般形式。为了指定一个类型，我们指定：

(i) 如何通过**形成规则**形成这种类型的新类型。（例如，当 $A$ 是类型且 $B$ 是类型时，我们可以形成函数类型 $A \to B$。当 $A$ 是类型且对于 $x : A$ 有 $B(x)$ 是类型时，我们可以形成依赖函数类型 $\prod_{(x:A)} B(x)$。）

(ii) 如何构造该类型的元素。这些被称为该类型的**构造子**或**引入规则**。（例如，函数类型有一个构造子，$\lambda$-抽象。回顾像 $f(x) :\equiv 2x$ 这样的直接定义可以等价地表述为 $\lambda$-抽象 $f :\equiv \lambda x.\, 2x$。）

(iii) 如何使用该类型的元素。这些被称为该类型的**消除子**或**消除规则**。（例如，函数类型有一个消除子，即函数应用。）

(iv) **计算规则**[7]，表达消除子如何作用于构造子。（例如，对于函数，计算规则说 $(\lambda x.\, \Phi)(a)$ 判断上等于将 $a$ 代入 $\Phi$ 中的 $x$。）

(v) 可选的**唯一性原则**[8]，表达映射进入或映射出该类型的唯一性。对于某些类型，唯一性原则刻画了映射到该类型的映射，通过陈述该类型的每个元素由将消除子应用于它的结果唯一确定，并且可以通过应用构造子从这些结果重构——从而表达构造子如何作用于消除子，与计算规则对偶。（例如，对于函数，唯一性原则说任何函数 $f$ 判断上等于展开的函数 $\lambda x.\, f(x)$，因此被其值唯一确定。）对于其他类型，唯一

[6] 上面对宇宙的描述是一个例外。

[7] 也称为 $\beta$-**归约**

[8] 也称为 $\eta$-**展开**

性原则说从该类型出发的每个映射（函数）由某些数据唯一确定。（一个例子是 第 1.7 节中引入的余积类型，其唯一性原则在 第 2.15 节中提到。）

当唯一性原则不作为判断相等性规则时，它通常仍然可以作为从该类型的其他规则得出的命题相等性来证明。在这种情况下，我们称之为**命题唯一性原则**。（在后面的章节中，我们偶尔也会遇到命题计算规则。）

?? 中的推理规则相应地组织和命名；例如，参见 ??，其中每种可能性都实现了。

构造序对的方式是显然的：给定 $a : A$ 和 $b : B$，我们可以形成 $(a, b) : A \times B$。类似地，有一种唯一的方式来构造 $\mathbf{1}$ 的元素，即我们有 $\star : \mathbf{1}$。我们期望 $A \times B$ 的每个元素都是序对，这是积的唯一性原则；我们不将此断言为类型论的规则，但稍后我们将证明它作为命题相等性。

现在，我们如何使用序对，即如何定义从积类型出发的函数？让我们首先考虑非依赖函数 $f : A \times B \to C$ 的定义。由于我们打算 $A \times B$ 的唯一元素是序对，我们期望能够通过规定当 $f$ 应用于序对 $(a, b)$ 时的结果来定义这样的函数。我们可以通过提供函数 $g : A \to B \to C$ 来规定这些结果。因此，我们引入一条新规则（积的消除规则），它说对于任何这样的 $g$，我们可以通过

$$f((a, b)) :\equiv g(a)(b)$$

定义函数 $f : A \times B \to C$。我们在此避免写 $g(a, b)$，以强调 $g$ 不是积上的函数。（然而，在本书后面，我们经常将 $g(a, b)$ 既用于积上的函数，也用于柯里化的双变量函数。）这个定义方程是积类型的计算规则。

注意，在集合论中，我们会通过以下事实来证明上述 $f$ 的定义：$A \times B$ 的每个元素都是有序对，因此只需在这样的序对上定义 $f$ 就足够了。相比之下，类型论颠倒了这种情况：我们假设一旦我们指定了 $A \times B$ 上的函数在序对上的值，它就是良定义的，从这个（或更精确地说，从它对依赖函数的更一般版本，见下文）我们将能够证明 $A \times B$ 的每个元素都是序对。从范畴论的角度来看，我们可以说我们将积 $A \times B$ 定义为指数 $B \to C$ 的左伴随，后者我们已经介绍了。

作为例子，我们可以推导出**投影** 函数

$$\mathsf{pr}_1 : A \times B \to A$$
$$\mathsf{pr}_2 : A \times B \to B$$

及定义方程

$$\mathsf{pr}_1((a, b)) :\equiv a$$
$$\mathsf{pr}_2((a, b)) :\equiv b.$$

与其每次想要定义函数时都调用这个函数定义原则，另一种方法是在一个普遍情况下调用它一次，然后在所有其他情况下简单地应用得到的函数。也就是说，我们可以定义一个类型为

$$\mathsf{rec}_{A\times B} : \prod_{C:\mathcal{U}} (A \to B \to C) \to A \times B \to C \tag{1.5.2}$$

的函数，及定义方程

$$\mathsf{rec}_{A\times B}(C, g, (a, b)) :\equiv g(a)(b).$$

然后与其通过定义方程直接定义像 $\mathsf{pr}_1$ 和 $\mathsf{pr}_2$ 这样的函数，我们可以定义

$$\mathsf{pr}_1 :\equiv \mathsf{rec}_{A\times B}(A, \lambda a.\, \lambda b.\, a)$$
$$\mathsf{pr}_2 :\equiv \mathsf{rec}_{A\times B}(B, \lambda a.\, \lambda b.\, b).$$

我们将函数 $\mathsf{rec}_{A\times B}$ 称为积类型的**递归子** 。名称递归子在这里有些不幸，因为没有发生递归。它来自积类型是归纳类型的一般框架的退化例子这一事实，对于像自然数这样的类型，递归子实际上将是递归的。我们也可以谈论笛卡尔积的**递归原则**，意思是我们可以如上所述通过给出它在序对上的值来定义函数 $f : A \times B \to C$ 的事实。

我们留给读者作为简单练习来证明递归子可以从投影推导出来，反之亦然。

我们也有单位类型的递归子：

$$\mathsf{rec}_{\mathbf{1}} : \prod_{C:\mathcal{U}} C \to \mathbf{1} \to C$$

及定义方程

$$\mathsf{rec}_{\mathbf{1}}(C, c, \star) :\equiv c.$$

虽然我们包含它以保持类型定义的模式，但 $\mathbf{1}$ 的递归子是完全无用的，因为我们可以通过简单地忽略类型 $\mathbf{1}$ 的参数来直接定义这样的函数。

为了能够在积类型上定义依赖函数，我们必须推广递归子。给定 $C : A \times B \to \mathcal{U}$，我们可以通过提供函数

$$g : \prod_{(x:A)} \prod_{(y:B)} C((x, y))$$

定义函数 $f : \prod_{(x:A\times B)} C(x)$，及定义方程

$$f((x, y)) :\equiv g(x)(y).$$

例如，通过这种方式，我们可以证明命题唯一性原则，它说 $A \times B$ 的每个元素都等于一个序对。具体地，我们可以构造一个函数

$$\mathsf{uniq}_{A\times B} : \prod_{x:A\times B} ((\mathsf{pr}_1(x), \mathsf{pr}_2(x)) =_{A\times B} x).$$

这里我们使用了恒等类型，我们将在下面的 第 1.12 节中介绍。然而，我们现在需要知道的只是对于任何 $x : A$ 有一个自反性元素 $\mathsf{refl}_x : x =_A x$。给定这个，我们可以定义

$$\mathsf{uniq}_{A\times B}((a, b)) :\equiv \mathsf{refl}_{(a,b)}.$$

这个构造有效，因为在 $x :\equiv (a, b)$ 的情况下，我们可以使用投影的定义方程计算

$$(\mathsf{pr}_1((a, b)), \mathsf{pr}_2((a, b))) \equiv (a, b)$$

因此，

$$\mathsf{refl}_{(a,b)} : (\mathsf{pr}_1((a, b)), \mathsf{pr}_2((a, b))) = (a, b)$$

类型良好，因为等式两边判断上相等。

更一般地，能够以这种方式定义依赖函数意味着要证明积的所有元素的性质，只需为其规范元素——有序对——证明它就足够了。当我们谈到

像自然数这样的归纳类型时，类似的性质将是能够通过归纳写证明的能力。因此，如果我们像上面那样在普遍情况下应用这个原则一次，我们称得到的函数为积类型的**归纳**：给定 $A, B : \mathcal{U}$ 我们有

$$\mathsf{ind}_{A\times B} : \prod_{C:A\times B\to\mathcal{U}} \Big(\prod_{(x:A)}\prod_{(y:B)} C((x,y))\Big) \to \prod_{x:A\times B} C(x)$$

及定义方程

$$\mathsf{ind}_{A\times B}(C, g, (a, b)) :\equiv g(a)(b).$$

类似地，我们可以说在序对上定义的依赖函数是从笛卡尔积的**归纳原则**获得的 。容易看出，递归子只是在族 $C$ 是常值的情况下归纳的特例。因为归纳描述了如何使用积类型的元素，归纳也被称为（**依赖**）**消除子**，而递归则被称为**非依赖消除子**。

单位类型的归纳比递归子更有用：

$$\mathsf{ind}_{\mathbf{1}} : \prod_{C:\mathbf{1}\to\mathcal{U}} C(\star) \to \prod_{x:\mathbf{1}} C(x)$$

及定义方程

$$\mathsf{ind}_{\mathbf{1}}(C, c, \star) :\equiv c.$$

归纳使我们能够证明 $\mathbf{1}$ 的命题唯一性原则，它断言其唯一的居民是 $\star$。即，我们可以构造

$$\mathsf{uniq}_{\mathbf{1}} : \prod_{x:\mathbf{1}} x = \star$$

通过使用定义方程

$$\mathsf{uniq}_{\mathbf{1}}(\star) :\equiv \mathsf{refl}_{\star}$$

或者等价地通过使用归纳：

$$\mathsf{uniq}_{\mathbf{1}} :\equiv \mathsf{ind}_{\mathbf{1}}(\lambda x.\, x = \star, \mathsf{refl}_{\star}).$$

## 1.6 依赖对类型（Σ-类型）

正如我们将函数类型（第 1.2 节）推广到依赖函数类型（第 1.4 节），将 第 1.5 节的积类型推广为允许序对的第二分量的类型依赖于第一分量的选择通常也是有用的。这被称为**依赖对类型**或 **Σ-类型**，因为在集合论中它对应于给定类型上的索引和（在余积或不交并的意义上）。

给定类型 $A : \mathcal{U}$ 和族 $B : A \to \mathcal{U}$，依赖对类型写作 $\sum_{(x:A)} B(x) : \mathcal{U}$。可选的记法有

$$\sum_{(x:A)} B(x) \qquad \sum_{(x:A)} B(x) \qquad \sum(x : A),\ B(x).$$

与 λ-抽象和 Π 等其他绑定构造一样，Σ 自动将作用域延伸到表达式的其余部分（除非有分隔符），因此例如 $\sum_{(x:A)} B(x) \to C$ 意味着 $\sum_{(x:A)} (B(x) \to C)$。

构造依赖对类型元素的方式是配对：给定 $a : A$ 和 $b : B(a)$，我们有 $(a, b) : \sum_{(x:A)} B(x)$。如果 $B$ 是常值的，那么依赖对类型就是普通的笛卡尔积类型：

$$\left(\sum_{x:A} B\right) \equiv (A \times B).$$

Σ-类型上的所有构造都作为积类型构造的直接推广出现，依赖函数常常替换非依赖函数。

例如，递归原则 说明要定义从 Σ-类型出发的非依赖函数 $f : (\sum_{(x:A)} B(x)) \to C$，我们提供一个函数 $g : \prod_{(x:A)} B(x) \to C$，然后我们可以通过定义方程定义 $f$

$$f((a, b)) :\equiv g(a)(b).$$

例如，我们可以从 Σ-类型推导第一投影：

$$\mathsf{pr}_1 : \left(\sum_{x:A} B(x)\right) \to A$$

通过定义方程

$$\mathsf{pr}_1((a, b)) :\equiv a.$$

然而，由于序对

$$(a, b) : \sum_{x:A} B(x)$$

的第二分量的类型是 $B(a)$，第二投影必须是依赖函数，其类型涉及第一投影函数：

$$\mathsf{pr}_2 : \prod_{p:\sum_{(x:A)} B(x)} B(\mathsf{pr}_1(p)).$$

因此我们需要 Σ-类型的归纳原则（依赖消除子）。这说明要从 Σ-类型构造到族 $C : (\sum_{(x:A)} B(x)) \to \mathcal{U}$ 的依赖函数，我们需要一个函数

$$g : \prod_{(a:A)} \prod_{(b:B(a))} C((a, b)).$$

然后我们可以推导一个函数

$$f : \prod_{p:\sum_{(x:A)} B(x)} C(p)$$

及定义方程

$$f((a, b)) :\equiv g(a)(b).$$

应用这个并令 $C(p) :\equiv B(\mathsf{pr}_1(p))$，我们可以定义

$$\mathsf{pr}_2 : \prod_{p:\sum_{(x:A)} B(x)} B(\mathsf{pr}_1(p))$$

及明显的方程

$$\mathsf{pr}_2((a, b)) :\equiv b.$$

为了说服自己这是正确的，我们注意到使用 $\mathsf{pr}_1$ 的定义方程有 $B(\mathsf{pr}_1((a, b))) \equiv B(a)$，而确实 $b : B(a)$。

我们可以将递归和归纳原则打包成 Σ 的递归子：

$$\mathsf{rec}_{\sum_{(x:A)} B(x)} : \prod_{(C:\mathcal{U})} \left(\prod_{(x:A)} B(x) \to C\right) \to \left(\sum_{(x:A)} B(x)\right) \to C$$

及定义方程

$$\mathsf{rec}_{\sum_{(x:A)} B(x)}(C, g, (a, b)) :\equiv g(a)(b)$$

以及相应的归纳算子：

$$\mathsf{ind}_{\sum_{(x:A)} B(x)} : \prod_{(C:(\sum_{(x:A)} B(x))\to\mathcal{U})} \left(\prod_{(a:A)}\prod_{(b:B(a))} C((a,b))\right) \to \prod_{(p:\sum_{(x:A)} B(x))} C(p)$$

及定义方程

$$\mathsf{ind}_{\sum_{(x:A)} B(x)}(C, g, (a,b)) :\equiv g(a)(b).$$

如前所述，递归子是当族 $C$ 是常值时归纳的特例。

作为进一步的例子，考虑以下原则，其中 $A$ 和 $B$ 是类型且 $R : A \to B \to \mathcal{U}$：

$$\mathsf{ac} : \left(\prod_{(x:A)}\sum_{(y:B)} R(x,y)\right) \to \left(\sum_{(f:A\to B)}\prod_{(x:A)} R(x, f(x))\right).$$

我们可以将 $R$ 视为 $A$ 和 $B$ 之间的证明相关关系，其中 $R(a,b)$ 是 $a : A$ 和 $b : B$ 相关的见证类型。那么 $\mathsf{ac}$ 直观地说如果我们有一个依赖函数 $g$ 为每个 $a : A$ 分配一个依赖对 $(b, r)$，其中 $b : B$ 且 $r : R(a,b)$，那么我们有一个函数 $f : A \to B$ 和一个依赖函数为每个 $a : A$ 分配一个 $R(a, f(a))$ 的见证。我们的直觉告诉我们可以直接将 $g$ 的值分解成它们的分量。确实，使用我们刚刚定义的投影，我们可以定义：

$$\mathsf{ac}(g) :\equiv \left(\lambda x.\, \mathsf{pr}_1(g(x)),\, \lambda x.\, \mathsf{pr}_2(g(x))\right).$$

为了验证这是类型良好的，注意如果 $g : \prod_{(x:A)} \sum_{(y:B)} R(x,y)$，我们有

$$\lambda x.\, \mathsf{pr}_1(g(x)) : A \to B,$$
$$\lambda x.\, \mathsf{pr}_2(g(x)) : \prod_{(x:A)} R(x, \mathsf{pr}_1(g(x))).$$

此外，类型 $\prod_{(x:A)} R(x, \mathsf{pr}_1(g(x)))$ 是将 $\mathsf{ac}$ 的陪域中被求和的类型族 $\lambda f. \prod_{(x:A)} R(x, f(x))$ 应用于函数 $\lambda x.\, \mathsf{pr}_1(g(x))$ 的结果：

$$\prod_{(x:A)} R(x, \mathsf{pr}_1(g(x))) \equiv \left(\lambda f. \prod_{(x:A)} R(x, f(x))\right)\left(\lambda x.\, \mathsf{pr}_1(g(x))\right).$$

因此，我们有

$$\left(\lambda x.\, \mathsf{pr}_1(g(x)),\, \lambda x.\, \mathsf{pr}_2(g(x))\right) : \sum_{(f:A\to B)}\prod_{(x:A)} R(x, f(x))$$

如所需。

如果我们将 Π 读作对所有而将 Σ 读作存在，那么函数 ac 的类型表达的是：*如果对所有* $x : A$ *存在* $y : B$ *使得* $R(x, y)$，*那么存在函数* $f : A \to B$ *使得对所有* $x : A$ *有* $R(x, f(x))$。由于这听起来像是选择公理的一个版本，函数 ac 传统上被称为**类型论选择公理**，正如我们刚刚展示的，它可以直接从类型论的规则证明，而不必作为公理接受。然而，注意实际上没有涉及选择，因为选择已经在前提中给出：我们所要做的只是将它拆分成两个函数：一个代表选择，另一个代表其正确性。在 第 3.8 节中我们将给出另一个更接近通常形式的选择公理表述。

依赖对类型常用于定义数学结构的类型，这些结构通常由若干相互依赖的数据组成。举一个简单的例子，假设我们想定义**原群**为一个类型 $A$ 连同一个二元运算 $m : A \to A \to A$。短语连同（以及同义的配备）的精确含义是一个原群是由类型 $A : \mathcal{U}$ 和运算 $m : A \to A \to A$ 组成的*序对* $(A, m)$。由于这个序对的第二分量 $m$ 的类型 $A \to A \to A$ 依赖于它的第一分量 $A$，这样的序对属于依赖对类型。因此，定义原群是一个类型 $A$ 连同一个二元运算 $m : A \to A \to A$ 应该理解为定义*原群的类型*为

$$\mathsf{Magma} :\equiv \sum_{A:\mathcal{U}} (A \to A \to A).$$

给定一个原群，我们用第一投影 $\mathsf{pr}_1$ 提取它的底层类型（它的载体），用第二投影 $\mathsf{pr}_2$ 提取它的运算。当然，由多于两片数据构建的结构需要迭代的对类型，它们可能只是部分依赖的；例如带点原群（配备基点 $e : A$ 的原群 $(A, m)$）的类型是

$$\mathsf{PointedMagma} :\equiv \sum_{A:\mathcal{U}} (A \to A \to A) \times A.$$

我们通常还想对这样的结构施加公理，例如使带点原群成为幺半群或群。这也可以用 Σ-类型完成；见 第 1.11 节。

在本书的其余部分，我们有时会明确说明这种定义，但最终我们相信读者能将它们从英语翻译成 Σ-类型。我们也通常遵循常见的数学实践，对这种结构及其载体使用相同的字母（这相当于在记法中隐含适当的投影函数）：即，我们会说原群 $A$ 及其运算 $m : A \to A \to A$。

注意 PointedMagma 的规范元素形如 $(A,(m,e))$，其中 $A:\mathcal{U}$，$m:A\to A\to A$，且 $e:A$。由于这种迭代 $\Sigma$-类型出现的频率，我们使用有序三元组、四元组等的通常记法来表示向右结合的嵌套对。即，我们有 $(x,y,z):\equiv(x,(y,z))$ 和 $(x,y,z,w):\equiv(x,(y,(z,w)))$，等等。

## 1.7 余积类型

给定 $A,B:\mathcal{U}$，我们引入它们的**余积**类型 $A+B:\mathcal{U}$。这对应于集合论中的不交并，我们也可以用这个名称称呼它。在类型论中，与函数和积的情况一样，余积必须是基本构造，因为没有预先给定的类型的并的概念。我们还引入一个零元版本：**空类型** $\mathbf{0}:\mathcal{U}$。

有两种方式构造 $A+B$ 的元素，要么作为 $\mathsf{inl}(a):A+B$（对于 $a:A$），要么作为 $\mathsf{inr}(b):A+B$（对于 $b:B$）。（名称 inl 和 inr 是左注入和右注入的缩写。）没有方式构造空类型的元素。

要构造一个非依赖函数 $f:A+B\to C$，我们需要函数 $g_0:A\to C$ 和 $g_1:B\to C$。然后 $f$ 通过定义方程定义

$$f(\mathsf{inl}(a)):\equiv g_0(a),$$
$$f(\mathsf{inr}(b)):\equiv g_1(b).$$

也就是说，函数 $f$ 是通过**分情况分析**定义的。如前所述，我们可以推导递归子：

$$\mathsf{rec}_{A+B}:\prod_{(C:\mathcal{U})}(A\to C)\to(B\to C)\to A+B\to C$$

及定义方程

$$\mathsf{rec}_{A+B}(C,g_0,g_1,\mathsf{inl}(a)):\equiv g_0(a),$$
$$\mathsf{rec}_{A+B}(C,g_0,g_1,\mathsf{inr}(b)):\equiv g_1(b).$$

我们总是可以构造函数 $f:\mathbf{0}\to C$ 而不必给出任何定义方程，因为没有 $\mathbf{0}$ 的元素需要在其上定义 $f$。因此，$\mathbf{0}$ 的递归子是

$$\mathsf{rec}_{\mathbf{0}}:\textstyle\prod_{(C:\mathcal{U})}\mathbf{0}\to C,$$

它构造从空类型到任何其他类型的规范函数。在逻辑上，它对应于原则 *ex falso quodlibet*（从假可推出任何事）。

要从余积构造依赖函数 $f : \prod_{(x:A+B)} C(x)$，我们假定给定族 $C : (A + B) \to \mathcal{U}$，并要求

$$g_0 : \prod_{a:A} C(\mathsf{inl}(a)),$$
$$g_1 : \prod_{b:B} C(\mathsf{inr}(b)).$$

这产生 $f$ 及定义方程：

$$f(\mathsf{inl}(a)) :\equiv g_0(a),$$
$$f(\mathsf{inr}(b)) :\equiv g_1(b).$$

我们将这个模式打包成余积的归纳原则：

$$\mathsf{ind}_{A+B} : \prod_{(C:(A+B)\to\mathcal{U})} \left(\prod_{(a:A)} C(\mathsf{inl}(a))\right) \to \left(\prod_{(b:B)} C(\mathsf{inr}(b))\right) \to \prod_{(x:A+B)} C(x).$$

如前所述，当族 $C$ 是常值时递归子出现。

空类型的归纳原则

$$\mathsf{ind}_{\mathbf{0}} : \prod_{(C:\mathbf{0}\to\mathcal{U})} \prod_{(z:\mathbf{0})} C(z)$$

给我们一种定义从空类型出发的平凡依赖函数的方式。

## 1.8 布尔类型

布尔类型 $\mathbf{2} : \mathcal{U}$ 旨在恰好有两个元素 $0_{\mathbf{2}}, 1_{\mathbf{2}} : \mathbf{2}$。很明显我们可以用余积 和单位类型将这个类型构造为 $\mathbf{1} + \mathbf{1}$。然而，由于它被频繁使用，我们在这里给出明确的规则。事实上，我们将观察到我们也可以反过来从 $\Sigma$-类型和 $\mathbf{2}$ 推导二元余积。

要推导函数 $f : \mathbf{2} \to C$，我们需要 $c_0, c_1 : C$ 并添加定义方程

$$f(0_{\mathbf{2}}) :\equiv c_0,$$
$$f(1_{\mathbf{2}}) :\equiv c_1.$$

递归子对应于函数式编程中的 if-then-else 构造：

$$\mathsf{rec}_{\mathbf{2}} : \prod_{C:\mathcal{U}} C \to C \to \mathbf{2} \to C$$

及定义方程

$$\mathsf{rec}_{\mathbf{2}}(C, c_0, c_1, 0_{\mathbf{2}}) :\equiv c_0,$$
$$\mathsf{rec}_{\mathbf{2}}(C, c_0, c_1, 1_{\mathbf{2}}) :\equiv c_1.$$

给定 $C : \mathbf{2} \to \mathcal{U}$，要推导依赖函数 $f : \prod_{(x:\mathbf{2})} C(x)$，我们需要 $c_0 : C(0_{\mathbf{2}})$ 和 $c_1 : C(1_{\mathbf{2}})$，在这种情况下我们可以给出定义方程

$$f(0_{\mathbf{2}}) :\equiv c_0,$$
$$f(1_{\mathbf{2}}) :\equiv c_1.$$

我们将此打包成归纳原则

$$\mathsf{ind}_{\mathbf{2}} : \prod_{(C:\mathbf{2}\to\mathcal{U})} C(0_{\mathbf{2}}) \to C(1_{\mathbf{2}}) \to \textstyle\prod_{(x:\mathbf{2})} C(x)$$

及定义方程

$$\mathsf{ind}_{\mathbf{2}}(C, c_0, c_1, 0_{\mathbf{2}}) :\equiv c_0,$$
$$\mathsf{ind}_{\mathbf{2}}(C, c_0, c_1, 1_{\mathbf{2}}) :\equiv c_1.$$

作为例子，使用归纳原则我们可以推出，正如预期的，$\mathbf{2}$ 的每个元素要么是 $1_{\mathbf{2}}$ 要么是 $0_{\mathbf{2}}$。如前所述，为了陈述这一点，我们使用尚未引入的等式类型，但我们只需要知道每个东西都等于它自己：$\mathsf{refl}_x : x = x$。因此，我们构造一个

$$\prod_{x:\mathbf{2}} (x = 0_{\mathbf{2}}) + (x = 1_{\mathbf{2}}) \tag{1.8.1}$$

的元素，即一个函数为每个 $x : \mathbf{2}$ 分配一个等式 $x = 0_{\mathbf{2}}$ 或等式 $x = 1_{\mathbf{2}}$。我们使用 $\mathbf{2}$ 的归纳原则定义这个元素，令 $C(x) :\equiv (x = 0_{\mathbf{2}}) + (x = 1_{\mathbf{2}})$；两个输入是 $\mathsf{inl}(\mathsf{refl}_{0_{\mathbf{2}}}) : C(0_{\mathbf{2}})$ 和 $\mathsf{inr}(\mathsf{refl}_{1_{\mathbf{2}}}) : C(1_{\mathbf{2}})$。换言之，我们的 (1.8.1) 的元素是

$$\mathsf{ind}_{\mathbf{2}}\big(\lambda x.\,(x = 0_{\mathbf{2}}) + (x = 1_{\mathbf{2}}),\, \mathsf{inl}(\mathsf{refl}_{0_{\mathbf{2}}}),\, \mathsf{inr}(\mathsf{refl}_{1_{\mathbf{2}}})\big).$$

我们已经说过 $\Sigma$-类型可以被视为类似于索引不交并，而余积是二元不交并。很自然地期望二元不交并 $A + B$ 可以被构造为双元素类型 $\mathbf{2}$ 上的索引不交并。为此我们需要一个类型族 $P : \mathbf{2} \to \mathcal{U}$ 使得 $P(0_{\mathbf{2}}) \equiv A$ 且 $P(1_{\mathbf{2}}) \equiv B$。确实，我们可以精确地通过 $\mathbf{2}$ 的递归原则获得这样的族。（能够通过归纳和递归定义*类型族*，使用宇宙 $\mathcal{U}$ 本身是一个类型这一事实，是类型论的一个微妙而重要的方面。）因此，我们本可以定义

$$A + B :\equiv \sum_{x:\mathbf{2}} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)$$

及

$$\begin{aligned}\mathsf{inl}(a) &:\equiv (0_{\mathbf{2}}, a),\\ \mathsf{inr}(b) &:\equiv (1_{\mathbf{2}}, b).\end{aligned}$$

我们留给读者作为练习从这个定义推导余积类型的归纳原则。（另见 ?? 和??。）

我们可以将同样的想法应用于积和 $\Pi$-类型：我们本可以定义

$$A \times B :\equiv \prod_{x:\mathbf{2}} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x).$$

序对可以用 $\mathbf{2}$ 的归纳构造：

$$(a, b) :\equiv \mathsf{ind}_{\mathbf{2}}(\mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B), a, b)$$

而投影是直接的应用

$$\begin{aligned}\mathsf{pr}_1(p) &:\equiv p(0_{\mathbf{2}}),\\ \mathsf{pr}_2(p) &:\equiv p(1_{\mathbf{2}}).\end{aligned}$$

以这种方式定义的二元积的归纳原则的推导更加复杂，并且需要函数外延性，我们将在 第 2.9 节中引入。此外，我们不能得到相同的判断等式；见 ??。这是将一种类型编码为另一种类型时的一个反复出现的问题；我们将在 ?? 中回到这个问题。

我们有时会分别将 $\mathbf{2}$ 的元素 $0_{\mathbf{2}}$ 和 $1_{\mathbf{2}}$ 称为假和真。然而，注意与经典数学不同，我们不使用 $\mathbf{2}$ 的元素作为真值 或作为命题。（相反我们将命题与类型等同；见 第 1.11 节。）特别是，类型 $A \to \mathbf{2}$ 通常不是 $A$ 的幂集；它只代表 $A$ 的可判定子集（见 第 3 章）。

## 1.9 自然数

到目前为止，我们有通过抽象操作构造新类型的规则，但要做具体的数学，我们还需要一些具体的类型，如数的类型。最基本的这种类型是自然数类型 $\mathbb{N} : \mathcal{U}$；一旦我们有了这个，我们可以构造整数、有理数、实数等等（见 第 11 章）。

$\mathbb{N}$ 的元素使用 $0 : \mathbb{N}$ 和后继运算 $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$ 构造。当表示自然数时，我们采用通常的十进制记法 $1 :\equiv \mathsf{succ}(0)$，$2 :\equiv \mathsf{succ}(1)$，$3 :\equiv \mathsf{succ}(2)$，…。

自然数的基本性质是我们可以通过递归定义函数并通过归纳进行证明——这里递归和归纳这两个词有了更熟悉的含义。要通过递归从自然数构造非依赖函数 $f : \mathbb{N} \to C$，只需提供一个起点 $c_0 : C$ 和一个下一步函数 $c_s : \mathbb{N} \to C \to C$。这产生 $f$ 及定义方程

$$f(0) :\equiv c_0,$$
$$f(\mathsf{succ}(n)) :\equiv c_s(n, f(n)).$$

我们说 $f$ 是通过**原始递归**定义的。

作为例子，我们来看如何定义一个将其参数加倍的自然数函数。在这种情况下我们有 $C :\equiv \mathbb{N}$。我们首先需要提供 $\mathsf{double}(0)$ 的值，这很容易：我们令 $c_0 :\equiv 0$。接下来，要计算自然数 $n$ 的 $\mathsf{double}(\mathsf{succ}(n))$ 的值，我们首先计算 $\mathsf{double}(n)$ 的值，然后执行两次后继运算。这由递推式 $c_s(n, y) :\equiv \mathsf{succ}(\mathsf{succ}(y))$ 捕获。注意 $c_s$ 的第二个参数 $y$ 代表*递归调用* $\mathsf{double}(n)$ 的结果。

因此，以这种方式通过原始递归定义 $\mathsf{double} : \mathbb{N} \to \mathbb{N}$，我们得到定义方程：

$$\begin{aligned} \mathsf{double}(0) &:\equiv 0 \\ \mathsf{double}(\mathsf{succ}(n)) &:\equiv \mathsf{succ}(\mathsf{succ}(\mathsf{double}(n))). \end{aligned}$$

这确实有正确的计算行为：例如，我们有

$$\begin{aligned} \mathsf{double}(2) &\equiv \mathsf{double}(\mathsf{succ}(\mathsf{succ}(0))) \\ &\equiv c_s(\mathsf{succ}(0), \mathsf{double}(\mathsf{succ}(0))) \\ &\equiv \mathsf{succ}(\mathsf{succ}(\mathsf{double}(\mathsf{succ}(0)))) \\ &\equiv \mathsf{succ}(\mathsf{succ}(c_s(0, \mathsf{double}(0)))) \\ &\equiv \mathsf{succ}(\mathsf{succ}(\mathsf{succ}(\mathsf{succ}(\mathsf{double}(0))))) \\ &\equiv \mathsf{succ}(\mathsf{succ}(\mathsf{succ}(\mathsf{succ}(c_0)))) \\ &\equiv \mathsf{succ}(\mathsf{succ}(\mathsf{succ}(\mathsf{succ}(0)))) \\ &\equiv 4. \end{aligned}$$

我们也可以通过原始递归定义多变量函数，方法是柯里化并允许 $C$ 是函数类型。例如，我们用 $C :\equiv \mathbb{N} \to \mathbb{N}$ 和以下起点和下一步数据定义加法 $\mathsf{add} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$：

$$\begin{aligned} c_0 &: \mathbb{N} \to \mathbb{N} \\ c_0(n) &:\equiv n \\ c_s &: \mathbb{N} \to (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N}) \\ c_s(m, g)(n) &:\equiv \mathsf{succ}(g(n)). \end{aligned}$$

因此我们得到 $\mathsf{add} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ 满足定义等式

$$\begin{aligned} \mathsf{add}(0, n) &\equiv n \\ \mathsf{add}(\mathsf{succ}(m), n) &\equiv \mathsf{succ}(\mathsf{add}(m, n)). \end{aligned}$$

照例，我们将 $\mathsf{add}(m, n)$ 写成 $m + n$。请读者验证 $2 + 2 \equiv 4$。

如在以前的情况中，我们可以将原始递归原则打包成递归子：

$$\mathsf{rec}_{\mathbb{N}} : \prod_{(C:\mathcal{U})} C \to (\mathbb{N} \to C \to C) \to \mathbb{N} \to C$$

及定义方程

$$\begin{aligned} \mathsf{rec}_{\mathbb{N}}(C, c_0, c_s, 0) &:\equiv c_0, \\ \mathsf{rec}_{\mathbb{N}}(C, c_0, c_s, \mathsf{succ}(n)) &:\equiv c_s(n, \mathsf{rec}_{\mathbb{N}}(C, c_0, c_s, n)). \end{aligned}$$

使用 $\mathsf{rec}_{\mathbb{N}}$ 我们可以将 double 和 add 表示如下：

$$\mathsf{double} :\equiv \mathsf{rec}_{\mathbb{N}}\big(\mathbb{N},\, 0,\, \lambda n.\, \lambda y.\, \mathsf{succ}(\mathsf{succ}(y))\big) \tag{1.9.1}$$

$$\mathsf{add} :\equiv \mathsf{rec}_{\mathbb{N}}\big(\mathbb{N} \to \mathbb{N},\, \lambda n.\, n,\, \lambda m.\, \lambda g.\, \lambda n.\, \mathsf{succ}(g(n))\big). \tag{1.9.2}$$

当然，所有仅使用原始递归原则可定义的函数都将是可计算的。（然而，高阶函数类型的存在——即以其他函数作为参数的函数——确实意味着我们可以定义比通常的原始递归函数更多的函数；见例如 ??。）这在构造性数学中是合适的；在 第 3.4 节 和第 3.8 节 中我们将看到如何扩充类型论使我们可以定义更一般的数学函数。

我们现在遵循与其他类型相同的方法，将原始递归推广到依赖函数以获得*归纳原则*。因此，假定给定族 $C : \mathbb{N} \to \mathcal{U}$，元素 $c_0 : C(0)$，和函数 $c_s : \prod_{(n:\mathbb{N})} C(n) \to C(\mathsf{succ}(n))$；那么我们可以构造 $f : \prod_{(n:\mathbb{N})} C(n)$ 及定义方程：

$$\begin{aligned} f(0) &:\equiv c_0, \\ f(\mathsf{succ}(n)) &:\equiv c_s(n, f(n)). \end{aligned}$$

我们也可以将此打包成单个函数

$$\mathsf{ind}_{\mathbb{N}} : \prod_{(C:\mathbb{N}\to\mathcal{U})} C(0) \to \Big(\prod_{(n:\mathbb{N})} C(n) \to C(\mathsf{succ}(n))\Big) \to \prod_{(n:\mathbb{N})} C(n)$$

及定义方程

$$\begin{aligned} \mathsf{ind}_{\mathbb{N}}(C, c_0, c_s, 0) &:\equiv c_0, \\ \mathsf{ind}_{\mathbb{N}}(C, c_0, c_s, \mathsf{succ}(n)) &:\equiv c_s(n, \mathsf{ind}_{\mathbb{N}}(C, c_0, c_s, n)). \end{aligned}$$

这里我们终于看到了与经典的归纳证明概念的联系。回想在类型论中我们用类型表示命题，通过居于相应类型来证明命题。特别地，自然数的性质由类型族 $P : \mathbb{N} \to \mathcal{U}$ 表示。从这个观点看，上述归纳原则说的是如果我们能证明 $P(0)$，且对任何 $n$ 我们能假设 $P(n)$ 证明 $P(\mathsf{succ}(n))$，那么对所有 $n$ 我们有 $P(n)$。这当然正是对自然数归纳证明的通常原则。

作为例子，考虑我们如何表示 $+$ 是结合的显式证明。（我们实际上不会以这种风格写出证明，但它作为理解归纳如何在类型论中形式表示的有用例子。）要推导

$$\mathsf{assoc} : \prod_{i,j,k:\mathbb{N}} i + (j + k) = (i + j) + k,$$

只需提供

$$\mathsf{assoc}_0 : \prod_{j,k:\mathbb{N}} 0 + (j + k) = (0 + j) + k$$

和

$$\mathsf{assoc}_s : \prod_{i:\mathbb{N}} \left( \prod_{j,k:\mathbb{N}} i + (j + k) = (i + j) + k \right) \to \prod_{j,k:\mathbb{N}} \mathsf{succ}(i) + (j + k) = (\mathsf{succ}(i) + j) + k.$$

要推导 $\mathsf{assoc}_0$，回想 $0 + n \equiv n$，因此 $0 + (j + k) \equiv j + k \equiv (0 + j) + k$。因此我们可以直接令

$$\mathsf{assoc}_0(j, k) :\equiv \mathsf{refl}_{j+k}.$$

对于 $\mathsf{assoc}_s$，回想 $+$ 的定义给出 $\mathsf{succ}(m) + n \equiv \mathsf{succ}(m + n)$，因此

$$\mathsf{succ}(i) + (j + k) \equiv \mathsf{succ}(i + (j + k)) \qquad \text{且}$$
$$(\mathsf{succ}(i) + j) + k \equiv \mathsf{succ}((i + j) + k).$$

因此，$\mathsf{assoc}_s$ 的输出类型等价于 $\mathsf{succ}(i + (j + k)) = \mathsf{succ}((i + j) + k)$。但它的输入（归纳假设）产出 $i + (j + k) = (i + j) + k$，所以只需援引如果两个自然数相等，那么它们的后继也相等这一事实。（我们将在 引理 2.2.1

中使用恒等类型的归纳原则证明这个显然的事实。）我们将后一个事实称为 $\mathsf{ap}_{\mathsf{succ}} : (m =_{\mathbb{N}} n) \to (\mathsf{succ}(m) =_{\mathbb{N}} \mathsf{succ}(n))$，所以我们可以定义

$$\mathsf{assoc}_s(i, h, j, k) :\equiv \mathsf{ap}_{\mathsf{succ}}(h(j, k)).$$

将这些与 $\mathsf{ind}_{\mathbb{N}}$ 放在一起，我们得到结合性的证明。

## 1.10 模式匹配与递归

自然数比到目前为止考虑的类型引入了额外的微妙之处。在余积的情况下，例如，我们可以用递归子定义函数 $f : A + B \to C$：

$$f :\equiv \mathsf{rec}_{A+B}(C, g_0, g_1)$$

或者通过给出定义方程：

$$f(\mathsf{inl}(a)) :\equiv g_0(a)$$
$$f(\mathsf{inr}(b)) :\equiv g_1(b).$$

从前一个表达式到后一个表达式，我们只需使用递归子的计算规则。反过来，给定任何定义方程

$$f(\mathsf{inl}(a)) :\equiv \Phi_0$$
$$f(\mathsf{inr}(b)) :\equiv \Phi_1$$

其中 $\Phi_0$ 和 $\Phi_1$ 是可能分别涉及变量 $a$ 和 $b$ 的表达式，我们可以用 $\lambda$-抽象将这些方程等价地用递归子表达：

$$f :\equiv \mathsf{rec}_{A+B}(C, \lambda a.\, \Phi_0, \lambda b.\, \Phi_1).$$

然而在自然数的情况下，像 double 这样的函数的定义方程：

$$\mathsf{double}(0) :\equiv 0 \tag{1.10.1}$$
$$\mathsf{double}(\mathsf{succ}(n)) :\equiv \mathsf{succ}(\mathsf{succ}(\mathsf{double}(n))) \tag{1.10.2}$$

在右侧涉及函数 double 本身。然而，我们仍然希望能够给出这些方程而不是 (1.9.1) 作为 double 的定义，因为它们更加方便和可读。解决方案是将 (1.10.2) 右侧的表达式 $\mathsf{double}(n)$ 解读为代表递归调用的结果，在形式 $\mathsf{double} :\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N}, c_0, c_s)$ 的定义中它将是 $c_s$ 的第二个参数。

更一般地，如果我们有函数 $f : \mathbb{N} \to C$ 的定义如

$$f(0) :\equiv \Phi_0$$
$$f(\mathsf{succ}(n)) :\equiv \Phi_s$$

其中 $\Phi_0$ 是类型 $C$ 的表达式，而 $\Phi_s$ 是类型 $C$ 的表达式，可能涉及变量 $n$ 以及符号 $f(n)$，我们可以将其翻译成定义

$$f :\equiv \mathsf{rec}_{\mathbb{N}}(C,\ \Phi_0,\ \lambda n.\, \lambda r.\, \Phi'_s)$$

其中 $\Phi'_s$ 是通过将 $f(n)$ 的所有出现替换为新变量 $r$ 从 $\Phi_s$ 获得的。

这种通过递归（或更一般地，通过归纳）定义函数（或依赖函数）的风格非常方便，我们经常采用它。它被称为通过**模式匹配**定义。当然，这与计算机程序员如何定义一个函数体实际包含对自身递归调用的递归函数非常相似。然而，与程序员不同，我们对可以进行什么样的递归调用有限制：为了使这样的定义可用递归原则重新表达，被定义的函数 $f$ 只能作为复合符号 $f(n)$ 的一部分出现在 $f(\mathsf{succ}(n))$ 的函数体中。否则，我们可以写出无意义的函数如

$$f(0) :\equiv 0$$
$$f(\mathsf{succ}(n)) :\equiv f(\mathsf{succ}(\mathsf{succ}(n))).$$

如果程序员写这样的函数，它会在任何正输入上无限地调用自己，进入无限循环并且永远不返回值。然而在数学中，要配得上这个名称，一个函数必须总是为每个输入值关联一个唯一的输出值，所以这是不可接受的。

当我们在 第 5 章 和第 6 章 和第 11 章 中引入更复杂的归纳类型时，这一点将更加重要。每当我们引入一种新的归纳定义时，我们总是首先推导它的归纳原则。只有这样我们才引入一种适当的模式匹配作为归纳原则的简写。

## 1.11 命题即类型

如导论中所述，在类型论中证明一个命题为真对应于展示该命题对应的类型的一个元素。 我们将这个类型的元素视为命题为真的*证据*或*见证*。（它们有时甚至被称为*证明*，但这个术语可能会误导，所以我们通常避免使用它。）然而一般来说，我们不会显式构造见证；相反我们用普通的数学散文呈现证明，以可以翻译成类型的元素的方式。这与在经典集合论中的推理没有区别，在那里我们也不期望看到使用谓词逻辑规则和集合论公理的显式推导。

然而，类型论对证明的观点在重要方面是不同的。类型论逻辑的基本原则是命题不仅仅是真或假，而是可以被看作其真的所有可能见证的集合。在这种概念下，证明不仅是交流数学的手段，而且是与更熟悉的对象如数、映射、群等同等地位的数学对象。因此，由于类型分类可用的数学对象并支配它们如何交互，命题不过是特殊的类型——即其元素是证明的类型。

使这种等同可行的基本观察是，我们在用英语表达的命题上的*逻辑*运算与其见证类型上的*类型论*运算之间有以下自然对应。

| 英语 | 类型论 |
|---|---|
| 真 | $\mathbf{1}$ |
| 假 | $\mathbf{0}$ |
| $A$ 且 $B$ | $A \times B$ |
| $A$ 或 $B$ | $A + B$ |
| 若 $A$ 则 $B$ | $A \to B$ |
| $A$ 当且仅当 $B$ | $(A \to B) \times (B \to A)$ |
| 非 $A$ | $A \to \mathbf{0}$ |

对应的要点是在每种情况下，构造和使用右边类型元素的规则对应于关于左边命题推理的规则。例如，证明 $A$ 且 $B$ 形式陈述的基本方式是证明 $A$ 并且也证明 $B$，而构造 $A \times B$ 的元素的基本方式是作为序对 $(a, b)$，其中 $a$ 是 $A$ 的元素（或见证）而 $b$ 是 $B$ 的元素（或见证）。如果我们想用 $A$ 且 $B$ 证明其他事情，我们可以自由地在证明中同时使用 $A$ 和 $B$，类似于 $A \times B$ 的归纳原则如何允许我们通过使用 $A$ 和 $B$ 的元素从它构造函数。

类似地，证明蕴含若 $A$ 则 $B$ 的基本方式是假设 $A$ 并证明 $B$，而构造 $A \to B$ 的元素的基本方式是给出一个表达式表示 $B$ 的元素（见证），它可能涉及类型 $A$ 的未指定变量元素（见证）。使用蕴含若 $A$ 则 $B$ 的基本方式是如果我们知道 $A$ 则推出 $B$，类似于我们可以将函数 $f : A \to B$ 应用于 $A$ 的元素来产生 $B$ 的元素。我们强烈鼓励读者做练习验证支配其他类型构造子的规则合理地翻译成逻辑。

特别值得注意的是空类型 $\mathbf{0}$ 对应于假。当从逻辑上讲时，我们将 $\mathbf{0}$ 的居民称为**矛盾**：因此没有方式证明矛盾，[9] 而从矛盾可以推出任何东西。我们还定义类型 $A$ 的**否定** 为

$$\neg A \;:\equiv\; A \to \mathbf{0}.$$

因此，$\neg A$ 的见证是函数 $A \to \mathbf{0}$，我们可以通过假设 $x : A$ 并推导 $\mathbf{0}$ 的元素来构造它。 注意虽然我们获得的逻辑是构造的，如导论中讨论的，这种反证法（假设 $A$ 并推导矛盾，得出 $\neg A$）在构造上是完全有效的：它只是援引否定的含义。不被允许的那种反证法是假设 $\neg A$ 并推导矛盾作为证明 $A$ 的方式。构造地，这样的论证只允许我们得出 $\neg\neg A$，读者可以验证没有明显的方式从 $\neg\neg A$（即从 $(A \to \mathbf{0}) \to \mathbf{0}$）得到 $A$。

上述将逻辑连接词翻译成类型形成操作被称为**命题即类型**：它给我们一种方式将用英语写的命题及其证明翻译成类型及其元素。例如，假设我们想证明以下重言式（德摩根定律之一）：

$$\text{若非 } A \text{ 且非 } B\text{，则非（}A \text{ 或 } B\text{）} \tag{1.11.1}$$

这个事实的普通英语证明可能如下。

> 假设非 $A$ 且非 $B$，并且也假设 $A$ 或 $B$；我们将推导矛盾。有两种情况。若 $A$ 成立，则由于非 $A$，我们有矛盾。类似地，若 $B$ 成立，则由于非 $B$，我们也有矛盾。因此在任一情况下我们都有矛盾，所以非（$A$ 或 $B$）。

[9] 更精确地说，没有基本的方式证明矛盾，即 $\mathbf{0}$ 没有构造子。如果我们的类型论是不一致的，那么会有某种更复杂的方式构造 $\mathbf{0}$ 的元素。

现在，根据上面给出的规则，对应于我们的重言式 (1.11.1) 的类型是

$$(A \to \mathbf{0}) \times (B \to \mathbf{0}) \to (A + B \to \mathbf{0}) \tag{1.11.2}$$

所以我们应该能够将上述证明翻译成这个类型的元素。

作为这种翻译如何工作的例子，让我们描述一个数学家阅读上述英语证明时如何可能同时在头脑中构造 (1.11.2) 的元素。介绍短语假设非 $A$ 且非 $B$ 翻译成定义一个函数，隐式应用其定义域 $(A \to \mathbf{0}) \times (B \to \mathbf{0})$ 上笛卡尔积的递归原则。这引入了未命名的变量 （假设）类型分别为 $A \to \mathbf{0}$ 和 $B \to \mathbf{0}$。当翻译成类型论时，我们必须给这些变量命名；让我们称它们为 $x$ 和 $y$。此时我们对 (1.11.2) 的元素的部分定义可以写成

$$f((x, y)) :\equiv \ \square : A + B \to \mathbf{0}$$

带有一个类型为 $A + B \to \mathbf{0}$ 的洞 $\square$ 表示还需要做什么。（我们可以等价地写 $f :\equiv \mathsf{rec}_{(A\to\mathbf{0})\times(B\to\mathbf{0})}(A + B \to \mathbf{0}, \lambda x.\, \lambda y.\, \square)$，使用递归子而不是模式匹配。）下一个短语也假设 $A$ 或 $B$；我们将推导矛盾表示通过函数定义填充这个洞，引入另一个未命名假设 $z : A + B$，导致证明状态：

$$f((x, y))(z) :\equiv \ \square : \mathbf{0}.$$

现在说有两种情况表示分情况分析，即应用余积 $A + B$ 的递归原则。如果我们用递归子写这个，它将是

$$f((x, y))(z) :\equiv \mathsf{rec}_{A+B}(\mathbf{0}, \lambda a.\, \square, \lambda b.\, \square, z)$$

而如果我们用模式匹配写，它将是

$$f((x, y))(\mathsf{inl}(a)) :\equiv \ \square : \mathbf{0}$$
$$f((x, y))(\mathsf{inr}(b)) :\equiv \ \square : \mathbf{0}.$$

注意在两种情况下我们现在有两个类型为 $\mathbf{0}$ 的洞要填充，对应于我们必须推导矛盾的两种情况。最后，从 $a : A$ 和 $x : A \to \mathbf{0}$ 得出矛盾只是将函数 $x$ 应用于 $a$，在另一种情况下类似。（注意应用函数的短语与应用假设或定理的短语的方便巧合。）因此我们最终的定义是

$$f((x, y))(\mathsf{inl}(a)) :\equiv x(a)$$
$$f((x, y))(\mathsf{inr}(b)) :\equiv y(b).$$

作为练习，你应该通过对任何类型 $A$ 和 $B$ 展示

$$((A + B) \to \mathbf{0}) \to (A \to \mathbf{0}) \times (B \to \mathbf{0})$$

的元素来验证相反的重言式若非（$A$ 或 $B$），则（非 $A$）且（非 $B$），使用我们刚刚引入的规则。

然而，并非所有经典重言式在这种解释下都成立。例如，规则 若非（$A$ 且 $B$），则（非 $A$）或（非 $B$）是无效的：我们一般不能构造相应类型

$$((A \times B) \to \mathbf{0}) \to (A \to \mathbf{0}) + (B \to \mathbf{0})$$

的元素。这反映了类型论的自然命题即类型逻辑是构造的这一事实。这意味着它不包括某些经典原则，如排中律（LEM ）或反证法，以及依赖于它们的其他原则，如德摩根定律的这个实例。

哲学上，构造逻辑之所以如此称呼，是因为它限于可以有效地执行的构造，也就是说那些具有计算意义的构造。不太精确地说，这意味着有某种算法逐步指定如何构建对象（以及作为特殊情况，如何看出定理为真）。这要求省略 LEM ，因为没有有效的程序来判定命题是真还是假。

类型论逻辑的构造性意味着它有内在的计算意义，这对计算机科学家来说很有趣。它还意味着类型论提供公理自由。例如，虽然默认情况下没有见证 LEM 的构造，但逻辑仍然与存在这样的构造兼容（见 第 3.4 节）。因此，因为类型论不否认 LEM ，我们可以一致地将其作为假设添加，并无限制地常规工作。在这方面，类型论丰富而非限制了常规数学实践。

我们鼓励不熟悉构造逻辑的读者通过更多例子来熟悉它。见 ?? 和?? 获取一些建议。

到目前为止我们只讨论了命题逻辑。现在我们考虑谓词逻辑，其中除了且和或等逻辑连接词外我们还有量词存在和对所有。在这种情况下，类型扮演双重角色：它们作为命题并且也作为常规意义上的类型，即我们量化的域。类型 $A$ 上的谓词表示为族 $P : A \to \mathcal{U}$，为每个元素 $a : A$ 分配类型 $P(a)$ 对应于 $P$ 对 $a$ 成立的命题。我们现在用量词的解释扩展上述翻译：

| 英语 | 类型论 |
|---|---|
| 对所有 $x:A$，$P(x)$ 成立 | $\prod_{(x:A)} P(x)$ |
| 存在 $x:A$ 使得 $P(x)$ | $\sum_{(x:A)} P(x)$ |

如前所述，我们可以证明（构造）谓词逻辑的重言式翻译成有居民的类型。例如，若对所有 $x:A$，$P(x)$ 且 $Q(x)$，则（对所有 $x:A$，$P(x)$）且（对所有 $x:A$，$Q(x)$）翻译为

$$(\prod_{(x:A)} P(x) \times Q(x)) \to (\prod_{(x:A)} P(x)) \times (\prod_{(x:A)} Q(x)).$$

这个重言式的非形式证明可能如下：

> 假设对所有 $x$，$P(x)$ 且 $Q(x)$。首先，我们假定给定 $x$ 并证明 $P(x)$。由假设，我们有 $P(x)$ 且 $Q(x)$，因此我们有 $P(x)$。其次，我们假定给定 $x$ 并证明 $Q(x)$。再次由假设，我们有 $P(x)$ 且 $Q(x)$，因此我们有 $Q(x)$。

第一句开始通过引入其假设的见证来定义蕴含作为函数：

$$f(p) :\equiv \ \square \ : (\prod_{(x:A)} P(x)) \times (\prod_{(x:A)} Q(x)).$$

此时有一个隐式使用的配对构造子来产生积类型的元素，在这个例子中由首先和其次这两个词多少标示：

$$f(p) :\equiv \Big( \ \square \ : \prod_{(x:A)} P(x) \ , \ \square \ : \prod_{(x:A)} Q(x) \ \Big).$$

短语我们假定给定 $x$ 并证明 $P(x)$ 现在表示以通常方式定义依赖函数，为其输入引入变量 。由于这在配对构造子内部，自然将其写成 λ-抽象：

$$f(p) :\equiv \Big( \ \lambda x. \ \big(\square \ : P(x)\big) \ , \ \square \ : \prod_{(x:A)} Q(x) \ \Big).$$

现在我们有 $P(x)$ 且 $Q(x)$ 援引假设，得到 $p(x) : P(x) \times Q(x)$，而因此我们有 $P(x)$ 隐式应用适当的投影：

$$f(p) :\equiv \Big( \ \lambda x. \, \mathsf{pr}_1(p(x)) \ , \ \square \ : \prod_{(x:A)} Q(x) \ \Big).$$

接下来的两句以明显的方式填充另一个洞：

$$f(p) :\equiv \Big( \lambda x.\, \mathsf{pr}_1(p(x)) \;,\; \lambda x.\, \mathsf{pr}_2(p(x)) \Big).$$

当然，我们一直用作例子的英语证明比数学家通常在实践中使用的要冗长得多；它们更像是在证明入门课程中使用的那种语言。执业数学家已经学会填补空白，所以在实践中我们可以省略大量细节，我们通常会这样做。然而，证明有效性的标准始终是它们可以翻译回相应类型的元素的构造。

作为更具体的例子，考虑如何定义自然数的不等式。一个自然的定义是 $n \leq m$ 当存在 $k : \mathbb{N}$ 使得 $n + k = m$。（这再次使用了我们将在下一节引入的恒等类型，但我们不需要关于它们的太多内容。）在命题即类型翻译下，这将产生：

$$(n \leq m) :\equiv \sum_{k:\mathbb{N}} (n + k = m).$$

读者被邀请从这个定义证明 $\leq$ 的熟悉性质。对于严格不等式，有几个自然的选择，如

$$(n < m) :\equiv \sum_{k:\mathbb{N}} (n + \mathsf{succ}(k) = m)$$

或

$$(n < m) :\equiv (n \leq m) \times \neg(n = m).$$

前者在构造数学中更自然，但在这种情况下它实际上等价于后者，因为 $\mathbb{N}$ 有可判定相等（见 第 3.4 节 和??）。

类型 $\sum_{(x:A)} P(x)$ 还有另一种解释。由于它的居民是元素 $x : A$ 连同 $P(x)$ 成立的见证，我们不将 $\sum_{(x:A)} P(x)$ 视为命题存在 $x : A$ 使得 $P(x)$，而可以将其视为满足 $P(x)$ 的所有元素 $x : A$ 的类型，即 $A$ 的子类型。

我们将在 第 3.5 节中回到这种解释。现在，我们注意到它允许我们将公理纳入我们在 第 1.6 节中讨论的作为数学结构的类型的定义中。例如，假设我们想定义**半群**为类型 $A$ 配备二元运算 $m : A \to A \to A$（即原群）并且对所有 $x, y, z : A$ 有 $m(x, m(y, z)) = m(m(x, y), z)$。后一个命题由类型

$$\prod_{x,y,z:A} m(x, m(y, z)) = m(m(x, y), z)$$

表示，所以半群的类型是

$$\mathsf{Semigroup} :\equiv \sum_{(A:\mathcal{U})} \sum_{(m:A\to A\to A)} \prod_{(x,y,z:A)} m(x, m(y, z)) = m(m(x, y), z),$$

即由半群组成的 Magma 的子类型。从 Semigroup 的居民我们可以通过应用适当的投影提取载体 $A$、运算 $m$ 和公理的见证。我们将在 第 2.14 节中回到这个例子。

还要注意我们可以使用类型论中的宇宙来表示高阶逻辑——即我们可以对所有命题或所有谓词进行量化。例如，我们可以将命题*对所有性质* $P : A \to \mathcal{U}$，*若* $P(a)$ *则* $P(b)$ 表示为

$$\prod_{P:A\to\mathcal{U}} P(a) \to P(b)$$

其中 $A : \mathcal{U}$ 且 $a, b : A$。然而，*先验地*这个命题生活在与我们量化的命题不同的、更高的宇宙中；即

$$\Big(\prod_{P:A\to\mathcal{U}_i} P(a) \to P(b)\Big) : \mathcal{U}_{i+1}.$$

我们将在 第 3.5 节中回到这个问题。

我们在这里描述了命题的证明相关 翻译，其中析取和存在语句的证明携带一些信息。例如，如果我们有 $A + B$ 的居民，视为 $A$ 或 $B$ 的见证，那么我们知道它来自 $A$ 还是 $B$。类似地，如果我们有 $\sum_{(x:A)} P(x)$ 的居民，视为存在 $x : A$ 使得 $P(x)$ 的见证，那么我们知道元素 $x$ 是什么（它是给定居民的第一投影）。

作为这种逻辑的证明相关性质的结果，我们可能有 $A$ 当且仅当 $B$（回忆这意味着 $(A \to B) \times (B \to A)$），但类型 $A$ 和 $B$ 表现出不同的行为。例如，容易验证 $\mathbb{N}$ 当且仅当 $\mathbf{1}$，但显然 $\mathbb{N}$ 和 $\mathbf{1}$ 在重要方面不同。陈述 $\mathbb{N}$ 当且仅当 $\mathbf{1}$ 只告诉我们当*作为纯粹命题看待*时，类型 $\mathbb{N}$ 代表与 $\mathbf{1}$ 相同的命题（在这种情况下是真命题）。我们有时用 $A$ 和 $B$ 是**逻辑等价**的来表达 $A$ 当且仅当 $B$。这要与将在 第 2.4 节 和第 4 章 中引入的更强的*类型等价*概念区分开：虽然 $\mathbb{N}$ 和 $\mathbf{1}$ 逻辑等价，但它们不是等价类型。

在 第 3 章中我们将引入一类称为纯粹命题的类型，对它们等价与逻辑等价一致。使用这些类型，我们将引入对上述逻辑的修改，这在某些情况下是适当的，其中析取和存在中包含的额外信息被丢弃。

最后，我们注意到命题即类型对应可以反过来看，允许我们将任何类型 $A$ 视为命题，我们通过展示 $A$ 的元素来证明它。有时我们将这个命题表述为 $A$ 是**有居民的**。即，当我们说 $A$ 有居民时，我们的意思是我们给出了 $A$ 的一个（特定的）元素，但我们选择不给那个元素命名。类似地，说 $A$ *没有居民*与给出 $\neg A$ 的元素是一样的。特别是，空类型 $\mathbf{0}$ 显然没有居民，因为 $\neg\mathbf{0} \equiv (\mathbf{0} \to \mathbf{0})$ 被 $\mathsf{id}_{\mathbf{0}}$ 居于。[10]

## 1.12 恒等类型

虽然前面的构造可以看作是标准集合论构造的推广，但我们处理恒等的方式似乎是类型论特有的。根据命题即类型的概念，同一类型的两个元素 $a, b : A$ 相等的*命题*必须对应于某个*类型*。由于这个命题依赖于 $a$ 和 $b$ 是什么，这些**等式类型**或**恒等类型**必须是依赖于 $A$ 的两个副本的类型族。

我们可以将这个族写成 $\mathsf{Id}_A : A \to A \to \mathcal{U}$（不要与恒等函数 $\mathsf{id}_A$ 混淆），使得 $\mathsf{Id}_A(a, b)$ 是表示 $a$ 和 $b$ 之间相等命题的类型。然而一旦我们熟悉了命题即类型，使用标准的等式符号也是方便的；因此 $a = b$ 也将是对应于 $a$ 等于 $b$ 的命题的*类型* $\mathsf{Id}_A(a, b)$ 的记法。为清楚起见，我们也可以写 $a =_A b$ 来指定类型 $A$。如果我们有 $a =_A b$ 的元素，我们可以说 $a$ 和 $b$ **相等**，或者有时是**命题相等**，如果我们想强调这与 第 1.1 节中讨论的判断等式 $a \equiv b$ 不同。

正如我们在 第 1.11 节中指出的命题即类型版本的或和存在可以包含比命题为真这一事实更多的信息，没有什么阻止类型 $a = b$ 也包含更多信息。确实，这是同伦解释的基石，在那里我们将 $a = b$ 的见证视为空间 $A$ 中 $a$ 和 $b$ 之间的*路径*或*等价*。正如空间的两点之间可以有多于一条路径，两个对象相等可以有多于一个见证。换言之，我们可以将 $a = b$ 视为 $a$ 和 $b$ 的*认同*的类型，而 $a$ 和 $b$ 可能有许多不同的认同方式。我们将在 第 2 章

[10] 这不应与类型论一致这一陈述混淆，后者是*元理论*声明说不可能通过遵循类型论规则获得 $\mathbf{0}$ 的元素。

中回到这种解释；现在我们专注于恒等类型的基本规则。就像本章考虑的所有其他类型一样，它将有形成、引入、消除和计算规则，它们在形式上以完全相同的方式行为。

形成规则说给定类型 $A : \mathcal{U}$ 和两个元素 $a, b : A$，我们可以在同一宇宙中形成类型 $(a =_A b) : \mathcal{U}$。构造 $a = b$ 的元素的基本方式是知道 $a$ 和 $b$ 是相同的。因此，引入规则是依赖函数

$$\mathsf{refl} : \prod_{a:A} (a =_A a)$$

称为**自反性**，它说 $A$ 的每个元素都等于它自己（以指定的方式）。我们将 $\mathsf{refl}_a$ 视为点 $a$ 处的常值路径。

特别是，这意味着如果 $a$ 和 $b$ 判断上相等，$a \equiv b$，那么我们也有元素 $\mathsf{refl}_a : a =_A b$。这是类型良好的，因为 $a \equiv b$ 意味着类型 $a =_A b$ 也判断上等于 $a =_A a$，这是 $\mathsf{refl}_a$ 的类型。

恒等类型的归纳原则（即消除规则）是类型论最微妙的部分之一，对同伦解释至关重要。我们首先考虑它的一个重要结果，即等量可以代换等量原则，如下所述：

**等同者不可区分：** 对于每个族

$$C : A \to \mathcal{U}$$

有函数

$$f : \prod_{(x,y:A)} \prod_{(p:x=_A y)} C(x) \to C(y)$$

使得

$$f(x, x, \mathsf{refl}_x) :\equiv \mathsf{id}_{C(x)}.$$

这说的是每个类型族 $C$ 都尊重等式，即将 $C$ 应用于 $A$ 的相等元素也产生结果类型之间的函数。所显示的等式陈述与自反性相关的函数是恒等函数（我们将看到，一般来说，函数 $f(x, y, p) : C(x) \to C(y)$ 总是类型等价）。

等同者不可区分可以被视为恒等类型的递归原则，类似于上面给出的布尔和自然数的递归原则。正如 $\mathsf{rec}_{\mathbb{N}}$ 给出对于某种类型的任何其他类型 $C$

的指定映射 $\mathbb{N} \to C$，等同者不可区分给出从 $x =_A y$ 到 $A$ 上某些其他自反二元关系的指定映射，即那些对某一元谓词 $C(x)$ 形如 $C(x) \to C(y)$ 的关系。我们也可以表述更一般的递归原则，关于更一般形式 $C(x, y)$ 的自反关系。然而，为了完全刻画恒等类型，我们必须将这个递归原则推广到归纳原则，它不仅考虑从 $x =_A y$ 出发的映射，还考虑其上的族。换言之，我们不仅考虑允许等量代换等量，还要考虑等式的证据 $p$。

### 1.12.1 路径归纳

恒等类型的归纳原则称为**路径归纳**，鉴于 第 2 章导论中将解释的同伦解释。它可以被看作是说恒等类型的族由形如 $\mathsf{refl}_x : x = x$ 的元素自由生成。

**路径归纳：** 给定族

$$C : \prod_{x,y:A} (x =_A y) \to \mathcal{U}$$

和函数

$$c : \prod_{x:A} C(x, x, \mathsf{refl}_x),$$

有函数

$$f : \prod_{(x,y:A)} \prod_{(p:x=_A y)} C(x, y, p)$$

使得

$$f(x, x, \mathsf{refl}_x) :\equiv c(x).$$

注意就像积、余积、自然数等的归纳原则一样，路径归纳允许我们定义表现出适当计算行为的指定函数。就像我们有从 $c_0 : C$ 和 $c_s : \mathbb{N} \to C \to C$ 通过递归定义的那个函数 $f : \mathbb{N} \to C$，它还满足 $f(0) \equiv c_0$ 和 $f(\mathsf{succ}(n)) \equiv c_s(n, f(n))$，我们有从 $c : \prod_{(x:A)} C(x, x, \mathsf{refl}_x)$ 通过路径归纳定义的那个函数 $f : \prod_{(x,y:A)} \prod_{(p:x=_A y)} C(x, y, p)$，它还满足 $f(x, x, \mathsf{refl}_x) \equiv c(x)$。

为了理解这个原则的含义，首先考虑 $C$ 不依赖于 $p$ 的更简单情况。那么我们有 $C : A \to A \to \mathcal{U}$，我们可以将其视为依赖于 $A$ 的两个元素的谓词。我们感兴趣的是知道何时命题 $C(x, y)$ 对某对元素 $x, y : A$ 成立。在这

种情况下，路径归纳的假设说我们知道 $C(x,x)$ 对所有 $x:A$ 成立，即如果我们在序对 $x,x$ 处求值 $C$，我们得到真命题——所以 $C$ 是自反关系。结论则告诉我们只要 $x=y$，$C(x,y)$ 就成立。这正是上面提到的对自反关系的更一般递归原则。

规则的一般归纳形式允许 $C$ 也依赖于 $x$ 和 $y$ 之间恒等的见证 $p:x=y$。在前提中，我们不仅用 $x,x$ 替换 $x,y$，还同时用自反性替换 $p$：要证明所有元素 $x,y$ 及它们之间的路径 $p:x=y$ 的性质，只需考虑元素是 $x,x$ 且路径是 $\mathsf{refl}_x:x=x$ 的所有情况。如果我们只将类型视为集合，不清楚这给我们带来什么好处，但由于 $x$ 和 $y$ 之间可能有许多不同的认同 $p:x=y$，在考虑类型 $x=_A y$ 上的族时追踪它们是有意义的。在 第 2 章中我们将看到这对同伦解释非常重要。

如果我们将路径归纳打包成单个函数，它采取形式：

$$\mathsf{ind}_{=_A}:\prod_{(C:\prod_{(x,y:A)}(x=_A y)\to\mathcal{U})}\left(\prod_{(x:A)}C(x,x,\mathsf{refl}_x)\right)\to\prod_{(x,y:A)}\prod_{(p:x=_A y)}C(x,y,p)$$

及等式

$$\mathsf{ind}_{=_A}(C,c,x,x,\mathsf{refl}_x):\equiv c(x).$$

函数 $\mathsf{ind}_{=_A}$ 传统上称为 $J$。我们将在 引理 2.3.1 中证明等同者不可区分是路径归纳的实例，并给它一个新名字和记法。

给定证明 $p:a=b$，路径归纳要求我们用相同的未知元素 $x$ 同时替换 $a$ 和 $b$；因此为了定义族 $C$ 的元素，对于 $A$ 的所有相等元素对，只需在对角线上定义它就足够了。然而在某些证明中，通过用 $a$ 替换 $b$ 的所有出现（或反之）来利用方程 $p:a=b$ 更简单，因为有时对等式中提到的特定元素 $a$ 比对一般未知 $x$ 更容易完成证明的其余部分。这促使恒等类型有第二个归纳原则，它说类型族 $a=_A x$ 由元素 $\mathsf{refl}_a:a=a$ 生成。如下所示，这第二个原则等价于第一个；它只是有时是更方便的表述。

**基点路径归纳：** 固定元素 $a:A$，并假定给定族

$$C:\prod_{x:A}(a=_A x)\to\mathcal{U}$$

和元素

$$c : C(a, \mathsf{refl}_a).$$

则我们得到函数

$$f : \prod_{(x:A)} \prod_{(p:a=x)} C(x, p)$$

使得

$$f(a, \mathsf{refl}_a) :\equiv c.$$

这里，$C(x,p)$ 是类型族，其中 $x$ 是 $A$ 的元素而 $p$ 是恒等类型 $a =_A x$ 的元素，对于固定的 $a$ 在 $A$ 中。基点路径归纳原则说要为所有 $x$ 和 $p$ 定义这个族的元素，只需考虑 $x$ 是 $a$ 且 $p$ 是 $\mathsf{refl}_a : a = a$ 的情况。

打包成函数，基点路径归纳变成：

$$\mathsf{ind}'_{=_A} : \prod_{(a:A)} \prod_{(C:\prod_{(x:A)}(a=_A x)\to\mathcal{U})} C(a, \mathsf{refl}_a) \to \prod_{(x:A)} \prod_{(p:a=_A x)} C(x, p)$$

及等式

$$\mathsf{ind}'_{=_A}(a, C, c, a, \mathsf{refl}_a) :\equiv c.$$

下面，我们证明路径归纳和基点路径归纳是等价的。因此，我们有时会马虎地将基点路径归纳也简单称为路径归纳，依靠读者从证明的形式推断是哪个原则。

*Remark* 1.12.1. 直观地，自然数的归纳原则表达了每个自然数要么是 0 要么对某个自然数 $n$ 形如 $\mathsf{succ}(n)$ 的事实，所以如果我们对这些情况证明性质（在第二种情况下有归纳假设），那么我们就对所有自然数证明了它。类似地，$A + B$ 的归纳原则表达了 $A + B$ 的每个元素要么形如 $\mathsf{inl}(a)$ 要么形如 $\mathsf{inr}(b)$ 的事实，等等。将同样的解读应用于路径归纳，我们可能说路径归纳表达了每条路径形如 $\mathsf{refl}_a$ 的事实，所以如果我们对自反性路径证明性质，那么我们就对所有路径证明了它。

然而，这种解读在路径的同伦解释的背景下相当令人困惑，其中两个元素 $a$ 和 $b$ 可能有许多不同的认同方式，因此恒等类型有许多不同的元素！怎么可能有许多不同的路径，但同时我们有一个归纳原则断言唯一的路径是自反性？

关键观察是被归纳定义的不是恒等类型，而是恒等族。特别是，路径归纳说类型的族 $(x =_A y)$，当 $x, y$ 变化于 $A$ 的所有元素时，由形如 $\mathsf{refl}_x$ 的元素归纳定义。这意味着要给出依赖于恒等族的一般元素 $(x, y, p)$ 的任何其他族 $C(x, y, p)$ 的元素，只需考虑形如 $(x, x, \mathsf{refl}_x)$ 的情况。在同伦解释中，这说三元组 $(x, y, p)$ 的类型，其中 $x$ 和 $y$ 是路径 $p$ 的端点（换言之，$\Sigma$-类型 $\sum_{(x,y:A)}(x = y)$），由每点 $x$ 处的常值环路归纳生成。如我们将在 第 2 章中看到的，在同伦论中对应于 $\sum_{(x,y:A)}(x = y)$ 的空间是自由路径空间——$A$ 中端点可变的路径的空间——确实这个空间的任何点都同伦于某点处的常值环路，因为我们可以简单地沿给定路径收缩它的一个端点。类似的事实在类型论中也成立：我们可以通过对 $p : x = y$ 的路径归纳证明 $(x, y, p) =_{\sum_{(x,y:A)}(x=y)} (x, x, \mathsf{refl}_x)$。

类似地，基点路径归纳说对于固定的 $a : A$，类型的族 $(a =_A y)$，当 $y$ 变化于 $A$ 的所有元素时，由元素 $\mathsf{refl}_a$ 归纳定义。因此，要给出依赖于这个族的一般元素 $(y, p)$ 的任何其他族 $C(y, p)$ 的元素，只需考虑情况 $(a, \mathsf{refl}_a)$。同伦地，这表达了从某选定点出发的路径的空间（那点处的基点路径空间，类型论上是 $\sum_{(y:A)}(a = y)$）可缩到选定点处的常值环路。同样，相应的事实在类型论中也成立：我们可以通过对 $p : a = y$ 的基点路径归纳证明 $(y, p) =_{\sum_{(y:A)}(a=y)} (a, \mathsf{refl}_a)$。还要注意根据 第 1.11 节中提到的 $\Sigma$-类型作为子类型的解释，类型 $\sum_{(y:A)}(a = y)$ 可以被视为等于 $a$ 的所有 $A$ 的元素的类型，即单点子集 $\{a\}$ 的类型论版本。

路径归纳和基点路径归纳都不提供给出族 $C(p)$ 的元素的方式，其中 $p$ 有两个固定端点 $a$ 和 $b$。特别是，对于依赖于环路的族 $C : (a =_A a) \to \mathcal{U}$，我们不能应用路径归纳只考虑 $C(\mathsf{refl}_a)$ 的情况，因此我们不能证明所有环路都是自反性。因此，归纳定义恒等族并不禁止恒等类型特定实例中的非自反性路径。换言之，路径 $p : x = x$ 作为 $(x = x)$ 的元素可能不等于自反性，但序对 $(x, p)$ 仍然会等于序对 $(x, \mathsf{refl}_x)$ 作为 $\sum_{(y:A)}(x = y)$ 的元素。

作为拓扑例子，考虑穿孔圆盘

$$\left\{ (x, y) \,\middle|\, 0 < x^2 + y^2 < 2 \right\}$$

中从 $(1, 0)$ 出发绕 $(0, 0)$ 处的洞一圈后返回 $(1, 0)$ 的环路。如果我们将两个端点都固定在 $(1, 0)$，这个环路不能在保持在穿孔圆盘内的同时变形为常值

路径，就像绕杆子缠绕的绳子如果我们握住两端就不能拉进来。然而，如果我们允许一个端点变化，环路可以收缩回常值，就像如果我们只握住一端就总能收回绳子。

### 1.12.2 路径归纳与基点路径归纳的等价性

上面引入的两个恒等类型归纳原则是等价的。容易看出路径归纳可从基点路径归纳原则推出。确实，让我们假设路径归纳的前提：

$$C : \prod_{x,y:A} (x =_A y) \to \mathcal{U},$$
$$c : \prod_{x:A} C(x, x, \mathsf{refl}_x).$$

现在，给定元素 $x : A$，我们可以将上述两者实例化，得到

$$C' : \prod_{y:A} (x =_A y) \to \mathcal{U},$$
$$C' :\equiv C(x),$$
$$c' : C'(x, \mathsf{refl}_x),$$
$$c' :\equiv c(x).$$

显然，$C'$ 和 $c'$ 匹配基点路径归纳的前提，因此我们可以构造

$$g : \prod_{(y:A)} \prod_{(p:x=y)} C'(y, p)$$

及定义等式

$$g(x, \mathsf{refl}_x) :\equiv c'.$$

现在我们观察到 $g$ 的陪域等于 $C(x, y, p)$。因此，释放我们的假设 $x : A$，我们可以推导函数

$$f : \prod_{(x,y:A)} \prod_{(p:x=_A y)} C(x, y, p)$$

及所需的判断等式 $f(x, x, \mathsf{refl}_x) \equiv g(x, \mathsf{refl}_x) :\equiv c' :\equiv c(x)$。

另一个证明是观察到任何这样的 $f$ 都可以作为 $\mathsf{ind}_{=_A}$ 的实例获得，所以只需将 $\mathsf{ind}_{=_A}$ 用 $\mathsf{ind}'_{=_A}$ 定义为

$$\mathsf{ind}_{=_A}(C, c, x, y, p) :\equiv \mathsf{ind}'_{=_A}(x, C(x), c(x), y, p).$$

另一个方向更棘手；不清楚我们如何能用路径归纳的特定实例推导基点路径归纳的特定实例。我们能做的是构造路径归纳的一个实例，它一次证明基点路径归纳的所有可能实例化。定义

$$\begin{aligned} D &: \prod_{x,y:A} (x =_A y) \to \mathcal{U}, \\ D(x, y, p) &:\equiv \prod_{C:\prod_{(z:A)}(x=_A z)\to\mathcal{U}} C(x, \mathsf{refl}_x) \to C(y, p). \end{aligned}$$

然后我们可以构造函数

$$\begin{aligned} d &: \prod_{x:A} D(x, x, \mathsf{refl}_x), \\ d &:\equiv \lambda x.\, \lambda C.\, \lambda(c : C(x, \mathsf{refl}_x)).\, c \end{aligned}$$

因此使用路径归纳得到

$$f : \prod_{(x,y:A)} \prod_{(p:x=_A y)} D(x, y, p)$$

及 $f(x, x, \mathsf{refl}_x) :\equiv d(x)$。展开 $D$ 的定义，我们可以扩展 $f$ 的类型：

$$f : \prod_{(x,y:A)} \prod_{(p:x=_A y)} \prod_{(C:\prod_{(z:A)}(x=_A z)\to\mathcal{U})} C(x, \mathsf{refl}_x) \to C(y, p).$$

现在给定 $a : A$ 连同 $x : A$ 和 $p : a =_A x$，我们可以推导基点路径归纳的结论：

$$f(a, x, p, C, c) : C(x, p).$$

注意我们也获得正确的定义等式。

另一个证明是观察到基点路径归纳的任何使用都是 $\mathsf{ind}'_{=_A}$ 的实例并定义

$$\begin{aligned}\mathsf{ind}'_{=_A}(a, C, c, x, p) :\equiv \\ &\mathsf{ind}_{=_A}\big((\lambda x, y.\, \lambda p.\, \textstyle\prod_{(C:\prod_{(z:A)}(x=_A z)\to\mathcal{U})} C(x, \mathsf{refl}_x) \to C(y, p)\big), \\ &\quad (\lambda x.\, \lambda C.\, \lambda d.\, d), a, x, p\big)(C, c).\end{aligned}$$

注意上面给出的构造使用宇宙。即，如果我们想用 $C : \prod_{(x:A)}(a =_A x) \to \mathcal{U}_i$ 建模 $\mathsf{ind}'_{=_A}$，我们需要用

$$D : \prod_{x,y:A} (x =_A y) \to \mathcal{U}_{i+1}$$

使用 $\mathsf{ind}_{=_A}$，因为 $D$ 对给定类型的所有 $C$ 进行量化。虽然这与我们对宇宙的定义兼容，但也可以不使用宇宙推导 $\mathsf{ind}'_{=_A}$：我们可以证明 $\mathsf{ind}_{=_A}$ 蕴含引理 2.3.1 和定理 3.11.8，而这两个原则直接蕴含 $\mathsf{ind}'_{=_A}$。我们将细节留给读者作为 ??。

我们可以使用恒等类型的前述任一表述来建立等式是等价关系，每个函数保持等式且每个族尊重等式。我们将细节留到下一章，在那里这将在同伦类型论的背景下推导和解释。

*Remark* 1.12.2. 我们强调尽管有一些不熟悉的特征，命题等式是同伦类型论中的数学等式。这个区别不属于判断等式，后者是类型论规则的元理论特征。例如，第 1.9 节中证明的自然数加法的结合性是命题等式，而不是判断等式。交换律也是如此（??）。甚至非常简单的交换性 $n + 1 = 1 + n$ 对一般的 $n$ 也不是判断等式（虽然对任何特定的 $n$ 它是判断等式，例如 $3 + 1 \equiv 1 + 3$，因为两者通过定义 $+$ 的计算规则都判断上等于 $4$）。我们只能通过使用恒等类型证明这些事实，因为我们只能将 $\mathbb{N}$ 的归纳原则应用于类型作为输出（而不是判断）。

### 1.12.3 不相等

最后，让我们也说说**不相等**，它是等式的否定：[11]

$$(x \neq_A y) :\equiv \neg(x =_A y).$$

如果 $x \neq y$，我们说 $x$ 和 $y$ **不相等** 或**不等**。就像否定一样，不相等在这里扮演的角色不如在经典数学中重要。例如，我们不能通过证明两个东西不是不相等来证明它们相等：那将是经典双重否定律的应用，见 第 3.4 节。

有时用正面方式表述不相等是有用的。例如，在 ?? 中我们将证明实数 $x$ 有逆当且仅当它到 0 的距离是正的，这是比 $x \neq 0$ 更强的要求。

[11] 我们用不等式指 $<$ 和 $\leq$。另外，注意这是命题恒等类型的否定。当然，否定判断等式 $\equiv$ 是没有意义的，因为判断不受逻辑运算的约束。

# 第 2 章　同伦类型论

同伦类型论中核心的新思想是类型可以被视为同伦论中的空间，或范畴论中的高阶群胚。

我们首先简要总结同伦论与高维范畴论之间的联系。在经典同伦论中，空间 $X$ 是一个配备了拓扑的点集， 而点 $x$ 与 $y$ 之间的路径由连续映射 $p : [0,1] \to X$ 表示，其中 $p(0) = x$ 且 $p(1) = y$。这个函数可以被视为在每个 “时刻” 给出 $X$ 中的一个点。对于许多目的而言，路径的严格相等（即逐点相等的函数）是一个过于精细的概念。例如，可以定义路径复合的运算（如果 $p$ 是从 $x$ 到 $y$ 的路径，$q$ 是从 $y$ 到 $z$ 的路径，那么复合 $p \cdot q$ 是从 $x$ 到 $z$ 的路径）和逆运算（$p^{-1}$ 是从 $y$ 到 $x$ 的路径）。然而，这些运算之间存在对于严格相等不成立的自然等式：例如，路径 $p \cdot p^{-1}$（当时间从 0 到 1 时，沿着同一条路线从 $x$ 走到 $y$，然后再走回来）并不严格等于恒等路径（始终停留在 $x$ 不动）。

补救方法是考虑一个称为同伦的更粗糙的路径相等概念。一对连续映射 $f : X_1 \to X_2$ 和 $g : X_1 \to X_2$ 之间的同伦是一个连续映射 $H : X_1 \times [0,1] \to X_2$，满足 $H(x,0) = f(x)$ 和 $H(x,1) = g(x)$。在从 $x$ 到 $y$ 的路径 $p$ 和 $q$ 的特定情况下，同伦是一个连续映射 $H : [0,1] \times [0,1] \to X$，使得对所有 $s \in [0,1]$ 有 $H(s,0) = p(s)$ 且 $H(s,1) = q(s)$。在这种情况下，我们还要求对所有 $t \in [0,1]$ 有 $H(0,t) = x$ 且 $H(1,t) = y$，从而对每个 $t$，函数 $H(-,t)$ 仍然是从 $x$ 到 $y$ 的路径；这种同伦被称为保端点的或相对于端点的。在简单情况下，我们可以把正方形 $[0,1] \times [0,1]$ 在 $H$ 下的像想象成 “填充了” $p$ 和 $q$ 之间的空间，尽管对于一般的 $X$ 这并没有实际意义；最好把 $H$ 想象成 $p$ 到 $q$ 的一个不移动端点的连续形变。由于 $[0,1] \times [0,1]$ 是二维的，我们也称 $H$ 为二维的路径之间的路径。

例如，因为 $p \cdot p^{-1}$ 沿着同一条路线走出去又走回来，你知道可以把 $p \cdot p^{-1}$ 连续地收缩到恒等路径——它不会，比如说，被空间中的一个洞缠住。同伦是一个等价关系，而复合、逆等运算都保持它。此外，在某点 $x_0$ 处的环路的同伦等价类（其中两个环路 $p$ 和 $q$ 在它们之间存在一个基点同伦时被等同，这是一个上述同伦 $H$，还需满足对所有 $t$ 有 $H(0,t) = H(1,t) = x_0$）构成一个称为*基本群*的群。这个群是空间的一个*代数不变量*，可以用来研究两个空间是否*同伦等价*（存在来回的连续映射，其复合同伦于恒等），因为等价的空间有同构的基本群。

因为同伦本身是一种二维路径，所以存在三维的*同伦之间的同伦*的自然概念，然后是*同伦之间的同伦之间的同伦*，依此类推。这个由点、路径、同伦、同伦之间的同伦、……组成的无限塔，配备了诸如基本群之类的代数运算，是一个称为（弱）$\infty$-*群胚*的代数结构的实例。$\infty$-群胚由一个对象的集合，然后是对象之间的*态射*的集合，然后是态射之间的态射的集合，依此类推组成，并配备了一些复杂的代数结构；第 $k$ 层的态射称为 $k$-**态射**。每个层次的态射都有恒等、复合和逆运算，它们是弱的，意思是它们只在下一层态射的意义上满足群胚定律（复合的结合律、恒等是复合的单位元、逆元可消），而这种弱性产生了进一步的结构。例如，因为态射复合的结合律 $p \cdot (q \cdot r) = (p \cdot q) \cdot r$ 本身是一个高维态射，所以需要一个额外的运算来关联各种结合律的证明：将 $p \cdot (q \cdot (r \cdot s))$ 重新结合成 $((p \cdot q) \cdot r) \cdot s$ 的各种方式产生了 Mac Lane 五边形。弱性还在不同层次之间产生非平凡的相互作用。

每个拓扑空间 $X$ 都有一个*基本 $\infty$-群胚*，其 $k$-态射是 $X$ 中的 $k$-维路径。$\infty$-群胚的弱性直接对应于路径只在同伦意义上构成群这一事实，$(k+1)$-路径充当 $k$-路径之间的同伦。此外，把空间看作 $\infty$-群胚保留了空间足够多的方面来进行同伦论研究：基本 $\infty$-群胚构造与 $\infty$-群胚的几何实现互为伴随，而且这个伴随保持同伦论（这被称为*同伦假设/定理*，它是假设还是定理取决于你如何定义 $\infty$-群胚）。例如，你可以容易地定义 $\infty$-群胚的基本群，如果你计算一个空间的基本 $\infty$-群胚的基本群，它将与该空间基本群的经典定义一致。由于这种对应，同伦论和高维范畴论紧密相关。

现在，在同伦类型论中，每个类型都可以看作具有 $\infty$-群胚的结构。回忆对于任何类型 $A$ 和任意 $x, y : A$，我们有恒等类型 $x =_A y$，也记作 $\mathsf{Id}_A(x,y)$

或简记为 $x = y$。从逻辑上讲，我们可以把 $x = y$ 的元素视为 $x$ 和 $y$ 相等的证据，或者说是 $x$ 与 $y$ 的等同。此外，类型论（不同于，比如说，一阶逻辑）允许我们把 $x =_A y$ 的这些元素也作为个体来考虑，它们可以成为进一步命题的主词。因此，我们可以迭代恒等类型：我们可以构造等同之间的等同的类型 $p =_{(x=_A y)} q$，以及 $r =_{(p=_{(x=_A y)} q)} s$，依此类推。这个恒等类型塔的结构精确对应于空间中连续路径和（更高阶）同伦之间的结构，或者 $\infty$-群胚。

因此，我们将经常把元素 $p : x =_A y$ 称为从 $x$ 到 $y$ 的**路径**；我们称 $x$ 为它的**起点**，$y$ 为它的**终点**。两条起点和终点相同的路径 $p, q : x =_A y$ 被称为**平行的**，在这种情况下，元素 $r : p =_{(x=_A y)} q$ 可以被视为一个同伦，或者态射之间的态射；我们经常称它为 **2-路径** 或**二维路径**。类似地，$r =_{(p=_{(x=_A y)} q)} s$ 是两条平行的二维路径之间的**三维路径** 的类型，依此类推。如果类型 $A$ 是“集合状的”，如 $\mathbb{N}$，这些迭代的恒等类型将是无趣的（见第 3.1 节），但在一般情况下，它们可以模拟非平凡的同伦类型。

同伦类型论与经典同伦论的一个重要区别是，同伦类型论提供了空间的**综合** 描述，在以下意义上。综合几何是欧几里得 [**?**]风格的几何：从一些基本概念（点和线）、构造（连接任意两点的线）和公理（所有直角相等）出发，逻辑地推导出结论。这与**解析** 几何形成对比，在解析几何中，点和线等概念用 $\mathbb{R}^n$ 中的笛卡尔坐标具体表示——线是点的集合——而基本构造和公理从这种表示推导出来。虽然经典同伦论是解析的（空间和路径由点构成），同伦类型论是综合的：点、路径和路径之间的路径是基本的、不可分割的、原始的概念。

此外，同伦类型论令人惊奇的事情之一是，所有的基本构造和公理——所有的高阶群胚结构——都自动地从恒等类型的归纳原理产生。回忆第 1.12 节，这说的是如果

- 对每个 $x, y : A$ 和每个 $p : x =_A y$ 我们有一个类型 $D(x, y, p)$，且
- 对每个 $a : A$ 我们有一个元素 $d(a) : D(a, a, \mathsf{refl}_a)$，

那么

- 存在一个元素 $\mathsf{ind}_{=_A}(D, d, x, y, p) : D(x, y, p)$，对每两个元素 $x, y : A$ 和 $p : x =_A y$ 成立，使得 $\mathsf{ind}_{=_A}(D, d, a, a, \mathsf{refl}_a) \equiv d(a)$。

换句话说，给定依赖函数

$$D : \prod_{x,y:A} (x = y) \to \mathcal{U}$$
$$d : \prod_{a:A} D(a, a, \mathsf{refl}_a)$$

存在一个依赖函数

$$\mathsf{ind}_{=_A}(D, d) : \prod_{(x,y:A)} \prod_{(p:x=y)} D(x, y, p)$$

使得

$$\mathsf{ind}_{=_A}(D, d, a, a, \mathsf{refl}_a) \equiv d(a) \tag{2.0.1}$$

对每个 $a : A$ 成立。通常，每次我们应用这个归纳规则时，我们要么不关心正在定义的具体函数，要么立即给它一个不同的名字。

非形式地，恒等类型的归纳原理说的是，如果我们想要构造一个依赖于恒等类型的居留元 $p : x =_A y$ 的对象（或证明一个陈述），那么只需在 $x$ 和 $y$（判断上）相同且 $p$（判断上）是自反性元素 $\mathsf{refl}_x : x = x$ 的特殊情况下进行构造（或证明）即可。在非形式书写时，我们可以用诸如 “由归纳，只需假设……” 这样的短语来表达这一点。这种到 “自反性情形” 的归约类似于自然数上普通归纳证明中到 “基础情形” 和 “归纳步骤” 的归约，也类似于对不交并或析取进行情形分析证明中的 “左情形” 和 “右情形”。

“转换规则” (2.0.1) 在自然数归纳证明的语境中不太熟悉，但在相关的递归定义概念中有一个类似的观念。如果一个序列 $(a_n)_{n\in\mathbb{N}}$ 通过给出 $a_0$ 并用 $a_n$ 来指定 $a_{n+1}$ 来定义，那么事实上所得序列的第 0 项就是给定的那个，而给定的关于 $a_{n+1}$ 与 $a_n$ 的递推关系对所得序列成立。（这可能看起来太明显以至于不值得说，但如果我们把递归定义视为计算序列值的算法，那么这正是执行该算法的过程。）规则 (2.0.1) 是类似的：它说的是如果我们通过指定当 $p$ 是 $\mathsf{refl}_x : x = x$ 时值应该是什么来为所有 $p : x = y$ 定义一个对象 $f(p)$，那么我们指定的值事实上就是 $f(\mathsf{refl}_x)$ 的值。

这个归纳原理赋予每个类型 $\infty$-群胚的结构，赋予两个类型之间的每个函数两个这样的群胚之间的 $\infty$-函子的结构。这从数学的角度来看很有趣，因为它给出了一种处理 $\infty$-群胚的新方法。从类型论的角度来看也很有趣，

因为它揭示了与每个类型和函数相关联的新运算。在本章的剩余部分，我们开始探索这种结构。

## 2.1 类型是高阶群胚

我们现在从归纳原理推导出高阶群胚结构的开端。我们从相等的对称性开始，用拓扑语言来说就是“路径可以被逆转”。

**Lemma 2.1.1.** *对于每个类型 $A$ 和每个 $x, y : A$，存在一个函数*

$$(x = y) \to (y = x)$$

*记作 $p \mapsto p^{-1}$，使得对每个 $x : A$ 有 $\mathsf{refl}_x{}^{-1} \equiv \mathsf{refl}_x$。我们称 $p^{-1}$ 为 $p$ 的逆。*

既然这是我们第一次把某些东西表述为“引理”或“定理”，让我们停下来考虑一下这意味着什么。回忆命题（可被证明的陈述）与类型等同，而引理和定理（已被证明的陈述）与*有居留元的*类型等同。因此，引理或定理的陈述应该被翻译成一个类型，如第 1.11 节中那样，而它的证明被翻译成该类型的一个居留元。根据全称量词“对于每个”的解释，对应于引理 2.1.1的类型是

$$\prod_{(A:\mathcal{U})} \prod_{(x,y:A)} (x = y) \to (y = x).$$

引理 2.1.1的证明将包括构造这个类型的一个元素，即为某个 $f$ 导出判断 $f : \prod_{(A:\mathcal{U})} \prod_{(x,y:A)} (x = y) \to (y = x)$。然后我们为这个元素 $f$ 引入记号 $(-)^{-1}$，其中参数 $A$、$x$ 和 $y$ 被省略并从上下文推断。（如第 1.1 节中所述，次要陈述“对每个 $x : A$ 有 $\mathsf{refl}_x{}^{-1} \equiv \mathsf{refl}_x$”应被视为一个单独的判断。）

*第一个证明.* 假设给定 $A : \mathcal{U}$，并令 $D : \prod_{(x,y:A)} (x = y) \to \mathcal{U}$ 为由 $D(x, y, p) :\equiv (y = x)$ 定义的类型族。换句话说，$D$ 是一个函数，它把任意 $x, y : A$ 和 $p : x = y$ 映射到一个类型，即类型 $y = x$。那么我们有一个元素

$$d :\equiv \lambda x.\, \mathsf{refl}_x : \prod_{x:A} D(x, x, \mathsf{refl}_x).$$

因此，恒等类型的归纳原理给出一个元素

$$\mathsf{ind}_{=_A}(D, d, x, y, p) : (y = x)$$

对每个 $p : (x = y)$。我们现在可以把所需的函数 $(-)^{-1}$ 定义为 $\lambda p.\, \mathsf{ind}_{=_A}(D, d, x, y, p)$，即我们设 $p^{-1} :\equiv \mathsf{ind}_{=_A}(D, d, x, y, p)$。转换规则 (2.0.1) 给出 $\mathsf{refl}_x{}^{-1} \equiv \mathsf{refl}_x$，如所要求的。 □

我们以一种非常形式的风格写出了这个证明，当恒等类型的归纳规则还不熟悉时这可能会有帮助。要更加形式化，我们可以说引理 2.1.1及其证明一起包含判断

$$\lambda A.\, \lambda x.\, \lambda y.\, \lambda p.\, \mathsf{ind}_{=_A}((\lambda x.\, \lambda y.\, \lambda p.\, (y = x)), (\lambda x.\, \mathsf{refl}_x), x, y, p) : \prod_{(A:\mathcal{U})} \prod_{(x,y:A)} (x = y) \to (y = x)$$

（连同一个额外的等式判断）。然而，我们最终更喜欢使用更自然的语言，如下面的等价证明。

*第二个证明.* 我们想要对每个 $x, y : A$ 和 $p : x = y$ 构造一个元素 $p^{-1} : y = x$。由归纳，只需在 $y$ 是 $x$ 且 $p$ 是 $\mathsf{refl}_x$ 的情形进行。但在这种情形下，$p$ 的类型 $x = y$ 和我们试图构造 $p^{-1}$ 的类型 $y = x$ 都只是 $x = x$。因此，在“自反性情形”中，我们可以简单地把 $\mathsf{refl}_x{}^{-1}$ 定义为 $\mathsf{refl}_x$。一般情形然后由归纳原理得出，而转换规则 $\mathsf{refl}_x{}^{-1} \equiv \mathsf{refl}_x$ 正是我们在自反性情形给出的证明。 □

我们将用两种风格写出接下来的几个证明，以帮助读者习惯后一种。接下来我们证明相等的传递性，或者说等价地“连接路径”。

**Lemma 2.1.2.** *对于每个类型 $A$ 和每个 $x, y, z : A$，存在一个函数*

$$(x = y) \to (y = z) \to (x = z),$$

*记作 $p \mapsto q \mapsto p \cdot q$，使得对任意 $x : A$ 有 $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$。我们称 $p \cdot q$ 为 $p$ 和 $q$ 的*连接*或*复合*。*

注意我们选择用与函数复合相反的顺序来记路径连接：从 $p : x = y$ 和 $q : y = z$ 我们得到 $p \cdot q : x = z$，而从 $f : A \to B$ 和 $g : B \to C$ 我们得到 $g \circ f : A \to C$（见??）。

第一个证明. 所需的函数有类型 $\prod_{(x,y,z:A)}(x = y) \to (y = z) \to (x = z)$。我们将改为定义一个具有等价类型 $\prod_{(x,y:A)}(x = y) \to \prod_{(z:A)}(y = z) \to (x = z)$ 的函数，这允许我们两次应用路径归纳。令 $D : \prod_{(x,y:A)}(x = y) \to \mathcal{U}$ 为类型族

$$D(x, y, p) :\equiv \prod_{(z:A)} \prod_{(q:y=z)} (x = z).$$

注意 $D(x, x, \mathsf{refl}_x) \equiv \prod_{(z:A)} \prod_{(q:x=z)}(x = z)$。因此，为了把恒等类型的归纳原理应用到这个 $D$，我们需要一个类型为

$$\prod_{x:A} D(x, x, \mathsf{refl}_x) \tag{2.1.3}$$

的函数，也就是说，类型为

$$\prod_{(x,z:A)} \prod_{(q:x=z)} (x = z).$$

现在令 $E : \prod_{(x,z:A)} \prod_{(q:x=z)} \mathcal{U}$ 为类型族 $E(x, z, q) :\equiv (x = z)$。注意 $E(x, x, \mathsf{refl}_x) \equiv (x = x)$。因此，我们有函数

$$e(x) :\equiv \mathsf{refl}_x : E(x, x, \mathsf{refl}_x).$$

通过将恒等类型的归纳原理应用于 $E$，我们得到一个函数

$$d : \prod_{(x,z:A)} \prod_{(q:x=z)} E(x, z, q).$$

但 $E(x, z, q) \equiv (x = z)$，所以 $d$ 的类型是 (2.1.3)。因此，我们可以使用这个函数 $d$ 并把恒等类型的归纳原理应用于 $D$，以获得我们所需的类型为

$$\prod_{x,y:A} (x = y) \to \prod_{z:A} (y = z) \to (x = z)$$

的函数，从而得到 $\prod_{(x,y,z:A)}(y = z) \to (x = y) \to (x = z)$。两个归纳原理的转换规则给出对任意 $x : A$ 有 $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$。 □

第二个证明. 我们想要对每个 $x, y, z : A$ 和每个 $p : x = y$ 和 $q : y = z$ 构造一个 $x = z$ 的元素。对 $p$ 进行归纳，只需假设 $y$ 是 $x$ 且 $p$ 是 $\mathsf{refl}_x$。在这种情形下，$q$ 的类型 $y = z$ 是 $x = z$。现在对 $q$ 进行归纳，只需也假设 $z$ 是 $x$ 且 $q$ 是 $\mathsf{refl}_x$。但在这种情形下，$x = z$ 是 $x = x$，而我们有 $\mathsf{refl}_x : (x = x)$。 □

读者可能觉得我们给出了这个引理的一个过于迂回的证明。事实上，我们可以在对 $p$ 进行归纳后就停下来，因为那时我们想要产生的是一个相等 $x = z$，而我们已经有了这样一个相等，即 $q$。为什么我们还要继续对 $q$ 进行另一次归纳？

答案是，如导论中所述，我们在做证明相关的数学。当我们证明一个引理时，我们在定义某个类型的一个居留元，而在证明过程中我们定义的具体元素可能很重要，不仅仅是该元素所居留的类型（即引理的陈述）。引理 2.1.2有三个明显的证明：我们可以对 $p$ 进行归纳，对 $q$ 进行归纳，或对它们两个都进行归纳。如果我们用三种不同的方式证明它，我们将有同一类型的三个不同元素。不难证明这三个元素相等（见??），但由于它们不是判断上相等的，仍然可能有理由偏好其中一个。

在引理 2.1.2的情形中，差异取决于计算规则。如果我们用对 $p$ 的单次归纳来证明引理，那么我们最终会得到形式为 $\mathsf{refl}_y \cdot q \equiv q$ 的计算规则。如果我们用对 $q$ 的单次归纳来证明它，我们将得到 $p \cdot \mathsf{refl}_y \equiv p$，而用双重归纳来证明它（如我们所做的）只给出 $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$。

在做形式化数学时，不对称的计算规则有时会很方便，因为它们允许计算机自动简化更多的东西。然而，在非形式数学中，甚至可以说在形式化的情况下，有一个行为不对称的连接运算并且必须记住哪一边是“特殊的”可能会令人困惑。对称地处理两边使证明更健壮；这就是为什么我们给出了这样的证明。（然而，这无可否认是一个风格选择。）

下表总结了我们迄今所做的事情的“相等”、“同伦”和“高阶群胚”观点。

| 相等 | 同伦 | $\infty$-群胚 |
|---|---|---|
| 自反性 | 常路径 | 恒等态射 |
| 对称性 | 路径的逆 | 逆态射 |
| 传递性 | 路径的连接 | 态射的复合 |

在实践中，传递性经常通过一系列中间步骤来证明相等。我们将使用常见的记号，如 $a = b = c = d$。如果中间表达式很长，或者我们想要指定每个相等的见证，我们可以写

$$\begin{aligned} a &= b && \text{（由 } p\text{）} \\ &= c && \text{（由 } q\text{）} \\ &= d && \text{（由 } r\text{）} \end{aligned}$$

在任一情况下，该记号表示构造元素 $(p \cdot q) \cdot r : (a = d)$。（我们为具体性选择左结合，尽管鉴于下面定理 2.1.4(iv)这几乎没有什么区别。）如果碰巧，比如说，$b$ 和 $c$ 判断上相等，那么我们可以写

$$\begin{aligned} a &= b && \text{（由 } p\text{）} \\ &\equiv c \\ &= d && \text{（由 } r\text{）} \end{aligned}$$

来表示构造 $p \cdot r : (a = d)$。我们也遵循常见的数学实践，不要求这种记号中的理由（“由 $p$” 和 “由 $r$”）提供所需的确切见证；相反，我们允许它们只是提及构造该见证的最重要（或最不明显）的成分。例如，如果 “引理 A” 陈述对所有 $x$ 和 $y$ 我们有 $f(x) = g(y)$，那么我们可以写 “由引理 A” 作为步骤 $f(a) = g(b)$ 的理由，相信读者会推断我们以 $x :\equiv a$ 和 $y :\equiv b$ 应用引理 A。如果我们相信读者能够猜出理由，我们也可以完全省略它。

现在，由于证明相关性，我们不能在证明了相等的 “对称性” 和 “传递性” 之后就停下来：我们需要知道这些相等上的*运算*是良好的。（这个问题在集合论中是看不见的，在那里对称性和传递性只是相等的*性质*，而不是路径上的结构。）从同伦论的观点来看，连接和取逆只是高阶群胚结构的 “第一层”——我们还需要这些运算上的相容性定律，以及更高维度的类似运算。例如，我们需要知道连接是*结合的*，以及取逆关于连接提供*逆元*。

**Lemma 2.1.4.** 假设 $A:\mathcal{U}$，$x,y,z,w:A$，$p:x=y$，$q:y=z$，$r:z=w$。我们有以下：

*(i)* $p=p\cdot \mathsf{refl}_y$ 且 $p=\mathsf{refl}_x\cdot p$。

*(ii)* $p^{-1}\cdot p=\mathsf{refl}_y$ 且 $p\cdot p^{-1}=\mathsf{refl}_x$。

*(iii)* $(p^{-1})^{-1}=p$。

*(iv)* $p\cdot(q\cdot r)=(p\cdot q)\cdot r$。

特别注意，(i)–(iv)本身是命题相等，居留在恒等类型的恒等类型中，如对 $p,q:x=y$ 有 $p=_{x=y}q$。拓扑地，它们是路径的路径，即同伦。在拓扑学中有一个熟悉的事实，当我们把路径 $p$ 与逆路径 $p^{-1}$ 连接时，我们并不真正得到一个常路径（对应于类型论中的相等 $\mathsf{refl}$）——相反，我们有一个从 $p\cdot p^{-1}$ 到常路径的同伦，或更高路径。

定理 *2.1.4*的证明. 所有证明都使用相等的归纳原理。

(i) 第一个证明：令 $D:\prod_{(x,y:A)}(x=y)\to\mathcal{U}$ 为由

$$D(x,y,p):\equiv(p=p\cdot\mathsf{refl}_y)$$

给出的类型族。那么 $D(x,x,\mathsf{refl}_x)$ 是 $\mathsf{refl}_x=\mathsf{refl}_x\cdot\mathsf{refl}_x$。由于 $\mathsf{refl}_x\cdot\mathsf{refl}_x\equiv\mathsf{refl}_x$，可得 $D(x,x,\mathsf{refl}_x)\equiv(\mathsf{refl}_x=\mathsf{refl}_x)$。因此，有一个函数

$$d:\equiv\lambda x.\,\mathsf{refl}_{\mathsf{refl}_x}:\prod_{x:A}D(x,x,\mathsf{refl}_x).$$

现在恒等类型的归纳原理给出一个元素 $\mathsf{ind}_{=_A}(D,d,x,y,p):(p=p\cdot\mathsf{refl}_y)$，对每个 $p:x=y$。另一个相等类似地证明。

第二个证明：对 $p$ 进行归纳，只需假设 $y$ 是 $x$ 且 $p$ 是 $\mathsf{refl}_x$。但在这种情形下，我们有 $\mathsf{refl}_x\cdot\mathsf{refl}_x\equiv\mathsf{refl}_x$。

(ii) 第一个证明：令 $D:\prod_{(x,y:A)}(x=y)\to\mathcal{U}$ 为由

$$D(x,y,p):\equiv(p^{-1}\cdot p=\mathsf{refl}_y)$$

给出的类型族。那么 $D(x,x,\mathsf{refl}_x)$ 是 ${\mathsf{refl}_x}^{-1} \cdot \mathsf{refl}_x = \mathsf{refl}_x$。由于 ${\mathsf{refl}_x}^{-1} \equiv \mathsf{refl}_x$ 且 $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$，我们得到 $D(x,x,\mathsf{refl}_x) \equiv (\mathsf{refl}_x = \mathsf{refl}_x)$。因此我们找到函数

$$d :\equiv \lambda x.\, \mathsf{refl}_{\mathsf{refl}_x} : \prod_{x:A} D(x,x,\mathsf{refl}_x).$$

现在路径归纳给出一个元素 $\mathsf{ind}_{=_A}(D,d,x,y,p) : p^{-1} \cdot p = \mathsf{refl}_y$，对 $A$ 中每个 $p : x = y$。另一个相等类似。

第二个证明：由归纳，只需假设 $p$ 是 $\mathsf{refl}_x$。但在这种情形下，我们有 $p^{-1} \cdot p \equiv {\mathsf{refl}_x}^{-1} \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$。

(iii) 第一个证明：令 $D : \prod_{(x,y:A)} (x = y) \to \mathcal{U}$ 为由

$$D(x,y,p) :\equiv ((p^{-1})^{-1} = p)$$

给出的类型族。那么 $D(x,x,\mathsf{refl}_x)$ 是类型 $(({\mathsf{refl}_x}^{-1})^{-1} = \mathsf{refl}_x)$。但由于对每个 $x : A$ 有 ${\mathsf{refl}_x}^{-1} \equiv \mathsf{refl}_x$，我们有 $({\mathsf{refl}_x}^{-1})^{-1} \equiv {\mathsf{refl}_x}^{-1} \equiv \mathsf{refl}_x$，因此 $D(x,x,\mathsf{refl}_x) \equiv (\mathsf{refl}_x = \mathsf{refl}_x)$。因此我们找到函数

$$d :\equiv \lambda x.\, \mathsf{refl}_{\mathsf{refl}_x} : \prod_{x:A} D(x,x,\mathsf{refl}_x).$$

现在路径归纳给出一个元素 $\mathsf{ind}_{=_A}(D,d,x,y,p) : (p^{-1})^{-1} = p$，对每个 $p : x = y$。

第二个证明：由归纳，只需假设 $p$ 是 $\mathsf{refl}_x$。但在这种情形下，我们有 $(p^{-1})^{-1} \equiv ({\mathsf{refl}_x}^{-1})^{-1} \equiv \mathsf{refl}_x$。

(iv) 第一个证明：令 $D_1 : \prod_{(x,y:A)} (x = y) \to \mathcal{U}$ 为由

$$D_1(x,y,p) :\equiv \prod_{(z,w:A)} \prod_{(q:y=z)} \prod_{(r:z=w)} \big(p \cdot (q \cdot r) = (p \cdot q) \cdot r\big)$$

给出的类型族。那么 $D_1(x,x,\mathsf{refl}_x)$ 是

$$\prod_{(z,w:A)} \prod_{(q:x=z)} \prod_{(r:z=w)} \big(\mathsf{refl}_x \cdot (q \cdot r) = (\mathsf{refl}_x \cdot q) \cdot r\big).$$

为了构造这个类型的元素，令 $D_2 : \prod_{(x,z:A)}(x = z) \to \mathcal{U}$ 为类型族

$$D_2(x, z, q) :\equiv \prod_{(w:A)} \prod_{(r:z=w)} \big(\mathsf{refl}_x \cdot (q \cdot r) = (\mathsf{refl}_x \cdot q) \cdot r\big).$$

那么 $D_2(x, x, \mathsf{refl}_x)$ 是

$$\prod_{(w:A)} \prod_{(r:x=w)} \big(\mathsf{refl}_x \cdot (\mathsf{refl}_x \cdot r) = (\mathsf{refl}_x \cdot \mathsf{refl}_x) \cdot r\big).$$

为了构造这个类型的元素，令 $D_3 : \prod_{(x,w:A)}(x = w) \to \mathcal{U}$ 为类型族

$$D_3(x, w, r) :\equiv \big(\mathsf{refl}_x \cdot (\mathsf{refl}_x \cdot r) = (\mathsf{refl}_x \cdot \mathsf{refl}_x) \cdot r\big).$$

那么 $D_3(x, x, \mathsf{refl}_x)$ 是

$$\big(\mathsf{refl}_x \cdot (\mathsf{refl}_x \cdot \mathsf{refl}_x) = (\mathsf{refl}_x \cdot \mathsf{refl}_x) \cdot \mathsf{refl}_x\big)$$

它判断上等于类型 $(\mathsf{refl}_x = \mathsf{refl}_x)$，因此由 $\mathsf{refl}_{\mathsf{refl}_x}$ 居留。因此，三次应用路径归纳规则，我们得到总体所需类型的一个元素。

第二个证明：由归纳，只需假设 $p$、$q$ 和 $r$ 都是 $\mathsf{refl}_x$。但在这种情形下，我们有

$$\begin{aligned} p \cdot (q \cdot r) &\equiv \mathsf{refl}_x \cdot (\mathsf{refl}_x \cdot \mathsf{refl}_x) \\ &\equiv \mathsf{refl}_x \\ &\equiv (\mathsf{refl}_x \cdot \mathsf{refl}_x) \cdot \mathsf{refl}_x \\ &\equiv (p \cdot q) \cdot r. \end{aligned}$$

因此，我们有 $\mathsf{refl}_{\mathsf{refl}_x}$ 居留这个类型。 □

*Remark* 2.1.5. 有其他方法来定义这些更高路径。例如，在定理 2.1.4(iv)中我们可以只对一条或两条路径进行归纳而不是全部三条。每种可能性将产生一个判断上不同的证明，但它们都将彼此相等。任何两个特定证明之间的这种相等又可以用归纳来证明，将所有相关路径归约到自反性，然后观察两个证明都把自己归约到自反性。

鉴于定理 2.1.4(iv)，我们将经常把 $p \cdot q \cdot r$ 写成 $(p \cdot q) \cdot r$，类似地把 $p \cdot q \cdot r \cdot s$ 写成 $((p \cdot q) \cdot r) \cdot s$，依此类推。我们为确定性选择左结合，但这没有实际区别。我们一般相信读者会在必要时插入定理 2.1.4(iv)的实例来重新结合这样的表达式。

我们实际上还没有真正完成高阶群胚结构：路径 (i)–(iv)也必须满足它们自己的更高相容性定律，这些定律本身是更高路径，依此类推 “一直到无穷大”（这可以用例如球状操纵子的概念来精确化）。然而，对于大多数目的来说，没有必要使整个无限维结构显式化。同伦类型论的好处之一是，所有这些结构都可以仅从恒等类型的归纳性质证明出来，因此我们可以根据需要使其中的多或少显式化。

特别地，在本书中我们不需要任何涉及精确化诸如 “所有更高层次的相容结构” 这样概念的复杂组合学。除了普通路径，我们将使用路径的路径（即类型 $p =_{x=_A y} q$ 的元素，对 $p, q : x = y$），如前所述我们称之为 *2-路径*或*二维路径*，也许偶尔使用路径的路径的路径（即类型 $r =_{p=_{x=_A y} q} s$ 的元素），我们称之为 *3-路径*或*三维路径*。可以定义*n-维路径* 的一般概念（见??），但我们不需要它。

然而，我们将使用一个特别重要和简单的高阶路径的情形，就是起点和终点相同的情况。在集合论中，命题 $a = a$ 是完全无趣的，但在同伦论中，从一个点到自身的路径称为*环路*，携带着许多有趣的高阶结构。因此，给定一个带有点 $a : A$ 的类型 $A$，我们定义它的**环路空间** $\Omega(A, a)$ 为类型 $a =_A a$。如果点 $a$ 从上下文中可以理解，我们有时可以简单地写 $\Omega A$。

由于 $\Omega A$ 的任意两个元素都是起点和终点相同的路径，它们可以被连接；因此我们有一个运算 $\Omega A \times \Omega A \to \Omega A$。更一般地，$A$ 的高阶群胚结构赋予 $\Omega A$ 类似的 “高阶群” 结构。

考虑 $A$ 的环路空间的环路空间也很有用，它是在 $a$ 处恒等环路上的二维环路的空间。这被写作 $\Omega^2(A, a)$，在类型论中由类型 $\mathsf{refl}_a =_{(a=_A a)} \mathsf{refl}_a$ 表示。虽然 $\Omega^2(A, a)$ 作为环路空间仍然是一个 “高阶群”，但它现在还有一些额外的结构，来自于其元素是一维环路之间的二维环路这一事实。

**Theorem 2.1.6** (Eckmann–Hilton)**.** *第二环路空间上的复合运算*

$$\Omega^2(A) \times \Omega^2(A) \to \Omega^2(A)$$

是交换的：对任意 $\alpha, \beta : \Omega^2(A)$ 有 $\alpha \cdot \beta = \beta \cdot \alpha$。

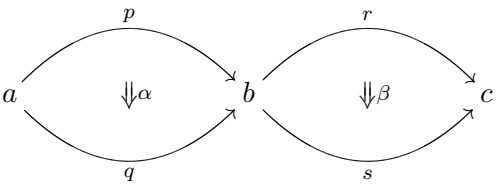
证明. 首先，观察 1-环路的复合 $\Omega A \times \Omega A \to \Omega A$ 诱导一个运算

$$\star : \Omega^2(A) \times \Omega^2(A) \to \Omega^2(A)$$

如下：考虑元素 $a, b, c : A$ 和 1- 与 2-路径，

$$\begin{aligned} p &: a = b, & r &: b = c \\ q &: a = b, & s &: b = c \\ \alpha &: p = q, & \beta &: r = s \end{aligned}$$

如下图所示（路径画成箭头）。




分别复合上面和下面的 1-路径，我们得到两条路径 $p \cdot r,\ q \cdot s : a = c$，然后它们之间有一个 “水平复合”

$$\alpha \star \beta : p \cdot r = q \cdot s$$

定义如下。首先，我们通过对 $r$ 进行路径归纳来定义 $\alpha \cdot_{\mathsf{r}} r : p \cdot r = q \cdot r$，使得

$$\alpha \cdot_{\mathsf{r}} \mathsf{refl}_b \equiv \mathsf{ru}_p^{-1} \cdot \alpha \cdot \mathsf{ru}_q$$

其中 $\mathsf{ru}_p : p = p \cdot \mathsf{refl}_b$ 是定理 2.1.4(i)中的右单位律。我们也可以类似地通过对 $\alpha$ 进行归纳，或对所有可见路径进行归纳来定义 $\cdot_{\mathsf{r}}$，导致不同的判断相等，但对于当前目的，通过对 $r$ 进行归纳的定义会使事情更简单。类似地，我们通过对 $q$ 进行归纳来定义 $q \cdot_{\mathsf{l}} \beta : q \cdot r = q \cdot s$，使得

$$\mathsf{refl}_b \cdot_{\mathsf{l}} \beta \equiv \mathsf{lu}_r^{-1} \cdot \beta \cdot \mathsf{lu}_s$$

其中 $\mathsf{lu}_r$ 表示左单位律。运算 $\blacktriangleright_{\mathsf{l}}$ 和 $\blacktriangleright_{\mathsf{r}}$ 称为**须化**。接下来，由于 $\alpha \blacktriangleright_{\mathsf{r}} r$ 和 $q \blacktriangleright_{\mathsf{l}} \beta$ 是可复合的 2-路径，我们可以定义**水平复合** 为：

$$\alpha \star \beta \;:\equiv\; (\alpha \blacktriangleright_{\mathsf{r}} r) \blacktriangleright (q \blacktriangleright_{\mathsf{l}} \beta).$$

现在假设 $a \equiv b \equiv c$，使得所有 1-路径 $p$、$q$、$r$ 和 $s$ 都是 $\Omega(A, a)$ 的元素，并进一步假设 $p \equiv q \equiv r \equiv s \equiv \mathsf{refl}_a$，使得 $\alpha : \mathsf{refl}_a = \mathsf{refl}_a$ 和 $\beta : \mathsf{refl}_a = \mathsf{refl}_a$ 以两种顺序都可复合。在那种情况下，我们有

$$\begin{aligned}
\alpha \star \beta &\equiv (\alpha \blacktriangleright_{\mathsf{r}} \mathsf{refl}_a) \blacktriangleright (\mathsf{refl}_a \blacktriangleright_{\mathsf{l}} \beta) \\
&= {\mathsf{ru}_{\mathsf{refl}_a}}^{-1} \blacktriangleright \alpha \blacktriangleright \mathsf{ru}_{\mathsf{refl}_a} \blacktriangleright {\mathsf{lu}_{\mathsf{refl}_a}}^{-1} \blacktriangleright \beta \blacktriangleright \mathsf{lu}_{\mathsf{refl}_a} \\
&\equiv {\mathsf{refl}_{\mathsf{refl}_a}}^{-1} \blacktriangleright \alpha \blacktriangleright \mathsf{refl}_{\mathsf{refl}_a} \blacktriangleright {\mathsf{refl}_{\mathsf{refl}_a}}^{-1} \blacktriangleright \beta \blacktriangleright \mathsf{refl}_{\mathsf{refl}_a} \\
&= \alpha \blacktriangleright \beta.
\end{aligned}$$

（回忆 $\mathsf{ru}_{\mathsf{refl}_a} \equiv \mathsf{lu}_{\mathsf{refl}_a} \equiv \mathsf{refl}_{\mathsf{refl}_a}$，由路径归纳的计算规则。）另一方面，我们可以类似地定义另一个水平复合

$$\alpha \star' \beta \;:\equiv\; (p \blacktriangleright_{\mathsf{l}} \beta) \blacktriangleright (\alpha \blacktriangleright_{\mathsf{r}} s)$$

而我们类似地学到

$$\alpha \star' \beta = \beta \blacktriangleright \alpha.$$

但一般来说，定义水平复合的两种方式一致，$\alpha \star \beta = \alpha \star' \beta$，正如我们通过对 $\alpha$ 和 $\beta$ 进行归纳然后对剩余的两条 1-路径进行归纳所能看到的那样，把一切都归约到自反性。因此我们有

$$\alpha \blacktriangleright \beta = \alpha \star \beta = \alpha \star' \beta = \beta \blacktriangleright \alpha. \qquad \square$$

上述事实被称为 *Eckmann–Hilton* 论证，来自经典同伦论，实际上它在下面第 8 章中被用来证明一个类型的高阶同伦群总是阿贝尔群。证明中定义的须化和水平复合运算也是类型的 $\infty$-群胚结构的一般部分。它们满足自己的定律（到更高同伦为止），如

$$\alpha \blacktriangleright_{\mathsf{r}} (p \blacktriangleright q) = (\alpha \blacktriangleright_{\mathsf{r}} p) \blacktriangleright_{\mathsf{r}} q$$

等等。从现在开始，我们相信读者会在需要时应用路径归纳来定义这类进一步的运算并验证它们的性质。

正如这个例子所暗示的，高阶路径类型的代数比每个层次上的类群胚结构要复杂得多；这些层次相互作用产生了许多进一步的运算和定律，就像同伦论中对迭代环路空间的研究一样。事实上，如在经典同伦论中，我们可以做以下一般定义：

**Definition 2.1.7.** 一个**有点类型** $(A, a)$ 是一个类型 $A : \mathcal{U}$ 连同一个点 $a : A$，称为它的**基点**。我们写 $\mathcal{U}_\bullet :\equiv \sum_{(A:\mathcal{U})} A$ 表示宇宙 $\mathcal{U}$ 中有点类型的类型。

**Definition 2.1.8.** 给定一个有点类型 $(A, a)$，我们定义 $(A, a)$ 的**环路空间**为以下有点类型：

$$\Omega(A, a) :\equiv ((a =_A a), \mathsf{refl}_a).$$

它的元素将被称为 $a$ 处的**环路**。对于 $n : \mathbb{N}$，有点类型 $(A, a)$ 的 $n$-**重迭代环路空间** $\Omega^n(A, a)$ 递归地定义为：

$$\Omega^0(A, a) :\equiv (A, a)$$
$$\Omega^{n+1}(A, a) :\equiv \Omega^n(\Omega(A, a)).$$

它的元素将被称为 $a$ 处的 $n$-**环路** 或 $n$-**维环路**。

我们将在第 7 章 和第 6 章 和第 8 章中回到迭代环路空间。

## 2.2 函数是函子

现在我们希望建立函数 $f : A \to B$ 在路径上函子性地行为。在传统类型论中，这等价于陈述函数尊重相等。拓扑地，这对应于说每个函数都是“连续的”，即保持路径。

**Lemma 2.2.1.** 假设 $f : A \to B$ 是一个函数。那么对任意 $x, y : A$ 有一个运算

$$\mathsf{ap}_f : (x =_A y) \to (f(x) =_B f(y)).$$

此外，对每个 $x : A$ 我们有 $\mathsf{ap}_f(\mathsf{refl}_x) \equiv \mathsf{refl}_{f(x)}$。

记号 $\mathsf{ap}_f$ 可以读作 $f$ 对路径的应用（application），或者 $f$ 在路径上的作用（action on paths）。

第一个证明. 令 $D : \prod_{(x,y:A)}(x = y) \to \mathcal{U}$ 为由

$$D(x, y, p) :\equiv (f(x) = f(y))$$

定义的类型族。那么我们有

$$d :\equiv \lambda x.\, \mathsf{refl}_{f(x)} : \prod_{x:A} D(x, x, \mathsf{refl}_x).$$

通过路径归纳，我们得到 $\mathsf{ap}_f : \prod_{(x,y:A)}(x = y) \to (f(x) = f(y))$。计算规则蕴含对每个 $x : A$ 有 $\mathsf{ap}_f(\mathsf{refl}_x) \equiv \mathsf{refl}_{f(x)}$。 □

第二个证明. 要为所有 $p : x = y$ 定义 $\mathsf{ap}_f(p)$，由归纳只需假设 $p$ 是 $\mathsf{refl}_x$。在这种情形下，我们可以定义 $\mathsf{ap}_f(p) :\equiv \mathsf{refl}_{f(x)} : f(x) = f(x)$。 □

我们经常把 $\mathsf{ap}_f(p)$ 简单地写成 $f(p)$。严格来说这是有歧义的，但一般不会引起混淆。它与范畴论中使用同一符号表示函子对对象和态射的应用的常见约定相匹配。

我们注意到 $\mathsf{ap}$ 以人们可能期望的所有方式函子性地行为。

**Lemma 2.2.2.** *对于函数 $f : A \to B$ 和 $g : B \to C$ 以及路径 $p : x =_A y$ 和 $q : y =_A z$，我们有：*

*(i)* $\mathsf{ap}_f(p \cdot q) = \mathsf{ap}_f(p) \cdot \mathsf{ap}_f(q)$。

*(ii)* $\mathsf{ap}_f(p^{-1}) = \mathsf{ap}_f(p)^{-1}$。

*(iii)* $\mathsf{ap}_g(\mathsf{ap}_f(p)) = \mathsf{ap}_{g\circ f}(p)$。

*(iv)* $\mathsf{ap}_{\mathsf{id}_A}(p) = p$。

证明. 留给读者。 □

正如定理 2.1.4中的相等那样，引理 2.2.2中的那些本身是路径，它们满足自己的相容性定律（可以用同样的方式证明），依此类推。

## 2.3 类型族是纤维化

由于依赖类型函数在类型论中是必不可少的，我们还需要引理 2.2.1的一个版本来处理这些。然而,这不是那么简单地陈述,因为如果 $f : \prod_{(x:A)} B(x)$ 且 $p : x = y$，那么 $f(x) : B(x)$ 和 $f(y) : B(y)$ 是不同类型的元素，所以先验地我们甚至不能问它们是否相等。缺失的成分是 $p$ 本身给了我们一种关联类型 $B(x)$ 和 $B(y)$ 的方法。

我们在第 1.12 节中已经见过这个，在那里我们称之为 “相同者的不可区分性”。我们现在为它引入一个不同的名字和记号，从现在开始我们将使用它。

**Lemma 2.3.1** (传输). 假设 $P$ 是 $A$ 上的类型族，$p : x =_A y$。那么有一个函数 $p_* : P(x) \to P(y)$。

第一个证明. 令 $D : \prod_{(x,y:A)}(x = y) \to \mathcal{U}$ 为由

$$D(x, y, p) :\equiv P(x) \to P(y)$$

定义的类型族。那么我们有函数

$$d :\equiv \lambda x.\, \mathsf{id}_{P(x)} : \prod_{x:A} D(x, x, \mathsf{refl}_x),$$

从而归纳原理给我们 $\mathsf{ind}_{=_A}(D, d, x, y, p) : P(x) \to P(y)$，对 $p : x = y$，我们定义它为 $p_*$。 □

第二个证明. 由归纳，只需假设 $p$ 是 $\mathsf{refl}_x$。但在这种情形下，我们可以取 $(\mathsf{refl}_x)_* : P(x) \to P(x)$ 为恒等函数。 □

有时候，需要记下传输运算发生的类型族 $P$。在这种情况下，我们可以写

$$\mathsf{transport}^P(p, -) : P(x) \to P(y).$$

回忆 $A$ 上的类型族 $P$ 可以被看作 $A$ 的元素的性质，当 $P(x)$ 有居留元时在 $x$ 处成立。那么传输引理说 $P$ 尊重相等，在以下意义上：如果 $x$ 等于 $y$，那么 $P(x)$ 成立当且仅当 $P(y)$ 成立。事实上，我们稍后将看到如果 $x = y$ 那么实际上 $P(x)$ 和 $P(y)$ 是等价的。

拓扑地，传输引理可以被看作纤维化中的“路径提升”运算。我们把类型族 $P : A \to \mathcal{U}$ 看作一个纤维化，底空间是 $A$，$P(x)$ 是 $x$ 上方的纤维，$\sum_{(x:A)} P(x)$ 是纤维化的**全空间**，带有第一投影 $\sum_{(x:A)}(P(x)) \to A$。纤维化的定义性质是，给定底空间 $A$ 中的路径 $p : x = y$ 和 $x$ 上方纤维中的点 $u : P(x)$，我们可以把路径 $p$ 提升为全空间中从 $u$ 开始的路径（而且这种提升可以连续地进行）。点 $p_*(u)$ 可以被视为这条提升路径的另一个端点。我们也可以在类型论中定义路径本身：

**Lemma 2.3.2** (路径提升性质)**.** 给定类型族 $P : A \to \mathcal{U}$ 以及 $u : P(x)$ 和路径 $p : x = y$，我们有

$$\mathsf{lift}(u, p) : (x, u) =_{(\sum_{(x:A)} P(x))} (y, p_*(u)).$$

证明. 由归纳，只需假设 $p$ 是 $\mathsf{refl}_x$。在这种情形下，$p_*(u) \equiv u$ 且我们有 $\mathsf{refl}_{(x,u)} : (x, u) = (x, u)$。 □

注意提升路径 $\mathsf{lift}(u, p)$ 位于 $\Sigma$-类型 $\sum_{(x:A)} P(x)$ 中；因此它的投影是 $A$ 中的路径，我们期望它是 $p$：

**Lemma 2.3.3.** 对于 $u : P(x)$ 和 $p : x = y$，我们有

$$\mathsf{pr}_1(\mathsf{lift}(u, p)) = p.$$

证明. 由路径归纳留给读者。 □

现在我们可以陈述依赖类型函数的“函子性”版本。

**Lemma 2.3.4.** 假设 $f : \prod_{(x:A)} P(x)$，那么对任意路径 $p : x = y$ 我们有

$$\mathsf{apd}_f(p) : p_*(f(x)) =_{P(y)} f(y).$$

证明. 由路径归纳。 □

名字 apd 代表依赖应用的“依赖 版本”（dependent ap）。注意当 $P$ 是常类型族时，$p_*(f(x))$ 等于 $f(x)$，因此 $\mathsf{apd}_f(p) : f(x) = f(y)$ 与 $\mathsf{ap}_f(p)$ 类型相同。在这种情况下，它们实际上相等：

**Lemma 2.3.5.** 当 $P : A \to \mathcal{U}$ 是常类型族 $P(x) :\equiv B$ 时，对 $f : A \to B$ 和 $p : x = y$ 我们有

$$\mathsf{apd}_f(p) = \mathsf{transportconst}_p^B(f(x)) \cdot \mathsf{ap}_f(p)$$

其中常类型族沿路径的传输是

$$\mathsf{transportconst}_p^B(b) : \mathsf{transport}^{(\lambda x.\, B)}(p, b) = b.$$

证明. 由路径归纳。 □

## 2.4 同伦与等价

到目前为止，我们已经看到恒等类型 $x =_A y$ 如何可以被视为两个元素 $x$ 和 $y$ 之间的等同、路径或等价的类型。现在我们研究函数之间和类型之间 “等同” 或 “相同” 的适当概念。在第 2.9 节 和第 2.10 节中，我们将看到同伦类型论允许我们把这些与恒等类型的实例等同起来，但在这样做之前我们需要独立地理解它们。

传统上，我们把两个函数视为相同的，如果它们在所有输入上取相等的值。在命题即类型的解释下，这建议两个函数 $f$ 和 $g$（可能是依赖类型的）应该相同，如果类型 $\prod_{(x:A)}(f(x) = g(x))$ 有居留元。在同伦解释下，这个依赖函数类型由连续路径或函子性等价组成，因此可以被视为同伦或自然同构的类型。我们将采用这方面的拓扑术语。

**Definition 2.4.1.** 设 $f, g : \prod_{(x:A)} P(x)$ 是类型族 $P : A \to \mathcal{U}$ 的两个截面。从 $f$ 到 $g$ 的**同伦**是类型

$$(f \sim g) :\equiv \prod_{x:A} (f(x) = g(x))$$

的依赖函数。

注意同伦与等同 $(f = g)$ 不一样。然而，在第 2.9 节中我们将引入一个公理使同伦和等同 “等价”。

以下证明留给读者。

**Lemma 2.4.2.** 同伦是每个依赖函数类型 $\prod_{(x:A)} P(x)$ 上的等价关系。即，我们有以下类型的元素

$$\prod_{f:\prod_{(x:A)} P(x)} (f \sim f)$$

$$\prod_{f,g:\prod_{(x:A)} P(x)} (f \sim g) \to (g \sim f)$$

$$\prod_{f,g,h:\prod_{(x:A)} P(x)} (f \sim g) \to (g \sim h) \to (f \sim h).$$

正如类型论中的函数自动是“函子”一样，同伦自动是“自然变换”。我们只对非依赖函数 $f, g : A \to B$ 陈述并证明这一点；在?? 中我们请读者把它推广到依赖函数。

回忆对于 $f : A \to B$ 和 $p : x =_A y$，我们可以写 $f(p)$ 来表示 $\mathsf{ap}_f(p)$。

**Lemma 2.4.3.** 假设 $H : f \sim g$ 是函数 $f, g : A \to B$ 之间的同伦，设 $p : x =_A y$。那么我们有

$$H(x) \cdot g(p) = f(p) \cdot H(y).$$

我们也可以把这画成一个交换图：

$$\begin{array}{ccc} f(x) & \overset{f(p)}{=\!=} & f(y) \\ {\scriptstyle H(x)}\Big\| & & \Big\|{\scriptstyle H(y)} \\ g(x) & \underset{g(p)}{=\!=} & g(y) \end{array}$$

证明. 由归纳，我们可以假设 $p$ 是 $\mathsf{refl}_x$。由于 $\mathsf{ap}_f$ 和 $\mathsf{ap}_g$ 在自反性上计算，在这种情况下我们必须证明的是

$$H(x) \cdot \mathsf{refl}_{g(x)} = \mathsf{refl}_{f(x)} \cdot H(x).$$

但这是成立的，因为两边都等于 $H(x)$。 □

**Corollary 2.4.4.** 设 $H : f \sim \mathsf{id}_A$ 是一个同伦，其中 $f : A \to A$。那么对任意 $x : A$ 我们有

$$H(f(x)) = f(H(x)).$$

这里 $f(x)$ 表示 $f$ 对 $x$ 的普通应用，而 $f(H(x))$ 表示 $\mathsf{ap}_f(H(x))$。

*证明.* 由 $H$ 的自然性，以下路径图交换：

$$\begin{array}{ccc} ffx & \xlongequal{f(Hx)} & fx \\ {\scriptstyle H(fx)}\Big\| & & \Big\|{\scriptstyle Hx} \\ fx & \xlongequal[Hx]{} & x \end{array}$$

即，$f(Hx) \cdot Hx = H(fx) \cdot Hx$。我们现在可以用 $(Hx)^{-1}$ 进行须化来消去 $Hx$，得到

$$f(Hx) = f(Hx) \cdot Hx \cdot (Hx)^{-1} = H(fx) \cdot Hx \cdot (Hx)^{-1} = H(fx)$$

如所要求的（省略了一些结合律路径）。 □

当然，像函数的函子性（引理 2.2.2）一样，引理 2.4.3中的等式是满足其自身相容性定律的路径，依此类推。

转向类型，从传统视角可以说函数 $f : A \to B$ 是一个**同构**，如果存在函数 $g : B \to A$ 使得复合 $f \circ g$ 和 $g \circ f$ 都逐点等于恒等，即使得 $f \circ g \sim \mathsf{id}_B$ 且 $g \circ f \sim \mathsf{id}_A$。从同伦视角来看这应该称为**同伦等价**，从范畴视角来看应该称为（高阶）*群胚的等价*。然而，当做证明相关的数学时，对应的类型

$$\sum_{g:B\to A} \big((f \circ g \sim \mathsf{id}_B) \times (g \circ f \sim \mathsf{id}_A)\big) \tag{2.4.5}$$

行为不好。例如，对于单个函数 $f : A \to B$，(2.4.5) 可能有多个不相等的居留元。（这与高阶范畴论中的观察密切相关，即通常需要考虑伴随等价而不是普通等价。）由于这个原因，我们给 (2.4.5) 以下历史上准确但听起来略带贬义的名字。

**Definition 2.4.6.** 对于函数 $f : A \to B$，$f$ 的**拟逆**是一个三元组 $(g, \alpha, \beta)$，由函数 $g : B \to A$ 和同伦 $\alpha : f \circ g \sim \mathsf{id}_B$ 及 $\beta : g \circ f \sim \mathsf{id}_A$ 组成。

因此，(2.4.5) 是 $f$ *的拟逆的类型*；我们可以把它记作 $\mathsf{qinv}(f)$。

*Example* 2.4.7. 恒等函数 $\mathsf{id}_A : A \to A$ 有一个拟逆，由 $\mathsf{id}_A$ 本身给出，连同由 $\alpha(y) :\equiv \mathsf{refl}_y$ 和 $\beta(x) :\equiv \mathsf{refl}_x$ 定义的同伦。

*Example* 2.4.8. 对任意 $p : x =_A y$ 和 $z : A$，函数

$$(p \centerdot -) : (y =_A z) \to (x =_A z) \qquad \text{和}$$
$$(- \centerdot p) : (z =_A x) \to (z =_A y)$$

有拟逆，分别由 $(p^{-1} \centerdot -)$ 和 $(- \centerdot p^{-1})$ 给出；见??。

*Example* 2.4.9. 对任意 $p : x =_A y$ 和 $P : A \to \mathcal{U}$，函数

$$\mathsf{transport}^P(p, -) : P(x) \to P(y)$$

有一个拟逆，由 $\mathsf{transport}^P(p^{-1}, -)$ 给出；这由?? 得出。

一般来说，我们只在类型 $A$ 和 $B$“行为像集合”（见第 3.1 节）的特殊情况下使用同构 这个词（以及类似的词如双射，和相关记号 $A \cong B$）。在这种情况下，类型 (2.4.5) 是没有问题的。我们将为改进的概念 $\mathsf{isequiv}(f)$ 保留等价这个词，它具有以下性质：

(i) 对每个 $f : A \to B$ 有一个函数 $\mathsf{qinv}(f) \to \mathsf{isequiv}(f)$。

(ii) 类似地，对每个 $f$ 我们有 $\mathsf{isequiv}(f) \to \mathsf{qinv}(f)$；因此两者逻辑等价（见第 1.11 节）。

(iii) 对任意两个居留元 $e_1, e_2 : \mathsf{isequiv}(f)$ 我们有 $e_1 = e_2$。

在第 4 章中我们将看到有许多不同的 $\mathsf{isequiv}(f)$ 定义满足这三个性质，但它们都是等价的。现在，为了使读者相信这样的东西存在，我们只提及最容易的定义：

$$\mathsf{isequiv}(f) :\equiv \Big(\sum_{g:B\to A} (f \circ g \sim \mathsf{id}_B)\Big) \times \Big(\sum_{h:B\to A} (h \circ f \sim \mathsf{id}_A)\Big). \quad (2.4.10)$$

我们现在可以为这个定义证明 (i)和 (ii)。函数 $\mathsf{qinv}(f) \to \mathsf{isequiv}(f)$ 容易定义，把 $(g, \alpha, \beta)$ 映到 $(g, \alpha, g, \beta)$。在另一方向，给定 $(g, \alpha, h, \beta)$，令 $\gamma$ 为复合同伦

$$g \overset{\beta}{\sim} h \circ f \circ g \overset{\alpha}{\sim} h,$$

意思是 $\gamma(x) :\equiv \beta(g(x))^{-1} \centerdot h(\alpha(x))$。现在定义 $\beta' : g \circ f \sim \mathsf{id}_A$ 为 $\beta'(x) :\equiv \gamma(f(x)) \centerdot \beta(x)$。那么 $(g, \alpha, \beta') : \mathsf{qinv}(f)$。

这个定义的性质 (iii)证明起来也不太难，但它需要识别笛卡尔积和依赖对类型的恒等类型，我们将在第 2.6 节 和第 2.7 节中讨论。因此，我们也推迟它；见第 4.3 节。此时，要带走的主要东西是有一个行为良好的类型，我们可以读作 "$f$ 是一个等价"，而且我们可以通过展示它的拟逆来证明 $f$ 是等价。在实践中，这是证明函数是等价的最常见方式。

按照证明相关的哲学，从 $A$ 到 $B$ 的等价被定义为函数 $f : A \to B$ 连同 $\mathsf{isequiv}(f)$ 的一个居留元，即它是等价的证明。我们写 $(A \simeq B)$ 表示从 $A$ 到 $B$ 的等价的类型，即类型

$$(A \simeq B) :\equiv \sum_{f:A\to B} \mathsf{isequiv}(f). \tag{2.4.11}$$

上面的性质 (iii)将确保如果两个等价作为函数相等（即底层的 $A \to B$ 的元素相等），那么它们作为等价也相等（见第 2.7 节）。因此，我们经常滥用记号并模糊等价与其底层函数之间的区别。例如，如果我们有函数 $f : A \to B$ 并且知道 $e : \mathsf{isequiv}(f)$，我们可以写 $f : A \simeq B$，而不是 $(f, e)$。或者反过来，如果我们有等价 $g : A \simeq B$，给定 $a : A$ 时我们可以写 $g(a)$，而不是 $(\mathsf{pr}_1 g)(a)$。

我们以如下观察作结：

**Lemma 2.4.12.** *类型等价是 $\mathcal{U}$ 上的等价关系。更具体地：*

*(i)* *对任意 $A$，恒等函数 $\mathsf{id}_A$ 是等价；因此 $A \simeq A$。*

*(ii)* *对任意 $f : A \simeq B$，我们有等价 $f^{-1} : B \simeq A$。*

*(iii)* *对任意 $f : A \simeq B$ 和 $g : B \simeq C$，我们有 $g \circ f : A \simeq C$。*

*证明.* 恒等函数显然是其自身的拟逆；因此它是等价。

如果 $f : A \to B$ 是等价，那么它有拟逆，比如说 $f^{-1} : B \to A$。那么 $f$ 也是 $f^{-1}$ 的拟逆，所以 $f^{-1}$ 是等价 $B \to A$。

最后，给定 $f : A \simeq B$ 和 $g : B \simeq C$ 带拟逆 $f^{-1}$ 和 $g^{-1}$，那么对任意 $a : A$ 我们有 $f^{-1}g^{-1}gfa = f^{-1}fa = a$，对任意 $c : C$ 我们有 $gff^{-1}g^{-1}c = gg^{-1}c = c$。因此 $f^{-1} \circ g^{-1}$ 是 $g \circ f$ 的拟逆，所以后者是等价。 □

## 2.5 类型构造子的高阶群胚结构

在第 1 章中，我们引入了许多构造新类型的方式：笛卡尔积、不交并、依赖积、依赖和等。在第 2.1 节 和第 2.2 节 和第 2.3 节中，我们看到同伦类型论中所有类型都像空间或高阶群胚那样行为。我们在本章剩余部分的目标是明确这种高阶结构在第 1 章中定义的特定类型情况下如何表现。

事实证明，对于许多类型 $A$，相等类型 $x =_A y$ 可以被刻画为，到等价为止，用构造 $A$ 的数据来表达。例如，如果 $A$ 是笛卡尔积 $B \times C$，且 $x \equiv (b, c)$ 和 $y \equiv (b', c')$，那么我们有等价

$$\big((b, c) = (b', c')\big) \simeq \big((b = b') \times (c = c')\big). \tag{2.5.1}$$

用更传统的语言说，两个有序对相等当且仅当它们的分量相等（但等价 (2.5.1) 说的比这更多）。恒等类型的高阶结构也可以用这些等价来表达；例如，对之间的两个相等的连接对应于成对的连接。

类似地，当类型族 $P : A \to \mathcal{U}$ 使用第 1 章中的类型构造规则逐纤维构建时，运算 $\mathsf{transport}^P(p, -)$ 可以被刻画为，到同伦为止，用进入 $P$ 的数据上的对应运算来表达。例如，如果 $P(x) \equiv B(x) \times C(x)$，那么我们有

$$\mathsf{transport}^P(p, (b, c)) = \big(\mathsf{transport}^B(p, b), \mathsf{transport}^C(p, c)\big).$$

最后，类型构造规则也是函子性的，如果函数 $f$ 从这种函子性构建，那么运算 $\mathsf{ap}_f$ 和 $\mathsf{apd}_f$ 可以基于进入 $f$ 的数据上的对应运算来计算。例如，如果 $g : B \to B'$ 和 $h : C \to C'$ 且我们定义 $f : B \times C \to B' \times C'$ 为 $f(b, c) :\equiv (g(b), h(c))$，那么模等价 (2.5.1)，我们可以把 $\mathsf{ap}_f$ 等同于“$(\mathsf{ap}_g, \mathsf{ap}_h)$”。

接下来的几节（第 2.6 节 至 第 2.13 节）将致力于为所有基本类型构造规则陈述并证明这类定理，每个基本类型构造子一节。这里我们遇到了当前可用类型论中某种明显的不足；正如后面章节将变得清晰的，如果这些恒等类型、传输等的刻画是判断上的等式，似乎会更方便和直观。然而，在第 1 章中呈现的理论中，恒等类型通过它们的归纳原理为所有类型统一定义，所以我们不能“重新定义”它们在不同类型是不同的东西。因此，本章要讨论的特定类型的刻画大部分是我们必须发现和证明的*定理*，如果可能的话。

实际上，第 1 章的类型论不足以证明两个类型构造子所需的定理：Π-类型和宇宙。由于这个原因，我们被迫在类型论中引入公理，以使那些“定理”成立。从类型论上讲，*公理*（参见第 1.1 节）是一个“原子”元素，被声明居留某个指定的类型，而没有规定其行为的规则，除了与它所居留的类型相关的规则。

Π-类型的公理（第 2.9 节）对类型论学家来说是熟悉的：它叫做*函数外延性*，（大致）陈述如果两个函数在第 2.4 节意义下同伦，那么它们相等。然而，宇宙的公理（第 2.10 节）是同伦类型论的新贡献，归功于 Voevodsky：它叫做*泛等公理*，（大致）陈述如果两个类型在第 2.4 节意义下等价，那么它们相等。我们已经在导论中评论过这个公理；它将在本书中扮演非常重要的角色。[1]

重要的是要注意，不是所有恒等类型都可以通过对类型构造的归纳来“确定”。反例包括大多数非平凡的高阶归纳类型（见第 6 章 和第 8 章）。例如，计算类型 $\mathbb{S}^n$（见??）的恒等类型等价于计算球面的高阶同伦群，这是代数拓扑中深刻且重要的研究领域。

## 2.6 笛卡尔积类型

给定类型 $A$ 和 $B$，考虑笛卡尔积类型 $A \times B$。对于任意元素 $x, y : A \times B$ 和路径 $p : x =_{A\times B} y$，通过函子性我们可以提取路径 $\mathsf{pr}_1(p) : \mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)$ 和 $\mathsf{pr}_2(p) : \mathsf{pr}_2(x) =_B \mathsf{pr}_2(y)$。因此，我们有函数

$$(x =_{A\times B} y) \to (\mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)) \times (\mathsf{pr}_2(x) =_B \mathsf{pr}_2(y)). \tag{2.6.1}$$

**Theorem 2.6.2.** *对任意 $x$ 和 $y$，函数 (2.6.1) 是等价。*

从逻辑上读，这说两个对相等当且仅当它们分量相等。从范畴论上读，这说积群胚中的态射是成对的态射。从同伦论上读，这说积空间中的路径是成对的路径。

[1] 我们选择把这些原理作为公理引入，但可能有其他方式来表述它们成立的类型论。见本章的注释。

证明. 我们需要一个反方向的函数：

$$(\mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)) \times (\mathsf{pr}_2(x) =_B \mathsf{pr}_2(y)) \to (x =_{A\times B} y). \tag{2.6.3}$$

通过笛卡尔积的归纳规则，我们可以假设 $x$ 和 $y$ 都是对，即对某些 $a, a' : A$ 和 $b, b' : B$ 有 $x \equiv (a, b)$ 和 $y \equiv (a', b')$。在这种情况下，我们需要的是函数

$$(a =_A a') \times (b =_B b') \to \big((a, b) =_{A\times B} (a', b')\big).$$

现在通过对其定义域中的笛卡尔积进行归纳，我们可以假设给定 $p : a = a'$ 和 $q : b = b'$。而通过两次路径归纳，我们可以假设 $a \equiv a'$ 且 $b \equiv b'$ 且 $p$ 和 $q$ 都是自反性。但在这种情况下，我们有 $(a, b) \equiv (a', b')$，所以我们可以取输出也是自反性。

剩下要证明 (2.6.3) 是 (2.6.1) 的拟逆。这是一系列简单的归纳，但必须按正确的顺序进行。

在一个方向，让我们从 $r : x =_{A\times B} y$ 开始。我们首先对 $r$ 进行路径归纳以假设 $x \equiv y$ 且 $r$ 是自反性。在这种情况下，由于 $\mathsf{ap}_{\mathsf{pr}_1}$ 和 $\mathsf{ap}_{\mathsf{pr}_2}$ 由路径归纳定义，(2.6.1) 把 $r \equiv \mathsf{refl}_x$ 映到对 $(\mathsf{refl}_{\mathsf{pr}_1 x}, \mathsf{refl}_{\mathsf{pr}_2 x})$。现在通过对 $x$ 的归纳，我们可以假设 $x \equiv (a, b)$，使得这是 $(\mathsf{refl}_a, \mathsf{refl}_b)$。因此，(2.6.3) 把它按定义映到 $\mathsf{refl}_{(a,b)}$，它（在我们当前的假设下）是 $r$。

在另一方向，如果我们从 $s : (\mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)) \times (\mathsf{pr}_2(x) =_B \mathsf{pr}_2(y))$ 开始，那么我们首先对 $x$ 和 $y$ 进行归纳以假设它们是对 $(a, b)$ 和 $(a', b')$，然后对 $s : (a =_A a') \times (b =_B b')$ 进行归纳将其归约为对 $(p, q)$，其中 $p : a = a'$ 且 $q : b = b'$。现在通过对 $p$ 和 $q$ 的归纳，我们可以假设它们是自反性 $\mathsf{refl}_a$ 和 $\mathsf{refl}_b$，在这种情况下 (2.6.3) 给出 $\mathsf{refl}_{(a,b)}$，然后 (2.6.1) 把我们带回 $(\mathsf{refl}_a, \mathsf{refl}_b) \equiv (p, q) \equiv s$。 □

特别地，我们已经证明 (2.6.1) 有逆 (2.6.3)，我们可以把它记作

$$\mathsf{pair}^{=} : (\mathsf{pr}_1(x) = \mathsf{pr}_1(y)) \times (\mathsf{pr}_2(x) = \mathsf{pr}_2(y)) \to (x = y).$$

注意这的一个特例给出了积的命题唯一性原理：$z = (\mathsf{pr}_1(z), \mathsf{pr}_2(z))$。

把 $\mathsf{pair}^{=}$ 视为 $x = y$ 的*构造子*或*引入规则*会很有帮助，类似于 $A \times B$ 本身的“配对”构造子，它在给定 $a : A$ 和 $b : B$ 时引入对 $(a, b)$。从这个角

度， (2.6.1) 的两个分量：

$$\begin{aligned}\mathsf{ap}_{\mathsf{pr}_1} &: (x = y) \to (\mathsf{pr}_1(x) = \mathsf{pr}_1(y))\\ \mathsf{ap}_{\mathsf{pr}_2} &: (x = y) \to (\mathsf{pr}_2(x) = \mathsf{pr}_2(y))\end{aligned}$$

是消除子或消除规则，类似于积类型的投影 $\mathsf{pr}_1$ 和 $\mathsf{pr}_2$。

类似地，见证 (2.6.3) 是 (2.6.1) 的拟逆的两个同伦分别由命题计算规则组成：

$$\begin{aligned}\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{pair}^{=}(p, q)) &= p\\ \mathsf{ap}_{\mathsf{pr}_2}(\mathsf{pair}^{=}(p, q)) &= q\end{aligned}$$

其中 $p : \mathsf{pr}_1 x = \mathsf{pr}_1 y$ 且 $q : \mathsf{pr}_2 x = \mathsf{pr}_2 y$，以及命题唯一性原理：

$$r = \mathsf{pair}^{=}(\mathsf{ap}_{\mathsf{pr}_1}(r), \mathsf{ap}_{\mathsf{pr}_2}(r)) \qquad \text{对于} r : x =_{A\times B} y.$$

我们还可以逐分量刻画 $A \times B$ 中路径的自反性、逆和复合：

$$\begin{aligned}\mathsf{refl}_{(z:A\times B)} &= \mathsf{pair}^{=}(\mathsf{refl}_{\mathsf{pr}_1 z}, \mathsf{refl}_{\mathsf{pr}_2 z})\\ p^{-1} &= \mathsf{pair}^{=}\big(\mathsf{ap}_{\mathsf{pr}_1}(p)^{-1}, \mathsf{ap}_{\mathsf{pr}_2}(p)^{-1}\big)\\ p \mathbin{\cdot} q &= \mathsf{pair}^{=}\big(\mathsf{ap}_{\mathsf{pr}_1}(p) \mathbin{\cdot} \mathsf{ap}_{\mathsf{pr}_1}(q), \mathsf{ap}_{\mathsf{pr}_2}(p) \mathbin{\cdot} \mathsf{ap}_{\mathsf{pr}_2}(q)\big).\end{aligned}$$

或者换种写法：

$$\begin{aligned}\mathsf{ap}_{\mathsf{pr}_i}(\mathsf{refl}_{(z:A\times B)}) &= \mathsf{refl}_{\mathsf{pr}_i z} && (i = 1, 2)\\ \mathsf{pair}^{=}(p^{-1}, q^{-1}) &= \mathsf{pair}^{=}(p, q)^{-1}\\ \mathsf{pair}^{=}(p \mathbin{\cdot} q, p' \mathbin{\cdot} q') &= \mathsf{pair}^{=}(p, p') \mathbin{\cdot} \mathsf{pair}^{=}(q, q').\end{aligned}$$

所有这些等式都可以通过对给定路径进行路径归纳然后返回自反性来推导。对于第 2.1 节中考虑的其余高阶群胚结构也是如此，尽管插入足够多的相容性路径以得到能类型检查的等式开始变得乏味。例如，如果我们用 $\mathsf{assoc}(p, q, r)$ 表示定理 2.1.4(iv)中路径的逆，用 $\mathsf{pair}^{\cdot}(p, q, p', q')$ 表示上面显示的最后一个路径，那么对任意 $u, v, z, w : A \times B$ 和适当类型的

$p, q, r, p', q', r'$ 我们有

$$\begin{aligned}
&\mathsf{pair}^{\cdot}(p \cdot q, r, p' \cdot q', r') \\
&\cdot (\mathsf{pair}^{\cdot}(p, q, p', q') \cdot_{\mathsf{r}} \mathsf{pair}^{=}(r, r')) \\
&\cdot \mathsf{assoc}(\mathsf{pair}^{=}(p, p'), \mathsf{pair}^{=}(q, q'), \mathsf{pair}^{=}(r, r')) \\
&= \quad \mathsf{ap}_{\mathsf{pair}^{=}}(\mathsf{pair}^{=}(\mathsf{assoc}(p, q, r), \mathsf{assoc}(p', q', r'))) \\
&\quad \cdot \mathsf{pair}^{\cdot}(p, q \cdot r, p', q' \cdot r') \\
&\quad \cdot (\mathsf{pair}^{=}(p, p') \cdot_{\mathsf{l}} \mathsf{pair}^{\cdot}(q, r, q', r')).
\end{aligned}$$

幸运的是，我们永远不需要使用任何这样的高维相容性。

现在我们考虑在类型族的逐点积中的传输。给定类型族 $A, B : Z \to \mathcal{U}$，我们滥用记号写 $A \times B : Z \to \mathcal{U}$ 表示由 $(A \times B)(z) :\equiv A(z) \times B(z)$ 定义的类型族。现在给定 $p : z =_Z w$ 和 $x : A(z) \times B(z)$，我们可以沿着 $p$ 传输 $x$ 得到 $A(w) \times B(w)$ 的一个元素。

**Theorem 2.6.4.** 在上述情况下，我们有

$$\mathsf{transport}^{A\times B}(p, x) =_{A(w)\times B(w)} (\mathsf{transport}^{A}(p, \mathsf{pr}_1 x), \mathsf{transport}^{B}(p, \mathsf{pr}_2 x)).$$

证明. 通过路径归纳，我们可以假设 $p$ 是自反性，在这种情况下我们有

$$\begin{aligned}
\mathsf{transport}^{A\times B}(p, x) &\equiv x \\
\mathsf{transport}^{A}(p, \mathsf{pr}_1 x) &\equiv \mathsf{pr}_1 x \\
\mathsf{transport}^{B}(p, \mathsf{pr}_2 x) &\equiv \mathsf{pr}_2 x.
\end{aligned}$$

因此，剩下要证明 $x = (\mathsf{pr}_1 x, \mathsf{pr}_2 x)$。但这正是积类型的命题唯一性原理，如我们上面所述，它从定理 2.6.2推出。 □

最后，我们考虑 $\mathsf{ap}$ 在笛卡尔积下的函子性。假设给定类型 $A, B, A', B'$ 和函数 $g : A \to A'$ 和 $h : B \to B'$；则我们可以定义函数 $f : A \times B \to A' \times B'$ 为 $f(x) :\equiv (g(\mathsf{pr}_1 x), h(\mathsf{pr}_2 x))$。

**Theorem 2.6.5.** 在上述情况下，给定 $x, y : A \times B$ 和 $p : \mathsf{pr}_1 x = \mathsf{pr}_1 y$ 和 $q : \mathsf{pr}_2 x = \mathsf{pr}_2 y$，我们有

$$f(\mathsf{pair}^{=}(p, q)) =_{(f(x)=f(y))} \mathsf{pair}^{=}(g(p), h(q)).$$

证明. 首先注意上述等式是良类型的。一方面，由于 $\mathsf{pair}^{=}(p,q) : x = y$ 我们有 $f(\mathsf{pair}^{=}(p,q)) : f(x) = f(y)$。另一方面，由于 $\mathsf{pr}_1(f(x)) \equiv g(\mathsf{pr}_1 x)$ 且 $\mathsf{pr}_2(f(x)) \equiv h(\mathsf{pr}_2 x)$，我们也有 $\mathsf{pair}^{=}(g(p), h(q)) : f(x) = f(y)$。

现在，通过归纳，我们可以假设 $x \equiv (a,b)$ 且 $y \equiv (a',b')$，在这种情况下我们有 $p : a = a'$ 和 $q : b = b'$。因此，通过路径归纳，我们可以假设 $p$ 和 $q$ 是自反性，在这种情况下所需的等式判断上成立。 □

## 2.7 Σ-类型

设 $A$ 是类型，$P : A \to \mathcal{U}$ 是类型族。回忆 Σ-类型，或依赖对类型，$\sum_{(x:A)} P(x)$ 是笛卡尔积类型的推广。因此，我们期望它的高阶群胚结构也是上一节的推广。特别地，它的路径应该是路径对，但需要一些思考才能给出这些路径的正确类型。

假设我们有 $\sum_{(x:A)} P(x)$ 中的路径 $p : w = w'$。则我们得到 $\mathsf{pr}_1(p) : \mathsf{pr}_1(w) = \mathsf{pr}_1(w')$。然而，我们不能直接问 $\mathsf{pr}_2(w)$ 是否与 $\mathsf{pr}_2(w')$ 相同，因为它们不必在同一个类型中。但我们可以沿着路径 $\mathsf{pr}_1(p)$ 传输 $\mathsf{pr}_2(w)$，这确实给出与 $\mathsf{pr}_2(w')$ 相同类型的元素。通过路径归纳，我们实际上得到路径 $\mathsf{pr}_1(p)_*(\mathsf{pr}_2(w)) = \mathsf{pr}_2(w')$。

回忆?? 前面的讨论，

$$\mathsf{pr}_1(p)_*(\mathsf{pr}_2(w)) = \mathsf{pr}_2(w')$$

可以被视为从 $\mathsf{pr}_2(w)$ 到 $\mathsf{pr}_2(w')$ 的路径类型，这些路径 “位于” $A$ 中的路径 $\mathsf{pr}_1(p)$ 之上。因此，我们说的是全空间中的路径 $w = w'$ 决定（并由之决定）$A$ 中的路径 $p : \mathsf{pr}_1(w) = \mathsf{pr}_1(w')$ 以及从 $\mathsf{pr}_2(w)$ 到 $\mathsf{pr}_2(w')$ 的位于 $p$ 之上的路径，这看起来是合理的。

*Remark* 2.7.1. 注意如果我们有 $x : A$ 和 $u, v : P(x)$ 使得 $(x,u) = (x,v)$，并不能推出 $u = v$ — 参见?? 获取反例。我们能得出的只是存在 $p : x = x$ 使得 $p_*(u) = v$。这是类型论新手常见的困惑来源，但从拓扑的观点看是有道理的：在纤维化的全空间中，恰好位于同一纤维中的两点之间存在路径 $(x,u) = (x,v)$，并不意味着存在完全在那个纤维内部的路径 $u = v$。

下一个定理说明我们也可以反过来进行这个过程。因为它是定理 2.6.2的直接推广，我们将更简洁。

**Theorem 2.7.2.** 假设 $P : A \to \mathcal{U}$ 是类型 $A$ 上的类型族，设 $w, w' : \sum_{(x:A)} P(x)$。则有等价

$$(w = w') \simeq \sum_{(p:\mathsf{pr}_1(w)=\mathsf{pr}_1(w'))} p_*(\mathsf{pr}_2(w)) = \mathsf{pr}_2(w').$$

证明. 我们定义函数

$$f : \prod_{w,w':\sum_{(x:A)} P(x)} (w = w') \to \sum_{(p:\mathsf{pr}_1(w)=\mathsf{pr}_1(w'))} p_*(\mathsf{pr}_2(w)) = \mathsf{pr}_2(w')$$

通过路径归纳，有

$$f(w, w, \mathsf{refl}_w) :\equiv (\mathsf{refl}_{\mathsf{pr}_1(w)}, \mathsf{refl}_{\mathsf{pr}_2(w)}).$$

我们想证明 $f$ 是等价。

在反方向，我们定义

$$g : \prod_{w,w':\sum_{(x:A)} P(x)} \Big( \sum_{p:\mathsf{pr}_1(w)=\mathsf{pr}_1(w')} p_*(\mathsf{pr}_2(w)) = \mathsf{pr}_2(w') \Big) \to (w = w')$$

首先对 $w$ 和 $w'$ 进行归纳，将它们分别拆分为 $(w_1, w_2)$ 和 $(w'_1, w'_2)$，所以只需证明

$$\Big( \sum_{p:w_1=w'_1} p_*(w_2) = w'_2 \Big) \to ((w_1, w_2) = (w'_1, w'_2)).$$

接下来，给定对 $\sum_{(p:w_1=w'_1)} p_*(w_2) = w'_2$，我们可以用 Σ-归纳得到 $p : w_1 = w'_1$ 和 $q : p_*(w_2) = w'_2$。对 $p$ 进行归纳，我们有 $q : (\mathsf{refl}_{w_1})_*(w_2) = w'_2$，只需证明 $(w_1, w_2) = (w_1, w'_2)$。但 $(\mathsf{refl}_{w_1})_*(w_2) \equiv w_2$，所以对 $q$ 进行归纳将目标归约为 $(w_1, w_2) = (w_1, w_2)$，我们可以用 $\mathsf{refl}_{(w_1,w_2)}$ 证明。

接下来我们证明 $f(g(r)) = r$ 对所有 $w$、$w'$ 和 $r$ 成立，其中 $r$ 的类型是

$$\sum_{(p:\mathsf{pr}_1(w)=\mathsf{pr}_1(w'))} (p_*(\mathsf{pr}_2(w)) = \mathsf{pr}_2(w')).$$

首先，我们通过对归纳拆分对 $w$、$w'$ 和 $r$，如同 $g$ 的定义中那样，然后用两次路径归纳将 $r$ 的两个分量都归约为 $\mathsf{refl}$ 。然后只需证明 $f(g(\mathsf{refl}_{w_1}, \mathsf{refl}_{w_2})) = (\mathsf{refl}_{w_1}, \mathsf{refl}_{w_2})$，这由定义为真。

类似地，为了证明 $g(f(p)) = p$ 对所有 $w$、$w'$ 和 $p : w = w'$ 成立，我们可以对 $p$ 进行路径归纳，然后对归纳拆分 $w$，此时只需证明 $g(f(\mathsf{refl}_{(w_1,w_2)})) = \mathsf{refl}_{(w_1,w_2)}$，这由定义为真。

因此，$f$ 有拟逆，从而是等价。 □

如同我们对笛卡尔积所做的，我们可以推导出一个命题唯一性原理作为特例。

**Corollary 2.7.3.** 对于 $z : \sum_{(x:A)} P(x)$，我们有 $z = (\mathsf{pr}_1(z), \mathsf{pr}_2(z))$。

证明. 我们有 $\mathsf{refl}_{\mathsf{pr}_1(z)} : \mathsf{pr}_1(z) = \mathsf{pr}_1(\mathsf{pr}_1(z), \mathsf{pr}_2(z))$，所以由定理 2.7.2只需展示路径 $(\mathsf{refl}_{\mathsf{pr}_1(z)})_*(\mathsf{pr}_2(z)) = \mathsf{pr}_2(\mathsf{pr}_1(z), \mathsf{pr}_2(z))$。但两边判断上都等于 $\mathsf{pr}_2(z)$。 □

与二元笛卡尔积一样，我们可以把定理 2.7.2的反方向视为引入形式 ($\mathsf{pair}^{=}$)，正方向视为消除形式 ($\mathsf{ap}_{\mathsf{pr}_1}$ 和 $\mathsf{ap}_{\mathsf{pr}_2}$)，而等价给出它们的命题计算规则和唯一性原理。

注意定理 2.3.2中定义的 $p : x = y$ 在 $u : P(x)$ 处的提升路径 $\mathsf{lift}(u, p)$ 可以与引入形式的特例等同：

$$\mathsf{pair}^{=}(p, \mathsf{refl}_{p_*(u)}) : (x, u) = (y, p_*(u)).$$

这出现在 Σ-类型上传输作用的陈述中，它也是二元笛卡尔积作用的推广：

**Theorem 2.7.4.** 假设我们有类型族

$$P : A \to \mathcal{U} \qquad 和 \qquad Q : \Big(\sum_{x:A} P(x)\Big) \to \mathcal{U}.$$

则我们可以构造 $A$ 上的类型族，定义为

$$x \mapsto \sum_{u:P(x)} Q(x, u).$$

对任意路径 $p : x = y$ 和任意 $(u, z) : \sum_{(u:P(x))} Q(x, u)$ 我们有

$$p_*(u, z) = \big(p_*(u),\ \mathsf{pair}^{=}(p, \mathsf{refl}_{p_*(u)})_*(z)\big).$$

证明. 直接由路径归纳。 □

我们把定理 2.6.5的推广陈述和证明留给读者（参见??），以及逐分量刻画 Σ-类型的自反性、逆和复合。

## 2.8 单位类型

平凡情况有时也很重要，所以我们简要提一下单位类型 $\mathbf{1}$ 的情况。

**Theorem 2.8.1.** 对任意 $x, y : \mathbf{1}$，我们有 $(x = y) \simeq \mathbf{1}$。

开始这个证明时可能很想对 $x$ 和 $y$ 进行 $\mathbf{1}$-归纳，将问题归约为 $(\star = \star) \simeq \mathbf{1}$。然而，此时我们会陷入困境，因为我们无法对 $p : \star = \star$ 进行路径归纳。因此，我们尽可能地处理一般的 $x$ 和 $y$，只在最后时刻才通过归纳将它们归约为 $\star$。

证明. 函数 $(x = y) \to \mathbf{1}$ 容易定义，把所有东西都映到 $\star$ 。反过来，对任意 $x, y : \mathbf{1}$ 我们可以通过归纳假设 $x \equiv \star \equiv y$。在这种情况下我们有 $\mathsf{refl}_\star : x = y$，产生常函数 $\mathbf{1} \to (x = y)$。

为了证明这些是互逆的，首先考虑元素 $u : \mathbf{1}$。我们可以假设 $u \equiv \star$，但这也是复合 $\mathbf{1} \to (x = y) \to \mathbf{1}$ 的结果。

另一方面，假设给定 $p : x = y$。通过路径归纳，我们可以假设 $x \equiv y$ 且 $p$ 是 $\mathsf{refl}_x$。然后我们可以假设 $x$ 是 $\star$ ，在这种情况下复合 $(x = y) \to \mathbf{1} \to (x = y)$ 把 $p$ 映到 $\mathsf{refl}_x$，即 $p$。 □

特别地，$\mathbf{1}$ 的任意两个元素相等。我们把用引入、消除、计算和唯一性规则表述这个等价留给读者。$\mathbf{1}$ 的传输引理就是常类型族的传输引理（??）。

## 2.9 Π-类型与函数外延性公理

给定类型 $A$ 和类型族 $B : A \to \mathcal{U}$，考虑依赖函数类型 $\prod_{(x:A)} B(x)$。我们期望 $\prod_{(x:A)} B(x)$ 中从 $f$ 到 $g$ 的路径类型 $f = g$ 等价于逐点路径的类型：

$$(f = g) \simeq \left(\prod_{x:A} (f(x) =_{B(x)} g(x))\right). \tag{2.9.1}$$

从传统角度，这说在每一点相等的两个函数作为函数相等。从拓扑角度，这说函数空间中的路径等同于连续同伦。从范畴论角度，这说函子范畴中的同构是同构的自然族。

然而，与前面几节的情况不同，第 1 章中呈现的基本类型论不足以证明 (2.9.1)。我们能说的只是存在某个函数

$$\mathsf{happly} : (f = g) \to \prod_{x:A} (f(x) =_{B(x)} g(x)) \tag{2.9.2}$$

它容易通过路径归纳定义。因此，目前我们假设：

**Axiom 2.9.3** (函数外延性)**.** 对任意 $A$、$B$、$f$ 和 $g$，函数 (2.9.2) 是等价。

我们将在后面的章节看到这个公理既从泛等性推出（参见第 2.10 节 和第 4.9 节），也从区间类型推出（参见?? 和??）。

特别地，2.9.3蕴含 (2.9.2) 有拟逆

$$\mathsf{funext} : \left(\prod_{x:A} (f(x) = g(x))\right) \to (f = g).$$

这个函数也被称为 “函数外延性”。如同我们在第 2.6 节中对 $\mathsf{pair}^{=}$ 所做的，我们可以把 funext 视为 $f = g$ 类型的引入规则。从这个角度，happly 是消除规则，而见证 funext 是 happly 的拟逆的同伦成为命题计算规则

$$\mathsf{happly}(\mathsf{funext}(h), x) = h(x) \qquad \text{对于} h : \prod_{x:A} (f(x) = g(x))$$

和命题唯一性原理：

$$p = \mathsf{funext}(x \mapsto \mathsf{happly}(p, x)) \qquad \text{对于} p : f = g.$$

我们也可以计算 Π-类型中的恒等、逆和复合；它们简单地由逐点运算给出：

$$\begin{aligned}
\mathsf{refl}_f &= \mathsf{funext}(x \mapsto \mathsf{refl}_{f(x)}) \\
\alpha^{-1} &= \mathsf{funext}(x \mapsto \mathsf{happly}(\alpha, x)^{-1}) \\
\alpha \cdot \beta &= \mathsf{funext}(x \mapsto \mathsf{happly}(\alpha, x) \cdot \mathsf{happly}(\beta, x)).
\end{aligned}$$

第一个等式从 happly 的定义推出，而第二个和第三个是简单的路径归纳。

因为非依赖函数类型 $A \to B$ 是依赖函数类型 $\prod_{(x:A)} B(x)$（当 $B$ 不依赖于 $x$）的特例，我们上面说的一切在非依赖情况下也适用。然而，传输规则在非依赖情况下稍微简单一些。给定类型 $X$，路径 $p : x_1 =_X x_2$，类型族 $A, B : X \to \mathcal{U}$，和函数 $f : A(x_1) \to B(x_1)$，我们有

$$\mathsf{transport}^{A\to B}(p, f) = \Big(x \mapsto \mathsf{transport}^{B}(p, f(\mathsf{transport}^{A}(p^{-1}, x)))\Big) \quad (2.9.4)$$

其中 $A \to B$ 滥用地表示由以下定义的类型族 $X \to \mathcal{U}$：

$$(A \to B)(x) :\equiv (A(x) \to B(x)).$$

换句话说，当我们沿着路径 $p : x_1 = x_2$ 传输函数 $f : A(x_1) \to B(x_1)$ 时，我们得到函数 $A(x_2) \to B(x_2)$，它把参数沿 $p$ 反向传输（在类型族 $A$ 中），应用 $f$，然后把结果沿 $p$ 正向传输（在类型族 $B$ 中）。这可以通过路径归纳容易地证明。

传输依赖函数类似，但更复杂。假设给定 $X$ 和 $p$ 如前，类型族 $A : X \to \mathcal{U}$ 和 $B : \prod_{(x:X)}(A(x) \to \mathcal{U})$，以及依赖函数 $f : \prod_{(a:A(x_1))} B(x_1, a)$。则对于 $a : A(x_2)$，我们有

$$\mathsf{transport}^{\Pi_A(B)}(p, f)(a) = \mathsf{transport}^{\widehat{B}}\Big((\mathsf{pair}^{=}(p^{-1}, \mathsf{refl}_{p^{-1}{}_*(a)}))^{-1},\ f(\mathsf{transport}^{A}(p^{-1}, a))\Big)$$

其中 $\Pi_A(B)$ 和 $\widehat{B}$ 分别表示类型族

$$\begin{array}{rcll} \Pi_A(B) & :\equiv & \big(x \mapsto \prod_{(a:A(x))} B(x, a)\big) & : \quad X \to \mathcal{U} \\ \widehat{B} & :\equiv & \big(w \mapsto B(\mathsf{pr}_1 w, \mathsf{pr}_2 w)\big) & : \quad \big(\sum_{(x:X)} A(x)\big) \to \mathcal{U}. \end{array} \quad (2.9.5)$$

如果这些公式看起来有点吓人，不要担心细节。基本思想与非依赖函数类型相同：我们反向传输参数，应用函数，然后正向传输结果。

现在回忆对于一般的类型族 $P : X \to \mathcal{U}$，在第 2.2 节中我们定义了在 $p : x =_X y$ 之上从 $u : P(x)$ 到 $v : P(y)$ 的依赖路径的类型为 $p_*(u) =_{P(y)} v$。当 $P$ 是函数类型的族时，有一种等价的方式来表示它，这通常更方便。

**Lemma 2.9.6.** 给定类型族 $A, B : X \to \mathcal{U}$ 和 $p : x =_X y$，以及 $f : A(x) \to B(x)$ 和 $g : A(y) \to B(y)$，我们有等价

$$\big(p_*(f) = g\big) \simeq \prod_{a:A(x)} \big(p_*(f(a)) = g(p_*(a))\big).$$

此外，如果 $q : p_*(f) = g$ 在这个等价下对应于 $\widehat{q}$，则对于 $a : A(x)$，路径

$$\mathsf{happly}(q, p_*(a)) : (p_*(f))(p_*(a)) = g(p_*(a))$$

等于连接路径 $i \centerdot j \centerdot k$，其中

- $i : (p_*(f))(p_*(a)) = p_*\big(f({p^{-1}}_*(p_*(a)))\big)$ 来自 (2.9.4)，
- $j : p_*\big(f({p^{-1}}_*(p_*(a)))\big) = p_*(f(a))$ 来自 *??* 和定理 *2.1.4*，且
- $k : p_*(f(a)) = g(p_*(a))$ 是 $\widehat{q}(a)$。

证明. 通过路径归纳，我们可以假设 $p$ 是自反性，在这种情况下所需的等价归约为函数外延性。第二个陈述然后从函数外延性的计算规则推出。 □

一般来说，我们经常想要考虑路径的连接，其中每条路径都来自某个先前证明的引理或假设的对象，用给连接中的每条路径命名的方式来描述这些可能相当乏味，如同我们在上面的第二个陈述中所做的那样。因此，我们采用以熟悉的数学风格写这种连接的约定，即“带有理由的等式链”，并允许我们省略读者容易填补的理由。例如，定理 2.9.6中的路径 $i \centerdot j \centerdot k$ 会这样写：

$$\begin{aligned}
(p_*(f))(p_*(a)) &= p_*\big(f({p^{-1}}_*(p_*(a)))\big) && (\text{由 (2.9.4)}) \\
&= p_*(f(a)) \\
&= g(p_*(a)). && (\text{由 } \widehat{q})
\end{aligned}$$

在普通数学中，这样的等式链只是证明两个东西相等。我们通过用它来描述它们之间的一条特定路径来增强这一点。

像往常一样，有一个依赖函数版本的定理 2.9.6，类似但更复杂。

**Lemma 2.9.7.** 给定类型族 $A : X \to \mathcal{U}$ 和 $B : \prod_{(x:X)} A(x) \to \mathcal{U}$ 和 $p : x =_X y$，以及 $f : \prod_{(a:A(x))} B(x,a)$ 和 $g : \prod_{(a:A(y))} B(y,a)$，我们有等价

$$\big(p_*(f) = g\big) \;\simeq\; \Big(\prod_{a:A(x)} \mathsf{transport}^{\widehat{B}}(\mathsf{pair}^{=}(p, \mathsf{refl}_{p_*(a)}), f(a)) = g(p_*(a))\Big)$$

其中 $\widehat{B}$ 如 (2.9.5)。

我们把这个的证明和表述合适的计算规则留给读者。

## 2.10 宇宙与泛等公理

给定两个类型 $A$ 和 $B$，我们可以把它们视为某个宇宙类型 $\mathcal{U}$ 的元素，从而形成恒等类型 $A =_{\mathcal{U}} B$。如导论中所述，泛等性就是把 $A =_{\mathcal{U}} B$ 与从 $A$ 到 $B$ 的等价类型 $(A \simeq B)$（我们在第 2.4 节中描述过）等同起来。我们通过以下典范函数来实现这个等同。

**Lemma 2.10.1.** 对于类型 $A, B : \mathcal{U}$，存在某个函数，

$$\mathsf{idtoeqv} : (A =_{\mathcal{U}} B) \to (A \simeq B), \tag{2.10.2}$$

定义见证明。

证明. 我们可以通过对相等的归纳直接构造它，但以下描述更方便。注意恒等函数 $\mathsf{id}_{\mathcal{U}} : \mathcal{U} \to \mathcal{U}$ 可以被视为由宇宙 $\mathcal{U}$ 索引的类型族；它把每个类型 $X : \mathcal{U}$ 赋给类型 $X$ 本身。（当被视为纤维化时，它的全空间是 “有点类型” 的类型 $\sum_{(A:\mathcal{U})} A$；另见第 4.8 节。）因此，给定路径 $p : A =_{\mathcal{U}} B$，我们有传输函数 $p_* : A \to B$。我们声称 $p_*$ 是等价。但通过归纳，只需假设 $p$ 是 $\mathsf{refl}_A$，在这种情况下 $p_* \equiv \mathsf{id}_A$，由例 2.4.7它是等价。因此，我们可以定义 $\mathsf{idtoeqv}(p)$ 为 $p_*$（连同上述它是等价的证明）。 □

我们想说 idtoeqv 是等价。然而，与函数类型的 happly 一样，第 1 章中描述的类型论不足以保证这一点。因此，与函数外延性一样，我们把这个性质表述为公理：Voevodsky 的泛等公理。

**Axiom 2.10.3** (泛等性). 对任意 $A, B : \mathcal{U}$，函数 (2.10.2) 是等价。

因此特别地，我们有

$$(A =_{\mathcal{U}} B) \simeq (A \simeq B).$$

从技术上讲，泛等公理是关于特定宇宙类型 $\mathcal{U}$ 的陈述。如果宇宙 $\mathcal{U}$ 满足这个公理，我们说它是**泛等的**。除非另有说明（例如在第 4.9 节中），我们将假设所有宇宙都是泛等的。

*Remark* 2.10.4. 对于泛等公理，重要的是我们使用如第 2.4 节中描述的 “好的” isequiv 版本来定义 $A \simeq B$，而不是（比如说）定义为 $\sum_{(f:A\to B)} \mathsf{qinv}(f)$。参见练习 4.6。

特别地，泛等性意味着等价的类型可以被等同。如同我们在前面几节所做的，把这个等价分解为以下几部分是有用的：

- $(A =_{\mathcal{U}} B)$ 的引入规则，记为 ua，表示 “泛等公理”：

$$\mathsf{ua} : (A \simeq B) \to (A =_{\mathcal{U}} B).$$

- 消除规则，即 idtoeqv，

$$\mathsf{idtoeqv} \equiv \mathsf{transport}^{X\mapsto X} : (A =_{\mathcal{U}} B) \to (A \simeq B).$$

- 命题计算规则，

$$\mathsf{transport}^{X\mapsto X}(\mathsf{ua}(f), x) = f(x).$$

- 命题唯一性原理：对任意 $p : A = B$，

$$p = \mathsf{ua}(\mathsf{transport}^{X\mapsto X}(p)).$$

我们也可以把宇宙中相等的自反性、连接和逆与等价上的对应运算等同起来：

$$\begin{aligned} \mathsf{refl}_A &= \mathsf{ua}(\mathsf{id}_A) \\ \mathsf{ua}(f) \cdot \mathsf{ua}(g) &= \mathsf{ua}(g \circ f) \\ \mathsf{ua}(f)^{-1} &= \mathsf{ua}(f^{-1}). \end{aligned}$$

第一个等式成立是因为 $\mathsf{id}_A = \mathsf{idtoeqv}(\mathsf{refl}_A)$ 由 idtoeqv 的定义，且 ua 是 idtoeqv 的逆。对于第二个，如果我们定义 $p :\equiv \mathsf{ua}(f)$ 和 $q :\equiv \mathsf{ua}(g)$，则我们有

$$\mathsf{ua}(g \circ f) = \mathsf{ua}(\mathsf{idtoeqv}(q) \circ \mathsf{idtoeqv}(p)) = \mathsf{ua}(\mathsf{idtoeqv}(p \cdot q)) = p \cdot q$$

使用?? 和 idtoeqv 的定义。第三个类似。

以下观察是?? 的特例，在应用泛等公理时经常有用。

**Lemma 2.10.5.** 对任意类型族 $B : A \to \mathcal{U}$ 和 $x, y : A$，路径 $p : x = y$ 和 $u : B(x)$，我们有

$$\begin{aligned}\mathsf{transport}^B(p, u) &= \mathsf{transport}^{X \mapsto X}(\mathsf{ap}_B(p), u) \\ &= \mathsf{idtoeqv}(\mathsf{ap}_B(p))(u).\end{aligned}$$

## 2.11 恒等类型

正如类型 $a =_A a'$ 被刻画到同构，对每个 $A$ 有单独的 “定义”，路径之间的路径类型 $p =_{a=_A a'} q$ （其中 $p, q : a =_A a'$）没有简单的刻画。然而，我们其他的一般类定理确实扩展到恒等类型，例如它们保持等价的事实。

**Theorem 2.11.1.** 如果 $f : A \to B$ 是等价，则对所有 $a, a' : A$，

$$\mathsf{ap}_f : (a =_A a') \to (f(a) =_B f(a'))$$

也是等价。

证明. 设 $f^{-1}$ 是 $f$ 的拟逆，带同伦

$$\alpha : \prod_{b:B} (f(f^{-1}(b)) = b) \qquad \text{和} \qquad \beta : \prod_{a:A} (f^{-1}(f(a)) = a).$$

$\mathsf{ap}_f$ 的拟逆本质上是

$$\mathsf{ap}_{f^{-1}} : (f(a) = f(a')) \to (f^{-1}(f(a)) = f^{-1}(f(a'))).$$

然而，为了从 $\mathsf{ap}_{f^{-1}}(q)$ 得到 $a =_A a'$ 的元素，我们必须在两侧连接路径 ${\beta_a}^{-1}$ 和 $\beta_{a'}$。为了证明这给出 $\mathsf{ap}_f$ 的拟逆，一方面我们必须证明对任意 $p : a =_A a'$ 有

$$ {\beta_a}^{-1} \cdot \mathsf{ap}_{f^{-1}}(\mathsf{ap}_f(p)) \cdot \beta_{a'} = p. $$

这从 ap 的函子性和同伦的自然性推出，引理 2.2.2 和引理 2.4.3。另一方面，我们必须证明对任意 $q : f(a) =_B f(a')$ 有

$$ \mathsf{ap}_f\big({\beta_a}^{-1} \cdot \mathsf{ap}_{f^{-1}}(q) \cdot \beta_{a'}\big) = q. $$

这个证明稍微复杂一些，但每一步仍然是引理 2.2.2 和引理 2.4.3的应用（或简单地消去逆路径）：

$$ \begin{aligned} &\mathsf{ap}_f\big({\beta_a}^{-1} \cdot \mathsf{ap}_{f^{-1}}(q) \cdot \beta_{a'}\big) \\ &\qquad = {\alpha_{f(a)}}^{-1} \cdot \alpha_{f(a)} \cdot \mathsf{ap}_f\big({\beta_a}^{-1} \cdot \mathsf{ap}_{f^{-1}}(q) \cdot \beta_{a'}\big) \cdot {\alpha_{f(a')}}^{-1} \cdot \alpha_{f(a')} \\ &\qquad = {\alpha_{f(a)}}^{-1} \cdot \mathsf{ap}_f\big(\mathsf{ap}_{f^{-1}}\big(\mathsf{ap}_f\big({\beta_a}^{-1} \cdot \mathsf{ap}_{f^{-1}}(q) \cdot \beta_{a'}\big)\big)\big) \cdot \alpha_{f(a')} \\ &\qquad = {\alpha_{f(a)}}^{-1} \cdot \mathsf{ap}_f\big(\beta_a \cdot {\beta_a}^{-1} \cdot \mathsf{ap}_{f^{-1}}(q) \cdot \beta_{a'} \cdot {\beta_{a'}}^{-1}\big) \cdot \alpha_{f(a')} \\ &\qquad = {\alpha_{f(a)}}^{-1} \cdot \mathsf{ap}_f(\mathsf{ap}_{f^{-1}}(q)) \cdot \alpha_{f(a')} \\ &\qquad = q. \end{aligned} $$

□

因此，如果对某个类型 $A$ 我们有 $a =_A a'$ 的完全刻画，类型 $p =_{a=_A a'} q$ 也随之确定。例如：

- 路径 $p = q$，其中 $p, q : w =_{A\times B} w'$，等价于路径对

$$ \mathsf{ap}_{\mathsf{pr}_1} p =_{\mathsf{pr}_1 w = _A \mathsf{pr}_1 w'} \mathsf{ap}_{\mathsf{pr}_1} q \quad \text{和} \quad \mathsf{ap}_{\mathsf{pr}_2} p =_{\mathsf{pr}_2 w = _B \mathsf{pr}_2 w'} \mathsf{ap}_{\mathsf{pr}_2} q. $$

- 路径 $p = q$，其中 $p, q : f =_{\prod_{(x:A)} B(x)} g$，等价于同伦

$$ \prod_{x:A} (\mathsf{happly}(p)(x) =_{f(x)=g(x)} \mathsf{happly}(q)(x)). $$

接下来我们考虑在路径族中的传输，即在 $C : A \to \mathcal{U}$ 中的传输，其中每个 $C(x)$ 是恒等类型。最简单的情况是 $C(x)$ 是 $A$ 本身中的路径类型，可能有一个端点固定。

**Lemma 2.11.2.** 对任意 $A$ 和 $a : A$，以及 $p : x_1 = x_2$，我们有

$$\begin{aligned}
&\mathsf{transport}^{x\mapsto(a=x)}(p, q) = q \cdot p && \text{对于 } q : a = x_1,\\
&\mathsf{transport}^{x\mapsto(x=a)}(p, q) = p^{-1} \cdot q && \text{对于 } q : x_1 = a,\\
&\mathsf{transport}^{x\mapsto(x=x)}(p, q) = p^{-1} \cdot q \cdot p && \text{对于 } q : x_1 = x_1。
\end{aligned}$$

证明. 对 $p$ 进行路径归纳，然后用复合的单位元律。 □

换句话说，用 $x \mapsto c = x$ 传输是后复合，用 $x \mapsto x = c$ 传输是逆变的前复合。这些可能作为范畴论中协变和逆变 hom-函子 $\hom(c, -)$ 和 $\hom(-, c)$ 的函子作用而为人熟知。

类似地，我们可以证明以下推论 2.11.2的更一般形式，它与?? 相关。

**Theorem 2.11.3.** 对于 $f, g : A \to B$，$p : a =_A a'$ 和 $q : f(a) =_B g(a)$，我们有

$$\mathsf{transport}^{x\mapsto f(x)=_B g(x)}(p, q) =_{f(a')=g(a')} (\mathsf{ap}_f p)^{-1} \cdot q \cdot \mathsf{ap}_g p.$$

因为 $\mathsf{ap}_{(x\mapsto x)}$ 是恒等函数且 $\mathsf{ap}_{(x\mapsto c)}$（其中 $c$ 是常数）是 $p \mapsto \mathsf{refl}_c$，推论 2.11.2是特例。一个更一般的版本是当 $B$ 可以是 $A$ 上索引的类型族时：

**Theorem 2.11.4.** 设 $B : A \to \mathcal{U}$ 和 $f, g : \prod_{(x:A)} B(x)$，$p : a =_A a'$ 和 $q : f(a) =_{B(a)} g(a)$。则我们有

$$\mathsf{transport}^{x\mapsto f(x)=_{B(x)} g(x)}(p, q) = (\mathsf{apd}_f(p))^{-1} \cdot \mathsf{ap}_{(\mathsf{transport}^B p)}(q) \cdot \mathsf{apd}_g(p).$$

最后，如第 2.9 节中那样，对于恒等类型的族，依赖路径有另一种等价刻画。

**Theorem 2.11.5.** 对于 $p : a =_A a'$，$q : a = a$ 和 $r : a' = a'$，我们有

$$\big(\mathsf{transport}^{x\mapsto(x=x)}(p, q) = r\big) \simeq \big(q \cdot p = p \cdot r\big).$$

证明. 对 $p$ 进行路径归纳，然后用与单位等式 $q \cdot 1 = q$ 和 $r = 1 \cdot r$ 复合是等价的事实。 □

存在涉及函数应用的更一般等价，类似于定理 2.11.3 和定理 2.11.4。

## 2.12 余积

到目前为止，我们考虑的大多数类型构造子是所谓的负的。直观地说，这意味着它们的元素由其在消除规则下的行为决定：（依赖）对由其投影决定，（依赖）函数由其值决定。负类型的恒等类型几乎总是可以直接刻画，连同所有高阶结构，如我们在第 2.6 节 至 第 2.9 节中所做的。宇宙不完全是负类型，但它的恒等类型表现类似：我们有直接的刻画（泛等性）和高阶结构的描述。恒等类型本身当然是特例。

现在我们考虑第一个正类型构造子的例子。再次非形式地说，正类型是由某些构造子“呈现”的类型，呈现的泛性质由其消除规则表达。（从范畴论角度，正类型有“映出”的泛性质，而负类型有“映入”的泛性质。）因为用呈现计算一般是不可计算的问题，对于正类型我们不能总是期望恒等类型有直接的刻画。然而，在许多特定情况下，刻画或部分刻画确实存在，可以用我们用这个例子介绍的一般方法获得。

（从技术上讲，我们选择的笛卡尔积和 $\Sigma$-类型的呈现也是正的。然而，因为这些类型也允许只稍有不同的负呈现，它们的恒等类型有直接的刻画，不需要这里要描述的方法。）

考虑余积类型 $A+B$，它由单射 $\mathsf{inl} : A \to A+B$ 和 $\mathsf{inr} : B \to A+B$“呈现”。直观地，我们期望 $A+B$ 不交地包含 $A$ 和 $B$ 的精确副本，所以我们应该有

$$(\mathsf{inl}(a_1) = \mathsf{inl}(a_2)) \simeq (a_1 = a_2) \tag{2.12.1}$$

$$(\mathsf{inr}(b_1) = \mathsf{inr}(b_2)) \simeq (b_1 = b_2) \tag{2.12.2}$$

$$(\mathsf{inl}(a) = \mathsf{inr}(b)) \simeq \mathbf{0}. \tag{2.12.3}$$

我们如下证明这一点。固定元素 $a_0 : A$；我们将刻画类型族

$$(x \mapsto (\mathsf{inl}(a_0) = x)) : A + B \to \mathcal{U}. \tag{2.12.4}$$

类似的论证会刻画类似的族 $x \mapsto (x = \mathsf{inr}(b_0))$（对任意 $b_0 : B$）。这些刻画一起蕴含 (2.12.1)–(2.12.3)。

为了刻画 (2.12.4)，我们将定义类型族 $\mathsf{code} : A + B \to \mathcal{U}$ 并证明 $\prod_{(x:A+B)}((\mathsf{inl}(a_0) = x) \simeq \mathsf{code}(x))$。因为我们想从这推出 (2.12.1)，我们

应该有 $\mathsf{code}(\mathsf{inl}(a)) = (a_0 = a)$，因为我们也想推出 (2.12.3)，我们应该有 $\mathsf{code}(\mathsf{inr}(b)) = \mathbf{0}$。关键洞察是我们可以用 $A + B$ 的递归原理来定义 $\mathsf{code} : A + B \to \mathcal{U}$ 由这两个方程：

$$\begin{aligned}\mathsf{code}(\mathsf{inl}(a)) &:\equiv (a_0 = a),\\ \mathsf{code}(\mathsf{inr}(b)) &:\equiv \mathbf{0}.\end{aligned}$$

这是在同伦类型论中进行同伦论时经常使用的证明技术的一个非常简单的例子；参见例如?? 和??。现在我们可以证明：

**Theorem 2.12.5.** 对所有 $x : A + B$ 我们有 $(\mathsf{inl}(a_0) = x) \simeq \mathsf{code}(x)$。

证明. 以下证明的关键是我们对所有点 $x$ 一起证明，使我们能用余积的消除原理。我们首先定义函数

$$\mathsf{encode} : \prod_{(x:A+B)} \prod_{(p:\mathsf{inl}(a_0)=x)} \mathsf{code}(x)$$

通过沿 $p$ 传输自反性：

$$\mathsf{encode}(x, p) :\equiv \mathsf{transport}^{\mathsf{code}}(p, \mathsf{refl}_{a_0}).$$

注意 $\mathsf{refl}_{a_0} : \mathsf{code}(\mathsf{inl}(a_0))$，因为 $\mathsf{code}(\mathsf{inl}(a_0)) \equiv (a_0 = a_0)$ 由 code 的定义。接下来，我们定义函数

$$\mathsf{decode} : \prod_{(x:A+B)} \prod_{(c:\mathsf{code}(x))} (\mathsf{inl}(a_0) = x).$$

为了定义 $\mathsf{decode}(x, c)$，我们可以首先用 $A + B$ 的消除原理根据 $x$ 是形如 $\mathsf{inl}(a)$ 还是形如 $\mathsf{inr}(b)$ 分情况。

在第一种情况，$x \equiv \mathsf{inl}(a)$，则 $\mathsf{code}(x) \equiv (a_0 = a)$，所以 $c$ 是 $a_0$ 和 $a$ 之间的等同。因此，$\mathsf{ap}_{\mathsf{inl}}(c) : (\mathsf{inl}(a_0) = \mathsf{inl}(a))$ 所以我们可以定义这为 $\mathsf{decode}(\mathsf{inl}(a), c)$。

在第二种情况，$x \equiv \mathsf{inr}(b)$，则 $\mathsf{code}(x) \equiv \mathbf{0}$，所以 $c$ 居留空类型。因此，$\mathbf{0}$ 的消除规则产生 $\mathsf{decode}(\mathsf{inr}(b), c)$ 的值。

这完成了 decode 的定义；现在我们证明 $\mathsf{encode}(x,-)$ 和 $\mathsf{decode}(x,-)$ 对所有 $x$ 是拟逆。一方面，假设给定 $x:A+B$ 和 $p:\mathsf{inl}(a_0)=x$；我们想证明

$$\mathsf{decode}(x,\mathsf{encode}(x,p))=p.$$

但现在通过（基于的）路径归纳，只需考虑 $x\equiv\mathsf{inl}(a_0)$ 和 $p\equiv\mathsf{refl}_{\mathsf{inl}(a_0)}$：

$$\begin{aligned}
\mathsf{decode}(x,\mathsf{encode}(x,p)) &\equiv \mathsf{decode}(\mathsf{inl}(a_0),\mathsf{encode}(\mathsf{inl}(a_0),\mathsf{refl}_{\mathsf{inl}(a_0)}))\\
&\equiv \mathsf{decode}(\mathsf{inl}(a_0),\mathsf{transport}^{\mathsf{code}}(\mathsf{refl}_{\mathsf{inl}(a_0)},\mathsf{refl}_{a_0}))\\
&\equiv \mathsf{decode}(\mathsf{inl}(a_0),\mathsf{refl}_{a_0})\\
&\equiv \mathsf{ap}_{\mathsf{inl}}(\mathsf{refl}_{a_0})\\
&\equiv \mathsf{refl}_{\mathsf{inl}(a_0)}\\
&\equiv p.
\end{aligned}$$

另一方面，设 $x:A+B$ 和 $c:\mathsf{code}(x)$；我们想证明 $\mathsf{encode}(x,\mathsf{decode}(x,c))=c$。我们可以再次根据 $x$ 分情况。如果 $x\equiv\mathsf{inl}(a)$，则 $c:a_0=a$ 且 $\mathsf{decode}(x,c)\equiv\mathsf{ap}_{\mathsf{inl}}(c)$，所以

$$\begin{aligned}
\mathsf{encode}(x,\mathsf{decode}(x,c)) &\equiv \mathsf{transport}^{\mathsf{code}}(\mathsf{ap}_{\mathsf{inl}}(c),\mathsf{refl}_{a_0})\\
&= \mathsf{transport}^{a\mapsto(a_0=a)}(c,\mathsf{refl}_{a_0}) && (\text{由??})\\
&= \mathsf{refl}_{a_0}\cdot c && (\text{由推论 2.11.2})\\
&= c.
\end{aligned}$$

最后，如果 $x\equiv\mathsf{inr}(b)$，则 $c:\mathbf{0}$，所以我们可以得出任何我们想要的结论。□

当然，如果我们固定 $b_0:B$ 而不是 $a_0:A$，有相应的定理。

特别地，定理 2.12.5蕴含对任意 $a:A$ 和 $b:B$ 有函数

$$\mathsf{encode}(\mathsf{inl}(a),-):(\mathsf{inl}(a_0)=\mathsf{inl}(a))\to(a_0=a)$$

和

$$\mathsf{encode}(\mathsf{inr}(b),-):(\mathsf{inl}(a_0)=\mathsf{inr}(b))\to\mathbf{0}.$$

第二个说 “$\mathsf{inl}(a_0)$ 不等于 $\mathsf{inr}(b)$”，即 inl 和 inr 的像是不交的。第一个的传统解读，其中恒等类型被视为命题，就是 inl 的单射性。定理 2.12.5的完整

同伦论陈述给出更多信息：类型 $\mathsf{inl}(a_0) = \mathsf{inl}(a)$ 和 $a_0 = a$ 实际上是等价的，$\mathsf{inr}(b_0) = \mathsf{inr}(b)$ 和 $b_0 = b$ 也是。

*Remark* 2.12.6. 特别地，因为二元类型 $\mathbf{2}$ 等价于 $\mathbf{1} + \mathbf{1}$，我们有 $0_{\mathbf{2}} \neq 1_{\mathbf{2}}$。

这个证明说明了描述路径空间的一般方法，我们将经常使用。为了刻画路径空间，第一步是定义比较纤维化 "code"，它提供路径的更显式描述。有几种不同的方法证明这样的比较纤维化等价于路径（我们在?? 中对同一结果展示几种不同的证明）。我们这里使用的叫做**编码-解码方法**：关键思想是对纤维化的所有实例一般地定义 $\mathsf{decode}$（即作为函数 $\prod_{(x:A+B)} \mathsf{code}(x) \to (\mathsf{inl}(a_0) = x)$），这样路径归纳可以用来分析 $\mathsf{decode}(x, \mathsf{encode}(x, p))$。

像往常一样，我们也可以刻画余积类型中传输的作用。给定类型 $X$，路径 $p : x_1 =_X x_2$，和类型族 $A, B : X \to \mathcal{U}$，我们有

$$\mathsf{transport}^{A+B}(p, \mathsf{inl}(a)) = \mathsf{inl}(\mathsf{transport}^{A}(p, a)),$$
$$\mathsf{transport}^{A+B}(p, \mathsf{inr}(b)) = \mathsf{inr}(\mathsf{transport}^{B}(p, b)),$$

其中像往常一样，上标中的 $A + B$ 滥用地表示类型族 $x \mapsto A(x) + B(x)$。证明是简单的路径归纳。

## 2.13 自然数

我们用编码-解码方法来刻画自然数的路径空间，它也是正类型。在这种情况下，我们不是固定一个端点，而是一次刻画双边路径空间。因此，恒等的编码是类型族

$$\mathsf{code} : \mathbb{N} \to \mathbb{N} \to \mathcal{U},$$

通过对 $\mathbb{N}$ 的双重递归如下定义：

$$\begin{aligned}
\mathsf{code}(0, 0) &:\equiv \mathbf{1} \\
\mathsf{code}(\mathsf{succ}(m), 0) &:\equiv \mathbf{0} \\
\mathsf{code}(0, \mathsf{succ}(n)) &:\equiv \mathbf{0} \\
\mathsf{code}(\mathsf{succ}(m), \mathsf{succ}(n)) &:\equiv \mathsf{code}(m, n).
\end{aligned}$$

我们也通过递归定义依赖函数 $r : \prod_{(n:\mathbb{N})} \mathsf{code}(n,n)$，有

$$r(0) :\equiv \star$$
$$r(\mathsf{succ}(n)) :\equiv r(n).$$

**Theorem 2.13.1.** 对所有 $m,n:\mathbb{N}$ 我们有 $(m=n) \simeq \mathsf{code}(m,n)$。

证明. 我们定义

$$\mathsf{encode} : \prod_{m,n:\mathbb{N}} (m=n) \to \mathsf{code}(m,n)$$

通过传输，$\mathsf{encode}(m,n,p) :\equiv \mathsf{transport}^{\mathsf{code}(m,-)}(p, r(m))$。（我们也可以直接通过路径归纳定义 encode，但用传输定义通常使后续计算更容易。）而我们定义

$$\mathsf{decode} : \prod_{m,n:\mathbb{N}} \mathsf{code}(m,n) \to (m=n)$$

通过对 $m,n$ 的双重归纳。当 $m$ 和 $n$ 都是 0 时，我们需要函数 $\mathbf{1} \to (0=0)$，我们定义它把所有东西映到 $\mathsf{refl}_0$。当 $m$ 是后继而 $n$ 是 0，或反过来，定义域 $\mathsf{code}(m,n)$ 是 $\mathbf{0}$，所以 $\mathbf{0}$ 的消除子就够了。当两者都是后继时，我们可以定义 $\mathsf{decode}(\mathsf{succ}(m),\mathsf{succ}(n))$ 为复合

$$\mathsf{code}(\mathsf{succ}(m),\mathsf{succ}(n)) \equiv \mathsf{code}(m,n) \xrightarrow{\mathsf{decode}(m,n)} (m=n) \xrightarrow{\mathsf{ap}_{\mathsf{succ}}} (\mathsf{succ}(m)=\mathsf{succ}(n)).$$

接下来我们证明 $\mathsf{encode}(m,n)$ 和 $\mathsf{decode}(m,n)$ 对所有 $m,n$ 是拟逆。

一方面，如果我们从 $p : m=n$ 开始，则通过对 $p$ 的归纳只需证明

$$\mathsf{decode}(n,n,\mathsf{encode}(n,n,\mathsf{refl}_n)) = \mathsf{refl}_n.$$

但 $\mathsf{encode}(n,n,\mathsf{refl}_n) \equiv r(n)$，所以只需证明 $\mathsf{decode}(n,n,r(n)) = \mathsf{refl}_n$。我们可以通过对 $n$ 的归纳证明这一点。如果 $n \equiv 0$，则 $\mathsf{decode}(0,0,r(0)) = \mathsf{refl}_0$ 由 decode 的定义。在后继的情况，由归纳假设我们有 $\mathsf{decode}(n,n,r(n)) = \mathsf{refl}_n$，所以只需观察 $\mathsf{ap}_{\mathsf{succ}}(\mathsf{refl}_n) \equiv \mathsf{refl}_{\mathsf{succ}(n)}$。

另一方面，如果我们从 $c : \mathsf{code}(m,n)$ 开始，则我们对 $m$ 和 $n$ 进行双重归纳。如果两者都是 0，则 $\mathsf{decode}(0,0,c) \equiv \mathsf{refl}_0$，而 $\mathsf{encode}(0,0,\mathsf{refl}_0) \equiv$

$r(0) \equiv \star$。因此，只需回忆第 2.8 节中 $\mathbf{1}$ 的每个居留元都等于 $\star$ 。如果 $m$ 是 0 但 $n$ 是后继，或反过来，则 $c : \mathbf{0}$，所以我们完成了。在两个后继的情况，我们有

$$\begin{aligned}
&\mathsf{encode}(\mathsf{succ}(m), \mathsf{succ}(n), \mathsf{decode}(\mathsf{succ}(m), \mathsf{succ}(n), c)) \\
&\qquad = \mathsf{encode}(\mathsf{succ}(m), \mathsf{succ}(n), \mathsf{ap}_{\mathsf{succ}}(\mathsf{decode}(m, n, c))) \\
&\qquad = \mathsf{transport}^{\mathsf{code}(\mathsf{succ}(m),-)}(\mathsf{ap}_{\mathsf{succ}}(\mathsf{decode}(m, n, c)), r(\mathsf{succ}(m))) \\
&\qquad = \mathsf{transport}^{\mathsf{code}(\mathsf{succ}(m),\mathsf{succ}(-))}(\mathsf{decode}(m, n, c), r(\mathsf{succ}(m))) \\
&\qquad = \mathsf{transport}^{\mathsf{code}(m,-)}(\mathsf{decode}(m, n, c), r(m)) \\
&\qquad = \mathsf{encode}(m, n, \mathsf{decode}(m, n, c)) \\
&\qquad = c
\end{aligned}$$

使用归纳假设。（实际上这个证明比必要的更长；参见练习 3.24。） □

特别地，我们有

$$\mathsf{encode}(\mathsf{succ}(m), 0) : (\mathsf{succ}(m) = 0) \to \mathbf{0} \tag{2.13.2}$$

这表明 “0 不是任何自然数的后继”。我们还有复合

$$\begin{aligned}
&(\mathsf{succ}(m) = \mathsf{succ}(n)) \xrightarrow{\mathsf{encode}} \\
&\qquad \mathsf{code}(\mathsf{succ}(m), \mathsf{succ}(n)) \equiv \mathsf{code}(m, n) \xrightarrow{\mathsf{decode}} (m = n)
\end{aligned} \tag{2.13.3}$$

这表明函数 succ 是单射的。

我们将在第 5 章 和第 6 章中研究更一般的正类型。在第 8 章中，我们将看到这里用来刻画余积和 $\mathbb{N}$ 的恒等类型的同样技术也可以用来计算球面的高阶同伦群。

## 2.14 例子：结构的相等

我们现在考虑一个例子来说明类型上的群胚结构与类型构造子之间的相互作用。在导论中我们提到泛等性的一个优点是两个同构的东西是可互换的，意思是涉及一个的每个性质或构造也适用于另一个。常见的 “记号滥

用”变成形式上为真。泛等性本身说等价的类型相等，因此可互换，这包括例如等同同构集合的常见做法。此外，当我们把其他数学对象定义为配备了结构或性质的集合，甚至一般类型时，我们可以从泛等性推导出它们的正确相等概念。我们将在第 9 章中用一个重要例子说明这一点，在那里我们以范畴的相等是等价、函子的相等是自然同构等方式定义范畴论的基本概念。特别参见??。在本节中，我们描述一个来自代数的非常简单的例子。

为简单起见，我们用*半群*作为例子，其中半群是配备了结合的 “乘法” 运算的类型。同样的思想适用于其他代数结构，如幺半群、群和环。回忆第 1.6 节 和第 1.11 节，一种数学结构的定义应该被解释为把这种结构的类型定义为某个迭代 $\Sigma$-类型。在半群的情况下这产生如下。

**Definition 2.14.1.** 给定类型 $A$,载体为 $A$ 的**半群结构**类型 $\mathsf{SemigroupStr}(\mathrm{A})$ 定义为

$$\mathsf{SemigroupStr}(A) :\equiv \sum_{(m:A\to A\to A)} \prod_{(x,y,z:A)} m(x,m(y,z)) = m(m(x,y),z).$$

**半群** 是一个类型连同这样的结构：

$$\mathsf{Semigroup} :\equiv \sum_{A:\mathcal{U}} \mathsf{SemigroupStr}(A)$$

在接下来的两个小节中，我们描述泛等性使得处理这样的半群更容易的两种方式。

### 2.14.1 提升等价

当不严格地工作时，人们可能会说集合 $A$ 和 $B$ 之间的双射 “显然” 诱导 $A$ 上的半群结构和 $B$ 上的半群结构之间的同构。用泛等性，这确实是显然的，因为给定类型 $A$ 和 $B$ 之间的等价，我们可以自动从 $A$ 上的半群结构推导出 $B$ 上的半群结构，而且证明这个推导是半群结构的等价。原因是 $\mathsf{SemigroupStr}$ 是类型的族，因此有由 $\mathsf{transport}$ 给出的类型之间路径上的作用：

$$\mathsf{transport}^{\mathsf{SemigroupStr}}(\mathsf{ua}(e)) : \mathsf{SemigroupStr}(A) \to \mathsf{SemigroupStr}(B).$$

此外，这个映射是等价，因为 $\mathsf{transport}^{C}(\alpha)$ 总是等价，其逆是 $\mathsf{transport}^{C}(\alpha^{-1})$，参见?? 和定理 2.1.4。

虽然泛等公理确保这个映射存在，我们需要使用前面几节证明的关于 transport 的事实来计算它实际做什么。设 $(m, a)$ 是 $A$ 上的半群结构，我们研究由

$$\mathsf{transport}^{\mathsf{SemigroupStr}}(\mathsf{ua}(e), (m, a))$$

给出的 $B$ 上的诱导半群结构。首先，因为 SemigroupStr(X) 被定义为 $\Sigma$-类型，由2.7.4，

$$\mathsf{transport}^{\mathsf{SemigroupStr}}(\mathsf{ua}(e), (m, a)) = (m', a')$$

其中 $m'$ 是 $B$ 上的诱导乘法运算

$$\begin{aligned} &m' : B \to B \to B \\ &m'(b_1, b_2) :\equiv \mathsf{transport}^{X \mapsto (X \to X \to X)}(\mathsf{ua}(e), m)(b_1, b_2) \end{aligned}$$

而 $a'$ 是 $m'$ 结合的诱导证明。再由2.7.4，我们有

$$\begin{aligned} &a' : \mathsf{Assoc}(B, m') \\ &a' :\equiv \mathsf{transport}^{(X,m) \mapsto \mathsf{Assoc}(X,m)}((\mathsf{pair}^{=}(\mathsf{ua}(e), \mathsf{refl}_{m'})), a), \end{aligned} \tag{2.14.2}$$

其中 $\mathsf{Assoc}(X, m)$ 是类型 $\prod_{(x,y,z:X)} m(x, m(y, z)) = m(m(x, y), z)$。由函数外延性，只需研究 $m'$ 应用于参数 $b_1, b_2 : B$ 时的行为。通过两次应用 (2.9.4)，我们有 $m'(b_1, b_2)$ 等于

$$\begin{aligned} &\mathsf{transport}^{X \mapsto X}\big(\mathsf{ua}(e), \\ &\qquad m(\mathsf{transport}^{X \mapsto X}(\mathsf{ua}(e)^{-1}, b_1), \mathsf{transport}^{X \mapsto X}(\mathsf{ua}(e)^{-1}, b_2))\big). \end{aligned}$$

然后，因为 ua 是 $\mathsf{transport}^{X \mapsto X}$ 的拟逆，这等于

$$e(m(e^{-1}(b_1), e^{-1}(b_2))).$$

因此，给定 $B$ 的两个元素，诱导乘法 $m'$ 用等价 $e$ 把它们送到 $A$，在 $A$ 中乘它们，然后用 $e$ 把结果带回 $B$，正如人们所期望的。

此外，虽然我们不展示证明，可以计算出 $m'$ 结合的诱导证明（参见 (2.14.2)）等于一个函数，把 $b_1, b_2, b_3 : B$ 送到由以下步骤给出的路径：

$$\begin{aligned} m'(m'(b_1,b_2),b_3) &= e(m(e^{-1}(m'(b_1,b_2)),e^{-1}(b_3))) \\ &= e(m(e^{-1}(e(m(e^{-1}(b_1),e^{-1}(b_2)))),e^{-1}(b_3))) \\ &= e(m(m(e^{-1}(b_1),e^{-1}(b_2)),e^{-1}(b_3))) \\ &= e(m(e^{-1}(b_1),m(e^{-1}(b_2),e^{-1}(b_3)))) \\ &= e(m(e^{-1}(b_1),e^{-1}(e(m(e^{-1}(b_2),e^{-1}(b_3)))))) \\ &= e(m(e^{-1}(b_1),e^{-1}(m'(b_2,b_3)))) \\ &= m'(b_1,m'(b_2,b_3)). \end{aligned} \tag{2.14.3}$$

这些步骤使用 $m$ 结合的证明 $a$ 和 $e$ 的逆律。从代数角度看，研究一个运算结合的证明的恒等可能看起来奇怪，但如果我们把 $A$ 和 $B$ 想成一般空间，路径之间有非平凡的同伦，这就有意义了。在第 3 章中，我们将引入集合的概念，它是只有平凡同伦的类型，如果我们考虑集合上的半群结构，则任何两个这样的结合性证明自动相等。

### 2.14.2 半群的相等

使用前面几节讨论的路径空间方程，我们可以研究两个半群何时相等。给定半群 $(A, m, a)$ 和 $(B, m', a')$，由定理 2.7.2，路径类型

$$(A, m, a) =_{\mathsf{Semigroup}} (B, m', a')$$

等于对的类型

$$\begin{aligned} &p_1 : A =_{\mathcal{U}} B \qquad \text{和} \\ &p_2 : \mathsf{transport}^{\mathsf{SemigroupStr}}(p_1, (m, a)) = (m', a'). \end{aligned}$$

由泛等性，$p_1$ 是某个等价 $e$ 的 $\mathsf{ua}(e)$。由定理 2.7.2、函数外延性和上面对类型族 SemigroupStr 中传输的分析，$p_2$ 等价于一对证明，第一个证明

$$\prod_{y_1,y_2:B} e(m(e^{-1}(y_1),e^{-1}(y_2))) = m'(y_1,y_2) \tag{2.14.4}$$

第二个证明 $a'$ 等于 (2.14.3) 中从 $a$ 构造的诱导结合性证明。但通过逆的消去，(2.14.4) 等价于

$$\prod_{x_1,x_2:A} e(m(x_1,x_2)) = m'(e(x_1),e(x_2)).$$

这说 $e$ 与二元运算交换，意思是它把 $A$ 中的乘法（即 $m$）带到 $B$ 中的乘法（即 $m'$）。关于 $a$ 和 $a'$ 的方程也可以做类似的重新安排。因此，半群的相等恰好由载体类型上与半群结构交换的等价组成。

对于一般类型，结合性的证明被认为是半群结构的一部分。然而，如果我们限制到集合类的类型（再次参见第 3 章），关于 $a$ 和 $a'$ 的方程平凡为真。此外，在这种情况下，集合之间的等价恰好是双射。因此，我们得到了*半群同构*的标准定义：载体集合上保持乘法运算的双射。也可以使用范畴论的同构定义，通过定义*半群同态*为保持乘法的映射，并得出半群的相等与两个互逆同态相同的结论；但我们不在这里展示细节；参见??。

结论是，由于泛等性，半群精确地在它们作为代数结构同构时相等。正如我们将在?? 中看到的，这个结论更一般地适用：在同伦类型论中，数学结构的所有构造自动尊重同构，无需任何繁琐的证明或记号滥用。

## 2.15 泛性质

通过组合前面几节描述的路径计算规则，我们可以证明各种类型构造运算满足预期的泛性质，以同伦方式解释为等价。例如，给定类型 $X, A, B$，我们有函数

$$(X \to A \times B) \to (X \to A) \times (X \to B) \tag{2.15.1}$$

定义为 $f \mapsto (\mathsf{pr}_1 \circ f, \mathsf{pr}_2 \circ f)$。

**Theorem 2.15.2.** (2.15.1) 是等价。

*证明.* 我们定义拟逆把 $(g,h)$ 映到 $\lambda x.\,(g(x),h(x))$。（从技术上讲，我们使用了笛卡尔积 $(X \to A) \times (X \to B)$ 的归纳原理，归约到对的情况。从现在起我们经常不显式提及就应用这个原理。）

现在给定 $f : X \to A \times B$，往返复合产生函数

$$\lambda x.\,(\mathsf{pr}_1(f(x)), \mathsf{pr}_2(f(x))). \tag{2.15.3}$$

由定理 2.6.2，对任意 $x : X$ 我们有 $(\mathsf{pr}_1(f(x)), \mathsf{pr}_2(f(x))) = f(x)$。因此，由函数外延性，函数 (2.15.3) 等于 $f$。

另一方面，给定 $(g, h)$，往返复合产生对 $(\lambda x.\, g(x), \lambda x.\, h(x))$。由函数的唯一性原理，这（判断上）等于 $(g, h)$。 □

实际上，我们也有这个泛性质的依赖类型版本。假设给定类型 $X$ 和类型族 $A, B : X \to \mathcal{U}$。则我们有函数

$$\Big(\prod_{x:X} (A(x) \times B(x))\Big) \to \Big(\prod_{x:X} A(x)\Big) \times \Big(\prod_{x:X} B(x)\Big) \tag{2.15.4}$$

如前定义为 $f \mapsto (\mathsf{pr}_1 \circ f, \mathsf{pr}_2 \circ f)$。

**Theorem 2.15.5.** (2.15.4) 是等价。

证明. 留给读者。 □

正如 Σ-类型是笛卡尔积的推广，它们满足这个泛性质的推广版本。直接跳到依赖类型版本，假设我们有类型 $X$ 和类型族 $A : X \to \mathcal{U}$ 和 $P : \prod_{(x:X)} A(x) \to \mathcal{U}$。则我们有函数

$$\Big(\prod_{x:X} \sum_{(a:A(x))} P(x, a)\Big) \to \Big(\sum_{(g:\prod_{(x:X)} A(x))} \prod_{(x:X)} P(x, g(x))\Big). \tag{2.15.6}$$

注意如果我们对某个 $B : X \to \mathcal{U}$ 有 $P(x, a) :\equiv B(x)$，则 (2.15.6) 归约为 (2.15.4)。

**Theorem 2.15.7.** (2.15.6) 是等价。

证明. 如前，我们定义拟逆把 $(g, h)$ 映到函数 $\lambda x.\, (g(x), h(x))$。现在给定 $f : \prod_{(x:X)} \sum_{(a:A(x))} P(x, a)$，往返复合产生函数

$$\lambda x.\, (\mathsf{pr}_1(f(x)), \mathsf{pr}_2(f(x))). \tag{2.15.8}$$

现在对任意 $x : X$，由定理 2.7.3（Σ-类型的唯一性原理）我们有

$$(\mathsf{pr}_1(f(x)), \mathsf{pr}_2(f(x))) = f(x).$$

因此，由函数外延性，(2.15.8) 等于 $f$。另一方面，给定 $(g, h)$，往返复合产生 $(\lambda x.\, g(x), \lambda x.\, h(x))$，如前它判断上等于 $(g, h)$。 □

这值得注意，因为 (2.15.6) 的命题即类型解释是 “选择公理”。如果我们把 $\Sigma$ 读作 “存在” 把 $\Pi$（有时）读作 “对所有”，我们可以把：

- $\prod_{(x:X)} \sum_{(a:A(x))} P(x,a)$ 读作 “对所有 $x : X$ 存在 $a : A(x)$ 使得 $P(x,a)$”，和
- $\sum_{(g:\prod_{(x:X)} A(x))} \prod_{(x:X)} P(x,g(x))$ 读作 “存在选择函数 $g : \prod_{(x:X)} A(x)$ 使得对所有 $x : X$ 有 $P(x,g(x))$”。

因此，定理 2.15.7说不仅选择公理是 “真的”，它的前件实际上等价于它的结论。（另一方面，经典数学家可能发现 (2.15.6) 不携带选择公理的通常含义，因为我们已经指定了 $g$ 的值，没有剩余的选择要做。我们将在第 3.8 节回到这一点。）

上述对类型的泛性质是 “映入” 的，这从积的范畴论概念中很熟悉。然而，对类型也有 “映出” 的泛性质，可能看起来不那么熟悉。在笛卡尔积的情况，非依赖版本简单地表达笛卡尔闭伴随：

$$\big((A \times B) \to C\big) \;\simeq\; \big(A \to (B \to C)\big).$$

这的依赖版本对类型族 $C : A \times B \to \mathcal{U}$ 表述：

$$\Big(\prod_{w:A\times B} C(w)\Big) \;\simeq\; \Big(\prod_{(x:A)} \prod_{(y:B)} C(x,y)\Big).$$

这里从右到左的函数就是 $A \times B$ 的归纳原理，而从左到右是在对上求值。我们把证明它们是拟逆留给读者。对 $\Sigma$-类型也有一个版本：

$$\Big(\prod_{w:\sum_{(x:A)} B(x)} C(w)\Big) \;\simeq\; \Big(\prod_{(x:A)} \prod_{(y:B(x))} C(x,y)\Big). \tag{2.15.9}$$

再次，从右到左的函数是归纳原理。

一些其他归纳原理也是这类泛性质的一部分。例如，路径归纳是如下等价的从右到左方向：

$$\Big(\prod_{(x:A)} \prod_{(p:a=x)} B(x,p)\Big) \;\simeq\; B(a, \mathsf{refl}_a) \tag{2.15.10}$$

对任意 $a : A$ 和类型族 $B : \prod_{(x:A)}(a = x) \to \mathcal{U}$。然而，带递归的归纳类型，如自然数，有更复杂的泛性质；参见第 5 章。

因为定理 2.15.2表达笛卡尔积的通常泛性质（在适当的同伦论意义上），有范畴论倾向的读者很可能想知道类型的其他极限和余极限。在?? 中我们要求读者证明余积类型 $A + B$ 也有预期的泛性质，而 $\mathbf{1}$（终对象）和 $\mathbf{0}$（初对象）的零元情况很容易。

对于拉回，预期的显式构造有效：给定 $f : A \to C$ 和 $g : B \to C$，我们定义

$$A \times_C B :\equiv \sum_{(a:A)} \sum_{(b:B)} (f(a) = g(b)). \tag{2.15.11}$$

在?? 中我们要求读者验证这一点。一些更一般的同伦极限可以用类似方式构造，但对于余极限我们将需要新的成分；参见第 6 章。

# 第 3 章　集合与逻辑

类型论，无论是形式的还是非形式的，都是一套操作类型及其元素的规则。但当用自然语言非形式地书写数学时，我们通常使用熟悉的词汇，特别是逻辑连接词如 “和” 和 “或”，以及逻辑量词如 “对于所有” 和 “存在”。与集合论相比，类型论为我们提供了不止一种方式将这些英语短语看作类型上的运算。这种潜在的歧义需要通过设定局部或全局约定、向非形式数学引入新的标注，或两者兼用来解决。这需要一些适应，但好处在于，由于类型论允许对逻辑进行这种更精细的分析，我们可以更忠实地表示数学，比集合论基础中的 “语言滥用” 更少。在本章中，我们将解释所涉及的问题，并证明我们所做的选择是合理的。

## 3.1　集合与 $n$-类型

为了解释类型论逻辑与集合论逻辑之间的联系，在类型论中有一个集合的概念是有帮助的。虽然一般的类型表现得像空间或高阶广群，但有一类子类型表现得更像传统集合论系统中的集合。从范畴论的角度，我们可以考虑离散广群，它由一个对象集合决定，只有恒等态射作为高阶态射；而从拓扑学的角度，我们可以考虑具有离散拓扑的空间。更一般地，我们可以考虑等价于这类空间的广群或空间；因为我们在类型论中所做的一切都是关于同伦的，我们不能指望区分它们。

直观上，我们期望一个类型在这种意义上 “是一个集合”，如果它没有高阶同伦信息：任何两条平行路径都相等（关于同伦），对于所有维度的平行高阶路径也是如此。幸运的是，因为同伦类型论中的一切都自动是函子性/连续的，结果只需要在最底层要求这一点就足够了。

**Definition 3.1.1.** 如果对于所有 $x, y : A$ 和所有 $p, q : x = y$，我们有 $p = q$，则类型 $A$ 是一个**集合**。

更精确地，命题 $\mathsf{isSet}(A)$ 被定义为类型

$$\mathsf{isSet}(A) :\equiv \prod_{(x,y:A)} \prod_{(p,q:x=y)} (p = q).$$

正如在第 1.1 节中提到的，同伦类型论中的集合不同于 ZF 集合论中的集合，因为没有全局的 “属于” 谓词 $\in$。它们更像结构数学和范畴论中使用的集合，其元素是 “抽象点”，我们用函数和关系赋予它们结构。这就是我们将它们用作大多数基于集合的数学的基础系统所需要的全部；我们将在第 10 章中看到一些例子。

哪些类型是集合？在第 7 章中我们将更深入地研究这个问题的更一般形式，但现在我们可以观察一些简单的例子。

*Example* 3.1.2. 类型 $\mathbf{1}$ 是一个集合。因为根据定理 2.8.1，对于任何 $x, y : \mathbf{1}$，类型 $(x = y)$ 等价于 $\mathbf{1}$ 。由于 $\mathbf{1}$ 的任意两个元素相等，这意味着 $x = y$ 的任意两个元素也相等。

*Example* 3.1.3. 类型 $\mathbf{0}$ 是一个集合，因为给定任何 $x, y : \mathbf{0}$，我们可以通过 $\mathbf{0}$ 的归纳原理推导出任何我们想要的结论。

*Example* 3.1.4. 自然数类型 $\mathbb{N}$ 也是一个集合。这由定理 2.13.1得出，因为所有等式类型 $x =_{\mathbb{N}} y$ 都等价于 $\mathbf{1}$ 或 $\mathbf{0}$ ，而 $\mathbf{1}$ 或 $\mathbf{0}$ 的任意两个居留者都相等。我们将在第 7 章中看到这个事实的另一个证明。

我们目前考虑的大多数类型构造子也保持集合性。

*Example* 3.1.5. 如果 $A$ 和 $B$ 是集合，那么 $A \times B$ 也是。因为给定 $x, y : A \times B$ 和 $p, q : x = y$，根据定理 2.6.2，我们有 $p = \mathsf{pair}^{=}(\mathsf{ap}_{\mathsf{pr}_1}(p), \mathsf{ap}_{\mathsf{pr}_2}(p))$ 和 $q = \mathsf{pair}^{=}(\mathsf{ap}_{\mathsf{pr}_1}(q), \mathsf{ap}_{\mathsf{pr}_2}(q))$。但 $\mathsf{ap}_{\mathsf{pr}_1}(p) = \mathsf{ap}_{\mathsf{pr}_1}(q)$ 因为 $A$ 是集合，$\mathsf{ap}_{\mathsf{pr}_2}(p) = \mathsf{ap}_{\mathsf{pr}_2}(q)$ 因为 $B$ 是集合；因此 $p = q$。

类似地，如果 $A$ 是集合且 $B : A \to \mathcal{U}$ 使得每个 $B(x)$ 是集合，则 $\sum_{(x:A)} B(x)$ 是集合。

*Example* 3.1.6. 如果 $A$ 是任意类型且 $B : A \to \mathcal{U}$ 使得每个 $B(x)$ 是集合，则类型 $\prod_{(x:A)} B(x)$ 是集合。因为假设 $f, g : \prod_{(x:A)} B(x)$ 且 $p, q : f = g$。根据函数外延性，我们有

$$p = \mathsf{funext}(x \mapsto \mathsf{happly}(p, x)) \quad \text{和} \quad q = \mathsf{funext}(x \mapsto \mathsf{happly}(q, x)).$$

但对于任何 $x : A$，我们有

$$\mathsf{happly}(p, x) : f(x) = g(x) \qquad \text{和} \qquad \mathsf{happly}(q, x) : f(x) = g(x),$$

所以由于 $B(x)$ 是集合，我们有 $\mathsf{happly}(p, x) = \mathsf{happly}(q, x)$。现在再次使用函数外延性，依赖函数 $(x \mapsto \mathsf{happly}(p, x))$ 和 $(x \mapsto \mathsf{happly}(q, x))$ 相等，因此（应用 $\mathsf{ap}_{\mathsf{funext}}$）$p$ 和 $q$ 也相等。

更多例子，参见练习 3.2 和练习 3.3。关于同伦类型论中所有集合的子系统（范畴）的更系统的研究，参见第 10 章。

集合只是所谓同伦 $n$-类型阶梯的第一级。下一级由 1-类型组成，它们类似于范畴论中的 1-广群。集合（我们也可以称之为 0-类型）的定义性质是它没有非平凡路径。类似地，1-类型的定义性质是它没有非平凡的路径之间的路径：

**Definition 3.1.7.** 如果对于所有 $x, y : A$ 和 $p, q : x = y$ 以及 $r, s : p = q$，我们有 $r = s$，则类型 $A$ 是一个 **1-类型**。

类似地，我们可以定义 2-类型、3-类型等等。我们将在第 7 章中归纳地定义 $n$-类型的一般概念，并研究不同 $n$ 值的 $n$-类型之间的关系。

然而，目前记住两个事实是有用的。首先，层级是向上封闭的：如果 $A$ 是 $n$-类型，那么 $A$ 是 $(n+1)$-类型。例如：

**Lemma 3.1.8.** *如果 $A$ 是集合（即 $\mathsf{isSet}(A)$ 有居留者），那么 $A$ 是 1-类型。*

*证明.* 假设 $f : \mathsf{isSet}(A)$；则对于任何 $x, y : A$ 和 $p, q : x = y$，我们有 $f(x, y, p, q) : p = q$。固定 $x$、$y$ 和 $p$，定义 $g : \prod_{(q:x=y)}(p = q)$ 为 $g(q) :\equiv f(x, y, p, q)$。则对于任何 $r : q = q'$，我们有 $\mathsf{apd}_g(r) : r_*(g(q)) = g(q')$。因此根据推论 2.11.2，我们有 $g(q) \cdot r = g(q')$。

特别地，假设给定 $x, y, p, q$ 和 $r, s : p = q$，如定义 3.1.7中那样，并如上定义 $g$。则 $g(p) \cdot r = g(q)$ 且 $g(p) \cdot s = g(q)$，因此通过消去 $r = s$。 □

其次，这种按层级对类型的分层不是退化的，即并非所有类型都是集合：

*Example* 3.1.9. 宇宙 $\mathcal{U}$ 不是集合。为了证明这一点，只需展示一个类型 $A$ 和一条路径 $p : A = A$，它不等于 $\mathsf{refl}_A$。取 $A = \mathbf{2}$，令 $f : A \to A$ 定义为 $f(0_{\mathbf{2}}) :\equiv 1_{\mathbf{2}}$ 和 $f(1_{\mathbf{2}}) :\equiv 0_{\mathbf{2}}$。则对所有 $x$ 有 $f(f(x)) = x$（通过简单的情形分析），所以 $f$ 是一个等价。因此，通过泛等性，$f$ 产生一条路径 $p : A = A$。

如果 $p$ 等于 $\mathsf{refl}_A$，那么（再次通过泛等性）$f$ 将等于 $A$ 的恒等函数。但这将意味着 $0_{\mathbf{2}} = 1_{\mathbf{2}}$，与注记 2.12.6矛盾。

在第 6 章 和第 8 章中，我们将证明对于任何 $n$，存在不是 $n$-类型的类型。

注意 $A$ 是 1-类型当且仅当对于任何 $x, y : A$，恒等类型 $x =_A y$ 是集合。（因此，定理 3.1.8也可以等价地理解为说集合的恒等类型也是集合。）这将是我们在第 7 章中给出的 $n$-类型递归定义的基础。

我们还可以将这种刻画“向下”从集合扩展。即，类型 $A$ 是集合当且仅当对于任何 $x, y : A$，$x =_A y$ 的任意两个元素相等。由于集合等价于 0-类型，如果一个类型具有后一性质（它的任意两个元素相等），自然称它为 $(-1)$-*类型*。这样的类型可以被看作*狭义上的命题*，它们的研究正是通常所称的“逻辑”；它将在本章的其余部分占据我们的注意力。

## 3.2 命题即类型？

到目前为止，我们一直遵循第 1.11 节中描述的直接的“命题即类型”哲学，根据这种哲学，英语短语如“存在 $x : A$ 使得 $P(x)$”被相应的类型如 $\sum_{(x:A)} P(x)$ 解释，命题的证明被视为判断某个特定元素居留于该类型。然而，我们也已经看到了一些方式，其中这种解读产生的“逻辑”对经典数学家来说似乎不太熟悉。例如，在定理 2.15.7中我们看到声明

$$\begin{array}{l}\text{“如果对所有 } x : X \text{ 存在 } a : A(x) \text{ 使得 } P(x, a)\text{，则存在一个函}\\ \text{数 } g : \prod_{(x:X)} A(x) \text{ 使得对所有 } x : X \text{ 我们有 } P(x, g(x))\text{”}\end{array} \tag{3.2.1}$$

在这种解读下总是真的，它看起来像经典的选择公理。这是命题即类型逻辑的一个值得注意的、通常也很有用的特性，但它也说明了它与逻辑的经典解释有多大的不同，在经典解释下选择公理不是一个逻辑真理，而是一个额外的“公理”。

另一方面，我们现在也可以证明看起来像经典双重否定律和排中律的相应声明与泛等公理不相容。

**Theorem 3.2.2.** *并非对所有 $A : \mathcal{U}$ 我们有 $\neg(\neg A) \to A$。*

*证明.* 回忆 $\neg A \equiv (A \to \mathbf{0})$。我们也将“并非……”读作算子 $\neg$。因此，为了证明这个声明，假设给定某个 $f : \prod_{(A:\mathcal{U})}(\neg\neg A \to A)$ 并构造 $\mathbf{0}$ 的一个元素就足够了。

以下证明的思想是观察到 $f$，像类型论中的任何函数一样，是“连续的”。通过泛等性，这意味着 $f$ 对于类型的等价是自然的。从这一点，以及一个无不动点的自等价，我们将能够得出矛盾。

令 $e : \mathbf{2} \simeq \mathbf{2}$ 为由 $e(1_{\mathbf{2}}) :\equiv 0_{\mathbf{2}}$ 和 $e(0_{\mathbf{2}}) :\equiv 1_{\mathbf{2}}$ 定义的等价，如定理 3.1.9中那样。令 $p : \mathbf{2} = \mathbf{2}$ 为通过泛等性对应于 $e$ 的路径，即 $p :\equiv \mathsf{ua}(e)$。则我们有 $f(\mathbf{2}) : \neg\neg\mathbf{2} \to \mathbf{2}$ 和

$$\mathsf{apd}_f(p) : \mathsf{transport}^{A\mapsto(\neg\neg A\to A)}(p, f(\mathbf{2})) = f(\mathbf{2}).$$

因此，对于任何 $u : \neg\neg\mathbf{2}$，我们有

$$\mathsf{happly}(\mathsf{apd}_f(p), u) : \mathsf{transport}^{A\mapsto(\neg\neg A\to A)}(p, f(\mathbf{2}))(u) = f(\mathbf{2})(u).$$

现在根据 (2.9.4)，在类型族 $A \mapsto (\neg\neg A \to A)$ 中沿 $p$ 传输 $f(\mathbf{2}) : \neg\neg\mathbf{2} \to \mathbf{2}$ 等于这样一个函数：它在类型族 $A \mapsto \neg\neg A$ 中沿 $p^{-1}$ 传输其参数，应用 $f(\mathbf{2})$，然后在类型族 $A \mapsto A$ 中沿 $p$ 传输结果：

$$\mathsf{transport}^{A\mapsto(\neg\neg A\to A)}(p, f(\mathbf{2}))(u) = \mathsf{transport}^{A\mapsto A}(p, f(\mathbf{2})(\mathsf{transport}^{A\mapsto\neg\neg A}(p^{-1}, u))).$$

然而，通过函数外延性，任意两点 $u, v : \neg\neg\mathbf{2}$ 相等，因为对于任何 $x : \neg\mathbf{2}$ 我们有 $u(x) : \mathbf{0}$，从而可以推导任何结论，特别是 $u(x) = v(x)$。因此，我们

有 $\mathsf{transport}^{A\mapsto\neg\neg A}(p^{-1}, u) = u$，所以从 $\mathsf{happly}(\mathsf{apd}_f(p), u)$ 我们得到等式

$$\mathsf{transport}^{A\mapsto A}(p, f(\mathbf{2})(u)) = f(\mathbf{2})(u).$$

最后，如第 2.10 节中讨论的，在类型族 $A \mapsto A$ 中沿路径 $p \equiv \mathsf{ua}(e)$ 传输等价于应用等价 $e$；因此我们有

$$e(f(\mathbf{2})(u)) = f(\mathbf{2})(u). \tag{3.2.3}$$

然而，我们也可以证明

$$\prod_{x:\mathbf{2}} \neg(e(x) = x). \tag{3.2.4}$$

这由对 $x$ 的情形分析得出：两种情况都直接从 $e$ 的定义和 $0_{\mathbf{2}} \neq 1_{\mathbf{2}}$（注记 2.12.6）得出。因此，将 (3.2.4) 应用于 $f(\mathbf{2})(u)$ 和 (3.2.3)，我们得到 $\mathbf{0}$ 的一个元素。 □

*Remark* 3.2.5. 特别地，这意味着不可能有 Hilbert 式的“选择算子”从每个非空类型中选择一个元素。关键点是没有这样的算子可以是自然的，而在泛等公理下，所有作用于类型的函数必须对于等价是自然的。

*Remark* 3.2.6. 然而，对于任何 $A$，$\neg\neg\neg A \to \neg A$ 仍然成立；参见??。

**Corollary 3.2.7.** *并非对所有 $A : \mathcal{U}$ 我们有 $A + (\neg A)$。*

*证明.* 假设我们有 $g : \prod_{(A:\mathcal{U})}(A + (\neg A))$。我们将证明 $\prod_{(A:\mathcal{U})}(\neg\neg A \to A)$，从而可以应用定理 3.2.2。因此，假设 $A : \mathcal{U}$ 和 $u : \neg\neg A$；我们想要构造 $A$ 的一个元素。

现在 $g(A) : A + (\neg A)$，所以通过情形分析，我们可以假设要么 $g(A) \equiv \mathsf{inl}(a)$ 对某个 $a : A$，要么 $g(A) \equiv \mathsf{inr}(w)$ 对某个 $w : \neg A$。在第一种情况下，我们有 $a : A$，而在第二种情况下我们有 $u(w) : \mathbf{0}$，从而可以得到任何我们想要的东西（如 $A$）。因此，在两种情况下我们都有 $A$ 的一个元素，如所愿。 □

因此，如果我们想假设泛等公理（当然，我们想这样做）并且仍然给自己留下经典推理的选项（这也是可取的），我们不能使用未修改的命题即类型原则将所有非形式数学陈述解释到类型论中，因为那样排中律将是假的。

然而，我们也不想完全抛弃命题即类型，因为它有许多好的性质（如简单性、构造性和可计算性）。我们现在讨论一种解决这些问题的命题即类型的修改；在第 3.10 节中，我们将回到何时使用哪种逻辑的问题。

## 3.3 纯粹命题

我们已经看到命题即类型逻辑既有好的性质也有不好的性质。两者有一个共同的原因：当类型被视为命题时，它们可以包含比单纯的真或假更多的信息，所有对它们的“逻辑”构造都必须尊重这些额外信息。这表明我们可以通过将注意力限制在不包含比真值更多信息的类型上，并只将这些类型视为逻辑命题，来获得一种更传统的逻辑。

这样的类型 $A$ 如果有居留者则为“真”，如果它的居留导致矛盾（即如果 $\neg A \equiv (A \to \mathbf{0})$ 有居留者）则为“假”。为了获得更传统的逻辑，我们想要避免的是将那些给出其元素会比单纯知道该类型有居留者提供更多信息的类型视为逻辑命题。例如，如果我们被给予 $\mathbf{2}$ 的一个元素，那么我们收到的信息比 $\mathbf{2}$ 包含某个元素这一事实更多。实际上，我们恰好多收到一比特信息：我们知道我们被给予的是 $\mathbf{2}$ 的哪个元素。相比之下，如果我们被给予 $\mathbf{1}$ 的一个元素，那么我们收到的信息不比 $\mathbf{1}$ 包含一个元素这一事实更多，因为 $\mathbf{1}$ 的任意两个元素彼此相等。这提示了以下定义。

**Definition 3.3.1.** 如果对所有 $x, y : P$ 我们有 $x = y$，则类型 $P$ 是一个**纯粹命题**。

注意，由于我们仍在类型论中做数学，这是类型论中的一个定义，这意味着它是一个类型——或者更确切地说，是一个类型族。具体来说，对于任何 $P : \mathcal{U}$，类型 $\mathsf{isProp}(P)$ 定义为

$$\mathsf{isProp}(P) :\equiv \prod_{x,y:P} (x = y).$$

因此，断言“$P$ 是一个纯粹命题”意味着展示 $\mathsf{isProp}(P)$ 的一个居留者，它是一个连接 $P$ 的任意两个元素的依赖函数。这个函数的连续性/自然性意味着不仅 $P$ 的任意两个元素相等，而且 $P$ 也不包含高阶同伦信息。

**Lemma 3.3.2.** 如果 $P$ 是一个纯粹命题且 $x_0 : P$，则 $P \simeq \mathbf{1}$。

证明. 定义 $f : P \to \mathbf{1}$ 为 $f(x) :\equiv \star$，$g : \mathbf{1} \to P$ 为 $g(u) :\equiv x_0$。结论由下一个引理得出，以及通过定理 2.8.1观察到 $\mathbf{1}$ 是一个纯粹命题。 □

**Lemma 3.3.3.** 如果 $P$ 和 $Q$ 是纯粹命题且 $P \to Q$ 和 $Q \to P$，则 $P \simeq Q$。

证明. 假设给定 $f : P \to Q$ 和 $g : Q \to P$。则对于任何 $x : P$，我们有 $g(f(x)) = x$，因为 $P$ 是一个纯粹命题。类似地，对于任何 $y : Q$，我们有 $f(g(y)) = y$，因为 $Q$ 是一个纯粹命题；因此 $f$ 和 $g$ 是拟逆。 □

也就是说，正如在第 1.11 节中承诺的，如果两个纯粹命题逻辑等价，那么它们是等价的。

在同伦论中，与 $\mathbf{1}$ 同伦等价的空间被称为可缩的。因此，任何有居留者的纯粹命题都是可缩的（另见第 3.11 节）。另一方面，无居留者的类型 $\mathbf{0}$ 也是（平凡地）一个纯粹命题。在经典数学中，至少这些是仅有的两种可能性。

纯粹命题也被称为次终对象（如果从范畴论的角度思考）、次单元素集（如果从集合论的角度思考）或 $h$-命题。第 3.1 节中的讨论表明我们也应该称它们为 $(-1)$-类型；我们将在第 7 章中回到这一点。形容词 “纯粹” 强调虽然任何类型都可以被视为命题（我们通过给出它的一个居留者来证明），但一个纯粹命题的类型不能有用地被视为不止是一个命题：在其真值的见证中不包含额外信息。

注意类型 $A$ 是集合当且仅当对所有 $x, y : A$，恒等类型 $x =_A y$ 是一个纯粹命题。另一方面，通过复制和简化定理 3.1.8的证明，我们有：

**Lemma 3.3.4.** 每个纯粹命题都是集合。

证明. 假设 $f : \mathsf{isProp}(A)$；因此对所有 $x, y : A$ 我们有 $f(x, y) : x = y$。固定 $x : A$ 并定义 $g(y) :\equiv f(x, y)$。则对于任何 $y, z : A$ 和 $p : y = z$，我们有 $\mathsf{apd}_g(p) : p_*(g(y)) = g(z)$。因此根据推论 2.11.2，我们有 $g(y) \cdot p = g(z)$，也就是说 $p = g(y)^{-1} \cdot g(z)$。因此，对于任何 $p, q : x = y$，我们有 $p = g(x)^{-1} \cdot g(y) = q$。 □

特别地，这意味着：

**Lemma 3.3.5.** 对于任何类型 $A$，类型 $\mathsf{isProp}(A)$ 和 $\mathsf{isSet}(A)$ 是纯粹命题。

证明. 假设 $f, g : \mathsf{isProp}(A)$。根据函数外延性，为了证明 $f = g$，只需对任何 $x, y : A$ 证明 $f(x, y) = g(x, y)$。但 $f(x, y)$ 和 $g(x, y)$ 都是 $A$ 中的路径，因此它们相等，因为根据 $f$ 或 $g$，$A$ 是一个纯粹命题，因此根据定理 3.3.4是一个集合。类似地，假设 $f, g : \mathsf{isSet}(A)$，也就是说对所有 $a, b : A$ 和 $p, q : a = b$，我们有 $f(a, b, p, q) : p = q$ 和 $g(a, b, p, q) : p = q$。但因为 $A$ 是集合（根据 $f$ 或 $g$），因此是 1-类型，所以 $f(a, b, p, q) = g(a, b, p, q)$；因此通过函数外延性 $f = g$。 □

我们目前见过另一个例子：第 2.4 节中的条件 (iii)断言对于任何函数 $f$，类型 $\mathsf{isequiv}(f)$ 应该是一个纯粹命题。

## 3.4 经典逻辑与直觉主义逻辑

有了纯粹命题的概念，我们现在可以给出同伦类型论中**排中律**的适当表述：

$$\mathsf{LEM} :\equiv \prod_{A:\mathcal{U}} \Big(\mathsf{isProp}(A) \to (A + \neg A)\Big). \tag{3.4.1}$$

类似地，**双重否定律** 是

$$\prod_{A:\mathcal{U}} \Big(\mathsf{isProp}(A) \to (\neg\neg A \to A)\Big). \tag{3.4.2}$$

这两者也容易证明彼此等价——参见练习 3.18——所以从现在起我们通常只讨论 LEM 。

这种 LEM 的表述避免了定理 3.2.2 和定理 3.2.7的“悖论”，因为 $\mathbf{2}$ 不是一个纯粹命题。为了将它与更一般的命题即类型表述区分开来，我们将后者重命名：

$$\mathsf{LEM}_\infty :\equiv \prod_{A:\mathcal{U}} (A + \neg A).$$

为了强调，适当的版本 (3.4.1) 可以记为 $\mathsf{LEM}_{-1}$；另见??。虽然 LEM 不是第 1 章中描述的基本类型论的结果，但它可以作为公理一致地假设（与其 $\infty$ 对应物不同）。例如，我们将在?? 中假设它。

然而，令人惊讶的是我们可以在不使用 LEM 的情况下走多远。很多时候，定义或定理的简单重新表述使我们能够避免调用排中律。虽然这有时需要一点适应，但通常是值得的，会产生更优雅和更一般的证明。我们在导论中讨论了这样做的一些好处。

例如，在经典数学中，双重否定经常被不必要地使用。一个非常简单的例子是常见的假设，即集合 $A$ 是“非空的”，这字面意思是 $A$ 不包含没有元素的情况不成立。几乎总是真正想表达的是 $A$ 确实包含至少一个元素这一正面断言，通过去除双重否定，我们使声明更少依赖于 LEM 。回忆我们说类型 $A$ 是*有居留者的* 当我们断言 $A$ 本身作为命题时（即我们构造 $A$ 的一个元素，通常不命名）。因此，在将经典证明翻译成构造逻辑时，我们经常将“非空”一词替换为“有居留者”（尽管有时我们必须将其替换为“纯粹有居留者”；参见第 3.7 节）。

类似地，在经典数学中找到不必要的反证法证明并不罕见。当然，反证法的经典形式是通过双重否定律进行的：我们假设 $\neg A$ 并推导出矛盾，从而推导出 $\neg\neg A$，因此通过双重否定我们得到 $A$。然而，从 $\neg A$ 推导矛盾通常可以稍微改述，从而产生 $A$ 的直接证明，避免需要 LEM 。

还需要注意的是，如果目标是证明一个*否定*，那么“反证法”不涉及 LEM 。事实上，由于 $\neg A$ 根据定义是类型 $A \to \mathbf{0}$，根据定义证明 $\neg A$ 就是在 $A$ 的假设下证明矛盾（$\mathbf{0}$ ）。类似地，双重否定律对于否定命题确实成立：$\neg\neg\neg A \to \neg A$。通过实践，人们学会更仔细地区分否定命题和非否定命题，并注意何时使用 LEM 以及何时不使用。

因此，与表面上看起来的相反，“构造性地”做数学通常不涉及放弃重要的定理，而是找到表述定义的最佳方式，使重要定理可以构造性地证明。也就是说，在首次研究一个主题时我们可以自由使用 LEM ，但一旦该主题被更好地理解，我们可以希望改进其定义和证明以避免该公理。这种观察在同伦类型论中更加明显，其中泛等性和高阶归纳类型的强大工具使我们能够构造性地攻克许多传统上需要经典推理的问题。我们将在第2部分中看到几个这样的例子。

还值得一提的是，即使在构造数学中，排中律也可以对某些命题成立。传统上给这类命题的名称是*可判定的*。

**Definition 3.4.3.**

(i) 如果 $A + \neg A$，则类型 $A$ 称为**可判定的**。

(ii) 类似地，如果

$$\prod_{a:A} (B(a) + \neg B(a)),$$

则类型族 $B : A \to \mathcal{U}$ 是**可判定的**。

(iii) 特别地，如果

$$\prod_{a,b:A} ((a = b) + \neg(a = b)),$$

则 $A$ 有**可判定相等性**。

因此，LEM 正是所有纯粹命题都是可判定的这一声明，因此所有纯粹命题族也是如此。特别地，LEM 意味着所有集合（在第 3.1 节的意义上）都有可判定相等性。在这个意义上有可判定相等性是非常强的；参见??。

## 3.5 子集与命题调整

作为纯粹命题有用性的另一个例子，我们讨论子集（更一般地，子类型）。假设 $P : A \to \mathcal{U}$ 是一个类型族，每个类型 $P(x)$ 被视为命题。则 $P$ 本身是 $A$ 上的一个*谓词*，或 $A$ 的元素的一个*性质*。

在集合论中，只要我们有集合 $A$ 上的谓词 $P$，我们就可以形成子集 $\{\, x \in A \mid P(x) \,\}$。正如在第 1.11 节中简要提到的，类型论中明显的类比是 $\Sigma$-类型 $\sum_{(x:A)} P(x)$。$\sum_{(x:A)} P(x)$ 的居留者当然是一个对 $(x, p)$，其中 $x : A$ 且 $p$ 是 $P(x)$ 的一个证明。然而，对于一般的 $P$，如果命题 $P(a)$ 有多于一个不同的证明，元素 $a : A$ 可能产生 $\sum_{(x:A)} P(x)$ 的多于一个不同的元素。这与子集的通常直觉相悖。但如果 $P$ 是一个纯粹命题，这就不会发生。

**Lemma 3.5.1.** *假设 $P : A \to \mathcal{U}$ 是一个类型族，使得对所有 $x : A$，$P(x)$ 是一个纯粹命题。如果 $u, v : \sum_{(x:A)} P(x)$ 满足 $\mathsf{pr}_1(u) = \mathsf{pr}_1(v)$，则 $u = v$。*

*证明.* 假设 $p : \mathsf{pr}_1(u) = \mathsf{pr}_1(v)$。根据定理 2.7.2，为证明 $u = v$，只需证明 $p_*(\mathsf{pr}_2(u)) = \mathsf{pr}_2(v)$。但 $p_*(\mathsf{pr}_2(u))$ 和 $\mathsf{pr}_2(v)$ 都是 $P(\mathsf{pr}_1(v))$ 的元素，而这是一个纯粹命题；因此它们相等。 □

例如，回忆在第 2.4 节中我们定义了

$$(A \simeq B) :\equiv \sum_{f:A\to B} \mathsf{isequiv}(f),$$

其中每个类型 $\mathsf{isequiv}(f)$ 应该是一个纯粹命题。由此得出，如果两个等价有相等的底层函数，则它们作为等价相等。

此后，如果 $P : A \to \mathcal{U}$ 是纯粹命题族（即每个 $P(x)$ 是一个纯粹命题），我们可以写

$$\{\, x : A \mid P(x) \,\} \tag{3.5.2}$$

作为 $\sum_{(x:A)} P(x)$ 的替代记号。（对于任意 $P$ 也没有技术原因不能使用这种记号，但由于意想不到的含义，这种用法可能会造成混淆。）如果 $A$ 是集合，我们称 (3.5.2) 为 $A$ 的一个**子集**；对于一般的 $A$ 我们可以称之为**子类型**。我们也可以称 $P$ 本身为 $A$ 的一个子集或子类型；这实际上更正确，因为孤立的类型 (3.5.2) 不记得它与 $A$ 的关系。

给定这样的 $P$ 和 $a : A$，我们可以写 $a \in P$ 或 $a \in \{\, x : A \mid P(x) \,\}$ 来指代纯粹命题 $P(a)$。如果它成立，我们可以说 $a$ 是 $P$ 的一个**成员**。类似地，如果 $\{\, x : A \mid Q(x) \,\}$ 是 $A$ 的另一个子集，那么我们说 $P$ **包含于** $Q$ 中，并写 $P \subseteq Q$，如果我们有 $\prod_{(x:A)}(P(x) \to Q(x))$。

作为子类型的进一步例子，我们可以在宇宙 $\mathcal{U}$ 中定义集合和纯粹命题的“子宇宙”：

$$\mathsf{Set}_{\mathcal{U}} :\equiv \{\, A : \mathcal{U} \mid \mathsf{isSet}(A) \,\},$$
$$\mathsf{Prop}_{\mathcal{U}} :\equiv \{\, A : \mathcal{U} \mid \mathsf{isProp}(A) \,\}.$$

$\mathsf{Set}_{\mathcal{U}}$ 的元素是一个类型 $A : \mathcal{U}$ 连同证据 $s : \mathsf{isSet}(A)$，$\mathsf{Prop}_{\mathcal{U}}$ 也类似。定理 3.5.1意味着 $(A, s) =_{\mathsf{Set}_{\mathcal{U}}} (B, t)$ 等价于 $A =_{\mathcal{U}} B$（因此等价于 $A \simeq B$）。因此，我们经常滥用记号简单地写 $A : \mathsf{Set}_{\mathcal{U}}$ 而不是 $(A, s) : \mathsf{Set}_{\mathcal{U}}$。如果没有必要指定所讨论的宇宙，我们也可以省略下标 $\mathcal{U}$ 。

回忆对于任意两个宇宙 $\mathcal{U}_i$ 和 $\mathcal{U}_{i+1}$，如果 $A : \mathcal{U}_i$ 则也有 $A : \mathcal{U}_{i+1}$。因此，对于任何 $(A, s) : \mathsf{Set}_{\mathcal{U}_i}$，我们也有 $(A, s) : \mathsf{Set}_{\mathcal{U}_{i+1}}$，$\mathsf{Prop}_{\mathcal{U}_i}$ 也类似，给

出自然映射

$$\mathsf{Set}_{\mathcal{U}_i} \to \mathsf{Set}_{\mathcal{U}_{i+1}}, \tag{3.5.3}$$

$$\mathsf{Prop}_{\mathcal{U}_i} \to \mathsf{Prop}_{\mathcal{U}_{i+1}}. \tag{3.5.4}$$

映射 (3.5.3) 不能是等价，因为那样我们就能重现康托尔集合论中熟悉的自指悖论。然而，虽然 (3.5.4) 在我们迄今所呈现的类型论中不自动是等价，但假设它是等价是一致的。也就是说，我们可以考虑将以下内容作为额外规则添加到类型论中。

**Axiom 3.5.5** (命题调整)**.** 映射 $\mathsf{Prop}_{\mathcal{U}_i} \to \mathsf{Prop}_{\mathcal{U}_{i+1}}$ 是一个等价。

我们称这个公理为**命题调整**，因为它意味着宇宙 $\mathcal{U}_{i+1}$ 中的任何纯粹命题都可以 "调整" 为较小宇宙 $\mathcal{U}_i$ 中的等价命题。如果 $\mathcal{U}_{i+1}$ 满足 LEM ，它会自动成立（参见练习 3.10）。我们不会一般地假设这个公理，尽管在某些地方我们将其用作显式假设。它是纯粹命题的一种非直谓性形式，通过避免使用它，类型论被称为保持直谓性。

在实践中，我们最常需要的是一个略有不同的陈述：考虑中的宇宙 $\mathcal{U}$ 包含一个 "分类所有纯粹命题" 的类型。换句话说，我们想要一个类型 $\Omega : \mathcal{U}$ 连同一个 $\Omega$-索引的纯粹命题族，它包含每个纯粹命题（关于等价）。如果 $\mathcal{U}$ 不是最小宇宙 $\mathcal{U}_0$，这个陈述由上述命题调整得出，因为那时我们可以定义 $\Omega :\equiv \mathsf{Prop}_{\mathcal{U}_0}$。

非直谓性的一个用途是定义幂集。将集合 $A$ 的**幂集**定义为 $A \to \mathsf{Prop}_{\mathcal{U}}$ 是自然的；但在没有非直谓性的情况下，这个定义依赖于（甚至关于等价依赖于）宇宙 $\mathcal{U}$ 的选择。但有了命题调整，我们可以将幂集定义为

$$\mathcal{P}(A) :\equiv (A \to \Omega),$$

它就独立于 $\mathcal{U}$ 了。另见??。

## 3.6 纯粹命题的逻辑

我们在第 1.1 节中提到，与只有一个基本概念（类型）的类型论相比，集合论基础有两个基本概念：集合和命题。因此，经典数学家习惯于分别操作这两类对象。

在类型论中可以恢复类似的二分法，集合论命题的角色由纯粹命题的类型（和类型族）扮演。在许多情况下，逻辑连接词和量词可以通过简单地将相应的类型构造子限制在纯粹命题上来在这种逻辑中表示。当然，这需要知道所讨论的类型构造子保持纯粹命题。

*Example* 3.6.1. 如果 $P$ 和 $Q$ 是纯粹命题，$P \times Q$ 也是。这可以用乘积中路径的刻画来容易地证明，就像定理 3.1.5但更简单。因此，连接词 “与” 保持纯粹命题。

*Example* 3.6.2. 如果 $A$ 是任意类型且 $P : A \to \mathcal{U}$ 使得对所有 $x : A$，类型 $P(x)$ 是一个纯粹命题，则 $\prod_{(x:A)} P(x)$ 是一个纯粹命题。证明就像定理 3.1.6但更简单：给定 $f, g : \prod_{(x:A)} P(x)$，对任何 $x : A$ 我们有 $f(x) = g(x)$，因为 $P(x)$ 是一个纯粹命题。但然后根据函数外延性，我们有 $f = g$。

特别地，如果 $P$ 是一个纯粹命题，则无论 $A$ 是什么，$A \to P$ 也是。更特别地，由于 $\mathbf{0}$ 是一个纯粹命题，$\neg A \equiv (A \to \mathbf{0})$ 也是。因此，连接词 “蕴含” 和 “非” 保持纯粹命题，量词 “对于所有” 也是如此。

另一方面，某些类型构造子不保持纯粹命题。即使 $P$ 和 $Q$ 是纯粹命题，$P + Q$ 一般也不是。例如，$\mathbf{1}$ 是一个纯粹命题，但 $\mathbf{2} = \mathbf{1} + \mathbf{1}$ 不是。从逻辑上讲，$P + Q$ 是 “或” 的一种 “纯构造” 形式：它的见证包含哪个析取支为真的额外信息。有时这非常有用，但如果我们想要一种保持纯粹命题的更经典的 “或”，我们需要一种方法将这个类型 “截断” 为纯粹命题，通过遗忘这些额外信息。

$\Sigma$-类型 $\sum_{(x:A)} P(x)$（其中 $A$ 是任意类型）也有同样的问题。这是 “存在 $x : A$ 使得 $P(x)$” 的纯构造解释，它记住见证 $x$，因此即使每个类型 $P(x)$ 是纯粹命题，它一般也不是纯粹命题。（回忆我们在第 3.5 节中观察到 $\sum_{(x:A)} P(x)$ 也可以被视为 “满足 $P(x)$ 的那些 $x : A$ 的子集”。）

## 3.7 命题截断

命题截断，也称为 $(-1)$-截断、括号类型或压缩类型，是一个额外的类型构造子，它将类型 “压缩” 或 “截断” 为纯粹命题，遗忘该类型居留者中包含的除其存在性以外的所有信息。

更精确地，对于任何类型 $A$，有一个类型 $\|A\|$。它有两个构造子：

- 对于任何 $a : A$ 我们有 $|a| : \|A\|$。
- 对于任何 $x, y : \|A\|$，我们有 $x = y$。

第一个构造子意味着如果 $A$ 有居留者，$\|A\|$ 也有。第二个确保 $\|A\|$ 是一个纯粹命题；通常我们不给这个事实的见证命名。

$\|A\|$ 的递归原理说：

- 如果 $B$ 是一个纯粹命题且我们有 $f : A \to B$，则有一个诱导的 $g : \|A\| \to B$ 使得对所有 $a : A$ 有 $g(|a|) \equiv f(a)$。

换句话说，任何从（$A$ 的居留性）推出的纯粹命题已经从 $\|A\|$ 推出。因此，$\|A\|$ 作为一个纯粹命题，不包含比 $A$ 的居留性更多的信息。（$\|A\|$ 还有一个归纳原理，但它不是特别有用；参见练习 3.17。）

在练习 3.14 和练习 3.15 和?? 中，我们将描述一些用更一般的东西构造 $\|A\|$ 的方法。现在，我们简单地将其作为第 1 章规则之外的额外规则假设。

有了命题截断，我们可以扩展“纯粹命题的逻辑”以涵盖析取和存在量词。具体地，$\|A + B\|$ 是“$A$ 或 $B$”的纯粹命题版本，它不“记得”哪个析取支为真的信息。

截断的递归原理意味着我们仍然可以在试图证明纯粹命题时对 $\|A + B\|$ 进行情形分析。也就是说，假设我们有假设 $u : \|A + B\|$ 且我们试图证明纯粹命题 $Q$。换句话说，我们试图定义 $\|A + B\| \to Q$ 的一个元素。由于 $Q$ 是一个纯粹命题，根据命题截断的递归原理，只需构造函数 $A + B \to Q$。但现在我们可以对 $A + B$ 进行情形分析。

类似地，对于类型族 $P : A \to \mathcal{U}$，我们可以考虑 $\left\|\sum_{(x:A)} P(x)\right\|$，它是“存在 $x : A$ 使得 $P(x)$”的纯粹命题版本。对于析取，通过结合截断和 $\Sigma$-类型的归纳原理，如果我们有类型 $\left\|\sum_{(x:A)} P(x)\right\|$ 的假设，我们可以在试图证明纯粹命题时引入新假设 $x : A$ 和 $y : P(x)$。换句话说，如果我们知道存在某个满足 $P(x)$ 的 $x : A$，但我们手头没有特定的这样的 $x$，那么只要我们不试图构造可能依赖于 $x$ 的特定值的东西，我们就可以自由使用这样的 $x$。要求余域是纯粹命题表达了结果不依赖于见证，因为这样类型的所有可能居留者必须相等。

为了第 11 章 和第 10 章中的集合层面数学的目的，我们主要处理集合和纯粹命题，使用传统逻辑记号仅指代 “命题截断逻辑” 是方便的。

**Definition 3.7.1.** 我们使用截断如下定义**传统逻辑记号**，其中 $P$ 和 $Q$ 表示纯粹命题（或其族）：

$$\begin{aligned}
\top &:\equiv \mathbf{1} \\
\bot &:\equiv \mathbf{0} \\
P \wedge Q &:\equiv P \times Q \\
P \Rightarrow Q &:\equiv P \to Q \\
P \Leftrightarrow Q &:\equiv P = Q \\
\neg P &:\equiv P \to \mathbf{0} \\
P \vee Q &:\equiv \|P + Q\| \\
\forall (x : A).\, P(x) &:\equiv \prod_{x:A} P(x) \\
\exists (x : A).\, P(x) &:\equiv \left\| \sum_{x:A} P(x) \right\|
\end{aligned}$$

记号 $\wedge$ 和 $\vee$ 在同伦论中也用于指向空间的粉碎积和楔积，我们将在第 6 章中介绍。这在技术上造成了潜在的冲突，但一般不会造成混淆。

类似地，在讨论如第 3.5 节中的子集时，我们可以使用传统的交集、并集和补集记号：

$$\begin{aligned}
\{\, x : A \mid P(x) \,\} \cap \{\, x : A \mid Q(x) \,\} &:\equiv \{\, x : A \mid P(x) \wedge Q(x) \,\}, \\
\{\, x : A \mid P(x) \,\} \cup \{\, x : A \mid Q(x) \,\} &:\equiv \{\, x : A \mid P(x) \vee Q(x) \,\}, \\
A \setminus \{\, x : A \mid P(x) \,\} &:\equiv \{\, x : A \mid \neg P(x) \,\}.
\end{aligned}$$

当然，在没有 LEM 的情况下，后者不是通常意义上的 “补集”：对于 $A$ 的每个子集 $B$，我们可能没有 $B \cup (A \setminus B) = A$。

## 3.8 选择公理

我们现在可以在同伦类型论中适当地表述选择公理。假设一个类型 $X$ 和类型族

$$A : X \to \mathcal{U} \qquad \text{和} \qquad P : \prod_{x:X} A(x) \to \mathcal{U},$$

并且

- $X$ 是一个集合，
- 对所有 $x : X$，$A(x)$ 是一个集合，且
- 对所有 $x : X$ 和 $a : A(x)$，$P(x, a)$ 是一个纯粹命题。

**选择公理** AC 断言在这些假设下，

$$\Big(\prod_{x:X} \Big\| \sum_{a:A(x)} P(x,a) \Big\|\Big) \to \Big\| \sum_{(g:\prod_{(x:X)} A(x))} \prod_{(x:X)} P(x, g(x)) \Big\|. \tag{3.8.1}$$

当然，这是 (3.2.1) 的直接翻译，其中我们将 “存在 $x : A$ 使得 $B(x)$” 读作 $\Big\|\sum_{(x:A)} B(x)\Big\|$，所以我们可以用熟悉的逻辑记号将声明写作

$$\Big(\forall(x : X).\, \exists(a : A(x)).\, P(x,a)\Big) \Rightarrow \Big(\exists(g : \prod_{x:X} A(x)).\, \forall(x : X).\, P(x, g(x))\Big).$$

特别地，注意命题截断出现了两次。定义域中的截断意味着我们假设对每个 $x$ 存在某个 $a : A(x)$ 使得 $P(x, a)$，但这些值没有以任何已知方式被选择或指定。陪域中的截断意味着我们得出存在某个函数 $g$，但这个函数没有以任何已知方式被确定或指定。

事实上，由于定理 2.15.7，这个公理也可以用更简单的形式表达。

**Lemma 3.8.2.** *选择公理 (3.8.1) 等价于以下声明：对于任何集合 $X$ 和任何 $Y : X \to \mathcal{U}$ 使得每个 $Y(x)$ 是集合，我们有*

$$\Big(\prod_{x:X} \Big\| Y(x) \Big\|\Big) \to \Big\| \prod_{x:X} Y(x) \Big\|. \tag{3.8.3}$$

这对应于经典选择公理的一个众所周知的等价形式，即“非空集族的笛卡尔积是非空的”。

证明. 根据定理 2.15.7， (3.8.1) 的陪域等价于

$$\Big\| \prod_{(x:X)} \sum_{(a:A(x))} P(x,a) \Big\|.$$

因此， (3.8.1) 等价于 (3.8.3) 的实例，其中

$$Y(x) :\equiv \sum_{a:A(x)} P(x,a).$$

（这根据定理 3.1.5 和定理 3.3.4是一个集合。）反过来，(3.8.3) 等价于 (3.8.1) 的实例，其中 $A(x) :\equiv Y(x)$ 且 $P(x,a) :\equiv \mathbf{1}$。因此，两者逻辑等价。由于两者都是纯粹命题，根据引理 3.3.3它们是等价类型。 □

与 LEM 一样，等价形式 (3.8.1) 和 (3.8.3) 不是我们基本类型论的结果，但它们可以一致地作为公理假设。

*Remark* 3.8.4. 容易证明 (3.8.3) 的右边总是蕴含左边。由于两者都是纯粹命题，根据引理 3.3.3，选择公理也等价于要求等价

$$\Big(\prod_{x:X} \big\|Y(x)\big\|\Big) \simeq \Big\|\prod_{x:X} Y(x)\Big\|$$

这说明了一个常见的陷阱：虽然依赖函数类型保持纯粹命题（定理 3.6.2），它们不与截断交换：$\left\|\prod_{(x:A)} P(x)\right\|$ 一般不等价于 $\prod_{(x:A)}\|P(x)\|$。选择公理，如果我们假设它，说这对集合是真的；正如我们将在下面看到的，它一般失效。

选择公理中类型是集合的限制可以在一定程度上放松。例如，我们可以允许 (3.8.1) 中的 $A$ 和 $P$，或 (3.8.3) 中的 $Y$，是任意类型族；这产生一个看起来更强但同样一致的声明。我们也可以用第 7 章中将考虑的更一般的 $n$-截断替代命题截断，得到一系列公理 $\mathsf{AC}_n$，插值于 (3.8.1)（我们简单称之为 AC ，或为强调称为 $\mathsf{AC}_{-1}$）和定理 2.15.7（我们称之为 $\mathsf{AC}_\infty$）之间。另见?? 和??。然而，观察到我们不能放松 $X$ 是集合的要求。

**Lemma 3.8.5.** 存在类型 $X$ 和族 $Y : X \to \mathcal{U}$ 使得每个 $Y(x)$ 是集合，但 (3.8.3) 为假。

证明. 定义 $X :\equiv \sum_{(A:\mathcal{U})} \|\mathbf{2} = A\|$，令 $x_0 :\equiv (\mathbf{2}, |\mathsf{refl}_{\mathbf{2}}|) : X$。则根据 $\Sigma$-类型中路径的刻画、$\|A = \mathbf{2}\|$ 是纯粹命题的事实和泛等性，对于任何 $(A, p), (B, q) : X$ 我们有 $((A, p) =_X (B, q)) \simeq (A \simeq B)$。特别地，$(x_0 =_X x_0) \simeq (\mathbf{2} \simeq \mathbf{2})$，所以如定理 3.1.9中那样，$X$ 不是集合。

另一方面，如果 $(A, p) : X$，则 $A$ 是集合；这由对 $p : \|\mathbf{2} = A\|$ 的截断归纳和 $\mathbf{2}$ 是集合的事实得出。由于当 $A$ 和 $B$ 是集合时 $A \simeq B$ 是集合，$x_1 =_X x_2$ 对任何 $x_1, x_2 : X$ 是集合，即 $X$ 是 1-类型。特别地，如果我们定义 $Y : X \to \mathcal{U}$ 为 $Y(x) :\equiv (x_0 = x)$，则每个 $Y(x)$ 是集合。

现在根据定义，对于任何 $(A, p) : X$ 我们有 $\|\mathbf{2} = A\|$，因此 $\|x_0 = (A, p)\|$。因此，我们有 $\prod_{(x:X)} \|Y(x)\|$。如果 (3.8.3) 对这个 $X$ 和 $Y$ 成立，那么我们也会有 $\left\|\prod_{(x:X)} Y(x)\right\|$。由于我们试图推导矛盾（$\mathbf{0}$），这是一个纯粹命题，我们可以假设 $\prod_{(x:X)} Y(x)$，即 $\prod_{(x:X)} (x_0 = x)$。但这意味着 $X$ 是一个纯粹命题，因此是一个集合，这是一个矛盾。 □

## 3.9 唯一选择原理

以下观察是平凡的，但非常有用。

**Lemma 3.9.1.** 如果 $P$ 是一个纯粹命题，则 $P \simeq \|P\|$。

证明. 当然，根据定义我们有 $P \to \|P\|$。由于 $P$ 是一个纯粹命题，将 $\|P\|$ 的泛性质应用于 $\mathsf{id}_P : P \to P$ 产生 $\|P\| \to P$。根据引理 3.3.3，这些函数是拟逆。 □

它的重要推论之一如下。

**Corollary 3.9.2** (唯一选择原理). 假设类型族 $P : A \to \mathcal{U}$ 满足

*(i)* 对每个 $x$，类型 $P(x)$ 是一个纯粹命题，且

*(ii)* 对每个 $x$ 我们有 $\|P(x)\|$。

则我们有 $\prod_{(x:A)} P(x)$。

证明. 直接由两个假设和前一个引理得出。 □

这个推论还封装了一种非常有用的推理技巧。即，假设我们知道 $\|A\|$，我们想用它来构造某个其他类型 $B$ 的元素。我们想在构造 $B$ 的元素时使用 $A$ 的元素，但这只有当 $B$ 是一个纯粹命题时才被允许，这样我们才能应用命题截断 $\|A\|$ 的归纳原理；我们一般能期望的最多是证明 $\|B\|$。相反，我们可以用额外的数据扩展 $B$，这些数据唯一地刻画我们想要构造的对象。具体地，我们定义谓词 $Q : B \to \mathcal{U}$ 使得 $\sum_{(x:B)} Q(x)$ 是一个纯粹命题。然后从 $A$ 的元素我们构造元素 $b : B$ 使得 $Q(b)$，因此从 $\|A\|$ 我们可以构造 $\left\|\sum_{(x:B)} Q(x)\right\|$，由于 $\left\|\sum_{(x:B)} Q(x)\right\|$ 等价于 $\sum_{(x:B)} Q(x)$，可以从中投影出 $B$ 的元素。练习 3.19中可以找到一个例子。

集合论数学中也出现类似的问题，虽然表现略有不同。如果我们试图定义函数 $f : A \to B$，并且依赖于元素 $a : A$ 我们能够证明某个 $b : B$ 的纯粹存在性，我们还没有完成，因为我们需要实际确定 $B$ 的一个元素，而不仅仅是证明其存在。一个选择当然是将论证精炼为 $b : B$ 的唯一存在性，就像我们在类型论中所做的那样。但在集合论中，这个问题通常可以通过应用选择公理更简单地避免，它为我们选择所需的元素。然而，在同伦类型论中，撇开任何避免选择的愿望不谈，可用的选择形式的适用性更低，因为它们要求选择的定义域是一个集合。因此，如果 $A$ 不是集合（例如可能是宇宙 $\mathcal{U}$），没有一致的选择形式允许我们简单地为每个 $a : A$ 选择一个 $B$ 的元素来定义 $f(a)$。

## 3.10 命题何时被截断？

乍看之下，$+$ 和 $\Sigma$ 的截断版本似乎实际上比非截断版本更接近 “或” 和 “存在” 的非形式数学意义。当然，它们更接近一阶逻辑中 “或” 和 “存在” 的精确意义，一阶逻辑是形式集合论的基础，因为后者不试图记住命题真值的任何见证。然而，认识到非形式数学的实践通常由非截断形式更准确地描述，这可能令人惊讶。

例如，考虑像 “每个素数要么是 2 要么是奇数” 这样的声明。工作中的

数学家会毫无顾虑地不仅使用这个事实来证明关于素数的定理，还会对素数进行构造，也许在 2 的情况下做一件事，在奇素数的情况下做另一件事。构造的最终结果不仅仅是某个声明的真值，而是一段可能依赖于素数奇偶性的数据。因此，从类型论的角度来看，这样的构造自然地使用余积类型 “$(p = 2) + (p$ 是奇数$)$” 的归纳原理来表述，而不是其命题截断。

诚然，这不是一个理想的例子，因为 “$p = 2$” 和 “$p$ 是奇数” 是互斥的，所以 $(p = 2) + (p$ 是奇数$)$ 实际上已经是一个纯粹命题，因此等价于其截断（参见练习 3.7）。更有说服力的例子来自存在量词。证明形如 “存在 $x$ 使得 ……” 的定理然后后来引用 “定理 Y 中构造的 $x$”（注意定冠词）并不罕见。而且，在推导这个 $x$ 的进一步性质时，人们可能使用诸如 “根据定理 Y 证明中 $x$ 的构造” 之类的短语。

一个非常常见的例子是 “$A$ 同构于 $B$”，严格来说这只意味着 $A$ 和 $B$ 之间存在某个同构。但几乎总是，在证明这样的声明时，人们展示一个特定的同构或证明某个先前已知的映射是同构，后来具体给出的是哪个同构往往很重要。

受集合论训练的数学家经常对这种 “语言滥用” 感到一丝愧疚。我们可能试图为它们道歉，在最终稿中删除它们，或用模糊的词如 “典范” 来搪塞。问题因以下事实而加剧：在形式化集合论中，技术上根本没有办法 “构造” 对象——我们只能证明具有某些性质的对象存在。因此，类型论中的非截断逻辑捕捉了非形式数学的一些常见实践，而集合论重构掩盖了这些实践。（这类似于泛等公理如何验证常见但在形式上不合理的将同构对象等同的实践。）

另一方面，有时截断逻辑是必不可少的。我们在 LEM 和 AC 的声明中看到了这一点；其他一些例子将在本书后面出现。因此，我们面临这样的问题：在写非形式类型论时，“或” 和 “存在”（以及常见的同义词如 “有” 和 “我们有”）应该意味着什么？

普遍共识可能是不可能的。也许取决于所做的数学类型，一种约定或另一种可能更有用——或者，也许，约定的选择可能无关紧要。在这种情况下，在数学论文开头的备注可能足以告知读者其中使用的语言约定。然而，即使选择了一个整体约定，另一种逻辑通常至少偶尔会出现，所以我们需要一种方式来引用它。更一般地，人们可以考虑用另一种在类型上表现类

似的操作替代命题截断，如双重否定操作 $A \mapsto \neg\neg A$，或第 7 章中将考虑的 $n$-截断。作为阐述的实验，在接下来的内容中我们将偶尔使用副词来表示命题截断等 “模态” 的应用。

例如，如果非截断逻辑是默认约定，我们可以使用副词**纯粹** 来表示命题截断。因此短语

“纯粹存在 $x : A$ 使得 $P(x)$”

表示类型 $\left\|\sum_{(x:A)} P(x)\right\|$。类似地，我们会说类型 $A$ 是**纯粹有居留者的** 来表示其命题截断 $\|A\|$ 有居留者（即我们有它的一个未命名元素）。注意这是副词 “纯粹” 在我们非形式数学英语中使用方式的定义，就像我们定义名词如 “群” 和 “环”，以及形容词如 “正则” 和 “正规”，具有精确的数学意义一样。我们不是声称 “merely” 的字典定义指的是命题截断；这个词的选择只是为了提醒数学读者，纯粹命题 “仅仅” 包含真值的信息，仅此而已。

另一方面，如果截断逻辑是当前的默认约定，我们可以使用副词如**纯然** 或**构造性地**来表示其缺失，所以

“纯然存在 $x : A$ 使得 $P(x)$”

将表示类型 $\sum_{(x:A)} P(x)$。我们也可以使用 “纯然” 或 “实际上” 只是为了强调没有截断，即使那是默认约定。

在本书中，我们将继续使用非截断逻辑作为默认约定，原因如下。

(1) 我们想鼓励新手尝试它，而不是仅仅因为更熟悉而坚持使用截断逻辑。

(2) 在类型论中使用截断逻辑作为默认会遭受与集合论基础相同的 “语言滥用”问题，非截断逻辑避免了这些问题。例如，我们将 “$A \simeq B$” 定义为 $A$ 和 $B$ 之间等价的类型，而不是其命题截断，意味着证明形如 “$A \simeq B$” 的定理就是字面上构造一个特定的这样的等价。这个特定的等价随后可以被引用。

(3) 我们想强调 “纯粹命题” 的概念不是类型论的基本部分。正如我们将在第 7 章中看到的，纯粹命题只是无限阶梯上的第二级，还有许多其他模态根本不在这个阶梯上。

(4) 许多在经典上是纯粹命题的声明在同伦类型论中不再是。当然，其中首要的是相等性。

(5) 另一方面，同伦类型论最有趣的观察之一是惊人数量的类型自动是纯粹命题，或可以略微修改使之成为纯粹命题，而不需要任何截断。（参见定理 3.3.5 和第 4 章 和第 7 章 和第 9 章 和第 10 章。）因此，虽然这些类型除了真值外不包含任何数据，我们仍然可以使用它们来构造非截断对象，因为不需要使用命题截断的归纳原理。如果命题截断默认应用于所有声明，这个有用的事实就更难表达了。

(6) 最后，截断对于本书中我们将做的大多数数学并不是非常有用，所以在它们出现时显式地记号它们更简单。

## 3.11 可缩性

在定理 3.3.2中我们观察到一个有居留者的纯粹命题必须等价于 $\mathbf{1}$，不难看出反过来也成立。具有这个性质的类型称为可缩的。可缩性的另一个等价定义，有时也很方便，如下。

**Definition 3.11.1.** 类型 $A$ 是**可缩的**，或称为**单元素集**，如果存在 $a : A$，称为**收缩中心**，使得对所有 $x : A$ 有 $a = x$。我们将指定的路径 $a = x$ 记为 $\mathsf{contr}_x$。

换句话说，类型 $\mathsf{isContr}(A)$ 定义为

$$\mathsf{isContr}(A) :\equiv \sum_{(a:A)} \prod_{(x:A)} (a = x).$$

注意在通常的命题即类型解读下，我们可以将 $\mathsf{isContr}(A)$ 读作 “$A$ 恰好包含一个元素”，或更精确地 “$A$ 包含一个元素，且 $A$ 的每个元素都等于那个元素”。

*Remark* 3.11.2. 我们也可以更拓扑地将 $\mathsf{isContr}(A)$ 读作 “存在点 $a : A$ 使得对所有 $x : A$ 存在从 $a$ 到 $x$ 的路径”。注意对经典的耳朵来说，这听起来像是连通性而不是可缩性的定义。关键点是这句话中 “存在” 的意义是连续/自然的。

表达连通性的更好方式是 $\sum_{(a:A)} \prod_{(x:A)} \|a = x\|$。如果假设 $A$ 是点化的，这确实是正确的——参见?? 后的注记——但一般来说一个类型可以是连通的而不是点化的。在?? 中我们将连通性定义为 $n$-连通性一般概念的 $n = 0$ 情形，在?? 中读者被要求证明这个定义等价于同时有 $\|A\|$ 和 $\prod_{(x,y:A)} \|x = y\|$。

**Lemma 3.11.3.** 对于类型 $A$，以下是逻辑等价的。

*(i)* $A$ 在定义 *3.11.1*的意义上是可缩的。

*(ii)* $A$ 是一个纯粹命题，且存在点 $a : A$。

*(iii)* $A$ 等价于 $\mathbf{1}$ 。

证明. 如果 $A$ 是可缩的，那么它当然有一个点 $a : A$（收缩中心），而对于任何 $x, y : A$ 我们有 $x = a = y$；因此 $A$ 是一个纯粹命题。反过来，如果我们有 $a : A$ 且 $A$ 是一个纯粹命题，则对于任何 $x : A$ 我们有 $x = a$；因此 $A$ 是可缩的。我们在定理 3.3.2中证明了 (ii)⇒(iii)，而反过来成立是因为 $\mathbf{1}$ 容易具有性质 (ii)。 □

**Lemma 3.11.4.** 对于任何类型 $A$，类型 $\mathsf{isContr}(A)$ 是一个纯粹命题。

证明. 假设给定 $c, c' : \mathsf{isContr}(A)$。我们可以假设 $c \equiv (a, p)$ 和 $c' \equiv (a', p')$，其中 $a, a' : A$ 且 $p : \prod_{(x:A)}(a = x)$ 和 $p' : \prod_{(x:A)}(a' = x)$。根据 Σ-类型中路径的刻画，要证明 $c = c'$，只需展示 $q : a = a'$ 使得 $q_*(p) = p'$。我们选择 $q :\equiv p(a')$。现在由于 $A$ 是可缩的（根据 $c$ 或 $c'$），根据定理 3.11.3它是一个纯粹命题。因此，根据定理 3.3.4 和定理 3.6.2，$\prod_{(x:A)}(a' = x)$ 也是；因此 $q_*(p) = p'$ 是自动的。 □

**Corollary 3.11.5.** 如果 $A$ 是可缩的，则 $\mathsf{isContr}(A)$ 也是。

证明. 根据定理 3.11.4和定理 3.11.3(ii)。 □

像纯粹命题一样，可缩类型被许多类型构造子保持。例如，我们有：

**Lemma 3.11.6.** 如果 $P : A \to \mathcal{U}$ 是一个类型族使得每个 $P(a)$ 是可缩的，则 $\prod_{(x:A)} P(x)$ 是可缩的。

证明. 根据定理 3.6.2，$\prod_{(x:A)} P(x)$ 是一个纯粹命题，因为每个 $P(x)$ 是。但它也有一个元素，即将每个 $x : A$ 送到 $P(x)$ 的收缩中心的函数。因此根据定理 3.11.3(ii)，$\prod_{(x:A)} P(x)$ 是可缩的。 □

（事实上，定理 3.11.6的声明等价于函数外延性公理。参见 第 4.9 节。）

当然，如果 $A$ 等价于 $B$ 且 $A$ 是可缩的，则 $B$ 也是。更一般地，$B$ 是 $A$ 的收缩核就足够了。根据定义，**收缩** 是函数 $r : A \to B$ 使得存在函数 $s : B \to A$，称为它的**截面**，和同伦 $\epsilon : \prod_{(y:B)}(r(s(y)) = y)$；则我们说 $B$ 是 $A$ 的**收缩核** 。

**Lemma 3.11.7.** 如果 $B$ 是 $A$ 的收缩核，且 $A$ 是可缩的，则 $B$ 也是。

证明. 令 $a_0 : A$ 为收缩中心。我们声称 $b_0 :\equiv r(a_0) : B$ 是 $B$ 的收缩中心。令 $b : B$；我们需要路径 $b = b_0$。但我们有 $\epsilon_b : r(s(b)) = b$ 和 $\mathsf{contr}_{s(b)} : s(b) = a_0$，所以通过复合

$$\epsilon_b{}^{-1} \cdot r\big(\mathsf{contr}_{s(b)}\big) : b = r(a_0) \equiv b_0.$$ □

可缩类型可能看起来不是非常有趣，因为它们都等价于 $\mathbf{1}$ 。这个概念有用的一个原因是有时一组单独非平凡的数据会共同形成一个可缩类型。一个重要的例子是有一个自由端点的路径空间。正如我们将在?? 中看到的，这个事实本质上封装了恒等类型的基点路径归纳原理。

**Lemma 3.11.8.** 对于任何 $A$ 和任何 $a : A$，类型 $\sum_{(x:A)}(a = x)$ 是可缩的。

证明. 我们选择点 $(a, \mathsf{refl}_a)$ 作为中心。现在假设 $(x, p) : \sum_{(x:A)}(a = x)$；我们必须证明 $(a, \mathsf{refl}_a) = (x, p)$。根据 $\Sigma$-类型中路径的刻画，只需展示 $q : a = x$ 使得 $q_*(\mathsf{refl}_a) = p$。但我们可以取 $q :\equiv p$，在这种情况下 $q_*(\mathsf{refl}_a) = p$ 由路径类型中传输的刻画得出。 □

当这种情况发生时，它可以允许我们使用非形式原理 “可缩数据可以自由忽略” 将复杂构造简化到等价。这个原理由许多引理组成，大部分我们留给读者；以下是一个例子。

**Lemma 3.11.9.** 令 $P : A \to \mathcal{U}$ 是一个类型族。

*(i)* 如果每个 $P(x)$ 是可缩的，则 $\sum_{(x:A)} P(x)$ 等价于 $A$。

*(ii)* 如果 $A$ 是可缩的且中心为 $a$，则 $\sum_{(x:A)} P(x)$ 等价于 $P(a)$。

证明. 在 (i)的情况下，我们证明 $\mathsf{pr}_1 : \sum_{(x:A)} P(x) \to A$ 是一个等价。对于拟逆我们定义 $g(x) :\equiv (x, c_x)$，其中 $c_x$ 是 $P(x)$ 的中心。复合 $\mathsf{pr}_1 \circ g$ 显然是 $\mathsf{id}_A$，而相反的复合通过使用每个 $P(x)$ 的收缩同伦于恒等。

我们将 (ii)的证明留给读者（参见练习 3.20）。 □

可缩类型有趣的另一个原因是它们将第 3.1 节中提到的 $n$-类型阶梯向下扩展了一级。

**Lemma 3.11.10.** 类型 $A$ 是一个纯粹命题当且仅当对所有 $x, y : A$，类型 $x =_A y$ 是可缩的。

证明. 对于“如果”，我们简单地观察到任何可缩类型都有居留者。对于“仅当”，我们在第 3.3 节中观察到每个纯粹命题都是集合，所以每个类型 $x =_A y$ 是一个纯粹命题。但它也有居留者（因为 $A$ 是一个纯粹命题），因此根据定理 3.11.3(ii)它是可缩的。 □

因此，可缩类型也可以称为 $(-2)$-**类型**。它们是 $n$-类型阶梯的最底层，将是我们在第 7 章中给出的 $n$-类型递归定义的基例。

## Notes

在类型论中可以定义集合、纯粹命题和可缩类型，所有高阶同伦都自动处理好，如第 3.1 节 和第 3.3 节 和第 3.11 节中那样，这一事实首先由 Voevodsky 观察到。事实上，他通过归纳定义了整个 $n$-类型层次结构，正如我们将在第 7 章中所做的那样。

定理 3.2.2 和定理 3.2.7本质上依赖于 Hedberg 的一个经典定理，我们将在?? 中证明它。命题即类型形式的 LEM 与泛等性矛盾这一蕴含由 Martín Escardó 在 Agda 邮件列表上观察到。我们给出的定理 3.2.2的证明归功于 Thierry Coquand。

命题截断是在 1983 年由 Constable [**?**] 作为“子集”和“商”类型的应用引入 NuPRL 的外延类型论中的。这里所称的“命题截断”在 NuPRL

类型论 [?]中被称为 “压缩”。直接刻画命题截断的规则，仍然在外延类型论中，在 [?]中给出。同伦类型论中的内涵版本由 Voevodsky 使用非直谓量化构造，后来由 Lumsdaine 使用高阶归纳类型构造（参见??）。

Voevodsky [Voe12] 提出了第 3.5 节中考虑的那种调整规则。这些显然与 Russell 在他和 Whitehead 的《数学原理》 [?]中提出的臭名昭著的可还原性公理相关。

副词 “纯然” 用于指非截断逻辑，是对编程语言中使用单子模态来模拟效果的引用；参见?? 和第 7 章的注记。

有许多不同的方式可以相对于类型论处理逻辑。例如，除了第 1.11 节中描述的普通命题即类型逻辑，以及第 3.6 节中描述的只使用纯粹命题的替代方案，人们还可以引入一个单独的命题 “种类”，它的行为有点像类型但不与它们等同。这是逻辑丰富类型论 [?]和一些拓扑斯及相关范畴的内部语言表述（例如 [?, ?]）以及证明助手 Coq 所采用的方法。这种方法更一般，但不那么强大。例如，唯一选择原理（第 3.9 节）在 Coq 中所谓的 setoid 范畴 [?]、逻辑丰富类型论 [?]和最小类型论 [?]中失效。因此，泛等公理使我们的类型论的行为更像拓扑斯的内部逻辑；另见第 10 章。

Martin-Löf [?] 提供了关于选择公理历史的讨论。当然，构造数学和直觉主义数学有着漫长而复杂的历史，我们在这里不会深入探讨；参见例如 [TvD88a, TvD88b]。

## Exercises

*Exercise* 3.1. 证明如果 $A \simeq B$ 且 $A$ 是集合，则 $B$ 也是。

*Exercise* 3.2. 证明如果 $A$ 和 $B$ 是集合，则 $A + B$ 也是。

*Exercise* 3.3. 证明如果 $A$ 是集合且 $B : A \to \mathcal{U}$ 是一个类型族使得对所有 $x : A$，$B(x)$ 是集合，则 $\sum_{(x:A)} B(x)$ 是集合。

*Exercise* 3.4. 证明 $A$ 是一个纯粹命题当且仅当 $A \to A$ 是可缩的。

*Exercise* 3.5. 证明 $\mathsf{isProp}(A) \simeq (A \to \mathsf{isContr}(A))$。

*Exercise* 3.6. 证明如果 $A$ 是一个纯粹命题，则 $A + (\neg A)$ 也是。因此，在 (3.4.1) 中不需要插入命题截断。

*Exercise* 3.7. 更一般地，证明如果 $A$ 和 $B$ 是纯粹命题且 $\neg(A \times B)$，则 $A + B$ 也是一个纯粹命题。

*Exercise* 3.8. 假设某个类型 $\mathsf{isequiv}(f)$ 满足第 2.4 节的条件 (i)–(iii)，证明类型 $\|\mathsf{qinv}(f)\|$ 满足相同的条件并等价于 $\mathsf{isequiv}(f)$。

*Exercise* 3.9. 证明如果 LEM 成立，则类型 $\mathsf{Prop} :\equiv \sum_{(A:\mathcal{U})} \mathsf{isProp}(A)$ 等价于 $\mathbf{2}$ 。

*Exercise* 3.10. 证明如果 $\mathcal{U}_{i+1}$ 满足 LEM ，则典范包含 $\mathsf{Prop}_{\mathcal{U}_i} \to \mathsf{Prop}_{\mathcal{U}_{i+1}}$ 是一个等价。

*Exercise* 3.11. 证明并非对所有 $A : \mathcal{U}$ 我们有 $\|A\| \to A$。（然而，可以有特定的类型使得 $\|A\| \to A$。练习 3.8意味着 $\mathsf{qinv}(f)$ 就是这样的。）

*Exercise* 3.12. 证明如果 LEM 成立，则对所有 $A : \mathcal{U}$ 我们有 $\big\|(\|A\| \to A)\big\|$。（这个性质是选择公理的一个非常简单的形式，在没有 LEM 的情况下可能失效；参见 [**?**]。）

*Exercise* 3.13. 我们在定理 3.2.7中证明了以下朴素形式的 LEM 与泛等性不一致：

$$\prod_{A:\mathcal{U}} (A + (\neg A))$$

在没有泛等性的情况下，这个公理是一致的。然而，证明它蕴含选择公理 (3.8.1)。

*Exercise* 3.14. 证明假设 LEM ，双重否定 $\neg\neg A$ 与命题截断 $\|A\|$ 有相同的递归原理，但计算规则是命题性的而不是判断性的。换句话说，证明假设 LEM ，如果 $B$ 是一个纯粹命题且我们有 $f : A \to B$，则有一个诱导的 $g : \neg\neg A \to B$ 使得对所有 $a : A$ 有 $g(|a|) = f(a)$。推导出（假设 LEM ）我们有 $\neg\neg A \simeq \|A\|$。因此，在 LEM 下，命题截断可以被定义而不是作为单独的类型构造子。

*Exercise* 3.15.

(i) 证明对于任何 $A : \mathcal{U}$，类型

$$\prod_{P:\mathsf{Prop}_{\mathcal{U}}} \Big((A \to P) \to P\Big)$$

与 $\|A\|$ 有相同的递归原理，至少相对于 $\mathcal{U}$ 中的命题，带有相同的判断性计算规则。

(ii) 前一部分考虑的类型不在相同的宇宙 $\mathcal{U}$ 中，它的递归原理只适用于 $\mathcal{U}$ 中的命题。证明如果我们假设命题调整，我们可以定义一个确实在相同宇宙 $\mathcal{U}$ 中的类型，并满足与 $\|A\|$ 相同的递归原理，虽然只有命题性计算规则。因此，在这种情况下我们也可以定义命题截断。

*Exercise* 3.16. 假设 LEM ，证明双重否定与纯粹命题对集合的全称量化交换。也就是说，证明如果 $X$ 是集合且每个 $Y(x)$ 是纯粹命题，则 LEM 蕴含

$$\Big(\prod_{x:X} \neg\neg Y(x)\Big) \simeq \Big(\neg\neg \prod_{x:X} Y(x)\Big). \tag{3.11.11}$$

观察到如果我们假设每个 $Y(x)$ 是集合，则 (3.11.11) 变得等价于选择公理 (3.8.3)。

*Exercise* 3.17. 证明第 3.7 节中给出的命题截断规则足以蕴含以下归纳原理：对于任何类型族 $B : \|A\| \to \mathcal{U}$ 使得每个 $B(x)$ 是一个纯粹命题，如果对每个 $a : A$ 我们有 $B(|a|)$，则对每个 $x : \|A\|$ 我们有 $B(x)$。

*Exercise* 3.18. 证明排中律 (3.4.1) 和双重否定律 (3.4.2) 逻辑等价。

*Exercise* 3.19. 假设 $P : \mathbb{N} \to \mathcal{U}$ 是一个可判定族（参见定义 3.4.3(ii)）的纯粹命题。证明

$$\Big\|\sum_{n:\mathbb{N}} P(n)\Big\| \to \sum_{n:\mathbb{N}} P(n).$$

*Exercise* 3.20. 证明定理 3.11.9(ii)：如果 $A$ 是可缩的且中心为 $a$，则 $\sum_{(x:A)} P(x)$ 等价于 $P(a)$。

*Exercise* 3.21. 证明 $\mathsf{isProp}(P) \simeq (P \simeq \|P\|)$。

*Exercise* 3.22. 与经典集合论一样，选择公理的有限版本是一个定理。证明当 $X$ 是有限类型 $\mathsf{Fin}(n)$（如?? 中定义的）时，选择公理 (3.8.1) 成立。

*Exercise* 3.23. 证明如果 $P : \mathbb{N} \to \mathcal{U}$ 是任何可判定族，则练习 3.19的结论为真。

*Exercise* 3.24. 通过首先证明 $\mathsf{code}(m, n)$ 对所有 $m, n$ 是一个纯粹命题来简化定理 2.13.1的证明。

# 第 4 章　等价

我们现在更详细地研究在第 2.4 节中简要介绍的类型等价的概念。具体地，我们将给出几种不同的方式来定义具有那里提到的性质的类型 $\mathsf{isequiv}(f)$。回忆我们希望 $\mathsf{isequiv}(f)$ 具有以下性质，这里重新陈述：

(i) $\mathsf{qinv}(f) \to \mathsf{isequiv}(f)$。

(ii) $\mathsf{isequiv}(f) \to \mathsf{qinv}(f)$。

(iii) $\mathsf{isequiv}(f)$ 是一个纯粹命题。

这里 $\mathsf{qinv}(f)$ 表示 $f$ 的拟逆类型：

$$\sum_{g:B\to A} \big((f \circ g \sim \mathsf{id}_B) \times (g \circ f \sim \mathsf{id}_A)\big).$$

根据函数外延性，$\mathsf{qinv}(f)$ 等价于类型

$$\sum_{g:B\to A} \big((f \circ g = \mathsf{id}_B) \times (g \circ f = \mathsf{id}_A)\big).$$

我们将定义三种不同的类型，都具有性质 (i)–(iii)，我们称它们为

- 半伴随等价，
- 双可逆映射，和
- 可缩函数。

我们还将证明所有这些类型都是等价的。这些名称故意有些繁琐，因为当我们知道它们都是等价的并具有性质 (i)–(iii)后，我们将恢复简单地说 “等价” 而不需要指定我们选择哪个特定定义。但为了本章中的比较，我们需要为每个定义取不同的名称。

然而，在我们研究不同的等价概念之前，我们先多解释一下为什么需要一个不同于拟可逆性的概念。

## 4.1 拟逆

我们说过 $\mathsf{qinv}(f)$ 是不令人满意的，因为它不是一个纯粹命题，而我们宁愿一个给定的函数最多以一种方式“是等价”。然而，我们还没有给出 $\mathsf{qinv}(f)$ 不是纯粹命题的证据。在本节中我们展示一个具体的反例。

**Lemma 4.1.1.** *如果 $f : A \to B$ 使得 $\mathsf{qinv}(f)$ 有居留者，则*

$$\mathsf{qinv}(f) \simeq \Big(\prod_{x:A}(x = x)\Big).$$

*证明.* 根据假设，$f$ 是一个等价；即我们有 $e : \mathsf{isequiv}(f)$，所以 $(f, e) : A \simeq B$。根据泛等性，$\mathsf{idtoeqv} : (A = B) \to (A \simeq B)$ 是一个等价，所以我们可以假设 $(f, e)$ 具有形式 $\mathsf{idtoeqv}(p)$，对某个 $p : A = B$。然后通过路径归纳，我们可以假设 $p$ 是 $\mathsf{refl}_A$，在这种情况下 $f$ 是 $\mathsf{id}_A$。因此我们简化为证明 $\mathsf{qinv}(\mathsf{id}_A) \simeq (\prod_{(x:A)}(x = x))$。现在根据定义我们有

$$\mathsf{qinv}(\mathsf{id}_A) \equiv \sum_{g:A\to A} \big((g \sim \mathsf{id}_A) \times (g \sim \mathsf{id}_A)\big).$$

根据函数外延性，这等价于

$$\sum_{g:A\to A} \big((g = \mathsf{id}_A) \times (g = \mathsf{id}_A)\big).$$

根据??，这等价于

$$\sum_{h:\sum_{(g:A\to A)}(g=\mathsf{id}_A)} (\mathsf{pr}_1(h) = \mathsf{id}_A)$$

然而，根据定理 3.11.8，$\sum_{(g:A\to A)}(g = \mathsf{id}_A)$ 是可缩的，中心为 $(\mathsf{id}_A, \mathsf{refl}_{\mathsf{id}_A})$；因此根据定理 3.11.9，这个类型等价于 $\mathsf{id}_A = \mathsf{id}_A$。根据函数外延性，$\mathsf{id}_A = \mathsf{id}_A$ 等价于 $\prod_{(x:A)} x = x$。 □

我们注意到练习 4.3要求上述引理的一个避免泛等性的证明。

因此，我们需要的是某个承认 $\prod_{(x:A)}(x = x)$ 的非平凡元素的 $A$。将 $A$ 看作高阶广群，$\prod_{(x:A)}(x = x)$ 的居留者是从 $A$ 的恒等函子到自身的自然变换。这样的变换被称为构成**范畴的中心**，因为自然性公理要求它们与所有态射交换。经典地，如果 $A$ 只是被视为单对象广群的群，那么这恰好产生其通常群论意义上的中心。这为下面的内容提供了一些动机。

**Lemma 4.1.2.** 假设我们有类型 $A$，带有 $a : A$ 和 $q : a = a$ 使得

*(i)* 类型 $a = a$ 是一个集合。

*(ii)* 对所有 $x : A$ 我们有 $\|a = x\|$。

*(iii)* 对所有 $p : a = a$ 我们有 $p \cdot q = q \cdot p$。

则存在 $f : \prod_{(x:A)}(x = x)$ 使得 $f(a) = q$。

证明. 令 $g : \prod_{(x:A)}\|a = x\|$ 如 (ii)所给。首先我们观察到每个类型 $x =_A y$ 是一个集合。因为是集合是一个纯粹命题，我们可以应用命题截断的归纳原理，并假设 $g(x) = |p|$ 和 $g(y) = |p'|$，对 $p : a = x$ 和 $p' : a = y$。在这种情况下，与 $p$ 和 $p'^{-1}$ 复合产生等价 $(x = y) \simeq (a = a)$。但 $(a = a)$ 根据 (i)是集合，所以 $(x = y)$ 也是集合。

现在，我们想通过给每个 $x$ 分配路径 $g(x)^{-1} \cdot q \cdot g(x)$ 来定义 $f$，但这不行因为 $g(x)$ 不居留于 $a = x$ 而是居留于 $\|a = x\|$，而类型 $(x = x)$ 可能不是纯粹命题，所以我们不能使用命题截断的归纳。相反我们可以应用第 3.9 节中提到的技巧：我们唯一地刻画我们想要构造的对象。让我们对每个 $x : A$ 定义类型

$$B(x) :\equiv \sum_{(r:x=x)} \prod_{(s:a=x)} (r = s^{-1} \cdot q \cdot s).$$

我们声称对每个 $x : A$，$B(x)$ 是一个纯粹命题。由于这个声称本身是一个纯粹命题，我们可以再次应用截断的归纳并假设 $g(x) = |p|$，对某个 $p : a = x$。现在假设给定 $B(x)$ 中的 $(r, h)$ 和 $(r', h')$；则我们有

$$h(p) \cdot h'(p)^{-1} : r = r'.$$

还需证明当沿这个等式传输时 $h$ 与 $h'$ 等同，根据恒等类型和函数类型中的传输（第 2.11 节 和第 2.9 节），这简化为对任何 $s : a = x$ 证明

$$h(s) = h(p) \cdot h'(p)^{-1} \cdot h'(s)$$

。但这两边都是 $(x = x)$ 中元素之间的等式，所以它由我们上面观察到的 $(x = x)$ 是集合得出。

因此，每个 $B(x)$ 是一个纯粹命题；我们声称 $\prod_{(x:A)} B(x)$。给定 $x : A$，我们现在可以调用命题截断的归纳原理来假设 $g(x) = |p|$，对 $p : a = x$。我们定义 $r :\equiv p^{-1} \cdot q \cdot p$；要居留于 $B(x)$，还需证明对任何 $s : a = x$ 我们有 $r = s^{-1} \cdot q \cdot s$。操作路径，这简化为证明 $q \cdot (p \cdot s^{-1}) = (p \cdot s^{-1}) \cdot q$。但这正是 (iii)的一个实例。 □

**Theorem 4.1.3.** *存在类型 $A$ 和 $B$ 以及函数 $f : A \to B$ 使得 $\mathsf{qinv}(f)$ 不是一个纯粹命题。*

*证明.* 只需展示一个类型 $X$ 使得 $\prod_{(x:X)}(x = x)$ 不是一个纯粹命题。如定理 3.8.5的证明中那样定义 $X :\equiv \sum_{(A:\mathcal{U})} \|\mathbf{2} = A\|$。只需展示一个 $f : \prod_{(x:X)}(x = x)$ 使其不等于 $\lambda x.\, \mathsf{refl}_x$。

令 $a :\equiv (\mathbf{2}, |\mathsf{refl}_{\mathbf{2}}|) : X$，令 $q : a = a$ 为对应于非恒等等价 $e : \mathbf{2} \simeq \mathbf{2}$（由 $e(0_{\mathbf{2}}) :\equiv 1_{\mathbf{2}}$ 和 $e(1_{\mathbf{2}}) :\equiv 0_{\mathbf{2}}$ 定义）的路径。我们想应用引理 4.1.2来构建一个 $f$。根据 $X$ 的定义、子集类型中的等式（第 3.5 节）和泛等性，我们有 $(a = a) \simeq (\mathbf{2} \simeq \mathbf{2})$，这是一个集合，所以 (i)成立。类似地，根据 $X$ 的定义和子集类型中的等式，我们有 (ii)。最后，?? 意味着每个等价 $\mathbf{2} \simeq \mathbf{2}$ 要么等于 $\mathsf{id}_{\mathbf{2}}$ 要么等于 $e$，所以我们可以通过四向情形分析证明 (iii)。

因此，我们有 $f : \prod_{(x:X)}(x = x)$ 使得 $f(a) = q$。由于 $e$ 不等于 $\mathsf{id}_{\mathbf{2}}$，$q$ 不等于 $\mathsf{refl}_a$，因此 $f$ 不等于 $\lambda x.\, \mathsf{refl}_x$。因此，$\prod_{(x:X)}(x = x)$ 不是一个纯粹命题。 □

更一般地，引理 4.1.2意味着任何 “Eilenberg–Mac Lane 空间” $K(G, 1)$，其中 $G$ 是非平凡交换群，将提供一个反例；参见第 8 章。我们使用的类型 $X$ 原来等价于 $K(\mathbb{Z}_2, 1)$。在第 6 章中我们将看到圆 $\mathbb{S}^1 = K(\mathbb{Z}, 1)$ 是另一个容易描述的例子。

我们现在继续描述更好的等价概念。

## 4.2 半伴随等价

在第 4.1 节中我们通过丢弃一个可缩类型得出 $\mathsf{qinv}(f)$ 等价于 $\prod_{(x:A)}(x = x)$。粗略地说，类型 $\mathsf{qinv}(f)$ 包含三个数据 $g$、$\eta$ 和 $\epsilon$，其中两个（$g$ 和 $\eta$）当 $f$ 是等价时可以一起被视为可缩的。问题是移除这些数据留下了一个剩

余的（$\epsilon$）。为了解决这个问题，想法是添加一个额外的数据，它与 $\epsilon$ 一起形成可缩类型。

**Definition 4.2.1.** 函数 $f : A \to B$ 是**半伴随等价**如果存在 $g : B \to A$ 和同伦 $\eta : g \circ f \sim \mathsf{id}_A$ 和 $\epsilon : f \circ g \sim \mathsf{id}_B$ 使得存在同伦

$$\tau : \prod_{x:A} f(\eta x) = \epsilon(fx).$$

因此我们有类型 $\mathsf{ishae}(f)$，定义为

$$\sum_{(g:B\to A)} \sum_{(\eta:g\circ f\sim \mathsf{id}_A)} \sum_{(\epsilon:f\circ g\sim \mathsf{id}_B)} \prod_{(x:A)} f(\eta x) = \epsilon(fx).$$

注意在上述定义中，关联 $\eta$ 和 $\epsilon$ 的融贯条件只涉及 $f$。我们可以考虑一个涉及 $g$ 的类似融贯条件：

$$\upsilon : \prod_{y:B} g(\epsilon y) = \eta(gy)$$

以及由此产生的类似定义 $\mathsf{ishae}'(f)$。

幸运的是，结果是每个条件都蕴含另一个：

**Lemma 4.2.2.** *对于函数 $f : A \to B$ 和 $g : B \to A$ 以及同伦 $\eta : g \circ f \sim \mathsf{id}_A$ 和 $\epsilon : f \circ g \sim \mathsf{id}_B$，以下条件逻辑等价：*

- $\prod_{(x:A)} f(\eta x) = \epsilon(fx)$
- $\prod_{(y:B)} g(\epsilon y) = \eta(gy)$

*证明.* 只需证明一个方向；另一个方向通过分别用 $B$、$g$ 和 $\epsilon$ 替换 $A$、$f$ 和 $\eta$ 得到。令 $\tau : \prod_{(x:A)} f(\eta x) = \epsilon(fx)$。固定 $y : B$。使用 $\epsilon$ 的自然性并应用 $g$，我们得到以下路径交换图：

$$\begin{array}{ccc} gfgfgy & \xrightarrow{gfg(\epsilon y)} & gfgy \\ {\scriptstyle g(\epsilon(fgy))}\Vert & & \Vert{\scriptstyle g(\epsilon y)} \\ gfgy & \xrightarrow[g(\epsilon y)]{} & gy \end{array}$$

在图的左边使用 $\tau(gy)$ 给我们

$$\begin{array}{ccc} gfgfgy & \xrightarrow{gfg(\epsilon y)} & gfgy \\ {\scriptstyle gf(\eta(gy))}\Vert & & \Vert{\scriptstyle g(\epsilon y)} \\ gfgy & \xrightarrow[g(\epsilon y)]{} & gy \end{array}$$

使用 $\eta$ 与 $g \circ f$ 的交换性（推论 2.4.4），我们有

$$\begin{array}{ccc} gfgfgy & \xrightarrow{gfg(\epsilon y)} & gfgy \\ {\scriptstyle \eta(gfgy)}\Vert & & \Vert{\scriptstyle g(\epsilon y)} \\ gfgy & \xrightarrow[g(\epsilon y)]{} & gy \end{array}$$

然而，根据 $\eta$ 的自然性我们也有

$$\begin{array}{ccc} gfgfgy & \xrightarrow{gfg(\epsilon y)} & gfgy \\ {\scriptstyle \eta(gfgy)}\Vert & & \Vert{\scriptstyle \eta(gy)} \\ gfgy & \xrightarrow[g(\epsilon y)]{} & gy \end{array}$$

因此，消去除右手边同伦以外的所有项，我们有 $g(\epsilon y) = \eta(gy)$，如所愿。 □

然而，重要的是我们不在 $\mathsf{ishae}(f)$ 的定义中同时包含 $\tau$ 和 $\upsilon$（这就是名称 “半伴随等价” 的由来）。如果我们这样做，那么在消去可缩类型后我们仍然会有一个剩余数据——除非我们添加另一个更高阶的融贯条件。一般来说，我们期望如果我们在奇数个融贯条件后截止会得到一个行为良好的类型。

当然，$\mathsf{ishae}(f) \to \mathsf{qinv}(f)$ 是显然的：简单地忘记融贯数据。另一个方向是同伦论和范畴论中标准论证的一个版本。

**Theorem 4.2.3.** *对于任何 $f : A \to B$ 我们有 $\mathsf{qinv}(f) \to \mathsf{ishae}(f)$。*

*证明.* 假设 $(g, \eta, \epsilon)$ 是 $f$ 的拟逆。我们必须提供一个四元组 $(g', \eta', \epsilon', \tau)$ 见证 $f$ 是半伴随等价。要定义 $g'$ 和 $\eta'$，我们可以做显然的选择，设 $g' :\equiv g$

和 $\eta' :\equiv \eta$。然而，在定义 $\epsilon'$ 时我们需要开始担心 $\tau$ 的构造，所以我们不能直接取 $\epsilon'$ 为 $\epsilon$。相反，我们取

$$\epsilon'(b) :\equiv \epsilon(f(g(b)))^{-1} \cdot (f(\eta(g(b))) \cdot \epsilon(b)).$$

现在我们需要找到

$$\tau(a) : f(\eta(a)) = \epsilon(f(g(f(a))))^{-1} \cdot (f(\eta(g(f(a)))) \cdot \epsilon(f(a))).$$

首先注意根据推论 2.4.4，我们有 $\eta(g(f(a))) = g(f(\eta(a)))$。因此，我们可以应用 引理 2.4.3来计算

$$\begin{aligned} f(\eta(g(f(a)))) \cdot \epsilon(f(a)) &= f(g(f(\eta(a)))) \cdot \epsilon(f(a)) \\ &= \epsilon(f(g(f(a)))) \cdot f(\eta(a)) \end{aligned}$$

从中我们得到所需的路径 $\tau(a)$。 □

结合引理 4.2.2（或对称化证明），我们也有 $\mathsf{qinv}(f) \to \mathsf{ishae}'(f)$。

还需证明 $\mathsf{ishae}(f)$ 是一个纯粹命题。为此，我们需要知道等价的纤维是可缩的。

**Definition 4.2.4.** 映射 $f : A \to B$ 在点 $y : B$ 上的**纤维** 是

$$\mathsf{fib}_f(y) :\equiv \sum_{x:A} (f(x) = y).$$

在同伦论中，这就是所谓的 $f$ 的同伦纤维。第 2.5 节中的路径引理给出纤维中路径的以下刻画：

**Lemma 4.2.5.** 对于任何 $f : A \to B$、$y : B$ 和 $(x, p), (x', p') : \mathsf{fib}_f(y)$，我们有

$$\big((x, p) = (x', p')\big) \simeq \Big( \sum_{\gamma : x = x'} f(\gamma) \cdot p' = p \Big)$$

**Theorem 4.2.6.** 如果 $f : A \to B$ 是一个半伴随等价，则对于任何 $y : B$，纤维 $\mathsf{fib}_f(y)$ 是可缩的。

证明. 令 $(g, \eta, \epsilon, \tau) : \mathsf{ishae}(f)$，并固定 $y : B$。作为 $\mathsf{fib}_f(y)$ 的收缩中心我们选择 $(gy, \epsilon y)$。现在取任何 $(x, p) : \mathsf{fib}_f(y)$；我们想要构造一条从 $(gy, \epsilon y)$ 到 $(x, p)$ 的路径。根据引理 4.2.5，只需给出路径 $\gamma : gy = x$ 使得 $f(\gamma) \cdot p = \epsilon y$。我们取 $\gamma :\equiv g(p)^{-1} \cdot \eta x$。则我们有

$$\begin{aligned} f(\gamma) \cdot p &= fg(p)^{-1} \cdot f(\eta x) \cdot p \\ &= fg(p)^{-1} \cdot \epsilon(fx) \cdot p \\ &= \epsilon y \end{aligned}$$

其中第二个等式由 $\tau x$ 得出，第三个等式是 $\epsilon$ 的自然性。 □

我们现在定义封装可缩数据对的类型。以下类型将拟逆 $g$ 与一个同伦组合在一起。

**Definition 4.2.7.** 给定函数 $f : A \to B$，我们定义类型

$$\begin{aligned} \mathsf{linv}(f) &:\equiv \sum_{g:B\to A} (g \circ f \sim \mathsf{id}_A) \\ \mathsf{rinv}(f) &:\equiv \sum_{g:B\to A} (f \circ g \sim \mathsf{id}_B) \end{aligned}$$

分别为 $f$ 的**左逆** 和**右逆**。如果 $\mathsf{linv}(f)$ 有居留者，我们称 $f$ 是**左可逆的**，类似地如果 $\mathsf{rinv}(f)$ 有居留者则称为**右可逆的**。

**Lemma 4.2.8.** *如果 $f : A \to B$ 有拟逆，则*

$$\begin{aligned} (f \circ -) &: (C \to A) \to (C \to B) \\ (- \circ f) &: (B \to C) \to (A \to C) \end{aligned}$$

*也有拟逆。*

证明. 如果 $g$ 是 $f$ 的拟逆，则 $(g \circ -)$ 和 $(- \circ g)$ 分别是 $(f \circ -)$ 和 $(- \circ f)$ 的拟逆。 □

**Lemma 4.2.9.** *如果 $f : A \to B$ 有拟逆，则类型 $\mathsf{rinv}(f)$ 和 $\mathsf{linv}(f)$ 是可缩的。*

证明. 根据函数外延性，我们有

$$\mathsf{linv}(f) \simeq \sum_{g:B\to A} (g \circ f = \mathsf{id}_A).$$

但这是 $(- \circ f)$ 在 $\mathsf{id}_A$ 上的纤维，所以根据定理 4.2.8 和定理 4.2.3 和定理 4.2.6，它是可缩的。类似地，$\mathsf{rinv}(f)$ 等价于 $(f \circ -)$ 在 $\mathsf{id}_B$ 上的纤维，因此可缩。 □

接下来我们定义将另一个同伦与额外融贯数据组合在一起的类型。

**Definition 4.2.10.** 对于 $f : A \to B$、左逆 $(g, \eta) : \mathsf{linv}(f)$ 和右逆 $(g, \epsilon) : \mathsf{rinv}(f)$，我们记

$$\mathsf{lcoh}_f(g,\eta) :\equiv \sum_{(\epsilon : f\circ g \sim \mathsf{id}_B)} \prod_{(y:B)} g(\epsilon y) = \eta(gy),$$
$$\mathsf{rcoh}_f(g,\epsilon) :\equiv \sum_{(\eta : g\circ f \sim \mathsf{id}_A)} \prod_{(x:A)} f(\eta x) = \epsilon(fx).$$

**Lemma 4.2.11.** 对于任何 $f, g, \epsilon, \eta$，我们有

$$\mathsf{lcoh}_f(g,\eta) \simeq \prod_{y:B} (fgy, \eta(gy)) =_{\mathsf{fib}_g(gy)} (y, \mathsf{refl}_{gy}),$$
$$\mathsf{rcoh}_f(g,\epsilon) \simeq \prod_{x:A} (gfx, \epsilon(fx)) =_{\mathsf{fib}_f(fx)} (x, \mathsf{refl}_{fx}).$$

证明. 使用引理 4.2.5。 □

**Lemma 4.2.12.** 如果 $f$ 是半伴随等价，则对于任何 $(g, \epsilon) : \mathsf{rinv}(f)$，类型 $\mathsf{rcoh}_f(g, \epsilon)$ 是可缩的。

证明. 根据引理 4.2.11和依赖函数类型保持可缩空间的事实，只需证明对每个 $x : A$，类型 $(gfx, \epsilon(fx)) =_{\mathsf{fib}_f(fx)} (x, \mathsf{refl}_{fx})$ 是可缩的。但根据定理 4.2.6，$\mathsf{fib}_f(fx)$ 是可缩的，而可缩空间的任何路径空间本身也是可缩的。 □

**Theorem 4.2.13.** 对于任何 $f : A \to B$，类型 $\mathsf{ishae}(f)$ 是一个纯粹命题。

证明. 根据练习 3.5，只需假设 $f$ 是半伴随等价并证明 $\mathsf{ishae}(f)$ 是可缩的。现在根据 $\Sigma$ 的结合性（??），类型 $\mathsf{ishae}(f)$ 等价于

$$\sum_{u:\mathsf{rinv}(f)} \mathsf{rcoh}_f(\mathsf{pr}_1(u), \mathsf{pr}_2(u)).$$

但根据引理 4.2.9 和引理 4.2.12和 $\Sigma$ 保持可缩性的事实，后一个类型也是可缩的。 □

因此，我们已经证明 $\mathsf{ishae}(f)$ 具有类型 $\mathsf{isequiv}(f)$ 的所有三个要求。在接下来的两节中我们考虑另外两种可能性。

## 4.3 双可逆映射

使用第 4.2 节中引入的语言，我们可以将第 2.4 节中提出的定义重新表述如下。

**Definition 4.3.1.** 如果 $f : A \to B$ 同时有左逆和右逆，我们说它是**双可逆的**：

$$\mathsf{biinv}(f) :\equiv \mathsf{linv}(f) \times \mathsf{rinv}(f).$$

在第 2.4 节中我们证明了 $\mathsf{qinv}(f) \to \mathsf{biinv}(f)$ 和 $\mathsf{biinv}(f) \to \mathsf{qinv}(f)$。还需证明以下内容。

**Theorem 4.3.2.** *对于任何 $f : A \to B$，类型 $\mathsf{biinv}(f)$ 是一个纯粹命题。*

证明. 我们可以假设 $f$ 是双可逆的并证明 $\mathsf{biinv}(f)$ 是可缩的。但由于 $\mathsf{biinv}(f) \to \mathsf{qinv}(f)$，根据引理 4.2.9，在这种情况下 $\mathsf{linv}(f)$ 和 $\mathsf{rinv}(f)$ 都是可缩的，而可缩类型的乘积是可缩的。 □

注意这也符合第 4.2 节开头提出的方案：我们将 $g$ 和 $\eta$ 组合成可缩类型，并添加一个与 $\epsilon$ 组合成可缩类型的额外数据。不同之处在于，我们不是添加一个更高阶的数据（2 维路径）来与 $\epsilon$ 组合，而是添加一个更低阶的数据（与左逆分开的右逆）。

**Corollary 4.3.3.** *对于任何 $f : A \to B$ 我们有 $\mathsf{biinv}(f) \simeq \mathsf{ishae}(f)$。*

证明. 我们有 $\mathsf{biinv}(f) \to \mathsf{qinv}(f) \to \mathsf{ishae}(f)$ 和 $\mathsf{ishae}(f) \to \mathsf{qinv}(f) \to \mathsf{biinv}(f)$。由于 $\mathsf{ishae}(f)$ 和 $\mathsf{biinv}(f)$ 都是纯粹命题，等价由引理 3.3.3得出。 □

## 4.4 可缩纤维

注意我们关于 $\mathsf{ishae}(f)$ 和 $\mathsf{biinv}(f)$ 的证明本质上使用了等价的纤维是可缩的这一事实。事实上，这个性质本身就是等价的充分定义。

**Definition 4.4.1** (可缩映射)**.** 如果对所有 $y : B$，纤维 $\mathsf{fib}_f(y)$ 是可缩的，则映射 $f : A \to B$ 是**可缩的**。

因此，类型 $\mathsf{isContr}(f)$ 定义为

$$\mathsf{isContr}(f) :\equiv \prod_{y:B} \mathsf{isContr}(\mathsf{fib}_f(y)) \tag{4.4.2}$$

注意在第 3.11 节中我们定义了类型可缩的意义。这里我们定义映射可缩的意义。我们的术语遵循一般的同伦论实践，即如果一个映射的所有（同伦）纤维都具有某个性质，则说该映射具有该性质。因此，类型 $A$ 可缩当且仅当映射 $A \to \mathbf{1}$ 可缩。从第 7 章开始我们也将可缩映射和类型称为 $(-2)$-截断的。

我们已经在定理 4.2.6中证明了 $\mathsf{ishae}(f) \to \mathsf{isContr}(f)$。反过来：

**Theorem 4.4.3.** 对于任何 $f : A \to B$ 我们有 $\mathsf{isContr}(f) \to \mathsf{ishae}(f)$。

证明. 令 $P : \mathsf{isContr}(f)$。我们通过将每个 $y : B$ 送到 $y$ 上纤维的收缩中心来定义逆映射 $g : B \to A$：

$$g(y) :\equiv \mathsf{pr}_1(\mathsf{pr}_1(Py)).$$

因此我们可以通过将 $y$ 映到 $g(y)$ 确实属于 $y$ 上纤维的见证来定义同伦 $\epsilon$：

$$\epsilon(y) :\equiv \mathsf{pr}_2(\mathsf{pr}_1(Py)).$$

还需定义 $\eta$ 和 $\tau$。这当然相当于给出 $\mathsf{rcoh}_f(g, \epsilon)$ 的元素。根据引理 4.2.11，这等于对每个 $x : A$ 给出 $f$ 在 $fx$ 上纤维中从 $(gfx, \epsilon(fx))$ 到 $(x, \mathsf{refl}_{fx})$ 的

路径。但这很容易：对于任何 $x : A$，类型 $\mathsf{fib}_f(fx)$ 根据假设是可缩的，因此这样的路径必须存在。我们可以显式构造它为

$$\big(\mathsf{pr}_2(P(fx))(gfx, \epsilon(fx))\big)^{-1} \cdot \big(\mathsf{pr}_2(P(fx))(x, \mathsf{refl}_{fx})\big). \qquad \square$$

也容易看出：

**Lemma 4.4.4.** 对于任何 $f$，类型 $\mathsf{isContr}(f)$ 是一个纯粹命题。

证明. 根据定理 3.11.4，每个类型 $\mathsf{isContr}(\mathsf{fib}_f(y))$ 是一个纯粹命题。因此，根据定理 3.6.2， (4.4.2) 也是。 $\square$

**Theorem 4.4.5.** 对于任何 $f : A \to B$ 我们有 $\mathsf{isContr}(f) \simeq \mathsf{ishae}(f)$。

证明. 我们已经建立了逻辑等价 $\mathsf{isContr}(f) \Leftrightarrow \mathsf{ishae}(f)$，且两者都是纯粹命题（定理 4.4.4 和定理 4.2.13）。因此，引理 3.3.3适用。 $\square$

通常，我们通过展示拟逆来证明函数是等价，但有时这个定义更方便。例如，它意味着在证明函数是等价时，我们可以自由地假设它的陪域有居留者。

**Corollary 4.4.6.** 如果 $f : A \to B$ 使得 $B \to \mathsf{isequiv}(f)$，则 $f$ 是一个等价。

证明. 要证明 $f$ 是等价，只需证明对任何 $y : B$，$\mathsf{fib}_f(y)$ 是可缩的。但如果 $e : B \to \mathsf{isequiv}(f)$，则给定任何这样的 $y$ 我们有 $e(y) : \mathsf{isequiv}(f)$，所以 $f$ 是等价，因此 $\mathsf{fib}_f(y)$ 是可缩的，如所愿。 $\square$

## 4.5 关于等价的定义

我们已经证明所有三个等价定义都满足三个理想性质并且两两等价：

$$\mathsf{isContr}(f) \simeq \mathsf{ishae}(f) \simeq \mathsf{biinv}(f).$$

（还有更多可能的等价定义，但我们止步于这三个。参见练习 3.8和本章的练习了解更多。）因此，我们可以选择其中任何一个作为 $\mathsf{isequiv}(f)$ 的“正式”定义。为了明确起见，我们选择定义

$$\mathsf{isequiv}(f) :\equiv \mathsf{ishae}(f).$$

这个选择对形式化是有利的，因为 $\mathsf{ishae}(f)$ 包含最直接有用的数据。另一方面，对于其他目的，$\mathsf{biinv}(f)$ 通常更容易处理，因为它不包含 2 维路径，其两个对称部分可以独立处理。然而，就本书而言，具体选择不会有太大差别。

在本章的其余部分，我们研究等价的一些其他性质和刻画。

## 4.6 满射与嵌入

当 $A$ 和 $B$ 是集合且 $f : A \to B$ 是等价时，我们也称它为**同构** 或**双射**。（我们对不是集合的类型避免这些词，因为在同伦论和高阶范畴论中它们通常表示比同伦等价更严格的“相同性”概念。）在集合论中，函数是双射当且仅当它既是单射又是满射。在类型论中也是如此，如果我们适当地表述这些条件。为清楚起见，当处理不是集合的类型时，我们将使用嵌入而不是单射。

**Definition 4.6.1.** 令 $f : A \to B$。

(i) 如果对每个 $b : B$ 我们有 $\|\mathsf{fib}_f(b)\|$，我们说 $f$ 是**满射** （或一个**满射函数**）。

(ii) 如果对每个 $x, y : A$，函数 $\mathsf{ap}_f : (x =_A y) \to (f(x) =_B f(y))$ 是等价，我们说 $f$ 是**嵌入**。

换句话说，$f$ 是满射如果 $f$ 的每个纤维纯粹有居留者，或等价地如果对所有 $b : B$ 纯粹存在 $a : A$ 使得 $f(a) = b$。用传统逻辑记号，$f$ 是满射如果 $\forall(b : B).\, \exists(a : A).\, (f(a) = b)$。这必须与更强的断言 $\prod_{(b:B)} \sum_{(a:A)} (f(a) = b)$ 区分；如果这成立我们说 $f$ 是**分裂满射**。（由于后一类型等价于 $\sum_{(g:B\to A)} \prod_{(b:B)} (f(g(b)) = b)$，是分裂满射与第 3.11 节中定义的是收缩是一样的。）

第 3.8 节中的选择公理恰好说每个集合之间的满射都是分裂的。然而，在泛等公理存在的情况下，所有满射都是分裂的这一说法是假的。在定理 3.8.5中我们构造了类型族 $Y : X \to \mathcal{U}$ 使得 $\prod_{(x:X)} \|Y(x)\|$ 但 $\neg \prod_{(x:X)} Y(x)$；对于任何这样的族，第一投影 $(\sum_{(x:X)} Y(x)) \to X$ 是不分裂的满射。

如果 $A$ 和 $B$ 是集合，则根据引理 3.3.3，$f$ 是嵌入当且仅当

$$\prod_{x,y:A} (f(x) =_B f(y)) \to (x =_A y). \tag{4.6.2}$$

在这种情况下我们说 $f$ 是**单射**，或一个**单射函数**。我们对不是集合的类型避免这个词，因为它们可能被解释为 (4.6.2)，这对非集合是一个行为不良的概念。任何集合之间的函数是满射当且仅当它是适当意义上的满态射也是真的，但这对更一般的类型失效，满射性通常是更重要的概念。

**Theorem 4.6.3.** *函数 $f : A \to B$ 是等价当且仅当它既是满射又是嵌入。*

*证明.* 如果 $f$ 是等价，则每个 $\mathsf{fib}_f(b)$ 是可缩的，因此 $\|\mathsf{fib}_f(b)\|$ 也是，所以 $f$ 是满射。我们在定理 2.11.1中证明了任何等价都是嵌入。

反过来，假设 $f$ 是满射嵌入。令 $b : B$；我们证明 $\sum_{(x:A)}(f(x) = b)$ 是可缩的。由于 $f$ 是满射，纯粹存在 $a : A$ 使得 $f(a) = b$。因此，$f$ 在 $b$ 上的纤维有居留者；还需证明它是纯粹命题。为此，假设给定 $x, y : A$，带有 $p : f(x) = b$ 和 $q : f(y) = b$。则由于 $\mathsf{ap}_f$ 是等价，存在 $r : x = y$ 使得 $\mathsf{ap}_f(r) = p \cdot q^{-1}$。然而，使用 $\Sigma$-类型中路径的刻画，后一等式重排为 $r_*(p) = q$。因此，与 $r$ 一起它展示了 $f$ 在 $b$ 上纤维中 $(x, p) = (y, q)$。 □

**Corollary 4.6.4.** *对于任何 $f : A \to B$ 我们有*

$$\mathsf{isequiv}(f) \simeq (\mathsf{isEmbedding}(f) \times \mathsf{isSurjective}(f)).$$

*证明.* 是满射和是嵌入都是纯粹命题；现在应用引理 3.3.3。 □

当然，这不能用作 “等价” 的定义，因为嵌入的定义引用了等价。然而，这个刻画仍然可以有用；参见??。我们将在第 7 章中推广它。

## 4.7 等价的封闭性质

我们已经在定理 2.4.12中看到等价对复合封闭。此外，我们有：

**Theorem 4.7.1** (三取二性质)**.** 假设 $f : A \to B$ 和 $g : B \to C$。如果 $f$、$g$ 和 $g \circ f$ 中任意两个是等价，则第三个也是。

证明. 如果 $g \circ f$ 和 $g$ 是等价，则 $(g \circ f)^{-1} \circ g$ 是 $f$ 的拟逆。一方面，我们有 $(g \circ f)^{-1} \circ g \circ f \sim \mathsf{id}_A$，另一方面我们有

$$\begin{aligned} f \circ (g \circ f)^{-1} \circ g &\sim g^{-1} \circ g \circ f \circ (g \circ f)^{-1} \circ g \\ &\sim g^{-1} \circ g \\ &\sim \mathsf{id}_B. \end{aligned}$$

类似地，如果 $g \circ f$ 和 $f$ 是等价，则 $f \circ (g \circ f)^{-1}$ 是 $g$ 的拟逆。 □

这是同伦论中等价的标准封闭条件。同样众所周知的是它们对收缩核封闭，意义如下。

**Definition 4.7.2.** 如果存在图

$$\begin{array}{ccccc} A & \xrightarrow{s} & X & \xrightarrow{r} & A \\ {\scriptstyle g}\downarrow & & {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[s']{} & Y & \xrightarrow[r']{} & B \end{array}$$

使得存在

(i) 同伦 $R : r \circ s \sim \mathsf{id}_A$。

(ii) 同伦 $R' : r' \circ s' \sim \mathsf{id}_B$。

(iii) 同伦 $L : f \circ s \sim s' \circ g$。

(iv) 同伦 $K : g \circ r \sim r' \circ f$。

(v) 对每个 $a : A$，路径 $H(a)$ 见证方块的交换性

$$\begin{array}{ccc} g(r(s(a))) & \xrightarrow{K(s(a))} & r'(f(s(a))) \\ {\scriptstyle g(R(a))}\Vert & & \Vert{\scriptstyle r'(L(a))} \\ g(a) & \xrightarrow[R'(g(a))^{-1}]{} & r'(s'(g(a))) \end{array}$$

则称函数 $g : A \to B$ 是函数 $f : X \to Y$ 的**收缩核**。

回忆在第 3.11 节中我们定义了一个类型是另一个类型的收缩核的意义。这是上述定义的特例，其中 $B$ 和 $Y$ 是 $\mathbf{1}$。反过来，就像可缩性一样，映射的收缩诱导其纤维的收缩。

**Lemma 4.7.3.** *如果函数 $g : A \to B$ 是函数 $f : X \to Y$ 的收缩核，则对每个 $b : B$，$\mathsf{fib}_g(b)$ 是 $\mathsf{fib}_f(s'(b))$ 的收缩核，其中 $s' : B \to Y$ 如定义 4.7.2中那样。*

*证明.* 假设 $g : A \to B$ 是 $f : X \to Y$ 的收缩核。则对任何 $b : B$ 我们有函数

$$\begin{aligned} &\varphi_b : \mathsf{fib}_g(b) \to \mathsf{fib}_f(s'(b)), \qquad \varphi_b(a,p) :\equiv (s(a), L(a) \cdot s'(p)), \\ &\psi_b : \mathsf{fib}_f(s'(b)) \to \mathsf{fib}_g(b), \qquad \psi_b(x,q) :\equiv (r(x), K(x) \cdot r'(q) \cdot R'(b)). \end{aligned}$$

则 $\psi_b(\varphi_b(a,p)) \equiv (r(s(a)), K(s(a)) \cdot r'(L(a) \cdot s'(p)) \cdot R'(b))$。我们声称对所有 $b : B$，$\psi_b$ 是以 $\varphi_b$ 为截面的收缩，也就是说对所有 $(a,p) : \mathsf{fib}_g(b)$ 我们有 $\psi_b(\varphi_b(a,p)) = (a,p)$。换句话说，我们想证明

$$\prod_{(b:B)} \prod_{(a:A)} \prod_{(p:g(a)=b)} \psi_b(\varphi_b(a,p)) = (a,p).$$

通过重排前两个 $\Pi$ 并应用定理 3.11.9的一个版本，这等价于

$$\prod_{a:A} \psi_{g(a)}(\varphi_{g(a)}(a, \mathsf{refl}_{g(a)})) = (a, \mathsf{refl}_{g(a)}).$$

对于任何 $a$，根据定理 2.7.2，这个对的等式等价于一对等式。第一分量由 $R(a) : r(s(a)) = a$ 相等，所以只需证明

$$R(a)_*\big(K(s(a)) \cdot r'(L(a)) \cdot R'(g(a))\big) = \mathsf{refl}_{g(a)}.$$

但这个传输计算为 $g(R(a))^{-1} \cdot K(s(a)) \cdot r'(L(a)) \cdot R'(g(a))$，所以所需路径由 $H(a)$ 给出。 □

**Theorem 4.7.4.** *如果 $g$ 是等价 $f$ 的收缩核，则 $g$ 也是等价。*

证明. 根据引理 4.7.3，$g$ 的每个纤维是 $f$ 的一个纤维的收缩核。因此，根据定理 3.11.7，如果后者都可缩，则前者也都可缩。 □

最后，我们证明纤维化等价可以用全空间等价来刻画。为了解释术语，回忆第 2.3 节中类型族 $P : A \to \mathcal{U}$ 可以被视为全空间 $\sum_{(x:A)} P(x)$ 上的纤维化，纤维化是投影 $\mathsf{pr}_1 : \sum_{(x:A)} P(x) \to A$。从这个观点看，给定两个类型族 $P, Q : A \to \mathcal{U}$，我们可以称函数 $f : \prod_{(x:A)}(P(x) \to Q(x))$ 为**纤维化映射**或**纤维化变换**。 这样的映射诱导全空间上的函数：

**Definition 4.7.5.** 给定类型族 $P, Q : A \to \mathcal{U}$ 和映射 $f : \prod_{(x:A)} P(x) \to Q(x)$，我们定义

$$\mathsf{total}(f) :\equiv \lambda w.\, (\mathsf{pr}_1 w, f(\mathsf{pr}_1 w, \mathsf{pr}_2 w)) : \sum_{x:A} P(x) \to \sum_{x:A} Q(x).$$

**Theorem 4.7.6.** 假设 $f$ 是类型 $A$ 上族 $P$ 和 $Q$ 之间的纤维化变换，令 $x : A$ 和 $v : Q(x)$。则我们有等价

$$\mathsf{fib}_{\mathsf{total}(f)}((x, v)) \simeq \mathsf{fib}_{f(x)}(v).$$

证明. 我们计算：

$$\begin{aligned}
\mathsf{fib}_{\mathsf{total}(f)}((x, v)) &\equiv \sum_{w:\sum_{(x:A)} P(x)} (\mathsf{pr}_1 w, f(\mathsf{pr}_1 w, \mathsf{pr}_2 w)) = (x, v) \\
&\simeq \sum_{(a:A)} \sum_{(u:P(a))} (a, f(a, u)) = (x, v) && \text{(根据 ??)} \\
&\simeq \sum_{(a:A)} \sum_{(u:P(a))} \sum_{(p:a=x)} p_*(f(a, u)) = v && \text{(根据定理 2.7.2)} \\
&\simeq \sum_{(a:A)} \sum_{(p:a=x)} \sum_{(u:P(a))} p_*(f(a, u)) = v \\
&\simeq \sum_{u:P(x)} f(x, u) = v && (*) \\
&\equiv \mathsf{fib}_{f(x)}(v).
\end{aligned}$$

等价 $(*)$ 由定理 3.11.9 和定理 3.11.8 和?? 得出。 □

如果每个 $f(x) : P(x) \to Q(x)$ 是等价，我们说纤维化变换 $f : \prod_{(x:A)} P(x) \to Q(x)$ 是**纤维化等价** 。

**Theorem 4.7.7.** *假设 $f$ 是类型 $A$ 上族 $P$ 和 $Q$ 之间的纤维化变换。则 $f$ 是纤维化等价当且仅当 $\mathsf{total}(f)$ 是等价。*

*证明.* 令 $f$、$P$、$Q$ 和 $A$ 如定理声明中那样。根据4.7.6，对所有 $x : A$ 和 $v : Q(x)$，$\mathsf{fib}_{\mathsf{total}(f)}((x, v))$ 可缩当且仅当 $\mathsf{fib}_{f(x)}(v)$ 可缩。因此，$\mathsf{fib}_{\mathsf{total}(f)}(w)$ 对所有 $w : \sum_{(x:A)} Q(x)$ 可缩当且仅当 $\mathsf{fib}_{f(x)}(v)$ 对所有 $x : A$ 和 $v : Q(x)$ 可缩。 □

## 4.8 对象分类子

在类型论中我们有一个基本的类型族概念，即函数 $B : A \to \mathcal{U}$。我们已经看到这样的族的行为有点像同伦论中的纤维化，纤维化是投影 $\mathsf{pr}_1 : \sum_{(a:A)} B(a) \to A$。同伦论中的一个基本事实是每个映射都等价于一个纤维化。有了泛等性，我们可以在类型论中证明同样的事情。

**Lemma 4.8.1.** *对于任何类型族 $B : A \to \mathcal{U}$，$\mathsf{pr}_1 : \sum_{(x:A)} B(x) \to A$ 在 $a : A$ 上的纤维等价于 $B(a)$:*

$$\mathsf{fib}_{\mathsf{pr}_1}(a) \simeq B(a)$$

*证明.* 我们有

$$\begin{aligned}
\mathsf{fib}_{\mathsf{pr}_1}(a) &:\equiv \sum_{u:\sum_{(x:A)} B(x)} \mathsf{pr}_1(u) = a \\
&\simeq \sum_{(x:A)} \sum_{(b:B(x))} (x = a) \\
&\simeq \sum_{(x:A)} \sum_{(p:x=a)} B(x) \\
&\simeq B(a)
\end{aligned}$$

使用恒等类型的左泛性质。 □

**Lemma 4.8.2.** 对于任何函数 $f : A \to B$，我们有 $A \simeq \sum_{(b:B)} \mathsf{fib}_f(b)$。

证明. 我们有

$$\begin{aligned}\sum_{b:B} \mathsf{fib}_f(b) &:\equiv \sum_{(b:B)} \sum_{(a:A)} (f(a) = b) \\ &\simeq \sum_{(a:A)} \sum_{(b:B)} (f(a) = b) \\ &\simeq A\end{aligned}$$

使用 $\sum_{(b:B)}(f(a) = b)$ 可缩的事实。 □

**Theorem 4.8.3.** 对于任何类型 $B$，有等价

$$\chi : \Big(\sum_{A:\mathcal{U}} (A \to B)\Big) \simeq (B \to \mathcal{U}).$$

证明. 我们必须构造拟逆

$$\begin{aligned}\chi &: \Big(\sum_{A:\mathcal{U}} (A \to B)\Big) \to B \to \mathcal{U} \\ \psi &: (B \to \mathcal{U}) \to \Big(\sum_{A:\mathcal{U}} (A \to B)\Big).\end{aligned}$$

我们定义 $\chi((A, f), b) :\equiv \mathsf{fib}_f(b)$，$\psi(P) :\equiv \Big((\sum_{(b:B)} P(b)), \mathsf{pr}_1\Big)$。现在我们必须验证 $\chi \circ \psi \sim \mathsf{id}$ 和 $\psi \circ \chi \sim \mathsf{id}$。

(i) 令 $P : B \to \mathcal{U}$。根据定理 4.8.1，对任何 $b : B$ 有 $\mathsf{fib}_{\mathsf{pr}_1}(b) \simeq P(b)$，所以直接得出 $P \sim \chi(\psi(P))$。

(ii) 令 $f : A \to B$ 是函数。我们必须找到路径

$$\Big(\textstyle\sum_{(b:B)} \mathsf{fib}_f(b),\ \mathsf{pr}_1\Big) = (A, f).$$

首先注意根据定理 4.8.2，我们有 $e : \sum_{(b:B)} \mathsf{fib}_f(b) \simeq A$，其中 $e(b, a, p) :\equiv a$ 和 $e^{-1}(a) :\equiv (f(a), a, \mathsf{refl}_{f(a)})$。根据定理 2.7.2，还需证明 $(\mathsf{ua}(e))_*(\mathsf{pr}_1) = f$。但根据泛等性的计算规则和 (2.9.4)，我们有 $(\mathsf{ua}(e))_*(\mathsf{pr}_1) = \mathsf{pr}_1 \circ e^{-1}$，而 $e^{-1}$ 的定义直接给出 $\mathsf{pr}_1 \circ e^{-1} \equiv f$。 □

特别地，这意味着我们有高阶拓扑斯理论意义上的对象分类子。回忆2.1.7中 $\mathcal{U}_\bullet$ 表示点化类型的类型 $\sum_{(A:\mathcal{U})} A$。

**Theorem 4.8.4.** 令 $f : A \to B$ 是函数。则图

$$\begin{array}{ccc} A & \xrightarrow{\vartheta_f} & \mathcal{U}_\bullet \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \mathsf{pr}_1} \\ B & \xrightarrow[\chi_f]{} & \mathcal{U} \end{array}$$

是拉回方块（参见??）。这里函数 $\vartheta_f$ 定义为

$$\lambda a.\, (\mathsf{fib}_f(f(a)), (a, \mathsf{refl}_{f(a)})).$$

证明. 注意我们有等价

$$\begin{aligned} A &\simeq \sum_{b:B} \mathsf{fib}_f(b) \\ &\simeq \sum_{(b:B)} \sum_{(X:\mathcal{U})} \sum_{(p:\mathsf{fib}_f(b)=X)} X \\ &\simeq \sum_{(b:B)} \sum_{(X:\mathcal{U})} \sum_{(x:X)} \mathsf{fib}_f(b) = X \\ &\simeq \sum_{(b:B)} \sum_{(Y:\mathcal{U}_\bullet)} \mathsf{fib}_f(b) = \mathsf{pr}_1 Y \\ &\equiv B \times_{\mathcal{U}} \mathcal{U}_\bullet \end{aligned}$$

这给我们复合等价 $e : A \simeq B \times_{\mathcal{U}} \mathcal{U}_\bullet$。我们可以逐步显示这个复合等价的作用

$$\begin{aligned} a &\mapsto (f(a),\ (a, \mathsf{refl}_{f(a)})) \\ &\mapsto (f(a),\ \mathsf{fib}_f(f(a)),\ \mathsf{refl}_{\mathsf{fib}_f(f(a))},\ (a, \mathsf{refl}_{f(a)})) \\ &\mapsto (f(a),\ \mathsf{fib}_f(f(a)),\ (a, \mathsf{refl}_{f(a)}),\ \mathsf{refl}_{\mathsf{fib}_f(f(a))}) \\ &\mapsto (f(a),\ (\mathsf{fib}_f(f(a)),\ (a, \mathsf{refl}_{f(a)})),\ \mathsf{refl}_{\mathsf{fib}_f(f(a))}). \end{aligned}$$

因此，我们得到同伦 $f \sim \mathsf{pr}_1 \circ e$ 和 $\vartheta_f \sim \mathsf{pr}_2 \circ e$。 □

## 4.9 泛等性蕴含函数外延性

在本章的最后一节，我们包含一个泛等公理蕴含函数外延性的证明。因此，在本节中我们不假设函数外延性公理。证明由两步组成。首先我们在4.9.4中证明泛等公理蕴含函数外延性的一个弱形式,定义在下面的4.9.1中。弱函数外延性原理反过来蕴含通常的函数外延性，而且不需要泛等公理（4.9.5）。

令 $\mathcal{U}$ 是一个宇宙；我们将明确指出在哪里假设它是泛等的。

**Definition 4.9.1. 弱函数外延性原理** 断言对于类型 $A$ 上的任何类型族 $P : A \to \mathcal{U}$，存在函数

$$\Big(\prod_{x:A} \mathsf{isContr}(P(x))\Big) \to \mathsf{isContr}\Big(\prod_{x:A} P(x)\Big)$$

。

下面的引理使用函数外延性很容易证明；这里的要点是它也由泛等性得出，而不需要单独假设函数外延性。

**Lemma 4.9.2.** 假设 $\mathcal{U}$ 是泛等的，对于任何 $A, B, X : \mathcal{U}$ 和任何 $e : A \simeq B$，有等价

$$(X \to A) \simeq (X \to B)$$

其底层映射是与 $e$ 的底层函数的后复合。

证明. 如引理 4.1.1的证明中那样，我们可以假设 $e = \mathsf{idtoeqv}(p)$，对某个 $p : A = B$。然后通过路径归纳，我们可以假设 $p$ 是 $\mathsf{refl}_A$，使得 $e = \mathsf{id}_A$。但在这种情况下，与 $e$ 的后复合是恒等，因此是等价。 □

**Corollary 4.9.3.** 令 $P : A \to \mathcal{U}$ 是可缩类型族，即

$$\prod_{x:A} \mathsf{isContr}(P(x)).$$

则投影 $\mathsf{pr}_1 : (\sum_{(x:A)} P(x)) \to A$ 是等价。假设 $\mathcal{U}$ 是泛等的，直接得出与 $\mathsf{pr}_1$ 的后复合给出等价

$$\alpha : \Big(A \to \sum_{x:A} P(x)\Big) \simeq (A \to A).$$

证明. 根据定理 4.8.1，对于 $\mathsf{pr}_1 : (\sum_{(x:A)} P(x)) \to A$ 和 $x : A$，我们有等价

$$\mathsf{fib}_{\mathsf{pr}_1}(x) \simeq P(x).$$

因此只要每个 $P(x)$ 可缩，$\mathsf{pr}_1$ 就是等价。断言现在是4.9.2的结果。 □

特别地，上述等价在 $\mathsf{id}_A$ 上的同伦纤维是可缩的。因此，我们可以通过证明依赖函数类型 $\prod_{(x:A)} P(x)$ 是 $\mathsf{fib}_\alpha(\mathsf{id}_A)$ 的收缩核来证明泛等性蕴含弱函数外延性。

**Theorem 4.9.4.** *在泛等宇宙 $\mathcal{U}$ 中，假设 $P : A \to \mathcal{U}$ 是可缩类型族，令 $\alpha$ 是4.9.3的函数。则 $\prod_{(x:A)} P(x)$ 是 $\mathsf{fib}_\alpha(\mathsf{id}_A)$ 的收缩核。作为结果，$\prod_{(x:A)} P(x)$ 是可缩的。换句话说，泛等公理蕴含弱函数外延性原理。*

证明. 定义函数

$$\begin{aligned} \varphi &: (\textstyle\prod_{(x:A)} P(x)) \to \mathsf{fib}_\alpha(\mathsf{id}_A), \\ \varphi(f) &:\equiv (\lambda x.\, (x, f(x)), \mathsf{refl}_{\mathsf{id}_A}), \end{aligned}$$

和

$$\begin{aligned} \psi &: \mathsf{fib}_\alpha(\mathsf{id}_A) \to \textstyle\prod_{(x:A)} P(x), \\ \psi(g, p) &:\equiv \lambda x.\, \mathsf{happly}(p, x)_*(\mathsf{pr}_2(g(x))). \end{aligned}$$

则根据依赖函数类型的唯一性原理，$\psi(\varphi(f)) = \lambda x.\, f(x)$，即 $f$。 □

我们现在证明弱函数外延性蕴含通常的函数外延性。回忆 (2.9.2) 中函数 $\mathsf{happly}(f, g) : (f = g) \to (f \sim g)$，它将函数的相等转换为同伦。在下面的证明中，不使用泛等公理。

**Theorem 4.9.5.** *弱函数外延性蕴含函数外延性2.9.3。*

证明. 我们想证明

$$\prod_{(A:\mathcal{U})} \prod_{(P:A\to\mathcal{U})} \prod_{(f,g:\prod_{(x:A)} P(x))} \mathsf{isequiv}(\mathsf{happly}(f, g)).$$

由于纤维化映射在全空间上诱导等价当且仅当它纤维化地是等价（根据定理 4.7.7），只需证明类型

$$\Big(\sum_{g:\prod_{(x:A)} P(x)} (f = g)\Big) \to \sum_{g:\prod_{(x:A)} P(x)} (f \sim g)$$

的函数（由 $\lambda(g : \prod_{(x:A)} P(x)).\, \mathsf{happly}(f, g)$ 诱导）是等价。由于左边的类型根据定理 3.11.8可缩，只需证明右边的类型：

$$\sum_{(g:\prod_{(x:A)} P(x))} \prod_{(x:A)} f(x) = g(x) \tag{4.9.6}$$

可缩。现在定理 2.15.7说这等价于

$$\prod_{(x:A)} \sum_{(u:P(x))} f(x) = u. \tag{4.9.7}$$

定理 2.15.7的证明使用函数外延性，但只用于一个复合。因此，不假设函数外延性，我们可以得出 (4.9.6) 是 (4.9.7) 的收缩核。而 (4.9.7) 是可缩类型的乘积，根据弱函数外延性原理可缩；因此 (4.9.6) 也可缩。 □

## Notes

配备拟逆的连续映射空间具有错误的同伦类型而不能是 “同伦等价空间” 这一事实在代数拓扑中是众所周知的。在那个背景下，“同伦等价空间” $(A \simeq B)$ 通常简单地定义为函数空间 $(A \to B)$ 中由同伦等价组成的子空间。在类型论中，这最接近于 $\sum_{(f:A\to B)} \|\mathsf{qinv}(f)\|$；参见练习 3.8。

同伦类型论中给出的第一个等价定义是我们称为 $\mathsf{isContr}(f)$ 的那个，它是由 Voevodsky 提出的。其他定义的可能性随后被各人观察到。关于伴随等价的基本定理如引理 4.2.2 和定理 4.2.3是高阶范畴论和同伦论中标准事实的改编。使用双可逆性作为等价的定义是 André Joyal 建议的。

第 4.6 节 和第 4.7 节中讨论的等价性质在同伦论中是众所周知的。它们中的大多数首先由 Voevodsky 在类型论中证明。

每个函数都等价于一个纤维化是同伦论中的标准事实。$(\infty, 1)$-范畴 论中对象分类子 的概念（定理 4.8.3的范畴类比）归功于 Rezk（参见 [Rez05, Lur09]）。

最后，泛等性蕴含函数外延性的事实（第 4.9 节）归功于 Voevodsky。我们的证明是他的证明的简化。练习 4.9也归功于 Voevodsky。

## Exercises

*Exercise* 4.1. 考虑 $f : A \to B$ 的“双边伴随等价数据”类型，

$$\sum_{(g:B\to A)} \sum_{(\eta:g\circ f\sim \mathsf{id}_A)} \sum_{(\epsilon:f\circ g\sim \mathsf{id}_B)} \Big(\prod_{x:A} f(\eta x) = \epsilon(fx)\Big) \times \Big(\prod_{y:B} g(\epsilon y) = \eta(gy)\Big).$$

根据引理 4.2.2，我们知道如果 $f$ 是等价，则这个类型有居留者。给出这个类型的刻画，类似于引理 4.1.1。

你能给出一个例子说明这个类型一般不是纯粹命题吗？（这在第 6 章之后会更容易。）

*Exercise* 4.2. 证明对于任何 $A, B : \mathcal{U}$，以下类型等价于 $A \simeq B$。

$$\sum_{R:A\to B\to\mathcal{U}} \Big(\prod_{a:A} \mathsf{isContr}\Big(\sum_{b:B} R(a,b)\Big)\Big) \times \Big(\prod_{b:B} \mathsf{isContr}\Big(\sum_{a:A} R(a,b)\Big)\Big).$$

你能从中提取出满足 $\mathsf{isequiv}(f)$ 三个要求的类型定义吗？

*Exercise* 4.3. 重新表述引理 4.1.1的证明，不使用泛等性。

*Exercise* 4.4 (不稳定八面体公理). 假设 $f : A \to B$ 和 $g : B \to C$ 以及 $b : B$。

(i) 证明存在自然映射 $\mathsf{fib}_{g\circ f}(g(b)) \to \mathsf{fib}_g(g(b))$，其在 $(b, \mathsf{refl}_{g(b)})$ 上的纤维等价于 $\mathsf{fib}_f(b)$。

(ii) 证明 $\mathsf{fib}_{g\circ f}(c) \simeq \sum_{(w:\mathsf{fib}_g(c))} \mathsf{fib}_f(\mathsf{pr}_1 w)$。

*Exercise* 4.5. 证明等价满足六取二性质：给定 $f : A \to B$ 和 $g : B \to C$ 以及 $h : C \to D$，如果 $g \circ f$ 和 $h \circ g$ 是等价，则 $f$、$g$、$h$ 和 $h \circ g \circ f$ 也是。用这个给出定理 2.11.1的高层次证明。

*Exercise* 4.6. 对于 $A, B : \mathcal{U}$，定义

$$\mathsf{idtoqinv}_{A,B} : (A = B) \to \sum_{f:A\to B} \mathsf{qinv}(f)$$

通过路径归纳以显然的方式。令 **qinv-泛等性**表示泛等公理的修改形式，它断言对所有 $A, B : \mathcal{U}$，函数 $\mathsf{idtoqinv}_{A,B}$ 有拟逆。

(i) 证明 qinv-泛等性可以在第 4.9 节的函数外延性证明中代替泛等性使用。

(ii) 证明 qinv-泛等性可以在定理 4.1.3的证明中代替泛等性使用。

(iii) 证明 qinv-泛等性是不一致的（即允许构造 $\mathbf{0}$ 的居留者）。因此，在泛等性的陈述中使用 "好的" isequiv 版本是必要的。

*Exercise* 4.7. 证明函数 $f : A \to B$ 是嵌入当且仅当以下两个条件成立：

(i) $f$ 是左可消的，即对任何 $x, y : A$，如果 $f(x) = f(y)$ 则 $x = y$。

(ii) 对任何 $x : A$，映射 $\mathsf{ap}_f : \Omega(A, x) \to \Omega(B, f(x))$ 是等价。

（特别地，如果 $A$ 是集合，则 $f$ 是嵌入当且仅当它是左可消的且对所有 $x : A$ 有 $\Omega(B, f(x))$ 可缩。）给出例子说明 (i)和 (ii)都不蕴含另一个。

*Exercise* 4.8. 证明从 $\mathbf{2}$ 到 $B$ 的左可消函数类型（参见练习 4.7）等价于 $\sum_{(x,y:B)}(x \neq y)$。给出从 $\mathbf{2}$ 到 $B$ 的嵌入类型的类似显式刻画。

*Exercise* 4.9. **朴素非依赖函数外延性公理**说对于 $A, B : \mathcal{U}$ 和 $f, g : A \to B$，存在函数 $(\prod_{(x:A)} f(x) = g(x)) \to (f = g)$。修改第 4.9 节的论证来证明这个公理蕴含完整的函数外延性公理（2.9.3）。

# 第 5 章　归纳


5.1　正类型简介

5.2　W-类型的唯一性

5.3　自然数

5.4　归纳原则的等价

5.5　函子化与自然性

5.6　高阶归纳类型的一般化

5.7　广义归纳定义

5.8　余归纳

# 第 6 章　高阶归纳类型

# 第 7 章　同伦 n-类型

# 第 2 部分

# 数学

# 第 8 章　同伦论

# 第 9 章　范畴论

**9.1　范畴与预范畴**

**9.2　函子与变换**

**9.3　伴随**

**9.4　等价**

**9.5　Yoneda 引理**

**9.6　严格范畴**

**9.7　dagger 范畴**

**9.8　Rezk 完备化**

**9.9　范畴的结构同一性原则**

# 第 10 章　集合论

## 10.1　-W-预拓扑

## 10.2　集合的范畴

## 10.3　基数

## 10.4　序数

## 10.5　累积层次

# 第 11 章　实数

# 附录

# 附录 A　形式类型论

# 参考文献


[AW09] Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 146:45–55, 2009. 4

[BCH13] Bruno Barras, Thierry Coquand, and Simon Huber. A generalization of Takeuti-Gandy interpretation. https://ncatlab.org/ufias2012/files/semi.pdf, 2013. 13

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940. 2

[Chu41] Alonzo Church. *The Calculi of Lambda Conversion*. Princeton University Press, 1941. 2

[Coq92] Thierry Coquand. The paradox of trees in type theory. *BIT Numerical Mathematics*, 32(1):10–14, 1992. 29

[GAA+13] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Thery. A machine-checked proof of the odd order theorem. In *Interactive Theorem Proving*, 2013. 7

[HS98] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In Giovanni Sambin and Jan M. Smith,

editors, *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 83–111. Oxford University Press, New York, 1998. 4

[KLN04] Fairouz Kamareddine, Twan Laan, and Rob Nederpelt. *A Modern Perspective on Type Theory: From its Origins until Today*. Number 29 in Applied Logic. Kluwer, 2004. 2

[KLV12] Chris Kapulkin, Peter LeFanu Lumsdaine, and Vladimir Voevodsky. The simplicial model of univalent foundations, 2012. arXiv:1211.2851. 13

[Kol32] Andrey Kolmogorov. Zur Deutung der intuitionistischen Logik. *Mathematische Zeitschrift*, 35:58–65, 1932. 10

[Law05] F. William Lawvere. An elementary theory of the category of sets (long version) with commentary. *Reprints in Theory and Applications of Categories*, 11:1–35, 2005. Reprinted and expanded from Proc. Nat. Acad. Sci. U.S.A. **52** (1964), With comments by the author and Colin McLarty. 7

[LH12] Daniel R. Licata and Robert Harper. Canonicity for 2-dimensional type theory. In *Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 337–348, New York, NY, USA, 2012. ACM. 12, 13

[LS17] Peter LeFanu Lumsdaine and Michael Shulman. Semantics of higher inductive types. arXiv:1705.07088, 2017. 13

[Lur09] Jacob Lurie. *Higher topos theory*. Number 170 in Annals of Mathematics Studies. Princeton University Press, 2009. arXiv:math.CT/0608040. 11, 175

[ML75] Per Martin-Löf. An intuitionistic theory of types: predicative part. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium '73, Proceedings of the Logic Colloquium*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 73–118. North-Holland, 1975. 2

[ML82] Per Martin-Löf. Constructive mathematics and computer programming. In L. Jonathan Cohen, Jerzy Łoś, Helmut Pfeiffer, and Klaus-Peter Podewski, editors, *Logic, Methodology and Philosophy of Science VI, Proceedings of the Sixth International Congress of Logic, Methodology and Philosophy of Science, Hannover 1979*, volume 104 of *Studies in Logic and the Foundations of Mathematics*, pages 153–175. North-Holland, 1982. 2

[ML84] Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in Proof Theory*. Bibliopolis, 1984. Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980. 2

[ML98] Per Martin-Löf. An intuitionistic theory of types. In Giovanni Sambin and Jan M. Smith, editors, *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 127–172. Oxford University Press, 1998. 2

[Pie02] Benjamin C. Pierce. *Types and Programming Languages*. MIT Press, 2002. 2

[Rez05] Charles Rezk. Toposes and homotopy toposes. http://www.math.uiuc.edu/~rezk/homotopy-topos-sketch.pdf, 2005. 11, 175

[Rus08] Bertand Russell. Mathematical logic based on the theory of types. *American Journal of Mathematics*, 30:222–262, 1908. 2

[Som10] Giovanni Sommaruga. *History and Philosophy of Constructive Type Theory*. Number 290 in Synthese Library. Kluwer, 2010. 2

[TV02] Bertrand Toën and Gabriele Vezzosi. Homotopical algebraic geometry I: Topos theory, 2002. arXiv:math/0207028. 11

[TvD88a] Anne Sjerp Troelstra and Dirk van Dalen. *Constructivism in mathematics. Vol. I*, volume 121 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1988. An introduction. 10, 149

[TvD88b] Anne Sjerp Troelstra and Dirk van Dalen. *Constructivism in mathematics. Vol. II*, volume 123 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1988. An introduction. 10, 149

[Voe06] Vladimir Voevodsky. A very short note on the homotopy $\lambda$-calculus. http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/Hlambda_short_current.pdf, 2006. 4

[Voe12] Vladimir Voevodsky. A universe polymorphic type system. https://ncatlab.org/ufias2012/files/Universe+polymorphic+type+sytem.pdf, 2012. 13, 149

# 索引


∞-groupoid, 4
$(\infty, 1)$-topos, 15

abuse
  of notation, 6
algorithm, 6, 7, 9, 10
automorphism
  fixed-point-free, 11
axiom
  of choice, 11

canonicity, 13
computer proof assistant, *see* proof assistant
consistency, 13
constructivity, 13
"continuity" of functions in type theory, 3
CW complex, 6

discrete
  space, 7, 10, 14

Elementary Theory of the Category of Sets, 7
excluded middle, 11
extraction of algorithms, 9, 10

Feit–Thompson theorem, 7
foundations, 1
foundations, univalent, 1
four-color theorem, 9
function
  continuous
    in classical homotopy theory, 3

homotopy
  equivalence
    topological, 3
  topological, 3
  type, 4

identity, 5
inaccessible cardinal, 13
informal type theory, **8**–**9**
interval
  topological unit, 4

Kan complex, 4, 13

$\lambda$-calculus, 2
Lawvere, 7, 10
logic
  constructive, 11

constructive vs classical, 11
intuitionistic, 11

mathematics
classical, 9, 11
constructive, 9–13
formalized, 2, 7, **8**–**9**, 15
model category, 4

$n$-type, 10

odd-order theorem, 7
open
problem, 13–15

path
topological, 4
Postnikov tower, 10
presentation
of a space as a CW complex, 7
programming, 2, 12
proof
assistant, 2, 9
proposition
as types, 9

Quillen model category, 4

Russell, Bertrand, 2

set, 7–8
set theory
Zermelo–Fraenkel, 7
simplicial
sets, 4
small
type, 5

theorem
Feit–Thompson, 7
four-color, 9
odd-order, 7
topological
path, 4
space, 3, 4
topos, 12
type
higher inductive, 6, 7
small, 5
type theory, 2
formal, 8–9
informal, **8**–**9**

unit
interval, 4
univalence axiom, 1, 5, 11
constructivity of, 13

Zermelo-Fraenkel set theory, *see* set theory
ZF, *see* set theory
ZFC, *see* set theory