# Homotopy Type Theory

## Univalent Foundations of Mathematics

The Univalent Foundations Program
Institute for Advanced Study

# Homotopy Type Theory

Univalent Foundations of Mathematics

# Homotopy Type Theory

## Univalent Foundations of Mathematics

The Univalent Foundations Program
Institute for Advanced Study

"Homotopy Type Theory: Univalent Foundations of Mathematics"


Book version: first-edition-15-ge428abf

MSC 2010 classification: 03-02, 55-02, 03B15



This book is freely available at http://homotopytypetheory.org/book/.

## Acknowledgment

Apart from the generous support from the Institute for Advanced Study, some contributors to the book were partially or fully supported by the following agencies and grants:

- Association of Members of the Institute for Advanced Study: a grant to the Institute for Advanced Study
- Agencija za raziskovalno dejavnost Republike Slovenije: P1–0294, N1–0011.
- Air Force Office of Scientific Research: FA9550-11-1-0143, and FA9550-12-1-0370.

  This material is based in part upon work supported by the AFOSR under the above awards. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the AFOSR.

- Engineering and Physical Sciences Research Council: EP/G034109/1, EP/G03298X/1.
- European Union's 7th Framework Programme under grant agreement nr. 243847 (ForMath).
- National Science Foundation: DMS-1001191, DMS-1100938, CCF-1116703, and DMS-1128155.

  This material is based in part upon work supported by the National Science Foundation under the above awards. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

- The Simonyi Fund: a grant to the Institute for Advanced Study

# Homotopy Type Theory

## Univalent Foundations of Mathematics

The Univalent Foundations Program
Institute for Advanced Study

# Homotopy Type Theory

Univalent Foundations of Mathematics

# Homotopy Type Theory

## Univalent Foundations of Mathematics

The Univalent Foundations Program
Institute for Advanced Study

"Homotopy Type Theory: Univalent Foundations of Mathematics"


Book version: first-edition-15-ge428abf

MSC 2010 classification: 03-02, 55-02, 03B15



This book is freely available at http://homotopytypetheory.org/book/.

## Acknowledgment

Apart from the generous support from the Institute for Advanced Study, some contributors to the book were partially or fully supported by the following agencies and grants:

- Association of Members of the Institute for Advanced Study: a grant to the Institute for Advanced Study
- Agencija za raziskovalno dejavnost Republike Slovenije: P1–0294, N1–0011.
- Air Force Office of Scientific Research: FA9550-11-1-0143, and FA9550-12-1-0370.

  This material is based in part upon work supported by the AFOSR under the above awards. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the AFOSR.

- Engineering and Physical Sciences Research Council: EP/G034109/1, EP/G03298X/1.
- European Union's 7th Framework Programme under grant agreement nr. 243847 (ForMath).
- National Science Foundation: DMS-1001191, DMS-1100938, CCF-1116703, and DMS-1128155.

  This material is based in part upon work supported by the National Science Foundation under the above awards. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

- The Simonyi Fund: a grant to the Institute for Advanced Study

# Preface

## IAS Special Year on Univalent Foundations

A Special Year on Univalent Foundations of Mathematics was held in 2012-13 at the Institute for Advanced Study, School of Mathematics, organized by Steve Awodey, Thierry Coquand, and Vladimir Voevodsky. The following people were the official participants.

| | | |
|---|---|---|
| Peter Aczel | Eric Finster | Alvaro Pelayo |
| Benedikt Ahrens | Daniel Grayson | Andrew Polonsky |
| Thorsten Altenkirch | Hugo Herbelin | Michael Shulman |
| Steve Awodey | André Joyal | Matthieu Sozeau |
| Bruno Barras | Dan Licata | Bas Spitters |
| Andrej Bauer | Peter Lumsdaine | Benno van den Berg |
| Yves Bertot | Assia Mahboubi | Vladimir Voevodsky |
| Marc Bezem | Per Martin-Löf | Michael Warren |
| Thierry Coquand | Sergey Melikhov | Noam Zeilberger |

There were also the following students, whose participation was no less valuable.

| | | |
|---|---|---|
| Carlo Angiuli | Guillaume Brunerie | Egbert Rijke |
| Anthony Bordg | Chris Kapulkin | Kristina Sojakova |

In addition, there were the following short- and long-term visitors, including student visitors, whose contributions to the Special Year were also essential.

| | | |
|---|---|---|
| Jeremy Avigad | Richard Garner | Nuo Li |
| Cyril Cohen | Georges Gonthier | Zhaohui Luo |
| Robert Constable | Thomas Hales | Michael Nahas |
| Pierre-Louis Curien | Robert Harper | Erik Palmgren |
| Peter Dybjer | Martin Hofmann | Emily Riehl |
| Martín Escardó | Pieter Hofstra | Dana Scott |
| Kuen-Bang Hou | Joachim Kock | Philip Scott |
| Nicola Gambino | Nicolai Kraus | Sergei Soloviev |

## About this book

We did not set out to write a book. The present work has its origins in our collective attempts to develop a new style of "informal type theory" that can be read and understood by a human being,

as a complement to a formal proof that can be checked by a machine. Univalent foundations is closely tied to the idea of a foundation of mathematics that can be implemented in a computer proof assistant. Although such a formalization is not part of this book, much of the material presented here was actually done first in the fully formalized setting inside a proof assistant, and only later "unformalized" to arrive at the presentation you find before you — a remarkable inversion of the usual state of affairs in formalized mathematics.

Each of the above-named individuals contributed something to the Special Year — and so to this book — in the form of ideas, words, or deeds. The spirit of collaboration that prevailed throughout the year was truly extraordinary.

Special thanks are due to the Institute for Advanced Study, without which this book would obviously never have come to be. It proved to be an ideal setting for the creation of this new branch of mathematics: stimulating, congenial, and supportive. May some trace of this unique atmosphere linger in the pages of this book, and in the future development of this new field of study.

The Univalent Foundations Program
Institute for Advanced Study
Princeton, April 2013

# 目　　录

# 引　言　(Introduction)

同伦类型论 (Homotopy type theory, HoTT) 是一个结合了多个不同领域的数学新分支，出乎意料地将它们联系在一起。它基于最近发现的 同伦论 (homotopy theory) 和 类型论 (type theory) 之间的联系。同伦论是代数拓扑 (algebraic topology) 和同调代数 (homological algebra) 的延伸，与高阶范畴论 (higher category theory) 有关；而类型论是数学逻辑 (mathematical logic) 和理论计算机科学 (theoretical computer science) 的分支。尽管这两个领域之间的联系目前仍是密切研究的焦点，但越来越清楚的是，这只是一个刚刚起步的学科，需要更多的时间和努力来完全理解它。它涉及到看似遥远的话题，如球体的同伦群 (homotopy groups of spheres)、类型检查的算法 (algorithms for type checking) 以及弱 $\infty$-群 (weak $\infty$-groupoids) 的定义。

同伦类型论还为数学的基础引入了新的想法。一方面，有 Voevodsky 提出的精妙的 单值性公理 (univalence axiom)。单值性公理特别意味着同构的结构可以被视为相同的，这一原则在数学家们日常工作中得到了愉快的使用，尽管它与传统基础“官方”教义不兼容。另一方面，我们还有 高阶归纳类型 (higher inductive types)，它们为一些同伦论的基本空间和构造提供了直接的、逻辑的描述：如球体 (spheres)、圆柱 (cylinders)、截断 (truncations)、局部化 (localizations) 等。这两种思想在经典集合论基础中都无法直接捕捉，但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 (logic of homotopy types)”。

这提示了一种数学基础的新概念，具有内在的同伦内容，一种“不变量”的数学对象的概念——以及便捷的机器实现，可以作为工作数学家的实际帮助。这就是 单值基础 (Univalent Foundations) 计划。本书旨在首次系统地阐述单值基础的基础，并收集这一新型推理风格的例子——但不要求读者掌握或学习任何形式逻辑，或使用任何计算机证明助手 (computer proof assistant)。

我们强调，同伦类型论是一个年轻的领域，单值基础仍在发展中。本书应被视为这个领域的一部分在撰写时的“快照”，而不是一个完成的建筑的精致阐述。正如我们稍后会简要讨论的那样，同伦类型论的许多方面尚未完全理解——有些甚至在此未被触及。最终的理论几乎可以肯定不会完全像本书中描述的那样，但它肯定至少会同样强大且功能强大；因此，我们相信单值基础最终会成为集合论之外的一个可行替代方案，成为大多数数学家进行非正式数学推理的“隐含基础”。

## 类型论 (Type theory)

类型论最初是由 Bertrand Russell 发明的 [?]，作为阻止当时研究的数学逻辑基础中的悖论的一种工具。在接下来的几十年里，它被许多人进一步发展，特别是 Church [?, ?] 将其与他的 $\lambda$-演算 ($\lambda$-calculus) 结合起来。尽管它通常不被视为经典数学的基础，集合论 (set theory) 更为常见，但类型论仍然有许多应用，尤其是在计算机科学 (computer science) 和编程语言理论 (theory of programming languages) 中 [?]。Per Martin-Löf [?, ?, ?, ?] 等人开发了 Church 类型系统的“预期性的”修改，这现在通常称为依赖的 (dependent)、构造性的 (constructive)、直觉主义的 (intuitionistic) 或简单地 Martin-Löf 类型论 (Martin-Löf type theory)。这是我们在此考虑的系统的基础；它最初是为了作为构造性数学 (constructive mathematics) 形式化的严格框架而设计的。在下文中，我们将经常使用“类型论”来专门指代这个系统和类似的系统，尽管类型论作为一个学科要广泛得多（参见 [?, ?]，了解类型论的历史）。

在类型论中，不同于集合论，物体是通过一种原始的 类型 (type) 概念来分类的，类似于编程语言中使用的数据类型 (data-types)。这些精心构建的类型可用于表达被分类对象的详细规范，从而产生推理这些对象的原则。举一个非常简单的例子，乘积类型 $A \times B$ 的对象被认为是 $(a, b)$ 的形式，因此

我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 $A \to B$ 的对象可以通过一个由类型 $A$ 的对象参数化的类型 $B$ 的对象获得，并且可以在类型 $A$ 的参数处求值。这种所有对象的严格可预测行为（与集合论的更自由的形成原则相比，允许不均匀集合的形成）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， 这些助手用于形式化数学并验证形式化证明的正确性。我们将在下面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) 的概念在一些难以精确定义的方面有所不同。我们相信，新的将类型视为空间 (spaces) 的想法，而不是作为奇怪的集合（可能在构造时没有使用经典逻辑），是向前迈出的重要一步。特别是，它解决了理解类型的元素的相等性 (equality) 与集合的元素的相等性之间的区别的问题。

在同伦论中，人们关心的是空间 (spaces) 和它们之间的连续映射 (continuous mappings)，以及它们之间的同伦。两者之间的 *同伦* (homotopy) 是指一个连续映射 $H : X \times [0,1] \to Y$，满足 $H(x,0) = f(x)$ 和 $H(x,1) = g(x)$。同伦 $H$ 可以被视为 $f$ 变形到 $g$ 的“连续变形”。如果存在连续映射来回，且它们的复合在同伦意义下等价，即如果它们是“同伦等价 (homotopy equivalent)”的，则称空间 $X$ 和 $Y$ 是 *同伦等价* (homotopy equivalent) 的，即 $X \simeq Y$。同伦等价的空间具有相同的代数不变量（例如，同调 (homology) 或基本群 (fundamental group)），并且被认为具有相同的 *同伦类型* (homotopy type)。

## 同伦类型论 (Homotopy type theory)

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”或更高阶群 (higher groupoids)，而将逻辑构造（例如乘积 $A \times B$）视为这些空间上的同伦不变量构造。通过这种方式，我们能够直接操作空间，而不必先开发点集拓扑 (point-set topology) 或其任何组合替代物（例如单纯集合论 (theory of simplicial sets)）。为了简要解释这一观点，首先考虑类型论的基本概念，即 *项* (term) $a$ 属于 *类型* (type) $A$，写作：

$$a : A.$$

这一表达式传统上被认为类似于：

“$a$ 是集合 $A$ 的元素”。

然而，在同伦类型论中，我们将其视为：

“$a$ 是空间 $A$ 的一个点”。

类似地，类型论中的每个函数 $f : A \to B$ 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”是纯同伦地对待的，而不是拓扑地对待的。例如，类型没有“开子集 (open subset)”的概念，也没有“序列收敛 (convergence)”的概念。我们只有“同伦”概念，如点之间的路径 (paths) 和路径之间的同伦，这些概念在同伦论的其他模型中也是有意义的（如单纯集合 (simplicial sets)）。因此，更准确地说，我们将类型视为 *∞-群* (groupoids)；这是同伦论的“不变量对象”的名称，这些对象可以由拓扑空间、单纯集合或任何其他同伦论模型表示。然而，有时使用“空间 (space)”和“路径 (path)”等拓扑术语是方便的，只要我们记住其他拓扑概念不适用。

（我们也可以使用术语 *同伦类型* (homotopy type) 来描述这些对象，这暗示了“一个类型（如类型论中的）从同伦的角度看”以及“一个空间从同伦论的角度看”这两种解释。这与“同伦类型”作为空间在同伦等价下的同伦类型的等价类 (equivalence class) 的经典含义有些不同，尽管它保留了诸如“这两个空间具有相同的同伦类型 (homotopy type)”之类的短语的意义。）

将类型解释为结构化对象 (structured objects) 而不是集合 (sets) 的想法有着悠久的历史，并且有助于澄清类型论中的各种神秘方面。例如，将类型解释为层 (sheaves) 有助于解释类型论逻辑的直觉性质，而将它们解释为部分等价关系 (partial equivalence relations) 或“域 (domains)”则有助于解释它的计算方面。这也意味着我们可以使用类型论推理来研究结构化对象，从而形成丰富的范畴逻辑 (categorical logic) 领域。同伦解释符合这一模式：它澄清了类型论中 *身份* (identity)（或相等性 (equality)）的性质，并允许我们在研究同伦论时使用类型论推理。

同伦解释的关键新思想是，两个相同类型 $A$ 的对象 $a, b : A$ 的逻辑身份 $a = b$ 可以理解为在空间 $A$ 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，它们可以被视为相同的，因为同伦只是 $B$ 中的路径的（连续的）家族 $p_x : f(x) \rightsquigarrow g(x)$，每个 $x : A$ 对应一个。在类型论中，对于每个类型 $A$，都有一个（以前略显神秘的）类型 $\mathsf{Id}_A$，用于识别 $A$ 的两个对象；在同伦类型论中，这只是所有连续映射 $I \to A$ 的 *路径空间* (path space) $A^I$。在这种方式下，术语 $p : \mathsf{Id}_A(a, b)$ 代表在 $A$ 中从 $a$ 到 $b$ 的路径 $p : a \rightsquigarrow b$。

同伦类型论的想法大约在 2006 年由 Awodey 和 Warren [?] 以及 Voevodsky [?] 独立提出，但受到了 Hofmann 和 Streicher 早期群解释 [?] 的启发。事实上，现在已知高维范畴论（尤其是弱 $\infty$-群 (weak $\infty$-groupoids) 论）与同伦论密切相关，正如 Grothendieck 所提出的，并且现在正被这两种类型的数学家密切研究。Awodey–Warren 和 Voevodsky 的原始语义模型使用了来自同伦论的公认概念和技术，现在也在高阶范畴论中使用，例如 Quillen 模型范畴和 Kan 单纯集合。

特别是，Voevodsky 构建了一个在 Kan 单纯集合中解释类型论的模型，并认识到这一解释满足了一个被他称为 *单值性* (univalence) 的进一步关键性质。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 (principle of extensionality for propositions) 证明是它的一个非常特殊的情况，Hofmann 和 Streicher 曾考虑过另一个以“宇宙外延性 (universe extensionality)”命名的特例）。将单值性作为新公理添加到类型论中具有深远的影响，许多影响是自然的、简化的和令人信服的。单值性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则不成立。

## **单值基础** (Univalent foundations)

简而言之，单值性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，其元素本身就是类型；这种类型称为 *宇宙* (universe)，通常表示为 $\mathcal{U}$。这些作为 $\mathcal{U}$ 的项的类型通常称为 *小类型* (small types)。像任何类型一样，$\mathcal{U}$ 有一个身份类型 $\mathsf{Id}_{\mathcal{U}}$，它表示小类型之间的等价关系 $A = B$。考虑类型为空间时，$\mathcal{U}$ 是一个空间，其中的点是空间；要理解它的身份类型，我们必须问，在 $\mathcal{U}$ 中，空间之间的路径 $p : A \rightsquigarrow B$ 是什么？单值性公理表示这种路径对应于同伦等价 $A \simeq B$，大致如上所述。更确切地说，给定任意（小）类型 $A$ 和 $B$，除了原始类型 $\mathsf{Id}_{\mathcal{U}}(A, B)$ 表示 $A$ 与 $B$ 的等价外，还有定义类型 $\mathsf{Equiv}(A, B)$ 表示 $A$ 到 $B$ 的等价。由于在任何对象上的恒等映射都是等价的，因此存在一个规范映射，

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

单值性公理声明该映射本身就是一个等价。简化地说，我们可以将其简洁地陈述如下：

**单值性公理**: $(A = B) \simeq (A \simeq B)$。

换句话说，身份等同于等价性。特别是，可以说“等价类型是相同的”。然而，这句话有些误导，因为它可能听起来像是一种“骨架性”条件，将等价性“折叠”为与身份重合，而实际上单值性是关于扩展身份的概念以与（未更改的）等价概念重合。

从同伦的角度来看，单值性意味着具有相同同伦类型的空间在宇宙 $\mathcal{U}$ 中由一条路径连接，符合小空间分类空间的直觉。然而，从逻辑的角度来看，这是一个全新的想法：它表示同构的事物可以被识别！数学家当然习惯于在实践中识别同构结构，但他们通常通过“滥用符号 (abuse of notation)”或其他一些非正式的手段这样做，知道所涉及的对象并不“真正”相同。但是，在这种新的基础框架中，这种结构可以在逻辑意义上正式识别，即每个涉及一个对象的性质或构造也适用于另一个对象。确实，这种识别现在是明确的，并且可以系统地沿着它传递性质和构造。此外，进行这种识别的不同方式本身也形成了一个结构，我们可以（并且应该！）考虑在内。

因此，总结一下，对于宇宙 $\mathcal{U}$ 的点 $A$ 和 $B$（即小类型），单值性公理将以下三种概念等同起来：

- （逻辑）$A$ 和 $B$ 的识别 $p : A = B$
- （拓扑）从 $A$ 到 $B$ 的路径 $p : A \rightsquigarrow B$ 在 $\mathcal{U}$ 中
- （同伦）$A$ 和 $B$ 之间的等价 $p : A \simeq B$。

### 高阶归纳类型 (Higher inductive types)

类型论的一个传统优势在于其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。从这一声明中可以算法性地提取数学归纳法 (principle of mathematical induction) 原则，该原则表征了自然数。更广泛的归纳定义涵盖了各种列表和良基树 (well-founded trees)，每个树都由相应的"归纳原则 (induction principle)"表征。这包括某些编程语言中使用的大多数数据结构；因此，类型论在形式化推理中的有用性得以体现。如果从非常广泛的意义上来理解，归纳定义还包括例如不交并 $A + B$ 之类的例子，这些例子可以被视为由两个注入 $A \to A + B$ 和 $B \to A + B$"归纳"生成的。这种情况下的"归纳原则"是"通过案例分析证明 (proof by case analysis)"，它表征了不交并。

在同伦论中，自然也会考虑到"归纳定义的空间"，这些空间不仅由一组 *点* (points) 生成，还由 *路径* (paths) 和更高阶路径的集合生成。经典上，这些空间被称为 CW *复形* (CW complexes)。例如，圆 $S^1$ 由一个点和从该点到自身的一条路径生成。类似地，2-球面 $S^2$ 由一个点 $b$ 和一条从 $b$ 的恒等路径到自身的二维路径生成，而环面 $T^2$ 由一个点、两条从该点到自身的路径 $p$ 和 $q$ 以及一条从 $p \cdot q$ 到 $q \cdot p$ 的二维路径生成。

通过在同伦类型论中将路径识别为身份，以上这些"归纳定义的空间"可以通过类型论中的"归纳原则"来表征，完全类似于自然数和不交并等经典例子。由此得出的 *高阶归纳类型* (higher inductive types) 提供了一种直接的"逻辑"方法来推理熟悉的空间，如球体，这些空间（与单值性相结合）可以用于以纯形式化的方式进行同伦论中的经典论证，如计算球体的同伦群。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，产生了对这两个学科的新见解。

此外，这只是冰山一角：许多同伦论中的抽象构造，如同伦余极限 (homotopy colimits)、悬挂 (suspensions)、Postnikov 塔 (Postnikov towers)、局部化 (localization)、完备 (completion) 和谱化 (spectrification) 等，也可以表示为高阶归纳类型。其中许多经典上是使用 Quillen 的"小物件论证 (small object argument)"构造的，这可以看作是一种有限的方式，算法性地描述了空间的无限 CW 复形表示，就像"零和后继"是自然数无限集合的有限算法描述一样。通过小物件论证产生的空间因其复杂性和难以理解而闻名；类型论方法可能更简单，绕过了任何显式构造，直接访问了适当的"归纳原则"。因此，单值性和高阶归纳类型的结合提示了在同伦论实践中某种革命的可能性。

### 单值基础中的集合 (Sets in univalent foundations)

我们声称单值基础最终可以作为"所有"数学的基础，但到目前为止，我们只讨论了同伦论。当然，类型论在没有新同伦类型论特性的情况下形式化数学的具体例子有很多， 例如最近在 Coq 中形式化的 Feit–Thompson 奇数阶定理 [?]。

但传统观点认为数学建立在集合论的基础上，即所有数学对象和构造都可以编码到 Zermelo–Fraenkel 集合论 (Zermelo–Fraenkel set theory, ZF) 这样的理论中。 然而，现在已经很明确，对于大多数集合论之外的数学，ZF 中集合的复杂层次结构实际上是不必要的：一种更"结构性"的理论，如 Lawvere 的集合范畴的基本理论 (Elementary Theory of the Category of Sets) 就足够了。

在单值基础中，基本对象是"同伦类型 (homotopy types)"而不是集合，但我们可以定义一个类的类型，其行为类似于集合。同伦地，这些可以被视为每个连通分量都是可收缩的空间，即那些同伦等价于离散空间的空间。一个定理表明，这类"集合"的范畴满足 Lawvere 的公理（或相关的公理，取决于理论的细节）。因此，任何可以在类似 ETCS 的理论中表示的数学（经验表明，实际上是所有数学）都可以同样地在单值基础中表示。

这支持了单值基础至少与现有的数学基础一样好的说法。在单值基础中工作的数学家可以以熟悉的方式用集合构建结构，更一般的同伦类型在基础背景中等待，直到需要它们为止。因此，本书中大多数应用选择了那些单值基础有某种 *新贡献*的领域，区别于现有的基础系统。

不出所料，同伦论和范畴论是其中的两个，但可能不太明显的是，单值基础甚至在集合论和实分析等学科中也提供了新的、有趣的东西。例如，单值性公理允许我们识别同构结构，而高阶归纳类型允许通过其普遍性质直接描述对象。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，甚至在 ZF 集合论中研究的对象也可以在单值基础中的集合内通过这样的归纳普遍性质来表征。

## 非正式类型论 (Informal type theory)

经典数学家在学习类型论时经常遇到的一个困难是它通常作为一个完全或部分形式化的演绎系统呈现。这种风格对于证明理论的研究非常有用，但对于实际应用中的非正式推理来说并不特别方便。甚至对于大多数对数学基础感兴趣的工作数学家来说，这种风格也并不熟悉。本书的一个目标是发展一种在单值基础中进行数学研究的非正式风格，这种风格既严格又精确，但更接近于日常数学的语言和风格。

在现今的数学中，人们通常以一种可以推测上形式化为 ZFC 这样的初等集合论系统的方式来构建和推理数学对象，至少在足够的创造力和耐心下是可以做到的。大多数情况下，人们甚至不需要意识到这一可能性，因为它与被认为"完全严格"的条件（即所有数学家通过教育和经验直觉上理解的条件）基本一致。但人们确实需要学会对"非正式集合论"的一些方面保持谨慎：如使用太大或不连贯的集合；选择公理及其等价物；甚至（对于本科生来说）反证法等方法。采用一种新的基础系统，如同伦类型论作为非正式推理的 *隐式形式基础*，将需要调整一些本能和实践。本书旨在作为这种"新数学"的一个例子，尽管它仍然是非正式的，但现在在原则上可以形式化为同伦类型论，而不是 ZFC，只要有足够的创造力和耐心。

值得强调的是，在这个新系统中，这种形式化可以带来实际的好处。类型论的形式系统适用于计算机系统，并已在现有的证明助手中实现。证明助手 (proof assistant) 是一种计算机程序，它引导用户构建完全形式化的证明，只允许有效的推理步骤。它还提供了一定程度的自动化，可以在库中搜索现有定理，甚至可以从结果（构造性）证明中提取算法。我们相信，这一单值基础计划的方面将其与其他基础方法区分开来，有可能为工作数学家提供一种新的实用性。事实上，基于旧类型论的证明助手已经用于形式化大量数学证明，如四色定理 (four-color theorem) 和费特-汤普森定理 (Feit–Thompson theorem)。单值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，即使是当前可用的实现（它们主要是对现有证明助手，如 Coq 和 Agda 的小修改）已经展示了它们的价值，不仅在形式化已知的证明中，还在发现新证明中。事实上，本书中描述的许多证明实际上是先以完全形式化的形式在证明助手中完成的，现在才第一次被"非形式化"——这与形式数学和非正式数学之间的通常关系相反。

可以想象，不久的将来，数学家将在单值基础系统中验证自己论文的正确性，形式化在证明助手中，并且这一过程将变得像在 TEX 中排版自己的论文一样自然。（这是否会成为出版商的梦想或噩梦还有待观察。）原则上，这对任何其他基础系统同样适用，但我们相信使用单值基础在实践上更容易实现，这一点在本书及其形式化对应物中得到了见证。

## 构造性 (Constructivity)

经典 基础与类型论之间最显著的差异之一是 *证明相关性* (proof relevance) 的思想，根据这一思想，数学陈述，甚至它们的证明，都成为一流的数学对象。在类型论中，我们通过类型表示数学陈述，这些类型可以同时被视为数学构造和数学断言，这一概念也被称为 *命题即类型* (propositions as types)。因此，我们可以将术语 $a : A$ 同时视为类型 $A$ 的一个元素（或在同伦类型论中，空间 $A$ 的一个点），也可以视为命题 $A$ 的证明。举个例子，假设我们有集合 $A$ 和 $B$（离散空间），考虑"$A$ 与 $B$ 同构"的陈述。在类型论中，这可以表达为：

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

这里将类型构造符 $\Sigma, \Pi, \times$ 分别读取为"存在 (there exists)"、"对于所有 (for all)"和"并且 (and)"会得出"$A$ 和 $B$ 同构"的通常表述；另一方面，将它们读取为和与积 (sums and products) 会得出 $A$ 和 $B$ 之间 *所有同构的类型* (type of all isomorphisms)！要证明 $A$ 和 $B$ 是同构的，只需构造一个证明 $p : \mathsf{Iso}(A,B)$，这与构造 $A$ 和 $B$ 之间的同构是相同的，即给出一对函数 $f, g$ 以及它们复合分别为恒等映射的 *证明*。这些证明本身无非是适当种类的同伦。通过这种方式，*证明一个命题等同于构造某个特定类型的元素*。特别是，证明"$A$ 和 $B$"的陈述就等同于证明 $A$ 并且证明 $B$，即给出类型 $A \times B$ 的一个元素。而证明 $A$ 推导 $B$ 就等同于找到 $A \to B$ 的一个元素，即从 $A$ 到 $B$ 的一个函数（确定一个从 $A$ 的证明到 $B$ 的证明的映射）。

命题即类型的逻辑是灵活的，并支持许多变体，例如使用仅一类类型来表示命题。在同伦类型论中，有一些自然的这种子类，产生于整个类型系统（就像经典同伦论中的空间）根据其更高同伦结构存在或崩溃的维度“分层 (stratified)”这一事实。特别是，Voevodsky 发现了一种纯类型论的 同伦 $n$-类型 (homotopy $n$-types) 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，通过高阶归纳类型，我们可以普遍地“截断 (truncate)”一个类型为 $n$-类型；在经典同伦论中，这将是其 $n$ 次 Postnikov 截断。对于逻辑而言，尤其重要的是同伦 $(-1)$-类型，我们称之为 纯粹命题 (mere propositions)。经典上，每个 $(-1)$-类型要么为空，要么是可收缩的；我们将这些可能性解释为真值 (truth values)“假”和“真”。

使用所有类型作为命题会产生一种非常“构造性 (constructive)”的逻辑；更多关于这方面的内容，参见 [?, ?, ?]。例如，每一个关于某物存在的证明都携带了足够的信息来实际找到该对象；每一个关于“$A$ 或 $B$”成立的证明要么是 $A$ 成立的证明，要么是 $B$ 成立的证明。因此，我们可以自动从每个证明中提取出一个算法； 这在计算机编程的应用中非常有用。

另一方面，这种逻辑确实偏离了数学中存在性证明的传统理解。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。对于这些原则，我们需要使用“$(-1)$ 截断”的逻辑，其中只有同伦 $(-1)$-类型表示命题。

更具体地说，考虑 选择公理：“如果对于每个 $x : A$，存在一个 $y : B$ 使得 $R(x, y)$，那么存在一个函数 $f : A \to B$ 使得对于所有 $x : A$，我们有 $R(x, f(x))$。”纯命题即类型的“存在”概念足够强大，可以使这一陈述简单地成立——然而，它并没有选择公理的所有后果。然而，在 $(-1)$-截断的逻辑中，这一陈述并非自动成立，而是一个强假设，具有与其经典集合论对应物类似的后果。

另一方面，考虑 排中律：“对于所有 $A$，要么 $A$，要么 $A$ 不成立。”在纯命题即类型的逻辑中解释这一点会得出与单值性公理不一致的陈述。因为证明“$A$”意味着展示它的一个元素，这一假设将给出一种统一的方法来从每个非空类型中选择一个元素——类似于 Hilbert 的选择算子。单值性意味着通过这种选择算子选择的 $A$ 的元素必须在所有自等价 (self-equivalences) 下是不变的，因为这些等价被视为自身份标识 (self-identities)，并且每个操作必须尊重身份；但是显然某些类型有没有固定点的自同构，例如我们可以交换两元素类型的元素。然而，“$(-1)$-截断排中律”虽然也不是自动成立的，但可以一致地假设其具有与经典数学中类似的后果。

换句话说，虽然纯命题即类型的逻辑在上述强算法意义上是“构造性”的，但默认的 $(-1)$-截断逻辑在不同意义上是“构造性”的（即由 Heyting 正式化为“直觉主义”的逻辑）；在后者中，我们可以自由地添加选择公理和排中律，从而获得可以称为“经典”的逻辑。因此，同伦类型论与构造性和经典逻辑兼容，还有许多其他逻辑。事实上，同伦视角揭示了经典逻辑和构造逻辑可以共存，作为不同系统的端点，在它们之间有无数可能性（$-1 < n < \infty$ 的同伦 $n$-类型）。我们可以谈论“$\mathsf{LEM}_n$ ”和“$\mathsf{AC}_n$ ”，其中 $\mathsf{AC}_\infty$ 是可证明的，而 $\mathsf{LEM}_\infty$ 与单值性不一致，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，适当假设下文中没有给出的下标 $(-1)$）。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，而一般类型仍然是“构造性的”。

值得强调的是，单值基础不 要求使用构造性或直觉主义逻辑。 大多数依赖于排中律和选择公理的经典数学都可以在单值基础中进行，只需假设这两个原则成立（在其正确的、$(-1)$-截断形式中）。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，当使用更少的假设证明定理时，它更强大，因为它适用于更多的例子。对于选择公理和排中律，情况也不例外：类型论承认许多有趣的“非标准”模型，如在层拓扑 (sheaf toposes) 中，经典性原则如选择公理和排中律往往会失败。同伦类型论在更高的拓扑 (higher toposes) 中也有类似的模型，如 [?, ?, ?] 中所研究的那样。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此类模型中内部有效。

其次，类型论的另一个优点是其可计算性。除了作为数学的基础，类型论也是一种形式的计算理论，可以被视为一种强大的编程语言。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：它们之间必须有一种和谐，允许所有证明作为程序“执行”。我们尚未完全理解同伦类型论引入的新原则，如单值性和高阶归纳类型，从这个角度来看，但基本轮廓正在显现；例如，参见 [?]。然而，长期以来已知原则如选择公理和排中律从根本上与可计算性对立，因为它们简单地断言某些东西存在而不给出任何计算方法。因此，避免它们是保持类型论作为计算理论特征所必需的。

幸运的是，构造性推理并不像看起来那么难。在某些情况下，只需重新表述一些定义，就可以使一个定理变得构造性，并使其证明更为优雅。此外，在单值基础中，这种情况似乎更常发生。例如：

(i) 在集合论基础中，在同伦论和范畴论中的某些点上，需要选择公理来执行超限构造。但是，通过高阶归纳类型，我们可以直接而构造性地编码这些构造。特别是，在 Chapter 8 中没有"综合"同伦论需要排中律或选择公理。

(ii) 在集合论基础中，陈述"每个全忠且本质上满的函子都是范畴的等价"是等价于选择公理的。但是，有了单值性公理，它就是真实的；参见 Chapter 9。

(iii) 在集合论中，为了获得"基数 (cardinal number)"和"序数 (ordinal number)"的概念，通常需要选择公理或基础公理 (axiom of foundation) 来得到表示集合和良序集合同构类的典型代表。但是，通过单值性和高阶归纳类型，我们可以通过截断宇宙直接获得这些代表；参见 Chapter 10。

(iv) 在集合论基础中，定义实数为 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。但是，通过高阶归纳类型，我们可以给出一种避免任何选择原则的良好表现版本；参见 Chapter 11。

当然，这些简化也可以被视为新方法最终不真正构造性的证据。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。关键是，在上述所有例子中，我们给出的理论版本具有独立的优点，无论是否假设排中律和选择公理可用。构造性，如果实现，将是一个额外的好处。

在讨论添加新原则，如单值性、高阶归纳类型、选择公理和排中律后，人们可能会想知道，结果系统是否仍然一致。（相对于集合论，类型论的原始优点之一是可以通过证明论手段看到它的一致性）。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 复形中都有一个模型，由 Voevodsky [?] 提出（参见 [?] 了解高阶归纳类型）。因此，它们已知相对于 ZFC（具有我们所需嵌套单值宇宙的不可达基数）是一致的。给出这种一致性更传统的类型论描述仍在进行中（例如，参见 [?, ?]）。

我们在 Table 1 中总结了类型论操作的不同观点。

| 类型 (Types) | 逻辑 (Logic) | 集合 (Sets) | 同伦 (Homotopy) |
|---|---|---|---|
| $A$ | 命题 (proposition) | 集合 (set) | 空间 (space) |
| $a : A$ | 证明 (proof) | 元素 (element) | 点 (point) |
| $B(x)$ | 谓词 (predicate) | 集合族 (family of sets) | 纤维 (fibration) |
| $b(x) : B(x)$ | 条件证明 (conditional proof) | 元素族 (family of elements) | 剖面 (section) |
| $\mathbf{0}, \mathbf{1}$ | $\bot, \top$ | $\emptyset, \{\emptyset\}$ | $\emptyset, *$ |
| $A + B$ | $A \vee B$ | 不交并 (disjoint union) | 上积 (coproduct) |
| $A \times B$ | $A \wedge B$ | 成对集合 (set of pairs) | 积空间 (product space) |
| $A \to B$ | $A \Rightarrow B$ | 函数集 (set of functions) | 函数空间 (function space) |
| $\sum_{(x:A)} B(x)$ | $\exists_{x:A} B(x)$ | 不交和 (disjoint sum) | 全空间 (total space) |
| $\prod_{(x:A)} B(x)$ | $\forall_{x:A} B(x)$ | 积 (product) | 剖面空间 (space of sections) |
| $\mathsf{Id}_A$ | 相等 (equality) $=$ | $\{ (x, x) \mid x \in A \}$ | 路径空间 (path space) $A^I$ |

表 1: 类型论操作的不同观点比较

## 开放问题 (Open problems)

对于那些有兴趣为这一数学新分支做出贡献的人来说，可能令人鼓舞的是，仍有许多有趣的开放问题。

最紧迫的问题之一可能是 Voevodsky 在 [?] 中提出的单值性公理的"构造性"。类型论的基本系统遵循 Gentzen 的自然演绎结构。逻辑连接符通过其引入规则定义，并通过计算规则证明其消除规则。遵循这一模式，使用 Tait 的可计算性方法（最初旨在分析 Gödel 的辩证法解释），可以证明类型论的 归一性 (normalization) 性质。这反过来又暗示了重要的性质，如类型检查的可判定性（这对于类型检查对应于证明检查是至关重要的属性，可以认为我们应该能够"认出证明"），以及所谓的"典范性

(canonicity) 性质”，即自然数类型的任何闭合项都归约为一个数码。这最后一个性质，以及引入/消除规则的统一结构，当使用公理扩展类型论时会丢失，例如函数外延性公理或单值性公理。

Voevodsky 提出了与使用单值性公理扩展的类型论的典范性问题相关的精确数学猜想：给定自然数类型的闭合项，是否总能找到一个数码和一个证明，证明该项等于该数码，其中这个等式证明可能本身使用了单值性公理？更一般地说，一个重要的问题是是否可以提供单值性公理的构造性证明。当添加其他同伦动机的构造时，如高阶归纳类型，情况又会如何？这些问题目前仍未解决，尽管目前正在开发方法以尝试找到答案。

另一个基本问题是处理本质上是集合的类型（如自然数）时的难题，这些类型是离散空间，只包含平凡路径。目前，同伦类型论实际上只能表征同伦等价的空间，这意味着这些“离散空间”可能仅同伦等价于离散空间。从类型论的角度来看，这意味着有许多路径与反身性相等，但不是“判断性地(judgmentally)”相等（参见 §1.1，了解“判断性”的含义）。虽然这种同伦不变性具有优势，但这些“无意义”的身份项确实在论证和构造中引入了不必要的复杂性，因此方便地系统地消除或折叠它们是很有意义的。

一个更专业但同样重要的问题是同伦类型论与 *高阶拓扑* (higher toposes) 研究之间的关系，该研究目前在高阶范畴论和同伦论的交叉点上进行。那些熟悉这两个主题的人越来越相信它们是密切相关的。例如，单值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“基本”反映。更一般地说，同伦类型论应该是 $(\infty,1)$-拓扑的“内部语言”，就像直觉主义高阶逻辑是普通 1-拓扑的内部语言一样。尽管有这种普遍共识，但细节仍需解决——特别是关于一致性和严格性的问题——解决这些问题无疑将进一步加深对两者的理解。

但目前最大的工作领域是正在进行的在这一新系统中形式化日常数学的工作。最近在形式化一些基础同伦论和范畴论的事实方面取得了令人鼓舞的成功；其中一些在 Chapters 8 and 9 中进行了描述。显然，还有很多工作需要完成。

同伦类型论社区维护了一个网站和一个博客，网址是 http://homotopytypetheory.org，并且还有一个讨论邮件列表。随时欢迎新来者加入！

## 如何阅读本书 (How to read this book)

本书分为两部分。Part I, “基础 (Foundations)”，发展了同伦类型论的基本概念。这是构建特定学科发展的数学基础，也是理解单值基础方法所必需的。对于程序员来说，这是“库代码 (library code)”。由于单值基础是一种新的、不同类型的数学，其基本概念需要一些时间来适应，因此 Part I 相当广泛。

Part II, “数学 (Mathematics)”，由四章组成，这四章在 Part I 的基本概念基础上，展示了单值基础在数学四个不同领域中的一些新成果：同伦论 (Chapter 8)、范畴论 (Chapter 9)、集合论 (Chapter 10) 和实分析 (Chapter 11)。Part II 中的各章相对独立，尽管有时会使用其他章中的引理。

想要认真理解单值基础并能够在其中工作，最终需要阅读并理解 Part I 的大部分内容。然而，想要仅了解单值基础及其作用的读者可能会理解，在阅读 Part II 中的“精华”内容之前，可能会觉得必须通读超过 200 页的内容令人却步。幸运的是，要阅读 Part II 中的章节并不需要 Part I 的所有内容。Part II 中的每一章都以其主题的简要概述开头，介绍单值基础对其的贡献，以及 Part I 的必要背景，因此有勇气的读者可以立即转到他们最喜欢的主题的相应章节。对于那些想要更深入理解 Part II 中一章或多章，但尚未准备好阅读 Part I 全部内容的读者，我们在此提供了 Part I 的简要总结，并对 Part II 中各章所需的部分进行了说明。

Chapter 1 讨论了类型论的基本概念，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将需要阅读 Chapter 1，因为类型论与集合论等其他基础之间存在许多微妙的差异。

Chapter 2 介绍了类型论的同伦视角，以及支持这一视角的基本概念，并描述了 Chapter 1 中类型论各组成部分的同伦行为。它还介绍了 *单值性公理* (univalence axiom)（??）——同伦类型论的两大基本创新之一。因此，它非常基础，我们鼓励每个人阅读，尤其是 §§2.1–2.4。

Chapter 3 介绍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性和直觉主义逻辑的关系。在这里，我们定义了排中律、选择公理和命题重缩放 (propositional resizing) 公理（尽管在本书的其余部分中，我们大多不需要假设其中任何一个），以及在表示传统逻辑时必不可少的 *命题截断* (propositional truncation)。本章是 Chapters 10 and 11 的基本背景，对于 Chapter 9 来说不那么

重要，对于 Chapter 8 来说则不是很必要。

Chapters 4 and 5 详细研究了两个特别主题：等价（和相关概念）以及广义归纳定义。虽然这些都是重要的主题，并且提供了对同伦类型论更深刻的理解，但大多数情况下，它们对 Part II 并不必要。Chapter 4 中的少数引理在此处和那里有使用，而 §§5.1, 5.6 and 5.7 中的一般讨论有助于为理解 Chapter 6 提供直觉。§5.7 中讨论的广义归纳定义在 Chapters 10 and 11 的某些地方也有使用。

Chapter 6 介绍了同伦类型论的第二个基本创新——**高阶归纳类型** (higher inductive types)——并给出了许多例子。高阶归纳类型是 Chapter 8 中的主要研究对象，其中一些在 Chapters 10 and 11 中也扮演了重要角色。它们对 Chapter 9 来说不那么重要，尽管在 §9.9 中使用了一个例子。

最后，Chapter 7 讨论了同伦 $n$-类型及其相关概念，如 $n$-连通类型。这些概念对于 Chapter 8 来说非常重要，但在 Part II 的其他部分中不那么重要，尽管在 §10.1 中使用了一些引理的 $n = -1$ 的情况。这完成了 Part I。如前所述，Part II 由四个基本无关的章节组成，每一章描述了单值基础在特定学科中的贡献。

在 Part II 的各章中，Chapter 8（同伦论）可能是最具革命性的。单值基础对同伦论有一种非常不同的“综合 (synthetic)”方法，在这种方法中，同伦类型是基本对象（即类型），而不是使用拓扑空间或其他集合论模型构造的。这使得经典代数拓扑定理的新证明风格成为可能，我们在此展示了一些示例，从 $\pi_1(\mathbb{S}^1) = \mathbb{Z}$ 到 Freudenthal 悬挂定理 (Freudenthal suspension theorem)。

在 Chapter 9（范畴论）中，我们发展了一些基本的 (1-) 范畴论，遵循单值性公理的原则，即 **相等即同构** (equality is isomorphism)。这具有令人愉快的效果，即确保所有定义和构造在范畴等价下自动不变：实际上，等价的范畴与等价的类型一样相等。（它还与高阶范畴论和高阶拓扑论有关。）

Chapter 10（集合论）研究了单值基础中的集合。集合的范畴具有其通常的性质，因此为不需要同伦或高阶范畴结构的任何数学提供基础。我们还观察到单值性使基数和序数更加愉快，高阶归纳类型产生了满足 Zermelo–Fraenkel 集合论通常公理的累积层次结构。

在 Chapter 11（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定义 Cauchy 实数，从而避免了构造数学中的一些相关问题。然后我们简要介绍了类似的方法来处理 Conway 的超实数 (surreal numbers)。

本书的每一章都以备注部分结尾，收集历史评论、文献参考和结果归属（尽可能）。我们还在每章末尾包含了练习，以帮助读者熟悉在单值基础中进行数学的过程。

最后，回想一下，本书是由大量人员协作完成的。我们尽最大努力实现术语和符号的一致性，并将数学按逻辑顺序排列，但很可能仍然存在一些不完美之处。我们请求读者对任何此类不幸表示宽恕，并欢迎为下一版的改进提供建议。

# Part I

# Foundations

Chapter 1

# 类 型 论 (Type theory)

## 1.1 类型论与集合论 (Type theory versus set theory)

同伦类型论 (Homotopy type theory) 是数学的基础语言之一，即作为 Zermelo–Fraenkel 集合论的替代品。然而，它在几个重要方面与集合论的行为有所不同，这可能需要一些时间来适应。在这里详细解释这些差异要求我们比在本书的其他部分更正式。如在引言中所述，我们的目标是非正式地书写类型论；但是对于习惯于集合论的数学家来说，起初更加精确可以帮助避免一些常见的误解和错误。

我们注意到，基于集合论的基础有两个"层次"：第一个层次是一阶逻辑的演绎系统，第二个层次是在该系统内部制定的特定理论的公理，比如 ZFC。因此，集合论不仅仅是关于集合，而是关于集合（第二层次的对象）和命题（第一层次的对象）之间的相互作用。

相比之下，类型论是它自己的演绎系统：它不需要在任何超结构如一阶逻辑中制定。相反，类型论只有一个基本概念：*类型* (types)。命题（我们可以证明、反驳、假设、否定等的陈述[1]）被识别为特定的类型，通过 Table 1 中所示的对应关系。因此，*证明一个定理*的数学活动被识别为 *构造一个对象* 的特殊情况——在这种情况下，构造的对象是代表一个命题的类型的居民。

这引出了类型论与集合论之间的另一个差异，但要解释它，我们必须稍微讨论一下演绎系统的基本情况。非正式地说，演绎系统是一组 **规则** (rules)，用于推导所谓的 **判断** (judgments)。如果我们将演绎系统视为一个正式游戏，那么判断就是我们按照游戏规则达到的"位置"。我们还可以将演绎系统视为一种代数理论，在这种情况下，判断是元素（如群的元素），而演绎规则是操作（如群乘法）。从逻辑的角度来看，判断可以被认为是"外部"的陈述，存在于元理论中，而与理论本身的"内部"陈述相对。

在一阶逻辑的演绎系统（集合论基于此）中，只有一种判断：给定命题有证明。也就是说，每个命题 $A$ 引发一个判断"$A$ 有证明"，所有判断都是这种形式。一阶逻辑中的规则，例如"从 $A$ 和 $B$ 推导出 $A \wedge B$"，实际上是一种"证明构造"规则，它表示在给定"$A$ 有证明"和"$B$ 有证明"的判断后，我们可以推断出"$A \wedge B$ 有证明"。注意，判断"$A$ 有证明"存在于与 *命题* $A$ 本身不同的层次，命题 $A$ 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 $A$"，或更松散地说"$a$ 是 $A$ 的元素"（或在同伦类型论中，"$a$ 是 $A$ 的点"）。当 $A$ 是表示命题的类型时，$a$ 可以称为 $A$ 可证明性的 *证人* (witness)，或者 $A$ 真实性的 *证据* (evidence)（甚至 $A$ 的 *证明* (proof)，但我们会尽量避免使用这种令人困惑的术语）。在这种情况下，当"$a : A$"可以在类型论中推导出时（对于某些 $a$），正好相当于在一阶逻辑中可以推导出类似的判断"$A$ 有证明"（基于假定的公理和数学编码的差异，我们将在整本书中讨论）。

另一方面，如果类型 $A$ 更像一个集合而不是命题（尽管我们将看到，这种区分可能会变得模糊），那么"$a : A$"可以看作是类比于集合论中的命题"$a \in A$"。然而，有一个本质的区别，"$a : A$"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 $b : B$

[1] 令人困惑的是，使用"命题 (proposition)"一词作为"定理 (theorem)"的同义词也是一种常见的做法。这种做法可以追溯到欧几里得。我们将自己限定在逻辑学家的用法中，按照这种用法，*命题*是一种*可以*被证明的陈述，而*定理*（或"引理 (lemma)"或"推论 (corollary)"）是*已经*被证明的陈述。因此，"$0 = 1$"及其否定"$\neg(0 = 1)$"都是命题，但只有后者是定理。

不成立”的声明，也不能“驳斥”判断“$a : A$”。

一种好的理解方式是，在集合论中，“隶属 (membership)”是一个关系，它可能在两个预先存在的对象“$a$”和“$A$”之间成立或不成立，而在类型论中，我们不能单独谈论一个元素“$a$”：每个元素本质上都是某个类型的元素，并且该类型（一般而言）是唯一确定的。因此，当我们非正式地说“令 $x$ 是一个自然数”时，在集合论中这可以简写为“令 $x$ 是一个事物并假设 $x \in \mathbb{N}$”，而在类型论中“令 $x : \mathbb{N}$”是一个原子语句：我们不能在没有指定其类型的情况下引入一个变量。

乍一看，这似乎是一种令人不适的限制，但可以说它更接近于“令 $x$ 是一个自然数”的直观数学意义。在实践中，似乎每当我们确实需要“$a \in A$”作为命题而非判断时，总是存在一个背景集合 $B$，其中 $a$ 已知是一个元素，而 $A$ 已知是一个子集。这种情况在类型论中也很容易表示，通过将 $a$ 视为类型 $B$ 的一个元素，并将 $A$ 视为 $B$ 上的一个谓词；见 §3.5。

类型论与集合论之间的最后一个差异是对等式的处理。数学中熟悉的等式概念是一个命题：例如，我们可以反驳一个等式或假设一个等式作为前提。由于在类型论中，命题是类型，这意味着等式也是一种类型：对于元素 $a, b : A$（即 $a : A$ 和 $b : A$），我们有一个类型“$a =_A b$”。（当然，在同伦类型论中，这种等式命题可以以不熟悉的方式表现出来：参见 §1.12 and Chapter 2 以及本书的其余部分）。当 $a =_A b$ 是可居住的，我们说 $a$ 和 $b$ 是 **（命题上）相等** (propositionally equal)

然而，在类型论中，还需要一种等式判断，存在于与判断“$x : A$”相同的层次上。这被称为 **判断相等** (judgmental equality)或 **定义相等** (definitional equality)我们将其写作 $a \equiv b : A$ 或简单地 $a \equiv b$。将其理解为“根据定义相等”是有帮助的。例如，如果我们通过等式 $f(x) = x^2$ 定义了一个函数 $f : \mathbb{N} \to \mathbb{N}$，那么表达式 $f(3)$ 根据定义等于 $3^2$。在理论内部，没有意义去否定或假设一个定义上的等式；我们不能说“如果 $x$ 根据定义等于 $y$，那么 $z$ 根据定义不等于 $w$”。根据定义是否相等只是扩展定义的问题；特别地，它是算法上可判定的（尽管算法本质上是元理论的，而不是理论内部的）。

随着类型论变得更加复杂，判断相等可能会变得比这更微妙，但这是一个良好的直觉起点。或者，如果我们将演绎系统视为一种代数理论，那么判断相等只是该理论中的相等，类似于群的元素之间的相等——唯一可能引起混淆的是，在类型论的演绎系统内部还存在一个对象（即类型“$a = b$”），它在内部表现为“相等”的概念。

我们需要判断相等的原因是它可以控制其他形式的判断“$a : A$”。例如，假设我们已经给出了 $3^2 = 9$ 的证明，即我们已经推导出 $p : (3^2 = 9)$ 的判断。那么同样的证人 $p$ 应该被视为 $f(3) = 9$ 的证明，因为 $f(3)$ 根据定义是 $3^2$。最好的表示方法是通过一个规则表示，给定判断 $a : A$ 和 $A \equiv B$，我们可以推导出判断 $a : B$。

因此，对于我们来说，类型论将是基于两种形式判断的演绎系统：

| 判断 | 意义 |
|---|---|
| $a : A$ | “$a$ 是类型 $A$ 的一个对象” |
| $a \equiv b : A$ | “$a$ 和 $b$ 是类型 $A$ 的定义相等对象” |

当引入一个定义上的等式时，即定义一件事物等于另一件事物时，我们将使用符号“:≡”。因此，上述函数 $f$ 的定义将写作 $f(x) :\equiv x^2$。

由于判断不能组合成更复杂的语句，因此符号“:”和“≡”的结合力比其他任何东西都要弱。[2] 因此，例如，“$p : x = y$”应解析为“$p : (x = y)$”，这是有意义的，因为“$x = y$”是一个类型，而不是“$(p : x) = y$”，这是无意义的，因为“$p : x$”是一个判断，不能与任何东西相等。类似地，“$A \equiv x = y$”只能解析为“$A \equiv (x = y)$”，尽管在极端情况下，像这样，最好添加括号来帮助理解。此外，稍后我们将使用链式相等的常见表示法——例如，写作 $a = b = c = d$ 表示“$a = b$ 和 $b = c$ 和 $c = d$，因此 $a = d$”——我们还将在这样的链中包括判断相等。上下文通常足以使意图清楚。

这也许是提到一个常见数学符号“$f : A \to B$”的适当时机，它表达了 $f$ 是从 $A$ 到 $B$ 的一个函数，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 $B$ 的函数类型的符号（这在类型论中是标准做法；参见 §1.4）。

[2] 在形式化类型论中，逗号和竖线可以结合得更松散。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，我们避免使用这种符号，直到 Appendix A。

判断可能依赖于形如 $x : A$ 的*假设*，其中 $x$ 是一个变量 而 $A$ 是一个类型。例如，在假设 $m, n : \mathbb{N}$ 的情况下，我们可以构造对象 $m + n : \mathbb{N}$。另一个例子是，假设 $A$ 是一个类型，$x, y : A$，并且 $p : x =_A y$，我们可以构造元素 $p^{-1} : y =_A x$。所有这些假设的集合被称为 **上下文** (context)；从拓扑的角度来看，它可以被视为一个"参数空间"。实际上，技术上讲，上下文必须是一个有序的假设列表，因为后来的假设可能依赖于先前的假设：假设 $x : A$ 只能在假设类型 $A$ 中出现的任何变量之后进行。如果假设中的类型 $A$ 表示一个命题，那么这个假设就是*假设*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名称。因此，在上述第二个例子中，我们可以改为说，假设 $x =_A y$，我们可以证明 $y =_A x$。然而，由于我们正在做"与证明相关的"数学，我们将经常将证明作为对象参考。例如，在上面的例子中，我们可能希望建立 $p^{-1}$ 以及传递性和自反性的证明行为类似于一个群组；参见 Chapter 2。

请注意，在这个词*假设*的意义下，我们可以假设命题等式（通过假设一个变量 $p : x = y$），但我们不能假设判断等式 $x \equiv y$，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等式的事情：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*替换*任何特定的元素 $a : A$ 以获得更具体的类型或元素。我们有时会使用类似"现在假设 $x \equiv a$"的语言来指代这个替换过程，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*判断等式，因为它不是我们可以展示一个见证的类型。然而，我们有时会将判断等式作为定理的一部分陈述，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 $f : A \to B$，然后我们做出附加判断 $f(x) \equiv y$。

在本章的其余部分，我们尝试给出类型论的非正式介绍，以满足本书的目的；在 Appendix A 中，我们给出了更正式的说明。除了某些显而易见的规则（例如判断相等的东西总是可以互相替换），类型论的规则可以分组为*类型形成器*。每个类型形成器包括一种构造类型的方法（可能使用先前构造的类型），以及构造和操作该类型的元素的规则。在大多数情况下，这些规则遵循一个相当可预测的模式，但我们不会在这里尝试使其精确化；然而，参见 §1.5 的开头以及 Chapter 5。

本章中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*。在基于判断的演绎系统描述中，*规则*是允许我们从一组判断中得出另一个判断的东西，而*公理*是我们一开始给出的判断。如果我们将演绎系统视为一个正式游戏，那么规则就是游戏规则，而公理是起始位置。如果我们将演绎系统视为一种代数理论，那么规则是该理论的操作，而公理是该理论的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（例如，从"$A \wedge B$ 有证明"推导出"$A$ 有证明"和"$B$ 有证明"的规则）：所有关于集合行为的信息都包含在公理中。相比之下，在类型论中，通常是*规则*包含了所有信息，而无需任何公理。例如，在 §1.5 中，我们将看到有一个规则允许我们从"$a : A$"和"$b : B$"中推导出判断"$(a, b) : A \times B$"，而在集合论中，类似的陈述将是（成对公理的）一个结果。

通过仅使用规则来制定类型论的优势在于规则是"过程性的"。特别是，这一特性使得类型论的良好计算属性成为可能（尽管它并不自动确保），例如"规范性 (canonicity)"。然而，虽然这种风格适用于传统类型论，但我们尚未理解如何以这种方式制定*同伦*类型论所需的一切。特别是，在 §2.7, ??, and Chapter 6 中，我们将不得不通过引入额外的公理来补充本章中介绍的类型论规则，特别是*一致性公理* (univalence axiom)。然而，在本章中，我们仅限于传统的基于规则的类型论。

## 1.2 **函数类型** (Function types)

给定类型 $A$ 和 $B$，我们可以构造类型 $A \to B$，它表示**函数** (functions)其定义域为 $A$，值域为 $B$。我们有时也将函数称为**映射** (maps)。与集合论不同，函数不是作为函数关系来定义的；而是作为类型论中的一个基本概念。我们通过规定函数的操作方式、如何构造它们以及它们诱导的等式来解释函数类型。

给定函数 $f : A \to B$ 和定义域中的一个元素 $a : A$，我们可以**应用** (apply)函数以获得值域 $B$ 的一个元素，表示为 $f(a)$，称为 $f$ 在 $a$ 处的**值** (value)。在类型论中，通常省略括号，仅用 $f\,a$ 表示 $f(a)$，我们有时也会这样做。

但我们如何构造 $A \to B$ 的元素？有两种等效的方法：直接定义或使用 $\lambda$-抽象。通过定义引入函数意味着我们通过给它一个名称——例如 $f$——来引入一个函数，并且我们定义 $f : A \to B$ 通过给出等式

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

其中 $x$ 是一个变量，$\Phi$ 是一个可以使用 $x$ 的表达式。为了使其有效，我们必须在假设 $x : A$ 的情况下检查 $\Phi : B$。

现在我们可以通过将 $\Phi$ 中的变量 $x$ 替换为 $a$ 来计算 $f(a)$。例如，考虑函数 $f : \mathbb{N} \to \mathbb{N}$，其定义为 $f(x) :\equiv x + x$。（我们将在 §1.9 中定义 $\mathbb{N}$ 和 $+$）。然后 $f(2)$ 在判断上等于 $2 + 2$。

如果我们不想为函数引入名称，我们可以使用 $\lambda$-**抽象** ($\lambda$-abstraction)给定一个类型为 $B$ 的表达式 $\Phi$，它可以使用 $x : A$，如上所述，我们写作 $\lambda(x : A).\Phi$ 以表示与(1.2.1) 相同的函数。因此，我们有

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

在前一段的示例中，我们有以下类型判断

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

作为另一个示例，对于任意类型 $A$ 和 $B$ 以及任何元素 $y : B$，我们有一个**常数函数** (constant function)$(\lambda(x : A).\, y) : A \to B$。

我们通常省略 $\lambda$-抽象中的变量 $x$ 的类型，写作 $\lambda x.\Phi$，因为从函数 $\lambda x.\Phi$ 具有类型 $A \to B$ 的判断中可以推断出变量 $x : A$ 的类型。按照约定，“$\lambda x.$”的变量绑定的“范围”是表达式的整个其余部分，除非用括号分隔。因此，例如，$\lambda x.\, x + x$ 应解析为 $\lambda x.\, (x + x)$，而不是 $(\lambda x.\, x) + x$（在这种情况下，无论如何它将是类型不合法的）。

另一种等效的符号是

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

我们有时也可以在表达式 $\Phi$ 中使用空白符号“$-$”代替变量，以表示隐含的 $\lambda$-抽象。例如，$g(x, -)$ 是另一种写作 $\lambda y.\, g(x, y)$ 的方式。

现在 $\lambda$-抽象是一个函数，因此我们可以将其应用于一个参数 $a : A$。然后我们有以下**计算规则** (computation rule)，这是一种定义相等性[3]：

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

其中 $\Phi'$ 是表达式 $\Phi$，其中所有 $x$ 的出现都被替换为 $a$。继续上述示例，我们有

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

注意，从任意函数 $f : A \to B$，我们可以构造一个 $\lambda$-抽象函数 $\lambda x.\, f(x)$。由于这在定义上是“将 $f$ 应用于其参数的函数”，我们认为它在定义上等于 $f$：[4]

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

此等式是**函数类型的唯一性原则** (uniqueness principle for function types)，因为它表明 $f$ 是由其值唯一确定的。

通过显式参数定义函数的引入可以通过使用 $\lambda$-抽象来简化为简单的定义：即，我们可以通过

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

的定义来读取 $f : A \to B$ 作为

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

在涉及变量的计算中，我们必须小心，当用涉及变量的表达式替换一个变量时，因为我们要保持表达式的绑定结构。所谓**绑定结构** (binding structure)，是指由诸如 $\lambda$、$\Pi$ 和 $\Sigma$ 等绑定符号（我们很快就

[3] 使用这个等式通常被称为 $\beta$-**变换** ($\beta$-conversion)$\beta$-变换或 $\beta$-**约简** ($\beta$-reduction)。$\beta$-变换

[4] 使用这个等式通常被称为 $\eta$-**变换** ($\eta$-conversion)或 $\eta$-**展开** ($\eta$-expansion)。

会见到后者）生成的无形联系，连接了引入变量的位置和使用变量的位置。例如，考虑 $f : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$ 的定义

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

y : ℕ f(y)$\lambda y.\, x + y$"定义 $f(x)$ 中的 $x$ 替换为 $y$，得到 $\lambda y.\, y + y$，这样做是错误的，因为这意味着 $y$ 被**捕获** (captured)。之前，替换的 $y$ 是指我们的假设，但现在它指的是 $\lambda$-抽象的参数。因此，这种天真的替换将破坏绑定结构，使我们能够进行语义上不正确的计算。

但是在这个例子中，$f(y)$ 是什么？请注意，受限（或"虚设的"）变量如表达式 $\lambda y.\, x + y$ 中的 $y$ 只有局部意义，并且可以一致地替换为任何其他变量，从而保持绑定结构。实际上，$\lambda y.\, x + y$ 被判断上等于[5]为 $\lambda z.\, x + z$。由此可得 $f(y)$ 判断上等于 $\lambda z.\, y + z$，并且这回答了我们的问题。（代替 $z$，任何与 $y$ 不同的变量都可以使用，得到相同的结果。）

当然，这对于任何数学家来说都应该很熟悉：这与如果 $f(x) :\equiv \int_1^2 \frac{dt}{x-t}$，则 $f(t)$ 不是 $\int_1^2 \frac{dt}{t-t}$，而是 $\int_1^2 \frac{ds}{t-s}$ 的现象是相同的。$\lambda$-抽象绑定虚设变量的方式与积分完全相同。

我们已经看到了如何定义一个变量的函数。一种定义多个变量函数的方法是使用笛卡尔积，这将在后面介绍；具有参数 $A$ 和 $B$ 并且结果在 $C$ 中的函数将具有类型 $f : A \times B \to C$。但是，还有另一种选择，它避免使用积类型，这称为**柯里化** (currying)（以数学家 Haskell Curry 命名）。

柯里化的想法是将具有两个输入 $a : A$ 和 $b : B$ 的函数表示为一个函数，该函数接受一个输入 $a : A$ 并返回另一个函数，然后该函数接受第二个输入 $b : B$ 并返回结果。即，我们认为两个变量的函数属于迭代函数类型 $f : A \to (B \to C)$。我们也可以在没有括号的情况下写作 $f : A \to B \to C$，默认约定为右结合。然后给定 $a : A$ 和 $b : B$，我们可以将 $f$ 应用于 $a$，然后将结果应用于 $b$，得到 $f(a)(b) : C$。为了避免括号的泛滥，我们允许自己写作 $f(a)(b)$ 为 $f(a,b)$，即使没有涉及积。在完全省略括号的情况下，我们写作 $f\,a\,b$ 为 $(f\,a)\,b$，现在默认结合性为左，使得 $f$ 按正确顺序应用其参数。

我们对显式参数定义的符号扩展到这种情况：我们可以通过给出等式 $f(x,y) :\equiv \Phi$ 定义一个命名函数 $f : A \to B \to C$，其中在假设 $x : A$ 和 $y : B$ 的情况下 $\Phi : C$。使用 $\lambda$-抽象，这相当于

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

也可以写作

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

我们也可以通过写作多个空白符号来隐式抽象多个变量，例如 $g(-,-)$ 意味着 $\lambda x.\, \lambda y.\, g(x,y)$。对三个或更多参数函数进行柯里化是我们刚刚描述的一个直接扩展。

## 1.3 宇宙和族 (Universes and families)

到目前为止，我们一直在非正式地使用表达式"$A$ 是一个类型"。我们现在将通过引入**宇宙** (universes)来使这一点更加精确。宇宙是一个类型，其元素是类型。与朴素集合论类似，我们可能希望有一个包含所有类型的宇宙 $\mathcal{U}_\infty$，包括其自身（即 $\mathcal{U}_\infty : \mathcal{U}_\infty$）。然而，正如在集合论中一样，这是不健全的，即我们可以从中推导出每个类型，包括表示命题 False 的空类型（参见 §1.7）都是居留的。例如，使用树作为集合的表示，我们可以直接编码 Russell 悖论 [?]。

为了避免悖论，我们引入了一个宇宙的层次结构

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

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** (cumulative)，即 $i$ 阶宇宙的所有元素也是 $i+1$ 阶宇宙的元素，即如果 $A : \mathcal{U}_i$，那么 $A : \mathcal{U}_{i+1}$ 也是如此。这很方便，但也带来了一些不太愉快的后果，即元素不再具有唯一的类型，并且在其他方面也有点棘手，这里不再赘述；详见注释。

[5]使用这个等式通常被称为 $\alpha$-**变换** ($\alpha$-conversion)。$\alpha$-变换

当我们说 $A$ 是一个类型时，我们的意思是它居于某个宇宙 $\mathcal{U}_i$。我们通常希望避免明确提及层级$i$，并假设层级可以以一致的方式分配；因此我们可以写作 $A:\mathcal{U}$，省略层级。这样我们甚至可以写作 $\mathcal{U}:\mathcal{U}$，这可以理解为 $\mathcal{U}_i:\mathcal{U}_{i+1}$，将指标隐含地留下。以这种方式书写宇宙称为**典型的歧义** (typical ambiguity)。这很方便，但也有点危险，因为它允许我们写出看似有效的证明，这些证明重现了自指的悖论。如果有任何关于一个论证是否正确的疑问，检查的方法是尝试为其中出现的所有宇宙一致地分配层级。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** (small types)

为了建模在给定类型 $A$ 上变化的类型集合，我们使用其type!family of| 类型! 族的值域为宇宙的函数 $B:A\to\mathcal{U}$。这些函数称为**类型族** (families of types)（有时也称为依赖类型 (dependent types)）；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合的族 $\mathsf{Fin}:\mathbb{N}\to\mathcal{U}$，其中 $\mathsf{Fin}(n)$ 是一个恰好有 $n$ 个元素的类型。（我们现在还不能定义族 $\mathsf{Fin}$——实际上，我们甚至还没有引入它的定义域 $\mathbb{N}$——但我们很快就会这样做；参见 Exercise 1.9）。我们可以用 $0_n, 1_n, \dots, (n-1)_n$ 来表示 $\mathsf{Fin}(n)$ 的元素，使用下标强调 $\mathsf{Fin}(n)$ 的元素与 $\mathsf{Fin}(m)$ 的元素不同，如果 $n$ 与 $m$ 不同，并且它们都与普通自然数不同（我们将在 §1.9 中介绍）。

一个更为平凡（但非常重要）的类型族示例是一个**常量类型族** (constant type family)，位于类型 $B:\mathcal{U}$，它当然是常函数 $(\lambda(x:A).B):A\to\mathcal{U}$。

作为一个非示例，在我们版本的类型论中，没有类型族“$\lambda(i:\mathbb{N}).\mathcal{U}_i$”。事实上，没有足够大的宇宙可以作为它的值域。此外，我们甚至不将宇宙 $\mathcal{U}_i$ 的索引 $i$ 与类型论中的自然数 $\mathbb{N}$ （后者将在 §1.9 中引入）等同。

## 1.4 依赖函数类型（Π-类型）(Dependent function types (Π-types))

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** (Π-type) 或**依赖函数类型** (dependent function type)。Π-类型的元素是**依赖函数** (dependent functions)，其值域类型可以根据函数应用到的定义域元素而变化。“Π-类型”这个名称之所以使用，是因为该类型也可以看作是给定类型上的笛卡尔积。

给定类型 $A:\mathcal{U}$ 和类型族 $B:A\to\mathcal{U}$，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表示法，例如

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

如果 $B$ 是一个常量族，那么依赖积类型就是普通函数类型：

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

实际上，所有 Π-类型的构造都是对普通函数类型相应构造的推广。

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 $f$ 是待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 $\Phi:B(x)$，我们写作

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

或者，我们可以使用 $\lambda$-**抽象** ($\lambda$-abstraction)

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

与非依赖函数一样，我们可以将依赖函数 $f:\prod_{(x:A)}B(x)$ 应用于参数 $a:A$，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 $f(a)\equiv\Phi'$ 和 $(\lambda x.\Phi)(a)\equiv\Phi'$，其中 $\Phi'$ 是通过将 $\Phi$ 中的所有 $x$ 替换为 $a$ 获得的（始终避免变量捕获）。同样，对于任何 $f:\prod_{(x:A)}B(x)$，我们有唯一性原则 $f\equiv(\lambda x.f(x))$。

例如，回想一下 §1.3 中有一个类型族 $\mathsf{Fin}:\mathbb{N}\to\mathcal{U}$，其值是标准有限集合，具有元素 $0_n, 1_n, \dots, (n-1)_n:\mathsf{Fin}(n)$。然后有一个依赖函数 $\mathsf{fmax}:\prod_{(n:\mathbb{N})}\mathsf{Fin}(n+1)$，它返回每个非空有限类

型的“最大”元素，即 $\mathsf{fmax}(n) :\equiv n_{n+1}$。如同 Fin 本身的情况一样，我们还不能定义 fmax，但我们很快就能做到；参见 Exercise 1.9。

我们现在可以定义的另一类重要的依赖函数类型是**多态函数** (polymorphic functions)在给定宇宙上。多态函数是一个以类型作为其参数之一的函数，然后对该类型的元素（或从中构造的其他类型的元素）进行操作。一个例子是多态恒等函数 $\mathsf{id} : \prod_{(A:\mathcal{U})} A \to A$，我们定义为 $\mathsf{id} :\equiv \lambda(A:\mathcal{U}).\lambda(x:A).x$。（如同 $\lambda$-抽象一样，$\Pi$s 自动对表达式的其余部分进行范围限定，除非用括号限定；因此 $\mathsf{id} : \prod_{(A:\mathcal{U})} A \to A$ 意味着 $\mathsf{id} : \prod_{(A:\mathcal{U})}(A \to A)$。这个约定虽然在数学中不常见，但在类型论中很常见。）

我们有时会将依赖函数的某些参数写作下标。例如，我们可以等效地定义多态恒等函数为 $\mathsf{id}_A(x) :\equiv x$。此外，如果可以从上下文推断出一个参数，我们可以完全省略它。例如，如果 $a : A$，那么写作 $\mathsf{id}(a)$ 是不含糊的，因为 id 必须意味着 $\mathsf{id}_A$，以便它可以应用于 $a$。

另一个稍微复杂的多态函数示例是“交换”操作，它交换（柯里化的）两个参数函数的顺序：

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

我们可以定义它为

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

我们也可以将类型参数作为下标写作：

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

请注意，如同我们对普通函数所做的，我们使用柯里化来定义具有多个参数的依赖函数（例如 swap）。然而，在依赖情况下，第二个定义域可以依赖于第一个定义域，而值域可以依赖于两者。即，给定 $A : \mathcal{U}$ 和类型族 $B : A \to \mathcal{U}$ 以及 $C : \prod_{(x:A)} B(x) \to \mathcal{U}$，我们可以构造类型 $\prod_{(x:A)} \prod_{(y:B(x))} C(x, y)$，这是具有两个参数的函数类型。如果 $B$ 是常量并且等于 $A$，我们可以简化表示法并写作 $\prod_{(x,y:A)}$；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 $a : A$ 以及 $b : B(a)$，我们有 $f(a)(b) : C(a, b)$，如前所述，我们将其写作 $f(a, b) : C(a, b)$。

## 1.5 积类型 (Product types)

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** (cartesian product)。我们还引入了一个零元积类型，称为**单位类型** (unit type) $\mathbf{1} : \mathcal{U}$。我们期望 $A \times B$ 的元素是对偶 $(a, b) : A \times B$，其中 $a : A$ 和 $b : B$，而 $\mathbf{1}$ 的唯一元素是某个特定对象 $\star : \mathbf{1}$。然而，与集合论中我们将有序对定义为特定集合然后将它们全部收集到笛卡尔积中不同，在类型论中，有序对是一个原始概念，与函数一样。

Remark 1.5.1. 在类型论中引入新类型的一般模式如下。我们在 §§1.2 and 1.4 中已经看到了这一模式[6]，所以值得强调一下其一般形式。要指定一个类型，我们需要指定：

(i) 如何通过**形成规则** (formation rules)形成这种新类型。（例如，当 $A$ 是一个类型并且 $B$ 是一个类型时，我们可以形成函数类型 $A \to B$。当 $A$ 是一个类型并且 $B(x)$ 是 $x : A$ 的类型时，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。这些称为该类型的**构造子** (constructors) 或**引入规则** (introduction rules)。（例如，函数类型有一个构造子，即 $\lambda$-抽象。回想一下，像 $f(x) :\equiv 2x$ 这样的直接定义可以等效地表达为 $\lambda$-抽象 $f :\equiv \lambda x.\, 2x$。）

[6] 上述宇宙的描述是一个例外。

(iii) 如何使用该类型的元素。这些称为该类型的**消去子** (eliminators) 或**消去规则** (elimination rules)。（例如，函数类型有一个消去子，即函数应用。）

(iv) 一个**计算规则** (computation rule)，表示消去子如何作用于构造子。（例如，对于函数，计算规则表明 $(\lambda x.\Phi)(a)$ 在判断上等于将 $a$ 代入 $\Phi$ 中的结果。）

(v) 一个可选的**唯一性原则** (uniqueness principle)，它表示从或向该类型映射的唯一性原则。对于某些类型，唯一性原则通过表明该类型的每个元素都可以通过将消去子应用于它来唯一确定，并且可以通过应用构造子从这些结果中重新构造，从而描述了向该类型的映射——因此以与计算规则对偶的方式表达了构造子如何作用于消去子。（例如，对于函数，唯一性原则表明任何函数 $f$ 在判断上等于"展开"的函数 $\lambda x.f(x)$，因此由其值唯一确定。）对于其他类型，唯一性原则表明从该类型的每个映射（函数）都是由某些数据唯一确定的。（一个例子是 §1.7 中引入的并类型，其唯一性原则在 §2.9 中提到。）

当唯一性原则不作为判断等式规则采用时，通常可以作为该类型的其他规则的*命题*等式证明。在这种情况下，我们称之为**命题唯一性原则** (propositional uniqueness principle)。（在后续章节中，我们还将偶尔遇到*命题计算规则* (propositional computation rules)。）

Appendix A.2 中的推理规则根据这种方式进行了组织和命名；参见 Appendix A.2.4，其中每种可能性都得到了实现。

构造对偶的方法显而易见：给定 $a:A$ 和 $b:B$，我们可以形成 $(a,b):A\times B$。同样，$\mathbf{1}$ 的元素的唯一构造方式是 $\star:\mathbf{1}$。我们期望"$A\times B$ 的每个元素都是对偶"，这是积的唯一性原则；我们不会将其作为类型论的规则，但我们将在稍后以命题等式的形式证明它。

现在，我们如何*使用*对偶，即我们如何定义从积类型到其他类型的函数？让我们首先考虑一个非依赖函数 $f:A\times B\to C$ 的定义。由于我们期望 $A\times B$ 的唯一元素是对偶，因此我们期望能够通过规定 $f$ 应用于对偶 $(a,b)$ 时的结果来定义这样的函数。我们可以通过提供一个函数 $g:A\to B\to C$ 来规定这些结果。因此，我们引入了一个新规则（积的消去规则），它表明对于任何这样的 $g$，我们可以定义一个函数 $f:A\times B\to C$ 为

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

我们在这里避免写作 $g(a,b)$，以强调 $g$ 不是一个关于积的函数。（然而，在本书的后续部分，我们将经常将 $g(a,b)$ 写作用于积函数和柯里化的双变量函数。）这个定义等式是积类型的计算规则。

请注意，在集合论中，我们会通过 $A\times B$ 的每个元素都是有序对的事实来证明上述 $f$ 的定义是合理的，因此定义 $f$ 在这些对偶上的值就足够了。相比之下，类型论颠倒了这种情况：我们假设当我们指定其在对偶上的值时，定义在 $A\times B$ 上的函数是合理的，并且从这一点（或者更准确地说，从其更一般的依赖函数版本）我们将能够*证明* $A\times B$ 的每个元素都是对偶。从范畴论的角度看，我们可以说我们定义了积 $A\times B$，使其成为"指数"$B\to C$ 的左伴随，我们已经引入了这一点。

例如，我们可以导出**投影函数** (projection functions)函数

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

其定义等式为

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

我们可以定义一个函数类型为

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

并定义等式为

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

然后，我们可以通过定义

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

来代替直接通过定义等式定义函数 $\mathsf{pr}_1$ 和 $\mathsf{pr}_2$。我们将函数 $\mathsf{rec}_{A\times B}$ 称为积类型的**递归子** (recursor)。这里使用"递归子"这个名字有点不幸，因为没有发生递归。这个名字来源于积类型是归纳类型一般框架的一个退化示例，对于诸如自然数的类型，递归子实际上将是递归的。我们也可以将笛卡尔积的递归原则称为上面所描述的通过给出对偶上的值来定义函数 $f : A \times B \to C$ 的事实。

我们留作简单的练习来展示如何从投影中导出递归子，反之亦然。

我们还定义了单位类型的递归子：

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

并定义等式为

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

虽然我们包括它以保持类型定义的模式，但单位的递归子完全没有用，因为我们可以直接通过简单地忽略单位类型的参数来定义这样的函数。

为了能够在积类型上定义依赖函数，我们必须泛化递归子。给定 $C : A \times B \to \mathcal{U}$，我们可以通过提供一个函数 $g : \prod_{(x:A)} \prod_{(y:B)} C((x, y))$ 并定义等式为

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

例如，通过这种方式我们可以证明命题唯一性原则，该原则指出 $A \times B$ 的每个元素都等于一个对偶。具体来说，我们可以构造一个函数

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

这里我们使用的是身份类型 (identity type)，我们将在 §1.12 中引入。然而，我们现在需要知道的是，对于任何 $x : A$，都有一个反身性元素 $\mathsf{refl}_x : x =_A x$。有了这个，我们可以定义

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

这种构造是有效的，因为在 $x :\equiv (a, b)$ 的情况下，我们可以计算

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

使用投影的定义等式。因此，

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

是类型良好的，因为等式的两边在判断上是相等的。

更一般地说，通过这种方式定义依赖函数的能力意味着要证明一个积的所有元素的属性，足以证明其规范元素（即有序对）的属性。当我们到达归纳类型（如自然数）时，对应的属性将是能够通过归纳法编写证明。因此，如果我们像上面那样做，并在通用情况下应用此原则一次，我们称产生的函数为积类型的**归纳法** (induction)：给定 $A, B : \mathcal{U}$，我们有

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

其定义等式为

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

同样，我们可以说通过积的归纳法得到的依赖函数被从积类型的**归纳原则** (induction principle) 中获得。很容易看出，递归子只是当族 $C$ 是常量时归纳法的特例。由于归纳法描述了如何使用积类型的元素，因此归纳法也被称为**（依赖）消去子** (dependent eliminator)，而递归子则是**非依赖消去子** (non-dependent eliminator)。

单位类型的归纳法比递归子更有用：

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

其定义等式为

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

归纳法使我们能够证明 $\mathbf{1}$ 的命题唯一性原则，即它的唯一居留元素是 $\star$。也就是说，我们可以构造

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

通过使用定义等式

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

或者等效地通过使用归纳法：

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

## 1.6 依赖对类型（Dependent pair types，Σ-types）

就像我们将函数类型（function types）（§1.2）推广到依赖函数类型（dependent function types）（§1.4）一样，通常我们也可以将积类型（product types）（§1.5）推广，以允许一对中的第二个分量的类型根据第一个分量的选择而变化。这被称为**依赖对类型**（dependent pair type），或者 Σ-type，因为在集合论中它对应于给定类型上的索引和（即并集或不交并集）。

给定类型 $A : \mathcal{U}$ 和族 $B : A \to \mathcal{U}$，依赖对类型记作 $\sum_{(x:A)} B(x) : \mathcal{U}$。替代的记号有：

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

就像其他绑定构造（例如 $\lambda$-抽象和 $\Pi$）一样，$\Sigma$ 的作用域 也自动覆盖表达式的其余部分，除非用括号限定，例如 $\sum_{(x:A)} B(x) \to C$ 表示 $\sum_{(x:A)} (B(x) \to C)$。

构造依赖对类型的元素的方法是配对：如果给定 $a : A$ 和 $b : B(a)$，则有 $(a, b) : \sum_{(x:A)} B(x)$。如果 $B$ 是常数，则依赖对类型是普通的笛卡尔积类型：

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

所有关于 Σ-types 的构造都可以看作是关于积类型的构造的直接推广，依赖函数通常替代非依赖函数。

例如，递归原理 表明，要定义从 Σ-type 到非依赖函数 $f : \left(\sum_{(x:A)} B(x)\right) \to C$，我们需要提供一个函数 $g : \prod_{(x:A)} B(x) \to C$，然后我们可以通过定义

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

来定义 $f$。例如，我们可以从 Σ-type 中推导出第一个投影：

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

通过定义方程

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

然而，由于一对的第二个分量的类型 $(a,b):\sum_{(x:A)} B(x)$ 是 $B(a)$，第二个投影必须是依赖的函数，其类型涉及第一个投影函数：

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

因此我们需要 归纳原理 对于 Σ-types（即"依赖消解器"）。这表明，为了构造从 Σ-type 到家族 $C:(\sum_{(x:A)} B(x)) \to \mathcal{U}$ 的依赖函数，我们需要一个函数

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

然后我们可以导出一个函数

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

定义方程为：

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

应用此定义 $C(p) :\equiv B(\mathsf{pr}_1(p))$，我们可以定义 $\mathsf{pr}_2 : \prod_{(p:\sum_{(x:A)} B(x))} B(\mathsf{pr}_1(p))$ 方程为

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

为了确认这是正确的，我们注意到 $B(\mathsf{pr}_1((a,b))) \equiv B(a)$，使用 $\mathsf{pr}_1$ 的定义方程，并且确实 $b:B(a)$。
我们可以将递归和归纳原理打包到 Σ 的递归器中：

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

定义方程为：

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

以及相应的归纳操作符：

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

定义方程为：

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

如前所述，当 $C$ 是常数时，递归器是归纳的特例。
再举一个例子，考虑以下原则，其中 $A$ 和 $B$ 是类型，$R:A\to B\to\mathcal{U}$：

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

我们可以将 $R$ 视为 $A$ 和 $B$ 之间的"与证明相关的关系"，其中 $R(a,b)$ 是 $a:A$ 和 $b:B$ 相关性的见证类型。然后 ac 直观地表明，如果我们有一个依赖函数 $g$，它为每个 $a:A$ 分配一个依赖对 $(b,r)$，其中 $b:B$ 和 $r:R(a,b)$，那么我们有一个函数 $f:A\to B$，并为每个 $a:A$ 分配一个见证，证明

$R(a, f(a))$。我们的直觉告诉我们，我们可以将 $g$ 的值分解为它们的分量。实际上，使用我们刚刚定义的投影，我们可以定义：

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

为了验证这是否是良类型的，注意到如果 $g : \prod_{(x:A)} \sum_{(y:B)} R(x,y)$，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 $\mathsf{ac}$ 的余域中的类型家族 $\lambda f.\, \prod_{(x:A)} R(x, f(x))$ 应用于函数 $\lambda x.\, \mathsf{pr}_1(g(x))$ 的结果：

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

因此，我们有

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

如要求所示。

如果我们将 $\Pi$ 读作“对于所有”而将 $\Sigma$ 读作“存在”，那么函数 $\mathsf{ac}$ 的类型表示：*如果对于所有 $x : A$ 存在 $y : B$ 使得 $R(x,y)$，那么存在一个函数 $f : A \to B$ 使得对于所有 $x : A$ 我们有 $R(x, f(x))$*。由于这听起来像选择公理的一个版本，函数 $\mathsf{ac}$ 传统上被称为**类型论的选择公理**（type-theoretic axiom of choice），如我们刚才所示，它可以直接从类型论的规则中证明，而不必作为公理。然而，注意到实际上没有涉及选择，因为选择已经在前提中给出：我们所要做的就是将其分解为两个函数：一个表示选择，另一个表示其正确性。在 §3.8 中，我们将给出一个更接近通常选择公理的“选择公理”的另一个表述。

依赖对类型通常用于定义数学结构的类型，这些结构通常由多个依赖数据构成。举一个简单的例子，假设我们要定义一个 magma，它是一个类型 $A$ 以及一个二元运算 $m : A \to A \to A$。短语“together with”（以及同义的“equipped with”）的精确含义是“一个 magma”是一个对 $(A, m)$，其中包含类型 $A : \mathcal{U}$ 和运算 $m : A \to A \to A$。由于这个对的第二个分量 $m$ 的类型 $A \to A \to A$ 依赖于其第一个分量 $A$，因此这些对属于依赖对类型。因此，定义“一个 magma 是一个类型 $A$ 和一个二元运算 $m : A \to A \to A$”应理解为定义 magma *的类型*为

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

给定一个 magma，我们可以使用第一个投影 $\mathsf{pr}_1$ 提取其基础类型（即其“载体”），并使用第二个投影 $\mathsf{pr}_2$ 提取其运算。当然，由多个数据构建的结构需要迭代对类型，这些对类型可能只是部分依赖的；例如，带基点的 magma（即配备有基点 $e : A$ 的 magma $(A, m)$）的类型为

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

我们通常还希望对这样的结构施加公理，例如将带基点的 magma 转变为单子或群。这也可以使用 $\Sigma$-types 来完成；见 §1.11。

在本书的其余部分，我们有时会明确地进行这样的定义，但最终我们信任读者将其从英语翻译成 $\Sigma$-types。我们还通常遵循常见的数学惯例，使用相同的字母表示这种结构及其载体（这意味着在符号中隐含地省略适当的投影函数）：也就是说，我们将使用带有运算 $m : A \to A \to A$ 的 magma $A$。注意，$\mathsf{PointedMagma}$ 的典型元素的形式为 $(A, (m, e))$，其中 $A : \mathcal{U}$，$m : A \to A \to A$，$e : A$。由于这种迭代 $\Sigma$-types 频繁出现，我们使用通常的有序三元组、四元组等符号来表示依赖的嵌套对，默认向右结合。也就是说，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（Coproduct types）

给定 $A, B : \mathcal{U}$，我们引入它们的**余积类型**（coproduct type）$A + B : \mathcal{U}$。这在集合论中对应于不交并（disjoint union），我们也可以用这个名称。在类型论中，与函数和积类型的情况一样，余积必须是一种基础构造，因为没有"类型联合"的预定义概念。我们还引入一个零元版本：**空类型**（empty type）$\mathbf{0} : \mathcal{U}$。

构造 $A + B$ 的元素有两种方式，要么是 $\mathsf{inl}(a) : A + B$ 其中 $a : A$，要么是 $\mathsf{inr}(b) : A + B$ 其中 $b : B$。（名称 inl 和 inr 分别是"左注入"（left injection）和"右注入"（right injection）的缩写。）没有构造空类型元素的方法。

要构造从 $A + B$ 到 $C$ 的非依赖函数 $f : A + B \to C$，我们需要提供函数 $g_0 : A \to C$ 和 $g_1 : B \to C$。然后 $f$ 的定义如下：

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

也就是说，函数 $f$ 是通过**情况分析**（case analysis）定义的。同样，我们可以导出余积类型的递归器：

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

其定义方程为：

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

我们总是可以构造一个从 $\mathbf{0}$ 到 $C$ 的函数 $f : \mathbf{0} \to C$，因为在 $\mathbf{0}$ 上没有元素可以定义 $f$。因此，$\mathbf{0}$ 的递归器为：

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

其构造了从空类型到任何其他类型的规范函数。从逻辑上讲，它对应于原理 ex falso quodlibet。

要从余积中构造依赖函数 $f : \prod_{(x:A+B)} C(x)$，我们假设给定了家族 $C : (A + B) \to \mathcal{U}$，并且需要提供：

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

这样我们可以得到 $f$，定义方程为：

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

我们将这个方案打包成余积类型的归纳原理：

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

同样，递归器出现在 $C$ 是常量的情况下。

空类型的归纳原理

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

提供了一种从空类型中定义一个琐碎的依赖函数的方法。

## 1.8 布尔类型（The type of booleans）

布尔类型 $\mathbf{2} : \mathcal{U}$ 被设计为只有两个元素 $0_2, 1_2 : \mathbf{2}$。很显然，我们可以将这个类型构造为余积类型 和单位类型 的余积 $\mathbf{1} + \mathbf{1}$。然而，由于它的频繁使用，我们在这里给出显式规则。实际上，我们还将观察到，我们也可以通过 $\Sigma$-types 和 $\mathbf{2}$ 导出二元余积。

要导出一个函数 $f : \mathbf{2} \to C$，我们需要 $c_0, c_1 : C$，并添加定义方程：

$$f(0_2) :\equiv c_0,$$
$$f(1_2) :\equiv c_1.$$

递归器对应于函数式编程中的 if-then-else 构造：

$$\mathsf{rec}_2 : \prod_{C:\mathcal{U}} C \to C \to \mathbf{2} \to C$$

定义方程为：

$$\mathsf{rec}_2(C, c_0, c_1, 0_2) :\equiv c_0,$$
$$\mathsf{rec}_2(C, c_0, c_1, 1_2) :\equiv c_1.$$

给定 $C : \mathbf{2} \to \mathcal{U}$，要导出依赖函数 $f : \prod_{(x:\mathbf{2})} C(x)$，我们需要 $c_0 : C(0_2)$ 和 $c_1 : C(1_2)$，在这种情况下我们可以给出定义方程：

$$f(0_2) :\equiv c_0,$$
$$f(1_2) :\equiv c_1.$$

我们将其打包为归纳原理：

$$\mathsf{ind}_2 : \prod_{(C:\mathbf{2}\to\mathcal{U})} C(0_2) \to C(1_2) \to \prod_{(x:2)} C(x)$$

定义方程为：

$$\mathsf{ind}_2(C, c_0, c_1, 0_2) :\equiv c_0,$$
$$\mathsf{ind}_2(C, c_0, c_1, 1_2) :\equiv c_1.$$

例如，使用归纳原理，我们可以推导出，如预期的那样，布尔类型的每个元素要么是 $1_2$，要么是 $0_2$。同样，为了陈述这个结果，我们使用尚未介绍的等式类型，但我们只需要知道每个元素都等于它自身，即 $\mathsf{refl}_x : x = x$。因此，我们构造了一个元素：

$$\prod_{x:\mathbf{2}} (x = 0_2) + (x = 1_2), \tag{1.8.1}$$

即：给每个 $x : \mathbf{2}$ 赋予一个等式 $x = 0_2$ 或一个等式 $x = 1_2$。我们使用 $\mathbf{2}$ 的归纳原理定义这个元素，其中 $C(x) :\equiv (x = 0_2) + (x = 1_2)$；两个输入分别是 $\mathsf{inl}(\mathsf{refl}_{0_2}) : C(0_2)$ 和 $\mathsf{inr}(\mathsf{refl}_{1_2}) : C(1_2)$。换句话说，我们构造的元素为：

$$\mathsf{ind}_2\big(\lambda x.\, (x = 0_2) + (x = 1_2),\, \mathsf{inl}(\mathsf{refl}_{0_2}),\, \mathsf{inr}(\mathsf{refl}_{1_2})\big).$$

我们已经指出，$\Sigma$-types 可以看作是类似于索引不交并，而余积是二元不交并。自然地，我们期望二元不交并 $A + B$ 可以通过两元素类型 $\mathbf{2}$ 构造为一个索引不交并。为此，我们需要一个类型族 $P : \mathbf{2} \to \mathcal{U}$，使得 $P(0_2) \equiv A$ 且 $P(1_2) \equiv B$。实际上，我们可以通过 $\mathbf{2}$ 的递归原理得到这样的家族。（使用归纳和递归定义类型族的能力，利用宇宙 $\mathcal{U}$ 也是一个类型的事实，是类型论的一个微妙而重要的方面。）因此，我们可以定义：

$$A + B :\equiv \sum_{x:\mathbf{2}} \mathsf{rec}_2(\mathcal{U}, A, B, x)$$

其构造方法如下：

$$\mathsf{inl}(a) :\equiv (0_2, a),$$
$$\mathsf{inr}(b) :\equiv (1_2, b).$$

我们留作练习，将这一定义中的余积类型的归纳原理推导出来。（参见 Exercise 1.5 and §5.2。）
我们可以将同样的想法应用于积类型（products）和 Π-types：我们可以定义

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

然后使用 **2** 的归纳构造对的类型：

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

而投影函数则为直接应用：

$$\mathsf{pr}_1(p) :\equiv p(0_2),$$
$$\mathsf{pr}_2(p) :\equiv p(1_2).$$

以这种方式定义二元积的归纳原理的推导稍微复杂一些，并且需要函数外延性（function extensionality），我们将在 §2.7 中介绍。此外，我们并没有得到相同的判断性等式；参见 Exercise 1.6。这是在用一种类型编码另一种类型时经常出现的问题；我们将在 §5.5 中再次讨论它。
我们偶尔会将 **2** 的元素 $0_2$ 和 $1_2$ 分别称为“假”（false）和“真”（true）。然而，请注意，与经典 数学不同，我们不使用 **2** 的元素作为真值 或命题。（相反，我们将命题与类型等同；参见 §1.11。）特别是，类型 $A \to \mathbf{2}$ 一般不是 $A$ 的幂集；它仅表示 $A$ 的“可判定”子集（参见 Chapter 3）。

## 1.9 自然数（The natural numbers）

到目前为止，我们已经有了通过抽象操作构造新类型的规则，但是为了进行具体的数学操作，我们还需要一些具体的类型，比如数字类型。最基本的这种类型是自然数的类型 $\mathbb{N} : \mathcal{U}$；一旦我们有了这个，我们就可以构造整数、有理数、实数，等等（见 Chapter 11）。
$\mathbb{N}$ 的元素是通过 $0 : \mathbb{N}$ 和后继 操作 $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$ 来构造的。在表示自然数时，我们采用通常的十进制表示法，如 $1 :\equiv \mathsf{succ}(0)$，$2 :\equiv \mathsf{succ}(1)$，$3 :\equiv \mathsf{succ}(2)$，依此类推。
自然数的一个基本属性是我们可以通过递归定义函数，并通过归纳进行证明——这里的“递归”和“归纳”有了更为熟悉的含义。为了通过递归从自然数中构造一个非依赖函数 $f : \mathbb{N} \to C$，只需提供一个起始点 $c_0 : C$ 和一个“下一步”函数 $c_s : \mathbb{N} \to C \to C$。这产生了一个定义方程为 的 $f$：

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

我们说 $f$ 是通过**原始递归**定义的。
例如，我们看看如何定义一个在自然数上的函数，该函数将其参数加倍。在这种情况下，我们有 $C :\equiv \mathbb{N}$。首先，我们需要提供 $\mathsf{double}(0)$ 的值，这很容易：我们设定 $c_0 :\equiv 0$。接下来，要计算 $\mathsf{double}(\mathsf{succ}(n))$ 的值，我们首先计算 $\mathsf{double}(n)$ 的值，然后执行两次后继操作。这通过递推公式 $c_s(n, y) :\equiv \mathsf{succ}(\mathsf{succ}(y))$ 表示。注意，$c_s$ 的第二个参数 $y$ 代表*递归调用* $\mathsf{double}(n)$ 的结果。
通过这种方式，用原始递归定义 $\mathsf{double} : \mathbb{N} \to \mathbb{N}$，因此我们得到定义方程：

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

这确实具有正确的计算行为：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，通过柯里化并允许 $C$ 是一个函数类型。例如，我们定义加法 $\mathsf{add} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$，其中 $C :\equiv \mathbb{N} \to \mathbb{N}$，并使用以下“起始点”和“下一步”数据：

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

我们因此得到 $\mathsf{add} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$，其满足定义等式

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

通常，我们将 $\mathsf{add}(m, n)$ 写作 $m + n$。读者可以验证 $2 + 2 \equiv 4$。
与之前的情况一样，我们可以将原始递归的原则打包成一个递归器：

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

定义等式为

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

使用 $\mathsf{rec}_{\mathbb{N}}$，我们可以如下表示 $\mathsf{double}$ 和 $\mathsf{add}$：

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

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

当然，只使用原始递归原则定义的所有函数都将是可计算的。（然而，存在更高的函数类型——即以其他函数为参数的函数——意味着我们可以定义比通常的原始递归函数更多的内容；例如，参见 Exercise 1.10。）这在构造性数学中是适当的；在 §§3.4 and 3.8 中，我们将看到如何增强类型论，以便定义更一般的数学函数。
现在我们遵循与其他类型相同的方法，将原始递归推广到依赖函数，从而获得归纳原理。因此，假设给定一个类型族 $C : \mathbb{N} \to \mathcal{U}$，一个元素 $c_0 : C(0)$，以及一个函数 $c_s : \prod_{(n:\mathbb{N})} C(n) \to C(\mathsf{succ}(n))$；然后我们可以构造一个函数 $f : \prod_{(n:\mathbb{N})} C(n)$，其定义等式为：

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

我们也可以将其打包为一个函数

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

其定义等式为：

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

在这里，我们终于看到了与经典归纳证明的联系。回想一下，在类型论中，我们用类型表示命题，并通过填充对应的类型来证明命题。特别是，自然数的性质由类型族 $P : \mathbb{N} \to \mathcal{U}$ 表示。从这个角度看，上述归纳原理表明，如果我们可以证明 $P(0)$，并且对于任何 $n$，如果我们假设 $P(n)$ 成立，那么我们可以证明 $P(\mathsf{succ}(n))$，那么我们就有 $P(n)$ 对所有 $n$ 都成立。当然，这正是经典的自然数归纳证明原则。

例如，考虑我们如何表示一个显式证明，即 $+$ 是结合的。（我们实际上不会以这种风格写出证明，但它作为理解归纳在类型论中的形式表示的有用例子。）要推导出

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

我们需要提供

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

和

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

为了推导出 $\mathsf{assoc}_0$，请回想 $0 + n \equiv n$，因此 $0 + (j + k) \equiv j + k \equiv (0 + j) + k$。因此，我们可以简单地设定

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

对于 $\mathsf{assoc}_s$，请回想加法定义给出了 $\mathsf{succ}(m) + n \equiv \mathsf{succ}(m + n)$，因此

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

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”）给出了 $i + (j + k) = (i + j) + k$，所以只需要调用一个事实，即如果两个自然数相等，那么它们的后继也是相等的。（我们将在 Lemma 2.2.1 中使用同一类型的归纳原理证明这一显而易见的事实。）我们称这个事实为 $\mathsf{ap}_{\mathsf{succ}} : (m =_{\mathbb{N}} n) \to (\mathsf{succ}(m) =_{\mathbb{N}} \mathsf{succ}(n))$，因此我们可以定义

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

将这些与 $\mathsf{ind}_{\mathbb{N}}$ 结合起来，我们得到一个结合律的证明。

## 1.10 模式匹配和递归（Pattern matching and recursion）

自然数比之前考虑的类型引入了额外的复杂性。例如，对于并集，我们可以定义一个函数 $f : A + B \to C$，要么使用递归器：

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

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 $f$ 的前者表达式到后者，我们只需使用递归器的计算规则。反之，给定任意定义方程

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

其中 $\Phi_0$ 和 $\Phi_1$ 是可能涉及变量 $a$ 和 $b$ 的表达式，我们可以通过使用 $\lambda$-抽象 将这些方程等价地表示为递归器：

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

然而，在自然数的情况下，像 double 这样的函数的“定义方程”：

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

涉及到函数 double 本身在右侧。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，因为它们更方便和可读。解决方案是将 (1.10.2) 右侧的“$\mathsf{double}(n)$”表达式理解为递归调用的结果，在形式为 $\mathsf{double} :\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N}, c_0, c_s)$ 的定义中，这将是 $c_s$ 的第二个参数。
更一般地说，如果我们有一个形式为 $f : \mathbb{N} \to C$ 的函数的“定义”，比如

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

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

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

其中 $\Phi'_s$ 是通过将“$f(n)$”的所有出现替换为新变量 $r$ 而得到的。
这种通过递归定义函数（或更一般地，通过归纳定义依赖函数）的风格是如此方便，我们经常采用它。这称为**模式匹配**定义。当然，它与计算机程序员可能定义一个递归函数并且函数体实际上包含对自身的递归调用非常相似。然而，与程序员不同的是，我们在可以进行递归调用的类型上受到限制：为了使这种定义能够用递归原则重新表达，正在定义的函数 $f$ 只能在 $f(\mathsf{succ}(n))$ 的函数体中作为复合符号“$f(n)$”的一部分出现。否则，我们可以写出无意义的函数，如

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

如果程序员写了这样一个函数，它只会对任何正输入调用自身永远不停，进入无限循环，永远不返回值。然而，在数学中，为了配得上“函数”这个名字，函数必须始终为每个输入值关联一个唯一的输出值，因此这是不可接受的。
当我们在 Chapters 5, 6 and 11 中引入更复杂的归纳类型时，这一点将变得更加重要。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理。然后我们引入一种适当的“模式匹配”，可以将其证明为归纳原理的简写。

## 1.11 命题作为类型（Propositions as types）

如引言中所述，在类型论中要证明某个命题为真，相当于展示一个对应于该命题的类型的元素。 我们将该类型的元素视为该命题为真的证据或见证。（有时它们甚至被称为证明，但这个术语可能会引

起误解，所以我们一般避免使用它。）然而，通常情况下，我们不会显式地构造见证；相反，我们会以普通数学语言呈现证明，使其可以翻译为某种类型的元素。这与经典集合论中的推理没有什么不同，在经典集合论中，我们也不期望看到使用谓词逻辑规则和集合论公理的明确推导。

然而，类型论对证明的视角在某些重要方面还是有所不同。类型论逻辑的基本原则是，一个命题不仅仅是对还是错，而是可以看作其真理的所有可能见证的集合。在这种观念下，证明不仅仅是数学交流的手段，它们本身就是数学对象，与数字、映射、群等更为熟悉的对象同等重要。因此，由于类型分类了可用的数学对象并规范了它们的交互方式，命题实际上只是特殊的类型——即，其元素为证明的类型。

促使这种认同成为可能的基本观察是，我们在命题的*逻辑*运算（用英语表达）和它们对应的见证类型的*类型论*运算之间有如下自然的对应关系。

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

这个对应关系的要点在于，每种情况下，右边类型的元素构造和使用规则对应于左边命题的推理规则。例如，要证明形式为“$A$ 和 $B$”的陈述的基本方法是证明 $A$ 并同时证明 $B$，而构造 $A \times B$ 的元素的基本方法是构造一个对 $(a, b)$，其中 $a$ 是 $A$ 的一个元素（或见证），$b$ 是 $B$ 的一个元素（或见证）。如果我们想要利用“$A$ 和 $B$”来证明其他东西，我们可以在证明过程中自由使用 $A$ 和 $B$，类似于 $A \times B$ 的归纳原则允许我们通过使用 $A$ 和 $B$ 的元素来构造一个函数。

类似地，证明一个蕴涵“如果 $A$ 则 $B$”的基本方法是假设 $A$ 并证明 $B$，而构造 $A \to B$ 元素的基本方法是给出一个表达式，该表达式表示 $B$ 的一个元素（见证），并且可能涉及一个未指定的 $A$ 类型的元素（见证）。而利用蕴涵“如果 $A$ 则 $B$”的基本方法是，如果我们知道 $A$，则推导出 $B$，类似于我们可以将函数 $f : A \to B$ 应用于 $A$ 的一个元素来生成 $B$ 的一个元素。我们强烈鼓励读者进行练习，验证其他类型构造器的规则是否合理地转化为逻辑。

特别值得注意的是，空类型 $\mathbf{0}$ 对应于虚假。在逻辑上，我们将 $\mathbf{0}$ 的一个元素称为**矛盾**：因此没有办法证明一个矛盾，[7] 而从一个矛盾中可以推导出任何东西。我们还定义类型 $A$ 的**否定** 为

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

因此，$\neg A$ 的见证是一个从 $A \to \mathbf{0}$ 的函数，我们可以通过假设 $x : A$ 并推导出 $\mathbf{0}$ 的一个元素来构造它。 注意，虽然我们得到的逻辑是“构造性的”，如引言中所讨论的，但这种“反证法”（假设 $A$ 并推导出矛盾，从而得出 $\neg A$）在构造性逻辑中是完全有效的：它只是调用了“否定”的*意义*。被禁止的那种“反证法”是，假设 $\neg A$ 并推导出矛盾，以此证明 $A$。在构造性逻辑中，这样的论证只能让我们得出 $\neg\neg A$，读者可以验证，从 $\neg\neg A$（即，从 $(A \to \mathbf{0}) \to \mathbf{0}$）推导出 $A$ 是没有明显的方法的。

上述逻辑连接词向类型构造操作的翻译被称为**命题作为类型**：它为我们提供了一种将用英语书写的命题及其证明翻译为类型及其元素的方法。例如，假设我们要证明以下恒真式（“德摩根律”之一）：

“*如果非 $A$ 且非 $B$，则非 $(A$ 或 $B)$*”。 (1.11.1)

这种事实的普通英语证明可能如下：

> 假设非 $A$ 且非 $B$，并且假设 $A$ 或 $B$；我们将得出一个矛盾。有两种情况。如果 $A$ 成立，那么由于非 $A$，我们有一个矛盾。类似地，如果 $B$ 成立，那么由于非 $B$，我们也有一个矛盾。因此，在任一情况下我们都有一个矛盾，所以非 $(A$ 或 $B)$。

[7] 更准确地说，没有*基本的*方法证明一个矛盾，即 $\mathbf{0}$ 没有构造函数。如果我们的类型论是不一致的，那么将会有一些更复杂的方法来构造一个 $\mathbf{0}$ 的元素。

现在，根据上述规则，对应于我们的恒真式 (1.11.1) 的类型是

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

因此我们应该能够将上述证明翻译为该类型的一个元素。
为了说明这种翻译的工作原理，让我们描述一下，一个数学家在阅读上述英语证明的同时，如何在脑海中构造出 (1.11.2) 的一个元素。介绍性的短语“假设非 $A$ 且非 $B$”翻译为定义一个函数，隐含地应用了其定义域 $(A \to \mathbf{0}) \times (B \to \mathbf{0})$ 的笛卡尔积的递归原理。这引入了未命名的变量（假设）类型为 $A \to \mathbf{0}$ 和 $B \to \mathbf{0}$。在翻译为类型论时，我们必须给这些变量命名；我们称它们为 $x$ 和 $y$。此时，我们可以写出 (1.11.2) 元素的部分定义为

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

其中 $A + B \to \mathbf{0}$ 类型的“空洞”$\square$ 表示剩下的工作。（我们可以等效地写成 $f :\equiv \mathsf{rec}_{(A\to\mathbf{0})\times(B\to\mathbf{0})}(A + B \to \mathbf{0}, \lambda x.\, \lambda y.\, \square)$，使用递归器而不是模式匹配。）下一短语“假设 $A$ 或 $B$；我们将推导出一个矛盾”指示通过函数定义填补这个空洞，介绍另一个未命名的假设 $z : A + B$，导致证明状态为：

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

现在说“有两种情况”表示情况分裂，即应用 $A + B$ 的并集递归原理。如果我们使用递归器写出来，它将是

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

如果我们使用模式匹配写出，它将是

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

请注意，在这两种情况下，我们现在都有两个类型为 $\mathbf{0}$ 的“空洞”需要填补，对应于我们必须推导矛盾的两个情况。最后，从 $a : A$ 和 $x : A \to \mathbf{0}$ 推导矛盾的结论只是函数 $x$ 对 $a$ 的应用，类似地在另一种情况下也是如此。（注意“应用函数”这个短语与“应用假设”或定理这个短语的巧合。）因此，我们的最终定义是

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

作为练习，你应该验证逆命题“如果非 ($A$ 或 $B$)，则（非 $A$）和（非 $B$)”，通过展示一个元素

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

对于任意类型 $A$ 和 $B$，使用我们刚刚介绍的规则。
然而，并非所有的经典恒真式在这种解释下都成立。例如，规则“如果非 ($A$ 和 $B$)，则（非 $A$）或（非 $B$)”在一般情况下不成立：我们通常无法构造出对应类型的一个元素

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

这反映了类型论的“命题作为类型”逻辑是*构造性的*。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，例如这个德摩根律的实例。
从哲学上讲，构造性逻辑之所以被称为“构造性”，是因为它只限于可以*有效地*完成的构造，这意味着具有某种计算意义。不必太精确地说，这意味着有某种算法指定了逐步构建对象的步骤（以及如何看到一个定理为真，作为一个特例）。这要求省略 LEM ，因为没有*有效的*程序来决定命题是真还是假。类型论逻辑的构造性意味着它具有内在的计算意义，这对计算机科学家来说很有趣。它还意味着类型论提供了*公理自由*。例如，虽然默认情况下没有见证 LEM 存在的构造，但该逻辑仍然与其存在相容

（见 §3.4）。因此，由于类型论并不否认 LEM ，我们可以一致地将其作为假设添加，并在不受限制的情况下进行常规操作。在这一点上，类型论丰富了，而不是约束了传统的数学实践。

我们鼓励不熟悉构造性逻辑的读者通过一些更多的例子来熟悉它。参见 Exercises 1.12 and 1.13 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，在这种逻辑中，除了像“和”和“或”这样的逻辑连接词外，我们还有量词“存在”和“全称”。在这种情况下，类型发挥了双重作用：它们既是命题，也作为类型的传统意义——即，我们量化的领域。类型 $A$ 上的谓词表示为族 $P : A \to \mathcal{U}$，将 $A$ 的每个元素 $a : A$ 映射到一个类型 $P(a)$，该类型对应于 $P$ 对 $a$ 成立的命题。我们现在通过对量词的解释来扩展上述翻译：

| 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 $P(x)$ | $\sum_{(x:A)} P(x)$ |

和以前一样，我们可以证明（构造性的）谓词逻辑的恒真式转化为可居住的类型。例如，如果对于所有 $x : A$，$P(x)$ 和 $Q(x)$ 成立，则（对于所有 $x : A$，$P(x)$）和（对于所有 $x : A$，$Q(x)$）可以翻译为

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

这种恒真式的非正式证明可能如下：

> 假设对于所有 $x$，$P(x)$ 和 $Q(x)$ 成立。首先，我们假设给定 $x$ 并证明 $P(x)$。根据假设，我们有 $P(x)$ 和 $Q(x)$，因此我们有 $P(x)$。其次，我们假设给定 $x$ 并证明 $Q(x)$。同样根据假设，我们有 $P(x)$ 和 $Q(x)$，因此我们有 $Q(x)$。

第一句话开始定义一个蕴涵作为函数，通过引入其假设的见证：

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

在这个例子中，使用配对构造器来生成一个积类型的元素有些明显，由“首先”和“其次”这些词有所标记：

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

短语“我们假设给定 $x$ 并证明 $P(x)$”现在表示以通常方式定义一个依赖函数，引入一个变量作为其输入。由于这是在配对构造器内部，因此自然地将其写为 $\lambda$-抽象：

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

现在，“我们有 $P(x)$ 和 $Q(x)$”调用假设，获得 $p(x) : P(x) \times Q(x)$，并且“因此我们有 $P(x)$”隐式地应用了适当的投影：

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

接下来的两句话以显而易见的方式填补了另一个空洞：

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

当然，我们一直使用的英语证明比数学家通常在实践中使用的要啰嗦得多；它们更像是“引论”课程中使用的语言。实习数学家已经学会了填补空白，因此在实践中我们可以省略大量细节，并且我们通常会这样做。然而，证明有效性的标准始终是，它们可以翻译回到构造对应类型的元素。

作为一个更具体的例子，考虑如何定义自然数的不等式。一种自然的定义是，$n \le m$ 如果存在一个 $k : \mathbb{N}$ 使得 $n + k = m$。（这再次使用了我们将在下一节中介绍的同一性类型，但我们实际上不需要关于它们的太多内容。）在“命题作为类型”翻译下，这将会产生：

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

我们邀请读者从这个定义中证明 $\le$ 的熟悉属性。对于严格不等式，有几种自然的选择，如

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

或者

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

在构造数学中，前者更自然，但在这种情况下它实际上等价于后者，因为 $\mathbb{N}$ 具有“可判定等式”（参见 §3.4 and Theorem 7.2.6）。

类型 $\sum_{(x:A)} P(x)$ 也有另一种解释。由于它的一个元素是 $x : A$ 的一个元素以及 $P(x)$ 成立的一个见证，我们可以将 $\sum_{(x:A)} P(x)$ 视为“存在 $x : A$ 使得 $P(x)$”的命题，而不是将其视为 $A$ 的“子类型”。我们将在 §3.5 中回到这种解释。现在我们注意到，它使我们能够将公理纳入 §1.6 中讨论的数学结构类型的定义中。例如，假设我们要定义一个**半群**，即一个类型 $A$，配备一个二元运算 $m : A \to A \to A$（即一个 magma），并且对于所有 $x, y, z : A$ 我们有 $m(x, m(y, z)) = m(m(x, y), z)$。这个命题由类型表示

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

因此，半群的类型为

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

即 Magma 的半群子类型。从 Semigroup 的一个元素中，我们可以通过应用适当的投影提取承载者 $A$、运算 $m$ 及其公理的见证。我们将在 ?? 中回到这个例子。

还要注意，我们可以使用类型论中的宇宙表示“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，我们可以表示命题*对于所有性质* $P : A \to \mathcal{U}$，*如果* $P(a)$ *则* $P(b)$ 为

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

其中 $A : \mathcal{U}$ 且 $a, b : A$。然而，这个命题*先验地*存在于不同的、更高的宇宙中，比我们量化的命题要高：

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

我们将在 §3.5 中回到这个问题。

我们在这里描述了一个“证明相关的”命题的翻译，其中析取和存在陈述的证明携带一些信息。例如，如果我们有一个 $A + B$ 的元素，将其视为“$A$ 或 $B$”的见证，那么我们知道它是来自 $A$ 还是 $B$。同样，如果我们有 $\sum_{(x:A)} P(x)$ 的一个元素，将其视为“存在 $x : A$ 使得 $P(x)$”的见证，那么我们知道元素 $x$ 是什么（它是给定元素的第一个投影）。

由于这种逻辑的证明相关性质，我们可能会遇到“$A$ 当且仅当 $B$”（回想一下，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，很容易验证“$\mathbb{N}$ 当且仅当 $\mathbf{1}$”，然而显然 $\mathbb{N}$ 和 $\mathbf{1}$ 在重要方面有所不同。陈述“$\mathbb{N}$ 当且仅当 $\mathbf{1}$”只告诉我们，当被视为一个纯粹的命题时，类型 $\mathbb{N}$ 表示与 $\mathbf{1}$ 相同的命题（在这种情况下，是真命题）。我们有时会通过说 $A$ 和 $B$ 是**逻辑等价**的

来表达“$A$ 当且仅当 $B$”。这与将在 §2.4 and Chapter 4 中引入的更强的“类型等价”的概念有所不同：尽管 $\mathbb{N}$ 和 $\mathbf{1}$ 逻辑等价，但它们不是等价类型。

在 Chapter 3 中，我们将引入一类称为“纯命题”的类型，对于这些类型，等价性和逻辑等价性是一致的。使用这些类型，我们将引入一种有时适用的逻辑修改，其中析取和存在命题所包含的附加信息将被丢弃。

最后，我们注意到，命题作为类型的对应关系可以反向看待，使我们能够将任何类型 $A$ 视为一个命题，我们通过展示 $A$ 的一个元素来证明它。有时我们会声明该命题为“$A$ 是**可居住的**”。也就是说，当我们说 $A$ 是可居住的时，我们的意思是我们已经给出了一个（特定的）$A$ 的元素，但我们选择不为该元素命名。同样地，若说 $A$ 是*不可居住的*，则相当于给出了 $\neg A$ 的一个元素。特别地，空类型 $\mathbf{0}$ 显然是不可居住的，因为 $\neg\mathbf{0} \equiv (\mathbf{0} \to \mathbf{0})$ 是由 $\mathsf{id}_{\mathbf{0}}$ 居住的。[8]

## 1.12 同一类型（Identity types）

尽管之前的构造可以被视为标准集合论构造的推广，但我们处理同一性的方法似乎是类型论特有的。根据命题即类型的观点，两个同类型元素 $a, b : A$ 相等的 *命题*必须对应某种 *类型*。由于这个命题取决于 $a$ 和 $b$ 是什么，这些 **等同性类型**（equality types）或 **同一类型**（identity types）必须是依赖于 $A$ 的两个副本的类型族。

我们可以将这个类型族写作 $\mathsf{Id}_A : A \to A \to \mathcal{U}$（不要与同一函数 $\mathsf{id}_A$ 混淆），因此 $\mathsf{Id}_A(a, b)$ 是表示 $a$ 和 $b$ 相等命题的类型。一旦我们熟悉了命题即类型的概念，使用标准的等号符号来表示这一点是很方便的；因此“$a = b$”也可以作为 *类型* $\mathsf{Id}_A(a, b)$ 的记号，对应 $a$ 等于 $b$ 的命题。为了清楚起见，我们还可以写作“$a =_A b$”以明确类型 $A$。如果我们有 $a =_A b$ 的元素，我们可以说 $a$ 和 $b$ 是**相等的**（equal），或者如果我们想强调这与在 §1.1 中讨论的判断性相等 $a \equiv b$ 不同，有时可以称为**命题性相等**（propositionally equal）。

正如我们在 §1.11 中所提到的，命题即类型版本的“或”与“存在”可以包含比命题为真的事实更多的信息，类型 $a = b$ 也可以包含更多信息。实际上，这是同伦解释的基石，在这种解释中，我们将 $a = b$ 的见证者视为 $A$ 空间中 $a$ 和 $b$ 之间的*路径*（paths）或*等价*（equivalences）。就像空间中的两个点之间可以有多条路径一样，两个对象相等的见证者也可能不止一个。换句话说，我们可以将 $a = b$ 视为 $a$ 和 $b$ 的*同一性*（identifications）的类型，$a$ 和 $b$ 之间可能有多种不同的同一方式。我们将在 Chapter 2 中回到这种解释；现在我们专注于同一类型的基本规则。与本章中讨论的所有其他类型一样，它将具有形式上完全相同的构造规则、引入规则、消去规则和计算规则。

构造规则表明，给定一个类型 $A : \mathcal{U}$ 和两个元素 $a, b : A$，我们可以在相同的宇宙中构造类型 $(a =_A b) : \mathcal{U}$。构造 $a = b$ 元素的基本方式是知道 $a$ 和 $b$ 是相同的。因此，引入规则是一个依赖函数

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

称为**反射性**（reflexivity），它表明 $A$ 的每个元素都等于其自身（以特定的方式）。我们将 $\mathsf{refl}_a$ 视为 $a$ 点上的 **常量路径**（constant path）。

特别地，这意味着如果 $a$ 和 $b$ 是*判断性*相等的，$a \equiv b$，那么我们也有 $\mathsf{refl}_a : a =_A b$ 的元素。这是类型良好的，因为 $a \equiv b$ 意味着类型 $a =_A b$ 也判断性地等于 $a =_A a$，这是 $\mathsf{refl}_a$ 的类型。

同一类型的归纳原理（即消去规则）是类型论中最微妙的部分之一，并且对同伦解释至关重要。我们首先考虑它的一个重要后果，即“相等的可以替代相等的”原则，如下所述：

**相同者的不可辨识性**（Indiscernibility of identicals）：对于每个类型族

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

存在一个函数

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

[8] 这不应与类型论一致性的声明相混淆，这是一种*元理论*声称，即按照类型论的规则，不可能获得 $\mathbf{0}$ 的元素。

使得

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

这表明每个类型族 $C$ 都尊重等同性，换句话说，将 $C$ 应用于 $A$ 的相等元素也会导致这些类型之间的函数。显示的等式表明，与反射性相关联的函数是恒等函数（我们将看到，通常，函数 $f(x,y,p) : C(x) \to C(y)$ 总是类型的等价）。

相同者的不可辨识性可以被视为同一类型的递归原理，类似于上面给出的布尔值和自然数的递归原理。就像 $\mathsf{rec}_{\mathbb{N}}$ 为某种特定类型 $C$ 给出了一个指定的映射 $\mathbb{N} \to C$，相同者的不可辨识性为 $A$ 上的某些反射的二元关系（即形如 $C(x) \to C(y)$ 的关系）给出了从 $x =_A y$ 到这些关系的指定映射。我们还可以对更一般的反射关系 $C(x,y)$ 形式提出一个更一般的递归原理。但是，为了完全表征同一类型，我们必须将这一递归原理推广为归纳原理，不仅考虑从 $x =_A y$ 出发的映射，还要考虑其上的类型族。换句话说，我们不仅考虑允许相等的替换相等的，还要考虑导致这种相等的证据 $p$。

### 1.12.1 路径归纳（Path induction）

同一类型的归纳原理称为**路径归纳**（path induction），在 Chapter 2 的介绍中将解释的同伦解释中，它可以被视为说明同一类型族由形如 $\mathsf{refl}_x : x = x$ 的元素自由生成。

**路径归纳：** 给定一个类型族

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

和一个函数

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

存在一个函数

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

使得

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

请注意，与乘积、余积、自然数等类型的归纳原理类似，路径归纳允许我们定义指定的函数，这些函数表现出适当的计算行为。就像我们有由 $c_0 : C$ 和 $c_s : \mathbb{N} \to C \to C$ 递归定义的函数 $f : \mathbb{N} \to C$，其满足 $f(0) \equiv c_0$ 和 $f(\mathsf{succ}(n)) \equiv c_s(n, f(n))$，我们有由 $c : \prod_{(x:A)} C(x,x,\mathsf{refl}_x)$ 路径归纳定义的函数 $f : \prod_{(x,y:A)} \prod_{(p:x=_A y)} C(x,y,p)$，其满足 $f(x,x,\mathsf{refl}_x) \equiv c(x)$。

要理解此原理的含义，首先考虑 $C$ 不依赖于 $p$ 的简单情况。然后我们有 $C : A \to A \to \mathcal{U}$，可以将其视为依赖于 $A$ 两个元素的谓词。我们对 $C(x,y)$ 在某对 $x,y : A$ 元素上何时为真感兴趣。在这种情况下，路径归纳的假设表明我们知道 $C(x,x)$ 对所有 $x : A$ 为真，即如果我们在对 $(x,x)$ 进行评估，我们得到了一个真命题——因此 $C$ 是一个反射关系。结论告诉我们当 $x = y$ 时，$C(x,y)$ 成立。这正是上面提到的关于反射关系的一般递归原理。

归纳规则的通用形式允许 $C$ 也依赖于 $p : x = y$ 作为 $x$ 和 $y$ 相等的见证。在前提中，我们不仅将 $x,y$ 替换为 $x,x$，还同时将 $p$ 替换为反射性：要证明某一性质适用于所有 $x,y$ 和连接它们的路径 $p : x = y$，只需考虑 $x,x$ 和路径为 $\mathsf{refl}_x : x = x$ 的情况即可。如果我们仅将类型视为集合，那么这对我们并没有多大意义，但由于 $x$ 和 $y$ 之间可能有多种不同的 $p : x = y$ 识别，所以在考虑 $x =_A y$ 类型上的族时保留这些信息是有意义的。在 Chapter 2 中，我们将看到这一点对同伦解释非常重要。

如果我们将路径归纳打包成一个单一的函数，它的形式是：

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

满足等式

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

函数 $\mathsf{ind}_{=_A}$ 传统上称为 $J$。我们将在 Lemma 2.3.1 中展示不可辨识性是路径归纳的一个实例，并为其赋予一个新名称和记号。

给定一个证据 $p : a = b$，路径归纳要求我们同时用相同的未知元素 $x$ 替换 $a$ 和 $b$；因此为了定义一个类型族 $C$ 的元素，对于 $A$ 的所有相等元素对，只需在对角线定义它即可。然而，在一些证明中，使用等式 $p : a = b$ 将 $b$ 的所有出现替换为 $a$（或反之）更为简单，因为有时对等式中提到的特定元素 $a$ 进行证明比对一般未知的 $x$ 进行证明更容易。这激发了同一类型的第二个归纳原理，该原理表明类型族 $a =_A x$ 是由元素 $\mathsf{refl}_a : a = a$ 生成的。如下所示，这个第二归纳原理与第一个是等价的；只是有时是更方便的表达方式。

**基于路径归纳：** 固定一个元素 $a : A$，并假设给定了一个类型族

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

和一个元素

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

然后我们可以获得一个函数

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

使得

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

在这里，$C(x, p)$ 是一个类型族，其中 $x$ 是 $A$ 的元素，$p$ 是同一类型 $a =_A x$ 的元素，对固定的 $a$ 在 $A$ 中。基于路径归纳原理表明，要定义此族的元素适用于所有 $x$ 和 $p$，只需考虑 $x$ 为 $a$ 且 $p$ 为 $\mathsf{refl}_a : a = a$ 的情况即可。

打包成一个函数，基于路径归纳变为：

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

满足等式

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

下面，我们展示路径归纳和基于路径归纳是等价的。由于这一点，我们有时会松散地将基于路径归纳称为“路径归纳”，依赖读者从证明的形式中推断出是哪一个原理。

Remark 1.12.1. 直观地说，自然数的归纳原理表示每个自然数要么是 0，要么具有形如 $\mathsf{succ}(n)$ 的形式，因此如果我们证明了这些情况（第二种情况带有归纳假设）的某种性质，那么我们就证明了所有自然数的这一性质。同样，$A + B$ 的归纳原理表示每个 $A + B$ 的元素要么是形如 $\mathsf{inl}(a)$，要么是 $\mathsf{inr}(b)$ 的形式，等等。应用此类解读到路径归纳上，我们可以说路径归纳表明每条路径都具有 $\mathsf{refl}_a$ 的形式，因此如果我们证明了反射路径的某个性质，那么我们就证明了所有路径的这一性质。

然而，这种解读在同伦解释路径的上下文中可能会令人困惑，在同伦解释中，$a$ 和 $b$ 之间可能有多种不同的识别方式，因此同一类型中可能有许多不同的元素！如果可能有许多不同的路径，怎么还能有归纳原理断言唯一的路径是反射的呢？

关键的观察是，归纳定义的不是同一*类型*，而是同一*族*。特别地，路径归纳表明类型族 $(x =_A y)$，当 $x, y$ 在 $A$ 中变化时，是由 $\mathsf{refl}_x$ 形式的元素归纳生成的。这意味着为了给定类型族 $C(x, y, p)$ 的元素，对所有族 $(x, y, p)$ 依赖于族 $(x, y, p)$ 的元素的情况，只需考虑 $(x, x, \mathsf{refl}_x)$ 形式的情况即可。在同伦解释中，这表明所有端点都可变的路径空间对应的空间是自由路径空间，并且事实上任何该空间中的点都同伦于某点的常量环。在类型论中同样成立：我们可以通过路径归纳证明 $(x, y, p) =_{\sum_{(x,y:A)}(x=y)} (x, x, \mathsf{refl}_x)$。同样，基于路径归纳表明，对于固定的 $a : A$，当 $y$ 在 $A$ 中变化时，类型族 $(a =_A y)$ 是由 $\mathsf{refl}_a$ 元素归纳生成的。因此，要定义此族的元素对所有 $y$ 和 $p$ 适用，只需考虑 $(a, \mathsf{refl}_a)$ 的情况即可。同伦地，这表明从某个选定点出发的路径空间（在该点的基于路径空间，类型论上是 $\sum_{(y:A)}(a = y)$）是可收

缩到选定点上的常量环的。同样，在类型论中这一事实也是成立的：我们可以通过基于路径归纳证明 $(y,p) =_{\sum_{(y:A)}(a=y)} (a,\mathsf{refl}_a)$。此外，根据 §1.11 中提到的 $\Sigma$ 类型作为子类型的解释，类型 $\sum_{(y:A)}(a=y)$ 可以视为“所有与 $a$ 相等的 $A$ 元素的类型”，这是一种类型论版本的“单例子集”。
无论是路径归纳还是基于路径归纳，都不能提供一种方法来给定具有两个固定端点 $a$ 和 $b$ 的族 $C(p)$ 的元素。特别地，对于依赖于循环的族 $C:(a=_A a)\to\mathcal{U}$，我们不能应用路径归纳并仅考虑 $C(\mathsf{refl}_a)$ 的情况，因此我们不能证明所有循环都是反射性的。因此，归纳定义同一族并不禁止在特定实例中的非反射性路径。换句话说，路径 $p:x=x$ 可能在 $(x=x)$ 中不等于反射性，但是对 $(x,p)$ 和 $(x,\mathsf{refl}_x)$ 作为 $\sum_{(y:A)}(x=y)$ 的元素，它们是相等的。
作为拓扑学的一个例子，考虑一个穿孔圆盘中的环（$\{\,(x,y)\mid 0<x^2+y^2<2\,\}$），它从 $(1,0)$ 开始绕 $(0,0)$ 绕行一圈然后回到 $(1,0)$。如果我们将两个端点都固定在 $(1,0)$，这个环就不能在保持在穿孔圆盘中的情况下缩成常量路径，就像将绳子缠绕在杆子上时，如果我们抓住两端，它就不能被拉进来一样。但是，如果我们允许一个端点变化，这个环就可以缩回到常量路径，就像我们只抓住绳子的一端，总是可以将其收回一样。

### 1.12.2 路径归纳与基于路径归纳的等价性

上面介绍的同一类型的两个归纳原理是等价的。可以很容易地看到，路径归纳可以从基于路径归纳的原理推导出来。事实上，我们假设路径归纳的前提条件：

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

现在，给定一个元素 $x:A$，我们可以实例化上述两个条件，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 $c'$ 符合基于路径归纳的前提条件，因此我们可以构造

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

并满足等式

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

现在我们观察到 $g$ 的值域等于 $C(x,y,p)$。因此，解除我们对 $x:A$ 的假设，我们可以推导出一个函数

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

并具有所需的判断等式 $f(x,x,\mathsf{refl}_x)\equiv g(x,\mathsf{refl}_x):\equiv c':\equiv c(x)$。
这个事实的另一种证明是观察到任何此类 $f$ 都可以作为 $\mathsf{ind}_{=_A}$ 的实例，因此只需将 $\mathsf{ind}_{=_A}$ 定义为 $\mathsf{ind}'_{=_A}$ 的形式：

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

另一种方向略显复杂；尚不清楚如何使用路径归纳的特定实例来推导基于路径归纳的特定实例。相反，我们可以构造一个路径归纳的实例，显示基于路径归纳的所有可能实例。定义

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

然后我们可以构造函数

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

并因此使用路径归纳获得

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

其 $f(x,x,\mathsf{refl}_x) :\equiv d(x)$。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 $a : A$ 以及 $x : A$ 和 $p : a =_A x$，我们可以推导出基于路径归纳的结论：

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

请注意，我们还获得了正确的定义等式。
另一个证明是观察到基于路径归纳的任何使用都是 $\mathsf{ind}'_{=_A}$ 的实例，并定义

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

请注意，上述构造使用了宇宙。如果我们想用 $C : \prod_{(x:A)}(a =_A x) \to \mathcal{U}_i$ 来建模 $\mathsf{ind}'_{=_A}$，我们需要使用 $\mathsf{ind}_{=_A}$，其中

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

因为 $D$ 量化了所有给定类型的 $C$。虽然这与我们对宇宙的定义兼容，但也可以在不使用宇宙的情况下推导出 $\mathsf{ind}'_{=_A}$：我们可以展示 $\mathsf{ind}_{=_A}$ 包含了 Lemmas 2.3.1 and 3.11.8，并且这两个原理直接推导了 $\mathsf{ind}'_{=_A}$。我们将细节留给读者作为 Exercise 1.7。
我们可以使用上述任何形式的同一类型来确定等同性是一种等价关系，每个函数都保留等同性，并且每个族都尊重等同性。我们将细节留到下一章，在同伦类型论的背景下推导和解释。

Remark 1.12.2. 我们强调尽管具有一些不熟悉的特征，但命题性等同性是同伦类型论中的*数学等同性*。这种区别不属于判断性等同性，它是类型论规则的一个元理论特征。例如，在 §1.9 中证明的自然数加法的结合律是*命题性*等同性，而不是判断性的。同样地，交换律（Exercise 1.16）也是如此。甚至非常简单的交换律 $n+1 = 1+n$ 对于一般的 $n$ 来说也不是判断性等同性（尽管对于任何特定的 $n$ 来说它是判断性的，例如 $3+1 \equiv 1+3$，因为两者都通过定义 $+$ 的计算规则判断性地等于 $4$）。我们只能通过使用同一类型来证明这些事实，因为我们只能通过类型的归纳原理而不是判断来应用自然数。

### 1.12.3 不等性（Disequality）

最后，我们也来讨论一下**不等性**（disequality），不等性它是等式的否定：[9]

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

如果 $x \neq y$，我们说 $x$ 和 $y$ **不相等**（unequal） 不相等或者 **不等**（not equal）。与否定一样，不等性在这里的作用比在经典数学! 经典数学中要小。比如，我们不能通过证明两个东西不等来证明它们相等：这将是双重否定律的一个应用，参见 §3.4。

有时，将不等性以一种正面的方式表达是有用的。例如，在 Theorem 11.2.4中，我们将证明一个实数 $x$ 存在逆元当且仅当它与 0 的距离为正，这比 $x \neq 0$ 的要求更强。

类型! 等式 |)

## Notes

这里介绍的类型论是 Martin-Löf 直觉主义类型论（Martin-Löf's intuitionistic type theory） [?, ?, ?, ?]的一个版本，它本身基于并受到 Brouwer [?], Heyting [?], Scott [?], de Bruijn [?], Howard [?], Tait [?, ?], 和 Lawvere [?]的基础性工作的影响。证明! 助手 Martin-Löf 的类型论有三个主要变种，它们分别在 NuPRL [?], Coq [?], 和 Agda [?] 计算机实现的类型论中得以体现。这里介绍的理论在许多方面与这些变体有所不同，其中一些差异对于同伦解释至关重要，而另一些则是技术上的便利或涉及尚未在同伦环境中研究的概念。

类型论! 内涵类型论! 外延内涵类型论外延! 类型论最重要的是，这里描述的类型论源自 Martin-Löf 内涵类型论（intensional version of Martin-Löf's type theory） [?]，而不是外延版本（extensional version） [?]。在外延理论中，判断等式和命题等式没有区别，而在内涵理论中，判断等式被认为是纯定义的，并允许对等式类型进行更广泛、与证明相关的解释，这对于同伦解释至关重要。从同伦的角度来看，外延类型论仅限于同伦离散集（见 §3.1），而内涵理论则允许具有高维结构的类型。NuPRL 系统 [?]是外延的，而 Coq [?] 和 Agda [?] 是内涵的。在内涵类型论中，不同的变体在身份证明的结构上有所不同。我们在此依赖的最自由解释允许对等式进行*与证明相关的*解释，而更严格的变体则施加了*唯一身份证明*（uniqueness of identity proofs, UIP） [?], 唯一! 身份证明即任何两个等式的证明在判断上是相等的，和 Axiom K [?]，公理!Streicher 的 Axiom K 即唯一的等式证明是反身性（up to judgmental equality）。这些额外的要求可以在 Coq 和 Agda 系统中选择性地施加。

在内涵理论之间的另一个变异点是判断等式的强度，特别是关于函数类型的对象。我们在此包括唯一性原则唯一性! 原则（$\eta$-转换）$f \equiv \lambda x.\, f(x)$，作为判断等式的一个原则。例如，该原则用于 §4.9 中，表明一致性蕴含了命题函数扩展性。对于其他类型，有时也会考虑唯一性原则。例如，笛卡尔积 $A \times B$ 的唯一性原则唯一性! 原则! 对于积类型 将是命题等式 $\mathsf{uniq}_{A\times B}$ 的一个判断版本，我们在 §1.5 中构造了它，表示 $u \equiv (\mathsf{pr}_1(u), \mathsf{pr}_2(u))$。这和对于依赖对的相应版本将是合理的选择（我们并没有做出这个选择），但我们不能包括所有这些规则，因为身份类型的唯一性原则会使所有更高的同伦结构变得平凡。因此，我们*不得不*将其排除在外，问题随后变成了在哪些地方划清界限。关于归纳类型，我们在 §5.5中进一步讨论了这些要点。

对于诸如乘积、$\Sigma$-类型、并集、自然数等归纳类型（见 Chapter 5），在归纳和递归的表述上有额外的选择。模式匹配我们将*归纳原则*作为基础，并从中导出*模式匹配*，但也可以采用相反的做法；见 Appendix A。通常在后一种情况下，还允许*深度*模式匹配；见 [?]。我们选择的原因有几个。其中一个原因是，在范畴语义学中，我们自然获得的是归纳原则。另一个原因是，指定允许的（深度）模式匹配类型是相当棘手的；例如，证明! 助手!Agda@Agda 模式匹配可以默认证明 Axiom K，公理!Streicher 的 Axiom K 尽管 –without-K 标志可以阻止这一点 [?]。最后，深度模式匹配对于*更高*的归纳类型（见 Chapter 6）还没有得到充分理解。因此，我们将仅使用 §1.10中描述的那些直接等同于应用归纳原则的模式匹配。

证明! 助手!Coq@Coq 与 Coq 的类型论不同，我们没有包括一个原始的命题类型。相反，如 §1.11所述，我们采纳了*命题即类型*（propositions-as-types, PAT）原则，将命题与类型等同。这最早由 de

[9] 我们使用"不等式（inequality）"来指代 $<$ 和 $\leq$。另外，注意这是 *命题性*（propositional）的等式类型的否定。当然，否定判断等式 $\equiv$ 是没有意义的，因为判断不属于逻辑运算的范畴。

Bruijn [?]、Howard [?]、Tait [?] 和 Martin-Löf [?] 提出。（我们的决定在 §§3.2 and 3.3中得到了更充分的解释。）

不过，我们确实包括了一个完整的累积层次的宇宙（universe），因此类型形成和等式判断变成了宇宙成员关系和等式判断的实例。为了方便起见，我们将宇宙的对象视为类型，而不是类型的代码；在[?]的术语中，这意味着我们使用“Russell 风格的宇宙（Russell-style universes）”而不是“Tarski 风格的宇宙（Tarski-style universes）”。类型! 宇宙!Tarski 风格类型! 宇宙!Russell 风格另一种选择是使用 Tarski 风格的宇宙，要求显式强制函数强制, 宇宙提升将一个宇宙的元素 $A : \mathcal{U}$ 转化为一个类型 $\mathsf{El}(A)$，并在非正式操作时忽略强制。

我们还将宇宙层次视为累积的，即 $\mathcal{U}_i$ 中的每个类型也是 $\mathcal{U}_j$ 中的一个类型，对于每个 $j \geq i$。有多种方式可以形式化地实现累积性：最简单的是包括一个规则，即如果 $A : \mathcal{U}_i$ 那么 $A : \mathcal{U}_j$。然而，这会带来一个恼人的后果，即对于一个类型族 $B : A \to \mathcal{U}_i$ 我们不能得出 $B : A \to \mathcal{U}_j$，尽管我们可以得出 $\lambda a.\, B(a) : A \to \mathcal{U}_j$。更复杂的解决方案是引入一个判断上的子类型关系 $<:$，由 $\mathcal{U}_i <: \mathcal{U}_j$ 生成，但这使得类型论的研究更加复杂。另一种选择是包括一个显式的强制函数 $\uparrow : \mathcal{U}_i \to \mathcal{U}_j$，在非正式操作时可以省略。

也不必要将宇宙按照自然数进行索引并线性排序。对于某些用途，假设每个宇宙都是某个更大宇宙的一个元素，同时具有“定向性”属性，即任何两个宇宙都共同包含在某个更大的宇宙中是更为适当的。还有许多其他可能的变化，例如包括一个包含所有 $\mathcal{U}_i$ 的宇宙“$\mathcal{U}_\omega$”（甚至更高的“巨基数”类型宇宙），或者将层次内部化为类型族 $\lambda i.\, \mathcal{U}_i$。后者实际上在 Agda 中得到了应用。

身份类型的路径归纳原则由 Martin-Löf [?] 提出。在 Martin-Löf 类型论设置中，基于路径的归纳规则由 Paulin-Mohring [?] 提出；它可以看作是 NuPRL [?, Section 8.1]中假设性判断的“逐点功能性（pointwise functionality）”概念的内涵推广。Streicher 使用 Axiom K 证明了 Martin-Löf 的规则蕴含了 Paulin-Mohring 的规则（见 §7.2），Altenkirch 和 Goguen 如 §1.12 中所述进行了证明，最后 Hofmann 在没有宇宙的情况下进行了证明（如 Exercise 1.7）；见 [?, §1.3 and Addendum]。

## Exercises

Exercise 1.1. 给定函数 $f : A \to B$ 和 $g : B \to C$，定义它们的**复合（composite）** 复合! 函数的函数! 复合$g \circ f : A \to C$。复合函数的结合性证明我们有 $h \circ (g \circ f) \equiv (h \circ g) \circ f$。

Exercise 1.2. 仅使用投影推导乘积的递归原则 $\mathsf{rec}_{A \times B}$，并验证定义等式是有效的。对 $\Sigma$ 类型做同样的操作。

Exercise 1.3. 仅使用投影和命题唯一性原则 $\mathsf{uniq}_{A \times B}$，推导乘积的归纳原则 $\mathsf{ind}_{A \times B}$。验证定义等式是有效的。将 $\mathsf{uniq}_{A \times B}$ 推广到 $\Sigma$ 类型，并对 $\Sigma$ 类型做同样的操作。（这需要 Chapter 2 中的概念。）

Exercise 1.4. 迭代器! 自然数的 假设仅给定自然数的迭代器（iterator）

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

其定义方程为

$$\mathsf{iter}(C, c_0, c_s, 0) :\equiv c_0,$$
$$\mathsf{iter}(C, c_0, c_s, \mathsf{succ}(n)) :\equiv c_s(\mathsf{iter}(C, c_0, c_s, n)),$$

推导出具有递归器 $\mathsf{rec}_{\mathbb{N}}$ 类型的函数。使用 $\mathbb{N}$ 的归纳原则证明该函数的定义方程命题上成立。

Exercise 1.5. 类型! 并集证明如果我们定义 $A + B :\equiv \sum_{(x:\mathbf{2})} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)$，那么我们可以给出 $\mathsf{ind}_{A+B}$ 的定义，使得 §1.7 中陈述的定义等式成立。

Exercise 1.6. 类型! 乘积证明如果我们定义 $A \times B :\equiv \prod_{(x:\mathbf{2})} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)$，那么我们可以给出 $\mathsf{ind}_{A \times B}$ 的定义，使得 §1.5 中陈述的定义等式命题上成立（即使用等式类型）。（这需要在 §2.7 中引入的函数扩展性公理（function extensionality axiom）。）

Exercise 1.7. 给出从 $\mathsf{ind}'_{=_A}$ 到 $\mathsf{ind}_{=_A}$ 的另一种推导，避免使用宇宙。（使用后续章节中的概念最为容易。）

Exercise 1.8. 自然数的乘法使用 $\mathsf{rec}_{\mathbb{N}}$ 定义乘法和指数运算。使用仅有的 $\mathsf{ind}_{\mathbb{N}}$ 验证 $(\mathbb{N}, +, 0, \times, 1)$ 是一个半环半环。你可能还需要使用等式的对称性和传递性，Lemmas 2.1.1 and 2.1.2。

Exercise 1.9. 有限! 集合，族的定义 §1.3 结尾处提到的类型族 $\mathsf{Fin} : \mathbb{N} \to \mathcal{U}$，以及在 §1.4 中提到的依赖函数 $\mathsf{fmax} : \prod_{(n:\mathbb{N})} \mathsf{Fin}(n+1)$。

Exercise 1.10. 函数!AckermannAckermann 函数证明 Ackermann 函数 $\mathsf{ack} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ 可以仅使用 $\mathsf{rec}_{\mathbb{N}}$ 定义，并满足以下方程：

$$\begin{aligned} \mathsf{ack}(0, n) &\equiv \mathsf{succ}(n), \\ \mathsf{ack}(\mathsf{succ}(m), 0) &\equiv \mathsf{ack}(m, 1), \\ \mathsf{ack}(\mathsf{succ}(m), \mathsf{succ}(n)) &\equiv \mathsf{ack}(m, \mathsf{ack}(\mathsf{succ}(m), n)). \end{aligned}$$

Exercise 1.11. 证明对于任意类型 $A$，我们有 $\neg\neg\neg A \to \neg A$。

Exercise 1.12. 使用命题即类型的解释，推导出以下恒真式：

(i) 如果 $A$，那么（如果 $B$ 则 $A$）。
(ii) 如果 $A$，则非（非 $A$）。
(iii) 如果（非 $A$ 或非 $B$），则非（$A$ 且 $B$）。

Exercise 1.13. 使用命题即类型的解释，推导出排中律的双重否定，即证明 非（非（$P$ 或非 $P$））。

Exercise 1.14. 为什么身份类型的归纳原则不允许我们构造一个函数 $f : \prod_{(x:A)} \prod_{(p:x=x)} (p = \mathsf{refl}_x)$ 并满足定义方程

$$f(x, \mathsf{refl}_x) :\equiv \mathsf{refl}_{\mathsf{refl}_x} \quad ?$$

Exercise 1.15. 证明从路径归纳推导出不可辨别性。

Exercise 1.16. 证明自然数加法是交换的：$\prod_{(i,j:\mathbb{N})} (i + j = j + i)$。

Chapter 2

# 同伦类型论 (Homotopy Type Theory)

同伦类型论中的一个核心新观点是，可以将类型视为同伦理论中的空间或范畴论中的高维群胚（higher-dimensional groupoids）。

我们首先简要介绍同伦理论与高维范畴论之间的联系。在经典同伦理论中，空间 $X$ 是一组带有拓扑结构的点集合，而点 $x$ 和 $y$ 之间的路径通过一个连续映射 $p : [0,1] \to X$ 来表示，其中 $p(0) = x$ 和 $p(1) = y$。这个函数可以被视为在每一个“时间点”提供 $X$ 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，可以定义路径拼接操作（如果 $p$ 是从 $x$ 到 $y$ 的路径，$q$ 是从 $y$ 到 $z$ 的路径，那么拼接 $p \cdot q$ 是从 $x$ 到 $z$ 的路径）和逆操作（$p^{-1}$ 是从 $y$ 到 $x$ 的路径）。然而，这些操作之间存在自然的等式，但对于严格相等来说并不成立：例如，路径 $p \cdot p^{-1}$（从 $x$ 走到 $y$，然后沿同一路径返回，时间从 0 走到 1）并不严格等于恒等路径（在所有时间点都停留在 $x$ 处）。

解决方法是考虑一种称为**同伦** (homotopy) 的路径相等的更粗略概念。两个连续映射 $f : X_1 \to X_2$ 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 $s \in [0,1]$，$H(s,0) = p(s)$ 和 $H(s,1) = q(s)$。在这种情况下，我们还要求对所有 $t \in [0,1]$，$H(0,t) = x$ 和 $H(1,t) = y$，以使得对每个 $t$，函数 $H(-,t)$ 再次是从 $x$ 到 $y$ 的路径；这种类型的同伦被称为**端点保持** (endpoint-preserving) 或**相对端点** (rel endpoints)。在简单的情况下，我们可以将 $H$ 的方形 $[0,1] \times [0,1]$ 的映像视为“填充”在 $p$ 和 $q$ 之间的空间，尽管对于一般的 $X$ 来说，这并不实际；更好的是将 $H$ 视为将 $p$ 连续变形为 $q$ 的过程，且不移动端点。由于 $[0,1] \times [0,1]$ 是二维的，我们也可以称 $H$ 为路径之间的二维**路径** (2-dimensional path)。

例如，由于 $p \cdot p^{-1}$ 沿同一路径来回走，你知道你可以将 $p \cdot p^{-1}$ 连续缩小到恒等路径——例如，它不会被空间中的一个洞卡住。同伦是一个等价关系，并且诸如拼接、逆操作等操作都尊重同伦。此外，在某个点 $x_0$ 处的闭环的同伦等价类（其中两个闭环 $p$ 和 $q$ 在它们之间存在**基点保持** (based) 同伦时被等同，这是一种满足 $H(0,t) = H(1,t) = x_0$ 的同伦）形成一个称为**基本群** (fundamental group) 的群。这个群是空间的一个**代数不变量** (algebraic invariant)，可以用来调查两个空间是否**同伦等价** (homotopy equivalent)（存在来回连续映射且它们的复合是同伦于恒等映射的），因为等价空间具有同构的基本群。

由于同伦本身是一种二维路径，因此自然存在三维**同伦之间的同伦** (homotopy between homotopies)，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……所组成的无限塔，带有诸如基本群的代数操作，是一种称为（**弱**）**∞-群胚** (weak ∞-groupoid) 的代数结构。∞-群胚包含一组对象，然后是对象之间的**态射** (morphisms)，然后是态射之间的**态射**，依此类推，并带有一些复杂的代数结构；在每一层级上的态射称为 $k$-**态射** ($k$-morphism)。每一层级的态射具有恒等、组合和逆操作，这些操作在某种意义上是弱的，即它们仅满足到下一层级的态射的群胚定律（组合的结合律、恒等是组合的单位元、逆元素消除），而这种弱性导致了进一步的结构。例如，由于组

合的结合律 $p \cdot (q \cdot r) = (p \cdot q) \cdot r$ 本身是一种高维态射，因此需要额外的操作来关联各种结合律的证明：将 $p \cdot (q \cdot (r \cdot s))$ 重新组合成 $((p \cdot q) \cdot r) \cdot s$ 的各种方式产生了 Mac Lane 的五边形。这种弱性还在各个层级之间产生了非平凡的相互作用。

每一个拓扑空间 $X$ 都具有一个**基本 $\infty$-群胚** (fundamental $\infty$-groupoid)，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仅同伦的事实，$(k+1)$-路径充当 $k$-路径之间的同伦。此外，将空间视为 $\infty$-群胚保持了足够的空间方面的特征以进行同伦理论：基本 $\infty$-群胚构造与 $\infty$-群胚作为空间的几何实现是伴随的，并且这种伴随保持了同伦理论（这被称为**同伦假设/定理** (homotopy hypothesis/theorem)，因为它是一个假设还是定理取决于你如何定义 $\infty$-群胚）。例如，你可以很容易地定义一个 $\infty$-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，它将与该空间的经典基本群的定义一致。由于这种对应关系，同伦理论与高维范畴论紧密相关。

现在，在同伦类型论中，每一个类型都可以被视为具有 $\infty$-群胚的结构。回想一下，对于任何类型 $A$，以及任何 $x, y : A$，我们有一个等同性类型 $x =_A y$，也可以写作 $\mathsf{Id}_A(x, y)$ 或简单地写作 $x = y$。从逻辑上讲，我们可以将 $x =_A y$ 的元素视为 $x$ 和 $y$ 相等的证据，或者是 $x$ 与 $y$ 的同一化 (identification)。此外，类型论（不同于一阶逻辑）允许我们将 $x =_A y$ 的元素也视为可以成为进一步命题的对象的个体。因此，我们可以**迭代**等同性类型：我们可以形成同一化 $p, q$ 之间的 $p =_{(x=_A y)} q$ 类型，以及同一化 $r, s$ 之间的 $r =_{(p=_{(x=_A y)} q)} s$ 类型，依此类推。这种等同性类型的层级结构与空间中的连续路径及其之间的（更高阶）同伦，或一个 $\infty$-群胚的结构完全对应。

因此，我们经常将一个元素 $p : x =_A y$ 称为从 $x$ 到 $y$ 的**路径** (path)，称 $x$ 为其**起点** (start point)，称 $y$ 为其**终点** (end point)。具有相同起点和终点的两条路径 $p, q : x =_A y$ 称为**平行** (parallel)，在这种情况下，元素 $r : p =_{(x=_A y)} q$ 可以被视为同伦，或者是态射之间的态射；我们通常称它为**二维路径** (2-path)或**二维路径** (2-dimensional path)。类似地，$r =_{(p=_{(x=_A y)} q)} s$ 是**三维路径** (3-dimensional paths)的类型，用于两个平行二维路径之间的同一化。如果类型 $A$ 是“类集（set-like）”的，如 $\mathbb{N}$，这些迭代的等同性类型将变得无趣（参见 §3.1），但在一般情况下，它们可以建模非平凡的同伦类型。

同伦类型论与经典同伦理论之间的一个重要区别在于，同伦类型论提供了一种**综合的** (synthetic) 描述空间的方式，如下所示。综合几何是欧几里得（Euclid）风格的几何学 [?]：从一些基本概念（点和线）、构造（连接任何两点的线）和公理（所有直角都相等）开始，逻辑地推导出结论。这与解析几何形成对比，其中点和线等概念用 $\mathbb{R}^n$ 中的笛卡尔坐标具体表示——线是点集——并且基本的构造和公理源自这种表示。虽然经典同伦理论是解析性的（空间和路径是由点构成的），同伦类型论是综合性的：点、路径及路径之间的路径是基本的、不可分割的、原始的概念。

此外，同伦类型论的一个令人惊叹之处在于，所有基本的构造和公理——所有更高维的群胚结构——都自动从等同性类型的归纳原理中产生。回想 §1.12 中的内容，该内容表明如果

- 对于每个 $x, y : A$ 以及每个 $p : x =_A y$，我们有一个类型 $D(x, y, p)$，并且
- 对于每个 $a : A$，我们有一个元素 $d(a) : D(a, a, \mathsf{refl}_a)$，

那么

- 对于每个元素 $x, y : A$ 及 $p : x =_A y$，存在一个元素 $\mathsf{ind}_{=_A}(D, d, x, y, p) : D(x, y, p)$，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

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

对于每个 $a : A$。通常，每当我们应用这个归纳规则时，我们要么不关心正在定义的特定函数，要么会立即给它一个不同的名称。

非正式地说，等同性类型的归纳原理表明，如果我们想要构造一个依赖于等同性类型的居住者 $p : x =_A y$ 的对象（或证明一个命题），那么仅在 $x$ 和 $y$ 是相同的（判断上的）且 $p$ 是反身元素 $\mathsf{refl}_x : x = x$（判断上的）时进行构造（或证明）就足够了。在非正式书写时，我们可能会用类似“通过归纳，可以假设……”的短语来表达这一点。这种对“反身情况”的简化类似于在自然数上的归纳证明中的“基例”和“归纳步骤”的简化，也类似于在不交并或析取上的例证证明中的“左例证”和“右例证”的简化。

在自然数上的归纳证明的背景下，“转换规则” (2.0.1) 并不常见，但在递归定义的相关概念中存在类似的概念。如果一个序列 $(a_n)_{n\in\mathbb{N}}$ 是通过给定 $a_0$ 并根据 $a_n$ 指定 $a_{n+1}$ 来定义的，那么实际上生成序列的 $0^{\text{th}}$ 项就是给定的，并且给定的递推关系在生成的序列上成立。（这似乎显而易见，不值得一提，但如果我们将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) 是类比的：它表明，如果我们通过指定当 $p$ 是 $\mathsf{refl}_x : x = x$ 时的值来为所有 $p : x = y$ 定义一个对象 $f(p)$，那么我们指定的值实际上就是 $f(\mathsf{refl}_x)$ 的值。

这个归纳原理赋予每个类型 $\infty$-群胚的结构，并且每个类型之间的函数都具有 $\infty$-函子之间的函子的结构。这从数学的角度来看很有趣，因为它提供了一种处理 $\infty$-群胚的新方式。这从类型理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的新操作。在本章的其余部分，我们将开始探索这个结构。

## 2.1 **类型是高阶群体** (Types are higher groupoids)

我们现在从归纳原理推导出高阶群体结构的开端。我们从等同的对称性开始，在拓扑语言中，这意味着“路径可以反向”。

Lemma 2.1.1. *对于每一个类型 $A$ 和每一个 $x, y : A$，存在一个函数*

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

*记作 $p \mapsto p^{-1}$，使得对每一个 $x : A$，有 ${\mathsf{refl}_x}^{-1} \equiv \mathsf{refl}_x$。我们称 $p^{-1}$ 为 $p$ 的逆元* (inverse)。

因为这是我们第一次声明某个东西为“引理”或“定理”，让我们停下来考虑一下这意味着什么。回想一下，命题（可以被证明的陈述）与类型等同，而引理和定理（已经被证明的陈述）与 *居住的* (inhabited) 类型等同。因此，引理或定理的陈述应被翻译成一个类型，如 §1.11 中那样，其证明被翻译成该类型的一个居住者。根据全称量词“对于每一个”的解释，Lemma 2.1.1 对应的类型是

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

Lemma 2.1.1 的证明将包含构建该类型的一个元素，即推导出某个 $f : \prod_{(A:\mathcal{U})} \prod_{(x,y:A)} (x = y) \to (y = x)$ 的判断。然后我们为这个元素 $f$ 引入符号 $(-)^{-1}$，其中 $A$、$x$ 和 $y$ 的参数被省略并从上下文推断。（如在 §1.1 中所述，次要陈述“对于每一个 $x : A$，有 ${\mathsf{refl}_x}^{-1} \equiv \mathsf{refl}_x$”应该被视为一个单独的判断。）

*第一个证明* (First proof). 假设给定 $A : \mathcal{U}$，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，将任意 $x, y : A$ 和 $p : x = y$ 分配给一个类型，即类型 $y = x$。然后我们有一个元素

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

因此，身份类型的归纳原理给我们提供了一个元素 $\mathsf{ind}_{=_A}(D, d, x, y, p) : (y = x)$ 对于每个 $p : (x = y)$。我们现在可以定义所需的函数 $(-)^{-1}$ 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 ${\mathsf{refl}_x}^{-1} \equiv \mathsf{refl}_x$，如所需。 □

我们以非常形式化的风格写出了这个证明，这在身份类型的归纳规则不熟悉时可能会有所帮助。为了更正式，我们可以说 Lemma 2.1.1 及其证明共同构成了以下判断

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

（以及额外的等同性判断）。然而，最终我们更喜欢使用更自然的语言，如以下等效的证明中那样。

**第二个证明** (Second proof). 我们希望为每个 $x,y:A$ 和 $p:x=y$ 构造一个元素 $p^{-1}:y=x$。通过归纳，我们只需在 $y$ 为 $x$ 并且 $p$ 为 $\mathsf{refl}_x$ 的情况下完成此操作。但在这种情况下，$p$ 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"反射性情况"中，我们可以将 ${\mathsf{refl}_x}^{-1}$ 简单地定义为 $\mathsf{refl}_x$。然后通过归纳原理得出一般情况，并且转换规则 ${\mathsf{refl}_x}^{-1}\equiv\mathsf{refl}_x$ 正是我们在反射性情况中给出的证明。 □

我们将以这两种风格写出接下来的几个证明，以帮助读者熟悉后一种风格。接下来我们证明等同性的传递性，或等价地，我们"连接路径"。

Lemma 2.1.2. *对于每一个类型 $A$ 和每一个 $x,y,z:A$，存在一个函数*

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

*写作 $p\mapsto q\mapsto p\mathbin{\centerdot} q$，使得对于任意 $x:A$，$\mathsf{refl}_x\mathbin{\centerdot}\mathsf{refl}_x\equiv\mathsf{refl}_x$。我们称 $p\mathbin{\centerdot} q$ 为 $p$ 和 $q$ 的* **连接** (concatenation) *或* **复合** (composite)。

注意，我们选择以与函数复合相反的顺序来表示路径连接：从 $p:x=y$ 和 $q:y=z$ 我们得到 $p\mathbin{\centerdot} q:x=z$，而从 $f:A\to B$ 和 $g:B\to C$ 我们得到 $g\circ f:A\to C$（见 Exercise 1.1）。

**第一个证明** (First proof). 所需的函数类型为 $\prod_{(x,y,z:A)}(x=y)\to(y=z)\to(x=z)$。我们将定义一个具有等效类型的函数 $\prod_{(x,y:A)}(x=y)\to\prod_{(z:A)}(y=z)\to(x=z)$，这允许我们进行两次路径归纳。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，为了将身份类型的归纳原理应用于此 $D$，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，即类型为

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

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

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

通过身份类型的归纳原理应用于 $E$，我们得到一个函数

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

但是 $E(x,z,q)\equiv(x=z)$，所以 $d$ 的类型是 (2.1.3)。因此，我们可以使用此函数 $d$ 并将身份类型的归纳原理应用于 $D$，以获得我们所需的函数类型

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

因此 $\prod_{(x,y,z:A)}(y=z)\to(x=y)\to(x=z)$。两个归纳原理的转换规则为任意 $x:A$ 给出了 $\mathsf{refl}_x\mathbin{\centerdot}\mathsf{refl}_x\equiv\mathsf{refl}_x$。 □

**第二个证明** (Second proof). 我们希望为每个 $x, y, z : A$ 和每个 $p : x = y$ 和 $q : y = z$ 构造一个 $x = z$ 的元素。通过对 $p$ 进行归纳，我们只需假设 $y$ 是 $x$ 并且 $p$ 是 $\mathsf{refl}_x$。在这种情况下，$q$ 的类型 $y = z$ 是 $x = z$。现在通过对 $q$ 进行归纳，我们只需再假设 $z$ 是 $x$ 并且 $q$ 是 $\mathsf{refl}_x$。但在这种情况下，$x = z$ 是 $x = x$，我们有 $\mathsf{refl}_x : (x = x)$。 □

读者可能会觉得我们给出的这个引理证明过于复杂。实际上，我们可以在对 $p$ 进行归纳后停止，因为此时我们要生成的是一个等同 $x = z$，而我们已经有这样的等同，即 $q$。为什么我们还要继续对 $q$ 进行另一个归纳?

答案是，如介绍中所述，我们正在做的是 **证明相关的** (proof-relevant) 数学。当我们证明一个引理时，我们是在定义某个类型的一个居住者，并且在证明过程中定义的具体元素可能很重要，而不仅仅是该元素所居住的类型（即引理的 **陈述**）。Lemma 2.1.2 有三个显而易见的证明：我们可以对 $p$ 进行归纳，对 $q$ 进行归纳，或对它们进行双重归纳。如果我们用三种不同的方式证明它，我们将有三个相同类型的不同元素。不难证明这三个元素是相等的（见 Exercise 2.1），但由于它们并非 **定义上**相等，因此可能仍有理由偏爱其中之一。

在 Lemma 2.1.2 的情况下，差异取决于计算规则。如果我们使用单一归纳对 $p$ 证明引理，那么我们将得到形式为 $\mathsf{refl}_y \cdot q \equiv q$ 的计算规则。如果我们使用单一归纳对 $q$ 证明，那么我们将得到 $p \cdot \mathsf{refl}_y \equiv p$，而我们用双重归纳（如我们所做的）证明则仅得到 $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$。

不对称的计算规则有时在做形式化数学时很方便，因为它们允许计算机自动简化更多内容。然而，在非正式数学中，甚至在形式化情况下，拥有一个表现不对称的连接操作并且必须记住哪个侧面是“特殊”的，可能会让人困惑。对两侧进行对称处理可以使证明更具稳健性；这就是为什么我们给出了我们所做的证明。（然而，这无疑是一个风格上的选择。）

下表总结了我们迄今为止所做的“等同性”、“同伦”和“高阶群体”的观点。

| 等同性 (Equality) | 同伦 (Homotopy) | $\infty$-群体 ($\infty$-Groupoid) |
|---|---|---|
| 反射性 | 常量路径 | 恒等态射 |
| 对称性 | 路径反转 | 逆态射 |
| 传递性 | 路径连接 | 态射复合 |

在实践中，传递性通常用于通过一系列中间步骤证明一个等同。我们将使用常见的符号表示，例如 $a = b = c = d$。如果中间表达式较长，或者我们想要指定每个等同的证据，我们可以写

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

在任何一种情况下，该符号表示构造元素 $(p \cdot q) \cdot r : (a = d)$。（为了具体化，我们选择左结合性，尽管考虑到 Lemma 2.1.4(iv)，这几乎没有区别。）如果发生 $b$ 和 $c$ 是判断上相等的情况，那么我们可以写

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

来表示构造 $p \cdot r : (a = d)$。我们还遵循常见的数学实践，不要求该符号中的正当理由（“由 $p$”和“由 $r$”）提供所需的确切证据；相反，我们允许它们简单地提及构造该证据时最重要（或最不明显）的成分。例如，如果“引理 A”声明对于所有 $x$ 和 $y$ 我们有 $f(x) = g(y)$，那么我们可以写“由引理 A”作为步骤 $f(a) = g(b)$ 的正当理由，并相信读者能够推断出我们将引理 A 应用于 $x :\equiv a$ 和 $y :\equiv b$。如果我们相信读者能够猜出正当理由，我们也可以完全省略一个正当理由。

现在，由于证明相关性，我们不能在证明了等同性的“对称性”和“传递性”后停止：我们需要知道这些等同性操作是良好行为的。（这个问题在集合论中是看不见的，因为对称性和传递性只是等同性的 **性质**，而不是路径上的结构。）从同伦论的观点来看，连接和反转只是高阶群体结构的“第一级”——我

们还需要这些操作的协调规律，以及更高维度的类似操作。例如，我们需要知道连接是 结合的 (associative)，并且反转提供了 连接的逆元 (inverses)。

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

(i) $p = p \cdot \mathsf{refl}_y$ 和 $p = \mathsf{refl}_x \cdot p$。
(ii) $p^{-1} \cdot p = \mathsf{refl}_y$ 和 $p \cdot p^{-1} = \mathsf{refl}_x$。
(iii) $(p^{-1})^{-1} = p$。
(iv) $p \cdot (q \cdot r) = (p \cdot q) \cdot r$。

特别注意，(i)–(iv) 本身是命题等同性，存在于身份类型 的身份类型中，例如 $p =_{x=y} q$ 对于 $p,q:x=y$。拓扑上，它们是 路径的路径 (paths of paths)，即同伦。在拓扑学中有一个熟悉的事实，当我们将路径 $p$ 与反转路径 $p^{-1}$ 连接时，我们不会字面上获得一个常量路径（对应于类型论中的等同性 $\mathsf{refl}$）——相反，我们有一个同伦或从 $p \cdot p^{-1}$ 到常量路径的更高路径。

证明 Lemma 2.1.4 (Proof of Lemma 2.1.4). 所有证明都使用身份等同的归纳原理。

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

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

然后 $D(x,x,\mathsf{refl}_x)$ 是 $\mathsf{refl}_x = \mathsf{refl}_x \cdot \mathsf{refl}_x$。因为 $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$，所以 $D(x,x,\mathsf{refl}_x) \equiv (\mathsf{refl}_x = \mathsf{refl}_x)$。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，身份类型的归纳原理为每个 $p:x=y$ 提供了一个元素 $\mathsf{ind}_{=_A}(D,d,x,y,p):(p = p \cdot \mathsf{refl}_y)$。另一个等同性的证明类似。

第二个证明 (Second proof): 通过对 $p$ 进行归纳，只需假设 $y$ 是 $x$ 并且 $p$ 是 $\mathsf{refl}_x$。但在这种情况下，我们有 $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$。

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

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

然后 $D(x,x,\mathsf{refl}_x)$ 是 ${\mathsf{refl}_x}^{-1} \cdot \mathsf{refl}_x = \mathsf{refl}_x$。因为 ${\mathsf{refl}_x}^{-1} \equiv \mathsf{refl}_x$ 并且 $\mathsf{refl}_x \cdot \mathsf{refl}_x \equiv \mathsf{refl}_x$，我们得到 $D(x,x,\mathsf{refl}_x) \equiv (\mathsf{refl}_x = \mathsf{refl}_x)$。因此，我们找到了函数

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

现在，路径归纳为每个 $p:x=y$ 提供了一个元素 $\mathsf{ind}_{=_A}(D,d,x,y,p):p^{-1} \cdot p = \mathsf{refl}_y$。另一个等同性类似。

第二个证明 (Second proof): 通过归纳，只需假设 $p$ 是 $\mathsf{refl}_x$。但在这种情况下，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 (First proof): 令 $D:\prod_{(x,y:A)}(x=y)\to\mathcal{U}$ 为由

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

然后 $D(x,x,\mathsf{refl}_x)$ 是类型 $(({\mathsf{refl}_x}^{-1})^{-1} = \mathsf{refl}_x)$。但因为 ${\mathsf{refl}_x}^{-1} \equiv \mathsf{refl}_x$ 对于每个 $x:A$，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，我们找到了函数

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

现在，路径归纳为每个 $p:x=y$ 提供了一个元素 $\mathsf{ind}_{=_A}(D,d,x,y,p):(p^{-1})^{-1} = p$。

第二个证明 (Second proof): 通过归纳，只需假设 $p$ 是 $\mathsf{refl}_x$。但在这种情况下，我们有 $(p^{-1})^{-1} \equiv ({\mathsf{refl}_x}^{-1})^{-1} \equiv \mathsf{refl}_x$。

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

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

然后 $D_1(x,x,\mathsf{refl}_x)$ 是

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

要构造此类型的元素，令 $D_2 : \prod_{(x,z:A)}(x = z) \to \mathcal{U}$ 为类型族

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

然后 $D_2(x,x,\mathsf{refl}_x)$ 是

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

要构造 此类型的元素，令 $D_3 : \prod_{(x,w:A)}(x = w) \to \mathcal{U}$ 为类型族

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

然后 $D_3(x,x,\mathsf{refl}_x)$ 是

$$\left(\mathsf{refl}_x \cdot (\mathsf{refl}_x \cdot \mathsf{refl}_x) = (\mathsf{refl}_x \cdot \mathsf{refl}_x) \cdot \mathsf{refl}_x\right)$$

它在定义上等同于类型 $(\mathsf{refl}_x = \mathsf{refl}_x)$，因此由 $\mathsf{refl}_{\mathsf{refl}_x}$ 居住。通过三次应用路径归纳规则，我们因此得到了整体所需类型的一个元素。

第二个证明 (Second proof): 通过归纳，只需假设 $p$、$q$ 和 $r$ 都是 $\mathsf{refl}_x$。但在这种情况下，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 $\mathsf{refl}_{\mathsf{refl}_x}$ 居住此类型。 □

Remark 2.1.5. 还有其他方式可以定义这些高阶路径。例如，在 Lemma 2.1.4(iv) 中，我们可能只对一条或两条路径进行归纳，而不是对所有三条路径进行归纳。每种可能性都会产生一个 定义上不同的证明，但它们都会相互等同。这种相互之间的等同性可以再次通过归纳证明，将所涉及的所有路径简化为反身性，然后观察到两个证明都简化为反身性。

根据 Lemma 2.1.4(iv)，我们通常将 $p \cdot q \cdot r$ 写作 $(p \cdot q) \cdot r$，类似地，将 $p \cdot q \cdot r \cdot s$ 写作 $((p \cdot q) \cdot r) \cdot s$ 等。我们出于明确性的目的选择左结合性，但这并没有实际影响。我们通常相信读者能够根据需要插入 Lemma 2.1.4(iv) 的实例，以便重新关联此类表达式。

我们还没有真正完成高阶群体结构：路径 (i)–(iv) 还必须满足它们自己的高阶协调 规律，这些规律本身是高阶路径，依此类推，直到“无穷大”（这可以通过例如球形操作代数的概念来精确化）。然而，对于大多数目的来说，没有必要明确化整个无限维结构。同伦类型论的一个优点是，所有这些结构都可以从身份类型的归纳性质出发证明，因此我们可以根据需要显式化这些结构。

特别是，在本书中我们将不需要明确化所有高阶层次的复杂组合学。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），如前面提到的，我们称之为 2-路径 (2-paths) 或 2 维路径 (2-dimensional paths)，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 (3-paths) 或 3 维路径 (3-dimensional paths)。定义一般的 $n$-维路径 ($n$-dimensional path) 的概念是可能的 （见 Exercise 2.4），但我们将不需要它。

然而，我们将使用一种特别重要且简单的高阶路径情况，即起点和终点相同的情况。在集合论中，命题 $a = a$ 完全没有趣味，但在同伦论中，从一点到自身的路径称为 环路 (loops) 并且包含了许多有趣的高阶结构。因此，给定一个具有点 $a : A$ 的类型 $A$，我们定义其 **环空间** (loop space) $\Omega(A, a)$ 为类型 $a =_A a$。如果点 $a$ 从上下文中可以理解，我们有时可能会简单地写作 $\Omega A$。

由于 $\Omega A$ 的任意两个元素是具有相同起点和终点的路径，它们可以连接；因此我们有一个操作 $\Omega A \times \Omega A \to \Omega A$。更一般地，$A$ 的高阶群体结构赋予 $\Omega A$ 类似的“高阶群体”结构。

考虑 $A$ 的环空间的 环空间 (loop space) 也很有用，这是在点 $a$ 上的单位环的 2 维环空间。它写作 $\Omega^2(A, a)$，在类型论中表示为类型 $\mathsf{refl}_a =_{(a=_A a)} \mathsf{refl}_a$。虽然 $\Omega^2(A, a)$ 作为一个环空间，再次是一个“高阶群体”，但由于其元素是 1 维环上 2 维环，这个环空间现在也有了一些附加结构。

Theorem 2.1.6 (埃克曼–希尔顿 (Eckmann–Hilton)). 第二环空间的组合操作

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

是可交换的：$\alpha \cdot \beta = \beta \cdot \alpha$，对于任意 $\alpha, \beta : \Omega^2(A)$。

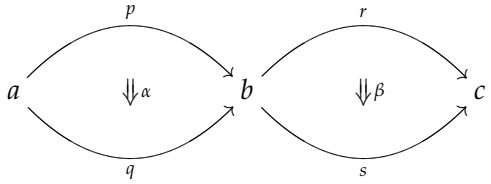
证明. 首先，观察到 1-环的组合 $\Omega A \times \Omega A \to \Omega A$ 引入了一个操作

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

如下：考虑元素 $a, b, c : A$ 和 1 维及 2 维路径，

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

如以下图所示（路径表示为箭头）。




分别连接上下的 1 维路径，我们得到两个路径 $p \cdot r,\ q \cdot s : a = c$，然后在它们之间有一个“水平组合”

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

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

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

其中 $\mathsf{ru}_p : p = p \cdot \mathsf{refl}_b$ 是来自 Lemma 2.1.4(i) 的右单位律。我们可以类似地通过对 $\alpha$ 进行归纳，或者对视线中的所有路径进行归纳来定义 $\cdot_{\mathsf{r}}$，从而产生不同的判断等同，但对于当前目的，通过对 $r$ 进行归纳定义将使事情变得更简单。类似地，我们定义 $q \cdot_{\mathsf{l}} \beta : q \cdot r = q \cdot s$ 通过对 $q$ 进行归纳，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。操作 $\cdot_{\mathsf{l}}$ 和 $\cdot_{\mathsf{r}}$ 被称为 **胡须** (whiskering)。接下来，由于 $\alpha \cdot_{\mathsf{r}} r$ 和 $q \cdot_{\mathsf{l}} \beta$ 是可组合的 2 维路径，我们可以定义 **水平组合** (horizontal composition) 为：

$$\alpha \star \beta \ :\equiv\ (\alpha \cdot_{\mathsf{r}} r) \cdot (q \cdot_{\mathsf{l}} \beta)$$

现在假设 $a \equiv b \equiv c$，因此所有 1 维路径 $p$、$q$、$r$ 和 $s$ 都是 $\Omega(A,a)$ 的元素，并且进一步假设 $p \equiv q \equiv r \equiv s \equiv \mathsf{refl}_a$，因此 $\alpha : \mathsf{refl}_a = \mathsf{refl}_a$ 和 $\beta : \mathsf{refl}_a = \mathsf{refl}_a$ 在两种顺序中都是可组合的。在这种情况下，我们有

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

（回想一下，$\mathsf{ru}_{\mathsf{refl}_a} \equiv \mathsf{lu}_{\mathsf{refl}_a} \equiv \mathsf{refl}_{\mathsf{refl}_a}$，根据路径归纳的计算规则。）另一方面，我们可以通过类似方式定义另一个水平组合

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

并且我们类似地得出

$$\alpha \star' \beta = \beta \cdot \alpha$$

但是，通常情况下，定义水平组合的两种方式是相同的，$\alpha \star \beta = \alpha \star' \beta$，我们可以通过对 $\alpha$ 和 $\beta$ 进行归纳，然后对剩下的两条 1 维路径进行归纳，将一切简化为反身性。因此我们有

$$\alpha \cdot \beta = \alpha \star \beta = \alpha \star' \beta = \beta \cdot \alpha \qquad \square$$

上述事实，被称为 **埃克曼–希尔顿论证** (Eckmann–Hilton argument)，来自经典同伦论，实际上它在下面的 Chapter 8 中被用来证明类型的高阶同伦群总是阿贝尔群。证明中定义的胡须和水平组合操作也是类型的 $\infty$-群体结构的一般部分。它们满足自己的规律（直到更高的同伦），例如

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

等等。从现在开始，我们相信读者在需要时应用路径归纳来定义此类进一步操作并验证其属性。如本例所示，更高路径类型的代数比各个层次上的类似群体结构更为复杂；这些层次相互作用，产生许多进一步的操作和规律，如同伦论中迭代环空间的研究那样。事实上，如同经典同伦论，我们可以做出以下一般定义：

Definition 2.1.7. 一个 **带点类型** (pointed type) $(A,a)$ 是一个类型 $A : \mathcal{U}$，以及一个点 $a : A$，称为它的 **基点** (basepoint)。我们写作 $\mathcal{U}_\bullet :\equiv \sum_{(A:\mathcal{U})} A$ 来表示宇宙 $\mathcal{U}$ 中带点类型的类型。

Definition 2.1.8. 给定一个带点类型 $(A,a)$，我们定义其 **环空间** (loop space) 为以下带点类型：

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

它的一个元素将被称为 $a$ 的 **环路** (loop)。对于 $n : \mathbb{N}$，$n$ 次迭代环空间的 **$n$-次迭代环空间** ($n$-fold iterated loop space) $\Omega^n(A,a)$ 带点类型 $(A,a)$ 递归定义为：

$$\begin{aligned}&\Omega^0(A,a)\\ &\Omega^{n+1}(A,a) :\equiv \Omega^n(\Omega(A,a))\end{aligned}$$

它的一个元素将被称为 **$n$-环路** ($n$-loop) 或 **$n$ 维环路** ($n$-dimensional loop) 在 $a$ 处。

我们将在 Chapters 6 to 8 中回到迭代环空间。

## 2.2 函数是函子 (Functions are functors)

我们现在希望证明函数 $f : A \to B$ 在路径上的行为是函子性的。在传统类型论中，这等价于函数尊重等同性的陈述。从拓扑学的角度来看，这相当于说每个函数都是“连续的”，即保持路径的。

Lemma 2.2.1. 假设 $f : A \to B$ 是一个函数。那么对于任意 $x, y : A$，存在一个操作

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

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

符号 $\mathsf{ap}_f$ 可以理解为 $f$ 对路径的 应用（application），也可以理解为 $f$ 在路径上的 作用（action on paths）。

第一种证明 (First proof). 令 $D : \prod_{(x,y:A)}(x = y) \to \mathcal{U}$ 为定义为

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

的类型族。然后我们有

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

通过路径归纳，我们得到 $\mathsf{ap}_f : \prod_{(x,y:A)}(x = y) \to (f(x) = f(y))$。计算规则意味着对于每个 $x : A$，$\mathsf{ap}_f(\mathsf{refl}_x) \equiv \mathsf{refl}_{f(x)}$。 □

第二种证明 (Second proof). 要定义 $\mathsf{ap}_f(p)$ 对于所有 $p : x = y$，通过归纳，假设 $p$ 是 $\mathsf{refl}_x$ 就足够了。在这种情况下，我们可以定义 $\mathsf{ap}_f(p) :\equiv \mathsf{refl}_{f(x)} : f(x) = f(x)$。 □

我们通常将 $\mathsf{ap}_f(p)$ 简写为 $f(p)$。严格来说，这是模棱两可的，但通常不会引起混淆。它符合范畴论中使用相同符号表示函子对对象和态射应用的常见约定。
我们注意到 $\mathsf{ap}$ 的行为是函子性的，以所有可能的方式体现了这一点。

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. 留给读者证明 (Left to the reader)。 □

与 Lemma 2.1.4 中的等同性一样，Lemma 2.2.2 中的等同性本身也是路径，这些路径满足它们自己的协调律（可以用同样的方法证明），依此类推。

## 2.3 类型族是纤维化 (Type families are fibrations)

由于 依赖类型函数在类型论中是必不可少的，我们还需要 Lemma 2.2.1 的一个版本。然而，这并不容易表述，因为如果 $f : \prod_{(x:A)} B(x)$ 且 $p : x = y$，那么 $f(x) : B(x)$ 和 $f(y) : B(y)$ 是属于不同类型的元素，因此 先验我们甚至不能询问它们是否相等。缺少的成分是 $p$ 本身为我们提供了一种将类型 $B(x)$ 和 $B(y)$ 关联起来的方法。
我们已经在 section 1.12 中看到了这一点，我们称之为“相同者的不可分辨性 (indiscernibility of identicals)”。现在我们引入一个不同的名称和符号，之后我们将使用这个名称和符号。

Lemma 2.3.1 (传递 (Transport)). 假设 $P$ 是一个在 $A$ 上的类型族，并且 $p : x =_A y$。那么存在一个函数 $p_* : P(x) \to P(y)$。

*第一种证明* (First proof). 令 $D : \prod_{(x,y:A)}(x = y) \to \mathcal{U}$ 为定义为

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

的类型族。然后我们有函数

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

因此归纳原理给我们 $p : x = y$ 时的 $\mathsf{ind}_{=_A}(D, d, x, y, p) : P(x) \to P(y)$，我们将其定义为 $p_*$。 □

*第二种证明* (Second proof). 通过归纳，假设 $p$ 是 $\mathsf{refl}_x$ 就足够了。但在这种情况下，我们可以将 $(\mathsf{refl}_x)_* : P(x) \to P(x)$ 视为恒等函数。 □

有时，有必要标注传递操作发生的类型族 $P$。在这种情况下，我们可以写作

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

回想一下，一个在类型 $A$ 上的类型族 $P$ 可以看作是 $A$ 的元素的一个性质，如果 $P(x)$ 是可居住的，则在 $A$ 中的 $x$ 处成立。然后传递引理表明 $P$ 尊重等同性，在这种意义上，如果 $x$ 等于 $y$，那么 $P(x)$ 成立当且仅当 $P(y)$ 成立。实际上，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*。

从拓扑学的角度来看，传递引理可以看作是纤维化中的“路径提升 (path lifting)”操作。我们将类型族 $P : A \to \mathcal{U}$ 看作是一个基空间为 $A$ 的*纤维化*，其中 $P(x)$ 是 $x$ 上的纤维，并且 $\sum_{(x:A)} P(x)$ 是纤维化的 **总空间** (total space)，第一投影为 $\sum_{(x:A)}(P(x)) \to A$。纤维化的定义属性是，在基空间 $A$ 中给定一条路径 $p : x = y$ 和纤维 $P(x)$ 上的一个点 $u$，我们可以将路径 $p$ 提升到总空间中以 $u$ 为起点的路径（并且这种提升可以连续地完成）。可以认为点 $p_*(u)$ 是这个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 (路径提升属性 (Path lifting property)). *令 $P : A \to \mathcal{U}$ 为一个在 $A$ 上的类型族，并假设我们有 $u : P(x)$ 对于某个 $x : A$。那么对于任意 $p : x = y$，我们有*

$$\mathsf{lift}(u, p) : (x, u) = (y, p_*(u))$$

*在 $\sum_{(x:A)} P(x)$ 中，使得 $\mathsf{pr}_1(\mathsf{lift}(u, p)) = p$。*

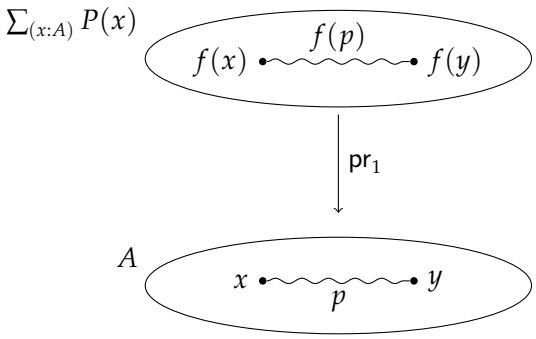
*证明.* 留给读者证明 (Left to the reader)。我们将在 ?? 中证明一个更一般的定理。 □

在经典的同伦论中，纤维化被定义为一个映射，其中*存在*路径的提升；而相对地，我们刚刚展示了在类型论中，每个类型族都带有一个*指定的*“路径提升函数”。这符合构造性数学的哲学，根据该哲学，除非展示出它，否则我们不能表明某事存在。它还自动确保路径提升是“连续地”选择的，因为正如我们所见，在类型论中所有函数都是“连续的”。

Remark 2.3.3. 尽管我们可能将类型族 $P : A \to \mathcal{U}$ 看作类似纤维化，但通常不建议说“纤维化 $P : A \to \mathcal{U}$”，因为这听起来像是在说基底是 $\mathcal{U}$ 而总空间是 $A$ 的纤维化。再次强调，当一个类型族 $P : A \to \mathcal{U}$ 被看作纤维化时，基底是 $A$，而总空间是 $\sum_{(x:A)} P(x)$。

我们也可能偶尔使用其他拓扑学术语来谈论类型族。例如，我们可能将依赖函数 $f : \prod_{(x:A)} P(x)$ 称为纤维化 $P$ 的 **截面** (section)，并且我们可能会说某事 **在纤维上** (fiberwise) 发生，如果它发生在每个 $P(x)$ 上。例如，截面 $f : \prod_{(x:A)} P(x)$ 表明 $P$ 是“在纤维上可居住的”。

现在我们可以证明 Lemma 2.2.1 的依赖版本。拓扑学直觉是，假设 $f : \prod_{(x:A)} P(x)$ 和路径 $p : x =_A y$，我们应该能够将 $f$ 应用于 $p$ 并在 $P$ 的总空间中获得一条“覆盖”$p$ 的路径，如下图所示。




我们可以从 Lemma 2.2.1 中获得这样的结果。给定 $f : \prod_{(x:A)} P(x)$，我们可以通过设置 $f'(x) :\equiv (x, f(x))$ 来定义一个非依赖函数 $f' : A \to \sum_{(x:A)} P(x)$，然后考虑 $f'(p) : f'(x) = f'(y)$。由于 $\mathsf{pr}_1 \circ f' \equiv \mathsf{id}_A$，通过 Lemma 2.2.2 我们有 $\mathsf{pr}_1(f'(p)) = p$；因此 $f'(p)$ 确实在这个意义上"覆盖"$p$。然而，从类型的角度来看，$f'(p)$ 并不明显覆盖 $A$ 中的任何特定路径（在本例中为 $p$），这在某些情况下可能很重要。

解决方案是使用传递引理。根据 Lemma 2.3.2 我们有一个从 $(x, u)$ 到 $(y, p_*(u))$ 的规范路径 $\mathsf{lift}(u, p)$，它覆盖 $p$。因此，任何从 $u : P(x)$ 到 $v : P(y)$ 的路径覆盖 $p$ 应该通过 $\mathsf{lift}(u, p)$ 以基本唯一的方式在 $P(y)$ 的纤维中完全提升。因此，直到等价性，定义"从 $u$ 到 $v$ 的路径覆盖 $p : x = y$"意义上意味着 $P(y)$ 中的路径 $p_*(u) = v$ 是合理的。而且，事实上，我们可以证明依赖函数会产生这样的路径。

Lemma 2.3.4 (依赖映射 (Dependent map)). 假设 $f : \prod_{(x:A)} P(x)$；那么我们有一个映射

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

第一种证明 (First proof). 令 $D : \prod_{(x,y:A)} (x = y) \to \mathcal{U}$ 为定义为

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

的类型族。然后 $D(x, x, \mathsf{refl}_x)$ 是 $(\mathsf{refl}_x)_*(f(x)) = f(x)$。但由于 $(\mathsf{refl}_x)_*(f(x)) \equiv f(x)$，我们得到 $D(x, x, \mathsf{refl}_x) \equiv (f(x) = f(x))$。因此，我们找到函数

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

现在路径归纳为每个 $p : x = y$ 给我们 $\mathsf{apd}_f(p) : p_*(f(x)) = f(y)$。 □

第二种证明 (Second proof). 通过归纳，假设 $p$ 是 $\mathsf{refl}_x$ 就足够了。但在这种情况下，所需的等式是 $(\mathsf{refl}_x)_*(f(x)) = f(x)$，这在判断上是成立的。 □

我们通常将这种意义上"覆盖其他路径"的路径称为依赖路径 (dependent paths)。它们将在 Chapter 6 中起到越来越重要的作用。在 §2.5 中我们将看到，对于一些特定类型的类型族，有等价的方法来表示依赖路径的概念，这在某些情况下更为方便。

现在回想一下，在 §1.4 中，一个非依赖类型的函数 $f : A \to B$ 只是当 $P$ 是常量类型族 $P(x) :\equiv B$ 时的依赖类型函数 $f : \prod_{(x:A)} P(x)$ 的特例。在这种情况下，$\mathsf{apd}_f$ 和 $\mathsf{ap}_f$ 是密切相关的，因为有以下引理：

Lemma 2.3.5. 如果 $P : A \to \mathcal{U}$ 定义为 $P(x) :\equiv B$ 对于一个固定的 $B : \mathcal{U}$，那么对于任意 $x, y : A$ 和 $p : x = y$ 以及 $b : B$ 我们有路径

$$\mathsf{transportconst}^B_p(b) : \mathsf{transport}^P(p, b) = b$$

**第一种证明** (First proof). 固定一个 $b : B$，令 $D : \prod_{(x,y:A)} (x = y) \to \mathcal{U}$ 为定义为

$$D(x, y, p) :\equiv (\mathsf{transport}^P(p, b) = b)$$

的类型族。然后 $D(x, x, \mathsf{refl}_x)$ 是 $(\mathsf{transport}^P(\mathsf{refl}_x, b) = b)$，根据传递的计算规则，这在判断上等于 $(b = b)$。因此，我们有函数

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

现在路径归纳给我们一个 $\prod_{(x,y:A)} \prod_{(p:x=y)} (\mathsf{transport}^P(p, b) = b)$ 的元素，正如我们所希望的那样。 □

**第二种证明** (Second proof). 通过归纳，假设 $y$ 是 $x$ 并且 $p$ 是 $\mathsf{refl}_x$ 就足够了。但是 $\mathsf{transport}^P(\mathsf{refl}_x, b) \equiv b$，因此在这种情况下，我们要证明的是 $b = b$，并且我们有 $\mathsf{refl}_b$ 可以证明。 □

因此，对于任意 $x, y : A$ 和 $p : x = y$ 以及 $f : A \to B$，通过分别与 $\mathsf{transportconst}^B_p(f(x))$ 及其逆连接，我们得到函数

$$\big(f(x) = f(y)\big) \to \big(p_*(f(x)) = f(y)\big) \qquad \text{和 (and)} \tag{2.3.6}$$

$$\big(p_*(f(x)) = f(y)\big) \to \big(f(x) = f(y)\big) \tag{2.3.7}$$

实际上，这些函数是逆等价的（将在 §2.4 中介绍），并且它们将 $\mathsf{ap}_f(p)$ 与 $\mathsf{apd}_f(p)$ 关联起来。

Lemma 2.3.8. *对于 $f : A \to B$ 和 $p : x =_A y$，我们有*

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

**第一种证明** (First proof). 令 $D : \prod_{(x,y:A)} (x = y) \to \mathcal{U}$ 为定义为

$$D(x, y, p) :\equiv \big(\mathsf{apd}_f(p) = \mathsf{transportconst}^B_p(f(x)) \cdot \mathsf{ap}_f(p)\big)$$

的类型族。因此，我们有

$$D(x, x, \mathsf{refl}_x) \equiv \big(\mathsf{apd}_f(\mathsf{refl}_x) = \mathsf{transportconst}^B_{\mathsf{refl}_x}(f(x)) \cdot \mathsf{ap}_f(\mathsf{refl}_x)\big)$$

但根据定义，这三个路径都是 $\mathsf{refl}_{f(x)}$，所以我们有

$$\mathsf{refl}_{\mathsf{refl}_{f(x)}} : D(x, x, \mathsf{refl}_x)$$

因此，路径归纳为我们提供了 $\prod_{(x,y:A)} \prod_{(p:x=y)} D(x, y, p)$ 的元素，这正是我们想要的。 □

**第二种证明** (Second proof). 通过归纳，假设 $y$ 是 $x$ 并且 $p$ 是 $\mathsf{refl}_x$ 就足够了。在这种情况下，我们要证明的是 $\mathsf{refl}_{f(x)} = \mathsf{refl}_{f(x)} \cdot \mathsf{refl}_{f(x)}$，这是在判断上成立的。 □

由于 $\mathsf{apd}_f$ 和 $\mathsf{ap}_f$ 的类型不同，通常使用不同的符号来表示它们会更清晰。
此时，我们希望读者开始熟悉身份类型的归纳证明。从现在起，我们不再给出两种证明风格，而是允许我们使用最清晰和方便的证明（通常是第二种，更简洁的证明）。以下是一些关于传递的有用引理；我们将其证明（采用任一风格）留给读者。

Lemma 2.3.9. *给定 $P : A \to \mathcal{U}$，其中 $p : x =_A y$ 和 $q : y =_A z$ 以及 $u : P(x)$，我们有*

$$q_*(p_*(u)) = (p \cdot q)_*(u)$$

Lemma 2.3.10. *对于函数 $f : A \to B$ 和一个类型族 $P : B \to \mathcal{U}$，以及任意 $p : x =_A y$ 和 $u : P(f(x))$，我们有*

$$\mathsf{transport}^{P \circ f}(p, u) = \mathsf{transport}^P(\mathsf{ap}_f(p), u)$$

Lemma 2.3.11. *对于 $P, Q : A \to \mathcal{U}$ 和一族函数 $f : \prod_{(x:A)} P(x) \to Q(x)$，以及任意 $p : x =_A y$ 和 $u : P(x)$，我们有*

$$\mathsf{transport}^Q(p, f_x(u)) = f_y(\mathsf{transport}^P(p, u))$$

## 2.4 同伦与等价 (Homotopies and equivalences)

到目前为止，我们已经看到，恒等类型 $x =_A y$ 可以被视为类型 $A$ 的两个元素 $x$ 和 $y$ 之间的 同一性 (identifications)、路径 (paths) 或 等价 (equivalences) 的一种类型。现在，我们探讨在 函数 (functions) 和 类型 (types) 之间 "同一性" 或 "相同性" 的适当概念。在 §2.7 and ?? 中，我们将看到，同伦类型论 (Homotopy Type Theory) 允许我们将这些与恒等类型的实例联系起来，但在此之前，我们需要独立理解它们。

传统上，如果两个函数在所有输入上都取相同的值，我们就认为它们是相同的。根据命题即类型 (propositions-as-types) 的解释，这表明两个函数 $f$ 和 $g$（可能是依赖类型 (dependently typed) 的函数）是相同的，如果类型 $\prod_{(x:A)}(f(x) = g(x))$ 是可居住的 (inhabited)。根据同伦的解释，这种依赖函数类型由 连续 (continuous) 路径或 函子等价 (functorial equivalences) 组成，因此可以被视为 同伦 (homotopies) 或 自然同构 (natural isomorphisms) 的类型。我们将采用拓扑学术语来表示这一点。

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 (sections)。从 $f$ 到 $g$ 的 **同伦** (homotopy) 是一个依赖函数，类型为

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

请注意，同伦 (homotopy) 不等同于恒等 $(f = g)$。然而，在 §2.7 中，我们将引入一个公理，使得同伦和恒等 "等价"。

以下证明留给读者。

Lemma 2.4.2. 对于每个依赖函数类型 $\prod_{(x:A)} P(x)$，同伦 (homotopy) 是一个等价关系 (equivalence relation)。也就是说，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 (type theory) 中的函数自动是 "函子 (functors)" 一样，同伦 (homotopies) 也自动是 "自然变换 (natural transformations)"。我们将在此只对非依赖函数 $f, g : A \to B$ 进行说明和证明；在 Exercise 2.18 中，我们要求读者将其推广到依赖函数。

回顾一下，对于 $f : A \to B$ 和 $p : x =_A y$，我们可以写 $f(p)$ 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，并且 $p : x =_A y$。那么我们有

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

我们也可以将其绘制为一个交换图 (commutative diagram)：

$$\begin{array}{ccc} f(x) & \xlongequal{f(p)} & f(y) \\ {\scriptstyle H(x)}\Big\| & & \Big\|{\scriptstyle H(y)} \\ g(x) & \xlongequal[g(p)]{} & g(y) \end{array}$$

证明. 通过归纳，我们可以假设 $p$ 是 $\mathsf{refl}_x$。由于 $\mathsf{ap}_f$ 和 $\mathsf{ap}_g$ 在反射性上计算，因此在这种情况下我们必须证明

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

但这可以通过两边都等于 $H(x)$ 来证明。 □

Corollary 2.4.4. 设 $H : f \sim \mathsf{id}_A$ 是一个同伦，且 $f : A \to A$。那么对于任意 $x : A$，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. 根据 $H$ 的自然性，以下路径的图表是交换的：

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

即 $f(Hx) \cdot Hx = H(fx) \cdot Hx$。我们现在可以通过 $(Hx)^{-1}$ 来操作以消去 $Hx$，得到

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

如所需 (其中一些结合性路径被忽略)。 □

当然，像函数的函子性 (Lemma 2.2.2) 一样，Lemma 2.4.3 中的等式是一个路径，它满足自己的相干性定律 (coherence laws) 等等。

接下来是类型，从传统的角度来看，可以说一个函数 $f : A \to B$ 是一个 同构 (isomorphism)，如果存在一个函数 $g : B \to A$，使得两个复合 $f \circ g$ 和 $g \circ f$ 在点上等同于恒等函数 (identity function)，即使得 $f \circ g \sim \mathsf{id}_B$ 并且 $g \circ f \sim \mathsf{id}_A$。从同伦的角度来看，这应该被称为 同伦等价 (homotopy equivalence)，而从范畴论 (categorical) 的角度来看，这应该被称为 高阶群 (higher groupoids) 的等价 (equivalence)。然而，在进行相关证明数学 (proof-relevant mathematics) 时，对应的类型

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

表现不佳。例如，对于单个函数 $f : A \to B$，可能存在多个不同的 (2.4.5) 的不等价成员。(这与高阶范畴理论 (higher category theory) 中的观察密切相关，即通常需要考虑 伴随等价 (adjoint equivalences) 而不是普通的等价性)。因此，我们给 (2.4.5) 一个历史上准确的，但稍微贬义的名称。

Definition 2.4.6. 对于一个函数 $f : A \to B$，$f$ 的 **拟逆** (quasi-inverse) 是一个三元组 $(g, \alpha, \beta)$，由一个函数 $g : B \to A$ 和同伦 $\alpha : f \circ g \sim \mathsf{id}_B$ 以及 $\beta : g \circ f \sim \mathsf{id}_A$ 组成。

因此，(2.4.5) 是 拟逆 (quasi-inverses) $f$ 的类型；我们可以用 $\mathsf{qinv}(f)$ 来表示它。

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，以及同伦 $\alpha(y) :\equiv \mathsf{refl}_y$ 和 $\beta(x) :\equiv \mathsf{refl}_x$。

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

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

具有拟逆 $(p^{-1} \cdot -)$ 和 $(- \cdot p^{-1})$；见 Exercise 2.6。

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

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

具有由 $\mathsf{transport}^P(p^{-1}, -)$ 给出的拟逆；这可以从 Lemma 2.3.9 中得到。

一般来说，我们只会在 $A$ 和 $B$ "表现得像集合" 的特殊情况下使用 同构 (isomorphism)（以及类似的词汇如 双射 (bijection)，以及关联的符号 $A \cong B$）。在这种情况下，类型 (2.4.5) 是没有问题的。我们将保留 等价 (equivalence) 一词用于改进的概念 $\mathsf{isequiv}(f)$，具有以下性质：

(i) 对于每个 $f : A \to B$，存在一个函数 $\mathsf{qinv}(f) \to \mathsf{isequiv}(f)$。
(ii) 同样地，对于每个 $f$ 我们有 $\mathsf{isequiv}(f) \to \mathsf{qinv}(f)$；因此这两者在逻辑上是等价的 (见 §1.11)。
(iii) 对于 $\mathsf{isequiv}(f)$ 的任何两个成员 $e_1, e_2$，我们有 $e_1 = e_2$。

在 Chapter 4 中，我们将看到 $\mathsf{isequiv}(f)$ 有许多不同的定义，这些定义都满足这三个属性，但它们都是等价的。现在，为了说服读者这样的东西是存在的，我们仅提及最简单的定义：

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

我们现在可以为该定义展示 (i) 和 (ii)。函数 $\mathsf{qinv}(f) \to \mathsf{isequiv}(f)$ 可以通过将 $(g, \alpha, \beta)$ 变为 $(g, \alpha, g, \beta)$ 来容易地定义。在另一种情况下，给定 $(g, \alpha, h, \beta)$，令 $\gamma$ 是组合的同伦

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

意味着 $\gamma(x) :\equiv \beta(g(x))^{-1} \cdot h(\alpha(x))\beta' : g \circ f \sim \mathsf{id}_A$ 由 $\beta'(x) :\equiv \gamma(f(x)) \cdot \beta(x)$ 给出。然后 $(g, \alpha, \beta') : \mathsf{qinv}(f)$。
对于这个定义，性质 (iii) 也不难证明，但它需要识别笛卡尔积 (cartesian products) 和依赖对类型 (dependent pair types) 的恒等类型，我们将在 §2.6 and ?? 中讨论它。因此，我们也将其推迟；见 §4.3。此时，主要需要记住的是存在一个良好定义的类型，我们可以将其称为 "$f$ 是等价"，并且我们可以通过提供一个拟逆来证明一个函数是等价的。实际上，这是证明函数是等价的最常见方法。
按照相关证明数学的理念，从 $A$ 到 $B$ 的等价被定义为一个函数 $f : A \to B$，连同 $\mathsf{isequiv}(f)$ 的一个成员，即一个它是等价的证明。我们用 $(A \simeq B)$ 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在函数上是相等的（即 $A \to B$ 的基础元素是相等的），那么它们作为等价也是相等的（见 ??）。因此，我们经常滥用符号，模糊等价及其基础函数之间的区别。例如，如果我们有一个函数 $f : A \to B$ 并且我们知道 $e : \mathsf{isequiv}(f)$，我们可以写 $f : A \simeq B$，而不是 $(f, e)$。或者相反，如果我们有一个等价 $g : A \simeq B$，我们可以在给定 $a : A$ 时写 $g(a)$，而不是 $(\mathsf{pr}_1 g)(a)$。
我们以观察为结尾：

Lemma 2.4.12. *类型等价是 $\mathcal{U}$ 上的一个等价关系。更具体地说：*

(i) *对于任意 $A$，恒等函数 $\mathsf{id}_A$ 是一个等价函数；因此 $A \simeq A$。*
(ii) *对于任意 $f : A \simeq B$，我们有一个等价 $f^{-1} : B \simeq A$。*
(iii) *对于任意 $f : A \simeq B$ 和 $g : B \simeq C$，我们有 $g \circ f : A \simeq C$。*

*证明.* 显然，恒等函数是其自身的拟逆；因此它是一个等价函数。
如果 $f : A \to B$ 是一个等价函数，那么它有一个拟逆，比如 $f^{-1} : B \to A$。那么 $f$ 也是 $f^{-1}$ 的拟逆，因此 $f^{-1}$ 是 $B \to A$ 的等价。
最后，给定 $f : A \simeq B$ 和 $g : B \simeq C$，它们分别具有拟逆 $f^{-1}$ 和 $g^{-1}$，则对于任意 $a : A$，我们有 $f^{-1}g^{-1}gfa = f^{-1}fa = a$，对于任意 $c : C$，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，因此后者是一个等价。 □

## 2.5 类型构造器的高阶群结构 (The higher groupoid structure of type formers)

在 Chapter 1 中，我们介绍了许多形成新类型的方法：笛卡尔积 (cartesian products)、不相交并 (disjoint unions)、依赖积 (dependent products)、依赖和 (dependent sums) 等等。在 §§2.1–2.3 中，

我们看到在同伦类型论 (Homotopy Type Theory) 中，*所有*类型都表现得像空间或高阶群 (higher groupoids)。我们在本章的剩余部分中的目标是明确这种高阶结构在 Chapter 1 中定义的特定类型中的表现方式。

事实证明，对于许多类型 $A$，等价类型 $x =_A y$ 可以用构造 $A$ 时使用的数据来进行等价的刻画。例如，如果 $A$ 是一个笛卡尔积 $B \times C$，并且 $x \equiv (b,c)$ 而 $y \equiv (b',c')$，那么我们有一个等价

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

用更传统的语言来说，当且仅当它们的分量相等时，两个有序对是相等的（但等价 (2.5.1) 表达的内容要更多）。恒等类型的高阶结构也可以通过这些等价来表达；例如，连接两个对之间的等式对应于分量的连接。

类似地，当一个类型族 $P : A \to \mathcal{U}$ 通过 Chapter 1 中的类型形成规则在 fiberwise 中构建时，操作 $\mathsf{transport}^P(p,-)$ 可以用传入 $P$ 的数据上对应的操作来进行同伦的刻画。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，类型形成规则也是函子的，如果一个函数 $f$ 是从这种函子性中构建的，那么操作 $\mathsf{ap}_f$ 和 $\mathsf{apd}_f$ 可以基于进入 $f$ 的数据上的对应操作来计算。例如，如果 $g : B \to B'$ 和 $h : C \to C'$ 并且我们定义 $f : B \times C \to B' \times C'$ 通过 $f(b,c) :\equiv (g(b),h(c))$，那么根据等价 (2.5.1)，我们可以将 $\mathsf{ap}_f$ 识别为 "$(\mathsf{ap}_g, \mathsf{ap}_h)$"。

接下来的几个部分 (2.6–??) 将致力于对所有基本类型形成规则给出这样的定理，并为每个基本类型构造器 (type former) 提供一个部分。在这里，我们遇到了一些当前可用类型理论中的明显缺陷；正如在后面的章节中将变得更加清楚，如果这些恒等类型、传输等的刻画是 *判断* (judgmental) 的等式，似乎会更方便和直观。然而，在 Chapter 1 中展示的理论中，恒等类型通过其归纳原则对所有类型进行了统一定义，因此我们不能 "重新定义" 它们在不同类型上的不同表现。因此，在本章中讨论的特定类型的刻画，大多是我们必须发现并证明的 *定理*。

实际上，Chapter 1 的类型理论不足以证明两个类型构造器的所需定理：Π-类型和宇宙 (universes)。因此，我们不得不在类型理论中引入公理，以使那些 "定理" 成立。类型理论上，一个 *公理* (axiom)（见 §1.1）是一个 "原子 (atomic)" 元素，被声明为占据某个特定类型，而没有任何规则来约束它的行为，除了与它所占据的类型有关的规则。

Π-类型的公理 (§2.7) 对类型论者来说很熟悉：它被称为 *函数外延性* (function extensionality)，其陈述（粗略地说）是，如果两个函数在 §2.4 的意义上是同伦的，那么它们是相等的。然而，宇宙 (universes) 的公理 (??) 是 Voevodsky 对同伦类型论 (Homotopy Type Theory) 的新贡献：它被称为 *单值性公理* (univalence axiom)，其陈述（粗略地说）是，如果两个类型在 §2.4 的意义上是等价的，那么它们是相等的。我们已经在引言中提到过这个公理；它将在本书中起到非常重要的作用。[1]

需要注意的是，并非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 (higher inductive types)（见 Chapters 6 and 8）。例如，计算 $\mathbb{S}^n$ 类型的恒等类型（见 §6.4）等同于计算球体的高阶同伦群 (higher homotopy groups)，这是代数拓扑学 (algebraic topology) 中一个深刻且重要的研究领域。

## 2.6 笛卡尔积类型 (Cartesian product types)

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出路径 $\mathsf{pr}_1(p) : \mathsf{pr}_1(x) =_A \mathsf{pr}_1(y)$ 和 $\mathsf{pr}_2(p) : \mathsf{pr}_2(x) =_B \mathsf{pr}_2(y)$。因此，我们有一个函数

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

Theorem 2.6.2. *对于任意 $x$ 和 $y$，函数 (2.6.1) 是一个等价。*

[1] 我们选择将这些原则作为公理引入，但也可能有其他方式来构造一个使它们成立的类型理论。参见本章的笔记。

从逻辑上讲，这表明两个对偶只有在它们的分量相等时才相等。从范畴论 (category-theoretically) 的角度看，这表明积群胚 (product groupoid) 中的态射 (morphisms) 是态射对。从同伦论 (homotopy-theoretically) 的角度看，这表明积空间 (product space) 中的路径是路径对。

证明. 我们需要一个反方向的函数：

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

根据笛卡尔积的归纳规则，我们可以假设 $x$ 和 $y$ 都是对偶，即 $x \equiv (a,b)$ 和 $y \equiv (a',b')$ 对于某些 $a,a' : A$ 和 $b,b' : B$。在这种情况下，我们需要的是一个函数

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

现在根据定义域中的笛卡尔积的归纳，我们可以假设给定 $p : a = a'$ 和 $q : b = b'$。通过两个路径归纳，我们可以假设 $a \equiv a'$ 和 $b \equiv b'$ 并且 $p$ 和 $q$ 都是反身性 (reflexivity)。但在这种情况下，我们有 $(a,b) \equiv (a',b')$，因此我们可以将输出也定义为反身性。

剩下的证明 (2.6.3) 是 (2.6.1) 的拟逆 (quasi-inverse)。这是一系列简单的归纳过程，但它们必须按正确的顺序进行。

在一个方向上，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 $x \equiv y$ 并且 $r$ 是反身性。在这种情况下，由于 $\mathsf{ap}_{\mathsf{pr}_1}$ 和 $\mathsf{ap}_{\mathsf{pr}_2}$ 是通过路径归纳定义的，(2.6.1) 将 $r \equiv \mathsf{refl}_x$ 转换为对 $(\mathsf{refl}_{\mathsf{pr}_1 x}, \mathsf{refl}_{\mathsf{pr}_2 x})$。现在通过对 $x$ 的归纳，我们可以假设 $x \equiv (a,b)$，因此这是 $(\mathsf{refl}_a, \mathsf{refl}_b)$。因此，(2.6.3) 将其按定义转换为 $\mathsf{refl}_{(a,b)}$，这（在我们当前的假设下）是 $r$。

在另一个方向上，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，将其简化为一对 $(p,q)$ 其中 $p : a = a'$ 和 $q : b = b'$。现在通过对 $p$ 和 $q$ 的归纳，我们可以假设它们是反身性 $\mathsf{refl}_a$ 和 $\mathsf{refl}_b$，在这种情况下 (2.6.3) 产生 $\mathsf{refl}_{(a,b)}$ 然后 (2.6.1) 将我们返回到 $(\mathsf{refl}_a, \mathsf{refl}_b) \equiv (p,q) \equiv s$。 □

特别是，我们已经证明 (2.6.1) 有一个逆 (2.6.3)，我们可以将其表示为

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

请注意，这种情况的一个特例产生了积类型的命题唯一性原则：$z = (\mathsf{pr}_1(z), \mathsf{pr}_2(z))$。

可以将 $\mathsf{pair}^{=}$ 视为 $x = y$ 的 *构造函数* (constructor) 或 *引入规则* (introduction rule)，类似于 $A \times B$ 本身的 “对偶 (pairing)” 构造函数，它在给定 $a : A$ 和 $b : B$ 时引入对偶 $(a,b)$。从这个角度看，(2.6.1) 的两个分量：

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

是 *消去规则* (elimination rules)。类似地，证明 (2.6.3) 是 (2.6.1) 的拟逆的两个同伦分别由 *命题计算规则* (propositional computation rules) 组成：

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

对于 $p : \mathsf{pr}_1 x = \mathsf{pr}_1 y$ 和 $q : \mathsf{pr}_2 x = \mathsf{pr}_2 y$，以及 *命题唯一性原则* (propositional uniqueness principle)：

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

我们还可以按分量描述 $A \times B$ 中路径的反身性、逆元和组合：

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

或者，写成不同的形式：

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

所有这些等式都可以通过对给定路径使用路径归纳并返回反身性来推导出来。对于 §2.1 中考虑的其余高阶群胚结构也是如此，尽管插入足够的其他一致路径以得到一个类型检查的等式开始变得乏味。例如，如果我们将 Lemma 2.1.4(iv) 中路径的逆元表示为 $\mathsf{assoc}(p,q,r)$，并将上面显示的最后一个路径表示为 $\mathsf{pair}^{\cdot}(p,q,p',q')$，那么对于任何适当类型的 $u,v,z,w:A\times B$ 和 $p,q,r,p',q',r'$，我们有

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

幸运的是，我们永远不必使用任何此类高维一致性。

我们现在考虑在点对点积类型族 (pointwise product of type families) 中的传输。给定类型族 $A,B:Z\to\mathcal{U}$，我们滥用符号将 $A\times B:Z\to\mathcal{U}$ 表示为类型族 $(A\times B)(z):\equiv A(z)\times B(z)$。现在给定 $p:z=_Z w$ 和 $x:A(z)\times B(z)$，我们可以沿 $p$ 传输 $x$ 以获得 $A(w)\times B(w)$ 的元素。

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 $p$ 是反身性，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，剩下的就是证明 $x=(\mathsf{pr}_1 x,\mathsf{pr}_2 x)$。但这是积类型的命题唯一性原则，正如我们在上面提到的，它来自于 Theorem 2.6.2。 □

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 $h:B\to B'$；然后我们可以定义一个函数 $f:A\times B\to A'\times B'$ 通过 $f(x):\equiv(g(\mathsf{pr}_1 x),h(\mathsf{pr}_2 x))$。

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，上述等式类型正确。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 $\mathsf{pr}_1(f(x))\equiv g(\mathsf{pr}_1 x)$ 和 $\mathsf{pr}_2(f(x))\equiv h(\mathsf{pr}_2 x)$，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 $x\equiv(a,b)$ 和 $y\equiv(a',b')$，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 $q$ 是反身性，在这种情况下，所需的等式在判断上成立。 □

## 2.7 Π-类型与函数外延性公理 (Π-types and the function extensionality axiom)

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 (pointwise paths):

$$(f = g) \simeq \left(\prod_{x:A} (f(x) =_{B(x)} g(x))\right) \tag{2.7.1}$$

从传统的角度来看，这意味着在每个点都相等的两个函数作为函数是相等的。从拓扑学的角度来看，这意味着函数空间中的路径与连续同伦 (continuous homotopy) 是相同的。从范畴论的角度来看，这意味着函子范畴中的同构是自然同构族 (natural family of isomorphisms)。
然而，与前几节的情况不同，Chapter 1 中提出的基本类型论不足以证明 (2.7.1)。我们只能说有一个确定的函数

$$\mathsf{happly} : (f = g) \to \prod_{x:A} (f(x) =_{B(x)} g(x)) \tag{2.7.2}$$

这个函数可以通过路径归纳容易地定义。因此，目前我们假设:

Axiom 2.7.3 (函数外延性 (Function extensionality)). *对于任意 $A$、$B$、$f$ 和 $g$，函数 (2.7.2) 是一个等价。*

我们将在后面的章节中看到，这个公理可以从一值性 (univalence) 得出 (见 ?? and §4.9)，也可以从区间类型 (interval type) 得出 (见 §6.3 和 Exercise 6.10)。
特别是，Axiom 2.7.3 意味着 (2.7.2) 有一个拟逆

$$\mathsf{funext} : \left(\prod_{x:A} (f(x) = g(x))\right) \to (f = g)$$

这个函数也被称为 “函数外延性”。正如我们在 §2.6 中对 $\mathsf{pair}^=$ 所做的那样，我们可以将 $\mathsf{funext}$ 视为 $f = g$ 类型的 *引入规则* (introduction rule)。从这个角度看，$\mathsf{happly}$ 是 *消去规则* (elimination rule)，而证明 $\mathsf{funext}$ 是 $\mathsf{happly}$ 的拟逆的同伦则成为命题计算规则 (propositional computation rule)

$$\mathsf{happly}(\mathsf{funext}(h), x) = h(x) \qquad \text{对于} h : \prod_{x:A} (f(x) = g(x))$$

以及命题唯一性原则 (propositional uniqueness principle):

$$p = \mathsf{funext}(x \mapsto \mathsf{happly}(p, x)) \qquad \text{对于} p : f = g$$

我们还可以在 Π-类型中计算恒等式、逆元和组合；它们仅通过逐点操作给出:

$$\begin{aligned} \mathsf{refl}_f &= \mathsf{funext}(x \mapsto \mathsf{refl}_{f(x)}) \\ \alpha^{-1} &= \mathsf{funext}(x \mapsto \mathsf{happly}(\alpha, x)^{-1}) \\ \alpha \cdot \beta &= \mathsf{funext}(x \mapsto \mathsf{happly}(\alpha, x) \cdot \mathsf{happly}(\beta, x)) \end{aligned}$$

这些等式中的第一个来自于 $\mathsf{happly}$ 的定义，而第二个和第三个是简单的路径归纳。
由于非依赖函数类型 $A \to B$ 是当 $B$ 不依赖于 $x$ 时的 $\prod_{(x:A)} B(x)$ 类型的特例，因此我们上面所说的内容也适用于非依赖情况。然而，在非依赖情况下，传输规则更为简单。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

$$\mathsf{transport}^{A \to B}(p, f) = \left(x \mapsto \mathsf{transport}^B(p, f(\mathsf{transport}^A(p^{-1}, x)))\right) \tag{2.7.4}$$

其中 $A \to B$ 滥用地表示由 $X \to \mathcal{U}$ 定义的类型族

$$(A \to B)(x) :\equiv (A(x) \to B(x))$$

换句话说，当我们沿路径 $p : x_1 = x_2$ 传输函数 $f : A(x_1) \to B(x_1)$ 时，我们得到函数 $A(x_2) \to B(x_2)$，它将其参数在类型族 $A$ 中沿 $p$ 向后传输，应用 $f$，然后在类型族 $B$ 中沿 $p$ 向前传输结果。这可以通过路径归纳容易证明。

传输依赖函数类似，但更为复杂。假设给定 $X$ 和 $p$，如前所述，类型族 $A : X \to \mathcal{U}$ 和 $B : \prod_{(x:X)}(A(x) \to \mathcal{U})$，以及依赖函数 $f : \prod_{(a:A(x_1))} B(x_1, a)$。然后对于 $a : A(x_2)$，我们有

$$\mathsf{transport}^{\Pi_A(B)}(p, f)(a) = \mathsf{transport}^{\widehat{B}}\Big(\big(\mathsf{pair}^{=}(p^{-1}, \mathsf{refl}_{p^{-1}{}_*(a)})\big)^{-1}, f(\mathsf{transport}^{A}(p^{-1}, a))\Big)$$

其中 $\Pi_A(B)$ 和 $\widehat{B}$ 分别表示类型族

$$\begin{array}{rcl} \Pi_A(B) & :\equiv & \big(x \mapsto \prod_{(a:A(x))} B(x, a)\big) \quad : \quad X \to \mathcal{U} \\ \widehat{B} & :\equiv & \big(w \mapsto B(\mathsf{pr}_1 w, \mathsf{pr}_2 w)\big) \quad : \quad \big(\sum_{(x:X)} A(x)\big) \to \mathcal{U} \end{array} \tag{2.7.5}$$

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：我们将参数向后传输，应用函数，然后再次将结果向前传输。

现在回想一下，对于一般的类型族 $P : X \to \mathcal{U}$，在 §2.2 中我们定义了 $p : x =_X y$ 上从 $u : P(x)$ 到 $v : P(y)$ 的 *依赖路径* (dependent paths) 类型为 $p_*(u) =_{P(y)} v$。当 $P$ 是函数类型族时，有一种等价的表示方法，它通常更方便。

Lemma 2.7.6. *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 $g : A(y) \to B(y)$，我们有一个等价*

$$\big(p_*(f) = g\big) \;\simeq\; \prod_{a:A(x)} \big(p_*(f(a)) = g(p_*(a))\big)$$

*此外，如果 $q : p_*(f) = g$ 在此等价下对应于 $\widehat{q}$，那么对于 $a : A(x)$，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*

- $i : (p_*(f))(p_*(a)) = p_*(f(p^{-1}{}_*(p_*(a))))$ *来自* (2.7.4)，
- $j : p_*(f(p^{-1}{}_*(p_*(a)))) = p_*(f(a))$ *来自* Lemmas 2.1.4 and 2.3.9，*以及*
- $k : p_*(f(a)) = g(p_*(a))$ *是* $\widehat{q}(a)$。

*证明.* 通过路径归纳，我们可以假设 $p$ 是反身性，在这种情况下，所需的等价简化为函数外延性。然后通过函数外延性的计算规则得出第二个陈述。 □

一般来说，我们经常需要考虑一些路径的连接，每个路径都来源于先前证明的引理或假设的对象，并且描述这个连接过程可能很乏味，就像我们在上述第二个陈述中所做的那样。因此，我们采用一种惯例，将这种连接写成熟悉的数学风格的 “具有原因的等式链 (chains of equalities with reasons)”，并允许我们省略读者可以轻松填补的原因。例如，Lemma 2.7.6 中的路径 $i \centerdot j \centerdot k$ 可以写成这样：

$$\begin{array}{rcll} (p_*(f))(p_*(a)) & = & p_*\big(f(p^{-1}{}_*(p_*(a)))\big) & \text{(由 (2.7.4))} \\ & = & p_*(f(a)) & \\ & = & g(p_*(a)) & \text{(由 } \widehat{q}\text{)} \end{array}$$

在普通数学中，这样的等式链只是证明两件事是相等的。我们通过使用它来描述它们之间的*特定路径*来增强这一点。

像往常一样，对于依赖函数类型，也有一个类似但更复杂的 Lemma 2.7.6 版本。

Lemma 2.7.7. 给定类型族 $A : X \to \mathcal{U}$ 和 $B : \prod_{(x:X)} A(x) \to \mathcal{U}$ 以及 $p : x =_X y$，以及 $f : \prod_{(a:A(x))} B(x,a)$ 和 $g : \prod_{(a:A(y))} B(y,a)$，我们有一个等价

$$(p_*(f) = g) \simeq \Big(\prod_{a:A(x)} \mathsf{transport}^{\widehat{B}}(\mathsf{pair}^{=}(p, \mathsf{refl}_{p_*(a)}), f(a)) = g(p_*(a))\Big)$$

其中 $\widehat{B}$ 如 (2.7.5) 所示。

我们留给读者去证明这一点并制定一个合适的计算规则。

## 2.8 余积 (Coproducts)

到目前为止，我们考虑的大多数类型构造器都被称为负的。直观地说，这意味着它们的元素是由它们在消去规则下的行为所决定的：一个（依赖）对由它的投影所决定，而一个（依赖）函数由它的值所决定。负类型的恒等类型几乎总是可以直接刻画的，包括我们在 §§2.6–2.7 中所做的所有更高结构。宇宙类型本质上不是负的类型，但它的恒等类型行为类似：我们有一个直接的刻画（单值性）和一个更高结构的描述。恒等类型本身当然是一个特殊情况。

现在我们考虑我们的第一个正的类型构造器示例。同样非正式地说，正的类型是由某些构造器 “呈现” 的，其呈现的通用性质通过它的消去规则表达出来。（从范畴论的角度来看，正的类型具有 “映射出” 的通用性质，而负的类型具有 “映射入” 的通用性质。）由于使用表示进行计算通常是一个不可计算的问题，对于正的类型，我们不能总是期望一个简单的恒等类型的刻画。然而，在许多特定情况下，确实存在一个刻画或部分刻画，并且可以通过我们在此示例中介绍的一般方法获得。

（技术上，我们对笛卡尔积和 $\Sigma$-类型的选择表示也是正的。然而，由于这些类型还允许一个仅有细微差别的负表示，它们的恒等类型有一个直接的刻画，不需要使用这里描述的方法。）

考虑余积类型 $A + B$，它由嵌入 $\mathsf{inl} : A \to A + B$ 和 $\mathsf{inr} : B \to A + B$ “呈现”。直观地，我们期望 $A + B$ 包含 $A$ 和 $B$ 的精确副本且彼此不相交，因此我们应该有

$$(\mathsf{inl}(a_1) = \mathsf{inl}(a_2)) \simeq (a_1 = a_2) \tag{2.8.1}$$
$$(\mathsf{inr}(b_1) = \mathsf{inr}(b_2)) \simeq (b_1 = b_2) \tag{2.8.2}$$
$$(\mathsf{inl}(a) = \mathsf{inr}(b)) \simeq \mathbf{0} \tag{2.8.3}$$

我们证明如下。固定一个元素 $a_0 : A$；我们将刻画类型族

$$(x \mapsto (\mathsf{inl}(a_0) = x)) : A + B \to \mathcal{U} \tag{2.8.4}$$

类似的论证将刻画任何 $b_0 : B$ 的类似族 $x \mapsto (x = \mathsf{inr}(b_0))$。这些刻画一起推导出 (2.8.1)–(2.8.3)。为了刻画 (2.8.4)，我们将定义一个类型族 $\mathsf{code} : A + B \to \mathcal{U}$ 并展示 $\prod_{(x:A+B)}((\mathsf{inl}(a_0) = x) \simeq \mathsf{code}(x))$。因为我们希望从中得出 (2.8.1)，所以我们应该有 $\mathsf{code}(\mathsf{inl}(a)) = (a_0 = a)$，并且因为我们还希望得出 (2.8.3)，我们应该有 $\mathsf{code}(\mathsf{inr}(b)) = \mathbf{0}$。关键的见解是，我们可以使用 $A + B$ 的递归原理通过这两个方程来定义 $\mathsf{code} : A + B \to \mathcal{U}$：

$$\mathsf{code}(\mathsf{inl}(a)) :\equiv (a_0 = a)$$
$$\mathsf{code}(\mathsf{inr}(b)) :\equiv \mathbf{0}$$

这是一个非常简单的证明技巧示例，当在同伦类型论中进行同伦论证时会频繁使用；参见例如 §8.1 and ??。现在我们可以证明：

Theorem 2.8.5. 对于所有 $x : A + B$，我们有 $(\mathsf{inl}(a_0) = x) \simeq \mathsf{code}(x)$。

证明. 以下证明的关键在于我们对所有点 $x$ 一起进行，这使我们可以使用余积的消去原理。我们首先定义一个函数

$$\mathsf{encode} : \prod_{(x:A+B)} \prod_{(p:\mathsf{inl}(a_0)=x)} \mathsf{code}(x)$$

通过沿着 $p$ 传输反射性：

$$\mathsf{encode}(x,p) :\equiv \mathsf{transport}^{\mathsf{code}}(p,\mathsf{refl}_{a_0})$$

注意 $\mathsf{refl}_{a_0} : \mathsf{code}(\mathsf{inl}(a_0))$，因为 code 的定义中 $\mathsf{code}(\mathsf{inl}(a_0)) \equiv (a_0 = a_0)$。接下来，我们定义一个函数

$$\mathsf{decode} : \prod_{(x:A+B)} \prod_{(c:\mathsf{code}(x))} (\mathsf{inl}(a_0) = x)$$

要定义 $\mathsf{decode}(x,c)$，我们可以首先使用 $A+B$ 的消去原理，根据 $x$ 是 $\mathsf{inl}(a)$ 形式还是 $\mathsf{inr}(b)$ 形式进行分类。
在第一种情况下，$x \equiv \mathsf{inl}(a)$，那么 $\mathsf{code}(x) \equiv (a_0 = a)$，因此 $c$ 是 $a_0$ 和 $a$ 之间的同一性。因此，$\mathsf{ap}_{\mathsf{inl}}(c) : (\mathsf{inl}(a_0) = \mathsf{inl}(a))$，我们可以定义 $\mathsf{decode}(\mathsf{inl}(a),c)$ 为这个值。
在第二种情况下，$x \equiv \mathsf{inr}(b)$，那么 $\mathsf{code}(x) \equiv \mathbf{0}$，因此 $c$ 存在于空类型中。因此，$\mathbf{0}$ 的消去规则生成了一个 $\mathsf{decode}(\mathsf{inr}(b),c)$ 的值。
这完成了 decode 的定义；现在我们展示对于所有 $x$，$\mathsf{encode}(x,-)$ 和 $\mathsf{decode}(x,-)$ 是拟逆的。一方面，假设给定 $x : A+B$ 和 $p : \mathsf{inl}(a_0) = x$；我们想要证明 $\mathsf{decode}(x,\mathsf{encode}(x,p)) = p$ 但是现在通过（基于的）路径归纳法，我们只需考虑 $x \equiv \mathsf{inl}(a_0)$ 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，让 $x : A+B$ 和 $c : \mathsf{code}(x)$；我们想要证明 $\mathsf{encode}(x,\mathsf{decode}(x,c)) = c$。我们可以再次根据 $x$ 进行分类。如果 $x \equiv \mathsf{inl}(a)$，那么 $c : a_0 = a$ 且 $\mathsf{decode}(x,c) \equiv \mathsf{ap}_{\mathsf{inl}}(c)$，因此

$$\begin{aligned}
\mathsf{encode}(x,\mathsf{decode}(x,c)) &\equiv \mathsf{transport}^{\mathsf{code}}(\mathsf{ap}_{\mathsf{inl}}(c),\mathsf{refl}_{a_0})\\
&= \mathsf{transport}^{a\mapsto(a_0=a)}(c,\mathsf{refl}_{a_0}) && \text{(见 Lemma 2.3.10)}\\
&= \mathsf{refl}_{a_0} \cdot c && \text{(见 ??)}\\
&= c
\end{aligned}$$

最后，如果 $x \equiv \mathsf{inr}(b)$，那么 $c : \mathbf{0}$，因此我们可以得出我们想要的任何结论。 □

当然，如果我们固定 $b_0 : B$ 而不是 $a_0 : A$，会有相应的定理。
特别地，Theorem 2.8.5 表明对于任何 $a : A$ 和 $b : B$，有以下函数

$$\mathsf{encode}(\mathsf{inl}(a),-) : (\mathsf{inl}(a_0) = \mathsf{inl}(a)) \to (a_0 = a)$$

和

$$\mathsf{encode}(\mathsf{inr}(b),-) : (\mathsf{inl}(a_0) = \mathsf{inr}(b)) \to \mathbf{0}$$

第二个函数表示 “$\mathsf{inl}(a_0)$ 不等于 $\mathsf{inr}(b)$”，即 inl 和 inr 的映像是不相交的。第一个函数的传统解释是，当将恒等类型视为命题时，它只是 inl 的单射性。Theorem 2.8.5 的完整同伦陈述提供了更多信息：类型 $\mathsf{inl}(a_0) = \mathsf{inl}(a)$ 和 $a_0 = a$ 实际上是等价的，就像 $\mathsf{inr}(b_0) = \mathsf{inr}(b)$ 和 $b_0 = b$ 也是等价的。

Remark 2.8.6. 特别地，由于二元类型 $\mathbf{2}$ 等价于 $\mathbf{1}+\mathbf{1}$，我们有 $0_{\mathbf{2}} \neq 1_{\mathbf{2}}$。

这个证明展示了一种刻画路径空间的一般方法，我们将经常使用这种方法。要刻画路径空间，第一步是定义一个比较纤维 “code”，它提供路径的更明确描述。有几种不同的方法可以证明这样的比较纤维与路径等价（我们在 §8.1 中展示了同一个结果的几个不同证明）。我们在这里使用的方法称为**编码-解码方法** (encode-decode method)：关键思想是为纤维的所有实例（即作为函数

$\prod_{(x:A+B)} \mathsf{code}(x) \to (\mathsf{inl}(a_0) = x)$）定义 $\mathsf{decode}$，这样路径归纳法可以用来分析 $\mathsf{decode}(x, \mathsf{encode}(x,p))$。
像往常一样，我们还可以描述余积类型中的传输操作。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

$$\mathsf{transport}^{A+B}(p, \mathsf{inl}(a)) = \mathsf{inl}(\mathsf{transport}^{A}(p,a))$$
$$\mathsf{transport}^{A+B}(p, \mathsf{inr}(b)) = \mathsf{inr}(\mathsf{transport}^{B}(p,b))$$

像往常一样，$\Sigma$ 上标处的 $A + B$ 滥用表示为类型族 $x \mapsto A(x) + B(x)$。证明是一个简单的路径归纳。

## 2.9 通用性质 (Universal properties)

通过结合前面各节中描述的路径计算规则，我们可以证明各种类型形成操作满足预期的通用性质，这些性质以同伦的方式解释为等价性。例如，给定类型 $X, A, B$，我们有一个函数

$$(X \to A \times B) \to (X \to A) \times (X \to B) \tag{2.9.1}$$

通过 $f \mapsto (\mathsf{pr}_1 \circ f, \mathsf{pr}_2 \circ f)$ 定义。

Theorem 2.9.2. (2.9.1) 是一个等价。

证明. 我们通过将 $(g,h)$ 发送到 $\lambda x.(g(x),h(x))$ 来定义拟逆。（技术上，我们使用了笛卡尔积 $(X \to A) \times (X \to B)$ 的归纳原理，来简化到对偶的情况。从现在起，我们将经常在不明确提及的情况下应用这一原理。）
现在给定 $f : X \to A \times B$，往返复合生成了函数

$$\lambda x.(\mathsf{pr}_1(f(x)), \mathsf{pr}_2(f(x))) \tag{2.9.3}$$

通过 Theorem 2.6.2，对于任何 $x : X$，我们有 $(\mathsf{pr}_1(f(x)), \mathsf{pr}_2(f(x))) = f(x)$。因此，通过函数外延性，函数 (2.9.3) 等于 $f$。
另一方面，给定 $(g,h)$，往返复合生成了对 $(\lambda x.g(x), \lambda x.h(x))$。通过函数的唯一性原理，这是（判断上）等于 $(g,h)$ 的。 □

事实上，我们还有一个这种通用性质的依赖类型版本。假设给定类型 $X$ 和类型族 $A, B : X \to \mathcal{U}$。那么我们有一个函数

$$\Big(\prod_{x:X} (A(x) \times B(x))\Big) \to \Big(\prod_{x:X} A(x)\Big) \times \Big(\prod_{x:X} B(x)\Big) \tag{2.9.4}$$

如前所述，通过 $f \mapsto (\mathsf{pr}_1 \circ f, \mathsf{pr}_2 \circ f)$ 定义。

Theorem 2.9.5. (2.9.4) 是一个等价。

证明. 留给读者。 □

正如 $\Sigma$-类型是笛卡尔积的广义化，它们满足这一通用性质的广义版本。直接跳到依赖类型版本，假设我们有一个类型 $X$ 和类型族 $A : X \to \mathcal{U}$ 和 $P : \prod_{(x:X)} A(x) \to \mathcal{U}$。那么我们有一个函数

$$\Big(\prod_{x:X} \sum_{(a:A(x))} P(x,a)\Big) \to \Big(\sum_{(g:\prod_{(x:X)} A(x))} \prod_{(x:X)} P(x, g(x))\Big) \tag{2.9.6}$$

注意，如果我们有 $P(x,a) :\equiv B(x)$ 对于某个 $B : X \to \mathcal{U}$，那么 (2.9.6) 简化为 (2.9.4)。

Theorem 2.9.7. (2.9.6) 是一个等价。

证明. 和之前一样,我们定义一个拟逆将 $(g,h)$ 发送到函数 $\lambda x.(g(x),h(x))$。现在给定 $f:\prod_{(x:X)}\sum_{(a:A(x))}P(x,a)$,往返复合生成了函数

$$\lambda x.(\mathsf{pr}_1(f(x)),\mathsf{pr}_2(f(x))) \tag{2.9.8}$$

现在对于任何 $x:X$，根据 ??（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，通过函数外延性，(2.9.8) 等于 $f$。另一方面，给定 $(g,h)$，往返复合生成了 $(\lambda x.g(x),\lambda x.h(x))$，它在判断上等于 $(g,h)$，如之前所示。 □

这一点值得注意，因为 (2.9.6) 的命题即类型的解释是 “选择公理”。如果我们将 Σ 读取为 “存在”，并且将 Π（有时）读取为 “对于所有”，我们可以解释为：

- $\prod_{(x:X)}\sum_{(a:A(x))}P(x,a)$ 解释为 “对于所有 $x:X$，存在 $a:A(x)$ 使得 $P(x,a)$”，并且
- $\sum_{(g:\prod_{(x:X)}A(x))}\prod_{(x:X)}P(x,g(x))$ 解释为 “存在一个选择函数 $g:\prod_{(x:X)}A(x)$，使得对于所有 $x:X$，我们有 $P(x,g(x))$”。

因此，Theorem 2.9.7 表明不仅选择公理 “为真”，其前提实际上等价于其结论。（另一方面，经典 数学家可能会发现 (2.9.6) 并不具有选择公理的通常意义，因为我们已经指定了 $g$ 的值，并且不再有任何选择需要进行。我们将在 §3.8 中回到这一点。）

上述对对类型的通用性质是 “映射入” 的，这是从范畴论中产品的概念中熟悉的。然而，对类型还有一个 “映射出” 的通用性质，这可能看起来不那么熟悉。在笛卡尔积的情况下，非依赖版本只是表达了笛卡尔闭合伴随：

$$\big((A\times B)\to C\big) \;\simeq\; \big(A\to(B\to C)\big)]$$

$$\mathrm{C}:\mathrm{A}\times B\to\mathcal{U}:$$

$$\Big(\prod_{w:A\times B}C(w)\Big) \;\simeq\; \Big(\prod_{(x:A)}\prod_{(y:B)}C(x,y)\Big)]$$

A×$B$ 的归纳原理，而从左到右的是在一对上进行的计算。我们留给读者证明它们是拟逆的。Σ-类型也有类似的版本：

$$\Big(\prod_{w:\sum_{(x:A)}B(x)}C(w)\Big) \;\simeq\; \Big(\prod_{(x:A)}\prod_{(y:B(x))}C(x,y)\Big) \tag{2.9.9}$$

同样，从右到左的函数是归纳原理。

一些其他的归纳原理也是这种通用性质的一部分。例如，路径归纳是如下等价的从右到左方向：

$$\Big(\prod_{(x:A)}\prod_{(p:a=x)}B(x,p)\Big) \;\simeq\; B(a,\mathsf{refl}_a) \tag{2.9.10}$$

对于任何 $a:A$ 和类型族 $B:\prod_{(x:A)}(a=x)\to\mathcal{U}$。然而，具有递归的归纳类型，如自然数，具有更复杂的通用性质；参见 Chapter 5。

由于 Theorem 2.9.2 表达了笛卡尔积的通常通用性质（在适当的同伦理论意义上），范畴论倾向的读者可能会想知道类型的其他极限和余极限。在 Exercise 2.9 中，我们请读者展示余积类型 $A+B$ 也具有预期的通用性质，$\mathbf{1}$（终端对象）的零元情况和 $\mathbf{0}$（初始对象）都很简单。

对于纤维积，预期的显式构造是有效的：给定 $f:A\to C$ 和 $g:B\to C$，我们定义

$$A\times_C B :\equiv \sum_{(a:A)}\sum_{(b:B)}(f(a)=g(b)) \tag{2.9.11}$$

在 Exercise 2.11 中，我们请读者验证这一点。一些更一般的同伦极限可以通过类似的方式构造，但对于余极限，我们需要一个新成分；参见 Chapter 6。

## Notes

等式类型的定义及其归纳原理归功于 Martin-Löf [?]。如 Chapter 1 中的笔记所述，我们的等式类型属于*内涵*类型论，而不是*外延*类型论。通常来说，某种等式的概念被称为 “内涵的”，如果它通过其特定的定义来区分对象，而被称为 “外延的”，如果它不区分具有相同 “外延” 或 “可观测行为” 的对象。按照弗雷格的术语，内涵等式比较*意义*，而外延等式只比较*指称*。我们还可以说某种等式比另一种等式 “更” 或 “更少” 的外延，意味着它考虑对象的更少或更多的内涵方面，分别地。

*内涵*类型论之所以得名，是因为它的*判断*等式，$x \equiv y$，是一种非常内涵的等式：它基本上表示 $x$ 和 $y$“具有相同的定义”，在我们扩展函数的定义方程后。相比之下，命题等式类型 $x = y$ 更具有外延性，即使是在没有任何公理的内涵类型论中：例如，我们可以通过归纳证明对于所有 $m, n : \mathbb{N}$，$n + m = m + n$，但我们不能说对于所有 $m, n : \mathbb{N}$，$n + m \equiv m + n$，因为加法的*定义*是不对称地对待其参数的。我们可以通过添加函数外延性（两个函数在所有输入上具有相同的行为时，它们是相等的，无论它们是如何定义的）和一致性（它可以看作是宇宙的一种外延性性质：两个类型在所有上下文中行为相同即相等）这样的公理，使内涵类型论中的等式类型更具有外延性。函数外延性和一致性在罗素和丘奇的最早的类型论中已经出现（“命题的外延性”）。

如前所述，*外延*类型论还包括一个 “反射规则”，其表示如果有一个元素 $p : x = y$，那么实际上 $x \equiv y$。因此，外延类型论之所以得名，是因为它不允许任何纯粹*内涵*的等式：反射规则迫使判断等式与更具外延性的等式类型重合。此外，可以从反射规则中推导出函数外延性（至少在函数的判断唯一性原理的存在下）。然而，反射规则还意味着所有的高阶群结构都崩塌了（参见 Exercise 2.14），因此与一致性公理不一致（参见 Example 3.1.9）。因此，将一致性视为一种外延性性质，可以说内涵类型论允许比外延类型论 “更外延” 的等式类型。

等式的对称性（反转）和传递性（连接）的证明在类型论中是众所周知的。这些使得每个类型成为一个 1-群（至多同伦）这一事实在 [?] 中被利用，用于首次为类型论提供 “同伦” 风格的语义。

实际的同伦解释，将等式类型解释为路径空间，并将类型族解释为纤维，是由 [?] 提出的，他们使用了 Quillen 模型范畴的形式主义。在论文 [?] 中，也给出了严格的 $\infty$-群的解释。对于在类型论中构造*所有*的高阶操作和一致性，请参阅 [?] 和 [?]。

诸如 $\mathsf{transport}^P(p, -)$ 和 $\mathsf{ap}_f$ 这样的操作，以及一个良好的等价概念，首先由 Voevodsky 在类型论中进行了广泛研究，他使用了证明助手 Coq 。随后，发现了许多其他等价定义，这些定义在 Chapter 4 中进行了比较。

§2.5 中描述的等式类型、传输等的 “计算” 解释由 [?] 强调。他们还描述了一种 “1-截断” 类型论（参见 Chapter 7），其中这些规则是判断等式。是否可以将这种规则扩展到完整的未截断理论是当前同伦类型论研究的一个课题。

“如果两个函数在逐点上是相等的，那么它们是相等的” 这一天真的函数外延性形式，是类型论中的一个常见公理，可以追溯到 [?]。在 [?] 中，研究了一些更强形式的函数外延性。我们使用的版本，识别函数类型的等式类型直至等价，首先由 Voevodsky 研究，他还证明了它可以由天真的版本（以及一致性；参见 §4.9）推导出来。

一致性公理也是由 Voevodsky 提出的。它最初由单纯集模型中的语义考虑所推动；参见 [?]。在群模型中，Hofmann 和 Streicher [?] 提出了类似的公理，称之为 “宇宙外延性”。它使用了拟逆 (2.4.5) 而不是一个良好的 “等价” 概念，因此仅对于 1-类型的宇宙是正确的（并且与一致性等价）（参见 Definition 3.1.7）。

在我们使用的类型论中，函数外延性和一致性必须作为公理假设，即断言某些类型中存在元素，但不根据该类型的规则构造这些元素。虽然这种方法是可行的，但它有一些缺点。例如，如果我们能够完全基于规则来构建类型论而不是断言公理，类型论在形式上会更好。此外，在 2.6–?? 中，这些定理仅为命题等式（路径）或等价，而非判断等式，因此在来回跨越它们时，我们必须明确提及。目前同伦类型论研究的一个方向是描述一种类型系统，其中这些规则是*判断*等式，从而一次性解决这两个问题。到目前为止，这仅在一些简单的情况下得以实现，尽管像 [?] 这样的初步结果是令人鼓舞的。还有其他可能的方法可以将一致性和函数外延性引入类型论，例如具有足够强大的 “高阶商” 或 “高阶归纳-递归类型”。

2.8–?? 中的简单结论，如 “$\mathsf{inl}$ 和 $\mathsf{inr}$ 是单射且不相交”，在类型论中是众所周知的，函数 $\mathsf{encode}$ 的构造是证明它们的通常方法。我们描述的更精细的方法，它（直至等价）描述了正类型的整个等式类

型，是一种更近的发展；参见例如 [?]。

类型论的选择公理 (2.9.6) 在 William Howard 的关于命题即类型对应的原始论文 [?] 中被注意到，并随着 Martin-Löf 引入其依赖类型论而进一步研究。在 Bourbaki 的集合论中，它被称为 “分配律”。

对于在同伦类型论中对纤维积和更一般的同伦极限的更全面（和形式化）的讨论，请参阅 [?]。有限图上的图 图表的极限 是最容易形式化的一般种类；对于类别（或更一般的 $(\infty,1)$-类别）的图表（或更一般的 $(\infty,1)$-类别），问题在于一般情况下需要（同伦一致）图表的无限多个一致性条件。解决这个问题是同伦类型论中的一个重要的开放问题。

## Exercises

Exercise 2.1. 展示 Lemma 2.1.2 的三个明显的证明是成对相等的。

Exercise 2.2. 展示在前一练习中构造的三个等式的证明形成了一个交换三角形。换句话说，如果将连接的三种定义表示为 $(p \centerdot_1 q)$、$(p \centerdot_2 q)$ 和 $(p \centerdot_3 q)$，那么连接的等式

$$(p \centerdot_1 q) = (p \centerdot_2 q) = (p \centerdot_3 q)$$

等于等式 $(p \centerdot_1 q) = (p \centerdot_3 q)$。

Exercise 2.3. 提供 Lemma 2.1.2 的第四种不同的证明，并证明它与其他证明是相等的。

Exercise 2.4. 通过对 $n$ 进行归纳定义一个类型 $A$ 中的 n-**维路径** 的一般概念，同时定义这些路径的边界类型。

Exercise 2.5. 证明 (2.3.6) 和 (2.3.7) 是逆等价。

Exercise 2.6. 证明如果 $p : x = y$，那么 $(p \centerdot -) : (y = z) \to (x = z)$ 是等价的。

Exercise 2.7. 从笛卡尔积到 $\Sigma$-类型推广 Theorem 2.6.5，并给出相关的证明。

Exercise 2.8. 为余积的情况构造 Theorem 2.6.5 的类似版本。

Exercise 2.9. 证明余积具有预期的通用性质，

$$(A + B \to X) \simeq (A \to X) \times (B \to X)$$

你能将其推广为涉及依赖函数的等价性吗?

Exercise 2.10. 证明 $\Sigma$-类型是 “结合的”，即对于任何 $A : \mathcal{U}$ 及其族 $B : A \to \mathcal{U}$ 和 $C : (\sum_{(x:A)} B(x)) \to \mathcal{U}$，我们有

$$\Big(\sum_{(x:A)} \sum_{(y:B(x))} C((x,y))\Big) \simeq \Big(\sum_{p:\sum_{(x:A)} B(x)} C(p)\Big)$$

Exercise 2.11. 一个（同伦）**交换方形** (commutative square)

$$\begin{array}{ccc} P & \xrightarrow{h} & A \\ {\scriptstyle k}\downarrow & & \downarrow{\scriptstyle f} \\ B & \xrightarrow[g]{} & C \end{array}$$

包含如图所示的函数 $f$、$g$、$h$ 和 $k$，以及一个路径 $f \circ h = g \circ k$。请注意，这正是纤维积 $(P \to A) \times_{P \to C} (P \to B)$ 的一个元素，如 (2.9.11) 中所定义。如果对于任何 $X$，诱导映射

$$(X \to P) \to (X \to A) \times_{(X \to C)} (X \to B)$$

是等价的，那么该交换方形称为一个（同伦）**纤维积方形** (pullback square)。证明 (2.9.11) 中定义的纤维积 $P :\equiv A \times_C B$ 是纤维积方形的一个角落。

Exercise 2.12. 假设给定两个交换方形

$$\begin{array}{ccccc} A & \longrightarrow & C & \longrightarrow & E \\ \downarrow & & \downarrow & & \downarrow \\ B & \longrightarrow & D & \longrightarrow & F \end{array}$$

并且假设右侧的方形是一个纤维积方形。证明当且仅当外部矩形是一个纤维积方形时，左侧的方形是一个纤维积方形。

Exercise 2.13. 证明 $(\mathbf{2} \simeq \mathbf{2}) \simeq \mathbf{2}$。

Exercise 2.14. 假设我们向类型论添加了“等式反射规则”，该规则表示如果有一个元素 $p : x = y$，那么实际上 $x \equiv y$。证明对于任何 $p : x = x$，我们有 $p \equiv \mathsf{refl}_x$。（这意味着每个类型都是集合，这一概念将在 §3.1 中引入；参见 §7.2。）

Exercise 2.15. 证明可以在不使用函数外延性的情况下加强 ?? 为

$$\mathsf{transport}^B(p, -) =_{B(x)\to B(y)} \mathsf{idtoeqv}(\mathsf{ap}_B(p))$$

。（在此类情况下，为了可读性和一致性，选择了看似较弱的表述。）

Exercise 2.16. 假设我们没有函数外延性（Axiom 2.7.3），而只假设存在一个元素

$$\mathsf{funext} : \prod_{(A:\mathcal{U})} \prod_{(B:A\to\mathcal{U})} \prod_{(f,g:\prod_{(x:A)} B(x))} (f \sim g) \to (f = g)$$

（没有假设它与 happly 之间有任何关系）。证明实际上，这足以推出整个函数外延性公理（即 happly 是一个等价）。这是由 Voevodsky 提出的；其证明较为复杂，可能需要来自后续章节的概念。

Exercise 2.17.

(i) 证明如果 $A \simeq A'$ 和 $B \simeq B'$，那么 $(A \times B) \simeq (A' \times B')$。
(ii) 给出该事实的两个证明，一个使用一致性，另一个不使用它，并证明两个证明是相等的。
(iii) 为其他类型构造的类似结果进行表述和证明：$\Sigma$、$\to$、$\Pi$ 和 $+$。

Exercise 2.18. 为依赖函数构造 Lemma 2.4.3 的版本，并给出相关的证明。

Chapter 3

# 集合与逻辑 (Sets and Logic)

类型论 (Type Theory) 无论是形式化的还是非形式化的，都是一组用于操纵类型 (types) 及其元素的规则。但是，当用自然语言非形式地写数学时，我们通常使用熟悉的词汇，特别是逻辑连词，如“和 (and)”和“或 (or)”，以及逻辑量词，如“对于所有 (for all)”和“存在 (there exists)”。与集合论 (Set Theory) 相比，类型论为我们提供了不止一种方式来将这些英文短语视为对类型的操作。这种潜在的歧义需要通过设置局部或全局的约定、引入新的注释到非正式数学中，或者两者兼而有之来解决。这需要一些适应，但因为类型论允许对逻辑进行更精细的分析，所以我们可以更忠实地表示数学，而不像集合论基础那样存在较少的“语言滥用”。在本章中，我们将解释涉及的问题，并证明我们所做选择的合理性。

## 3.1 集合与 $n$-类型 (Sets and $n$-types)

为了解释类型论的逻辑与集合论的逻辑之间的联系，在类型论中引入 *集合* (Set) 的概念是有帮助的。虽然一般的类型表现得像空间或更高的群体 (Higher Groupoids)，但其中有一个子类，它们的行为更像传统集合论系统中的集合。从范畴学 (Category Theory) 的角度，我们可以考虑 *离散* (Discrete) 群体 (Groupoids)，它们由一组对象 (Objects) 决定，并且只有恒等 (Identity) 态射 (Morphisms) 作为更高的态射；而从拓扑学 (Topology) 的角度，我们可以考虑具有离散拓扑的空间。更一般地，我们可以考虑与这类对象 *等价* (Equivalent) 的群体或空间；由于我们在类型论中所做的一切都是同伦 (Homotopy) 意义上的，我们不能指望区分这些。

直观地说，如果一个类型没有更高的同伦信息，即任意两条平行路径是相等的（同伦意义上的），并且在所有维度上平行的更高路径也是如此，我们会认为一个类型在这种意义上“是一个集合”。幸运的是，因为同伦类型论 (Homotopy Type Theory) 中的一切都是自动函子性/连续的，所以实际上只需在底层级别上询问这一点就足够了。

Definition 3.1.1. 一个类型 $A$ 是一个 **集合** (Set) 当且仅当对于所有 $x, y : A$ 和所有 $p, q : x = y$，我们有 $p = q$。

更精确地，命题 $\mathsf{isSet}(A)$ 被定义为类型

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

如 §1.1 中所述，同伦类型论中的集合与 ZF 集合论中的集合不同，因为没有全局的“隶属谓词 (membership predicate)”$\in$。它们更像结构数学和范畴学中使用的集合，其元素是“抽象点 (abstract points)”，我们通过函数 (Functions) 和关系 (Relations) 给予它们结构。这就是我们需要使用它们作为大多数基于集合的数学的基础系统的全部；我们将在 Chapter 10 中看到一些例子。

哪些类型是集合？在 Chapter 7 中，我们将深入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 ??，对于任何 $x,y:\mathbf{1}$，类型 $(x=y)$ 等价于 $\mathbf{1}$ 。由于 $\mathbf{1}$ 的任何两个元素都是相等的，这意味着 $x=y$ 的任何两个元素都是相等的。

Example 3.1.3. 类型 $\mathbf{0}$ 是一个集合，因为给定任何 $x,y:\mathbf{0}$，我们可以根据 $\mathbf{0}$ 的归纳原理推导出任何我们想要的结论。

Example 3.1.4. 自然数 (Natural Numbers) 的类型 $\mathbb{N}$ 也是一个集合。这可以通过 ?? 得出，因为所有等式类型 $x =_{\mathbb{N}} y$ 都等价于 $\mathbf{1}$ 或 $\mathbf{0}$ ，并且 $\mathbf{1}$ 或 $\mathbf{0}$ 的任何两个元素都是相等的。我们将在 Chapter 7 中看到这个事实的另一个证明。

到目前为止，我们所考虑的大多数类型形成操作也保留了集合的属性。

Example 3.1.5. 如果 $A$ 和 $B$ 是集合，那么 $A\times B$ 也是集合。因为给定 $x,y:A\times B$ 和 $p,q:x=y$，根据 Theorem 2.6.2，我们有 $p=\mathsf{pair}^{=}(\mathsf{ap}_{\mathsf{pr}_1}(p),\mathsf{ap}_{\mathsf{pr}_2}(p))$ 和 $q=\mathsf{pair}^{=}(\mathsf{ap}_{\mathsf{pr}_1}(q),\mathsf{ap}_{\mathsf{pr}_2}(q))$。但是 $\mathsf{ap}_{\mathsf{pr}_1}(p)=\mathsf{ap}_{\mathsf{pr}_1}(q)$，因为 $A$ 是一个集合，并且 $\mathsf{ap}_{\mathsf{pr}_2}(p)=\mathsf{ap}_{\mathsf{pr}_2}(q)$，因为 $B$ 是一个集合；因此 $p=q$。类似地，如果 $A$ 是一个集合并且 $B:A\to\mathcal{U}$ 使得每个 $B(x)$ 都是一个集合，那么 $\sum_{(x:A)}B(x)$ 是一个集合。

Example 3.1.6. 如果 $A$ 是 任意类型并且 $B:A\to\mathcal{U}$ 使得每个 $B(x)$ 都是一个集合，那么类型 $\prod_{(x:A)}B(x)$ 是一个集合。假设 $f,g:\prod_{(x:A)}B(x)$ 并且 $p,q:f=g$。根据函数扩展性 (Function Extensionality)，我们有

$$p=\mathsf{funext}(x\mapsto\mathsf{happly}(p,x))\quad\text{并且}\quad q=\mathsf{funext}(x\mapsto\mathsf{happly}(q,x))$$

但是对于任意 $x:A$，我们有

$$\mathsf{happly}(p,x):f(x)=g(x)\qquad\text{并且}\qquad\mathsf{happly}(q,x):f(x)=g(x)$$

因此由于 $B(x)$ 是一个集合，我们有 $\mathsf{happly}(p,x)=\mathsf{happly}(q,x)$。现在再次使用函数扩展性，依赖函数 $(x\mapsto\mathsf{happly}(p,x))$ 和 $(x\mapsto\mathsf{happly}(q,x))$ 是相等的，因此（应用 $\mathsf{ap}_{\mathsf{funext}}$） $p$ 和 $q$ 也是相等的。

有关更多示例，请参见 Exercises 3.2 and 3.3。有关在同伦类型论中所有集合的子系统（范畴 (Category)）的更系统的研究，请参见 Chapter 10。

集合只是被称为 同伦 $n$-类型 (Homotopy $n$-types) 的阶梯的第一层。下一层包括 1-类型 (1-types)，它们在范畴论中类似于 1-群体 (1-Groupoids)。集合的定义性质（我们也可以称之为 0-类型 (0-type)）是它没有非平凡的路径。类似地，1-类型的定义性质是它没有路径之间的非平凡路径：

Definition 3.1.7. 一个类型 $A$ 是一个 1-**类型** (1-type) 如果对于所有 $x,y:A$ 和 $p,q:x=y$ 以及 $r,s:p=q$，我们有 $r=s$。

同样地，我们可以定义 2-类型、3-类型，依此类推。我们将在 Chapter 7 中递归定义 $n$-类型，并研究不同 $n$ 值的 $n$-类型之间的关系。

然而，目前有两个事实是有用的。首先，这些级别是向上闭合的：如果 $A$ 是一个 $n$-类型，那么 $A$ 是一个 $(n+1)$-类型。例如：

Lemma 3.1.8. *如果 $A$ 是一个集合（即 $\mathsf{isSet}(A)$ 是被占用的），那么 $A$ 是一个 1-类型。*

*证明.* 假设 $f:\mathsf{isSet}(A)$；那么对于所有 $x,y:A$ 和 $p,q:x=y$ 我们有 $f(x,y,p,q):p=q$。固定 $x$, $y$ 和 $p$，并定义 $g:\prod_{(q:x=y)}(p=q)$ 通过 $g(q):\equiv f(x,y,p,q)$。然后对于任意 $r:q=q'$，我们有 $\mathsf{apd}_g(r):r_*(g(q))=g(q')$。因此，根据 ??，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 Definition 3.1.7 中所述，并定义上述的 $g$。然后 $g(p)\cdot r=g(q)$ 并且 $g(p)\cdot s=g(q)$，因此通过抵消 $r=s$。 □

其次，这种按级别划分类型的方法不是退化的，因为不是所有的类型都是集合：

Example 3.1.9. 宇宙 $\mathcal{U}$ 不是一个集合。为了证明这一点，足以展示一个类型 $A$ 和一条路径 $p:A=A$，该路径不等于 $\mathsf{refl}_A$。取 $A=\mathbf{2}$，并定义 $f:A\to A$ 为 $f(0_{\mathbf{2}}):\equiv 1_{\mathbf{2}}$ 和 $f(1_{\mathbf{2}}):\equiv 0_{\mathbf{2}}$。然后对于所有 $x$ 我们有 $f(f(x))=x$（通过一个简单的情况分析），因此 $f$ 是一个等价。因此，根据单一性 (Univalence)，$f$ 产生一条路径 $p:A=A$。

如果 $p$ 等于 $\mathsf{refl}_A$，那么（再次根据单一性） $f$ 将等于 $A$ 的恒等函数。但这将意味着 $0_{\mathbf{2}}=1_{\mathbf{2}}$，这与 Remark 2.8.6 矛盾。

在 Chapters 6 and 8 中，我们将证明对于任意 $n$，存在不属于 $n$-类型的类型。
注意，当且仅当对于任意 $x, y : A$，等式类型 $x =_A y$ 是一个集合时，$A$ 是一个 1-类型。（因此，Lemma 3.1.8 可以等效地理解为集合的等式类型也是集合。）这将成为我们在 Chapter 7 中给出的 $n$-类型的递归定义的基础。
我们也可以将这种从集合的表征"向下"扩展。也就是说，当且仅当对于任意 $x, y : A$，$x =_A y$ 的任意两个元素是相等的，一个类型 $A$ 是一个集合。由于集合等价于 0-类型，因此如果类型在后者属性下成立，我们称之为 $(-1)$-*类型* ($(-1)$-type)。这些类型可以被视为 *狭义命题* (Propositions in a Narrow Sense)，它们的研究正是通常所说的"逻辑"；我们将在本章的剩余部分讨论它。

## 3.2 命题作为类型? (Propositions as Types?)

到目前为止，我们一直遵循 §1.11 中描述的简单"命题即类型 (Propositions as Types)"的哲学，根据该哲学，英语短语如"存在一个 $x : A$ 使得 $P(x)$"被解释为相应的类型 $\sum_{(x:A)} P(x)$，其中一个陈述的证明被视为判断某些特定元素占据该类型。然而，我们也看到了从这种阅读中得出的"逻辑"对一个经典数学家来说似乎不太熟悉的一些方式。例如，在 Theorem 2.9.7 中，我们看到以下陈述

$$\text{"如果对于所有 } x : X \text{ 存在一个 } a : A(x) \text{ 使得 } P(x,a)\text{，那么存在一个函数 } g : \textstyle\prod_{(x:X)} A(x) \text{ 使得对于所有 } x : X \text{ 我们有 } P(x, g(x))\text{"}, \tag{3.2.1}$$

看起来像经典的 *选择公理* (Axiom of Choice)，在这种阅读下总是成立的。这是命题即类型逻辑的一个值得注意且通常有用的特性，但它也表明它与经典逻辑解释的显著不同，在经典逻辑解释下，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，我们现在还可以证明，看起来像经典 *双重否定律* (Law of Double Negation) 和 *排中律* (Law of Excluded Middle) 的相应陈述与单一性公理 (Univalence Axiom) 不兼容。

Theorem 3.2.2. *对于所有 $A : \mathcal{U}$，$\neg(\neg A) \to A$ 并不成立。*

*证明.* 回忆 $\neg A \equiv (A \to \mathbf{0})$。我们还将"并不成立"读作操作符 $\neg$。因此，为了证明这一陈述，足以假设给定某个 $f : \prod_{(A:\mathcal{U})}(\neg\neg A \to A)$ 并构造一个 $\mathbf{0}$ 的元素。
以下证明的思路是观察 $f$ 像类型论中的任何函数一样是"连续的"。根据单一性，这意味着 $f$ 在类型的等价性方面是 *自然的* (Natural)。通过这一点和一个无固定点的自等价性 (Fixed-Point-Free Autoequivalence)，我们将能够提取一个矛盾。
令 $e : \mathbf{2} \simeq \mathbf{2}$ 为等价，由 $e(1_2) :\equiv 0_2$ 和 $e(0_2) :\equiv 1_2$ 定义，如 Example 3.1.9 中所示。令 $p : \mathbf{2} = \mathbf{2}$ 为由单一性对应的路径，即 $p :\equiv \mathsf{ua}(e)$。然后我们有 $f(\mathbf{2}) : \neg\neg\mathbf{2} \to \mathbf{2}$ 并且

$$\mathsf{apd}_f(p) : \mathsf{transport}^{A\mapsto(\neg\neg A\to A)}(p, f(\mathbf{2})) = f(\mathbf{2})$$

因此，对于任意 $u : \neg\neg\mathbf{2}$，我们有

$$\mathsf{happly}(\mathsf{apd}_f(p), u) : \mathsf{transport}^{A\mapsto(\neg\neg A\to A)}(p, f(\mathbf{2}))(u) = f(\mathbf{2})(u)$$

现在通过 (2.7.4)，沿路径 $p$ 在类型族 $A \mapsto (\neg\neg A \to A)$ 中传输 $f(\mathbf{2}) : \neg\neg\mathbf{2} \to \mathbf{2}$，等于沿路径 $p^{-1}$ 在类型族 $A \mapsto \neg\neg A$ 中传输其参数，应用 $f(\mathbf{2})$，然后沿类型族 $A \mapsto A$ 中的路径 $p$ 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，任何两个点 $u, v : \neg\neg\mathbf{2}$ 通过函数扩展性是相等的，因为对于任意 $x : \neg\mathbf{2}$ 我们有 $u(x) : \mathbf{0}$ 因此我们可以得出任何结论，特别是 $u(x) = v(x)$。因此，我们有 $\mathsf{transport}^{A\mapsto\neg\neg A}(p^{-1}, u) = u$，因此从 $\mathsf{happly}(\mathsf{apd}_f(p), u)$ 我们得到一个等式

$$\mathsf{transport}^{A\mapsto A}(p, f(\mathbf{2})(u)) = f(\mathbf{2})(u)$$

最后，如 ?? 中讨论的，在类型族 $A \mapsto A$ 中沿路径 $p \equiv \mathsf{ua}(e)$ 传输等效于应用等价 $e$；因此我们有

$$e(f(\mathbf{2})(u)) = f(\mathbf{2})(u) \tag{3.2.3}$$

然而，我们还可以证明

$$\prod_{x:\mathbf{2}} \neg(e(x) = x) \tag{3.2.4}$$

这可以通过对 $x$ 的情况分析得出：两种情况都是从 $e$ 的定义和 $\mathbf{0}_\mathbf{2} \neq \mathbf{1}_\mathbf{2}$（Remark 2.8.6）这一事实直接得出的。因此，将 (3.2.4) 应用于 $f(\mathbf{2})(u)$ 和 (3.2.3)，我们得到一个 $\mathbf{0}$ 的元素。 □

Remark 3.2.5. 特别地，这意味着不能有一个希尔伯特风格的"选择算子"，它选择每个非空类型的一个元素。关键在于没有这样的算子可以是 *自然的* (Natural)，而在单一性公理下，所有作用于类型的函数必须在类型等价方面是自然的。

Remark 3.2.6. 然而，对于任何 $A$，仍然有 $\neg\neg\neg A \to \neg A$；参见 Exercise 1.11。

Corollary 3.2.7. *对于所有 $A : \mathcal{U}$，$A + (\neg A)$ 并不成立。*

*证明.* 假设我们有 $g : \prod_{(A:\mathcal{U})}(A + (\neg A))$。我们将证明 $\prod_{(A:\mathcal{U})}(\neg\neg A \to A)$，因此我们可以应用 Theorem 3.2.2。因此，假设 $A : \mathcal{U}$ 并且 $u : \neg\neg A$；我们想构造一个 $A$ 的元素。
现在 $g(A) : A + (\neg A)$，因此通过情况分析，我们可以假设 $g(A) \equiv \mathsf{inl}(a)$ 对于某个 $a : A$，或者 $g(A) \equiv \mathsf{inr}(w)$ 对于某个 $w : \neg A$。在第一种情况下，我们有 $a : A$，而在第二种情况下我们有 $u(w) : \mathbf{0}$，因此我们可以得到任何我们想要的（例如 $A$）。因此，在这两种情况下，我们都有一个 $A$ 的元素，如所需。 □

因此，如果我们想要假设单一性公理（当然，我们想要这样做）并且仍然给自己留下使用经典推理 (Classical Reasoning) 的可能性（这也是可取的），我们不能使用未经修改的命题即类型原则来将 *所有*非正式数学陈述解释为类型论，因为那样排中律 (Law of Excluded Middle) 将是错误的。然而，我们也不想完全放弃命题即类型，因为它有许多好的属性（如简单性、构造性和可计算性）。我们现在讨论一种修改的命题即类型逻辑，它解决了这些问题；在 §3.10 中，我们将回到使用何种逻辑的问题。

## 3.3 **纯粹命题** (Mere Propositions)

我们已经看到，命题即类型逻辑既有好的方面，也有不好的方面。它们都有一个共同的原因：当类型被视为命题时，它们可以包含比单纯的真理或谬误更多的信息，并且所有"逻辑"结构都必须尊重这些额外的信息。这表明，我们可以通过限制注意那些不包含比真值更多信息的类型，只将这些类型视为逻辑命题，从而获得更*传统*的逻辑。
这样的类型 $A$ 将在被占用时"真"，而如果它的占用导致矛盾（即 $\neg A \equiv (A \to \mathbf{0})$ 被占用）时"假"。我们要避免的是，将那些类型作为逻辑命题对待，在这些类型中，给出它们的一个元素会比单纯知道该类型被占用提供更多的信息。例如，如果我们被给定 $\mathbf{2}$ 的一个元素，那么我们得到的信息比单纯的 $\mathbf{2}$ 包含一些元素的信息更多。事实上，我们得到了确切的 *一比特* (one bit) 更多的信息：我们知道我们得到了 $\mathbf{2}$ 的 *哪个*元素。相反，如果我们被给定 $\mathbf{1}$ 的一个元素，那么我们得到的信息不会比 $\mathbf{1}$ 包含一个元素的信息更多，因为 $\mathbf{1}$ 的任何两个元素都是相等的。这表明了以下定义。

Definition 3.3.1. 一个类型 $P$ 是一个 **纯粹命题** (Mere Proposition) 当且仅当对于所有 $x, y : P$ 我们有 $x = y$。

请注意，由于我们仍在 *类型论中* (in Type Theory) 做数学，这是一个 *类型论中的定义* (Definition in Type Theory)，这意味着它是一个类型 — 或者更确切地说，是一个类型族 (Type Family)。具体来说，对于任意 $P : \mathcal{U}$，类型 $\mathsf{isProp}(P)$ 被定义为

$$\mathsf{isProp}(P) :\equiv \prod_{x,y:P} (x = y)$$

因此，要断言"$P$ 是一个纯粹命题"意味着展示一个 $\mathsf{isProp}(P)$ 的占用者，这是一个通过路径连接 $P$ 的任何两个元素的依赖函数 (Dependent Function)。这种函数的连续性/自然性意味着不仅 $P$ 的任何两个元素都是相等的，而且 $P$ 也不包含任何更高的同伦。

Lemma 3.3.2. 如果 $P$ 是一个纯粹命题并且 $x_0 : P$，那么 $P \simeq \mathbf{1}$。

证明. 定义 $f : P \to \mathbf{1}$ 为 $f(x) :\equiv \star$，并定义 $g : \mathbf{1} \to P$ 为 $g(u) :\equiv x_0$。该声明由下一个引理和 ?? 中的观察 $\mathbf{1}$ 是一个纯粹命题得出。 □

Lemma 3.3.3. 如果 $P$ 和 $Q$ 是纯粹命题，且 $P \to Q$ 且 $Q \to P$，那么 $P \simeq Q$。

证明. 假设给定 $f : P \to Q$ 和 $g : Q \to P$。那么对于任意 $x : P$，我们有 $g(f(x)) = x$ 因为 $P$ 是一个纯粹命题。类似地，对于任意 $y : Q$ 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 (Quasi-Inverses)。 □

也就是说，正如在 §1.11 中承诺的，如果两个纯粹命题在逻辑上是等价的，那么它们是等价的。

在同伦论 (Homotopy Theory) 中，与 $\mathbf{1}$ 同伦等价的空间被称为 可收缩的 (Contractible)。因此，任何被占用的纯粹命题都是可收缩的（另见 §3.11）。另一方面，未占用的类型 $\mathbf{0}$ 也是（虚拟地）一个纯粹命题。至少在经典数学中，这些是唯一的两种可能性。

纯粹命题在范畴论中也被称为 次终端对象 (Subterminal Objects)（如果从范畴论角度考虑），在集合论中被称为 次单例 (Subsingletons)，或者 h-命题 (h-Propositions)。 在 §3.1 中的讨论表明，我们也应该称它们为 $(-1)$-类型 ($(-1)$-types)；我们将在 Chapter 7 中回到这个问题。形容词“纯粹 (mere)”强调了虽然任何类型都可以被视为命题（我们通过给出其占用者来证明），但一个纯粹命题的类型不能有用地被视为任何 更多的命题：其真实性的见证中不包含任何额外的信息。

请注意，一个类型 $A$ 是一个集合当且仅当对于所有 $x, y : A$，等式类型 $x =_A y$ 是一个纯粹命题。另一方面，通过复制并简化 Lemma 3.1.8 的证明，我们有：

Lemma 3.3.4. 每个纯粹命题都是一个集合。

证明. 假设 $f : \mathsf{isProp}(A)$；因此对于所有 $x, y : A$ 我们有 $f(x,y) : x = y$。固定 $x : A$ 并定义 $g(y) :\equiv f(x,y)$。然后对于任意 $y, z : A$ 和 $p : y = z$，我们有 $\mathsf{apd}_g(p) : p_*(g(y)) = g(z)$。因此根据 ??，我们有 $g(y) \cdot p = g(z)$，这就是说 $p = g(y)^{-1} \cdot g(z)$。因此，对于任意 $p, q : x = y$，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 $f, g : \mathsf{isProp}(A)$。根据函数扩展性，要证明 $f = g$ 只需证明对于任意 $x, y : A$，$f(x,y) = g(x,y)$。但是 $f(x,y)$ 和 $g(x,y)$ 都是 $A$ 中的路径，因此它们是相等的，因为无论是 $f$ 还是 $g$，我们都有 $A$ 是一个纯粹命题，因此根据 Lemma 3.3.4，它是一个集合。同样地，假设 $f, g : \mathsf{isSet}(A)$，也就是说，对于所有 $a, b : A$ 和 $p, q : a = b$，我们有 $f(a,b,p,q) : p = q$ 和 $g(a,b,p,q) : p = q$。但正因为 $A$ 是一个集合（无论是通过 $f$ 还是 $g$），因此是一个 1-类型，这意味着 $f(a,b,p,q) = g(a,b,p,q)$；因此根据函数扩展性，$f = g$。 □

到目前为止，我们已经看到了另一个例子：在 §2.4 中条件 (iii) 声明对于任意函数 $f$，类型 $\mathsf{isequiv}(f)$ 应该是一个纯粹命题。

## 3.4 经典逻辑 vs. 直觉主义逻辑 (Classical vs. Intuitionistic Logic)

有了纯粹命题 (Mere Proposition) 的概念，我们现在可以在同伦类型论中正确地表述 **排中律** (Law of Excluded Middle)：

$$\mathsf{LEM} :\equiv \prod_{A:\mathcal{U}} \Big(\mathsf{isProp}(A) \to (A + \neg A)\Big) \tag{3.4.1}$$

类似地，**双重否定律** (Law of Double Negation) 为：

$$\prod_{A:\mathcal{U}} \Big(\mathsf{isProp}(A) \to (\neg\neg A \to A)\Big) \tag{3.4.2}$$

两者很容易被证明是等价的——见 Exercise 3.18——所以从现在开始我们通常只讨论 LEM 。这种 LEM 的表述避免了 Theorem 3.2.2 and Corollary 3.2.7 中的"悖论 (Paradoxes)"，因为 **2** 不是一个纯粹命题。为了将其与更一般的命题即类型 (Propositions as Types) 表述区分开来，我们将后者重命名为：

$$\mathsf{LEM}_\infty :\equiv \prod_{A:\mathcal{U}} (A + \neg A)$$

为强调起见，可以将正式版本 (3.4.1) 记为 $\mathsf{LEM}_{-1}$；另见 Exercise 7.7。虽然 LEM 不是 Chapter 1 中描述的基本类型论的结果，但它可以作为一个公理被一致地假设（不像其 $\infty$ 对应部分）。例如，我们将在 §10.4 中假设它。

然而，令人惊讶的是，我们在不使用 LEM 的情况下可以走多远。通常，一个定义或定理的简单重述可以使我们避免使用排中律。虽然这有时需要一些适应，但通常是值得的，因为这样可以得到更优雅和更通用的证明。我们在介绍中讨论了一些这样做的好处。

例如，在经典数学 (Classical Mathematics) 中，双重否定经常被不必要地使用。一个非常简单的例子是通常假设一个集合 $A$ 是"非空"的，这实际上意味着 $A$ *不包含 任何*元素。几乎总是我们真正想表达的是积极的断言，即 $A$ *确实*包含至少一个元素，通过去除双重否定，我们使这个陈述减少了对 LEM 的依赖。请记住，我们说一个类型 $A$ 是 *被占用的* (Inhabited) 的，当我们断言 $A$ 本身是一个命题（即我们构造一个 $A$ 的元素，通常是未命名的）。因此，当将一个经典证明翻译为构造逻辑时，我们通常将"非空"一词替换为"被占用的"（尽管有时我们必须将其替换为"仅仅被占用的"；参见 §3.7）。

类似地，在经典数学中发现不必要的反证法 (Proof by Contradiction) 也并不罕见。当然，经典形式的反证法是通过双重否定律进行的：我们假设 $\neg A$ 并得出一个矛盾，从而推导出 $\neg\neg A$，然后通过双重否定我们得到 $A$。然而，通常可以将从 $\neg A$ 推导出矛盾的过程稍作改动，以便直接证明 $A$，从而避免使用 LEM 。

还需要注意的是，如果目标是证明一个 *否定命题* (Negation)，那么"通过反证法"并不涉及 LEM 。事实上，由于 $\neg A$ 定义为类型 $A \to \mathbf{0}$，因此证明 $\neg A$ 就是证明在 $A$ 的假设下得出一个矛盾 ($\mathbf{0}$)。类似地，双重否定律对于否定命题是成立的：$\neg\neg\neg A \to \neg A$。通过练习，可以更仔细地区分否定和非否定命题，并注意到何时使用 LEM ，何时不使用。

因此，与表面看起来相反，进行"构造性 (Constructive)"数学通常并不涉及放弃重要的定理，而是找到最佳方式陈述定义，从而使重要的定理能够构造性地证明。也就是说，我们可以在第一次研究一个主题时自由使用 LEM ，但一旦更好地理解了该主题，我们可以希望通过改进其定义和证明来避免使用该公理。在 *同伦* (Homotopy) 类型论中，这种观察更加明显，因为单一性 (Univalence) 和高阶归纳类型 (Higher Inductive Types) 的强大工具允许我们构造性地解决许多传统上需要经典推理 (Classical Reasoning) 的问题。我们将在 Part II 中看到几个例子。

还值得一提的是，即使在构造数学中，排中律对 *某些*命题也是成立的。传统上对这些命题的称呼是 *可判定的* (Decidable)。

Definition 3.4.3.

(i) 一个类型 $A$ 被称为 **可判定的** (Decidable)，如果 $A + \neg A$。

(ii) 类似地，一个类型族 $B : A \to \mathcal{U}$ 是 **可判定的** (Decidable)，如果 $\prod_{(a:A)} (B(a) + \neg B(a))$

(iii) 特别地，$A$ 具有 **可判定的相等性** (Decidable Equality)，如果 $\prod_{(a,b:A)} ((a = b) + \neg(a = b))$

因此，LEM 正是所有纯粹命题是可判定的的陈述，因此所有纯粹命题的族也是可判定的。特别地，LEM 意味着所有集合（在 §3.1 的意义上）都有可判定的相等性。在这个意义上，具有可判定的相等性是非常强的；见 Theorem 7.2.5。

## 3.5 子集和命题重缩放 (Subsets and Propositional Resizing)

作为纯粹命题的另一个有用性例子，我们讨论子集（以及更一般的子类型）。假设 $P : A \to \mathcal{U}$ 是一个类型族，其中每个类型 $P(x)$ 都被视为一个命题。那么 $P$ 本身就是 $A$ 上的一个 *谓词* (Predicate)，或者是 $A$ 元素的一个 *属性* (Property)。

在集合论中，只要我们有一个集合 $A$ 上的谓词 $P$，我们就可以形成子集 $\{\, x \in A \mid P(x) \,\}$。正如在 §1.11 中简要提到的，在类型论中显而易见的类似物是 $\Sigma$-类型 $\sum_{(x:A)} P(x)$。$\sum_{(x:A)} P(x)$ 的一个元素当然是一个对 $(x, p)$，其中 $x : A$ 且 $p$ 是 $P(x)$ 的一个证明。然而，对于一般的 $P$，一个元素 $a : A$ 可能会导致 $\sum_{(x:A)} P(x)$ 中的多个不同元素，如果命题 $P(a)$ 有多个不同的证明的话。这与通常的子集直觉相悖。但是，如果 $P$ 是一个 *纯粹命题*，那么这种情况就不会发生。

Lemma 3.5.1. *假设 $P : A \to \mathcal{U}$ 是一个类型族，使得对于所有 $x : A$，$P(x)$ 是一个纯粹命题。如果 $u, v : \sum_{(x:A)} P(x)$ 使得 $\mathsf{pr}_1(u) = \mathsf{pr}_1(v)$，那么 $u = v$。*

*证明.* 假设 $p : \mathsf{pr}_1(u) = \mathsf{pr}_1(v)$。通过 ??, 要证明 $u = v$, 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，而它是一个纯粹命题；因此它们是相等的。 □

例如，回想在 §2.4 中我们定义了

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

其中每个类型 $\mathsf{isequiv}(f)$ 都应该是一个纯粹命题。这意味着，如果两个等价性 (Equivalences) 具有相同的基础函数，那么它们作为等价性是相等的。

从现在起，如果 $P : A \to \mathcal{U}$ 是一个纯粹命题的族（即每个 $P(x)$ 是一个纯粹命题），我们可以写作

$$\{\, x : A \mid P(x) \,\} \tag{3.5.2}$$

作为 $\sum_{(x:A)} P(x)$ 的替代符号。（在技术上没有理由不对任意 $P$ 使用这个符号，但这种用法可能会由于意外的含义而令人困惑。）如果 $A$ 是一个集合，我们称 (3.5.2) 为 $A$ 的一个 **子集** (Subset)；对于一般的 $A$，我们可以称它为 **子类型** (Subtype)。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，因为类型 (3.5.2) 单独不记得它与 $A$ 的关系。

给定这样的 $P$ 和 $a : A$，我们可以写作 $a \in P$ 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** (Member)。类似地，如果 $\{\, x : A \mid Q(x) \,\}$ 是 $A$ 的另一个子集，那么我们说 $P$ **包含在** (Contained) $Q$ 中，并写作 $P \subseteq Q$，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”：

$$\mathsf{Set}_\mathcal{U} :\equiv \{\, A : \mathcal{U} \mid \mathsf{isSet}(A) \,\}$$
$$\mathsf{Prop}_\mathcal{U} :\equiv \{\, A : \mathcal{U} \mid \mathsf{isProp}(A) \,\}$$

$\mathsf{Set}_\mathcal{U}$ 的一个元素是一个类型 $A : \mathcal{U}$ 以及证据 $s : \mathsf{isSet}(A)$，$\mathsf{Prop}_\mathcal{U}$ 类似。Lemma 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，我们将经常滥用符号并简单地写作 $A : \mathsf{Set}_\mathcal{U}$ 而不是 $(A, s) : \mathsf{Set}_\mathcal{U}$。如果没有必要指定讨论的宇宙，我们也可以省略下标 $\mathcal{U}$ 。

请记住，对于任意两个宇宙 $\mathcal{U}_i$ 和 $\mathcal{U}_{i+1}$，如果 $A : \mathcal{U}_i$，那么 $A : \mathcal{U}_{i+1}$ 也是如此。因此，对于任意 $(A, s) : \mathsf{Set}_{\mathcal{U}_i}$，我们也有 $(A, s) : \mathsf{Set}_{\mathcal{U}_{i+1}}$，$\mathsf{Prop}_{\mathcal{U}_i}$ 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) 不能是等价的，因为这样我们可以重现康托尔集合论 (Cantorian Set Theory) 中熟悉的自引用悖论。然而，尽管 (3.5.4) 并不自动是等价的，但在我们目前介绍的类型论中，它是一致的，因此我们可以假设它是等价的。也就是说，我们可以考虑将以下公理添加到类型论中。

Axiom 3.5.5 (命题重缩放 (Propositional Resizing)). *映射 $\mathsf{Prop}_{\mathcal{U}_i} \to \mathsf{Prop}_{\mathcal{U}_{i+1}}$ 是等价的。*

我们将这个公理称为 **命题重缩放** (Propositional Resizing)，因为它意味着宇宙 $\mathcal{U}_{i+1}$ 中的任何纯粹命题都可以“重缩放”到较小宇宙 $\mathcal{U}_i$ 中的一个等价命题。如果 $\mathcal{U}_{i+1}$ 满足 LEM ，则它会自动成立（参见 Exercise 3.10）。我们一般不会假设这个公理，尽管在某些地方我们会明确地使用它作为一个假设。

它是纯粹命题的一种 *不定界性* (Impredicativity)，通过避免使用它，类型论保持 *定界性* (Predicativity)。

在实践中，我们最常需要的是一个稍微不同的陈述：即所考虑的宇宙 $\mathcal{U}$ 包含一个“分类所有纯粹命题”的类型。换句话说，我们需要一个类型 $\Omega : \mathcal{U}$ 以及一个 $\Omega$ 索引的纯粹命题的族，它包含了所有纯粹命题（到等价为止）。如果 $\mathcal{U}$ 不是最小宇宙 $\mathcal{U}_0$，这个陈述可以从上述命题重缩放中得出，因为我们可以定义 $\Omega :\equiv \mathsf{Prop}_{\mathcal{U}_0}$。

不定界性的一个用途是定义幂集 (Power Set)。将集合 $A$ 的 **幂集** (Power Set) 定义为 $A \to \mathsf{Prop}_{\mathcal{U}}$ 是很自然的；但在没有不定界性的情况下，这个定义（即使是等价的）依赖于宇宙 $\mathcal{U}$ 的选择。但是通过命题重缩放，我们可以将幂集定义为

$$\mathcal{P}(A) :\equiv (A \to \Omega)$$

这样就与 $\mathcal{U}$ 无关了。另见 §10.1.4。

## 3.6 **纯粹命题的逻辑** (The Logic of Mere Propositions)

我们在 §1.1 中提到，与只有一种基本概念（类型）的类型论不同，集合论基础有两种基本概念：集合和命题。因此，经典数学家 (Classical Mathematician) 习惯于分别操作这两种对象。

在类型论中，恢复类似的二分法是可能的，集合论命题的作用由 纯粹命题的类型（和类型族）扮演。在许多情况下，通过简单地将对应的类型生成器限制为纯粹命题，我们可以在这种逻辑中表示逻辑连结词和量词。当然，这需要知道对应的类型生成器是否保持纯粹命题。

Example 3.6.1. 如果 $A$ 和 $B$ 是纯粹命题，那么 $A \times B$ 也是纯粹命题。使用积 (Product) 中路径 (Paths) 的刻画来证明这一点很容易，就像 Example 3.1.5，但更简单。因此，连结词“和”(And) 保持纯粹命题。

Example 3.6.2. 如果 $A$ 是任意类型，$B : A \to \mathcal{U}$ 是类型族，使得对于所有 $x : A$，类型 $B(x)$ 是纯粹命题，那么 $\prod_{(x:A)} B(x)$ 是纯粹命题。证明与 Example 3.1.6 类似，但更简单：给定 $f, g : \prod_{(x:A)} B(x)$，对于任意 $x : A$，我们有 $f(x) = g(x)$，因为 $B(x)$ 是纯粹命题。但是通过函数外延性 (Function Extensionality)，我们有 $f = g$。

特别地，如果 $B$ 是纯粹命题，那么无论 $A$ 是什么，$A \to B$ 也是纯粹命题。更特别地，因为 $\mathbf{0}$ 是纯粹命题，所以 $\neg A \equiv (A \to \mathbf{0})$ 也是纯粹命题。因此，连结词“蕴涵”(Implies) 和“非”(Not) 保持纯粹命题，连同“对所有人”(For All) 的量词。

另一方面，一些类型生成器不保持纯粹命题。即使 $A$ 和 $B$ 是纯粹命题，$A + B$ 一般也不是。例如，$\mathbf{1}$ 是纯粹命题，但 $\mathbf{2} = \mathbf{1} + \mathbf{1}$ 不是。从逻辑上讲，$A + B$ 是一种“纯粹构造性”的“或”(Or)：它的证据包含了哪个支节 (Disjunct) 为真的额外信息。有时这是非常有用的，但如果我们想要一种更经典的“或” (Or)，保持纯粹命题，我们需要一种方法通过忘记这些额外的信息来将此类型“截断”(Truncate) 成纯粹命题。

同样的问题出现在 $\Sigma$-类型 $\sum_{(x:A)} P(x)$ 中。这是“存在一个 $x : A$ 使得 $P(x)$ 为真”的纯粹构造性解释，它记住了见证 (Witness) $x$，因此即使每个类型 $P(x)$ 都是纯粹命题，它通常也不是纯粹命题。（回想在 §3.5 中我们观察到 $\sum_{(x:A)} P(x)$ 也可以被视为“那些 $x : A$ 满足 $P(x)$ 的子集”。）

## 3.7 **命题截断** (Propositional Truncation)

命题截断 (Propositional Truncation)，也称为 $(-1)$-截断 (Bracketing Type)、括号类型 (Bracket Type) 或 压缩类型 (Squash Type)，是一种附加类型生成器，它将类型“压缩”或“截断”到纯粹命题，忘记存在该类型的元素之外的所有信息。

更准确地说，对于任意类型 $A$，存在一个类型 $\|A\|$。它有两个构造器：

- 对于任意 $a : A$，我们有 $|a| : \|A\|$。
- 对于任意 $x, y : \|A\|$，我们有 $x = y$。

第一个构造器意味着，如果 $A$ 是被占用的，那么 $\|A\|$ 也是被占用的。第二个确保 $\|A\|$ 是一个纯粹命题；通常我们不为这个事实命名证据。

$\|A\|$ 的递归原理表明：

- 如果 $B$ 是一个纯粹命题，我们有 $f : A \to B$，那么存在一个诱导的 $g : \|A\| \to B$，使得对于所有 $a : A$，$g(|a|) \equiv f(a)$。

换句话说，任何从 $A$（的被占用性）得出的纯粹命题已经可以从 $\|A\|$ 得出。（也有一个 $\|A\|$ 的归纳原理，但它不是特别有用；见 Exercise 3.17。）

在 Exercises 3.14 and 3.15 and §6.9 中，我们将描述一些使用更一般的事物构造 $\|A\|$ 的方法。目前，我们简单地假设它作为 Chapter 1 规则之外的一个附加规则。

使用命题截断，我们可以扩展“纯粹命题逻辑”以涵盖析取 (Disjunction) 和存在量词 (Existential Quantifier)。具体来说，$\|A + B\|$ 是“$A$ 或 $B$”的纯粹命题版本，不会“记住”哪个支节 (Disjunct) 为真的信息。

截断的递归原理意味着我们仍然可以对 $\|A + B\|$ 进行情况分析 当试图证明一个纯粹命题时。也就是说，假设我们有一个 $u : \|A + B\|$ 并且我们试图证明一个纯粹命题 $Q$。换句话说，我们试图定义一个 $\|A + B\| \to Q$ 的元素。由于 $Q$ 是一个纯粹命题，根据命题截断的递归原理，我们只需构造一个 $A + B \to Q$ 的函数。但现在我们可以使用 $A + B$ 的情况分析。

类似地，对于类型族 $P : A \to \mathcal{U}$，我们可以考虑 $\left\|\sum_{(x:A)} P(x)\right\|$，这是“存在一个 $x : A$ 使得 $P(x)$”的纯粹命题版本。与析取一样，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，当试图证明一个纯粹命题时，我们可以引入新的假设 $x : A$ 和 $y : P(x)$。换句话说，如果我们知道存在一些 $x : A$ 使得 $P(x)$，但我们手头没有这样的特定 $x$，那么我们可以自由地使用这样的 $x$，只要我们不试图构造可能依赖于 $x$ 的特定值的任何东西。要求余类型是一个纯粹命题表达了结果对见证的独立性，因为该类型的所有可能的居民都必须是相等的。

在 Chapters 10 and 11 的集合级数学中，我们处理的大多数集合和纯粹命题时，使用传统的逻辑符号来仅仅指代“命题截断逻辑”是方便的。

Definition 3.7.1. 我们使用如下定义的 **传统逻辑符号** (Traditional Logical Notation) 使用截断如下，其中 $P$ 和 $Q$ 表示纯粹命题（或它们的族）：

$$\begin{aligned}
\top &:\equiv \mathbf{1} \\
\bot &:\equiv \mathbf{0} \\
P \wedge Q &:\equiv P \times Q \\
P \Rightarrow Q &:\equiv P \to Q \\
P \Leftrightarrow Q &:\equiv P = Q \\
\neg P &:\equiv P \to \mathbf{0} \\
P \vee Q &:\equiv \|P + Q\| \\
\forall (x : A).\, P(x) &:\equiv \prod_{x:A} P(x) \\
\exists (x : A).\, P(x) &:\equiv \left\|\sum_{x:A} P(x)\right\|
\end{aligned}$$

符号 $\wedge$ 和 $\vee$ 也用于同伦理论中表示指向空间的 smash 积和楔积 (Wedge Product)，我们将在 Chapter 6 中介绍。这从技术上讲可能会引起冲突，但通常不会引起混淆。

类似地，当讨论子集如 §3.5 中时，我们可以使用传统符号表示交集 (Intersection)、并集 (Union) 和补集 (Complement)：

$$\begin{gathered}
\{\, x : A \mid P(x) \,\} \cap \{\, x : A \mid Q(x) \,\} :\equiv \{\, x : A \mid P(x) \wedge Q(x) \,\} \\
\{\, x : A \mid P(x) \,\} \cup \{\, x : A \mid Q(x) \,\} :\equiv \{\, x : A \mid P(x) \vee Q(x) \,\} \\
A \setminus \{\, x : A \mid P(x) \,\} :\equiv \{\, x : A \mid \neg P(x) \,\}
\end{gathered}$$

当然，在没有 LEM 的情况下，后者不是通常意义上的“补集”：我们可能没有 $B \cup (A \setminus B) = A$ 对于每一个 $A$ 的子集 $B$。

## 3.8 选择公理 (The Axiom of Choice)

我们现在可以在同伦类型论 (Homotopy Type Theory) 中正确地表述选择公理 (Axiom of Choice)。假设有一个类型 $X$ 和类型族

$$A : X \to \mathcal{U} \qquad \text{以及} \qquad P : \prod_{x:X} A(x) \to \mathcal{U}$$

并且假设

- $X$ 是一个集合，
- 对所有 $x : X$，$A(x)$ 是一个集合，并且
- 对所有 $x : X$ 和 $a : A(x)$，$P(x,a)$ 是一个纯粹命题 (Mere Proposition)。

**选择公理** (Axiom of Choice) $\mathsf{AC}$ 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，我们将"存在 $x : A$ 使得 $B(x)$"读取为 $\left\|\sum_{(x:A)} B(x)\right\|$，因此我们可以将其表示为熟悉的逻辑符号

$$\Big(\forall(x : X).\, \exists(a : A(x)).\, P(x,a)\Big) \Rightarrow \Big(\exists(g : \textstyle\prod_{(x:X)} A(x)).\, \forall(x : X).\, P(x, g(x))\Big)$$

特别地，注意命题截断 (Propositional Truncation) 出现了两次。领域中的截断意味着我们假设对于每个 $x$，存在一些 $a : A(x)$ 使得 $P(x,a)$，但这些值没有被选择或以任何已知的方式指定。而共域中的截断意味着我们得出结论，存在某个函数 $g$，但该函数没有以任何已知的方式确定或指定。事实上，由于 Theorem 2.9.7，这个公理也可以以更简单的形式表达。

Lemma 3.8.2. *选择公理 (3.8.1) 等价于如下陈述：对于任意集合 $X$ 和任意 $Y : X \to \mathcal{U}$ 使得每个 $Y(x)$ 是一个集合，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 (Classical) 选择公理的一个众所周知的等价形式，即"一个非空集合族的笛卡尔积是非空的"。

*证明.* 由 Theorem 2.9.7，(3.8.1) 的共域等价于

$$\Big\| \prod_{(x:X)} \sum_{(a:A(x))} P(x,a) \Big\|$$

因此，(3.8.1) 等价于 (3.8.3) 的一个特例，即 $Y(x) :\equiv \sum_{(a:A(x))} P(x,a)$（这由 Example 3.1.5 and Lemma 3.3.4 证明是一个集合。）反之，(3.8.3) 等价于 (3.8.1) 的一个特例，即 $A(x) :\equiv Y(x)$ 和 $P(x,a) :\equiv \mathbf{1}$。因此，两者在逻辑上是等价的。由于它们都是纯粹命题 (Mere Proposition)，根据 Lemma 3.3.3 它们是等价类型。 □

与 $\mathsf{LEM}$ 一样，等价形式 (3.8.1) 和 (3.8.3) 不是我们基本类型论 (Basic Type Theory) 的结果，但可以作为公理一致地假设。

Remark 3.8.4. 很容易证明 (3.8.3) 的右侧总是意味着左侧。由于两者都是纯粹命题，根据 Lemma 3.3.3，选择公理也等价于要求等价性

$$\Big(\prod_{x:X} \Big\| Y(x) \Big\|\Big) \simeq \Big\| \prod_{x:X} Y(x) \Big\|$$

这说明了一个常见的陷阱：尽管依赖函数类型 (Dependent Function Type) 保持纯粹命题 (Example 3.6.2)，但它们与截断不交换：$\left\|\prod_{(x:A)} P(x)\right\|$ 一般不等价于 $\prod_{(x:A)} \|P(x)\|$。如果我们假设选择公理 (Axiom of Choice)，它表示这个陈述在集合上成立；正如我们将看到的，它在一般情况下是失败的。

选择公理中对作为集合的类型的限制可以在某种程度上放宽。例如，我们可以允许 (3.8.1) 中的 $A$ 和 $P$，或者 (3.8.3) 中的 $Y$，成为任意类型族；这会导致一个看似更强但同样一致的陈述。我们还可以将命题截断替换为将在 Chapter 7 中考虑的更一般的 $n$-截断 (n-Truncation)，得到一系列公理 $\mathsf{AC}_n$，在 (3.8.1)（我们简单地称之为 AC 或为了强调称之为 $\mathsf{AC}_{-1}$）和 Theorem 2.9.7（我们将其称为 $\mathsf{AC}_\infty$）之间插值。另见 Exercises 7.8 and 7.10。然而，请注意，我们不能放宽 $X$ 必须是集合的要求。

Lemma 3.8.5. *存在一个类型 $X$ 和一个族 $Y : X \to \mathcal{U}$，使得每个 $Y(x)$ 是一个集合，但使 (3.8.3) 失败。*

*证明.* 定义 $X :\equiv \sum_{(A:\mathcal{U})} \|\mathbf{2} = A\|$，并令 $x_0 :\equiv (\mathbf{2}, |\mathsf{refl}_{\mathbf{2}}|) : X$。然后通过 $\Sigma$-类型中的路径识别、$\|A = \mathbf{2}\|$ 是纯粹命题的事实，以及单一性 (Univalence)，对于任意 $(A,p),(B,q) : X$，我们有 $((A,p) =_X (B,q)) \simeq (A \simeq B)$。特别地，$(x_0 =_X x_0) \simeq (\mathbf{2} \simeq \mathbf{2})$，因此如 Example 3.1.9 所述，$X$ 不是一个集合。
另一方面，如果 $(A,p) : X$，那么 $A$ 是一个集合；这通过对 $p : \|\mathbf{2} = A\|$ 的截断进行归纳和 $\mathbf{2}$ 是集合的事实得出。由于 $A \simeq B$ 是集合当且仅当 $A$ 和 $B$ 是集合，因此对于任意 $x_1, x_2 : X$，$x_1 =_X x_2$ 是一个集合，即 $X$ 是 1-类型。特别地，如果我们通过 $Y : X \to \mathcal{U}$ 定义 $Y(x) :\equiv (x_0 = x)$，则每个 $Y(x)$ 是一个集合。
现在根据定义，对于任意 $(A,p) : X$，我们有 $\|\mathbf{2} = A\|$，因此我们有 $\|x_0 = (A,p)\|$。因此，我们有 $\prod_{(x:X)} \|Y(x)\|$。如果 (3.8.3) 对于此 $X$ 和 $Y$ 成立，那么我们也将有 $\left\|\prod_{(x:X)} Y(x)\right\|$。由于我们试图得出矛盾 ($\mathbf{0}$)，这是一个纯粹命题，因此我们可以假设 $\prod_{(x:X)} Y(x)$，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，这与前述结论矛盾。 □

## 3.9 唯一选择原则 (The Principle of Unique Choice)

以下观察虽然简单，但非常有用。

Lemma 3.9.1. *如果 $P$ 是一个纯粹命题 (Mere Proposition)，那么 $P \simeq \|P\|$。*

*证明.* 当然，我们根据定义有 $P \to \|P\|$。并且由于 $P$ 是一个纯粹命题，$\|P\|$ 的普遍性质应用于 $\mathsf{id}_P : P \to P$ 产生了 $\|P\| \to P$。通过 Lemma 3.3.3，这些函数是准逆的。 □

它的一个重要推论是如下定理。

Corollary 3.9.2 (唯一选择原则 (The Principle of Unique Choice)). *假设有一个类型族 $P : A \to \mathcal{U}$，使得*

(i) *对于每个 $x$，类型 $P(x)$ 是一个纯粹命题，并且*
(ii) *对于每个 $x$，我们有 $\|P(x)\|$。*

*那么我们有 $\prod_{(x:A)} P(x)$。*

*证明.* 直接由两个假设和前述引理得出。 □

该推论还概括了一个非常有用的推理技巧。也就是说，假设我们知道 $\|A\|$，并且我们想利用这一点来构造某个其他类型 $B$ 的元素。我们想要利用 $A$ 的一个元素来构造 $B$ 的一个元素，但这只有在 $B$ 是一个纯粹命题的情况下才允许，因此我们可以应用命题截断 $\|A\|$ 的归纳原则；在一般情况下，我们最多可以期望证明 $\|B\|$。
相反，我们可以通过附加数据扩展 $B$，这些数据唯一地描述我们希望构造的对象。具体来说，我们定义一个谓词 $Q : B \to \mathcal{U}$，使得 $\sum_{(x:B)} Q(x)$ 是一个纯粹命题。然后从 $A$ 的一个元素构造出一个 $b : B$ 使得 $Q(b)$ 成立，因此从 $\|A\|$ 我们可以构造出 $\left\|\sum_{(x:B)} Q(x)\right\|$，因为 $\left\|\sum_{(x:B)} Q(x)\right\|$ 等价于 $\sum_{(x:B)} Q(x)$，可以从中投射出一个 $B$ 的元素。一个例子可以在 Exercise 3.19 中找到。
在集合论 (Set-Theoretic) 数学中也会出现类似的问题，尽管表现方式略有不同。如果我们试图定义一个函数 $f : A \to B$，并且根据一个元素 $a : A$，我们能够证明某个 $b : B$ 的纯存在性，我们还没有完成任务，因为我们需要实际定位一个 $B$ 的元素，而不仅仅是证明它的存在。当然，一种选择是将论

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。但是在集合论中，这个问题通常可以通过应用选择公理 (Axiom of Choice) 更简单地避免，选择所需的元素。然而，在同伦类型论中 (Homotopy Type Theory)，除了避免选择的愿望外，可用的选择形式适用性更差，因为它们要求选择的领域 (Domain of Choice) 是一个集合。因此，如果 $A$ 不是一个集合（例如可能是一个宇宙 $\mathcal{U}$），那么没有一致的选择形式可以简单地为每个 $a : A$ 选择一个 $B$ 的元素来定义 $f(a)$。

## 3.10 命题何时被截断? (When Are Propositions Truncated?)

乍一看，$+$ 和 $\Sigma$ 的截断版本实际上比未截断的版本更接近非正式数学中“或”和“存在”的含义。当然，它们更接近于正式集合论 (Formal Set Theory) 背后的第一级逻辑 (First-Order Logic) 中“或”和“存在”的精确含义，因为后者并不试图记住命题的任何证据。然而，令人惊讶的是，非正式数学实践通常通过未截断形式更准确地描述。

例如，考虑“每个素数要么是 2，要么是奇数”这样的陈述。实际工作的数学家不会因为使用这一事实不仅仅是为了证明素数的*定理* (Theorem)，而且也用于对素数进行*构造* (Construction) 而感到愧疚，可能在 2 的情况下做一件事，而在奇数素数的情况下做另一件事。从类型论的角度来看，这样的构造自然地使用了“$(p = 2) + (p$ 是奇数$)$”的上积类型 (Coproduct Type) 的归纳原则，而不是它的命题截断。

诚然，这并不是一个理想的例子，因为“$p = 2$”和“$p$ 是奇数”是相互排斥的，因此 $(p = 2) + (p$ 是奇数$)$ 实际上已经是一个纯粹命题，因此等价于它的截断 (见 Exercise 3.7)。更有说服力的例子来自存在量词 (Existential Quantifier)。通常证明形式为“存在 $x$ 使得 …”的定理，然后在后续中引用“在定理 Y 中构造的 $x$”（请注意定冠词）。此外，当推导出该 $x$ 的进一步性质时，可以使用诸如“通过在定理 Y 的证明中构造的 $x$”之类的短语。

一个非常常见的例子是“$A$ 同构于 $B$”，这严格来说仅意味着存在*某个* $A$ 和 $B$ 之间的同构。但几乎不可避免地，在证明此类陈述时，人们会展示一个特定的同构，或者证明某个先前已知的映射是同构，通常后来引用时这个特定的同构至关重要。

受过集合论训练的数学家在使用此类“语言滥用 (Abuse of Language)”时通常会感到有些内疚。我们可能会试图为此道歉，在最终草稿中清除它们，或通过诸如“规范 (Canonical)”这样的模糊词来掩饰。这个问题由于在形式化集合论中，实际上没有任何方法可以“构造”对象—我们只能证明具有某些性质的对象存在而变得更加严重。类型论中的未截断逻辑 (Untruncated Logic) 因此捕捉到了非正式数学中一些常见的实践，而集合论的重构则掩盖了这些实践。（这类似于单一性公理 (Univalence Axiom) 验证了常见但在形式上没有理由的做法，即将同构对象视为相同。）

另一方面，有时截断的逻辑是必不可少的。我们已经在 LEM 和 AC 的陈述中看到过这种情况；本书后面还会出现其他一些例子。因此，我们面临的问题是：在编写非正式类型论时，“或”和“存在”等词的含义应该是什么（以及“有”、“我们有”等常见的同义词）？

普遍共识可能是不可能的。或许取决于所做的数学类型，某个约定可能更有用——或者，约定的选择可能无关紧要。在这种情况下，在数学论文的开头做一个注释，告知读者其中使用的语言约定可能就足够了。然而，即使选择了一种整体约定，另一种逻辑类型通常也会至少偶尔出现，因此我们需要一种方法来引用它。更普遍地说，人们可以考虑用另一种对类型表现类似的操作替代命题截断，如将类型 $A$ 映射为 $\neg\neg A$ 的双重否定操作，或者将在 Chapter 7 中考虑的 $n$-截断。作为说明的实验，在后续部分我们将偶尔使用*副词* (Adverbs) 来表示命题截断等“模态 (Modalities)”的应用。

例如，如果未截断逻辑是默认约定，我们可以使用副词“**仅仅** (Merely)”来表示命题截断。因此短语

$$\text{“仅仅存在一个 } x : A \text{ 使得 } P(x)\text{”}$$

表示类型 $\left\|\sum_{(x:A)} P(x)\right\|$。类似地，我们会说一个类型 $A$ 是**仅仅居住的** (Merely Inhabited) 来表示它的命题截断 $\|A\|$ 是居住的（即我们有一个未命名的元素）。请注意，这是一种对“仅仅”副词的*定义* (Definition)，它将在我们的非正式数学英语中使用，就像我们定义名词“群 (Group)”和“环 (Ring)”，以及形容词“正则 (Regular)”和“正态 (Normal)”具有精确的数学意义一样。我们并不声称“仅仅”这个词典定义指的是命题截断；选择这个词只是为了提醒读者，纯粹命题只包含“仅仅”的真值信息而已。

另一方面，如果截断逻辑是当前默认约定，我们可以使用“**纯粹** (Purely)”或“**构造性** (Constructively)”这样的副词来表示其缺失，因此

“纯粹存在一个 $x : A$ 使得 $P(x)$”

将表示类型 $\sum_{(x:A)} P(x)$。我们还可以使用“纯粹”或“实际上 (Actually)”仅仅为了强调截断的缺失，即使这是默认约定。

在本书中，我们将继续使用未截断逻辑作为默认约定，原因有很多。

(1) 我们希望鼓励新手尝试未截断逻辑，而不是仅仅因为它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设置会遇到与集合论基础相同的“语言滥用”问题，而未截断逻辑可以避免这些问题。例如，我们将“$A \simeq B$”定义为 $A$ 和 $B$ 之间的同构类型，而不是它的命题截断，这意味着证明形式为“$A \simeq B$”的定理实际上是构造一个特定的同构。然后可以在后续引用中使用这个特定的同构。

(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 Chapter 7 中看到的，纯粹命题只是无限阶梯上的第二阶梯，并且还有许多其他模态 (Modalities) 并不在这个阶梯上。

(4) 许多在经典上 (Classically) 是纯粹命题的陈述在同伦类型论中不再是如此。当然，其中最重要的是等式。

(5) 另一方面，同伦类型论中一个最有趣的观察是，令人惊讶的是许多类型是*自动地* (Automatically) 纯粹命题，或者可以通过稍微修改成为纯粹命题，无需进行任何截断。（见 Lemma 3.3.5 and Chapters 4, 7, 9 and 10。）因此，尽管这些类型不包含除真值之外的任何数据，但我们仍然可以使用它们构造未截断对象，因为不需要使用命题截断的归纳原则。如果默认对所有陈述应用命题截断，这个有用的事实就难以表达。

(6) 最后，截断对我们将在本书中进行的大部分数学研究没有太大用处，因此更简单的做法是在它们出现时明确地记述它们。

## 3.11 **收缩性** (Contractibility)

在 Lemma 3.3.2 中，我们观察到一个居住的纯粹命题必须等价于 $\mathbf{1}$，并且很容易看出，反之亦然。具有这种性质的类型称为*收缩* (Contractible)。另一个等价的收缩性定义，有时也很方便，是如下的定义。

Definition 3.11.1. 类型 $A$ 是**收缩的** (Contractible)，或**单点** (Singleton) 类型，如果存在 $a : A$，称为**收缩中心** (Center of Contraction)，使得对于所有 $x : A$ 都有 $a = x$。我们用 $\mathsf{contr}_x$ 表示指定的路径 $a = x$。

换句话说，$\mathsf{isContr}(A)$ 类型定义为

$$\mathsf{isContr}(A) :\equiv \sum_{(a:A)} \prod_{(x:A)} (a = x).$$

注意，在通常的命题即类型 (Propositions-as-Types) 解读下，我们可以将 $\mathsf{isContr}(A)$ 读作“$A$ 恰好包含一个元素”，或者更精确地说“$A$ 包含一个元素，并且 $A$ 的每个元素都等于那个元素”。

Remark 3.11.2. 我们也可以将 $\mathsf{isContr}(A)$ 更拓扑地 (Topologically) 读作“存在一个点 $a : A$，使得对于所有 $x : A$ 存在从 $a$ 到 $x$ 的路径”。注意，对于经典的 (Classical) 耳朵来说，这听起来更像是*连通性* (Connectedness) 的定义，而不是收缩性。关键是，这句话中的“存在”的含义是一个连续/自然的存在。表达连通性更好的方式是 $\sum_{(a:A)} \prod_{(x:A)} \|a = x\|$。如果假设 $A$ 是有基点的 (Pointed)，这确实是正确的——参见 Lemma 7.5.11 之后的备注——但一般来说，一个类型可以是连通的而不一定有基点。在 §7.5 中，我们将连通性定义为一般 $n$-连通性概念的 $n = 0$ 情况，并且在 Exercise 7.6 中，读者被要求展示这一定义等价于同时具有 $\|A\|$ 和 $\prod_{(x,y:A)} \|x = y\|$。

Lemma 3.11.3. *对于类型 $A$，以下条件在逻辑上等价。*

(i) *$A$ 是 Definition 3.11.1 中定义的收缩性类型。*

(ii) $A$ 是一个纯粹命题，并且存在一个点 $a : A$。

(iii) $A$ 等价于 $\mathbf{1}$ 。

证明. 如果 $A$ 是收缩的，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。反之，如果我们有 $a : A$ 并且 $A$ 是一个纯粹命题，那么对于任何 $x : A$，我们有 $x = a$；因此 $A$ 是收缩的。我们在 Lemma 3.3.2 中展示了 (ii)⇒(iii)，而反之亦然，因为 $\mathbf{1}$ 很容易具有性质 (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 $a, a' : A$ 并且 $p : \prod_{(x:A)}(a = x)$ 和 $p' : \prod_{(x:A)}(a' = x)$。通过 $\Sigma$-类型中路径的表征，展示 $c = c'$ 就足以展示一个 $q : a = a'$ 使得 $q_*(p) = p'$。我们选择 $q :\equiv p(a')$。现在，由于 $A$ 是收缩的（通过 $c$ 或 $c'$），根据 Lemma 3.11.3，它是一个纯粹命题。因此，根据 Lemma 3.3.4 and Example 3.6.2，$\prod_{(x:A)}(a' = x)$ 也是纯粹命题；因此 $q_*(p) = p'$ 是自动成立的。 □

Corollary 3.11.5. 如果 $A$ 是收缩的，那么 $\mathsf{isContr}(A)$ 也是收缩的。

证明. 根据 Lemma 3.11.4 和 Lemma 3.11.3(ii)。 □

像纯粹命题一样，收缩类型在许多类型构造器下也是保持的。例如，我们有：

Lemma 3.11.6. 如果 $P : A \to \mathcal{U}$ 是一个类型族，并且每个 $P(a)$ 是收缩的，那么 $\prod_{(x:A)} P(x)$ 是收缩的。

证明. 根据 Example 3.6.2，$\prod_{(x:A)} P(x)$ 是一个纯粹命题，因为每个 $P(x)$ 都是。但它也有一个元素，即将每个 $x : A$ 映射到 $P(x)$ 的收缩中心的函数。因此根据 Lemma 3.11.3(ii)，$\prod_{(x:A)} P(x)$ 是收缩的。 □

（事实上，Lemma 3.11.6 的陈述等价于函数外延性公理。参见 §4.9。）

当然，如果 $A$ 等价于 $B$ 并且 $A$ 是收缩的，那么 $B$ 也是。更一般地说，只要 $B$ 是 $A$ 的**重载** (Retract)。根据定义，**重载** (Retraction) 是一个函数 $r : A \to B$，使得存在一个函数 $s : B \to A$，称为它的**截面** (Section)，以及一个同伦 $\epsilon : \prod_{(y:B)}(r(s(y)) = y)$；然后我们说 $B$ 是 $A$ 的**重载**。

Lemma 3.11.7. 如果 $B$ 是 $A$ 的重载，并且 $A$ 是收缩的，那么 $B$ 也是收缩的。

证明. 令 $a_0 : A$ 为收缩中心。我们声明 $b_0 :\equiv r(a_0) : B$ 是 $B$ 的收缩中心。令 $b : B$；我们需要一个路径 $b = b_0$。但我们有 $\epsilon_b : r(s(b)) = b$ 和 $\mathsf{contr}_{s(b)} : s(b) = a_0$，因此通过合成

$$\epsilon_b{}^{-1} \cdot r\big(\mathsf{contr}_{s(b)}\big) : b = r(a_0) \equiv b_0.$$

□

收缩类型可能看起来不是很有趣，因为它们都等价于 $\mathbf{1}$ 。该概念有用的一个原因是，有时一些单独的非平凡数据集合将共同形成一个收缩类型。一个重要的例子是具有一个自由端点的路径空间。正如我们将在 §5.8 中看到的，这个事实本质上概括了基于路径的恒等类型归纳原则。

Lemma 3.11.8. 对于任何 $A$ 和任何 $a : A$，类型 $\sum_{(x:A)}(a = x)$ 是收缩的。

证明. 我们选择点 $(a, \mathsf{refl}_a)$ 作为收缩中心。现在假设 $(x, p) : \sum_{(x:A)}(a = x)$；我们必须展示 $(a, \mathsf{refl}_a) = (x, p)$。通过 $\Sigma$-类型中路径的表征，展示 $q : a = x$ 就足以展示 $q_*(\mathsf{refl}_a) = p$。但我们可以取 $q :\equiv p$，在这种情况下，通过路径类型中的传输表征，$q_*(\mathsf{refl}_a) = p$ 是成立的。 □

当这种情况发生时，它可以允许我们简化一个复杂的构造直到等价，使用非正式原则，即收缩数据可以自由忽略。该原则由许多引理组成，其中大部分我们留给读者，下面是一个例子。

Lemma 3.11.9. 令 $P : A \to \mathcal{U}$ 为一个类型族。

(i) 如果每个 $P(x)$ 是收缩的，那么 $\sum_{(x:A)} P(x)$ 等价于 $A$。

(ii) 如果 $A$ 是以 $a$ 为中心的收缩类型，那么 $\sum_{(x:A)} P(x)$ 等价于 $P(a)$。

证明. 在 (i) 的情况下，我们展示 $\mathsf{pr}_1 : \sum_{(x:A)} P(x) \to A$ 是等价的。对于逆函数，我们定义 $g(x) :\equiv (x, c_x)$，其中 $c_x$ 是 $P(x)$ 的收缩中心。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 $P(x)$ 的收缩同伦到恒等式。
我们将 (ii) 的证明留给读者（参见 Exercise 3.20）。 □

另一个收缩类型有趣的原因是它们将 §3.1 中提到的 $n$-类型阶梯向下延伸了一级。

Lemma 3.11.10. 类型 $A$ 是一个纯粹命题，当且仅当对于所有 $x, y : A$，类型 $x =_A y$ 是收缩的。

证明. 对于"如果"，我们只是观察到任何收缩类型都是居住的。对于"仅当"，我们在 §3.3 中观察到每个纯粹命题都是集合，因此每个类型 $x =_A y$ 是一个纯粹命题。但它也是居住的（因为 $A$ 是一个纯粹命题），因此根据 Lemma 3.11.3(ii) 它是收缩的。 □

因此，收缩类型也可以称为 $(-2)$-**类型**。它们是 $n$-类型阶梯的最低阶，将成为我们在 Chapter 7 中递归定义 $n$-类型的基准。

## Notes

Voevodsky 首先观察到，在类型论中可以定义集合、纯粹命题和收缩类型，如 §§3.1, 3.3 and 3.11 中的所有高级同伦自动处理。事实上，他通过归纳定义了整个 $n$-类型阶层，就像我们将在 Chapter 7 中做的那样。

Theorem 3.2.2 and Corollary 3.2.7 本质上依赖于 Hedberg 的经典定理，我们将在 §7.2 中证明。命题即类型形式的 LEM 与单值性矛盾的含义是由 Martín Escardó 在 Agda 邮件列表中观察到的。我们给出的 Theorem 3.2.2 的证明是 Thierry Coquand 提出的。

命题截断首次出现在 NuPRL 的扩展类型论中，是由 Constable 在 1983 年提出的 [?]，作为"子集 (Subset)"和"商集 (Quotient)"类型的应用。这里称为"命题截断"的东西在 NuPRL 类型论 [?] 中被称为"压缩 (Squashing)"。在 [?] 中给出了直接表征命题截断的规则，仍在扩展类型论中。同伦类型论中的内涵版本是由 Voevodsky 使用不定型量化构建的，后来 Lumsdaine 使用更高的归纳类型构建（参见 §6.9）。

Voevodsky [?] 提出了 §3.5 中考虑的那种缩放规则。这些显然与 Russell 在他的《数学原理》中提出的臭名昭著的可缩减性公理有关 [?]。

副词"纯粹"用于指代未截断逻辑，是对在编程语言中使用模态模态 (Monadic Modalities) 模型效应的引用；参见 §7.7 和 Chapter 7 的笔记。

关于逻辑相对于类型论的处理方式，有许多不同的方式。例如，除了 §1.11 中描述的普通命题即类型逻辑和 §3.6 中描述的仅使用纯粹命题的替代方法外，还可以引入一个单独的"排序"命题，它们的行为类似于类型，但不与类型同一。这是逻辑丰富类型论 [?] 以及一些拓扑学 (Toposes) 和相关范畴内部语言展示中的方法（例如 [?, ?]），以及证明助手 Coq. 这种方法更为一般，但不那么强大。例如，Coq 中所谓的 Setoid 范畴中独特选择原则 (§3.9) 失败了 [?]，在逻辑丰富类型论中也是如此 [?]，在最小类型论中也是如此 [?]。因此，单值性公理使我们的类型论更像是拓扑的内部逻辑；见 Chapter 10。

Martin-Löf [?] 提供了关于选择公理历史的讨论。当然，构造性和直觉主义数学有着悠久而复杂的历史，我们在这里不会深入探讨；例如参见 [?, ?]。

## Exercises

Exercise 3.1. 证明如果 $A \simeq B$ 并且 $A$ 是集合，那么 $B$ 也是集合。

Exercise 3.2. 证明如果 $A$ 和 $B$ 是集合，那么 $A + B$ 也是集合。

Exercise 3.3. 证明如果 $A$ 是集合并且 $B : A \to \mathcal{U}$ 是一个类型族，使得 $B(x)$ 对所有 $x : A$ 是集合，那么 $\sum_{(x:A)} B(x)$ 是集合。

Exercise 3.4. 证明 $A$ 是一个纯粹命题，当且仅当 $A \to A$ 是收缩的。

Exercise 3.5. 证明 $\mathsf{isProp}(A) \simeq (A \to \mathsf{isContr}(A))$。

Exercise 3.6. 证明如果 $A$ 是一个单纯命题（mere proposition），那么 $A + (\neg A)$ 也是一个单纯命题。因此，在 (3.4.1) 中不需要插入命题截断（propositional truncation）。

Exercise 3.7. 更一般地，证明如果 $A$ 和 $B$ 是单纯命题，并且 $\neg(A \times B)$，那么 $A + B$ 也是一个单纯命题。

Exercise 3.8. 假设某个类型 $\mathsf{isequiv}(f)$ 满足 §2.4 中的条件 (i)–(iii)，证明类型 $\|\mathsf{qinv}(f)\|$ 满足相同的条件并且与 $\mathsf{isequiv}(f)$ 等价。

Exercise 3.9. 证明如果 LEM 成立，那么类型 $\mathsf{Prop} :\equiv \sum_{(A:\mathcal{U})} \mathsf{isProp}(A)$ 与 $\mathbf{2}$ 等价。

Exercise 3.10. 证明如果 $\mathcal{U}_{i+1}$ 满足 LEM ，那么 $\mathsf{Prop}_{\mathcal{U}_i} \to \mathsf{Prop}_{\mathcal{U}_{i+1}}$ 的典范包含是一个等价。

Exercise 3.11. 证明并非对于所有类型 $A : \mathcal{U}$ 我们都有 $\|A\| \to A$。（然而，确实存在某些特定的类型使得 $\|A\| \to A$ 成立。Exercise 3.8 表明 $\mathsf{qinv}(f)$ 就是其中之一。）

Exercise 3.12. 证明如果 LEM 成立，那么对于所有类型 $A : \mathcal{U}$ 我们有 $\big\|(\|A\| \to A)\big\|$。（这种性质是选择公理的一个非常简单的形式，在没有 LEM 的情况下可能不成立；参见 [?]。）

Exercise 3.13. 我们在 Corollary 3.2.7 中证明了以下简单形式的 LEM 与单值性（univalence）不一致：

$$\prod_{A:\mathcal{U}} (A + (\neg A))$$

在没有单值性的情况下，这个公理是一致的。然而，证明它暗含了选择公理 (3.8.1)。

Exercise 3.14. 证明假设 LEM ，双重否定 $\neg\neg A$ 具有与命题截断 $\|A\|$ 相同的递归原则，但具有命题计算规则而不是判断规则。换句话说，证明假设 LEM ，如果 $B$ 是一个单纯命题并且我们有 $f : A \to B$，那么有一个诱导的 $g : \neg\neg A \to B$，使得对所有 $a : A$ 都有 $g(|a|) = f(a)$。推导出（假设 LEM ）我们有 $\neg\neg A \simeq \|A\|$。因此，在 LEM 下，命题截断可以定义出来，而不是作为一个单独的类型构造。

Exercise 3.15. 证明如果我们假设如 §3.5 中的命题调整，那么类型

$$\prod_{P:\mathsf{Prop}} \Big((A \to P) \to P\Big)$$

具有与 $\|A\|$ 相同的递归原则，并且具有相同的判断计算规则。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 ，证明双重否定与集上的单纯命题的全称量化是可交换的。也就是说，证明如果 $X$ 是一个集并且每个 $Y(x)$ 都是一个单纯命题，那么 LEM 暗含

$$\Big(\prod_{x:X} \neg\neg Y(x)\Big) \simeq \Big(\neg\neg \prod_{x:X} Y(x)\Big) \tag{3.11.11}$$

请注意，如果我们改为假设每个 $Y(x)$ 是一个集，那么 (3.11.11) 变得等价于选择公理 (3.8.3)。

Exercise 3.17. 证明 §3.7 中给出的命题截断的规则足以暗含以下归纳原则：对于任意类型族 $B : \|A\| \to \mathcal{U}$，如果每个 $B(x)$ 都是一个单纯命题，并且对于每个 $a : A$ 我们都有 $B(|a|)$，那么对于每个 $x : \|A\|$ 我们都有 $B(x)$。

Exercise 3.18. 证明中间排律 (3.4.1) 和双重否定律 eq:ldn 在逻辑上是等价的。

Exercise 3.19. 假设 $P : \mathbb{N} \to \mathcal{U}$ 是一个可判定的族（参见 Definition 3.4.3(ii)）的单纯命题。证明

$$\Big\|\sum_{n:\mathbb{N}} P(n)\Big\| \to \sum_{n:\mathbb{N}} P(n)$$

Exercise 3.20. 证明 Lemma 3.11.9(ii)：如果 $A$ 是以 $a$ 为中心的可收缩集（contractible），那么 $\sum_{(x:A)} P(x)$ 与 $P(a)$ 是等价的。

Exercise 3.21. 证明 $\mathsf{isProp}(P) \simeq (P \simeq \|P\|)$。

Exercise 3.22. 正如经典集合论中那样，有限版本的选择公理是一个定理。证明当 $X$ 是一个有限类型 $\mathsf{Fin}(n)$（如 Exercise 1.9 中定义的）时，选择公理 (3.8.1) 成立。

Exercise 3.23. 证明 Exercise 3.19 的结论对于任意可判定族 $P : \mathbb{N} \to \mathcal{U}$ 也成立。

Exercise 3.24. 通过首先证明对于所有 $m, n$，$\mathsf{code}(m, n)$ 是一个单纯命题来简化 ?? 的证明。

Chapter 4

# 等 价 性 (Equivalences)

我们现在详细研究在 §2.4中简要介绍的*类型的等价性* (equivalence of types) 概念。具体来说，我们将给出几种不同的方式来定义具有前述特性的一种类型 $\mathsf{isequiv}(f)$。回想一下，我们希望 $\mathsf{isequiv}(f)$ 具有以下性质，在此处我们再次陈述这些性质：

(i) $\mathsf{qinv}(f) \to \mathsf{isequiv}(f)$。
(ii) $\mathsf{isequiv}(f) \to \mathsf{qinv}(f)$。
(iii) $\mathsf{isequiv}(f)$ 是一个纯命题 (mere proposition)。

这里 $\mathsf{qinv}(f)$ 表示 $f$ 的准逆 (quasi-inverse) 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 (function extensionality)，这意味着 $\mathsf{qinv}(f)$ 等价于类型

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

我们将定义三种具有性质 (i)–(iii) 的类型，分别称为：

- 半伴随等价 (half adjoint equivalences)，
- 双可逆映射 (bi-invertible maps)，和
- 可缩函数 (contractible functions)。

我们还将证明这些类型是等价的。这些名称特意显得有些繁琐，因为在我们知道它们都是等价的并具有性质 (i)–(iii)之后，我们会简单地使用“等价性 (equivalence)”这个词，而不需要指定我们选择了哪种特定定义。但为了本章中的比较目的，我们需要为每个定义提供不同的名称。
在我们研究等价性的不同概念之前，我们先稍微解释一下为什么需要一个不同于准可逆性的概念。

## 4.1 **准逆** (Quasi-inverses)

我们已经提到 $\mathsf{qinv}(f)$ 是不令人满意的，因为它不是一个纯命题，而我们希望一个给定的函数最多以一种方式“成为等价的”。然而，我们还没有证明 $\mathsf{qinv}(f)$ 不是一个纯命题。在本节中，我们展示一个具体的反例。

Lemma 4.1.1. *如果* $f : A \to B$ *是使得* $\mathsf{qinv}(f)$ *可居住的* (inhabited)，*那么*

$$\mathsf{qinv}(f) \simeq \Big(\prod_{x:A} (x = x)\Big)$$

证明. 根据假设，$f$ 是一个等价性，也就是说我们有 $e : \mathsf{isequiv}(f)$，因此 $(f,e) : A \simeq B$。根据单值性 (univalence)，$\mathsf{idtoeqv} : (A = B) \to (A \simeq B)$ 是一个等价性，因此我们可以假设 $(f,e)$ 的形式为 $\mathsf{idtoeqv}(p)$，其中 $p : A = B$。然后根据路径归纳 (path induction)，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 Exercise 2.10，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 Lemma 3.11.8，$\sum_{(g:A\to A)}(g = \mathsf{id}_A)$ 是以 $(\mathsf{id}_A, \mathsf{refl}_{\mathsf{id}_A})$ 为中心的可缩 (contractible) 类型；因此根据 Lemma 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 Exercise 4.3 要求在避免单值性的情况下证明上述引理。
因此，我们需要一些 $A$，它允许 $\prod_{(x:A)}(x = x)$ 的非平凡 (nontrivial) 元素。将 $A$ 视为一个更高阶的群胚 (higher groupoid)，$\prod_{(x:A)}(x = x)$ 的一个居留元素 (inhabitant) 是从 $A$ 的恒等函子 (identity functor) 到其自身的自然变换 (natural transformation)。这样的变换被称为一个范畴的**中心** (center of a category)，因为自然性公理要求它们与所有态射 (morphisms) 交换。传统上，如果 $A$ 只是一个被视为单对象群胚 (one-object groupoid) 的群体，那么这将恰好得出通常群论意义上的中心。这为以下内容提供了一些动机。

Lemma 4.1.2. *假设我们有一个类型 $A$ 和 $a : A$ 以及 $q : a = a$，使得*

(i) *类型 $a = a$ 是一个集合 (set)。*
(ii) *对于所有 $x : A$，我们有 $\|a = x\|$。*
(iii) *对于所有 $p : a = a$，我们有 $p \cdot q = q \cdot p$。*

*那么存在 $f : \prod_{(x:A)}(x = x)$，且有 $f(a) = q$。*

证明. 令 $g : \prod_{(x:A)}\|a = x\|$ 为 (ii)给出的值。首先我们观察到每个类型 $x =_A y$ 是一个集合。因为作为一个集合是一个纯命题，我们可以应用命题截断 (propositional truncation) 的归纳原则，假设 $g(x) = |p|$ 且 $g(y) = |p'|$，其中 $p : a = x$ 和 $p' : a = y$。在这种情况下，与 $p$ 和 $p'^{-1}$ 组成的映射等价于 $(x = y) \simeq (a = a)$。但由于 (i) 中 $(a = a)$ 是一个集合，因此 $(x = y)$ 也是一个集合。
现在，我们希望通过为每个 $x$ 分配路径 $g(x)^{-1} \cdot q \cdot g(x)$ 来定义 $f$，但这不奏效，因为 $g(x)$ 不属于 $a = x$，而是 $\|a = x\|$，并且类型 $(x = x)$ 可能不是一个纯命题，因此我们不能使用命题截断的归纳法。相反，我们可以应用 §3.9 中提到的技巧：我们唯一地刻画我们希望构造的对象。让我们定义，对于每个 $x : A$，类型

$$B(x) :\equiv \sum_{(r:x=x)} \prod_{(s:a=x)} (r = s^{-1} \cdot q \cdot s)$$

我们声称每个 $B(x)$ 是一个纯命题。由于这个声明本身是一个纯命题，我们可以再次应用命题截断的归纳法，并假设 $g(x) = |p|$，其中 $p : a = x$。现在假设给定 $(r,h)$ 和 $(r',h')$ 在 $B(x)$ 中；然后我们有

$$h(p) \cdot h'(p)^{-1} : r = r'$$

剩下的是展示，当沿着这个等式传递时，$h$ 与 $h'$ 相同，这通过在恒等类型和函数类型中传输 (?? and §2.7)，减少为展示

$$h(s) = h(p) \cdot h'(p)^{-1} \cdot h'(s)$$

对于任何 $s : a = x$。但这两侧都是 $(x = x)$ 的元素之间的等式，因此它遵循我们之前的观察，即 $(x = x)$ 是一个集合。
因此，每个 $B(x)$ 是一个纯命题；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法，假设 $g(x) = |p|$，其中 $p : a = x$。我们定义 $r :\equiv p^{-1} \cdot q \cdot p$；为了填充 $B(x)$，剩下的是展示对于任何 $s : a = x$ 我们有 $r = s^{-1} \cdot q \cdot s$。操作路径，这减少为展示 $q \cdot (p \cdot s^{-1}) = (p \cdot s^{-1}) \cdot q$。但这只是 (iii) 的一个实例。 □

Theorem 4.1.3. *存在类型 $A$ 和 $B$ 以及一个函数 $f : A \to B$ 使得 $\mathsf{qinv}(f)$ 不是一个纯命题。*

*证明.* 这足以展示一个类型 $X$，使得 $\prod_{(x:X)}(x = x)$ 不是一个纯命题。定义 $X :\equiv \sum_{(A:\mathcal{U})} \|\mathbf{2} = A\|$，如在 Lemma 3.8.5 的证明中所述。展示一个 $f : \prod_{(x:X)}(x = x)$，它不同于 $\lambda x.\, \mathsf{refl}_x$ 即可。
令 $a :\equiv (\mathbf{2}, |\mathsf{refl}_{\mathbf{2}}|) : X$，并令 $q : a = a$ 是对应于非恒等的等价性 $e : \mathbf{2} \simeq \mathbf{2}$ 的路径，其定义为 $e(0_{\mathbf{2}}) :\equiv 1_{\mathbf{2}}$ 和 $e(1_{\mathbf{2}}) :\equiv 0_{\mathbf{2}}$。我们希望应用 Lemma 4.1.2 来构建一个 $f$。根据 $X$ 的定义，子集类型中的等式 (§3.5) 和单值性，我们有 $(a = a) \simeq (\mathbf{2} \simeq \mathbf{2})$，这是一个集合，因此 (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，Exercise 2.13 表明每个等价性 $\mathbf{2} \simeq \mathbf{2}$ 都等于 $\mathsf{id}_{\mathbf{2}}$ 或 $e$，因此我们可以通过四种情况分析展示 (iii)。
因此，我们有 $f : \prod_{(x:X)}(x = x)$，且有 $f(a) = q$。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，$\prod_{(x:X)}(x = x)$ 不是一个纯命题。 □

更普遍地，Lemma 4.1.2 表明任何"Eilenberg–Mac Lane 空间 (Eilenberg–Mac Lane space)"$K(G, 1)$，其中 $G$ 是一个非平凡的阿贝尔 (abelian) 群体，将提供一个反例；参见 Chapter 8。我们使用的类型 $X$ 结果是等价于 $K(\mathbb{Z}_2, 1)$。在 Chapter 6 中，我们将看到圆 $S^1 = K(\mathbb{Z}, 1)$ 是另一个易于描述的例子。
我们现在继续描述更好的等价性概念。

## 4.2 半伴随等价 (Half adjoint equivalences)

在 §4.1 中，我们得出结论认为 $\mathsf{qinv}(f)$ 通过丢弃一个可缩类型等价于 $\prod_{(x:A)}(x = x)$。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 $\eta$) 可以一起在 $f$ 是等价性时被视为可缩的。问题在于去掉这些数据后还剩下一个 ($\epsilon$)。为了解决这个问题，想法是增加一个*额外的*数据，这样 $\epsilon$ 和它一起构成一个可缩类型。

Definition 4.2.1. 一个函数 $f : A \to B$ 是一个**半伴随等价** (half adjoint equivalence)，如果存在 $g : B \to A$ 和同伦 $\eta : g \circ f \sim \mathsf{id}_A$ 和 $\epsilon : f \circ g \sim \mathsf{id}_B$，使得存在一个同伦

$$\tau : \prod_{x:A} f(\eta x) = \epsilon(fx)$$

因此我们定义类型 $\mathsf{ishae}(f)$ 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，$\eta$ 和 $\epsilon$ 的一致性条件 (coherence condition) 仅涉及 $f$。我们可以考虑一个涉及 $g$ 的类似一致性条件：

$$v : \prod_{y:B} g(\epsilon y) = \eta(gy)$$

并得到一个类似的定义 $\mathsf{ishae}'(f)$。
幸运的是，事实证明这些条件中的每一个都意味着另一个：

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 $\epsilon : f \circ g \sim \mathsf{id}_B$，以下条件是逻辑等价的：*

- $\prod_{(x:A)} f(\eta x) = \epsilon(fx)$

- $\prod_{(y:B)} g(\epsilon y) = \eta(gy)$

证明. 证明一个方向就足够了；另一个方向通过替换 $A$、$f$ 和 $\eta$ 为 $B$、$g$ 和 $\epsilon$ 分别得到。设 $\tau : \prod_{(x:A)} f(\eta x) = \epsilon(fx)$。固定 $y : B$。使用 $\epsilon$ 的自然性 (naturality) 并应用 $g$，我们得到以下路径的交换图：

$$\begin{array}{ccc} gfgfgy & \xrightarrow{gfg(\epsilon y)} & gfgy \\ {\scriptstyle g(\epsilon(fgy))}\Big\| & & \Big\|{\scriptstyle g(\epsilon y)} \\ gfgy & \xrightarrow[g(\epsilon y)]{} & gy \end{array}$$

在图的左侧使用 $\tau(gy)$ 得到

$$\begin{array}{ccc} gfgfgy & \xrightarrow{gfg(\epsilon y)} & gfgy \\ {\scriptstyle gf(\eta(gy))}\Big\| & & \Big\|{\scriptstyle g(\epsilon y)} \\ gfgy & \xrightarrow[g(\epsilon y)]{} & gy \end{array}$$

使用 $\eta$ 与 $g \circ f$ 的交换 (Corollary 2.4.4)，我们有

$$\begin{array}{ccc} gfgfgy & \xrightarrow{gfg(\epsilon y)} & gfgy \\ {\scriptstyle \eta(gfgy)}\Big\| & & \Big\|{\scriptstyle g(\epsilon y)} \\ gfgy & \xrightarrow[g(\epsilon y)]{} & gy \end{array}$$

然而，根据 $\eta$ 的自然性，我们也有

$$\begin{array}{ccc} gfgfgy & \xrightarrow{gfg(\epsilon y)} & gfgy \\ {\scriptstyle \eta(gfgy)}\Big\| & & \Big\|{\scriptstyle \eta(gy)} \\ gfgy & \xrightarrow[g(\epsilon y)]{} & gy \end{array}$$

因此，取消所有但右侧的同伦，我们得到 $g(\epsilon y) = \eta(gy)$ 如所期望的。 □

然而，重要的是我们不要在 $\mathsf{ishae}(f)$ 的定义中包含两者 $\tau$ 和 $v$（因此称之为“半伴随等价 (half adjoint equivalence)”）。如果我们这么做，那么在取消可缩类型后，我们仍然会剩下一个数据——除非我们添加另一个更高的一致性条件。一般来说，如果我们在一个奇数个一致性条件之后切断，我们预期会得到一个良好行为的类型。

当然，很明显 $\mathsf{ishae}(f) \to \mathsf{qinv}(f)$：只需忘记一致性数据。另一个方向是同伦理论 (homotopy theory) 和范畴论 (category theory) 中的标准论点的一种版本。

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 $f$ 的准逆 (quasi-inverse)。我们必须提供一个四元组 $(g', \eta', \epsilon', \tau)$ 来证明 $f$ 是一个半伴随等价 (half adjoint equivalence)。为了定义 $g'$ 和 $\eta'$，我们可以选择显而易见的选择，将 $g'$g$\eta'\eta$。然而，在 $\epsilon'$ 的定义中，我们需要开始考虑 $\tau$ 的构造，因此不能直接按照常规选择 $\epsilon'$ 为 $\epsilon$。相反，我们选择

$$\epsilon'(b) :\equiv \epsilon(f(g(b)))^{-1} \cdot (f(\eta(g(b))) \cdot \epsilon(b))$$

现在我们需要找到

$$\tau(a) : f(\eta(a)) = \epsilon(f(g(f(a))))^{-1} \cdot (f(\eta(g(f(a)))) \cdot \epsilon(f(a)))$$

首先注意到，根据 Corollary 2.4.4，我们有 Lemma 2.4.3 计算

$$\begin{aligned} f(\eta(g(f(a)))) \cdot \epsilon(f(a)) &= f(g(f(\eta(a)))) \cdot \epsilon(f(a)) \\ &= \epsilon(f(g(f(a)))) \cdot f(\eta(a)) \end{aligned}$$

从而得到所需的路径 $\tau(a)$。 □

结合 Lemma 4.2.2（或对称化的证明），我们也有 $\mathsf{qinv}(f) \to \mathsf{ishae}'(f)$。
剩下的是展示 $\mathsf{ishae}(f)$ 是一个纯命题。为此，我们需要知道等价性的纤维 (fiber) 是可缩的。

Definition 4.2.4. 函数 $f : A \to B$ 在点 $y : B$ 上的**纤维** (fiber) 定义为

$$\mathsf{fib}_f(y) :\equiv \sum_{x:A} (f(x) = y)$$

在同伦理论中，这被称为 $f$ 的同伦纤维 (homotopy fiber)。§2.5 中的路径引理得出以下关于纤维中路径的刻画：

Lemma 4.2.5. 对于任何 $f : A \to B$、$y : B$ 以及 $(x, p), (x', p') : \mathsf{fib}_f(y)$，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，那么对于任何 $y : B$，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 $\mathsf{fib}_f(y)$ 的缩并中心 (center of contraction)，我们选择 $(gy, \epsilon y)$。现在取任何 $(x, p) : \mathsf{fib}_f(y)$；我们要构造从 $(gy, \epsilon y)$ 到 $(x, p)$ 的路径。根据 Lemma 4.2.5，足以给出一个路径 $\gamma : gy = x$，使得 $f(\gamma) \cdot p = \epsilon y$。我们取 $\gamma :\equiv g(p)^{-1} \cdot \eta x$。然后我们有

$$\begin{aligned} f(\gamma) \cdot p &= fg(p)^{-1} \cdot f(\eta x) \cdot p \\ &= fg(p)^{-1} \cdot \epsilon(fx) \cdot p \\ &= \epsilon y \end{aligned}$$

其中第二个等式由 $\tau x$ 得到，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。以下类型将准逆 $g$ 与其中一个同伦结合起来。

Definition 4.2.7. 给定一个函数 $f : A \to B$，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** (left inverses) 和**右逆** (right inverses)。如果 $\mathsf{linv}(f)$ 可居住，我们称 $f$ 是**左可逆** (left invertible) 的，如果 $\mathsf{rinv}(f)$ 可居住，我们称 $f$ 是**右可逆** (right invertible) 的。

Lemma 4.2.8. 如果 $f : A \to B$ 有一个准逆，那么以下映射也有准逆：

$$\begin{aligned} (f \circ -) &: (C \to A) \to (C \to B) \\ (- \circ f) &: (B \to C) \to (A \to C) \end{aligned}$$

证明. 如果 $g$ 是 $f$ 的准逆，那么 $(g \circ -)$ 和 $(- \circ g)$ 分别是 $(f \circ -)$ 和 $(- \circ f)$ 的准逆。 □

Lemma 4.2.9. 如果 $f : A \to B$ 有一个准逆，那么类型 $\mathsf{rinv}(f)$ 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 $\Sigma$ 的纤维 (fiber) (§3.5)，因此根据 Lemma 4.2.8 and Theorems 4.2.3 and 4.2.6，它是可缩的。类似地，$\mathsf{rinv}(f)$ 等价于 $\Sigma$ 上的 $\mathsf{id}_B$ 的纤维 (fiber)，因此也是可缩的。 □

接下来，我们定义将另一个同伦与额外的一致性数据结合起来的类型。

Definition 4.2.10. 对于 $f : A \to B$，一个左逆 $(g,\eta) : \mathsf{linv}(f)$ 和一个右逆 $(g,\epsilon) : \mathsf{rinv}(f)$，我们表示

$$\mathsf{lcoh}_f(g,\eta) :\equiv \sum_{(\epsilon:f\circ g\sim \mathsf{id}_B)} \prod_{(y:B)} g(\epsilon y) = \eta(gy)$$
$$\mathsf{rcoh}_f(g,\epsilon) :\equiv \sum_{(\eta:g\circ f\sim \mathsf{id}_A)} \prod_{(x:A)} f(\eta x) = \epsilon(fx)$$

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 Lemma 4.2.5。 □

Lemma 4.2.12. 如果 $f$ 是一个半伴随等价，那么对于任何 $(g,\epsilon) : \mathsf{rinv}(f)$，类型 $\mathsf{rcoh}_f(g,\epsilon)$ 是可缩的。

证明. 根据 Lemma 4.2.11 和从属函数类型保持可缩空间的事实，足以展示对于每个 $x : A$，类型 $(gfx, \epsilon(fx)) =_{\mathsf{fib}_f(fx)} (x, \mathsf{refl}_{fx})$ 是可缩的。但根据 Theorem 4.2.6，$\mathsf{fib}_f(fx)$ 是可缩的，并且任何可缩空间的路径空间本身就是可缩的。 □

Theorem 4.2.13. 对于任何 $f : A \to B$，类型 $\mathsf{ishae}(f)$ 是一个纯命题。

证明. 根据 Exercise 3.5，假设 $f$ 是一个半伴随等价，并展示 $\mathsf{ishae}(f)$ 是可缩的就足够了。现在根据 $\Sigma$ 的结合律 (Exercise 2.10)，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 Lemmas 4.2.9 and 4.2.12 和 $\Sigma$ 保持可缩性的事实，后者类型也是可缩的。 □

因此，我们已经证明 $\mathsf{ishae}(f)$ 具有作为类型 $\mathsf{isequiv}(f)$ 的所有三种特性。在接下来的两节中，我们将讨论一些其他可能性。

## 4.3 双可逆映射 (Bi-invertible maps)

使用在 §4.2 引入的语言，我们可以重新表述在 §2.4 提出的定义，如下所示。

Definition 4.3.1. 我们说 $f : A \to B$ 是 **双可逆** (bi-invertible) 映射，如果它同时具有左逆元和右逆元：

$$\mathsf{biinv}(f) :\equiv \mathsf{linv}(f) \times \mathsf{rinv}(f).$$

在 §2.4 中，我们证明了 $\mathsf{qinv}(f) \to \mathsf{biinv}(f)$ 和 $\mathsf{biinv}(f) \to \mathsf{qinv}(f)$。剩下的是以下内容。

Theorem 4.3.2. 对于任意 $f : A \to B$，类型 $\mathsf{biinv}(f)$ 是一个单纯命题 (mere proposition)。

证明. 我们假设 $f$ 是双可逆的并证明 $\mathsf{biinv}(f)$ 是收缩的 (contractible)。但由于 $\mathsf{biinv}(f) \to \mathsf{qinv}(f)$，根据 Lemma 4.2.9，在这种情况下 $\mathsf{linv}(f)$ 和 $\mathsf{rinv}(f)$ 都是收缩的，而收缩类型的乘积也是收缩的。 □

注意，这也符合在 §4.2 开头提出的建议：我们将 $g$ 和 $\eta$ 组合成一个收缩类型，并添加一个额外的数据，它与 $\epsilon$ 组合成一个收缩类型。不同之处在于，我们没有添加一个 更高的数据（一个二维路径）来与 $\epsilon$ 组合，而是添加了一个 更低的数据（一个与左逆元分开的右逆元）。

Corollary 4.3.3. 对于任意 $f : A \to B$，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 $\mathsf{biinv}(f)$ 都是单纯命题，因此根据 Lemma 3.3.3 可以得出等价性。 □

## 4.4 **收缩纤维** (Contractible fibers)

注意，我们关于 $\mathsf{ishae}(f)$ 和 $\mathsf{biinv}(f)$ 的证明中，使用了等价的纤维是收缩的这一事实。实际上，这个性质本身就是等价的一个充分定义。

Definition 4.4.1 (收缩映射 (Contractible maps)). 映射 $f : A \to B$ 是 **收缩的** (contractible)，如果对所有 $y : B$，纤维 $\mathsf{fib}_f(y)$ 是收缩的。

因此，类型 $\mathsf{isContr}(f)$ 被定义为

$$\mathsf{isContr}(f) :\equiv \prod_{y:B} \mathsf{isContr}(\mathsf{fib}_f(y)) \tag{4.4.2}$$

注意，在 §3.11 中，我们定义了 类型收缩的含义。在这里，我们定义了 映射收缩的含义。我们的术语遵循一般同伦理论的实践，即如果一个映射的所有（同伦）纤维都具有某个性质，则该映射具有该性质。因此，当映射 $A \to \mathbf{1}$ 是收缩的时，类型 $A$ 是收缩的。从 Chapter 7 开始，我们也会称收缩映射和类型为 $(-2)$-截断的 ($(-2)$-truncated)。

我们已经在 Theorem 4.2.6 中展示了 $\mathsf{ishae}(f) \to \mathsf{isContr}(f)$。反之亦然：

Theorem 4.4.3. 对于任意 $f : A \to B$ 我们有 $\mathsf{isContr}(f) \to \mathsf{ishae}(f)$。

证明. 让 $P : \mathsf{isContr}(f)$。我们通过将每个 $y : B$ 映射到 $y$ 处纤维的收缩中心来定义一个逆映射 $g : B \to A$：

$$g(y) :\equiv \mathsf{pr}_1(\mathsf{pr}_1(Py))$$

因此，我们可以通过将 $y$ 映射到 $g(y)$ 确实属于 $y$ 处纤维的见证，来定义同伦 $\epsilon$：

$$\epsilon(y) :\equiv \mathsf{pr}_2(\mathsf{pr}_1(Py))$$

剩下的是定义 $\eta$ 和 $\tau$。这当然意味着给出一个 $\mathsf{rcoh}_f(g, \epsilon)$ 的元素。根据 Lemma 4.2.11，这相当于给出对于每个 $x : A$，从 $f$ 在 $fx$ 处的纤维中的 $(gfx, \epsilon(fx))$ 到 $(x, \mathsf{refl}_{fx})$ 的路径。但这很简单：对于任意 $x : A$，类型 $\mathsf{fib}_f(fx)$ 根据假设是收缩的，因此这样的路径必须存在。我们可以显式地构造它为

$$\big(\mathsf{pr}_2(P(fx))(gfx, \epsilon(fx))\big)^{-1} \cdot \big(\mathsf{pr}_2(P(fx))(x, \mathsf{refl}_{fx})\big).$$ □

同样容易看出：

Lemma 4.4.4. 对于任意 $f$，类型 $\mathsf{isContr}(f)$ 是一个单纯命题。

证明. 根据 Lemma 3.11.4，每个类型 $\mathsf{isContr}(\mathsf{fib}_f(y))$ 是一个单纯命题。因此，根据 Example 3.6.2，(4.4.2) 也是如此。 □

Theorem 4.4.5. 对于任意 $f : A \to B$ 我们有 $\mathsf{isContr}(f) \simeq \mathsf{ishae}(f)$。

证明. 我们已经建立了 $\mathsf{isContr}(f) \Leftrightarrow \mathsf{ishae}(f)$ 的逻辑等价关系，并且它们都是单纯命题 (Lemma 4.4.4 and Theorem 4.2.13)。因此，Lemma 3.3.3 适用。 □

通常，我们通过给出一个准逆元来证明一个函数是等价的，但有时这个定义更方便。例如，它暗示了当证明一个函数是等价的时，我们可以假设它的陪域是非空的。

Corollary 4.4.6. *如果 $f : A \to B$ 使得 $B \to \mathsf{isequiv}(f)$，那么 $f$ 是等价的。*

证明. 为了证明 $f$ 是等价的，足以证明对于任意 $y : B$，$\mathsf{fib}_f(y)$ 是收缩的。但是如果 $e : B \to \mathsf{isequiv}(f)$，那么对于任意 $y$，我们有 $e(y) : \mathsf{isequiv}(f)$，因此 $f$ 是等价的，并且因此 $\mathsf{fib}_f(y)$ 是收缩的，如所需。 □

## 4.5 关于等价定义的思考 (On the definition of equivalences)

我们已经证明所有三个等价的定义满足三个理想的性质，并且它们是成对等价的：

$$\mathsf{isContr}(f) \simeq \mathsf{ishae}(f) \simeq \mathsf{biinv}(f).$$

（还有更多可能的等价定义，但我们将在此处讨论这三个。参见 Exercise 3.8 和本章中的习题，以了解更多。）因此，我们可以选择其中任何一个作为 $\mathsf{isequiv}(f)$ 的“定义”。为了明确起见，我们选择定义

$$\mathsf{isequiv}(f) :\equiv \mathsf{ishae}(f)$$

这个选择对形式化很有利，因为 $\mathsf{ishae}(f)$ 包含了最直接有用的数据。另一方面，对于其他目的，$\mathsf{biinv}(f)$ 通常更容易处理，因为它不包含二维路径，并且它的两个对称部分可以独立处理。然而，对于本书的目的，具体选择几乎没有区别。

在本章的其余部分，我们研究等价的一些其他性质和特征。

## 4.6 满射与嵌入 (Surjections and embeddings)

当 $A$ 和 $B$ 是集合 (sets) 并且 $f : A \to B$ 是等价的时，我们也称它为 **同构** (isomorphism) 或 **双射** (bijection)。（对于不是集合的类型，我们避免使用这些词，因为在同伦理论和高阶范畴论中，它们通常表示比同伦等价更严格的“相同”概念。）在集合论中，当且仅当一个函数既是单射 (injective) 又是满射 (surjective) 时，它是一个双射。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，在处理非集合的类型时，我们将使用 *嵌入* (embeddings) 一词来代替单射。

Definition 4.6.1. 设 $f : A \to B$。

(i) 我们说 $f$ 是 **满射** (surjective)（或称为 **满射** (surjection)）如果对于每个 $b : B$，我们有 $\left\|\mathsf{fib}_f(b)\right\|$。

(ii) 我们说 $f$ 是一个 **嵌入** (embedding) 如果对于每个 $x, y : A$，函数 $\mathsf{ap}_f : (x =_A y) \to (f(x) =_B f(y))$ 是一个等价。

换句话说，如果 $f$ 的每个纤维只是居住的，或者等价地，如果对于所有 $b : B$，仅存在一个 $a : A$ 使得 $f(a) = b$，则 $f$ 是满射。在传统逻辑符号中，如果 $\forall(b : B).\, \exists(a : A).\, (f(a) = b)$，则 $f$ 是满射。这必须与更强的断言 $\prod_{(b:B)} \sum_{(a:A)} (f(a) = b)$ 区分开来；如果这成立，我们说 $f$ 是一个 **分裂满射** (split surjection)。（因为这个后者的类型等价于 $\sum_{(g:B\to A)} \prod_{(b:B)} (f(g(b)) = b)$，所以分裂满射与 §3.11 中定义的 *再牵引* (retraction) 是相同的。）

§3.8 中的选择公理 (axiom of choice) 恰好说明了每个 *集合之间的*满射都是分裂的。然而，在单一性公理 (univalence axiom) 的存在下，简单地说，*所有*满射都不是分裂的。在 Lemma 3.8.5 中，我们构建了一个类型族 $Y : X \to \mathcal{U}$，使得 $\prod_{(x:X)} \|Y(x)\|$ 但 $\neg \prod_{(x:X)} Y(x)$；对于任何此类族，$Y$ 的第一次投影 $(\sum_{(x:X)} Y(x)) \to X$ 是一个未分裂的满射。

如果 $A$ 和 $B$ 是集合，那么根据 Lemma 3.3.3，当且仅当

$$\prod_{x,y:A} (f(x) =_B f(y)) \to (x =_A y). \tag{4.6.2}$$

时，$f$ 是嵌入的。在这种情况下，我们说 $f$ 是 **单射** (injective)，或 **单射** (injection)。对于非集合的类型，我们避免使用这些词，因为它们可能会被解释为 (4.6.2)，这是对非集合类型不合适的概念。同样地，任何集合之间的函数如果且仅如果它是一个适当意义上的 上同态 (epimorphism)，则它是满射，但这对更一般的类型来说也是不成立的，并且满射通常是更重要的概念。

Theorem 4.6.3. *一个函数 $f : A \to B$ 当且仅当它既是满射又是嵌入时是等价的。*

*证明.* 如果 $f$ 是等价的，那么每个 $\mathsf{fib}_f(b)$ 都是收缩的，因此 $\left\|\mathsf{fib}_f(b)\right\|$ 也是收缩的，因此 $f$ 是满射的。我们在 ?? 中证明了任何等价都是嵌入的。
反之，假设 $f$ 是一个满射嵌入。设 $b : B$；我们证明 $\sum_{(x:A)}(f(x) = b)$ 是收缩的。因为 $f$ 是满射的，所以仅存在一个 $a : A$ 使得 $f(a) = b$。因此，$f$ 在 $b$ 上的纤维是非空的；剩下的是证明它是一个单纯命题。为此，假设给定 $x, y : A$ 且 $p : f(x) = b$ 和 $q : f(y) = b$。然后由于 $\mathsf{ap}_f$ 是一个等价的，存在 $r : x = y$ 使得 $\mathsf{ap}_f(r) = p \cdot q^{-1}$。然而，使用在 $\Sigma$-类型中的路径表征，后者等式重新排列为 $r_*(p) = q$。因此，连同 $r$ 一起，它展示了在 $f$ 在 $b$ 上的纤维中的 $(x, p) = (y, q)$。 □

Corollary 4.6.4. *对于任意 $f : A \to B$ 我们有*

$$\mathsf{isequiv}(f) \simeq (\mathsf{isEmbedding}(f) \times \mathsf{isSurjective}(f))$$

*证明.* 作为一个满射和嵌入都是单纯命题；现在应用 Lemma 3.3.3。 □

当然，这不能用作“等价”的定义，因为嵌入的定义涉及到等价的概念。然而，这个表征仍然是有用的；参见 ??。我们将在 Chapter 7 中对其进行推广。

## 4.7 **等价的闭包性质** (Closure properties of equivalences)

我们已经在 Lemma 2.4.12 中看到，等价关系在复合下是封闭的。此外，我们还有：

Theorem 4.7.1 (三选二性质 (The 2-out-of-3 property)). *假设 $f : A \to B$ 和 $g : B \to C$。如果 $f$、$g$ 和 $g \circ f$ 中的任意两个是等价的，那么第三个也是等价的。*

*证明.* 如果 $g \circ f$ 和 $g$ 是等价的，那么 $(g \circ f)^{-1} \circ g$ 是 $f$ 的准逆 (quasi-inverse)。一方面，我们有 $(g \circ f)^{-1} \circ g \circ f \sim \mathsf{id}_A$，而另一方面我们有

$$\begin{aligned} f \circ (g \circ f)^{-1} \circ g &\sim g^{-1} \circ g \circ f \circ (g \circ f)^{-1} \circ g \\ &\sim g^{-1} \circ g \\ &\sim \mathsf{id}_B. \end{aligned}$$

同样地，如果 $g \circ f$ 和 $f$ 是等价的，那么 $f \circ (g \circ f)^{-1}$ 是 $g$ 的准逆。 □

这是同伦理论中关于等价的一个标准闭包条件。同样众所周知的是，它们在以下意义下对于再牵引 (retracts) 也是封闭的。

Definition 4.7.2. 若有一个图

$$\begin{array}{ccccc} A & \xrightarrow{s} & X & \xrightarrow{r} & A \\ {\scriptstyle g}\downarrow & & {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[s']{} & Y & \xrightarrow[r']{} & B \end{array}$$

则称函数 $g : A \to B$ 是函数 $f : X \to Y$ 的 **再牵引** (retract)，其中存在

(i) 一个同伦 $R : r \circ s \sim \mathsf{id}_A$。
(ii) 一个同伦 $R' : r' \circ s' \sim \mathsf{id}_B$。
(iii) 一个同伦 $L : f \circ s \sim s' \circ g$。
(iv) 一个同伦 $K : g \circ r \sim r' \circ f$。
(v) 对于每个 $a : A$，存在一个路径 $H(a)$ 见证下图的交换性

$$\begin{array}{ccc} g(r(s(a))) & \xrightarrow{K(s(a))} & r'(f(s(a))) \\ {\scriptstyle g(R(a))}\Big\| & & \Big\|{\scriptstyle r'(L(a))} \\ g(a) & \xrightarrow[R'(g(a))^{-1}]{} & r'(s'(g(a))) \end{array}$$

回顾一下，在 §3.11 中我们定义了类型如何成为另一类型的再牵引。这是上述定义的一个特例，其中 $B$ 和 $Y$ 是 $\mathbf{1}$。相反，正如与可收缩性一样，映射的再牵引会诱导其纤维的再牵引。

Lemma 4.7.3. *如果函数 $g : A \to B$ 是函数 $f : X \to Y$ 的再牵引，那么对于每个 $b : B$，$\mathsf{fib}_g(b)$ 是 $\mathsf{fib}_f(s'(b))$ 的再牵引，其中 $s' : B \to Y$ 如 Definition 4.7.2 所述。*

*证明.* 假设 $g : A \to B$ 是 $f : X \to Y$ 的再牵引。那么对于任何 $b : B$，我们有函数

$$\begin{aligned} &\varphi_b : \mathsf{fib}_g(b) \to \mathsf{fib}_f(s'(b)), && \varphi_b(a,p) :\equiv (s(a), L(a) \cdot s'(p)), \\ &\psi_b : \mathsf{fib}_f(s'(b)) \to \mathsf{fib}_g(b), && \psi_b(x,q) :\equiv (r(x), K(x) \cdot r'(q) \cdot R'(b)). \end{aligned}$$

那么我们有 $\psi_b(\varphi_b(a,p)) \equiv (r(s(a)), K(s(a)) \cdot r'(L(a) \cdot s'(p)) \cdot R'(b))$。我们声称 $\psi_b$ 是一个具有截面的再牵引 $\varphi_b$ 对于所有 $b : B$，也就是说，对于所有 $(a,p) : \mathsf{fib}_g(b)$，我们有 $\psi_b(\varphi_b(a,p)) = (a,p)$。换句话说，我们要证明

$$\prod_{(b:B)} \prod_{(a:A)} \prod_{(p:g(a)=b)} \psi_b(\varphi_b(a,p)) = (a,p)$$

通过重新排列前两个 $\Pi$ 并应用 Lemma 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 ??，这个对偶对等价于一对等式。由于 $R(a) : r(s(a)) = a$，所以我们只需要证明

$$R(a)_*(K(s(a)) \cdot r'(L(a)) \cdot R'(g(a))) = \mathsf{refl}_{g(a)}$$

但这种变换计算为 $g(R(a))^{-1} \cdot K(s(a)) \cdot r'(L(a)) \cdot R'(g(a))$，所以所需的路径由 $H(a)$ 给出。 □

Theorem 4.7.4. *如果 $g$ 是等价映射 $f$ 的再牵引，那么 $g$ 也是等价映射。*

*证明.* 根据 Lemma 4.7.3，$g$ 的每个纤维都是 $f$ 的纤维的再牵引。因此，根据 Lemma 3.11.7，如果后者都是可收缩的，那么前者也是。 □

最后，我们展示了纤维等价性 (fiberwise equivalences) 可以通过全空间等价 (equivalences of total spaces) 来表征。为了解释这个术语，回顾 §2.3，其中类型族 $P : A \to \mathcal{U}$ 可以看作是 $A$ 上的一个纤维丛，其全空间为 $\sum_{(x:A)} P(x)$，纤维丛的投影为 $\mathsf{pr}_1 : \sum_{(x:A)} P(x) \to A$。从这个角度来看，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 **纤维映射** (fiberwise map) 或 **纤维变换** (fiberwise transformation)。这样的映射在全空间上诱导一个函数：

Definition 4.7.5. 给定类型族 $P, Q : A \to \mathcal{U}$ 和一个映射 $f : \prod_{(x:A)} P(x) \to Q(x)$，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 $Q$ 之间的一个纤维变换，并设 $x : A$ 和 $v : Q(x)$。那么我们有一个等价关系

$$\mathsf{fib}_{\mathsf{total}(f)}((x, v)) \simeq \mathsf{fib}_{f(x)}(v)$$

证明. 我们计算如下：

$$\begin{aligned}
\mathsf{fib}_{\mathsf{total}(f)}((x,v)) &\equiv \sum_{w:\sum_{(x:A)} P(x)} (\mathsf{pr}_1 w, f(\mathsf{pr}_1 w, \mathsf{pr}_2 w)) = (x, v) \\
&\simeq \sum_{(a:A)} \sum_{(u:P(a))} (a, f(a, u)) = (x, v) && \text{(by Exercise 2.10)} \\
&\simeq \sum_{(a:A)} \sum_{(u:P(a))} \sum_{(p:a=x)} p_*(f(a, u)) = v && \text{(by ??)} \\
&\simeq \sum_{(a:A)} \sum_{(p:a=x)} \sum_{(u:P(a))} p_*(f(a, u)) = v \\
&\simeq \sum_{u:P(x)} f(x, u) = v && (*) \\
&\equiv \mathsf{fib}_{f(x)}(v)
\end{aligned}$$

等式 $(*)$ 由 Lemmas 3.11.8 and 3.11.9 and Exercise 2.10 得出。 □

我们称纤维变换 $f : \prod_{(x:A)} P(x) \to Q(x)$ 为 **纤维等价** (fiberwise equivalence)如果每个 $f(x) : P(x) \to Q(x)$ 是一个等价。

Theorem 4.7.7. 假设 $f$ 是类型 $A$ 上族 $P$ 和 $Q$ 之间的一个纤维变换。则 $f$ 是纤维等价当且仅当 $\mathsf{total}(f)$ 是等价映射。

证明. 设 $f$、$P$、$Q$ 和 $A$ 如定理所述。根据 Theorem 4.7.6, 对所有 $x : A$ 和 $v : Q(x)$, $\mathsf{fib}_{\mathsf{total}(f)}((x, v))$ 可收缩当且仅当 $\mathsf{fib}_{f(x)}(v)$ 可收缩。因此，$\mathsf{fib}_{\mathsf{total}(f)}(w)$ 对所有 $w : \sum_{(x:A)} Q(x)$ 可收缩当且仅当 $\mathsf{fib}_{f(x)}(v)$ 对所有 $x : A$ 和 $v : Q(x)$ 可收缩。 □

## 4.8 对象分类器 (The object classifier)

在类型论中，我们有一个基本的“类型族”的概念，即函数 $B : A \to \mathcal{U}$。我们已经看到，这样的族在某种程度上类似于同伦理论中的纤维丛 (fibration)，纤维丛为投影 $\mathsf{pr}_1 : \sum_{(a:A)} B(a) \to A$。同伦理论中的一个基本事实是每个映射都等价于一个纤维丛。借助单值化 (univalence)，我们可以在类型论中证明同样的事情。

Lemma 4.8.1. 对于任意类型族 $B : A \to \mathcal{U}$，$\mathsf{pr}_1 : \sum_{(x:A)} B(x) \to A$ 在 $a : A$ 处的纤维等价于 $B(a)$：

$$\mathsf{fib}_{\mathsf{pr}_1}(a) \simeq B(a)$$

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 (left universal property)。 □

Lemma 4.8.2. 对于任意函数 $f : A \to B$，我们有 $A \simeq \sum_{(b:B)} \mathsf{fib}_f(b)$。

证明. 我们有

$$\begin{aligned}\sum_{b:B} \mathsf{fib}_f(b) &:\equiv \sum_{(b:B)} \sum_{(a:A)} (f(a) = b) \\ &\simeq \sum_{(a:A)} \sum_{(b:B)} (f(a) = b) \\ &\simeq A\end{aligned}$$

利用 $\sum_{(b:B)}(f(a) = b)$ 是可收缩的事实。 □

Theorem 4.8.3. 对于任意类型 $B$，存在一个等价

$$\chi : \Big(\sum_{A:\mathcal{U}} (A \to B)\Big) \simeq (B \to \mathcal{U})$$

证明. 我们需要构造准逆

$$\begin{aligned}\chi &: \Big(\sum_{A:\mathcal{U}} (A \to B)\Big) \to B \to \mathcal{U} \\ \psi &: (B \to \mathcal{U}) \to \Big(\sum_{A:\mathcal{U}} (A \to B)\Big)\end{aligned}$$

我们定义 $\chi$ 为 $\chi((A, f), b) :\equiv \mathsf{fib}_f(b)$，定义 $\psi$ 为 $\psi(P) :\equiv \big((\sum_{(b:B)} P(b)), \mathsf{pr}_1\big)$。现在我们需要验证 $\chi \circ \psi \sim \mathsf{id}$ 和 $\psi \circ \chi \sim \mathsf{id}$。

(i) 设 $P : B \to \mathcal{U}$。通过 Lemma 4.8.1，$\mathsf{fib}_{\mathsf{pr}_1}(b) \simeq P(b)$ 对于任意 $b : B$，所以这立即得出 $P \sim \chi(\psi(P))$。

(ii) 设 $f : A \to B$ 是一个函数。我们需要找到一个路径

$$\big(\textstyle\sum_{(b:B)} \mathsf{fib}_f(b),\ \mathsf{pr}_1\big) = (A, f)$$

首先注意到，通过 Lemma 4.8.2，我们有 $e : \sum_{(b:B)} \mathsf{fib}_f(b) \simeq A$，定义为 $e(b, a, p) :\equiv a$ 和 $e^{-1}(a) :\equiv (f(a), a, \mathsf{refl}_{f(a)})$。根据 ??，剩下的部分是要证明 $(\mathsf{ua}(e))_*(\mathsf{pr}_1) = f$。但通过单值化的计算规则和 (2.7.4)，我们有 $(\mathsf{ua}(e))_*(\mathsf{pr}_1) = \mathsf{pr}_1 \circ e^{-1}$，并且 $e^{-1}$ 的定义立即得出 $\mathsf{pr}_1 \circ e^{-1} \equiv f$。 □

特别地，这意味着我们在更高拓扑论 (higher topos theory) 的意义上有一个对象分类器 (object classifier)。回顾 Definition 2.1.7 中 $\mathcal{U}_\bullet$ 表示带指点类型的类型 $\sum_{(A:\mathcal{U})} A$。

Theorem 4.8.4. 设 $f : A \to B$ 是一个函数。则图

$$\begin{array}{ccc} A & \xrightarrow{\vartheta_f} & \mathcal{U}_\bullet \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \mathsf{pr}_1} \\ B & \xrightarrow[\chi_f]{} & \mathcal{U} \end{array}$$

是一个拉回方块 (pullback square)（见 Exercise 2.11）。其中函数 $\vartheta_f$ 定义为

$$\lambda a.\, (\mathsf{fib}_f(f(a)), (a, \mathsf{refl}_{f(a)}))$$

证明. 注意我们有等价关系

$$\begin{aligned} A &\simeq \sum_{b:B} \mathsf{fib}_f(b) \\ &\simeq \sum_{(b:B)} \sum_{(X:\mathcal{U})} \sum_{(p:\mathsf{fib}_f(b)=X)} X \\ &\simeq \sum_{(b:B)} \sum_{(X:\mathcal{U})} \sum_{(x:X)} \mathsf{fib}_f(b) = X \\ &\simeq \sum_{(b:B)} \sum_{(Y:\mathcal{U}_\bullet)} \mathsf{fib}_f(b) = \mathsf{pr}_1 Y \\ &\equiv B \times_{\mathcal{U}} \mathcal{U}_\bullet \end{aligned}$$

这给我们一个复合等价 $e : A \simeq B \times_{\mathcal{U}} \mathcal{U}_\bullet$。我们可以通过以下步骤逐步显示这个复合等价的操作：

$$\begin{aligned} a &\mapsto (f(a),\ (a, \mathsf{refl}_{f(a)})) \\ &\mapsto (f(a),\ \mathsf{fib}_f(f(a)),\ \mathsf{refl}_{\mathsf{fib}_f(f(a))},\ (a, \mathsf{refl}_{f(a)})) \\ &\mapsto (f(a),\ \mathsf{fib}_f(f(a)),\ (a, \mathsf{refl}_{f(a)}),\ \mathsf{refl}_{\mathsf{fib}_f(f(a))}) \end{aligned}$$

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 一致性 (Univalence) 蕴含函数外延性 (Function Extensionality)

在本章的最后一节中，我们将证明一致性公理 (univalence axiom) 蕴含了函数外延性 (function extensionality)。因此，在本节中我们不使用函数外延性公理。该证明分为两步。首先，我们在 Theorem 4.9.4 中展示了一致性公理 (univalence) 蕴含了函数外延性的弱形式，定义见下文 Definition 4.9.1。然后，弱函数外延性原则 (weak function extensionality) 又蕴含了通常的函数外延性，而这一过程不依赖于一致性公理 (Theorem 4.9.5)。

令 $\mathcal{U}$ 为一个宇宙；我们将在适当的位置明确指出何时假设其为一致的。

Definition 4.9.1. **弱函数外延性原则** (weak function extensionality principle) 断言存在一个函数

$$\Big(\prod_{x:A} \mathsf{isContr}(P(x))\Big) \to \mathsf{isContr}\Big(\prod_{x:A} P(x)\Big)$$

对于任意类型 $A$ 上的类型族 $P : A \to \mathcal{U}$。

使用函数外延性可以轻松证明以下引理；关键在于它也可以在不假设函数外延性的情况下从一致性得到。

Lemma 4.9.2. 假设 $\mathcal{U}$ 是一致的，对于任意的 $A, B, X : \mathcal{U}$ 和任意的 $e : A \simeq B$，存在一个等价性

$$(X \to A) \simeq (X \to B)$$

其底层映射由 $e$ 的底层函数通过后合成给出。

证明. 如 Lemma 4.1.1 的证明中，我们可以假设 $e = \mathsf{idtoeqv}(p)$，其中 $p : A = B$。然后通过路径归纳法 (path induction)，我们可以假设 $p$ 为 $\mathsf{refl}_A$，因此 $e = \mathsf{id}_A$。但在这种情况下，后合成 $e$ 是恒等映射，因此是等价的。 □

Corollary 4.9.3. 设 $P : A \to \mathcal{U}$ 是一族可收缩类型 (contractible types)，即 $\prod_{(x:A)} \mathsf{isContr}(P(x))\mathsf{pr}_1 : (\sum_{(x:A)} P(x)) \to A$ 是一个等价性。如果假设 $\mathcal{U}$ 是一致的，那么通过后合成 $\mathsf{pr}_1$ 立即得到一个等价性

$$\alpha : \Big(A \to \sum_{x:A} P(x)\Big) \simeq (A \to A)$$

证明. 根据 Lemma 4.8.1，对于 $\mathsf{pr}_1 : \sum_{(x:A)} P(X) \to A$ 和 $x : A$，我们有一个等价性

$$\mathsf{fib}_{\mathsf{pr}_1}(x) \simeq P(x)$$

因此，当每个 $P(x)$ 是可收缩时，$\mathsf{pr}_1$ 是一个等价性。这个断言现在是 Lemma 4.9.2 的一个推论。 □

特别地，上述等价性在 $\mathsf{id}_A$ 处的同伦纤维是可收缩的。因此，我们可以通过展示依赖函数类型 $\prod_{(x:A)} P(x)$ 是 $\mathsf{fib}_\alpha(\mathsf{id}_A)$ 的一个收缩来证明一致性蕴含了弱函数外延性。

Theorem 4.9.4. *在一个一致的宇宙 $\mathcal{U}$ 中，假设 $P : A \to \mathcal{U}$ 是一族可收缩类型，并且令 $\alpha$ 为 Corollary 4.9.3 中的函数。那么 $\prod_{(x:A)} P(x)$ 是 $\mathsf{fib}_\alpha(\mathsf{id}_A)$ 的一个收缩。由此可见，$\prod_{(x:A)} P(x)$ 是可收缩的。换句话说，一致性公理蕴含了弱函数外延性原则。*

证明. 定义函数

$$\varphi : (\textstyle\prod_{(x:A)} P(x)) \to \mathsf{fib}_\alpha(\mathsf{id}_A)$$
$$\varphi(f) :\equiv (\lambda x.\, (x, f(x)), \mathsf{refl}_{\mathsf{id}_A})$$

以及

$$\psi : \mathsf{fib}_\alpha(\mathsf{id}_A) \to \textstyle\prod_{(x:A)} P(x)$$
$$\psi(g, p) :\equiv \lambda x.\, \mathsf{happly}(p, x)_*(\mathsf{pr}_2(g(x)))$$

那么 $\psi(\varphi(f)) = \lambda x.\, f(x)$，即 $f$，根据依赖函数类型的唯一性原理 (uniqueness principle)。 □

现在我们展示弱函数外延性蕴含了通常的函数外延性。回忆自 (2.7.2) 的函数 $\mathsf{happly}(f, g) : (f = g) \to (f \sim g)$，其将函数的相等性转化为同伦。在接下来的证明中，未使用一致性公理。

Theorem 4.9.5. *弱函数外延性蕴含函数外延性* Axiom 2.7.3。

证明. 我们想要证明

$$\prod_{(A:\mathcal{U})} \prod_{(P:A\to\mathcal{U})} \prod_{(f,g:\prod_{(x:A)} P(x))} \mathsf{isequiv}(\mathsf{happly}(f, g))$$

由于纤维映射 (fiberwise map) 当且仅当它是纤维的等价性时会在整个空间上诱导出等价性 (Theorem 4.7.7)，我们只需展示以下类型的函数

$$\Big(\sum_{g:\prod_{(x:A)} P(x)} (f = g)\Big) \to \sum_{g:\prod_{(x:A)} P(x)} (f \sim g)$$

由 $\lambda(g : \prod_{(x:A)} P(x)).\, \mathsf{happly}(f, g)$ 诱导出的函数是一个等价性。由于左侧的类型是可收缩的 (Lemma 3.11.8)，我们只需证明右侧的类型：

$$\sum_{(g:\prod_{(x:A)} P(x))} \prod_{(x:A)} f(x) = g(x) \tag{4.9.6}$$

是可收缩的。现在 Theorem 2.9.7 表明这与以下类型等价：

$$\prod_{(x:A)} \sum_{(u:P(x))} f(x) = u \tag{4.9.7}$$

Theorem 2.9.7 的证明使用了函数外延性，但仅对其中一个复合函数使用。因此，在不假设函数外延性的情况下，我们可以得出 (4.9.6) 是 (4.9.7) 的一个收缩。并且 (4.9.7) 是一个可收缩类型的乘积，由弱函数外延性原则可知，它是可收缩的；因此 (4.9.6) 也是可收缩的。 □

## Notes

带有拟逆的连续映射空间具有错误的同伦类型，而不是"同伦等价空间"的事实在代数拓扑学中是众所周知的。在这种情况下，"同伦等价空间"$(A \simeq B)$ 通常仅定义为由同伦等价构成的函数空间 $(A \to B)$ 的子空间。在类型论中，这将最接近于 $\sum_{(f:A\to B)} \|\mathsf{qinv}(f)\|$；见 Exercise 3.8。

在同伦类型论 (Homotopy Type Theory) 中，第一个等价性定义是由 Voevodsky 提出的，即我们称为 $\mathsf{isContr}(f)$ 的定义。其他定义的可能性后来被不同的人注意到。关于伴随等价性 (adjoint equivalences) 的基本定理，例如 Lemma 4.2.2 and Theorem 4.2.3 是从高等范畴论 (higher category theory) 和同伦论 (homotopy theory) 中的标准事实改编而来的。使用双可逆性 (bi-invertibility) 作为等价性定义是由 André Joyal 提出的。

在 §§4.6 and 4.7 中讨论的等价性性质在同伦论中是众所周知的。它们中的大多数首次在类型论中被 Voevodsky 证明。

每个函数等价于纤维化 (fibration) 是同伦论中的标准事实。对象分类器 (object classifier) 的概念 在 $(\infty,1)$-范畴 (category) 理论中（Theorem 4.8.3 的范畴对应物）是 Rezk 提出的 (参见 [?, ?])。

最后，一致性蕴含函数外延性 (§4.9) 的事实归功于 Voevodsky。我们的证明是他证明的简化版。Exercise 4.9 也归功于 Voevodsky。

## Exercises

Exercise 4.1. 考虑 $f : A \to B$ 的"两侧伴随等价性 (two-sided adjoint equivalence) 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 Lemma 4.2.2，我们知道如果 $f$ 是等价性，那么该类型是可居住的。给出一个与 Lemma 4.1.1 类似的该类型的表征。

你能给出一个例子来展示这个类型通常不是一个单纯命题 (mere proposition) 吗?（在学习 Chapter 6 后，这将变得更容易。）

Exercise 4.2. 证明对于任意 $A, B : \mathcal{U}$，以下类型等价于 $A \simeq B$。

$$\sum_{R:A\to B\to\mathcal{U}} \Big(\prod_{a:A} \mathsf{isContr}\Big(\sum_{b:B} R(a,b)\Big)\Big) \times \Big(\prod_{b:B} \mathsf{isContr}\Big(\sum_{a:A} R(a,b)\Big)\Big)$$

你能从中提取出一个满足 $\mathsf{isequiv}(f)$ 的三个要求的类型定义吗?

Exercise 4.3. 在不使用一致性的情况下重新表述 Lemma 4.1.1 的证明。

Exercise 4.4 (不稳定八面体公理 (The unstable octahedral axiom)). 假设 $f : A \to B$ 和 $g : B \to C$ 以及 $b : B$。

(i) 证明存在一个自然映射 $\mathsf{fib}_{g\circ f}(g(b)) \to \mathsf{fib}_g(g(b))$，其纤维在 $(b, \mathsf{refl}_{g(b)})$ 处等价于 $\mathsf{fib}_f(b)$。

(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. 证明等价性满足 2-out-of-6 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 $h \circ g$ 是等价的，那么 $f, g, h$ 和 $h \circ g \circ f$ 也是等价的。使用此结果来给出 ?? 的更高级别的证明。

Exercise 4.6. 对于 $A, B : \mathcal{U}$，定义

$$\mathsf{idtoqinv}_{A,B} : (A = B) \to \sum_{f:A\to B} \mathsf{qinv}(f)$$

通过路径归纳法 (path induction) 以显而易见的方式定义。令 qinv-**一致性** (univalence) 表示一致性公理的修改形式，该公理断言对于所有 $A, B : \mathcal{U}$，函数 $\mathsf{idtoqinv}_{A,B}$ 具有一个拟逆。

(i) 证明在 §4.9 中，可以用 qinv-一致性代替一致性来证明函数外延性。
(ii) 证明在 Theorem 4.1.3 中，可以用 qinv-一致性代替一致性。
(iii) 证明 qinv-一致性是不一致的 (inconsistent)（即允许构造 **0** 的一个居住者）。因此，在一致性声明中使用“良好”版本的 isequiv 是至关重要的。

Exercise 4.7. 证明当且仅当以下两个条件成立时，函数 $f : A \to B$ 是一个嵌入 (embedding)：

(i) $f$ 是 *左可消的* (left cancellable)，即对于任意 $x, y : A$，如果 $f(x) = f(y)$，那么 $x = y$。
(ii) 对于任意 $x : A$，映射 $\mathsf{ap}_f : \Omega(A, x) \to \Omega(B, f(x))$ 是一个等价性。

（特别地，如果 $A$ 是一个集合，那么 $f$ 是一个嵌入，如果且仅当它是左可消的，并且对于所有 $x : A$，$\Omega(B, f(x))$ 是可收缩的。）给出例子证明 (i) 和 (ii) 中的任一项都不能推出另一项。

Exercise 4.8. 证明左可消函数 $\mathbf{2} \to B$ 的类型（参见 Exercise 4.7）等价于 $\sum_{(x,y:B)}(x \neq y)$。给出一个类似的显式表征 $\mathbf{2} \to B$ 的嵌入类型。

Exercise 4.9. **天真的非依赖函数外延性公理** (naïve non-dependent function extensionality axiom) 断言，对于 $A, B : \mathcal{U}$ 和 $f, g : A \to B$，存在一个函数 $(\prod_{(x:A)} f(x) = g(x)) \to (f = g)$。修改 §4.9 中的论证，证明该公理蕴含了完整的函数外延性公理 (Axiom 2.7.3)。

# Chapter 5

# 归　　纳　　(Induction)

在 Chapter 1 中，我们介绍了许多从旧类型生成新类型的方法。除了（依赖）函数类型和宇宙 (universe) 类型外，所有这些规则都是一般性概念“归纳定义 (inductive definition)”的特例。 在本章中，我们将更广泛地研究归纳定义。

## 5.1 **归纳类型简介** (Introduction to inductive types)

一种 *归纳类型* (inductive type) $X$ 可以直观地理解为由某个有限集合的 *构造器* (constructors) 自由生成的类型，这些构造器中的每一个都是一个以 $X$ 为目标域的函数（具有一定数量的参数）。这包括零参数的函数，它们只是 $X$ 的元素。
在描述特定的归纳类型时，我们用项目符号列出构造器。例如，来自 §1.8 的类型 **2** 是由以下构造器归纳生成的：

- $0_2 : \mathbf{2}$
- $1_2 : \mathbf{2}$

类似地，**1** 是由以下构造器归纳生成的：

- $\star : \mathbf{1}$

而 **0** 则没有任何构造器。一个构造器函数接受参数的示例是余积 (coproduct) $A + B$，它由以下两个构造器生成：

- $\mathsf{inl} : A \to A + B$
- $\mathsf{inr} : B \to A + B$。

另一个接受多个参数的示例是笛卡尔积 (cartesian product) $A \times B$，它由以下构造器生成：

- $(-, -) : A \to B \to A \times B$。

重要的是，我们还允许定义的归纳类型的构造器接受来自被定义的归纳类型的参数。例如，自然数类型 (natural numbers) $\mathbb{N}$ 的构造器为：

- $0 : \mathbb{N}$
- $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$。

另一个有用的例子是类型 $\mathsf{List}(A)$，它表示某个类型 $A$ 的有限列表，其构造器为：

- $\mathsf{nil} : \mathsf{List}(A)$
- $\mathsf{cons} : A \to \mathsf{List}(A) \to \mathsf{List}(A)$。

直观上，我们应该将归纳类型理解为由其构造器 *自由生成的*类型。即，归纳类型的元素正是通过从无到有并反复应用构造器获得的。（我们将在 §5.8 and Chapter 6 中看到，这种概念在处理更一般的归纳定义时需要略微修改，但现在我们这样理解是足够的。）例如，对于 $\mathbf{2}$ ，我们应该期望唯一的元素是 $0_\mathbf{2}$ 和 $1_\mathbf{2}$。类似地，对于 $\mathbb{N}$，我们应该期望每个元素要么是 0，要么是通过对某个"先前构造的"自然数应用 $\mathsf{succ}$ 获得的。

然而，我们不是直接断言这样的性质，而是通过 *归纳原理* (induction principle)，也称为 *（依赖）消去规则* (dependent elimination rule) 来表达这些性质。我们已经在 Chapter 1 中见过这些原理。例如，$\mathbf{2}$ 的归纳原理是：

- 在证明关于 $\mathbf{2}$ 的 *所有*居民的陈述 $E : \mathbf{2} \to \mathcal{U}$ 时，只需要证明其对于 $0_\mathbf{2}$ 和 $1_\mathbf{2}$ 成立，即给出证明 $e_0 : E(0_\mathbf{2})$ 和 $e_1 : E(1_\mathbf{2})$。

此外，当应用于构造器 $0_\mathbf{2}$ 和 $1_\mathbf{2}$ 时，所得证明 $\mathsf{ind}_\mathbf{2}(E, e_0, e_1) : \prod_{(b:\mathbf{2})} E(b)$ 的行为符合预期；这一原理通过 *计算规则* (computation rules) 表达：：

- 我们有 $\mathsf{ind}_\mathbf{2}(E, e_0, e_1, 0_\mathbf{2}) \equiv e_0$。
- 我们有 $\mathsf{ind}_\mathbf{2}(E, e_0, e_1, 1_\mathbf{2}) \equiv e_1$。

因此，布尔类型 (type $\mathbf{2}$) 的归纳原理允许我们通过 *情况分析* (case analysis) 进行推理。由于这两个构造器都不接受任何参数，因此这是布尔类型所需的全部内容。

然而，对于自然数，情况分析通常是不够的：在对应于归纳步骤 $\mathsf{succ}(n)$ 的情况下，我们还希望假定要证明的命题已经对 $n$ 成立。这给出了以下归纳原理：

- 在证明关于 *所有*自然数的命题 $E : \mathbb{N} \to \mathcal{U}$ 时，只需证明其对 0 和 $\mathsf{succ}(n)$ 成立，并假设其对 $n$ 成立，即构造 $e_z : E(0)$ 和 $e_s : \prod_{(n:\mathbb{N})} E(n) \to E(\mathsf{succ}(n))$。

与布尔类型的情况类似，我们也有与函数 $\mathsf{ind}_\mathbb{N}(E, e_z, e_s) : \prod_{(x:\mathbb{N})} E(x)$ 相关的计算规则：

- $\mathsf{ind}_\mathbb{N}(E, e_z, e_s, 0) \equiv e_z$。
- $\mathsf{ind}_\mathbb{N}(E, e_z, e_s, \mathsf{succ}(n)) \equiv e_s(n, \mathsf{ind}_\mathbb{N}(E, e_z, e_s, n))$ 对于任意 $n : \mathbb{N}$。

因此，依赖函数 $\mathsf{ind}_\mathbb{N}(E, e_z, e_s)$ 可以理解为在参数 $x : \mathbb{N}$ 上递归定义，通过我们称为 **递归** (recurrences) 的函数 $e_z$ 和 $e_s$。当 $x$ 为零时，函数仅返回 $e_z$。当 $x$ 是另一个自然数 $n$ 的后继时，结果是通过取递归 $e_s$ 并替换特定的前任 $n$ 以及递归调用值 $\mathsf{ind}_\mathbb{N}(E, e_z, e_s, n)$ 来获得的。

上述所有示例的归纳原理都具有这种家族相似性。在 §5.6 中，我们将讨论"归纳定义"的一般概念以及如何为其推导出适当的 *归纳原理* (induction principle)，但首先我们将探讨归纳定义之间的各种共性。

例如，我们在 Chapter 1 的每个示例中都提到，通过归纳原理我们可以推导出 *递归原理* (recursion principle)，其中目标域是简单类型（而非族）。虽然归纳和递归原理可能看起来很奇怪，因为它们仅仅产生了一个函数的 *存在性*，而似乎没有唯一地描述它。然而，事实上，归纳原理足够强大，可以证明其自身的 *唯一性原理* (uniqueness principle)，如下定理所示。

Theorem 5.1.1. *设 $f, g : \prod_{(x:\mathbb{N})} E(x)$ 为两个函数，它们满足以下递归关系*

$$e_z : E(0) \qquad \text{和} \qquad e_s : \prod_{n:\mathbb{N}} E(n) \to E(\mathsf{succ}(n))$$

*并且满足命题等价，即*

$$f(0) = e_z \qquad \text{和} \qquad g(0) = e_z$$

*以及*

$$\prod_{n:\mathbb{N}} f(\mathsf{succ}(n)) = e_s(n, f(n)),$$
$$\prod_{n:\mathbb{N}} g(\mathsf{succ}(n)) = e_s(n, g(n))$$

*那么 $f$ 和 $g$ 是相等的。*

证明. 我们对类型族 $D(x) :\equiv f(x) = g(x)$ 使用归纳法。对于基础情况，我们有

$$f(0) = e_z = g(0)$$

。对于归纳情况，假设 $n : \mathbb{N}$，使得 $f(n) = g(n)$。然后

$$f(\mathsf{succ}(n)) = e_s(n, f(n)) = e_s(n, g(n)) = g(\mathsf{succ}(n))$$

。第一个和最后一个等式来自于对 $f$ 和 $g$ 的假设。中间等式来自于归纳假设以及应用保持等式的事实。这给出了 $f$ 和 $g$ 之间的逐点等价；调用函数扩展性 (function extensionality) 完成证明。 □

请注意，唯一性原理甚至适用于仅在命题等价下满足递归关系的函数，即路径。当然，从归纳原理得到的特定函数在判断上 (judgmentally) 满足这些递归关系；我们将在 §5.5 中回到这一点。另一方面，该定理本身仅断言函数之间的命题等价（另见 Exercise 5.2）。从同伦 (homotopical) 视角来看，询问这个路径是否 一致，即 $f = g$ 的等价是否唯一存在是自然的；我们将在 §5.4 中看到这确实是事实。当然，类似的函数唯一性定理通常也可以为其他归纳类型表述和证明。在下一节中，我们将展示这一唯一性性质与同一性 (univalence) 结合如何表明，像自然数这样的归纳类型完全由其引入、消去和计算规则表征。

## 5.2 归纳类型的唯一性 (Uniqueness of inductive types)

我们已经定义了“自然数 (natural numbers)”作为一个特定类型 $\mathbb{N}$ ，它具有特定的归纳生成器 0 和 $\mathsf{succ}$。然而，根据上一节中描述的类型论中归纳定义的一般原则，并没有什么阻止我们以相同的方式定义 另一个类型。也就是说，假设我们定义 $\mathbb{N}'$ 为由以下构造器生成的归纳类型：

- $0' : \mathbb{N}'$
- $\mathsf{succ}' : \mathbb{N}' \to \mathbb{N}'$。

那么 $\mathbb{N}'$ 将具有与 $\mathbb{N}$ 相同的归纳和递归原理。在证明关于所有这些“新”自然数的命题 $E : \mathbb{N}' \to \mathcal{U}$ 时，只需给出证明 $e_z : E(0')$ 和 $e_s : \prod_{(n:\mathbb{N}')} E(n) \to E(\mathsf{succ}'(n))$。函数 $\mathsf{rec}_{\mathbb{N}'}(E, e_z, e_s) : \prod_{(n:\mathbb{N}')} E(n)$ 的计算规则如下：

- $\mathsf{rec}_{\mathbb{N}'}(E, e_z, e_s, 0') \equiv e_z$，
- $\mathsf{rec}_{\mathbb{N}'}(E, e_z, e_s, \mathsf{succ}'(n)) \equiv e_s(n, \mathsf{rec}_{\mathbb{N}'}(E, e_z, e_s, n))$ 对于任意 $n : \mathbb{N}'$。

但是 $\mathbb{N}$ 和 $\mathbb{N}'$ 之间的关系是什么?
这不仅仅是一个学术问题，因为类似于自然数的结构可以在许多其他地方找到。例如，我们可以将自然数与具有一个元素的类型上的列表相对应（这是可以说的最早出现的形式，发现于洞穴墙壁上），与非负整数 (non-negative integers)、有理数 (rationals) 和实数 (reals) 的子集相对应，等等。从编程的角度来看，我们的自然数的“一元 (unary)”表示是非常低效的，因此有时我们可能更喜欢使用二进制表示。我们希望能够将所有这些版本的“自然数”彼此识别，以便在它们之间传递构造和结果。
当然，如果两个版本的自然数满足相同的归纳原理，那么它们将具有相同的诱导结构。例如，回想在 §1.9 中定义的函数 $\mathsf{double}$ 的例子。对于我们的新自然数，可以通过复制并添加引号来轻松定义类似的函数：

$$\mathsf{double}' :\equiv \mathsf{rec}_{\mathbb{N}'}(\mathbb{N}', 0', \lambda n.\, \lambda m.\, \mathsf{succ}'(\mathsf{succ}'(m)))$$

看似简单，但它带来了显而易见的缺点，即导致大量重复。不仅函数必须被重复，所有引理和它们的证明也必须被“加引号”。
在传统数学中，人们只是宣布 $\mathbb{N}$ 和 $\mathbb{N}'$ 显然是“相同的”，并且可以根据需要将它们互换。这通常没有问题，但它掩盖了相当多的细节，扩大了非正式数学与其精确描述之间的差距。在同伦类型论 (homotopy type theory) 中，我们可以做得更好。
首先观察我们可以定义以下映射：

- $f :\equiv \mathsf{rec}_{\mathbb{N}}(\mathbb{N}', 0', \lambda n.\mathsf{succ}') : \mathbb{N} \to \mathbb{N}'$,
- $g :\equiv \mathsf{rec}_{\mathbb{N}'}(\mathbb{N}, 0, \lambda n.\mathsf{succ}) : \mathbb{N}' \to \mathbb{N}$。

由于 $g$ 和 $f$ 的复合满足与 $\mathbb{N}$ 上的恒等函数相同的递归关系，Theorem 5.1.1 得出 $\prod_{(n:\mathbb{N})} g(f(n)) = n$，而同一定理的"加引号"版本得出 $\prod_{(n:\mathbb{N}')} f(g(n)) = n$。因此，$f$ 和 $g$ 是准逆 (quasi-inverses)，因此 $\mathbb{N} \simeq \mathbb{N}'$。我们现在可以沿着这个等价将 $\mathbb{N}$ 上的函数直接转移到 $\mathbb{N}'$ 上（反之亦然），例如

$$\mathsf{double}' :\equiv \lambda n. f(\mathsf{double}(g(n)))$$

不难证明这个版本的 $\mathsf{double}'$ 与前一个版本相等。
当然，这并不令人惊讶；这种同构正是数学家在设想"识别"$\mathbb{N}$ 与 $\mathbb{N}'$ 时的方式。然而，"通过同构转移 (transfer across an isomorphism)"的机制取决于被转移的对象；它并不总是像将单个函数与 $f$ 和 $g$ 预先和后续组合那样简单。考虑例如一个简单的引理：

$$\prod_{n:\mathbb{N}'} \mathsf{double}'(\mathsf{succ}'(n)) = \mathsf{succ}'(\mathsf{succ}'(\mathsf{double}'(n)))$$

插入正确的 $f$s 和 $g$s 仅仅比直接在 $n : \mathbb{N}'$ 上重新证明它稍微容易一点。
在此处，同一性公理 (univalence axiom) 发挥作用：由于 $\mathbb{N} \simeq \mathbb{N}'$，我们还具有 $\mathbb{N} =_{\mathcal{U}} \mathbb{N}'$，即 $\mathbb{N}$ 和 $\mathbb{N}'$ 作为类型是 *相等的*。现在，身份 (identity) 的归纳原理保证了与 $\mathbb{N}$ 相关的任何构造或证明都可以自动转移到 $\mathbb{N}'$ 上。我们只需将函数或定理的类型视为一个以类型为索引的类型族 $P : \mathcal{U} \to \mathcal{U}$，给定的对象是 $P(\mathbb{N})$ 的一个元素，并沿着路径 $\mathbb{N} = \mathbb{N}'$ 传输即可。这涉及的开销要少得多。
为了简单起见，我们在两个具有 *相同*定义的类型 $\mathbb{N}$ 和 $\mathbb{N}'$ 的情况下描述了这种方法。然而，在实践中更常见的情况是定义并不完全相同，但无论如何，一个归纳原理蕴涵了另一个。例如，考虑从一个单元素类型构造的列表类型 $\mathsf{List}(\mathbf{1})$，它由以下内容生成：

- 一个元素 $\mathsf{nil} : \mathsf{List}(\mathbf{1})$，以及
- 一个函数 $\mathsf{cons} : \mathbf{1} \times \mathsf{List}(\mathbf{1}) \to \mathsf{List}(\mathbf{1})$。

这与 $\mathbb{N}$ 的定义不同，它并没有产生相同的归纳原理。$\mathsf{List}(\mathbf{1})$ 的归纳原理表明，对于任意 $E : \mathsf{List}(\mathbf{1}) \to \mathcal{U}$ 及其递归数据 $e_{\mathsf{nil}} : E(\mathsf{nil})$ 和 $e_{\mathsf{cons}} : \prod_{(u:\mathbf{1})} \prod_{(\ell:\mathsf{List}(\mathbf{1}))} E(\ell) \to E(\mathsf{cons}(u,\ell))$，存在 $f : \prod_{(\ell:\mathsf{List}(\mathbf{1}))} E(\ell)$，使得 $f(\mathsf{nil}) \equiv e_{\mathsf{nil}}$ 且 $f(\mathsf{cons}(u,\ell)) \equiv e_{\mathsf{cons}}(u,\ell,f(\ell))$。（我们将在 §5.6 中看到如何推导归纳定义的归纳原理。）
现在假设我们定义 $0'' :\equiv \mathsf{nil} : \mathsf{List}(\mathbf{1})$，并通过 $\mathsf{succ}''(\ell) :\equiv \mathsf{cons}(\star,\ell)$ 定义 $\mathsf{succ}'' : \mathsf{List}(\mathbf{1}) \to \mathsf{List}(\mathbf{1})$。那么，对于任意 $E : \mathsf{List}(\mathbf{1}) \to \mathcal{U}$ 及其 $e_0 : E(0'')$ 和 $e_s : \prod_{(\ell:\mathsf{List}(\mathbf{1}))} E(\ell) \to E(\mathsf{succ}''(\ell))$，我们可以定义

$$e_{\mathsf{nil}} :\equiv e_0$$
$$e_{\mathsf{cons}}(\star,\ell,x) :\equiv e_s(\ell,x)$$

（在 $e_{\mathsf{cons}}$ 的定义中，我们使用 $\mathbf{1}$ 的归纳原理假设 $u$ 是 $\star$。）现在我们可以应用 $\mathsf{List}(\mathbf{1})$ 的归纳原理，得到 $f : \prod_{(\ell:\mathsf{List}(\mathbf{1}))} E(\ell)$，使得

$$f(0'') \equiv f(\mathsf{nil}) \equiv e_{\mathsf{nil}} \equiv e_0$$
$$f(\mathsf{succ}''(\ell)) \equiv f(\mathsf{cons}(\star,\ell)) \equiv e_{\mathsf{cons}}(\star,\ell,f(\ell)) \equiv e_s(\ell,f(\ell))$$

因此，$\mathsf{List}(\mathbf{1})$ 满足与 $\mathbb{N}$ 相同的归纳原理，因此（根据上述相同的论证）与其相等。
最后，这些结论并不局限于自然数：它们适用于任何归纳类型。如果我们有一个归纳定义的类型 $W$，以及另一个类型 $W'$，它满足与 $W$ 相同的归纳原理，那么可以推断出 $W \simeq W'$，因此 $W = W'$。我们使用为 $W$ 和 $W'$ 推导出的递归原理构造映射 $W \to W'$ 和 $W' \to W$，然后使用每个的归纳原理证明两个复合等于恒等映射。例如，在 Chapter 1 中，我们看到可以将 $A + B$ 的并 (coproduct) 也定义为 $\sum_{(x:\mathbf{2})} \mathsf{rec}_{\mathbf{2}}(\mathcal{U}, A, B, x)$。后者的类型满足与前者相同的归纳原理；因此它们是规范等价的。
这当然与范畴论 (category theory) 中的熟悉事实非常相似，即如果两个对象具有相同的 *泛性质* (universal property)，那么它们是等价的。在 §5.4 中我们将看到归纳类型实际上具有一个泛性质，因此这是该一般原则的表现形式。

## 5.3 W-类型 (W-types)

归纳类型非常广泛，这对于它们的实用性和适用性来说是极好的，但使它们整体上难以研究。幸运的是，它们都可以形式化地简化为几个特殊情况。本书的范围之外不会讨论这种简化——这对在实践中使用类型论的数学家来说也无关紧要——但我们将花一些时间讨论我们尚未遇到的基本特殊情况之一。这些是 Martin-Löf 的 W-*类型*，也称为 *良基树* (well-founded trees) 类型。W-类型是自然数、列表和二叉树等类型的推广，它们足够通用，能够封装 *任何*归纳类型的"递归"方面。

一个特定的 W-类型由给定的两个参数 $A : \mathcal{U}$ 和 $B : A \to \mathcal{U}$ 指定，在这种情况下，得到的 W-类型写作 $\mathsf{W}_{(a:A)}B(a)$。类型 $A$ 表示 $\mathsf{W}_{(a:A)}B(a)$ 的 *标签* (labels) 类型，它们作为构造器 (constructors)（然而，我们保留这个词用于归纳定义中出现的实际函数）。例如，在将自然数定义为 W-类型时，类型 $A$ 是由两个元素 $0_2$ 和 $1_2$ 组成的类型 $\mathbf{2}$，因为获得自然数的方式恰好有两种——它将是 0 或者是另一个自然数的继承者。

类型族 $B : A \to \mathcal{U}$ 用于记录标签的元数 (arity)：一个标签 $a : A$ 将采用一个以 $B(a)$ 为索引的归纳参数族。我们可以因此将其理解为"$B(a)$-多个"标签的参数。这些参数由一个函数 $f : B(a) \to \mathsf{W}_{(a:A)}B(a)$ 表示，理解为对于任意 $b : B(a)$，$f(b)$ 是标签 $a$ 的第"$b$"个参数。因此，W-类型 $\mathsf{W}_{(a:A)}B(a)$ 可以被视为良基树的类型，其中节点由 $A$ 的元素标记，每个由 $a : A$ 标记的节点具有 $B(a)$-多个分支。

在自然数的情况下，标签 $0_2$ 具有 0 元数，因为它构造了常量 0；标签 $1_2$ 具有 1 元数，因为它构造了其参数的继承者。我们可以通过对 $\mathbf{2}$ 的简单消去定义一个函数 $\mathsf{rec}_2(\mathcal{U}, \mathbf{0}, \mathbf{1})$ 到类型宇宙；这个函数对于 $0_2$ 返回空类型 $\mathbf{0}$，对于 $1_2$ 返回单位类型 $\mathbf{1}$。因此我们可以定义

$$\mathbf{N}^{\mathbf{w}} :\equiv \mathsf{W}_{(b:\mathbf{2})}\mathsf{rec}_2(\mathcal{U}, \mathbf{0}, \mathbf{1}, b)$$

其中上标 $\mathbf{w}$ 用于将这种自然数版本与之前使用的版本区分开来。类似地，我们可以将类型 $A$ 上的列表 定义为具有 $\mathbf{1} + A$ 个标签的 W-类型：一个用于空列表的零元标签，再加上每个 $a : A$ 的一个一元标签，对应于将 $a$ 添加到列表的头部：

$$\mathsf{List}(A) :\equiv \mathsf{W}_{(x:\mathbf{1}+A)}\mathsf{rec}_{\mathbf{1}+A}(\mathcal{U},\ \mathbf{0},\ \lambda a.\mathbf{1},\ x)$$

。一般来说，由 $A : \mathcal{U}$ 和 $B : A \to \mathcal{U}$ 指定的 W-类型 $\mathsf{W}_{(x:A)}B(x)$ 是由以下构造器生成的归纳类型：

- $\mathsf{sup} : \prod_{(a:A)} \left(B(a) \to \mathsf{W}_{(x:A)}B(x)\right) \to \mathsf{W}_{(x:A)}B(x)$。

构造器 $\mathsf{sup}$（"上确界 (supremum)"的缩写）接受一个标签 $a : A$ 和一个函数 $f : B(a) \to \mathsf{W}_{(x:A)}B(x)$，该函数表示标签 $a$ 的参数，并构造出 $\mathsf{W}_{(x:A)}B(x)$ 的一个新元素。使用我们之前将自然数编码为 W-类型的方法，我们可以例如定义

$$0^{\mathbf{w}} :\equiv \mathsf{sup}(0_2,\ \lambda x.\, \mathsf{rec}_{\mathbf{0}}(\mathbf{N}^{\mathbf{w}}, x))$$

换句话说，我们使用标签 $0_2$ 来构造 $0^{\mathbf{w}}$。然后，$\mathsf{rec}_2(\mathcal{U}, \mathbf{0}, \mathbf{1}, 0_2)$ 计算为 $\mathbf{0}$，正如它应该的那样，因为 $0_2$ 是一个零元标签。因此，我们需要构造一个函数 $f : \mathbf{0} \to \mathbf{N}^{\mathbf{w}}$，该函数表示提供给 $0_2$ 的（零个）参数。这当然是微不足道的，使用对 $\mathbf{0}$ 的简单消去就能实现。同样地，我们可以定义 $1^{\mathbf{w}}$ 和一个继承者函数 $\mathsf{succ}^{\mathbf{w}}$

$$1^{\mathbf{w}} :\equiv \mathsf{sup}(1_2,\ \lambda x.\, 0^{\mathbf{w}})$$
$$\mathsf{succ}^{\mathbf{w}} :\equiv \lambda n.\, \mathsf{sup}(1_2,\ \lambda x.\, n)$$

我们有以下针对 W-类型的归纳原理：

- 在证明关于所有 W-类型 $\mathsf{W}_{(x:A)}B(x)$ 元素的命题 $E : \left(\mathsf{W}_{(x:A)}B(x)\right) \to \mathcal{U}$ 时，只需证明对于 $\mathsf{sup}(a, f)$ 的情况即可，假设它对所有 $b : B(a)$ 的 $f(b)$ 都成立。换句话说，只需给出一个证明

$$e : \prod_{(a:A)} \prod_{(f:B(a)\to \mathsf{W}_{(x:A)}B(x))} \prod_{(g:\prod_{(b:B(a))} E(f(b)))} E(\mathsf{sup}(a, f))$$

变量 $g$ 代表我们的归纳假设，即所有标签 $a$ 的参数都满足 $E$。为了陈述这一点，我们对所有类型 $B(a)$ 的元素进行量化，因为每个 $b : B(a)$ 对应于标签 $a$ 的一个参数 $f(b)$。

我们如何在编码为 W-类型的自然数上定义函数 double？我们希望使用 $\mathbf{N}^{\mathbf{w}}$ 的递归原理，其余元类型为 $\mathbf{N}^{\mathbf{w}}$ 本身。因此我们需要构造一个合适的函数

$$e : \prod_{(a:\mathbf{2})} \prod_{(f:B(a)\to\mathbf{N}^{\mathbf{w}})} \prod_{(g:B(a)\to\mathbf{N}^{\mathbf{w}})} \mathbf{N}^{\mathbf{w}}$$

它将表示 double 函数的递归；为了简化，我们将类型族 $\mathsf{rec}_{\mathbf{2}}(\mathcal{U}, \mathbf{0}, \mathbf{1})$ 表示为 $B$。

显然，$e$ 将是一个以 $a : \mathbf{2}$ 为第一个参数的函数。下一步是对 $a$ 进行案例分析，并根据它是 $0_{\mathbf{2}}$ 还是 $1_{\mathbf{2}}$ 进行处理。这表明以下形式的定义

$$e :\equiv \lambda a.\, \mathsf{ind}_{\mathbf{2}}(C, e_0, e_1, a)$$

其中

$$C :\equiv \lambda a.\, \prod_{(f:B(a)\to\mathbf{N}^{\mathbf{w}})} \prod_{(g:B(a)\to\mathbf{N}^{\mathbf{w}})} \mathbf{N}^{\mathbf{w}}$$

如果 $a$ 是 $0_{\mathbf{2}}$，则类型 $B(a)$ 变为 $\mathbf{0}$。因此，给定 $f : \mathbf{0} \to \mathbf{N}^{\mathbf{w}}$ 和 $g : \mathbf{0} \to \mathbf{N}^{\mathbf{w}}$，我们想要构造 $\mathbf{N}^{\mathbf{w}}$ 的一个元素。由于标签 $0_{\mathbf{2}}$ 表示 $\mathbf{0}$，它需要零个归纳参数，变量 $f$ 和 $g$ 是无关紧要的。我们返回 $0^{\mathbf{w}}$ 作为结果：

$$e_0 :\equiv \lambda f.\, \lambda g.\, 0^{\mathbf{w}}$$

类似地，如果 $a$ 是 $1_{\mathbf{2}}$，则类型 $B(a)$ 变为 $\mathbf{1}$。由于标签 $1_{\mathbf{2}}$ 表示继承者 操作符，它需要一个归纳参数——前驱——由变量 $f : \mathbf{1} \to \mathbf{N}^{\mathbf{w}}$ 表示。对前驱的递归调用的值由变量 $g : \mathbf{1} \to \mathbf{N}^{\mathbf{w}}$ 表示。因此，取这个值（即 $g(\star)$）并将继承者函数应用两次即可得到所需的结果：

$$e_1 :\equiv\ \lambda f.\, \lambda g.\, \mathsf{succ}^{\mathbf{w}}(\mathsf{succ}^{\mathbf{w}}(g(\star)))$$

将这些结合在一起，我们有

$$\mathsf{double} :\equiv \mathsf{rec}_{\mathbf{N}^{\mathbf{w}}}(\mathbf{N}^{\mathbf{w}}, e)$$

其中 $e$ 的定义如上。

函数 $\mathsf{rec}_{\mathsf{W}_{(x:A)}B(x)}(E, e) : \prod_{(w:\mathsf{W}_{(x:A)}B(x))} E(w)$ 的关联计算规则如下。

- 对于任意 $a : A$ 和 $f : B(a) \to \mathsf{W}_{(x:A)}B(x)$，我们有

$$\mathsf{rec}_{\mathsf{W}_{(x:A)}B(x)}(E, e, \mathsf{sup}(a, f)) \equiv e(a, f, (\lambda b.\, \mathsf{rec}_{\mathsf{W}_{(x:A)}B(x)}(E, e, f(b))))$$

换句话说，函数 $\mathsf{rec}_{\mathsf{W}_{(x:A)}B(x)}(E, e)$ 满足递归 $e$。

根据上述计算规则，函数 double 表现如预期：

$$\begin{aligned}
\mathsf{double}(0^{\mathbf{w}}) &\equiv \mathsf{rec}_{\mathbf{N}^{\mathbf{w}}}(\mathbf{N}^{\mathbf{w}}, e, \mathsf{sup}(0_{\mathbf{2}},\ \lambda x.\, \mathsf{rec}_{\mathbf{0}}(\mathbf{N}^{\mathbf{w}}, x))) \\
&\equiv e(0_{\mathbf{2}}, (\lambda x.\, \mathsf{rec}_{\mathbf{0}}(\mathbf{N}^{\mathbf{w}}, x)), (\lambda x.\, \mathsf{double}(\mathsf{rec}_{\mathbf{0}}(\mathbf{N}^{\mathbf{w}}, x)))) \\
&\equiv e_0((\lambda x.\, \mathsf{rec}_{\mathbf{0}}(\mathbf{N}^{\mathbf{w}}, x)), (\lambda x.\, \mathsf{double}(\mathsf{rec}_{\mathbf{0}}(\mathbf{N}^{\mathbf{w}}, x)))) \\
&\equiv 0^{\mathbf{w}}
\end{aligned}$$

并且

$$\begin{aligned}
\mathsf{double}(1^{\mathbf{w}}) &\equiv \mathsf{rec}_{\mathbf{N}^{\mathbf{w}}}(\mathbf{N}^{\mathbf{w}}, e, \mathsf{sup}(1_{\mathbf{2}},\ \lambda x.\, 0^{\mathbf{w}})) \\
&\equiv e(1_{\mathbf{2}}, (\lambda x.\, 0^{\mathbf{w}}), (\lambda x.\, \mathsf{double}(0^{\mathbf{w}}))) \\
&\equiv e_1((\lambda x.\, 0^{\mathbf{w}}), (\lambda x.\, \mathsf{double}(0^{\mathbf{w}}))) \\
&\equiv \mathsf{succ}^{\mathbf{w}}(\mathsf{succ}^{\mathbf{w}}((\lambda x.\, \mathsf{double}(0^{\mathbf{w}}))(\star))) \\
&\equiv \mathsf{succ}^{\mathbf{w}}(\mathsf{succ}^{\mathbf{w}}(0^{\mathbf{w}}))
\end{aligned}$$

等等。

就像自然数一样，我们可以证明 W-类型的唯一性定理：

Theorem 5.3.1. 设 $g, h : \prod_{(w:\mathsf{W}_{(x:A)}B(x))} E(w)$ 是两个满足递归

$$e : \prod_{a,f}\Big(\prod_{b:B(a)} E(f(b))\Big) \to E(\mathsf{sup}(a,f))$$

命题的函数，即满足

$$\prod_{a,f} g(\mathsf{sup}(a,f)) = e(a,f,\lambda b.\, g(f(b)))$$
$$\prod_{a,f} h(\mathsf{sup}(a,f)) = e(a,f,\lambda b.\, h(f(b)))$$

然后 $g$ 和 $h$ 是相等的。

## 5.4 归纳类型是初始代数 (Inductive types are initial algebras)

如前所述，归纳类型 (Inductive types) 也具有范畴论中的一种普遍性质。它们是同伦初始代数 (homotopy-initial algebras)：在由特定构造子 (constructors) 决定的“代数 (algebras)”范畴中，同伦初始代数是初始对象（在一致同伦 (coherent homotopy) 范畴中）。举一个简单的例子，考虑自然数 (natural numbers)。这里适当的“代数”是一种类型，具有与 $\mathbb{N}$ 的构造子所赋予的相同结构。

Definition 5.4.1. $\mathbb{N}$-**代数** ($\mathbb{N}$-algebra) 是一种类型 $C$，具有两个元素 $c_0 : C$ 和 $c_s : C \to C$。这种代数的类型为：

$$\mathbb{N}\mathsf{Alg} :\equiv \sum_{C:\mathcal{U}} C \times (C \to C).$$

Definition 5.4.2. $\mathbb{N}$-**同态** ($\mathbb{N}$-homomorphism) 在 $\mathbb{N}$-代数 $(C, c_0, c_s)$ 和 $(D, d_0, d_s)$ 之间的同态是一个函数 $h : C \to D$，使得对所有 $c : C$，都有 $h(c_0) = d_0$ 且 $h(c_s(c)) = d_s(h(c))$。这些同态的类型为：

$$\mathbb{N}\mathsf{Hom}((C,c_0,c_s),(D,d_0,d_s)) :\equiv \sum_{(h:C\to D)} (h(c_0) = d_0) \times \textstyle\prod_{(c:C)} (h(c_s(c)) = d_s(h(c))).$$

因此，我们有一个 $\mathbb{N}$-代数和 $\mathbb{N}$-同态的范畴，$\mathbb{N}$ 是这个范畴的初始对象。范畴论专家会立即认识到这就是范畴中自然数对象的定义。

当然，由于我们的类型表现得像 $\infty$-群体 ($\infty$-groupoids)，我们实际上有一个 $\mathbb{N}$-代数的 $(\infty,1)$-范畴，我们应当要求 $\mathbb{N}$ 在合适的 $(\infty,1)$-范畴意义上是初始的。幸运的是，我们可以在不需要定义 $(\infty,1)$-范畴的情况下表述这一点。

Definition 5.4.3. 称 $\mathbb{N}$-代数 $I$ 为**同伦初始** (homotopy-initial)，简称 h-initial，如果对于任意其他的 $\mathbb{N}$-代数 $C$，从 $I$ 到 $C$ 的 $\mathbb{N}$-同态类型是可收缩的。即：

$$\mathsf{isHinit}_{\mathbb{N}}(I) :\equiv \prod_{C:\mathbb{N}\mathsf{Alg}} \mathsf{isContr}(\mathbb{N}\mathsf{Hom}(I,C)).$$

当它们存在时，同伦初始代数是唯一的——不仅仅是同构意义上的唯一，而是通过单值性公理 (univalence axiom) 来实现的等同。

Theorem 5.4.4. 任何两个同伦初始的 $\mathbb{N}$-代数是相等的。因此，同伦初始的 $\mathbb{N}$-代数的类型是一个单命题 (mere proposition)。

证明. 假设 $I$ 和 $J$ 是同伦初始的 $\mathbb{N}$-代数。那么 $\mathbb{N}\mathsf{Hom}(I,J)$ 是可收缩的，因此包含某个 $\mathbb{N}$-同态 $f : I \to J$，同样我们有一个 $\mathbb{N}$-同态 $g : J \to I$。现在，复合 $g \circ f$ 是从 $I$ 到 $I$ 的 $\mathbb{N}$-同态，$\mathsf{id}_I$ 也是如此；但是 $\mathbb{N}\mathsf{Hom}(I,I)$ 是可收缩的，所以 $g \circ f = \mathsf{id}_I$。同样，$f \circ g = \mathsf{id}_J$。因此 $I \simeq J$，也就是说 $I = J$。由于可收缩性是一个单命题，并且依赖乘积保留单命题，因此同伦初始也是一个单命题。因此，任何两个证明 $I$（或 $J$）是同伦初始的证明必然是相等的，从而完成了证明。 □

现在我们有了以下定理。

Theorem 5.4.5. $\mathbb{N}$-代数 $(\mathbb{N}, \mathbf{0}, \mathsf{succ})$ 是同伦初始的。

证明概述. 固定任意 $\mathbb{N}$-代数 $(C, c_0, c_s)$。$\mathbb{N}$ 的递归原理给出了一个函数 $f : \mathbb{N} \to C$，定义如下：

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

这两个等式使得 $f$ 成为一个 $\mathbb{N}$-同态，我们可以将其作为 $\mathbb{N}\mathsf{Hom}(\mathbb{N}, C)$ 的收缩中心。唯一性定理（见 Theorem 5.1.1）然后表明，任何其他的 $\mathbb{N}$-同态都等于 $f$。 □

为了将其置于更一般的背景下，考虑自函子 (endofunctor) 的代数的概念。注意，使一种类型 $C$ 成为 $\mathbb{N}$-代数与给出一个函数 $c : C + \mathbf{1} \to C$ 是等价的，并且一个函数 $f : C \to D$ 是 $\mathbb{N}$-同态，当且仅当 $f \circ c \sim d \circ (f + \mathbf{1})$。在范畴语言中，这意味着 $\mathbb{N}$-代数是类型范畴中自函子 $F(X) :\equiv X + 1$ 的代数。对于更一般的情况，考虑与 $A : \mathcal{U}$ 和 $B : A \to \mathcal{U}$ 关联的 W-类型 (W-types)。在这种情况下，我们有一个关联的**多项式函子** (polynomial functor)：

$$P(X) = \sum_{x:A} (B(x) \to X) \tag{5.4.6}$$

实际上，这个赋值仅在同伦意义上是函子的，但这在接下来的讨论中没有影响。根据定义，$P$-**代数** ($P$-algebra)是类型 $C$，配有一个函数 $s_C : PC \to C$。根据 $\Sigma$-类型的通用性质，这相当于给出一个函数 $\prod_{(a:A)}(B(a) \to C) \to C$。我们还将这种对象称为 W-**代数** (W-algebras)，我们记为

$$\mathsf{WAlg}(A, B) :\equiv \sum_{(C:\mathcal{U})} \prod_{(a:A)} (B(a) \to C) \to C$$

类似地，对于 $P$-代数 $(C, s_C)$ 和 $(D, s_D)$，一个 **同态** (homomorphism)在它们之间的同态 $(f, s_f) : (C, s_C) \to (D, s_D)$ 由一个函数 $f : C \to D$ 和一个在 $PC \to D$ 之间的同伦构成

$$s_f : f \circ s_C \ = s_D \circ Pf,$$

其中 $Pf : PC \to PD$ 是 $P$ 在 $f : C \to D$ 上的可轻易定义的作用结果。这样的代数同态可以表示为：

$$\begin{array}{ccc} PC & \xrightarrow{Pf} & PD \\ {\scriptstyle s_C}\downarrow & {\scriptstyle s_f} & \downarrow{\scriptstyle s_D} \\ C & \xrightarrow[f]{} & D \end{array}$$

在元素的层面上，$f$ 是一个 $P$-同态 (或 W-**同态** (W-homomorphism))若

$$f(s_C(a, h)) = s_D(a, f \circ h)$$

我们有 W-同态的类型：

$$\mathsf{WHom}_{A,B}((C, s_C), (D, s_D)) :\equiv \sum_{(f:C\to D)} \prod_{(a:A)} \prod_{(h:B(a)\to C)} f(s_C(a, h)) = s_D(a, f \circ h)$$

最后，一个 $P$-代数 $(C, s_C)$ 被称为**同伦初始** (homotopy-initial)如果对于每个 $P$-代数 $(D, s_D)$，从 $(C, s_C)$ 到 $(D, s_D)$ 的所有代数同态的类型是可收缩的。即：

$$\mathsf{isHinit}_{\mathsf{W}}(A, B, I) :\equiv \prod_{C:\mathsf{WAlg}(A,B)} \mathsf{isContr}(\mathsf{WHom}_{A,B}(I, C))$$

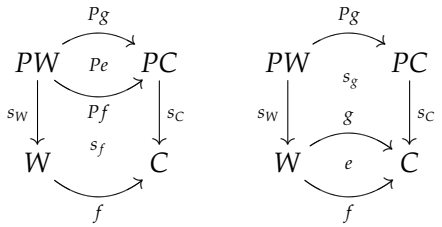
现在与 Theorem 5.4.5 类似的定理是：

Theorem 5.4.7. *对于任意类型 $A:\mathcal{U}$ 和类型族 $B:A\to\mathcal{U}$，W-代数 $(\mathsf{W}_{(x:A)}B(x),\mathsf{sup})$ 是同伦初始的。*

*证明概述.* 假设我们有 $A:\mathcal{U}$ 和 $B:A\to\mathcal{U}$，并考虑关联的多项式函子 $P(X):\equiv\sum_{(x:A)}(B(x)\to X)$。令 $W:\equiv\mathsf{W}_{(x:A)}B(x)$。然后使用 §5.3 中的 W-引入规则，我们有一个结构映射 $s_W:\equiv\mathsf{sup}:PW\to W$。我们要证明代数 $(W,s_W)$ 是同伦初始的。因此，让我们考虑另一个代数 $(C,s_C)$ 并证明 W-同态从 $(W,s_W)$ 到 $(C,s_C)$ 的类型 $T:\equiv\mathsf{WHom}_{A,B}((W,s_W),(C,s_C))$ 是可收缩的。为此，观察到 W-消去规则和 W-计算规则允许我们定义一个 W-同态 $(f,s_f):(W,s_W)\to(C,s_C)$，从而表明 $T$ 是可居住的。此外，有必要表明，对于每个 W-同态 $(g,s_g):(W,s_W)\to(C,s_C)$，存在一个等式证明

$$p:(f,s_f)=(g,s_g) \tag{5.4.8}$$

这利用了一个事实，即一般来说，$(f,s_f)=(g,s_g)$ 形式的类型等价于我们所称的从 $f$ 到 $g$ 的 **代数 2-胞腔** (algebra 2-cells)的类型，其标准元素是 $(e,s_e)$ 形式的对，其中 $e:f=g$ 并且 $s_e$ 是在以下粘贴图所代表的等式证明之间的更高等式证明：




鉴于此事实，为证明存在 (5.4.8) 中的元素，足以证明存在从 $f$ 到 $g$ 的代数 2-胞腔 $(e,s_e)$。等式证明 $e:f=g$ 现在通过函数扩展性和 W-消去法则构造，以保证所需的等式证明 $s_e$ 的存在。 □

## 5.5 同伦归纳类型 (Homotopy-inductive types)

在 §5.3 中，我们展示了如何将自然数 (natural numbers) 编码为 W-类型 (W-types)，如下所示：

$$\begin{aligned}\mathbf{N}^{\mathbf{w}}&:\equiv\mathsf{W}_{(b:\mathbf{2})}\mathsf{rec}_{\mathbf{2}}(\mathcal{U},\mathbf{0},\mathbf{1},b),\\ 0^{\mathbf{w}}&:\equiv\mathsf{sup}(0_{\mathbf{2}},(\lambda x.\,\mathsf{rec}_{\mathbf{0}}(\mathbf{N}^{\mathbf{w}},x))),\\ \mathsf{succ}^{\mathbf{w}}&:\equiv\lambda n.\,\mathsf{sup}(1_{\mathbf{2}},(\lambda x.\,n))\end{aligned}$$

我们还展示了如何使用递归原理 (recursion principle) 在 $\mathbf{N}^{\mathbf{w}}$ 上定义 double 函数。然而，当涉及到归纳原理 (induction principle) 时，这种编码不再令人满意：给定 $E:\mathbf{N}^{\mathbf{w}}\to\mathcal{U}$ 和递归 $e_z:E(0^{\mathbf{w}})$ 以及 $e_s:\prod_{(n:\mathbf{N}^{\mathbf{w}})}E(n)\to E(\mathsf{succ}^{\mathbf{w}}(n))$，我们只能构造出一个依赖函数 $r(E,e_z,e_s):\prod_{(n:\mathbf{N}^{\mathbf{w}})}E(n)$，它仅在命题意义上（即通过路径）满足给定的递归。这意味着无法从 W-类型的规则中显然推导出自然数的计算规则 (computation rules)，这些规则给出的是判断相等性 (judgmental equalities)。

当我们考虑*同伦归纳类型* (homotopy-inductive types) 时，这个问题消失了。在同伦归纳类型中，所有的计算规则都是在路径意义上陈述的，即符号 $\equiv$ 被替换为 $=$。例如，同伦版 W-类型 $\mathsf{W}^{\mathsf{h}}$ 的计算规则变为：

- 对于任何 $a:A$ 和 $f:B(a)\to\mathsf{W}^h_{(x:A)}B(x)$，我们有

$$\mathsf{rec}_{\mathsf{W}^h_{(x:A)}B(x)}(E,e,\mathsf{sup}(a,f))=e\Big(a,f,\big(\lambda b.\,\mathsf{rec}_{\mathsf{W}^h_{(x:A)}B(x)}(E,f(b))\big)\Big)$$

同伦归纳类型在计算性质方面有一个显而易见的缺点——通过归纳原理构造的任何函数的行为现在只能通过命题性方式描述。但是，其他许多考虑因素也驱使我们考虑同伦归纳类型。例如，尽管我们在 §5.4 中证明了归纳类型是同伦初始代数，但并非每个同伦初始代数都是归纳类型（即满足相应的归纳原理）——但每个同伦初始代数*确实*是同伦归纳类型。类似地，当其中一个（或两个）涉及的类

型仅是同伦归纳类型时，我们可能希望应用来自 §5.2 的唯一性论证——例如，证明 W-类型编码的 $\mathbb{N}$ 等价于通常的 $\mathbb{N}$。

此外，同伦归纳类型的概念现在是类型论内部的。例如，这意味着我们可以形成所有自然数对象的类型并对其进行断言。在 W-类型的情况下，我们可以将同伦 W-类型 $\mathsf{W}_{(x:A)}B(x)$ 表征为任何配有上确界 (supremum) 函数和满足适当（命题性）计算规则的归纳原理的类型：

$$\mathsf{W}_d(A,B) :\equiv \sum_{(W:\mathcal{U})} \sum_{(\mathsf{sup}:\prod_{(a)}(B(a)\to W)\to W)} \prod_{(E:W\to\mathcal{U})} \prod_{(e:\prod_{(a,f)}(\prod_{(b:B(a))} E(f(b)))\to E(\mathsf{sup}(a,f)))} \sum_{(\mathsf{ind}:\prod_{(w:W)} E(w))} \prod_{(a,f)} \mathsf{ind}(\mathsf{sup}(a,f)) = e(a,\lambda b.\,\mathsf{ind}(f(b)))$$

在 Chapter 6 中，我们将看到为什么命题性计算规则值得考虑的其他原因。

在本节中，我们将陈述一些关于同伦归纳类型的基本事实。我们省略了大部分证明，因为它们有些技术性。

Theorem 5.5.1. *对于任何 $A:\mathcal{U}$ 和 $B:A\to\mathcal{U}$，类型 $\mathsf{W}_d(A,B)$ 是一个单命题。*

事实证明，存在 W-类型的另一种等价表征，使用递归原理，加上某些**唯一性** (uniqueness) 和**一致性** (coherence) 法则。首先，我们给出递归原理：

- 当从 W-类型 $\mathsf{W}^h_{(x:A)}B(x)$ 构造一个到类型 $C$ 的函数时，只需给出它在 $\mathsf{sup}(a,f)$ 处的值，假设我们已经给出所有 $f(b)$ 的值，其中 $b:B(a)$。换句话说，只需构造一个函数

$$c:\prod_{a:A}(B(a)\to C)\to C$$

与 $\mathsf{rec}_{\mathsf{W}^h_{(x:A)}B(x)}(C,c):(\mathsf{W}_{(x:A)}B(x))\to C$ 关联的计算规则如下：

- 对于任何 $a:A$ 和 $f:B(a)\to\mathsf{W}^h_{(x:A)}B(x)$，我们有一个等式见证 $\beta(C,c,a,f)$：

$$\mathsf{rec}_{\mathsf{W}^h_{(x:A)}B(x)}(C,c,\mathsf{sup}(a,f)) = c(a,\lambda b.\,\mathsf{rec}_{\mathsf{W}^h_{(x:A)}B(x)}(C,c,f(b)))$$

此外，我们断言以下唯一性原则，表明由相同递归定义的任何两个函数是相等的：

- 设 $C:\mathcal{U}$ 和 $c:\prod_{(a:A)}(B(a)\to C)\to C$ 给定。设 $g,h:(\mathsf{W}^h_{(x:A)}B(x))\to C$ 是两个函数，它们在命题相等性上满足递归 $c$，即我们有

$$\beta_g:\prod_{a,f} g(\mathsf{sup}(a,f)) = c(a,\lambda b.\,g(f(b)))$$
$$\beta_h:\prod_{a,f} h(\mathsf{sup}(a,f)) = c(a,\lambda b.\,h(f(b)))$$

那么 $g$ 和 $h$ 是相等的，即存在 $\alpha(C,c,f,g,\beta_g,\beta_h)$ 类型 $g=h$ 的元素。

回想一下，当我们有一个归纳原理而不仅仅是递归原理时，这种命题唯一性原则是可导出的（见 Theorem 5.3.1）。但仅有递归时，唯一性原则不再是可导出的——事实上，该陈述甚至不是真的（练习题）。因此，我们将其作为一个公理来假设。我们还假设以下一致性法则，告诉我们唯一性证明在典型元素上的行为：

- 对于任何 $a:A$ 和 $f:B(a)\to C$，以下图命题性地交换：

$$\begin{array}{ccc} g(\mathsf{sup}(a,f)) & \xrightarrow{\beta_g} & c(a,\lambda b.\,g(f(b))) \\ {\scriptstyle \alpha(\mathsf{sup}(a,f))}\downarrow & & \downarrow{\scriptstyle c(a,-)(\mathsf{funext}(\lambda b.\,\alpha(f(b))))} \\ h(\mathsf{sup}(a,f)) & \xrightarrow[\beta_h]{} & c(a,\lambda b.\,h(f(b))) \end{array}$$

其中 $\alpha$ 是路径 $\alpha(C,c,f,g,\beta_g,\beta_h):g=h$ 的缩写。

将所有这些数据结合起来得出 $\mathsf{W}_{(x:A)}B(x)$ 的另一种表征，即配有上确界函数、满足简单消去、计算、唯一性和一致性规则的类型：

$$\begin{aligned}\mathsf{W}_s(A,B) :\equiv \sum_{(W:\mathcal{U})} \sum_{(\mathsf{sup}:\prod_{(a)}(B(a)\to W)\to W)} \prod_{(C:\mathcal{U})} \prod_{(c:\prod_{(a)}(B(a)\to C)\to C)} \\ \sum_{(\mathsf{rec}:W\to C)} \sum_{(\beta:\prod_{(a,f)}\mathsf{rec}(\mathsf{sup}(a,f))=c(a,\lambda b.\,\mathsf{rec}(f(b))))} \prod_{(g:W\to C)} \prod_{(h:W\to C)} \prod_{(\beta_g:\prod_{(a,f)}g(\mathsf{sup}(a,f))=c(a,\lambda b.\,g(f(b))))} \\ \prod_{(\beta_h:\prod_{(a,f)}h(\mathsf{sup}(a,f))=c(a,\lambda b.\,h(f(b))))} \sum_{(\alpha:\prod_{(w:W)}g(w)=h(w))} \prod_{(a,f)} \\ \alpha(\mathsf{sup}(a,f))\cdot\beta_h = \beta_g\cdot c(a,-)(\mathsf{funext}\ \lambda b.\,\alpha(f(b)))\end{aligned}$$

Theorem 5.5.2. *对于任意 $A:\mathcal{U}$ 和 $B:A\to\mathcal{U}$，类型 $\mathsf{W}_s(A,B)$ 是一个单命题。*

最后，我们有一个第三个非常简洁的表征，将 $\mathsf{W}_{(x:A)}B(x)$ 作为同伦初始的 W-代数：

$$\mathsf{W}_h(A,B) :\equiv \sum_{I:\mathsf{WAlg}(A,B)} \mathsf{isHinit}_{\mathsf{W}}(A,B,I)$$

Theorem 5.5.3. *对于任意 $A:\mathcal{U}$ 和 $B:A\to\mathcal{U}$，类型 $\mathsf{W}_h(A,B)$ 是一个单命题。*

事实证明，所有三个 W-类型的表征实际上是等价的：

Lemma 5.5.4. *对于任意 $A:\mathcal{U}$ 和 $B:A\to\mathcal{U}$，我们有*

$$\mathsf{W}_d(A,B)\simeq\mathsf{W}_s(A,B)\simeq\mathsf{W}_h(A,B)$$

实际上，我们有以下定理，这比 Theorem 5.4.7 更进一步：

Theorem 5.5.5. *满足 W-类型的构造、引入、消去和命题计算规则的类型正是同伦初始的 W-代数。*

*证明概述.* 检查 Theorem 5.4.7 的证明，我们发现只需要命题性计算规则即可证明 $\mathsf{W}_{(x:A)}B(x)$ 的同伦初始性。对于相反的推论，我们假设关联到 $A:\mathcal{U}$ 和 $B:A\to\mathcal{U}$ 的多项式函子具有一个同伦初始代数 $(W,s_W)$；我们证明 $W$ 满足 W-类型的命题性规则。W-引入规则是简单的；即，对于 $a:A$ 和 $t:B(a)\to W$，我们定义 $\mathsf{sup}(a,t):W$ 为将结构映射 $s_W:PW\to W$ 应用于 $(a,t):PW$ 的结果。对于 W-消去规则，假设其前提，特别是 $C':W\to\mathcal{U}$。使用其他前提，可以证明类型 $C:\equiv\sum_{(w:W)}C'(w)$ 可以配备一个结构映射 $s_C:PC\to C$。通过 $W$ 的同伦初始性，我们获得了一个代数同态 $(f,s_f):(W,s_W)\to(C,s_C)$。此外，第一投影 $\mathsf{pr}_1:C\to W$ 可以配备一个同态的结构，因此我们得到如下形式的图

$$\begin{array}{ccccc} PW & \xrightarrow{Pf} & PC & \xrightarrow{P\mathsf{pr}_1} & PW \\ {\scriptstyle s_W}\downarrow & & \downarrow{\scriptstyle s_C} & & \downarrow{\scriptstyle s_W} \\ W & \xrightarrow[f]{} & C & \xrightarrow[\mathsf{pr}_1]{} & W. \end{array}$$

但是，恒等函数 $1_W:W\to W$ 有一个代数同态的标准结构，因此，通过 $(W,s_W)$ 到自身的同态类型的可收缩性，复合 $(f,s_f)$ 和 $(\mathsf{pr}_1,s_{\mathsf{pr}_1})$ 必然存在一个等式证明，并且 $(1_W,s_{1_W})$。这特别意味着存在一个等式证明 $p:\mathsf{pr}_1\circ f=1_W$。
由于 $(\mathsf{pr}_2\circ f)w:C((\mathsf{pr}_1\circ f)w)$，我们可以定义

$$\mathsf{rec}(w,c) :\equiv p_*((\mathsf{pr}_2\circ f)w):C(w)$$

其中传输 $p_*$ 是相对于家庭

$$\lambda u.\,C\circ u:(W\to W)\to W\to\mathcal{U}$$

验证命题 W-计算规则是一个计算，涉及形式 $p_*$ 的操作的自然性。 □

最后，正如所愿，我们可以将同伦自然数编码为同伦 W-类型：

Theorem 5.5.6. *可以从 W-类型的命题计算规则中推导出带有命题计算规则的自然数规则。*

## 5.6 **归纳定义的一般语法** (The general syntax of inductive definitions)

到目前为止，我们仅讨论了一些特定的归纳类型：**0**、**1**、**2**、$\mathbb{N}$、并积 (coproducts)、积 (products)、$\Sigma$ 类型、$\mathsf{W}$ 类型等。然而，类型论的一个重要方面是能够定义*新的*归纳类型，而不仅仅局限于某些特定的固定列表。然而，为了能够做到这一点，我们需要知道哪些"归纳定义"是有效或合理的。

为了说明不是所有"看起来像归纳定义"的东西都有意义，考虑以下类型 $C$ 的"构造函数"：

- $g : (C \to \mathbb{N}) \to C$。

这种类型 $C$ 的递归原则 (recursion principle) 应该说，给定一个类型 $P$，为了构造一个函数 $f : C \to P$，只需考虑当输入 $c : C$ 具有形式 $g(\alpha)$（对于某个 $\alpha : C \to \mathbb{N}$）的情况。此外，我们希望能够以某种方式使用 $f$ 应用于 $\alpha$ 的"递归数据"。然而，目前尚不清楚如何将 $f$ 应用于 $\alpha$，因为它们都是以 $C$ 为定义域的函数。

我们可以通过假设（不合理地）存在某种方法将 $f$ 应用于 $\alpha$ 并得到一个 $P \to \mathbb{N}$ 的函数，写下 $C$ 的"递归原则"。然后，递归规则的输入会要求一个类型 $P$，以及一个函数

$$h : (C \to \mathbb{N}) \to (P \to \mathbb{N}) \to P \tag{5.6.1}$$

其中 $h$ 的两个参数是 $\alpha$ 和"应用 $f$ 于 $\alpha$ 的结果"。然而，所得函数 $f : C \to P$ 的计算规则会是什么？观察其他计算规则，我们可能期望类似"$f(g(\alpha)) \equiv h(\alpha, f(\alpha))$"的结果，其中 $\alpha : C \to \mathbb{N}$，但如我们所见，"$f(\alpha)$"是没有意义的。$C$ 的归纳原则（induction principle）甚至更成问题；如何写出假设都不清楚。

另一方面，我们可以通过忽略 $C$ 在 $\alpha$ 定义域中的"递归"存在，将其视为仅仅是一个自然数族的索引类型，写出 $C$ 的另一个"递归原则"。在这种情况下，输入将要求一个类型 $P$ 以及一个函数

$$h : (C \to \mathbb{N}) \to P$$

所以递归原则的类型将是 $\mathsf{rec}_C : \prod_{(P:\mathcal{U})}((C \to \mathbb{N}) \to P) \to C \to P$，类似地对于归纳原则。现在，可以写出计算规则，即 $\mathsf{rec}_C(P, h, g(\alpha)) \equiv h(\alpha)$。然而，具有这种递归器和计算规则的类型 $C$ 的存在结果是不一致的。参见 Exercises 5.7 to 5.10 的证明和其他变式。

这个例子表明了归纳定义的一个限制：所有构造函数的定义域必须是类型被定义的**协变函子** (covariant functors)，以便我们可以"将 $f$ 应用于它们"来得到"递归调用"的结果。换句话说，如果我们将正在定义的类型的所有出现替换为一个变量 $X : \mathcal{U}$，那么每个构造函数的定义域必须是可以作为 $X$ 的协变函子的表达式。这对于我们迄今为止讨论的所有例子都是如此。例如，对于构造函数 $\mathsf{inl} : A \to A + B$，相关的函子恒等于 $A$（即 $X \mapsto A$），而对于构造函数 $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$，函子是恒等函子 $(X \mapsto X)$。

然而，这个必要条件仍然不充分。协变性防止了归纳类型出现在其构造函数定义域中的单一函数类型的左边，如上面"构造函数"$g$ 中的参数 $C \to \mathbb{N}$，因为这会产生一个逆变 (contravariant) 函子，而不是协变的。然而，由于两个逆变函子的复合是协变的，诸如 $((X \to \mathbb{N}) \to \mathbb{N})$ 的双重函数类型又变得协变。这使我们能够重现康托尔式的悖论。

例如，考虑一个"归纳类型"$D$，其构造函数如下：

- $k : ((D \to \mathsf{Prop}) \to \mathsf{Prop}) \to D$。

假设存在这样的类型，我们定义函数

$$\begin{aligned} r &: D \to (D \to \mathsf{Prop}) \to \mathsf{Prop} \\ f &: (D \to \mathsf{Prop}) \to D \\ p &: (D \to \mathsf{Prop}) \to (D \to \mathsf{Prop}) \to \mathsf{Prop} \end{aligned}$$

由

$$\begin{aligned} r(k(\theta)) &:\equiv \theta \\ f(\delta) &:\equiv k(\lambda x.\, (x = \delta)) \\ p(\delta) &:\equiv \lambda x.\, \delta(f(x)) \end{aligned}$$

这里 $r$ 是通过 $D$ 的递归原则定义的，而 $f$ 和 $p$ 是显式定义的。然后，对于任意 $\delta : D \to \mathsf{Prop}$，我们有 $r(f(\delta)) = \lambda x.\,(x = \delta)$。

特别地，因此如果 $f(\delta) = f(\delta')$，那么我们有一个路径 $s : (\lambda x.\,(x = \delta)) = (\lambda x.\,(x = \delta'))$。于是，$\mathsf{happly}(s, \delta) : (\delta = \delta) = (\delta = \delta')$，因此特别地 $\delta = \delta'$ 成立。因此，$f$ 是"单射"的（尽管 a priori $D$ 可能不是集合）。这听起来已经很可疑了——我们将"幂集"的"注入"到 $D$ 中——并且通过一些额外的工作我们可以将其转化为一个矛盾。

假设给定 $\theta : (D \to \mathsf{Prop}) \to \mathsf{Prop}$，并通过

$$\delta(d) :\equiv \exists(\gamma : D \to \mathsf{Prop}).\,(f(\gamma) = d) \times \theta(\gamma) \tag{5.6.2}$$

定义函数 $\delta : D \to \mathsf{Prop}$。我们声称 $p(\delta) = \theta$。通过函数外延性 (function extensionality)，只需证明对于任意 $\gamma : D \to \mathsf{Prop}$，$p(\delta)(\gamma) =_{\mathsf{Prop}} \theta(\gamma)$。通过一致性，我们只需证明每个都能推出另一个。现在根据 $p$ 的定义，我们有

$$\begin{aligned} p(\delta)(\gamma) &\equiv \delta(f(\gamma)) \\ &\equiv \exists(\gamma' : D \to \mathsf{Prop}).\,(f(\gamma') = f(\gamma)) \times \theta(\gamma') \end{aligned}$$

显然，如果 $\theta(\gamma)$ 成立，则这成立，因为我们可以取 $\gamma' :\equiv \gamma$。另一方面，如果我们有 $\gamma'$ 使得 $f(\gamma') = f(\gamma)$ 并且 $\theta(\gamma')$ 成立，则 $\gamma' = \gamma$ 因为 $f$ 是单射的，因此也 $\theta(\gamma)$ 成立。

这完成了证明 $p(\delta) = \theta$。因此，每个元素 $\theta : (D \to \mathsf{Prop}) \to \mathsf{Prop}$ 都是某个元素 $\delta : D \to \mathsf{Prop}$ 的 $p$ 的像。然而，如果我们通过经典的对角线化定义 $\theta$：

$$\theta(\gamma) :\equiv \neg p(\gamma)(\gamma) \quad \text{对于所有 } \gamma : D \to \mathsf{Prop}$$

那么从 $\theta = p(\delta)$ 我们推导出 $p(\delta)(\delta) = \neg p(\delta)(\delta)$。这是一个矛盾：没有命题可以与其否定相等。（假设 $P \Leftrightarrow \neg P$，如果 $P$，那么 $\neg P$，因此 $\mathbf{0}$；于是 $\neg P$，但然后 $P$，因此 $\mathbf{0}$。）

Remark 5.6.3. 这里需要解决一个宇宙大小的问题。一般来说，一个归纳类型必须存在于一个已经包含其定义中所有类型的宇宙中。因此，如果在 $D$ 的定义中，模糊的记号 $\mathsf{Prop}$ 表示 $\mathsf{Prop}_{\mathcal{U}}$，那么我们没有 $D : \mathcal{U}$ 但只有 $D : \mathcal{U}'$ 对于某个更大的宇宙 $\mathcal{U}'$ 使得 $\mathcal{U} : \mathcal{U}'$。在一个前视的理论中，因此 (5.6.2) 的右侧存在于 $\mathsf{Prop}_{\mathcal{U}'}$ 中，而不是 $\mathsf{Prop}_{\mathcal{U}}$。因此，这一矛盾确实需要命题调整公理 (propositional resizing axiom) 在 §3.5 中提到。

这个反例表明我们应该禁止一个归纳类型出现在其构造函数定义域的箭头左边，即使该出现嵌套在其他箭头中，以至于最终成为协变的。这种限制称为**严格正性** (strict positivity)（普通的"正性"本质上是协变性），事实证明这是足够的。

因此，总结而言，一个有效的归纳类型 $W$ 的定义由一个*构造函数*列表组成。每个构造函数都被分配一个类型，该类型是一个函数类型，接受一定数量（可能为零）的输入（可能相互依赖）并返回 $W$ 的元素。最后，我们允许 $W$ 本身出现在其构造函数的输入类型中，但必须严格为正性。这本质上意味着构造函数的每个参数要么是一个不涉及 $W$ 的类型，要么是某种迭代函数类型，其结果类型为 $W$。例如，以下是一个有效的构造函数类型：

$$c : (A \to W) \to (B \to C \to W) \to D \to W \to W \tag{5.6.4}$$

所有这些函数类型也可以是依赖函数 ($\Pi$-types)。[1]

请注意，我们要求归纳定义由一个*有限*的构造函数列表给出。这仅仅是因为我们必须将其写在页面上。如果我们想要一个表现得像是具有无限构造函数的归纳类型，我们可以简单地通过某个无限类型来参数化一个构造函数。例如，类似 $\mathbb{N} \to W \to W$ 的构造函数可以被视为形式上等价于可数多个形如 $W \to W$ 的构造函数。当然，无限性现在是*内部*的，但这对于任何基础系统都是应有的。同样地，如果我们想要一个构造函数接受"无限多个参数"，我们可以允许它接受由某个无限类型参数化的参数族，例如 $(\mathbb{N} \to W) \to W$ 接受 $W$ 元素的无限序列。

[1] 在 §5.4 中的语言中，严格正性确保了相关的自函子是多项式函子。众所周知，在范畴论中，并非所有自函子都可以具有初始代数；限制为多项式函子确保了一致性。我们可以考虑这种条件的各种放松，但在本书中我们将自己限制在这里定义的严格正性。

现在，一旦我们有了这样的归纳定义，我们可以用它做什么呢？首先，有一个**递归原则**，它表明为了定义一个函数 $f : W \to P$，只需考虑当输入 $w : W$ 来源于构造函数之一的情况，并允许我们递归调用该构造函数的输入上的 $f$。对于示例构造函数 (5.6.4)，我们需要 $P$ 具有类型的函数

$$d : (A \to W) \to (A \to P) \to (B \to C \to W) \to (B \to C \to P) \to D \to W \to P \to P \tag{5.6.5}$$

在这些假设下，递归原则产生 $f : W \to P$，而且“显然保留构造函数数据”——这就是计算规则，其中我们使用输入的协变性。例如，在示例 (5.6.4) 中，计算规则表示对于任意 $\alpha : A \to W$、$\beta : B \to C \to W$、$\delta : D$ 和 $\omega : W$，我们有

$$f(c(\alpha, \beta, \delta, \omega)) \equiv d(\alpha, f \circ \alpha, \beta, f \circ \beta, \delta, \omega, f(\omega)) \tag{5.6.6}$$

一般归纳类型 $W$ 的**归纳原则**仅稍微复杂一些。当然，我们从一个类型族 $P : W \to \mathcal{U}$ 开始，我们要求该类型族具备构造函数数据，“覆盖”$W$ 的构造函数数据。这意味着递归调用参数如 $A \to P$ 必须被替换为依赖函数，其类型如 $\prod_{(a:A)} P(\alpha(a))$。在 (5.6.4) 的完整示例中，归纳原则的对应假设要求

$$d : \prod_{\alpha:A\to W} \left( \prod_{a:A} P(\alpha(a)) \right) \to \prod_{\beta:B\to C\to W} \left( \prod_{(b:B)} \prod_{(c:C)} P(\beta(b,c)) \right) \to \prod_{(\delta:D)} \prod_{(\omega:W)} P(\omega) \to P(c(\alpha,\beta,\delta,\omega)) \tag{5.6.7}$$

对应的计算规则看起来与 (5.6.6) 相同。当然，递归原则是归纳原则的特例，其中 $P$ 是常数族。如我们之前提到的，归纳原则也被称为**消元器** (eliminator)，递归原则被称为**非依赖消元器** (non-dependent eliminator)。

如 §1.10 中所讨论的，我们还允许自己隐式调用归纳和递归原则，为归纳原则的每个假设表达式写一个带有 $:\equiv$ 的定义方程。这称为通过（依赖）**模式匹配** (pattern matching) 给出定义。在我们的示例中，这意味着我们可以定义 $f : \prod_{(w:W)} P(w)$ 通过

$$f(c(\alpha, \beta, \delta, \omega)) :\equiv \cdots$$

其中 $\alpha : A \to W$ 和 $\beta : B \to C \to W$ 和 $\delta : D$ 和 $\omega : W$ 是变量绑定在右侧。此外，右侧可能涉及形式为 $f(\alpha(a))$、$f(\beta(b,c))$ 和 $f(\omega)$ 的递归调用。当这个定义重新打包为归纳原则时，我们将这些递归调用替换为新变量 $\bar{\alpha}(a)$、$\bar{\beta}(b,c)$ 和 $\bar{\omega}$，其类型分别为

$$\bar{\alpha} : \prod_{a:A} P(\alpha(a))$$
$$\bar{\beta} : \prod_{(b:B)} \prod_{(c:C)} P(\beta(b,c))$$
$$\bar{\omega} : P(\omega)$$

然后我们可以写为

$$f :\equiv \mathsf{ind}_W(P,\ \lambda\alpha.\, \lambda\bar{\alpha}.\, \lambda\beta.\, \lambda\bar{\beta}.\, \lambda\delta.\, \lambda\omega.\, \lambda\bar{\omega}.\, \cdots)$$

其中 $\mathsf{ind}_W$ 的第二个参数具有 (5.6.7) 的类型。

我们不会尝试给出一个有效归纳定义的语法及其结果归纳和递归原则及模式匹配规则的正式表述。尽管可以做到（实际上，如果实现一个计算机证明助理，这是必要的），但并没有提供额外的见解。通过练习，我们可以自动推导出任何归纳定义的归纳和递归原则，并在不加思考的情况下使用它们。

## 5.7 归纳类型的推广 (Generalizations of inductive types)

归纳类型的概念在类型论中已经研究多年，并且承认了许多，许多推广：归纳类型族、互递归归纳类型、归纳-归纳类型、归纳-递归类型等。在本节中，我们将概述其中的一些，书中稍后将使用的少数几种。（在 Chapter 6 中，我们将更深入地研究一种特定于同伦类型论的归纳类型的不同推广。）

这些推广中的大多数涉及允许我们同时定义多个类型。我们已经看到的一个非常简单的例子是并积 $A + B$。如果我们每次想考虑两个类型的并积时都要分别写下 $\mathbb{N} + \mathbb{N}$、$\mathbb{N} + \mathbf{2}$、$\mathbf{2} + \mathbf{2}$ 等等的归纳定义，这将是非常乏味的。相反，我们进行一个定义，其中 $A$ 和 $B$ 是代表类型的变量；在类型论中，它们被称为**参数**。因此，从技术上讲，定义的结果不是单一类型，而是一个类型族 $+ : \mathcal{U} \to \mathcal{U} \to \mathcal{U}$，接受两个类型作为输入并产生它们的并积。类似地，列表类型 $\mathsf{List}(A)$ 是一个类型族 $\mathsf{List}(-) : \mathcal{U} \to \mathcal{U}$，其中类型 $A$ 是参数。

在数学中，这种事情显而易见，以至于不值得一提，但我们提到这一点是为了与下一个例子形成对比。请注意，每个类型 $A + B$ 是*独立地*定义的归纳类型，如同每个类型 $\mathsf{List}(A)$。相比之下，我们也可以考虑定义一个整个类型族 $B : A \to \mathcal{U}$，*同时*进行归纳。不同之处在于，现在构造函数可以更改索引 $a : A$，因此我们不能说个别类型 $B(a)$ 是归纳定义的，而只能说整个类型族是归纳定义的。

标准的例子是*指定长度的列表类型*，通常称为**向量** (vectors)。我们固定一个参数类型 $A$，并定义一个类型族 $\mathsf{Vec}_n(A)$，其中 $n : \mathbb{N}$，其构造函数如下：

- 一个长度为零的向量 $\mathsf{nil} : \mathsf{Vec}_0(A)$，
- 一个函数 $\mathsf{cons} : \prod_{(n:\mathbb{N})} A \to \mathsf{Vec}_n(A) \to \mathsf{Vec}_{\mathsf{succ}(n)}(A)$。

与列表相比，向量（固定类型 $A$ 的元素）形成一个以其长度为索引的类型族。虽然 $A$ 是一个参数，但我们说 $n : \mathbb{N}$ 是归纳族的**索引**。一个单独的类型，如 $\mathsf{Vec}_3(A)$，不是归纳定义的：构造 $\mathsf{Vec}_3(A)$ 元素的构造函数来自于家族中的不同类型，如 $\mathsf{cons} : A \to \mathsf{Vec}_2(A) \to \mathsf{Vec}_3(A)$。

特别是，归纳原则必须参考整个类型族；因此，假设和结论必须适当地对索引进行量化。在向量的情况下，归纳原则声明，给定一个类型族 $C : \prod_{(n:\mathbb{N})} \mathsf{Vec}_n(A) \to \mathcal{U}$，以及

- 一个元素 $c_{\mathsf{nil}} : C(0, \mathsf{nil})$，和
- 一个函数 $c_{\mathsf{cons}} : \prod_{(n:\mathbb{N})} \prod_{(a:A)} \prod_{(\ell:\mathsf{Vec}_n(A))} C(n, \ell) \to C(\mathsf{succ}(n), \mathsf{cons}(a, \ell))$

存在一个函数 $f : \prod_{(n:\mathbb{N})} \prod_{(\ell:\mathsf{Vec}_n(A))} C(n, \ell)$，使得

$$f(0, \mathsf{nil}) \equiv c_{\mathsf{nil}}$$
$$f(\mathsf{succ}(n), \mathsf{cons}(a, \ell)) \equiv c_{\mathsf{cons}}(n, a, \ell, f(\ell))$$

归纳族的一个用途是*归纳定义谓词*。例如，我们可以定义谓词 $\mathsf{iseven} : \mathbb{N} \to \mathcal{U}$ 作为归纳族，索引为 $\mathbb{N}$，其构造函数如下：

- 一个元素 $\mathsf{even}_0 : \mathsf{iseven}(0)$，
- 一个函数 $\mathsf{even}_{ss} : \prod_{(n:\mathbb{N})} \mathsf{iseven}(n) \to \mathsf{iseven}(\mathsf{succ}(\mathsf{succ}(n)))$。

换句话说，我们规定 0 是偶数，并且如果 $n$ 是偶数，那么 $\mathsf{succ}(\mathsf{succ}(n))$ 也是偶数。这些构造函数显然不会构造出 $\mathsf{iseven}(1)$ 的元素，并且由于 iseven 应该由这些构造函数自由生成，因此不应有这样的元素。（然而，实际证明 $\neg\mathsf{iseven}(1)$ 并非完全平凡）。iseven 的归纳原则表明，为了证明关于所有偶数自然数的某些性质，只需证明它对 0 成立并验证其在加二时保持不变。

归纳定义的谓词在计算机形式化数学和软件验证中得到广泛使用。但在本书中，我们不会有太多的使用，除了 §§10.3 and 11.5 中的几个例外。

另一个重要的特例是当归纳族的索引类型是有限的。在这种情况下，我们可以等价地将归纳定义表达为通过*相互归纳* (mutual induction) 定义的一组有限类型。例如，我们可以通过相互归纳定义 even 和 odd 类型，其中 even 由构造函数生成：

- $0 : \mathsf{even}$，和
- $\mathsf{esucc} : \mathsf{odd} \to \mathsf{even}$，

而 odd 由一个构造函数生成

- $\mathsf{osucc} : \mathsf{even} \to \mathsf{odd}$。

请注意，even 和 odd 是简单类型（不是类型族），但它们的构造函数可以相互引用。如果我们将此定义表示为一个归纳类型族 $\mathsf{paritynat} : \mathbf{2} \to \mathcal{U}$，其中 $\mathsf{paritynat}(0_{\mathbf{2}})$ 和 $\mathsf{paritynat}(1_{\mathbf{2}})$ 分别表示 even 和 odd，那么它将具有以下构造函数：

- $0 : \mathsf{paritynat}(0_{\mathbf{2}})$，
- $\mathsf{esucc} : \mathsf{paritynat}(1_{\mathbf{2}}) \to \mathsf{paritynat}(0_{\mathbf{2}})$，
- $\mathsf{osucc} : \mathsf{paritynat}(0_{\mathbf{2}}) \to \mathsf{paritynat}(1_{\mathbf{2}})$。

当明确表示为相互归纳定义时，even 和 odd 的归纳原则表明，给定 $C : \mathsf{even} \to \mathcal{U}$ 和 $D : \mathsf{odd} \to \mathcal{U}$，以及

- $c_0 : C(0)$，
- $c_s : \prod_{(n:\mathsf{odd})} D(n) \to C(\mathsf{esucc}(n))$，
- $d_s : \prod_{(n:\mathsf{even})} C(n) \to D(\mathsf{osucc}(n))$，

存在 $f : \prod_{(n:\mathsf{even})} C(n)$ 和 $g : \prod_{(n:\mathsf{odd})} D(n)$，使得

$$\begin{aligned} f(0) &\equiv c_0 \\ f(\mathsf{esucc}(n)) &\equiv c_s(g(n)) \\ g(\mathsf{osucc}(n)) &\equiv d_s(f(n)) \end{aligned}$$

特别是，正如我们只能“同时”对归纳族进行归纳一样，我们必须同时对 even 和 odd 进行归纳。在本书中，我们也不会有太多对相互归纳定义的使用。

进一步的、更激进的推广是允许定义一个类型族 $B : A \to \mathcal{U}$，其中不仅 $B(a)$ 的类型被定义，类型 $A$ 本身也是作为一个整体通过归纳定义的。换句话说，我们不仅为 $B(a)$ 指定了构造函数，也同时为 $A$ 本身指定了构造函数，后者可以引用 $B(a)$ 的元素。这可以看作是一个归纳族，其中索引与被索引的类型同时被归纳定义，或者作为一个类型依赖于另一个类型的相互归纳定义。更复杂的依赖结构也是可能的。一般来说，这些称为**归纳-归纳定义** (inductive-inductive definitions)。在本书中，我们不会使用它们，但其更高的变体（见 Chapter 6）将在 Chapter 11 的几个实验性示例中出现。

最后一个要提到的推广是**归纳-递归定义** (inductive-recursive definitions)，其中一个类型和一个*递归*函数同时被定义。也就是说，我们固定一个已知类型 $P$，并给出一个归纳类型 $A$ 的构造函数，同时使用 $A$ 的递归原则定义一个函数 $f : A \to P$——其中的转折点是，$A$ 的构造函数也被允许引用 $f$ 的值。我们尚不知道如何从同伦的角度来证明这种定义的合理性，因此在本书中我们不会使用它们。

## 5.8 同一性类型与同一性系统 (Identity types and identity systems)

我们现在希望指出，在同伦类型论中起着核心作用的*同一性类型* (identity types) 也可以被认为是归纳地定义的。具体来说，它们是具有索引的“归纳族”，如 §5.7 中所述。事实上，有*两种*方式将同一性类型描述为归纳族，导致了 Chapter 1 中描述的两种归纳原则，即路径归纳 (path induction) 和基于路径的归纳 (based path induction)。

在这两种定义中，类型 $A$ 是一个参数。在第一个定义中，我们通过以下构造函数归纳地定义了一个族 $=_A : A \to A \to \mathcal{U}$，其两个索引属于 $A$：

- 对于任何 $a : A$，有一个元素 $\mathsf{refl}_a : a =_A a$。

通过类比其他归纳族，我们可以从这个定义中提取出归纳原则。它声明给定任何 $C : \prod_{(a,b:A)}(a =_A b) \to \mathcal{U}$，以及 $d : \prod_{(a:A)} C(a, a, \mathsf{refl}_a)$，存在 $f : \prod_{(a,b:A)} \prod_{(p:a=_A b)} C(a, b, p)$ 使得 $f(a, a, \mathsf{refl}_a) \equiv d(a)$。这正是同一性类型的路径归纳原则。

对于第二个定义，我们考虑一个元素 $a_0 : A$ 作为参数，以及 $A : \mathcal{U}$，并通过以下构造函数归纳地定义了一个族 $(a_0 =_A -) : A \to \mathcal{U}$，其一个索引属于 $A$：

- 一个元素 $\mathsf{refl}_{a_0} : a_0 =_A a_0$。

注意，由于 $a_0 : A$ 被固定为一个参数，构造函数 $\mathsf{refl}_{a_0}$ 并没有作为一个函数出现在归纳定义中，而只是作为一个元素出现。这个定义的归纳原则表明，给定 $C : \prod_{(b:A)}(a_0 =_A b) \to \mathcal{U}$ 以及一个元素 $d : C(a_0, \mathsf{refl}_{a_0})$，存在 $f : \prod_{(b:A)} \prod_{(p:a_0=_A b)} C(b,p)$ 使得 $f(a_0, \mathsf{refl}_{a_0}) \equiv d$。这正是同一性类型的基于路径的归纳原则。

将同一性类型视为归纳类型的观点历史上曾引起一些混淆，因为如 §5.1 中提到的直觉认为，归纳类型的所有元素都应该通过反复应用其构造函数获得。对于普通的归纳类型，如 $\mathbf{2}$ 和 $\mathbb{N}$，确实如此：我们在 Eq. (1.8.1) 中看到，$\mathbf{2}$ 的每个元素要么是 $0_{\mathbf{2}}$，要么是 $1_{\mathbf{2}}$，同样可以证明 $\mathbb{N}$ 的每个元素要么是 0，要么是继承者 (successor)。

然而，这对于同一性类型来说并不是真的：虽然只有一个构造函数 $\mathsf{refl}$，但并非每个路径都等同于常量路径。更确切地说，我们不能仅通过使用同一性类型的归纳原则（无论是哪一种）来证明 $a =_A a$ 的每个居住者都等于 $\mathsf{refl}_a$。为了真正展示一个反例，我们需要一些额外的原则，比如同值性公理 (univalence axiom)——回想一下在 Example 3.1.9 中，我们使用同值性展示了一个特定的路径 $\mathbf{2} =_{\mathcal{U}} \mathbf{2}$，它并不等于 $\mathsf{refl}_{\mathbf{2}}$。

关键是，正如同伦初始代数的研究所验证的那样，一个归纳定义应该被视为由其构造函数**自由生成**的。当然，自由生成的结构可能包含其生成器之外的元素：例如，由两个符号 $x$ 和 $y$ 自由生成的自由群 (free group) 不仅包含 $x$ 和 $y$，还包含诸如 $xy$、$yx^{-1}y$ 和 $x^3y^2x^{-2}yx$ 之类的词语。一般来说，自由结构的元素是通过应用不仅是生成器，还有环境结构的操作（如果我们谈论自由群，则是群操作）获得的。

在归纳类型的情况下，我们谈论的是自由生成的**类型**——那么，类型结构的"操作"是什么呢？如果类型被视为类似于**集合**，正如传统上在类型论中所做的那样，那么就没有这样的操作，因此我们期望在归纳类型中没有其他元素，除了由其构造函数生成的那些元素。在同伦类型论中，我们将类型视为类似于**空间**或 $\infty$-群体 (spaces or $\infty$-groupoids)，在这种情况下，有许多关于**路径**的操作（如连接、逆等）——这将在 Chapter 6 中变得重要——但在**对象**（元素）上仍然没有操作。因此，对于我们来说，例如，$\mathbf{2}$ 的每个元素要么是 $0_{\mathbf{2}}$，要么是 $1_{\mathbf{2}}$，而 $\mathbb{N}$ 的每个元素要么是 0，要么是继承者，这仍然是正确的。

然而，正如我们在 Chapter 2 中看到的，将类型视为 $\infty$-群体也意味着将函数视为函子，这包括类型族 $B : A \to \mathcal{U}$。因此，视为归纳类型族的同一性类型 $(a_0 =_A -)$ 实际上是一个**自由生成的函子** $A \to \mathcal{U}$。具体来说，它是由一个元素 $\mathsf{refl}_{a_0} : F(a_0)$ 自由生成的函子 $F : A \to \mathcal{U}$。一个函子确实对对象有操作，即 $A$ 的态射（路径）的作用。

在范畴论中，**约内达引理** (Yoneda lemma) 告诉我们，对于任何范畴 $A$ 和对象 $a_0$，由 $F(a_0)$ 的一个元素自由生成的函子是可表函子 (representable functor) $\mathsf{hom}_A(a_0, -)$。因此，我们应该期望同一性类型 $(a_0 =_A -)$ 是这个可表函子，这实际上正是我们看待它的方式：$(a_0 =_A b)$ 是从 $a_0$ 到 $b$ 的 $A$ 中的态射（路径）的空间。

将同一性类型视为归纳族的一个原因是应用 §§5.2 and 5.5 中的唯一性原则。具体来说，我们可以通过给出另一个在 $A \times A$ 上满足相同归纳原则的类型族，来刻画同一性类型 $A$ 的家族（up to equivalence）。这表明了以下定义和定理。

Definition 5.8.1. 设 $A$ 是一个类型，$a_0 : A$ 是一个元素。

- 一个 **带点谓词** (pointed predicate) 在 $(A, a_0)$ 上是一个带有元素 $r_0 : R(a_0)$ 的族 $R : A \to \mathcal{U}$。
- 对于带点谓词 $(R, r_0)$ 和 $(S, s_0)$，如果映射族 $g : \prod_{(b:A)} R(b) \to S(b)$ 满足 $g(a_0, r_0) = s_0$，则称其为**带点的** (pointed)。我们有

$$\mathsf{ppmap}(R,S) :\equiv \sum_{g:\prod_{(b:A)} R(b)\to S(b)} (g(a_0, r_0) = s_0)$$

- 在 $a_0$ 处的**同一性系统** (identity system) 是一个带点谓词 $(R, r_0)$，满足对于任何类型族 $D : \prod_{(b:A)} R(b) \to \mathcal{U}$ 和 $d : D(a_0, r_0)$，存在函数 $f : \prod_{(b:A)} \prod_{(r:R(b))} D(b, r)$ 使得 $f(a_0, r_0) = d$。

Theorem 5.8.2. *对于 $(A, a_0)$ 上的带点谓词 $(R, r_0)$，以下条件逻辑等价：*

(i) *$(R, r_0)$ 在 $a_0$ 处是一个同一性系统。*

(ii) 对于任何带点谓词 $(S, s_0)$，类型 $\mathsf{ppmap}(R,S)$ 是收缩的。

(iii) 对于任何 $b : A$，映射 $\mathsf{transport}^R(-, r_0) : (a_0 =_A b) \to R(b)$ 是等价的。

(iv) 类型 $\sum_{(b:A)} R(b)$ 是收缩的。

注意，等价 (i)⇔(ii)⇔(iii) 是同一性类型 $a_0 =_A -$ 作为一个在 $A$ 的一个元素上变化的归纳族的 Lemma 5.5.4 的版本。当然，(ii)–(iv) 是单纯命题，因此逻辑等价意味着实际等价。（条件 (i) 也是单纯命题，但我们不会证明这一点。）还要注意，与 (i)–(iii) 不同，陈述 (iv) 没有提到 $a_0$ 或 $r_0$。

证明. 首先，假设 (i) 并设 $(S, s_0)$ 是一个带点谓词。定义 $D(b,r) :\equiv S(b)$ 并设 $d :\equiv s_0 : S(a_0) \equiv D(a_0, r_0)$。由于 $R$ 是一个同一性系统，我们有 $f : \prod_{(b:A)} R(b) \to S(b)$ 使得 $f(a_0, r_0) = s_0$；因此 $\mathsf{ppmap}(R,S)$ 是居住的。现在假设 $(f, f_r), (g, g_r) : \mathsf{ppmap}(R,S)$，并定义 $D(b,r) :\equiv (f(b,r) = g(b,r))$，并设 $d :\equiv f_r \cdot g_r^{-1} : f(a_0, r_0) = s_0 = g(a_0, r_0)$。然后，由于 $R$ 是一个同一性系统，我们有 $h : \prod_{(b:A)} \prod_{(r:R(b))} D(b,r)$ 使得 $h(a_0, r_0) = f_r \cdot g_r^{-1}$。通过函数外延性和在 $\Sigma$-类型和路径类型中的路径的特性，这些数据产生了一个等式 $(f, f_r) = (g, g_r)$。因此 $\mathsf{ppmap}(R,S)$ 是一个居住的单纯命题，因此是收缩的；所以 (ii) 成立。

现在假设 (ii)，并定义 $S(b) :\equiv (a_0 = b), s_0 :\equiv \mathsf{refl}_{a_0} : S(a_0)$。然后 $(S, s_0)$ 是一个带点谓词，$\lambda b.\, \lambda p.\, \mathsf{transport}^R(p, r) : \prod_{(b:A)} S(b) \to R(b)$ 是一个从 $S$ 到 $R$ 的带点映射族。根据假设，$\mathsf{ppmap}(R,S)$ 是收缩的，因此是居住的，因此也存在从 $R$ 到 $S$ 的带点映射族。并且在任何一个方向上的复合都是从 $R$ 到 $R$ 和从 $S$ 到 $S$ 的带点映射族，因此等同于身份，因为 $\mathsf{ppmap}(R,R)$ 和 $\mathsf{ppmap}(S,S)$ 是收缩的。因此 (iii) 成立。

现在假设 (iii)，条件 (iv) 可由 Lemma 3.11.8 得出，使用的是 $\Sigma$-类型尊重等价的事实（Theorem 4.7.7 的“如果”方向）。

最后，假设 (iv)，并设 $D : \prod_{(b:A)} R(b) \to \mathcal{U}$ 和 $d : D(a_0, r_0)$。我们可以等价地将 $D$ 表示为一个族 $D' : (\sum_{(b:A)} R(b)) \to \mathcal{U}$。现在，由于 $\sum_{(b:A)} R(b)$ 是收缩的，我们有

$$p : \prod_{u:\sum_{(b:A)} R(b)} (a_0, r_0) = u.$$

此外，由于收缩类型的路径类型再次收缩，我们有 $p((a_0, r_0)) = \mathsf{refl}_{(a_0,r_0)}$。定义 $f(u) :\equiv \mathsf{transport}^{D'}(p(u), d)$，得到 $f : \prod_{(u:\sum_{(b:A)} R(b))} D'(u)$，或等价地 $f : \prod_{(b:A)} \prod_{(r:R(b))} D(b,r)$。最后，我们有

$$f(a_0, r_0) \equiv \mathsf{transport}^{D'}(p((a_0, r_0)), d) = \mathsf{transport}^{D'}(\mathsf{refl}_{(a_0,r_0)}, d) = d$$

因此，(i) 成立。 □

我们可以推导出类似的结果，用于同一性类型 $=_A$，将其视为一个在 $A$ 的两个元素上变化的族。

Definition 5.8.3. 在一个类型 $A$ 上的**同一性系统**是一个族 $R : A \to A \to \mathcal{U}$，带有一个函数 $r_0 : \prod_{(a:A)} R(a,a)$，使得对于任何类型族 $D : \prod_{(a,b:A)} R(a,b) \to \mathcal{U}$ 和 $d : \prod_{(a:A)} D(a,a,r_0(a))$，存在函数 $f : \prod_{(a,b:A)} \prod_{(r:R(a,b))} D(a,b,r)$ 使得 $f(a,a,r_0(a)) = d(a)$ 对所有 $a : A$ 成立。

Theorem 5.8.4. 对于 $R : A \to A \to \mathcal{U}$ 带有 $r_0 : \prod_{(a:A)} R(a,a)$，以下条件逻辑等价：

(i) $(R, r_0)$ 在 $A$ 上是一个同一性系统。

(ii) 对于所有 $a_0 : A$，带点谓词 $(R(a_0), r_0(a_0))$ 是在 $a_0$ 处的同一性系统。

(iii) 对于任何 $S : A \to A \to \mathcal{U}$ 和 $s_0 : \prod_{(a:A)} S(a,a)$，类型

$$\sum_{(g:\prod_{(a,b:A)} R(a,b) \to S(a,b))} \prod_{(a:A)} g(a,a,r_0(a)) = s_0(a)$$

是收缩的。

(iv) 对于任何 $a, b : A$，映射 $\mathsf{transport}^{R(a)}(-, r_0(a)) : (a =_A b) \to R(a,b)$ 是等价的。

(v) 对于任何 $a : A$，类型 $\sum_{(b:A)} R(a,b)$ 是收缩的。

证明. 等价 (i)⇔(ii) 完全按照同一性类型的路径归纳原则和基于路径的归纳原则的等价性证明；参见 §1.12。然后，等价于 (iv) 和 (v) 来自 Theorem 5.8.2，而 (iii) 是直接的。 □

这个特征描述之所以有趣的原因之一是它提供了一种陈述同值性和函数外延性的替代方式。对于宇宙 $\mathcal{U}$ 的同值性公理 (univalence axiom) 表示类型族

$$(- \simeq -) : \mathcal{U} \to \mathcal{U} \to \mathcal{U}$$

连同 $\mathsf{id} : \prod_{(A:\mathcal{U})}(A \simeq A)$ 满足 Theorem 5.8.4(iv)。因此，它等价于相应的 (i) 版本，我们可以将其陈述如下。

Corollary 5.8.5 (等价归纳 (Equivalence induction)). 给定任何类型族 $D : \prod_{(A,B:\mathcal{U})}(A \simeq B) \to \mathcal{U}$ 和函数 $d : \prod_{(A:\mathcal{U})} D(A, A, \mathsf{id}_A)$，存在 $f : \prod_{(A,B:\mathcal{U})} \prod_{(e:A\simeq B)} D(A, B, e)$ 使得对于所有 $A : \mathcal{U}$，$f(A, A, \mathsf{id}_A) = d(A)$。

换句话说，要证明关于所有等价的某些东西，仅需证明关于身份映射的某些东西。我们已经在 Lemma 4.1.1 中使用了这一原理（没有以普遍性陈述它）。
类似地，函数外延性表明，对于任何 $B : A \to \mathcal{U}$，类型族

$$(- \sim -) : \left(\prod_{a:A} B(a)\right) \to \left(\prod_{a:A} B(a)\right) \to \mathcal{U}$$

连同 $\lambda f.\, \lambda a.\, \mathsf{refl}_{f(a)}$ 满足 Theorem 5.8.4(iv)。因此，它也等价于相应的 (i) 版本。

Corollary 5.8.6 (同伦归纳 (Homotopy induction)). 给定任何 $D : \prod_{(f,g:\prod_{(a:A)} B(a))}(f \sim g) \to \mathcal{U}$ 和 $d : \prod_{(f:\prod_{(a:A)} B(a))} D(f, f, \lambda x.\, \mathsf{refl}_{f(x)})$，存在

$$k : \prod_{(f,g:\prod_{(a:A)} B(a))} \prod_{(h:f\sim g)} D(f, g, h)$$

使得 $k(f, f, \lambda x.\, \mathsf{refl}_{f(x)}) = d(f)$ 对所有 $f$ 成立。

## Notes

归纳定义在数学中有着悠久的历史，可以追溯到弗雷格 (Frege) 和皮亚诺 (Peano) 的自然数公理。更一般的"归纳谓词"并不罕见，但在集合论基础中，它们通常是显式构造的，或者作为适当类子集的交集，或者使用沿序数的超限迭代，而不是作为基本概念。
在类型论中，归纳定义的特殊情况可以追溯到 Martin-Löf 的原始论文：[?] 提出了归纳定义的谓词和关系的一般概念；归纳类型的概念（但仅有实例，而不是作为一般概念）出现在 Martin-Löf 的第一篇类型论论文中 [?]；然后作为一个普遍概念出现在 [?] 中。
归纳类型的一般概念由 Constable 和 Mendler 于 1985 年引入 [?]。在强类型论中归纳类型的一般模式由 [?] 提出。后续发展包括 [?, ?]。
归纳-递归 (inductive-recursive) 定义的概念出现在 [?] 中。一个重要的类型论概念是树类型 (tree types) 的概念（在类型论中 Post 系统概念的通用表达），它出现在 [?] 中。
自然数的普遍性质作为 $\mathbb{N}$-代数范畴中的初始对象 (initial object) 由 Lawvere 提出 [?]。这后来被推广为 W-类型作为多项式端函子 (polynomial endofunctors) 的初始代数的描述，由 [?] 提出。此类普遍性质与相应归纳原则（§§5.4 and 5.5）之间的同伦-同调论等价由 [?] 提出。
有关在类型论的同伦-同调论语义中的归纳类型的实际构造，请参见 [?, ?, ?]。

## Exercises

Exercise 5.1. 从 §5.1 中 $\mathsf{List}(A)$ 作为归纳类型的定义中推导出类型 $\mathsf{List}(A)$ 的归纳原则。

Exercise 5.2. 构造两个在自然数上满足相同递归 (recurrence) $(e_z, e_s)$ 的函数，但它们不完全相等。

Exercise 5.3. 构造两个在相同类型 $E$ 上的不同递归 $(e_z, e_s)$，它们都在相同函数 $f : \mathbb{N} \to E$ 上判断成立。

Exercise 5.4. 证明对于任何类型族 $E : \mathbf{2} \to \mathcal{U}$，归纳算子

$$\mathsf{ind}_{\mathbf{2}}(E) : \big(E(0_{\mathbf{2}}) \times E(1_{\mathbf{2}})\big) \to \prod_{b:\mathbf{2}} E(b)$$

是一个等价。

Exercise 5.5. 证明对于 $\mathbb{N}$ 的类似陈述与 Exercise 5.4 不成立。

Exercise 5.6. 证明如果我们假设 W-类型的简单而不是依赖消去 (elimination)，则唯一性属性（Theorem 5.3.1 的类似物）不成立。也就是说，展示一个满足 W-类型递归原则的类型，但其函数不唯一地由其递归确定。

Exercise 5.7. 假设在 §5.6 开头的类型 $C$ 的"归纳定义"中，我们用 $\mathbf{0}$ 替换 $\mathbb{N}$ 。类似于 (5.6.1)，我们可能会考虑带有假设的这个类型的递归原则

$$h : (C \to \mathbf{0}) \to (P \to \mathbf{0}) \to P.$$

证明即使没有计算规则，这个递归原则也是不一致的，即它允许我们构造 $\mathbf{0}$ 的一个元素。

Exercise 5.8. 现在考虑一个具有构造函数 $\mathsf{scott} : (D \to D) \to D$ 的"归纳类型"$D$。§5.6 中建议的 $C$ 的第二递归算子导致 $D$ 的以下递归算子：

$$\mathsf{rec}_D : \prod_{P:\mathcal{U}} ((D \to D) \to (D \to P) \to P) \to D \to P$$

其计算规则为 $\mathsf{rec}_D(P, h, \mathsf{scott}(\alpha)) \equiv h(\alpha, (\lambda d.\, \mathsf{rec}_D(P, h, \alpha(d))))$。证明这也导致矛盾。

Exercise 5.9. 设 $A$ 是一个任意类型，并且一般地考虑具有构造函数 $\mathsf{lawvere} : (L_A \to A) \to L_A$ 的类型 $L_A$ 的"归纳定义"。§5.6 中建议的 $C$ 的第二递归算子导致 $L_A$ 的以下递归算子：

$$\mathsf{rec}_{L_A} : \prod_{P:\mathcal{U}} ((L_A \to A) \to P) \to L_A \to P$$

其计算规则为 $\mathsf{rec}_{L_A}(P, h, \mathsf{lawvere}(\alpha)) \equiv h(\alpha)$。使用这个证明 $A$ 具有**不动点属性** (fixed-point property)，即对于每个函数 $f : A \to A$，存在一个 $a : A$ 使得 $f(a) = a$。特别地，如果 $A$ 是一个没有不动点属性的类型（例如 $\mathbf{0}$ 、$\mathbf{2}$ 或 $\mathbb{N}$ ），则 $L_A$ 是不一致的。

Exercise 5.10. 继续 Exercise 5.9，考虑 $L_{\mathbf{1}}$，它显然没有不一致，因为 $\mathbf{1}$ 确实具有不动点属性。制定 $L_{\mathbf{1}}$ 的归纳原则及其计算规则，类似于其递归算子，并使用它证明它是收缩的。

Exercise 5.11. 在 §5.1 中我们定义了类型 $\mathsf{List}(A)$，表示类型 $A$ 的有限列表。考虑类似的归纳定义类型 $\mathsf{Lost}(A)$，其唯一构造函数为

$$\mathsf{cons} : A \to \mathsf{Lost}(A) \to \mathsf{Lost}(A).$$

证明 $\mathsf{Lost}(A)$ 等价于 $\mathbf{0}$ 。

Exercise 5.12. 假设 $A$ 是一个单纯命题，并且 $B : A \to \mathcal{U}$。

(i) 证明 $\mathsf{W}_{(a:A)} B(a)$ 是一个单纯命题。
(ii) 证明 $\mathsf{W}_{(a:A)} B(a)$ 等价于 $\sum_{(a:A)} \neg B(a)$。
(iii) 不使用 $\mathsf{W}_{(a:A)} B(a)$，证明 $\sum_{(a:A)} \neg B(a)$ 是 §5.5 意义上的同伦 W-类型 $\mathsf{W}^h_{(a:A)} B(a)$。

Exercise 5.13. 设 $A : \mathcal{U}$ 并且 $B : A \to \mathcal{U}$。

(i) 证明 $\Big(\sum_{(a:A)} \neg B(a)\Big) \to \Big(\mathsf{W}_{(a:A)} B(a)\Big)$。

(ii) 证明 $\left(\mathsf{W}_{(a:A)}B(a)\right) \to \left(\neg\prod_{(a:A)} B(a)\right)$。

Exercise 5.14. 设 $A : \mathcal{U}$ 并假设 $B : A \to \mathcal{U}$ 是可判定的，即 $\prod_{(a:A)}(B(a) + \neg B(a))$（参见 Definition 3.4.3）。证明 $\left(\mathsf{W}_{(a:A)}B(a)\right) \to \left(\sum_{(a:A)} \neg B(a)\right)$。

Exercise 5.15. 证明以下条件逻辑等价。

(i) 对于任何 $A : \mathsf{Set}$ 和 $B : A \to \mathsf{Prop}$，$\left(\mathsf{W}_{(a:A)}B(a)\right) \to \left\|\sum_{(a:A)} \neg B(a)\right\|$。

(ii) 对于任何 $A : \mathsf{Set}$ 和 $B : A \to \mathsf{Prop}$，$\left(\neg\prod_{(a:A)} B(a)\right) \to \left\|\mathsf{W}_{(a:A)}B(a)\right\|$。

(iii) 排中律（如 §3.4 中所述）。

同样，使用 Corollary 3.2.7，证明假设任何 (i) 或 (ii) 条件在所有 $A : \mathcal{U}$ 和 $B : A \to \mathcal{U}$ 上成立是不一致的。

Exercise 5.16. 对于 $A : \mathcal{U}$ 和 $B : A \to \mathcal{U}$，定义

$$W'_{A,B} :\equiv \prod_{R:\mathcal{U}} \left(\prod_{a:A} (B(a) \to R) \to R\right) \to R$$

$W'_{A,B}$ 被称为 $\mathsf{W}_{(a:A)}B(a)$ **的非限制性编码** (impredicative encoding)。注意，与 $\mathsf{W}_{(a:A)}B(a)$ 不同，它存在于高于 $A$ 和 $B$ 的宇宙中。

(i) 证明 $W'_{A,B}$ 在逻辑上等价（如 §1.11 中定义）于 $\mathsf{W}_{(a:A)}B(a)$。

(ii) 证明 $W'_{A,B}$ 意味着 $\neg\neg\sum_{(a:A)} \neg B(a)$。

(iii) 不使用 $\mathsf{W}_{(a:A)}B(a)$，证明 $W'_{A,B}$ 满足与 $\mathsf{W}_{(a:A)}B(a)$ 相同的递归原则，用于定义进入宇宙 $\mathcal{U}$ 的类型的函数（它本身不属于其中）。

(iv) 使用 $\mathsf{LEM}$ 举一个 $A : \mathcal{U}$ 和 $B : A \to \mathcal{U}$ 的例子，使得 $W'_{A,B}$ 不等同于 $\mathsf{W}_{(a:A)}B(a)$。

Exercise 5.17. 证明对于任何 $A : \mathcal{U}$ 和 $B : A \to \mathcal{U}$，我们有

$$\neg\left(\mathsf{W}_{(a:A)}B(a)\right) \simeq \neg\left(\sum_{a:A} \neg B(a)\right)$$

换句话说，$\mathsf{W}_{(a:A)}B(a)$ 是空的，当且仅当它没有空构造函数 (nullary constructor)。（比较 Exercise 5.11。）

Chapter 6

# 高阶归纳类型 (Higher Inductive Types)

## 6.1 引言 (Introduction)

与我们在 Chapter 5 中讨论的一般归纳类型类似，高阶归纳类型 (higher inductive types) 是通过一些构造器生成新类型的一般模式。但与普通归纳类型不同，在定义高阶归纳类型时，我们可以有“构造器”不仅生成该类型的点，还生成该类型中的路径和更高阶的路径。例如，我们可以考虑由以下构造器生成的高阶归纳类型 $\mathbb{S}^1$：

- 一个点 $\mathsf{base} : \mathbb{S}^1$，以及
- 一个路径 $\mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}$。

这应当与例如 **2** 的定义完全类似，**2** 是由以下构造器生成的：

- 一个点 $0_{\mathbf{2}} : \mathbf{2}$ 和
- 一个点 $1_{\mathbf{2}} : \mathbf{2}$，

或由以下构造器生成的 $\mathbb{N}$ 的定义：

- 一个点 $0 : \mathbb{N}$ 以及
- 一个函数 $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$。

当我们将类型视为高阶群体 (higher groupoids) 时，更一般的“生成”概念是非常自然的：由于高阶群体是一个具有路径和更高阶路径以及点的“多重排序对象”，我们应当允许所有维度上的“生成器”。我们将普通类型的构造器（如 $\mathsf{base}$）称为**点构造器** (point constructors)或普通构造器，而将其他构造器（如 $\mathsf{loop}$）称为**路径构造器** (path constructors)或高阶构造器。每个路径构造器必须指定路径的起点和终点，我们称其为**源点** (source)和**目标** (target)；对于 $\mathsf{loop}$，源点和目标均为 $\mathsf{base}$。

请注意，路径构造器如 $\mathsf{loop}$ 生成的是同一性类型 (identity type) 的一个新元素，它（至少在先验上）不等于任何先前存在的此类元素。特别地，$\mathsf{loop}$ 先验上不等于 $\mathsf{refl}_{\mathsf{base}}$（尽管证明它们确实不相等需要一些思考；见 Lemma 6.4.1）。这就是 $\mathbb{S}^1$ 与普通归纳类型 **1** 的区别。

关于这种推广，有几个重要点需要说明。

首先，应认真对待“生成”一词，其意义与通过一些集合自由生成群相同。特别地，因为高阶群体带有路径和更高阶路径上的运算，当这种对象由某些构造器“生成”时，这些运算会创建更多不直接来自构造器的路径。例如，在高阶归纳类型 $\mathbb{S}^1$ 中，构造器 $\mathsf{loop}$ 并不是唯一的从 $\mathsf{base}$ 到 $\mathsf{base}$ 的非平凡路径；我们还有“$\mathsf{loop} \cdot \mathsf{loop}$”和“$\mathsf{loop} \cdot \mathsf{loop} \cdot \mathsf{loop}$”等等，以及 $\mathsf{loop}^{-1}$ 等，所有这些都是不同的。这可能看起来如此显而易见，以至于不值得一提，但它与“普通”归纳类型的行为有所不同，在“普通”归纳类型中，我们可以预期在归纳类型中看不到除直接由构造器“放入”的任何东西。

其次，这种生成确实是自由生成的：高阶归纳类型实际上不允许我们施加“公理”，例如强制“$\mathsf{loop} \cdot \mathsf{loop}$”等于 $\mathsf{refl}_{\mathsf{base}}$。然而，在 $\infty$-群体 的世界中，“自由生成”和“表示”之间几乎没有区别，因为我们可以通过添加一个新的二维生成器将两个路径同伦地等同（例如，在 $\mathsf{base} = \mathsf{base}$ 中的路径 $\mathsf{loop} \cdot \mathsf{loop} = \mathsf{refl}_{\mathsf{base}}$）。当然，我们需要担心这个新生成器是否应满足其自身的“公理”，等等，但原则上，任何“表示”都可以通过将公理转化为构造器而转换为“自由”表示。正如我们将看到的，通过添加“截断构造器 (truncation constructors)”我们可以使用高阶归纳类型来表达经典概念，例如群表示。

第三，尽管高阶归纳类型包含生成该类型路径的“构造器”，它仍然是单一类型的归纳定义。特别地，正如我们将看到的，它是具有普遍性质的高阶归纳类型本身（通常通过归纳原则表示），而不是它的同一性类型。高阶归纳类型的同一性类型保留了任何同一性类型的通常归纳原则（即路径归纳），并未获得任何新的归纳原则。

因此，识别高阶归纳类型的同一性类型可能不是一件简单的事情，与我们在 Chapter 2 中能够给出显式描述所有传统类型构造操作下的同一性类型行为的方式形成对比。例如，$\mathbb{S}^1$ 中是否有从 $\mathsf{base}$ 到 $\mathsf{base}$ 的路径不是简单地由 $\mathsf{loop}$ 及其逆元的副本复合而成的？直观上，答案应该是否定的（事实上也是），但证明这一点并不简单。实际上，这类问题迅速将我们引向诸如计算球体的同伦群这类问题，这是代数拓扑中一个长期存在的问题，并且没有已知的简单公式。同伦类型论为解决这些问题带来了新的强大视角，但它也要求类型论变得与这些问题的答案一样复杂。

第四，构造器的“维度”（即它们输出点、路径、路径之间的路径等）与生成的类型在哪些维度上具有非平凡同伦之间没有直接联系。一个简单的例子是，如果一个归纳类型 $B$ 具有类型为 $A \to B$ 的构造器，那么 $A$ 中的任何路径和更高阶路径都会导致 $B$ 中的路径和更高阶路径，尽管该构造器根本不是“高阶”构造器。同样的事情也发生在高阶构造器中：具有类型 $A \to (x =_B y)$ 的构造器不仅意味着 $A$ 的点在 $B$ 中生成从 $x$ 到 $y$ 的路径，而且 $A$ 中的路径也会在这些路径之间生成路径，等等。正如我们将看到的，这种可能性是高阶归纳类型力量的主要来源之一。

另一方面，即使是不具有高阶类型输入的构造器也可能生成“意外”的高阶路径。例如，在由以下构造器生成的二维球体 $\mathbb{S}^2$ 中：

- 一个点 $\mathsf{base} : \mathbb{S}^2$，以及
- 在 $\mathsf{base} = \mathsf{base}$ 中的二维路径 $\mathsf{surf} : \mathsf{refl}_{\mathsf{base}} = \mathsf{refl}_{\mathsf{base}}$，

存在一个从 $\mathsf{refl}_{\mathsf{refl}_{\mathsf{base}}}$ 到自身的非平凡三维路径。拓扑学家会认识到这个路径是霍普夫纤维丛 (Hopf fibration) 的一个体现。从范畴论的角度来看，这与上面提到的 $\mathbb{S}^1$ 中包含的不仅仅是 $\mathsf{loop}$ 还有 $\mathsf{loop} \cdot \mathsf{loop}$ 等现象属于同一类：只不过在高阶群体中，存在运算能够提升维度。事实上，我们在 §2.1 中看到了许多这样的运算：结合律和单位律不仅仅是性质，而是运算，其输入是 1 路径，输出是 2 路径。

## 6.2 归纳原则和依赖路径 (Induction principles and dependent paths)

当我们描述像圆周这样的高阶归纳类型是由某些构造器生成时，我们必须通过给出类似于 Chapter 1 中基本类型构造器的规则来解释这意味着什么。构造器本身提供了引入规则，但要解释消去规则（即归纳和递归原则）则需要更多思考。在本书中，我们不会尝试给出什么构成“高阶归纳定义”的一般表述以及如何从这种定义中提取消去规则——事实上，这是一个微妙的问题，当前正在研究中。相反，我们将依赖一些一般的非正式讨论和大量示例。

递归原则通常很容易描述：给定任何带有与构造器为高阶归纳类型提供的结构相同的结构的类型，存在一个函数将构造器映射到该结构。例如，在 $\mathbb{S}^1$ 的情况下，递归原则表示给定一个带有点 $b : B$ 和路径 $\ell : b = b$ 的类型 $B$，存在一个函数 $f : \mathbb{S}^1 \to B$ 使得 $f(\mathsf{base}) = b$ 且 $\mathsf{ap}_f(\mathsf{loop}) = \ell$。

然而，存在一个问题，即这些计算规则是判定性（judgmental）等式还是命题性等式（路径）。对于普通归纳类型，我们毫无顾虑地将其设为判定性等式，尽管我们在 Chapter 5 中看到，将它们设为命题性等式仍然会产生同一类型（等同于等价关系）。在普通情况下，可以认为计算规则实际上是定义性等式，如导言中描述的直观意义那样。

对于高阶归纳类型，这一点就不那么清楚了。更重要的是，由于运算 $\mathsf{ap}_f$ 并不是真正的类型论的基本部分，而是我们使用同一性类型的归纳原则定义的东西（我们也可以用其他等价的方式定义），因此

在*判定性*等式中显式引用它似乎不合适。判定性等式是推理系统的一部分，不应依赖我们可能在该系统*内部*做出的特定定义选择。还有语义和实现问题需要考虑；参见注释。

将高阶归纳类型的*点*构造器的计算规则设为判定性似乎是无问题的。在上面的例子中，这意味着我们有 $f(\mathsf{base}) \equiv b$，判定性地。这种选择促进了高阶归纳类型的计算视角。而且，这也极大地简化了我们的工作，否则第二个计算规则 $\mathsf{ap}_f(\mathsf{loop}) = \ell$ 作为命题性等式将无法良好定义；我们将不得不将其一侧或另一侧与 $f(\mathsf{base})$ 和 $b$ 的指定同一性进行复合。（当然，当我们谈论更高维度的路径时，这种问题确实会出现，但在这里这不会是我们关心的重点。另见 §6.7。）因此，我们将点构造器的计算规则视为判定性，而路径和更高阶路径的计算规则视为命题性。[1]

Remark 6.2.1. 回想一下，对于普通归纳类型，我们将递归定义函数的计算规则视为不仅仅是判定性等式，而是*定义性*等式，因此我们可以使用 $:\equiv$ 符号。例如，截断前驱函数 (truncated predecessor) $p : \mathbb{N} \to \mathbb{N}$ 定义为 $p(0) :\equiv 0$ 和 $p(\mathsf{succ}(n)) :\equiv n$。在高阶归纳类型的情况下，这种符号对于点构造器（例如 $f(\mathsf{base}) :\equiv b$）是合理的，但对于路径构造器可能会产生误导，因为等式 $f(\mathsf{loop}) = \ell$ 并不是判定性的。因此，我们混合使用符号，而是写成 $f(\mathsf{loop}) := \ell$ 表示这种“定义上的命题性等式”。

现在，归纳原则（依赖消去）又如何呢？回想一下，对于一个普通归纳类型 $W$，要通过归纳证明 $\prod_{(x:W)} P(x)$，我们必须为 $W$ 的每个构造器指定一个作用于 $P$ 的运算，该运算作用于 $W$ 中该构造器之上的“纤维”。例如，如果 $W$ 是自然数 $\mathbb{N}$，那么要通过归纳证明 $\prod_{(x:\mathbb{N})} P(x)$，我们必须指定

- 一个在构造器 $0 : \mathbb{N}$ 之上的纤维中的元素 $b : P(0)$，以及
- 对于每个 $n : \mathbb{N}$，一个函数 $P(n) \to P(\mathsf{succ}(n))$。

第二个可以被视为一个位于构造器 $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$ 之上的“$P \to P$”函数，一般化为 $b : P(0)$ 位于构造器 $0 : \mathbb{N}$ 之上。

类比地，要证明 $\prod_{(x:\mathbb{S}^1)} P(x)$，我们应当指定

- 一个在构造器 $\mathsf{base} : \mathbb{S}^1$ 之上的纤维中的元素 $b : P(\mathsf{base})$，以及
- 一个从 $b$ 到 $b$ 的路径“位于构造器 $\mathsf{loop} : \mathsf{base} = \mathsf{base}$ 之上”。

请注意，尽管 $\mathbb{S}^1$ 包含 $\mathsf{loop}$ 以外的路径（如 $\mathsf{refl}_{\mathsf{base}}$ 和 $\mathsf{loop} \cdot \mathsf{loop}$），我们只需要指定一个位于构造器*本身*之上的路径。这表达了 $\mathbb{S}^1$ 是“由其构造器自由生成”的直觉。

然而，问题是，从 $b$ 到 $b$ 的路径“位于”另一条路径之上是什么意思。这绝对不意味着仅仅是一条 $b = b$ 的路径，因为那将是一条位于纤维 $P(\mathsf{base})$ 中的路径（在拓扑上，位于 $\mathsf{base}$ 处的*恒等*路径之上的路径）。然而，实际上，我们已经在 Chapter 2 中回答了这个问题：在 Lemma 2.3.4 之前的讨论中，我们得出结论，从 $u : P(x)$ 到 $v : P(y)$ 的一条路径位于 $p : x = y$ 之上可以用纤维 $P(y)$ 中的路径 $p_*(u) = v$ 表示。由于在本章中我们将大量使用这种**依赖路径**，因此我们为它们引入一个特殊符号：

$$(u =^P_p v) :\equiv (\mathsf{transport}^P(p, u) = v). \tag{6.2.2}$$

Remark 6.2.3. 定义依赖路径还有其他可能的方法。例如，我们可以考虑 $u = (p^{-1})_*(v)$，而不是 $p_*(u) = v$。我们还可以将其作为更一般的“异质等式 (heterogeneous equality)”的特例获得，或直接定义为归纳类型族。所有这些定义都会导致等价类型，因此从这个意义上讲，我们选择哪一个并不重要。然而，选择 $p_*(u) = v$ 作为定义使得推导依赖路径的其他内容变得最容易，例如 $\mathsf{apd}_f$ 生成它们的方式，或我们可以使用 §2.5 中的传递引理 (transport lemmas) 在特定类型族中计算它们。

有了依赖路径的概念，我们现在可以更精确地陈述 $\mathbb{S}^1$ 的归纳原则：给定 $P : \mathbb{S}^1 \to \mathcal{U}$ 和

- 一个元素 $b : P(\mathsf{base})$，以及
- 一个路径 $\ell : b =^P_{\mathsf{loop}} b$，

存在一个函数 $f : \prod_{(x:\mathbb{S}^1)} P(x)$，使得 $f(\mathsf{base}) \equiv b$ 并且 $\mathsf{apd}_f(\mathsf{loop}) = \ell$。与非依赖情况类似，我们称通过 $f(\mathsf{base}) :\equiv b$ 和 $\mathsf{apd}_f(\mathsf{loop}) := \ell$ 定义 $f$。

[1] 特别地，在 §1.1 的语言中，这意味着我们的高阶归纳类型是*规则*（规定如何引入这些类型及其元素、其归纳原则以及其点构造器的计算规则）和*公理*（路径构造器的计算规则，断言某些同一性类型由未指定的项占据）的混合体。我们可以希望最终会有一个更好的类型论，其中高阶归纳类型像同值性 (univalence) 一样，仅使用规则而没有公理。

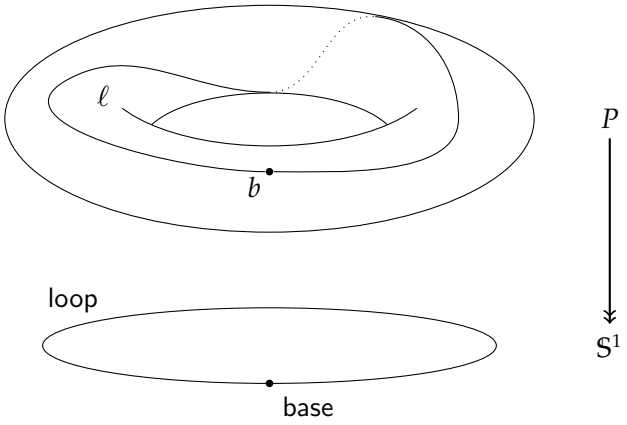



图 6.1: 圆周 $\mathbb{S}^1$ 的拓扑归纳原则

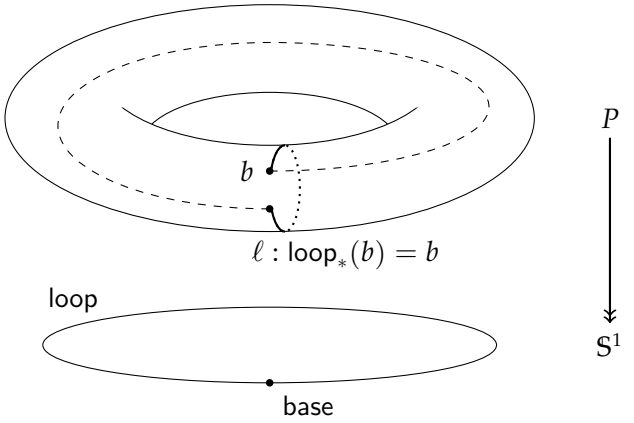



图 6.2: 圆周 $\mathbb{S}^1$ 的类型论归纳原则

Remark 6.2.4. 在非正式描述此归纳原则的应用时，我们将其视为将目标"$P(x)$ 对于所有 $x : \mathbb{S}^1$"分为两种情况，有时我们会使用诸如"当 $x$ 是 base 时"和"当 $x$ 沿 loop 变化时"之类的短语引入这两个部分。这里没有为"沿路径变化"指定具体的数学含义：这只是表示相应证明部分的开始的一种方便方式；参见 Lemma 6.4.2 了解示例。

拓扑上，$\mathbb{S}^1$ 的归纳原则可以如 Figure 6.1 所示进行可视化。给定一个圆周上的纤维丛（在图中是一个环面），定义此纤维丛的截面等同于给出一个 base 上的纤维中的点 $b$ 以及一个从 $b$ 到 $b$ 位于 loop 之上的路径。我们从类型论的角度理解这一点，使用我们的依赖路径定义，如 Figure 6.2 所示：从 $b$ 到 $b$ 沿 loop 的路径在纤维 $P(\mathsf{base})$ 中由 $\mathsf{loop}_*(b)$ 到 $b$ 的路径表示。

当然，我们期望可以通过将 $P$ 设为常类型族来从归纳原则推导出递归原则。事实上确实如此，尽管从依赖的 loop 计算规则（它涉及 $\mathsf{apd}_f$）推导出非依赖的计算规则（它涉及 $\mathsf{ap}_f$）出人意料地有点棘手。

Lemma 6.2.5. *如果 $A$ 是一个带有 $a : A$ 和 $p : a =_A a$ 的类型，那么存在一个函数 $f : \mathbb{S}^1 \to A$ 使得*

$$f(\mathsf{base}) :\equiv a$$
$$\mathsf{ap}_f(\mathsf{loop}) := p.$$

*证明.* 我们想要将 $\mathbb{S}^1$ 的归纳原则应用于常类型族 $(\lambda x.\, A) : \mathbb{S}^1 \to \mathcal{U}$。为此所需的假设是 $(\lambda x.\, A)(\mathsf{base}) \equiv A$ 的一点，我们有（即 $a : A$），以及 $a =^{x\mapsto A}_{\mathsf{loop}} a$ 中的一个依赖路径，或等价地 $\mathsf{transport}^{x\mapsto A}(\mathsf{loop}, a) = a$。该类型与 $p$ 所在的类型 $a =_A a$ 不同，但它等价于后者，因为根据 Lemma 2.3.5 我们有 $\mathsf{transportconst}^A_{\mathsf{loop}}(a) : \mathsf{transport}^{x\mapsto A}(\mathsf{loop}, a) = a$。因此，给定 $a : A$ 和 $p : a = a$，我们可以考虑复合

$$\mathsf{transportconst}^A_{\mathsf{loop}}(a) \cdot p : (a =^{x\mapsto A}_{\mathsf{loop}} a)$$

应用归纳原则，我们得到 $f : \mathbb{S}^1 \to A$ 使得

$$f(\mathsf{base}) \equiv a \qquad \text{and} \tag{6.2.6}$$
$$\mathsf{apd}_f(\mathsf{loop}) = \mathsf{transportconst}^A_{\mathsf{loop}}(a) \cdot p \tag{6.2.7}$$

现在需要推导出 $\mathsf{ap}_f(\mathsf{loop}) = p$ 等式。然而，根据 Lemma 2.3.8，我们有

$$\mathsf{apd}_f(\mathsf{loop}) = \mathsf{transportconst}^A_{\mathsf{loop}}(f(\mathsf{base})) \cdot \mathsf{ap}_f(\mathsf{loop})$$

将此与 (6.2.7) 结合，并取消 transportconst 的出现（根据 (6.2.6) 它们是相同的），我们得到 $\mathsf{ap}_f(\mathsf{loop}) = p$。 □

我们还有一个相应的唯一性原则。

Lemma 6.2.8. *如果 $A$ 是一个类型，且 $f, g : \mathbb{S}^1 \to A$ 是两个映射，$p, q$ 是两个等式：*

$$p : f(\mathsf{base}) =_A g(\mathsf{base})$$
$$q : f(\mathsf{loop}) =^{\lambda x.\, x =_A x}_{p} g(\mathsf{loop})$$

*那么对于所有 $x : \mathbb{S}^1$，我们有 $f(x) = g(x)$。*

*证明.* 我们在类型族 $P(x) :\equiv (f(x) = g(x))$ 处应用 $\mathbb{S}^1$ 的归纳原则。当 $x$ 为 base 时，$p$ 正是我们需要的。当 $x$ 沿 loop 变化时，我们需要 $p =^{\lambda x.\, f(x) = g(x)}_{\mathsf{loop}} p$，根据 ???? 这一点可以简化为 $q$。 □

这两个引理暗示了圆周的预期普遍性质：

Lemma 6.2.9. *对于任何类型 $A$，我们有一个自然的等价*

$$(\mathbb{S}^1 \to A) \;\simeq\; \sum_{x:A} (x = x)$$

*证明.* 我们有一个标准函数 $f : (\mathbb{S}^1 \to A) \to \sum_{(x:A)} (x = x)$，定义为 $f(g) :\equiv (g(\mathsf{base}), g(\mathsf{loop}))$。反过来，我们有 $g : \sum_{(x:A)} (x = x) \to (\mathbb{S}^1 \to A)$，定义为将一对 $(b, \ell)$ 变为由圆周的递归原则给出的函数 $\mathbb{S}^1 \to A$。
现在，通过圆周递归原则的计算规则，$f \circ g \sim \mathsf{id}$。而且通过唯一性原则，由于 $(g \circ f)(\mathsf{loop}) =^{\lambda x.\, x =_A x}_{\mathsf{refl}_{\mathsf{base}}} \mathsf{loop}$，再加上圆周递归原则的计算规则，$g \circ f \sim \mathsf{id}$。因此，$f$ 有一个准逆元 (quasi-inverse)，因此是一个等价。 □

与 §5.5 中一样，我们可以表明 Lemma 6.2.9 的结论等价于具有命题性计算规则的归纳原则。其他高阶归纳类型也满足类似于 Lemmas 6.2.5 and 6.2.9 的引理；我们通常会将它们的证明留给读者。现在我们继续考虑许多示例。

## 6.3 **区间** (The interval)

**区间** (interval)，我们记作 $I$，它可能比圆更简单的高阶归纳类型。它由以下部分生成：

- 一个点 $0_I : I$,
- 一个点 $1_I : I$, 和
- 一条路径 $\mathsf{seg} : 0_I =_I 1_I$.

区间的递归原理表明，给定一个类型 $B$ 以及

- 一个点 $b_0 : B$,
- 一个点 $b_1 : B$, 和

- 一条路径 $s : b_0 = b_1$,

存在一个函数 $f : I \to B$ 使得 $f(0_I) \equiv b_0$, $f(1_I) \equiv b_1$, 并且 $f(\mathsf{seg}) = s$。归纳原理表明，给定 $P : I \to \mathcal{U}$ 以及

- 一个点 $b_0 : P(0_I)$,
- 一个点 $b_1 : P(1_I)$, 和
- 一条路径 $s : b_0 =^P_{\mathsf{seg}} b_1$,

存在一个函数 $f : \prod_{(x:I)} P(x)$ 使得 $f(0_I) \equiv b_0$, $f(1_I) \equiv b_1$, 并且 $\mathsf{apd}_f(\mathsf{seg}) = s$。
纯粹从同伦角度来看，区间并不是真的有趣：

Lemma 6.3.1. *类型 $I$ 是可约的。*

*证明.* 我们证明对所有 $x : I$, 我们有 $x =_I 1_I$。换句话说，我们希望有一个函数 $f$ 的类型为 $\prod_{(x:I)}(x =_I 1_I)$。我们开始定义 $f$，如下所示：

$$f(0_I) :\equiv \mathsf{seg} \ : 0_I =_I 1_I,$$
$$f(1_I) :\equiv \mathsf{refl}_{1_I} : 1_I =_I 1_I.$$

剩下的就是定义 $\mathsf{apd}_f(\mathsf{seg})$，它必须具有类型 $\mathsf{seg} =^{\lambda x.\, x =_I 1_I}_{\mathsf{seg}} \mathsf{refl}_{1_I}$。根据定义，这个类型是 $\mathsf{seg}_*(\mathsf{seg}) =_{1_I =_I 1_I} \mathsf{refl}_{1_I}$，其结果等价于 $\mathsf{seg}^{-1} \cdot \mathsf{seg} = \mathsf{refl}_{1_I}$。但该类型中有一个规范元素，即路径逆元实际上是逆元的证明。 □

然而，从类型论的角度来看，区间仍然具有一些有趣的特性，类似于经典同伦论中的拓扑区间。例如，它使我们能够给出函数外延性（function extensionality）的简单证明。（当然，正如 §4.9 所述，在下面的证明期间，我们暂停使用函数外延性公理的整体假设。）

Lemma 6.3.2. *如果 $f, g : A \to B$ 是两个函数，并且对每个 $x : A$，$f(x) = g(x)$，那么 $f = g$ 在类型 $A \to B$ 中成立。*

*证明.* 我们称已有的证明为 $p : \prod_{(x:A)}(f(x) = g(x))$。对于所有 $x : A$，我们定义一个函数 $\widetilde{p}_x : I \to B$ 如下：

$$\widetilde{p}_x(0_I) :\equiv f(x),$$
$$\widetilde{p}_x(1_I) :\equiv g(x),$$
$$\widetilde{p}_x(\mathsf{seg}) :\equiv p(x).$$

我们现在定义 $q : I \to (A \to B)$ 如下：

$$q(i) :\equiv (\lambda x.\, \widetilde{p}_x(i))$$

那么 $q(0_I)$ 是函数 $\lambda x.\, \widetilde{p}_x(0_I)$，它等于 $f$，因为 $\widetilde{p}_x(0_I)$ 是由 $f(x)$ 定义的。同样地，我们有 $q(1_I) = g$，因此

$$q(\mathsf{seg}) : f =_{(A \to B)} g$$ □

在 Exercise 6.10 中，我们要求读者从 Lemma 6.3.2 完成函数外延性公理的完整证明。

## 6.4 **圆和球** (Circles and spheres)

我们已经讨论过圆 $\mathbb{S}^1$ 作为高阶归纳类型，它由以下部分生成：

- 一个点 $\mathsf{base} : \mathbb{S}^1$, 和
- 一条路径 $\mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}$.

它的归纳原理表明，给定 $P : \mathbb{S}^1 \to \mathcal{U}$，以及 $b : P(\mathsf{base})$ 和 $\ell : b =^P_{\mathsf{loop}} b$，我们有 $f : \prod_{(x:\mathbb{S}^1)} P(x)$，并且 $f(\mathsf{base}) \equiv b$ 且 $\mathsf{apd}_f(\mathsf{loop}) = \ell$。它的非依赖递归原理表明，给定 $B$ 以及 $b : B$ 和 $\ell : b = b$，我们有 $f : \mathbb{S}^1 \to B$，并且 $f(\mathsf{base}) \equiv b$ 且 $f(\mathsf{loop}) = \ell$。

我们观察到圆是非平凡的。

Lemma 6.4.1. $\mathsf{loop} \neq \mathsf{refl}_{\mathsf{base}}$。

证明. 假设 $\mathsf{loop} = \mathsf{refl}_{\mathsf{base}}$。然后，由于对于任何带有 $x : A$ 和 $p : x = x$ 的类型 $A$，存在一个函数 $f : \mathbb{S}^1 \to A$ 定义为 $f(\mathsf{base}) :\equiv x$ 且 $f(\mathsf{loop}) := p$，我们有

$$p = f(\mathsf{loop}) = f(\mathsf{refl}_{\mathsf{base}}) = \mathsf{refl}_x$$

但这意味着每种类型都是一个集合，正如我们所见，这并不成立（见 Example 3.1.9）。 □

圆还具有以下有趣的特性，它作为反例的来源非常有用。

Lemma 6.4.2. 存在一个元素 $\prod_{(x:\mathbb{S}^1)}(x = x)$，它不等于 $x \mapsto \mathsf{refl}_x$。

证明. 我们通过 $\mathbb{S}^1$-归纳定义 $f : \prod_{(x:\mathbb{S}^1)}(x = x)$。当 $x$ 是 $\mathsf{base}$ 时，我们设 $f(\mathsf{base}) :\equiv \mathsf{loop}$。现在，当 $x$ 沿着 $\mathsf{loop}$ 变化时（见 Remark 6.2.4），我们必须证明 $\mathsf{transport}^{x\mapsto x=x}(\mathsf{loop}, \mathsf{loop}) = \mathsf{loop}$。然而，在 ?? 中，我们观察到 $\mathsf{transport}^{x\mapsto x=x}(p, q) = p^{-1} \cdot q \cdot p$，所以我们需要证明的是 $\mathsf{loop}^{-1} \cdot \mathsf{loop} \cdot \mathsf{loop} = \mathsf{loop}$。但是，通过消去逆元，这显然是正确的。
要证明 $f \neq (x \mapsto \mathsf{refl}_x)$，我们只需证明 $f(\mathsf{base}) \neq \mathsf{refl}_{\mathsf{base}}$。但 $f(\mathsf{base}) = \mathsf{loop}$，所以这正是前一个引理的内容。 □

例如，这使我们能够扩展 Example 3.1.9，表明包含圆的任何宇宙都不能是一个 1-型（1-type）。

Corollary 6.4.3. 如果类型 $\mathbb{S}^1$ 属于某个宇宙 $\mathcal{U}$，那么 $\mathcal{U}$ 不是一个 1-型。

证明. 类型 $\mathbb{S}^1 = \mathbb{S}^1$ 在 $\mathcal{U}$ 中，通过一值性，可以等价于 $\mathbb{S}^1$ 的自等价类型 $\mathbb{S}^1 \simeq \mathbb{S}^1$，所以只需证明 $\mathbb{S}^1 \simeq \mathbb{S}^1$ 不是一个集合。为此，只需证明其等式类型 $\mathsf{id}_{\mathbb{S}^1} =_{(\mathbb{S}^1 \simeq \mathbb{S}^1)} \mathsf{id}_{\mathbb{S}^1}$ 不是一个单纯命题（mere proposition）。由于作为等价是一个单纯命题，这个类型等价于 $\mathsf{id}_{\mathbb{S}^1} =_{(\mathbb{S}^1 \to \mathbb{S}^1)} \mathsf{id}_{\mathbb{S}^1}$。但通过函数外延性，这等价于 $\prod_{(x:\mathbb{S}^1)}(x = x)$，正如我们在 Lemma 6.4.2 中所见，其中包含两个不相等的元素。 □

我们还提到，2-球 $\mathbb{S}^2$ 应该是高阶归纳类型，由以下部分生成：

- 一个点 $\mathsf{base} : \mathbb{S}^2$, 和
- 一个 2 维路径 $\mathsf{surf} : \mathsf{refl}_{\mathsf{base}} = \mathsf{refl}_{\mathsf{base}}$ 在 $\mathsf{base} = \mathsf{base}$ 中。

对于 $\mathbb{S}^2$ 的递归原理并不难：它表明给定 $B$，其中 $b : B$ 和 $s : \mathsf{refl}_b = \mathsf{refl}_b$，我们有 $f : \mathbb{S}^2 \to B$，其中 $f(\mathsf{base}) \equiv b$ 且 $\mathsf{ap}^2_f(\mathsf{surf}) = s$。这里 "$\mathsf{ap}^2_f(\mathsf{surf})$" 表示将 $f$ 的函子性扩展到二维路径的操作，可以精确地表述如下。

Lemma 6.4.4. 给定 $f : A \to B$ 和 $x, y : A$ 以及 $p, q : x = y$ 和 $r : p = q$，我们有路径 $\mathsf{ap}^2_f(r) : f(p) = f(q)$。

证明. 通过路径归纳法，我们可以假设 $p \equiv q$ 并且 $r$ 是反射性。但此时我们可以定义 $\mathsf{ap}^2_f\big(\mathsf{refl}_p\big) :\equiv \mathsf{refl}_{f(p)}$。 □

为了陈述一般的归纳原理，我们需要该引理的依赖函数版本，这反过来需要依赖二维路径的概念。如前所述，有多种方式来定义这种事物；其中一种方式是通过传输的二维版本。

Lemma 6.4.5. 给定 $P : A \to \mathcal{U}$ 和 $x, y : A$ 以及 $p, q : x = y$ 和 $r : p = q$，对于任何 $u : P(x)$，我们有 $\mathsf{transport}^2(r, u) : p_*(u) = q_*(u)$。

证明. 通过路径归纳法。 □

现在，假设给定 $x,y:A$ 和 $p,q:x=y$ 以及 $r:p=q$，还有点 $u:P(x)$ 和 $v:P(y)$，以及依赖路径 $h:u =^P_p v$ 和 $k:u =^P_q v$。根据我们对依赖路径的定义，这意味着 $h:p_*(u)=v$ 和 $k:q_*(u)=v$。因此，将依赖二维路径的类型定义为

$$(h =^P_r k) :\equiv (h = \mathsf{transport}^2(r,u)\cdot k)$$

我们现在可以陈述 Lemma 6.4.4 的依赖版本。

Lemma 6.4.6. 给定 $P:A\to\mathcal{U}$ 和 $x,y:A$ 以及 $p,q:x=y$ 和 $r:p=q$ 以及一个函数 $f:\prod_{(x:A)}P(x)$，我们有 $\mathsf{apd}^2_f(r):\mathsf{apd}_f(p) =^P_r \mathsf{apd}_f(q)$。

证明. 路径归纳。 □

现在我们可以陈述 $\mathbb{S}^2$ 的归纳原理：假设给定 $P:\mathbb{S}^2\to\mathcal{U}$，其中 $b:P(\mathsf{base})$ 和 $s:\mathsf{refl}_b =^Q_{\mathsf{surf}} \mathsf{refl}_b$，其中 $Q :\equiv \lambda p.\, b =^P_p b$。然后存在一个函数 $f:\prod_{(x:\mathbb{S}^2)}P(x)$ 使得 $f(\mathsf{base})\equiv b$ 并且 $\mathsf{apd}^2_f(\mathsf{surf}) = s$。

当然，这种显式的方法随着维度的升高变得越来越复杂。因此，如果我们想为所有 $n$ 定义 $n$-球，我们需要一些更系统的方法。一种方法是直接处理 $n$-维循环（$n$-dimensional loops），而不是一般的 $n$-维路径。

回顾 §2.1 中 点化类型 (pointed types) $\mathcal{U}_*$ 的定义，以及 $n$ 次循环空间 $\Omega^n:\mathcal{U}_*\to\mathcal{U}_*$ (Definitions 2.1.7 and 2.1.8)。现在我们可以定义 $n$-球 $\mathbb{S}^n$ 作为由以下部分生成的高阶归纳类型

- 一个点 $\mathsf{base}:\mathbb{S}^n$, 和
- 一个 $n$-循环 $\mathsf{loop}_n:\Omega^n(\mathbb{S}^n,\mathsf{base})$。

为了写出这种表示的归纳原理，我们需要定义 “依赖 $n$-循环 (dependent $n$-loop)” 的概念，以及依赖函数对 $n$-循环的作用。我们将此留给读者（见 Exercise 6.4）；在下一节中，我们将讨论一种不同的定义球的方法，有时更易于处理。

## 6.5 悬挂 (Suspensions)

类型 $A$ 的 **悬挂** (suspension) 是使 $A$ 的点变为路径的通用方式（因此 $A$ 中的路径变为 2-路径，以此类推）。它是一个类型 $\Sigma A$，由以下生成器定义：[2]

- 一个点 $\mathsf{N}:\Sigma A$,
- 一个点 $\mathsf{S}:\Sigma A$, 和
- 一个函数 $\mathsf{merid}:A\to(\mathsf{N} =_{\Sigma A} \mathsf{S})$。

这些名称意在表明某种 “球体”，具有北极、南极，以及从一个极点到另一个极点的一组 $A$ 的子午线。实际上，正如我们将看到的，如果 $A=\mathbb{S}^1$，那么它的悬挂等价于普通球面的表面，即 $\mathbb{S}^2$。

悬挂的递归原理表明，给定类型 $B$ 以及

- 点 $n,s:B$ 和
- 函数 $m:A\to(n=s)$,

我们有一个函数 $f:\Sigma A\to B$ 使得 $f(\mathsf{N})\equiv n$ 和 $f(\mathsf{S})\equiv s$，并且对所有 $a:A$，我们有 $f(\mathsf{merid}(a)) = m(a)$。类似地，归纳原理表明，给定 $P:\Sigma A\to\mathcal{U}$，并且

- 一个点 $n:P(\mathsf{N})$,
- 一个点 $s:P(\mathsf{S})$, 和
- 对每个 $a:A$，一条路径 $m(a):n =^P_{\mathsf{merid}(a)} s$,

[2] 这里有一个不幸的符号冲突，依赖对类型对也用 $\Sigma$ 表示。然而，通常上下文可以消除歧义。

存在一个函数 $f : \prod_{(x:\Sigma A)} P(x)$ 使得 $f(\mathsf{N}) \equiv n$，$f(\mathsf{S}) \equiv s$，并且对每个 $a : A$，我们有 $\mathsf{apd}_f(\mathsf{merid}(a)) = m(a)$。
我们的第一个关于悬挂的观察是，它提供了另一种定义圆的方法。

Lemma 6.5.1. $\Sigma\mathbf{2} \simeq \mathbb{S}^1$。

证明. 定义 $f : \Sigma\mathbf{2} \to \mathbb{S}^1$ 通过递归使得 $f(\mathsf{N}) :\equiv \mathsf{base}$ 和 $f(\mathsf{S}) :\equiv \mathsf{base}$，同时 $f(\mathsf{merid}(0_\mathbf{2})) := \mathsf{loop}$，但 $f(\mathsf{merid}(1_\mathbf{2})) := \mathsf{refl}_{\mathsf{base}}$。定义 $g : \mathbb{S}^1 \to \Sigma\mathbf{2}$ 通过递归使得 $g(\mathsf{base}) :\equiv \mathsf{N}$ 并且 $g(\mathsf{loop}) := \mathsf{merid}(0_\mathbf{2}) \cdot \mathsf{merid}(1_\mathbf{2})^{-1}$。我们现在展示 $f$ 和 $g$ 是准逆元。
首先，我们通过归纳证明 $g(f(x)) = x$ 对所有 $x : \Sigma\mathbf{2}$ 成立。如果 $x \equiv \mathsf{N}$，那么 $g(f(\mathsf{N})) \equiv g(\mathsf{base}) \equiv \mathsf{N}$，所以我们有 $\mathsf{refl}_\mathsf{N} : g(f(\mathsf{N})) = \mathsf{N}$。如果 $x \equiv \mathsf{S}$，那么 $g(f(\mathsf{S})) \equiv g(\mathsf{base}) \equiv \mathsf{N}$，并且我们选择等式 $\mathsf{merid}(1_\mathbf{2}) : g(f(\mathsf{S})) = \mathsf{S}$。剩下的就是展示，对于任何 $y : \mathbf{2}$，当 $x$ 沿着 $\mathsf{merid}(y)$ 变化时，这些等式是保持的，这就是说，当 $\mathsf{refl}_\mathsf{N}$ 沿着 $\mathsf{merid}(y)$ 传输时，它产生 $\mathsf{merid}(1_\mathbf{2})$。通过路径空间和拉回纤维丛中的传输，这意味着我们需要证明

$$g(f(\mathsf{merid}(y)))^{-1} \cdot \mathsf{refl}_\mathsf{N} \cdot \mathsf{merid}(y) = \mathsf{merid}(1_\mathbf{2})$$

当然，我们可以消除 $\mathsf{refl}_\mathsf{N}$。现在通过 $\mathbf{2}$-归纳法，我们可以假设 $y \equiv 0_\mathbf{2}$ 或者 $y \equiv 1_\mathbf{2}$。如果 $y \equiv 0_\mathbf{2}$，那么我们有

$$\begin{aligned} g(f(\mathsf{merid}(0_\mathbf{2})))^{-1} \cdot \mathsf{merid}(0_\mathbf{2}) &= g(\mathsf{loop})^{-1} \cdot \mathsf{merid}(0_\mathbf{2}) \\ &= (\mathsf{merid}(0_\mathbf{2}) \cdot \mathsf{merid}(1_\mathbf{2})^{-1})^{-1} \cdot \mathsf{merid}(0_\mathbf{2}) \\ &= \mathsf{merid}(1_\mathbf{2}) \cdot \mathsf{merid}(0_\mathbf{2})^{-1} \cdot \mathsf{merid}(0_\mathbf{2}) \\ &= \mathsf{merid}(1_\mathbf{2}) \end{aligned}$$

而如果 $y \equiv 1_\mathbf{2}$，那么我们有

$$\begin{aligned} g(f(\mathsf{merid}(1_\mathbf{2})))^{-1} \cdot \mathsf{merid}(1_\mathbf{2}) &= g(\mathsf{refl}_{\mathsf{base}})^{-1} \cdot \mathsf{merid}(1_\mathbf{2}) \\ &= \mathsf{refl}_\mathsf{N}{}^{-1} \cdot \mathsf{merid}(1_\mathbf{2}) \\ &= \mathsf{merid}(1_\mathbf{2}) \end{aligned}$$

因此，对于所有 $x : \Sigma\mathbf{2}$，我们有 $g(f(x)) = x$。
现在我们通过归纳证明 $f(g(x)) = x$ 对所有 $x : \mathbb{S}^1$ 成立。如果 $x \equiv \mathsf{base}$，那么 $f(g(\mathsf{base})) \equiv f(\mathsf{N}) \equiv \mathsf{base}$，所以我们有 $\mathsf{refl}_{\mathsf{base}} : f(g(\mathsf{base})) = \mathsf{base}$。剩下的就是展示，当 $x$ 沿着 $\mathsf{loop}$ 变化时，这个等式是保持的，这就是说，它沿着 $\mathsf{loop}$ 传输到它自身。同样地，通过路径空间和拉回纤维丛中的传输，这意味着需要证明

$$f(g(\mathsf{loop}))^{-1} \cdot \mathsf{refl}_{\mathsf{base}} \cdot \mathsf{loop} = \mathsf{refl}_{\mathsf{base}}$$

然而，我们有

$$\begin{aligned} f(g(\mathsf{loop})) &= f\Big(\mathsf{merid}(0_\mathbf{2}) \cdot \mathsf{merid}(1_\mathbf{2})^{-1}\Big) \\ &= f(\mathsf{merid}(0_\mathbf{2})) \cdot f(\mathsf{merid}(1_\mathbf{2}))^{-1} \\ &= \mathsf{loop} \cdot \mathsf{refl}_{\mathsf{base}} \end{aligned}$$

因此这很容易跟随。 □

从拓扑学上讲，二点空间 $\mathbf{2}$ 也被称为 0 维球 (0-dimensional sphere)，$\mathbb{S}^0$。（例如，它是 $\mathbb{R}^1$ 中距离原点为 1 的点集，正如拓扑 1-球是 $\mathbb{R}^2$ 中距离原点为 1 的点集一样。）因此，Lemma 6.5.1 可以暗示性地表述为 $\Sigma\mathbb{S}^0 \simeq \mathbb{S}^1$。实际上，这个模式继续：我们可以通过递归定义所有的球

$$\mathbb{S}^0 :\equiv \mathbf{2} \qquad \text{并且} \qquad \mathbb{S}^{n+1} :\equiv \Sigma\mathbb{S}^n \tag{6.5.2}$$

我们甚至可以从低一维开始定义 $\mathbb{S}^{-1} :\equiv \mathbf{0}$，并观察到 $\Sigma\mathbf{0} \simeq \mathbf{2}$。
要仔细证明这与上一节中的 $\mathbb{S}^n$ 的定义一致，需要使后者更加明确。然而，我们可以证明递归定义具有我们期望另一种定义应具有的相同的通用性质。如果 $(A, a_0)$ 和 $(B, b_0)$ 是点化类型（基点通常隐式地保留），则令 $\mathsf{Map}_*(A, B)$ 表示基映射类型：

$$\mathsf{Map}_*(A, B) :\equiv \sum_{f:A\to B} (f(a_0) = b_0)$$

注意，任何类型 $A$ 都可以产生一个点化类型 $A_+ :\equiv A + \mathbf{1}$，基点为 $\mathsf{inr}(\star)$；这称为 *添加一个不相交的基点*。

Lemma 6.5.3. *对于类型 $A$ 和点化类型 $(B, b_0)$，我们有*

$$\mathsf{Map}_*(A_+, B) \simeq (A \to B)$$

注意，在右边我们有普通的 *非点化函数的类型*从 $A$ 到 $B$。

*证明.* 从左到右，给定 $f : A_+ \to B$，其中 $p : f(\mathsf{inr}(\star)) = b_0$，我们有 $f \circ \mathsf{inl} : A \to B$。从右到左，给定 $g : A \to B$，我们定义 $g' : A_+ \to B$ 通过 $g'(\mathsf{inl}(a)) :\equiv g(a)$ 和 $g'(\mathsf{inr}(u)) :\equiv b_0$。我们留给读者证明这些是准逆操作。 □

特别地，注意到 $\mathbf{2} \simeq \mathbf{1}_+$。因此，对于任何点化类型 $B$，我们有

$$\mathsf{Map}_*(\mathbf{2}, B) \simeq (\mathbf{1} \to B) \simeq B$$

现在回顾循环空间操作 $\Omega$ 作用于点化类型，定义 $\Omega(A, a_0) :\equiv (a_0 =_A a_0, \mathsf{refl}_{a_0})$。我们也可以使悬挂 $\Sigma$ 作用于点化类型，通过 $\Sigma(A, a_0) :\equiv (\Sigma A, \mathsf{N})$。

Lemma 6.5.4. *对于点化类型 $(A, a_0)$ 和 $(B, b_0)$ 我们有*

$$\mathsf{Map}_*(\Sigma A, B) \simeq \mathsf{Map}_*(A, \Omega B)$$

*证明.* 我们首先观察到以下一系列等价关系：

$$\begin{aligned}
\mathsf{Map}_*(\Sigma A, B) &:\equiv \sum_{f:\Sigma A\to B} (f(\mathsf{N}) = b_0) \\
&\simeq \sum_{f:\sum_{(b_n:B)}\sum_{(b_s:B)}(A\to(b_n=b_s))} (\mathsf{pr}_1(f) = b_0) \\
&\simeq \sum_{(b_n:B)}\sum_{(b_s:B)} \big(A \to (b_n = b_s)\big) \times (b_n = b_0) \\
&\simeq \sum_{(p:\sum_{(b_n:B)}(b_n=b_0))}\sum_{(b_s:B)} (A \to (\mathsf{pr}_1(p) = b_s)) \\
&\simeq \sum_{b_s:B} (A \to (b_0 = b_s))
\end{aligned}$$

第一个等价关系是通过悬挂的通用性质，它表明

$$\Big(\Sigma A \to B\Big) \simeq \Big(\sum_{(b_n:B)}\sum_{(b_s:B)} (A \to (b_n = b_s))\Big)$$

从右到左的函数由递归器给出（见 Exercise 6.11）。第二个和第三个等价关系是通过 Exercise 2.10，以及组件的重新排序。最后一个等价关系是通过 Lemma 3.11.9，因为根据 Lemma 3.11.8，$\sum_{(b_n:B)}(b_n = b_0)$ 是可约的，中心为 $(b_0, \mathsf{refl}_{b_0})$。

现在通过以下等价链完成证明：

$$\begin{aligned}\sum_{b_s:B}(A\to(b_0=b_s)) &\simeq \sum_{(b_s:B)}\ \sum_{(g:A\to(b_0=b_s))}\ \sum_{(q:b_0=b_s)}(g(a_0)=q)\\ &\simeq \sum_{(r:\sum_{(b_s:B)}(b_0=b_s))}\ \sum_{(g:A\to(b_0=\mathsf{pr}_1(r)))}(g(a_0)=\mathsf{pr}_2(r))\\ &\simeq \sum_{g:A\to(b_0=b_0)}(g(a_0)=\mathsf{refl}_{b_0})\\ &\equiv \mathsf{Map}_*(A,\Omega B)\end{aligned}$$

类似于之前，第一个和最后一个等价关系通过 Lemmas 3.11.8 and 3.11.9，第二个通过 Exercise 2.10 和组件的重新排序。 □

特别地，对于按 (6.5.2) 定义的球，我们有

$$\mathsf{Map}_*(\mathbb{S}^n,B)\simeq\mathsf{Map}_*(\mathbb{S}^{n-1},\Omega B)\simeq\cdots\simeq\mathsf{Map}_*(\mathbf{2},\Omega^n B)\simeq\Omega^n B$$

因此，这些 $\mathbb{S}^n$ 具有我们期望从 §6.4 中直接基于 $n$ 次循环空间 定义的球应具有的通用性质。

## 6.6 胞腔复形 (Cell complexes)

在经典拓扑学中，*胞腔复形* (cell complex) 是通过逐步将圆盘沿其边界附加而得到的空间。如果一个 $n$ 维圆盘的边界被限制在严格低于 $n$ 维的圆盘内（即 $(n-1)$ 骨架 (skeleton)），则该空间称为 CW 复形 (CW complex)。

任何有限 CW 复形都可以表示为一个高阶归纳类型 (higher inductive type)，通过将 $n$ 维圆盘转化为 $n$ 维路径，并将附加映射 (attaching map)的像分为路径构造器 (path constructor) 的源 (source)和目标 (target)，每个都写成低维路径的组合。我们在 §6.4 中对 $\mathbb{S}^1$ 和 $\mathbb{S}^2$ 的明确定义具有这种形式。

另一个例子是圆环 $T^2$，它由以下生成：

- 一个点 $b:T^2$,
- 一条路径 $p:b=b$,
- 另一条路径 $q:b=b$, 以及
- 一个 2-路径 $t:p\cdot q=q\cdot p$。

也许看出这是一个圆环的最简单方法是从一个矩形开始，它有四个角 $a,b,c,d$，四条边 $p,q,r,s$，以及一个内部路径，它显然是从 $p\cdot q$ 到 $r\cdot s$ 的 2-路径 $t$：

$$\begin{array}{ccc} a & \xrightarrow{\ p\ } & b \\ {\scriptstyle r}\downarrow & \Downarrow t & \downarrow{\scriptstyle q} \\ c & \xrightarrow[\ s\ ]{} & d \end{array}$$

现在将边 $r$ 与 $q$ 以及边 $s$ 与 $p$ 识别，从而也识别了所有四个角。在拓扑学上，可以看出这种识别产生了一个圆环。

对于圆环的归纳原理是我们到目前为止写出的最棘手的一个。给定 $P:T^2\to\mathcal{U}$，对于截面 $\prod_{(x:T^2)}P(x)$，我们需要

- 一个点 $b':P(b)$,
- 一条路径 $p':b'=^P_p b'$,
- 一条路径 $q':b'=^P_q b'$, 以及
- 一个 2-路径 $t'$，在 $t$ 上方连接复合路径 $p'\cdot q'$ 和 $q'\cdot p'$。

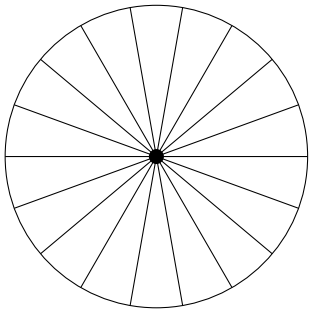


图 6.3: 由一个中心与辐条组成的二维圆盘

为了理解这最后的数据，我们需要一个依赖路径的复合操作，但这并不难定义。然后，归纳原理给出一个函数 $f : \prod_{(x:T^2)} P(x)$ 使得 $f(b) \equiv b'$ 且 $\mathsf{apd}_f(p) = p'$ 和 $\mathsf{apd}_f(q) = q'$ 并且类似于 "$\mathsf{apd}_f^2(t) = t'$" 的等式。然而，这在目前状态下是类型不正确的，首先是因为等式 $\mathsf{apd}_f(p) = p'$ 和 $\mathsf{apd}_f(q) = q'$ 不是判断等式 (judgmental equality)，其次是因为 $\mathsf{apd}_f$ 仅在同伦 (homotopy) 意义上保留路径的连接。我们将细节留给读者（见 Exercise 6.1）。

当然，圆环的另一个定义是 $T^2 :\equiv \mathbb{S}^1 \times \mathbb{S}^1$（在 Exercise 6.3 中，我们要求读者验证这两个定义的等价性）。然而，胞腔复形的定义很容易推广到其他没有这样描述的空间，例如克莱因瓶 (Klein bottle)、射影平面 (projective plane) 等等。但是，书写归纳原理变得越来越困难，要求我们定义依赖 $n$-路径和 $\mathsf{apd}$ 作用于 $n$-路径的概念。幸运的是，一旦我们掌握了球面 (spheres)，就有办法绕过这个问题。

## 6.7 中心与辐条 (Hubs and spokes)

在拓扑学中，人们通常通过沿着 $(n-1)$ 维边界球体 (boundary spheres) 附加 $n$ 维圆盘来构建 CW 复形。然而，另一种表达方式是通过粘合在 $(n-1)$ 维球体上的锥体 (cone)。也就是说，我们将一个圆盘视为由一个锥顶（或"中心 (hub)"）组成，辐条 (meridians)（或"辐条 (spokes)"）将该点连续地连接到边界上的每个点，如 Figure 6.3 所示。

我们可以利用这一思想来表达包含 $n$ 维路径构造器的高阶归纳类型，其中 $n > 1$，而这些构造器仅包含一维路径构造器。关键在于，我们可以通过一个 $(n-1)$ 维对象参数化的连续的一维路径族来获得一个 $n$ 维路径。最简单的 $(n-1)$ 维对象是 $(n-1)$ 球面，尽管在某些情况下，一个不同的对象可能更合适。（回顾一下，我们能够在 §6.5 中使用悬挂 (suspensions) 递归地定义球面，这仅涉及一维路径构造器。事实上，悬挂也可以看作是这种思想的一个实例，因为它涉及一个由被悬挂的类型参数化的一维路径族。）

例如，前一节中的圆环 $T^2$ 可以通过以下方式生成：

- 一个点 $b : T^2$,
- 一条路径 $p : b = b$,
- 另一条路径 $q : b = b$,
- 一个点 $h : T^2$, 以及
- 对每个 $x : \mathbb{S}^1$，一条路径 $s(x) : f(x) = h$，其中 $f : \mathbb{S}^1 \to T^2$ 定义为 $f(\mathsf{base}) :\equiv b$ 和 $f(\mathsf{loop}) := p \cdot q \cdot p^{-1} \cdot q^{-1}$。

这个版本的圆环的归纳原理表明，给定 $P : T^2 \to \mathcal{U}$，对于截面 $\prod_{(x:T^2)} P(x)$，我们需要

- 一个点 $b' : P(b)$,
- 一条路径 $p' : b' =_p^P b'$,
- 一条路径 $q' : b' =_q^P b'$,
- 一个点 $h' : P(h)$, 以及

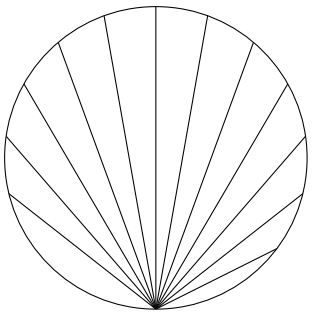


图 6.4: 无中心的辐条

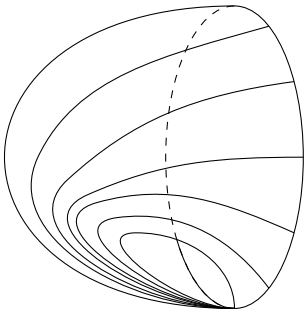


图 6.5: 无中心的辐条 II

- 对每个 $x : \mathbb{S}^1$，一条路径 $g(x) =^P_{s(x)} h'$，其中 $g : \prod_{(x:\mathbb{S}^1)} P(f(x))$ 定义为 $g(\mathsf{base}) :\equiv b'$ 和 $\mathsf{apd}_g(\mathsf{loop}) \coloneqq t(p' \centerdot q' \centerdot (p')^{-1} \centerdot (q')^{-1})$。在后者中，$\centerdot$ 表示依赖路径的连接，$t : (b' =^P_{f(\mathsf{loop})} b') \simeq (b' =^{P\circ f}_{\mathsf{loop}} b')$ 的定义留给读者完成。

注意，不需要依赖 2-路径或 $\mathsf{apd}^2$。我们将计算规则的书写留给读者。

Remark 6.7.1. 有人可能会质疑引入中心点 $h$ 的必要性；为什么我们不能简单地添加连续连接圆盘边界到边界上某个点的路径，如 Figure 6.4 所示？然而，这在没有进一步修改的情况下是行不通的。因为，如果给定某个 $f : \mathbb{S}^1 \to X$，我们给出一个路径构造器连接每个 $f(x)$ 到 $f(\mathsf{base})$，那么我们最终得到的更像是 Figure 6.5 中的图像，一个顶点被扭曲并粘合到其底部某个点上的锥体。问题在于，从 $f(\mathsf{base})$ 到其本身的指定路径可能不是反身性 (reflexivity)。我们可以通过添加一个 2-维路径构造器来解决这个问题，以确保这种情况，但使用一个独立的中心避免了需要任何维度超过 1 的路径构造器。

Remark 6.7.2. 请注意，这种将高阶路径转换为一阶路径的方法并不保留这些路径的判断计算规则，但它确实保留了命题计算规则。

## 6.8 **推挤** (Pushouts)

从范畴论 (category-theoretic) 的角度来看，任何基础系统的一个重要方面是构造极限 (limits) 和余极限 (colimits) 的能力。在集合论基础中，这些是集合的极限和余极限，而在我们的情况下，它们是*类型* (types) 的极限和余极限。我们在 §2.9 中已经看到，笛卡尔乘积类型具有类型范畴积 (categorical product of types) 的正确通用性质 (universal property)，而在 Exercise 2.9 中，余积类型 (coproduct types) 也具有它们预期的通用性质。

如 §2.9 中所述，可以使用等式类型 (identity types) 和 $\Sigma$ 类型来构造更一般的极限，例如 $f : A \to C$ 和 $g : B \to C$ 的纤维积 (pullback) 是 $\sum_{(a:A)} \sum_{(b:B)} (f(a) = g(b))$（见 Exercise 2.11）。然而，更一般的*余极限*需要识别来自不同类型的元素，高阶归纳类型 (higher inductive types) 对此非常适用。由于我们所有的构造都是同伦不变的 (homotopy-invariant)，因此我们所有的余极限都是*同伦余极限* (homotopy colimits)，但为了简洁起见，我们省略了无处不在的形容词。

本节我们讨论*推挤* (pushouts)，作为也许最简单且最有用的余极限之一。确实，人们期望所有有限余极限（对于一个合适的同伦定义“有限”）可以通过推挤和有限余积构造。也可以使用高阶归纳类型直接构造更一般的余极限，但这有些技术性问题，并且不完全令人满意，因为我们尚未有一个完全通用的同伦一致图的良好概念。

假设给定一个类型和函数的跨图：

$$\mathscr{D} = \begin{array}{ccc} C & \xrightarrow{g} & B \\ {\scriptstyle f}\downarrow & & \\ A & & \end{array}$$

该跨图的**推挤** (pushout) 是高阶归纳类型 $A \sqcup^C B$，由以下内容呈现：

- 一个函数 $\mathsf{inl} : A \to A \sqcup^C B$,
- 一个函数 $\mathsf{inr} : B \to A \sqcup^C B$, 以及
- 对于每个 $c : C$，一个路径 $\mathsf{glue}(c) : (\mathsf{inl}(f(c)) = \mathsf{inr}(g(c)))$。

换句话说，$A \sqcup^C B$ 是 $A$ 和 $B$ 的不交并 (disjoint union)，并且对于每个 $c : C$，有一个 $f(c)$ 和 $g(c)$ 相等的证据。递归原理 (recursion principle) 表示，如果 $D$ 是另一个类型，我们可以通过定义以下内容来定义一个映射 $s : A \sqcup^C B \to D$：

- 对于每个 $a : A$，$s(\mathsf{inl}(a))$ 的值：$D$，
- 对于每个 $b : B$，$s(\mathsf{inr}(b))$ 的值：$D$，以及
- 对于每个 $c : C$，$\mathsf{ap}_s(\mathsf{glue}(c))$ 的值：$s(\mathsf{inl}(f(c))) = s(\mathsf{inr}(g(c)))$。

我们将归纳原理的表述留给读者。它还暗示了唯一性原理 (uniqueness principle)，即如果 $s, s' : A \sqcup^C B \to D$ 是两个映射，并且

$$\begin{aligned} s(\mathsf{inl}(a)) &= s'(\mathsf{inl}(a)) \\ s(\mathsf{inr}(b)) &= s'(\mathsf{inr}(b)) \\ \mathsf{ap}_s(\mathsf{glue}(c)) &= \mathsf{ap}_{s'}(\mathsf{glue}(c)) \qquad \text{(基于前两个等式)} \end{aligned}$$

对于每个 $a, b, c$，则 $s = s'$。
为了表述推挤的通用性质 (universal property)，我们引入以下定义。

Definition 6.8.1. 给定一个跨图 $\mathscr{D} = (A \xleftarrow{f} C \xrightarrow{g} B)$ 和一个类型 $D$，在 $\mathscr{D}$ 下，顶点为 $D$ 的**余锥** (cocone)由函数 $i : A \to D$ 和 $j : B \to D$ 以及同伦 $h : \prod_{(c:C)} (i(f(c)) = j(g(c)))$ 组成：

$$\begin{array}{ccc} C & \xrightarrow{g} & B \\ {\scriptstyle f}\downarrow & \swarrow{\scriptstyle h} & \downarrow{\scriptstyle j} \\ A & \xrightarrow[i]{} & D \end{array}$$

我们用 $\mathsf{cocone}_{\mathscr{D}}(D)$ 表示所有此类余锥的类型，即

$$\mathsf{cocone}_{\mathscr{D}}(D) :\equiv \sum_{(i:A\to D)} \sum_{(j:B\to D)} \prod_{(c:C)} (i(f(c)) = j(g(c))).$$

当然，存在一个在 $\mathscr{D}$ 下，顶点为 $A \sqcup^C B$ 的标准余锥，由 $\mathsf{inl}$、$\mathsf{inr}$ 和 $\mathsf{glue}$ 组成。

$$\begin{array}{ccc} C & \xrightarrow{g} & B \\ {\scriptstyle f}\downarrow & \swarrow{\scriptstyle \mathsf{glue}} & \downarrow{\scriptstyle \mathsf{inr}} \\ A & \xrightarrow[\mathsf{inl}]{} & A \sqcup^C B \end{array}$$

以下引理表明这是通用的此类余锥。

Lemma 6.8.2. *对于任何类型 $E$，存在等价*

$$(A \sqcup^C B \to E) \;\simeq\; \mathsf{cocone}_{\mathscr{D}}(E).$$

*证明.* 我们考虑任意类型 $E : \mathcal{U}$。存在一个标准函数 $c_{\sqcup}$，定义如下

$$\begin{cases} (A \sqcup^C B \to E) & \longrightarrow & \mathsf{cocone}_{\mathscr{D}}(E) \\ \qquad\qquad t & \longmapsto & (t \circ \mathsf{inl}, t \circ \mathsf{inr}, \mathsf{ap}_t \circ \mathsf{glue}) \end{cases}$$

我们用非正式符号 $t \mapsto t \circ c_{\sqcup}$ 表示此函数。我们证明这是一个等价。
首先，给定 $c = (i, j, h) : \mathsf{cocone}_{\mathscr{D}}(E)$，我们需要构造一个从 $A \sqcup^C B$ 到 $E$ 的映射 $\mathsf{s}(c)$。

$$\begin{array}{ccc} C & \xrightarrow{g} & B \\ {\scriptstyle f}\downarrow & \overset{h}{\swarrow} & \downarrow{\scriptstyle j} \\ A & \xrightarrow[i]{} & E \end{array}$$

映射 $\mathsf{s}(c)$ 定义如下：

$$\begin{aligned} \mathsf{s}(c)(\mathsf{inl}(a)) &:\equiv i(a), \\ \mathsf{s}(c)(\mathsf{inr}(b)) &:\equiv j(b), \\ \mathsf{ap}_{\mathsf{s}(c)}(\mathsf{glue}(x)) &:= h(x) \end{aligned}$$

我们定义了一个映射

$$\begin{cases} \mathsf{cocone}_{\mathscr{D}}(E) & \longrightarrow & (A \sqcup^C B \to E) \\ c & \longmapsto & \mathsf{s}(c) \end{cases}$$

我们需要证明此映射是 $t \mapsto t \circ c_{\sqcup}$ 的逆映射。一方面，如果 $c = (i, j, h) : \mathsf{cocone}_{\mathscr{D}}(E)$，我们有

$$\begin{aligned} \mathsf{s}(c) \circ c_{\sqcup} &= (\mathsf{s}(c) \circ \mathsf{inl}, \mathsf{s}(c) \circ \mathsf{inr}, \mathsf{ap}_{\mathsf{s}(c)} \circ \mathsf{glue}) \\ &= (\lambda a.\, \mathsf{s}(c)(\mathsf{inl}(a)),\ \lambda b.\, \mathsf{s}(c)(\mathsf{inr}(b)),\ \lambda x.\, \mathsf{ap}_{\mathsf{s}(c)}(\mathsf{glue}(x))) \\ &= (\lambda a.\, i(a),\ \lambda b.\, j(b),\ \lambda x.\, h(x)) \\ &\equiv (i, j, h) \\ &= c \end{aligned}$$

另一方面，如果 $t : A \sqcup^C B \to E$，我们想证明 $\mathsf{s}(t \circ c_{\sqcup}) = t$。对于 $a : A$，我们有

$$\mathsf{s}(t \circ c_{\sqcup})(\mathsf{inl}(a)) = t(\mathsf{inl}(a))$$

因为 $t \circ c_{\sqcup}$ 的第一个分量是 $t \circ \mathsf{inl}$。以同样的方式，对于 $b : B$，我们有

$$\mathsf{s}(t \circ c_{\sqcup})(\mathsf{inr}(b)) = t(\mathsf{inr}(b))$$

对于 $x : C$，我们有

$$\mathsf{ap}_{\mathsf{s}(t \circ c_{\sqcup})}(\mathsf{glue}(x)) = \mathsf{ap}_t(\mathsf{glue}(x))$$

因此 $\mathsf{s}(t \circ c_{\sqcup}) = t$。
这证明了 $c \mapsto \mathsf{s}(c)$ 是 $t \mapsto t \circ c_{\sqcup}$ 的准逆映射，正如所需。 □

许多标准的同伦理论构造可以表示为（同伦）推挤。

- 跨图 $\mathbf{1} \leftarrow A \to \mathbf{1}$ 的推挤是 **悬挂** (suspension) $\Sigma A$（见 §6.5）。
- $A \xleftarrow{\mathsf{pr}_1} A \times B \xrightarrow{\mathsf{pr}_2} B$ 的推挤称为 $A$ 和 $B$ 的**连接** (join)，记为 $A * B$。
- $\mathbf{1} \leftarrow A \xrightarrow{f} B$ 的推挤称为 $f$ 的**锥体** (cone) 或**余纤维** (cofiber)。
- 如果 $A$ 和 $B$ 具有基点 (basepoints) $a_0 : A$ 和 $b_0 : B$，则 $A \xleftarrow{a_0} \mathbf{1} \xrightarrow{b_0} B$ 的推挤是**楔积** (wedge) $A \vee B$。
- 如果 $A$ 和 $B$ 是如前所述的有基点类型，则定义 $f : A \vee B \to A \times B$ 为 $f(\mathsf{inl}(a)) :\equiv (a, b_0)$ 和 $f(\mathsf{inr}(b)) :\equiv (a_0, b)$，其中 $f(\mathsf{glue}) := \mathsf{refl}_{(a_0, b_0)}$。那么 $f$ 的锥体称为**劈积** (smash product) $A \wedge B$。

我们将在 Chapters 7 and 8 中进一步讨论推挤。

Remark 6.8.3. 正如在 §3.7 中所述，对于有基点的空间，劈积和楔积的符号 $\wedge$ 和 $\vee$ 也用于逻辑中的“与 (and)”和“或 (or)”。由于同伦类型论中的类型既可以表现得像空间，也可以表现得像命题，因此存在冲突的潜在可能性——但由于它们很少同时这样做，通常上下文可以消除歧义。此外，劈积和楔积仅适用于有基点的空间，而唯一的有基点的纯命题是 $\top \equiv \mathbf{1}$——并且我们有 $\mathbf{1} \wedge \mathbf{1} = \mathbf{1}$ 和 $\mathbf{1} \vee \mathbf{1} = \mathbf{1}$ 对于 $\wedge$ 和 $\vee$ 的任何含义都成立。

Remark 6.8.4. 请注意，余极限通常不保留截断性 (truncatedness)。例如，$\mathbb{S}^0$ 和 $\mathbf{1}$ 都是集合 (sets)，但 $\mathbf{1} \leftarrow \mathbb{S}^0 \to \mathbf{1}$ 的推挤是 $\mathbb{S}^1$，它不是集合。因此，如果我们对 $n$-类型范畴 (category of $n$-types) 中的余极限感兴趣（特别是在集合的范畴中），我们需要以某种方式“截断”余极限。我们将在 §6.9 and Chapters 7 and 10 中回到这一点。

## 6.9 截断 (Truncations)

在 §3.7中，我们介绍了命题截断 (propositional truncation) 作为一种新的类型构造操作；现在我们发现它可以作为高阶归纳类型 (higher inductive types) 的一种特殊情况来获得。这将理解截断的问题简化为理解高阶归纳类型的问题，而高阶归纳类型至少是可以系统处理的。这个现象也很有趣，因为它为我们提供了第一个真正递归 (recursive) 的高阶归纳类型的例子，即它的构造器从正在定义的类型中获取输入（就像后继 $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$ 一样）。

设 $A$ 是一个类型；我们定义其命题截断 $\|A\|$ 为由以下构造器生成的高阶归纳类型：

- 一个函数 $|-| : A \to \|A\|$，以及
- 对于每个 $x, y : \|A\|$，一个路径 $x = y$。

请注意，第二个构造器实际上是断言 $\|A\|$ 是一个纯命题 (mere proposition)。因此，$\|A\|$ 的定义可以解释为 $\|A\|$ 是由 $A \to \|A\|$ 的函数和它是一个纯命题的事实自由生成的。

该高阶归纳定义的递归原理 (recursion principle) 很容易写出来：它表示给定任意类型 $B$，以及

- 一个函数 $g : A \to B$，以及
- 对于任意 $x, y : B$，一个路径 $x =_B y$，

存在一个函数 $f : \|A\| \to B$ 使得

- 对于所有 $a : A$，$f(|a|) \equiv g(a)$，以及
- 对于任意 $x, y : \|A\|$，函数 $\mathsf{ap}_f$ 将 $\|A\|$ 中指定的路径 $x = y$ 映射到 $B$ 中指定的路径 $f(x) = f(y)$（命题性地）。

这些正是我们在 §3.7中为命题截断的递归原理所陈述的假设——一个 $A \to B$ 的函数，且 $B$ 是一个纯命题——结论的第一部分正是我们在那里陈述的内容。结论的第二部分（$\mathsf{ap}_f$ 的作用）之前没有提到，但在这种情况下是空洞的，因为 $B$ 是一个纯命题，所以任何两条路径在其中都自动相等。

$\|A\|$ 还有一个归纳原理 (induction principle)，它表示给定任意 $B : \|A\| \to \mathcal{U}$ 以及

- 一个函数 $g : \prod_{(a:A)} B(|a|)$，以及
- 对于任意 $x, y : \|A\|$ 和 $u : B(x)$ 以及 $v : B(y)$，一个依赖路径 $q : u =^B_{p(x,y)} v$，其中 $p(x, y)$ 是从 $\|A\|$ 的第二个构造器中获得的路径，

存在 $f : \prod_{(x:\|A\|)} B(x)$ 使得 $f(|a|) \equiv g(a)$ 对于 $a : A$ 成立，并且还有另一个计算规则。然而，由于在任何两个纯命题之间至多可以存在一个函数（最多到同伦），这个归纳原理并没有太大用处（参见 Exercise 3.17）。

我们可以扩展这个想法来构造类似的截断，落在 $n$-类型中，对于任意 $n$。例如，我们可以定义 0-截断 (0-truncation) $\|A\|_0$ 为由以下内容生成的：

- 一个函数 $|-|_0 : A \to \|A\|_0$，以及

- 对于每个 $x,y : \|A\|_0$ 和每个 $p,q : x = y$，一个路径 $p = q$。

然后 $\|A\|_0$ 将由一个 $A \to \|A\|_0$ 的函数以及 $\|A\|_0$ 是一个集合 (set) 的断言自由生成。它的一个自然的归纳原理表示，给定 $B : \|A\|_0 \to \mathcal{U}$ 以及

- 一个函数 $g : \prod_{(a:A)} B(|a|_0)$，以及
- 对于任意 $x,y : \|A\|_0$，以及 $z : B(x)$ 和 $w : B(y)$，以及每个 $p,q : x = y$ 和 $r : z =^B_p w$ 以及 $s : z =^B_q w$，一个 $v : r =^{z=^B_- w}_{u(x,y,p,q)} s$ 的 2 路径，其中 $u(x,y,p,q) : p = q$ 是从 $\|A\|_0$ 的第二个构造器中获得的，

存在 $f : \prod_{(x:\|A\|_0)} B(x)$ 使得 $f(|a|_0) \equiv g(a)$ 对于所有 $a : A$ 成立，并且 $\mathsf{apd}^2_f(u(x,y,p,q))$ 是上面指定的 2 路径。（如同命题情况，后者的条件被证明是不太重要的。）然而，从这可以证明一个更有用的归纳原理。

Lemma 6.9.1. *假设给定 $B : \|A\|_0 \to \mathcal{U}$ 以及 $g : \prod_{(a:A)} B(|a|_0)$，并假设每个 $B(x)$ 是一个集合。然后存在 $f : \prod_{(x:\|A\|_0)} B(x)$ 使得 $f(|a|_0) \equiv g(a)$ 对于所有 $a : A$ 成立。*

*证明.* 只需为上述的任意 $x,y,z,w,p,q,r,s$ 构造一个 2 路径 $v : r =^B_{u(x,y,p,q)} s$ 即可。然而，根据依赖 2 路径的定义，这只是 $B(y)$ 中的一个普通 2 路径。由于 $B(y)$ 是一个集合，因此在任何两个平行路径之间都存在一个 2 路径。 □

这暗示了预期的通用性质。

Lemma 6.9.2. *对于任意集合 $B$ 和任意类型 $A$，与 $|-|_0 : A \to \|A\|_0$ 组合的合成决定了一个等价*

$$(\|A\|_0 \to B) \simeq (A \to B)$$

*证明.* 当 $B$ 是常数族时，Lemma 6.9.1 的特例提供了一个从右到左的映射，它是从左到右的“与 $|-|_0$ 组合”函数的右逆。要证明它也是左逆，设 $h : \|A\|_0 \to B$，并通过将 Lemma 6.9.1 应用于复合 $a \mapsto h(|a|_0)$ 来定义 $h' : \|A\|_0 \to B$。因此，对于任意 $a : A$，$h'(|a|_0) = h(|a|_0)$。
然而，由于 $B$ 是一个集合，对于任意 $x : \|A\|_0$，类型 $h(x) = h'(x)$ 是一个纯命题，因此也是一个集合。因此，依据 Lemma 6.9.1，观察到 $h'(|a|_0) = h(|a|_0)$ 对于任意 $a : A$ 表明对于任意 $x : \|A\|_0$，$h(x) = h'(x)$，因此 $h = h'$。 □

例如，这使我们能够构造集合的余极限。我们已经看到，如果 $A \xleftarrow{f} C \xrightarrow{g} B$ 是一组集合，则推挤 $A \sqcup^C B$ 可能不再是一个集合。（例如，如果 $A$ 和 $B$ 是 $\mathbf{1}$ 而 $C$ 是 $\mathbf{2}$，则推挤是 $\mathbb{S}^1$。）然而，我们可以通过截断构造一个集合的推挤，并具有与其他集合相关的预期通用性质。

Lemma 6.9.3. *设 $A \xleftarrow{f} C \xrightarrow{g} B$ 是一组集合。则对于任意集合 $E$，存在一个典型的等价*

$$\left(\left\|A \sqcup^C B\right\|_0 \to E\right) \simeq \mathsf{cocone}_{\mathscr{D}}(E)$$

*证明.* 结合 Lemmas 6.8.2 and 6.9.2 中的等价性。 □

我们将 $\left\|A \sqcup^C B\right\|_0$ 称为**集合推挤** (set-pushout)的 $f$ 和 $g$，以将其与（同伦）推挤 $A \sqcup^C B$ 区分开来。或者，我们可以直接修改 §6.8 中的推挤定义，以包括 0 截断构造器，从而避免事后截断。类似的备注也适用于任何类型的集合余极限；我们将在 Chapter 10 中进一步探讨这个问题。
然而，虽然上述的 0 截断定义是可行的——它给出了我们想要的结果，并且是一致的——它有几个问题。首先，它不太适合高阶归纳类型的一般理论。通常，直接处理像我们为 $\|A\|_0$ 给出的第二个构造器这样的构造器是很棘手的，因为它们的输入不仅涉及正在定义的类型中的元素，还涉及其中的路径。
然而，这个问题可以相当容易地解决。回想一下，在 §5.1 中我们提到，我们可以通过让构造器获取一个类型 $W$ 的归纳类型 $W$ 的“无限多的参数”，从而允许一个构造器接受无限多个 $W$ 类型的参数，

例如通过让它获取类型 $\mathbb{N} \to W$ 的单个参数。这背后有一个普遍的原则：为了模拟具有奇特输入的构造器，使用辅助归纳类型（如 $\mathbb{N}$ ）对它们进行参数化，将输入简化为具有归纳域的简单函数。对于 0 截断，我们可以考虑辅助的高阶归纳类型 $S$，它由两个点 $a, b : S$ 和两条路径 $p, q : a = b$ 生成。然后，$\|A\|_0$ 中看似奇怪的构造器可以被替换为无可非议的

- 对于每个 $f : S \to \|A\|_0$，一条路径 $\mathsf{ap}_f(p) = \mathsf{ap}_f(q)$。

因为给定 $S$ 的映射等同于给出两个点和它们之间的两条平行路径，这将产生相同的归纳原理。然而，我们当前的 0 截断定义的一个更严重的问题是，它不能很好地推广。如果我们想描述一个统一定义的"$n$-截断"到 $n$-类型中的概念，对于任意 $n : \mathbb{N}$，那么这种方法是不可行的，因为第二个构造器需要的参数数量随着 $n$ 的增加而增加。因此，在 §7.3 中，我们将使用一个不同的想法来构造这些，基于一个观察，即上述引入的类型 $S$ 等价于圆 $\mathbb{S}^1$。这包括了 0 截断作为一个特殊情况，并满足 Lemmas 6.9.1 and 6.9.2 的广义版本。

## 6.10 **商** (Quotients)

集合的一种特别重要的余极限是通过关系的商 (quotient)。也就是说，设 $A$ 是一个集合，$R : A \times A \to \mathsf{Prop}$ 是一族纯命题（**纯关系** (mere relation)）。它的商应该是以下两个投影的集合并等化子 (set-coequalizer)

$$\textstyle\sum_{(a,b:A)} R(a,b) \rightrightarrows A$$

我们也可以直接描述它，作为由以下内容生成的高阶归纳类型 $A/R$：

- 一个函数 $q : A \to A/R$；
- 对于每个 $a, b : A$，如果 $R(a,b)$，则有一个等式 $q(a) = q(b)$；以及
- 0 截断构造器：对于所有 $x, y : A/R$ 和 $r, s : x = y$，我们有 $r = s$。

我们有时会将这个高阶归纳类型 $A/R$ 称为 $A$ 关于 $R$ 的**集合商**，以强调它按定义生成了一个集合。(在同伦理论中有更一般的"商"概念，但它们大多超出了本书的范围。然而，在 §9.9 中，我们将考虑一个类型关于 1-群胚 (1-groupoid) 的"商"，这是集合商之上的下一个层次。)

Remark 6.10.1. 事实上，定义集合商时，并不需要 $A$ 是一个集合。尽管如此，这通常是最感兴趣的情况。

Lemma 6.10.2. *函数 $q : A \to A/R$ 是满射。*

*证明.* 我们必须证明，对于任意 $x : A/R$，存在一个 $a : A$ 使得 $q(a) = x$。我们使用 $A/R$ 的归纳原理。第一个情况是显然的：如果 $x$ 是 $q(a)$，那么当然存在一个 $a$，使得 $q(a) = q(a)$。由于目标是一个纯命题，它自动尊重所有路径构造器，因此我们完成了证明。 □

现在我们可以证明集合商具有预期的集合并等化子的通用性质。

Lemma 6.10.3. *对于任意集合 $B$，预合成 $q$ 得到一个等价*

$$(A/R \to B) \;\simeq\; \Big(\sum_{(f:A\to B)} \prod_{(a,b:A)} R(a,b) \to (f(a) = f(b))\Big)$$

*证明.* 从右到左的 $- \circ q$ 的准逆是 $A/R$ 的递归原理。也就是说，给定 $f : A \to B$，使得 $\prod_{(a,b:A)} R(a,b) \to (f(a) = f(b))$ 我们通过定义 $\bar{f} : A/R \to B$ 为 $\bar{f}(q(a)) :\equiv f(a)$。这个定义等式正好说明了 $(f \mapsto \bar{f})$ 是 $(- \circ q)$ 的右逆。
为了证明它也是左逆，我们必须证明，对于任意 $g : A/R \to B$ 和 $x : A/R$，我们有 $g(x) = \overline{g \circ q}(x)$。然而，根据 Lemma 6.10.2，仅存在一个 $a$ 使得 $q(a) = x$。由于我们想要的等式是一个纯命题，我们可以假设确实存在这样的 $a$，在这种情况下，$g(x) = g(q(a)) = \overline{g \circ q}(q(a)) = \overline{g \circ q}(x)$。 □

当然，经典情况下通常考虑 $R$ 是一个**等价关系** (equivalence relation)，即我们有

- **自反性** (reflexivity)：$\prod_{(a:A)} R(a,a)$，
- **对称性** (symmetry)：$\prod_{(a,b:A)} R(a,b) \to R(b,a)$，以及
- **传递性** (transitivity)：$\prod_{(a,b,c:C)} R(a,b) \times R(b,c) \to R(a,c)$。

在这种情况下，集合商 $A/R$ 具有额外的良好性质，正如我们将在 §10.1 中看到的：例如，我们有 $R(a,b) \simeq (q(a) =_{A/R} q(b))$。我们通常将等价关系 $R(a,b)$ 以中缀形式写为 $a \sim b$。
等价关系的商也可以通过其他方式构造。集合论的方法是将等价类视为幂集 $A$ 的子集。我们也可以在类型论中模仿这种“泛型 (impredicative)”构造。

Definition 6.10.4. 如果对于任意 $b : A$ 我们有 $R(a,b) \simeq P(b)$，则谓词 $P : A \to \mathsf{Prop}$ 是关系 $R : A \times A \to \mathsf{Prop}$ 的**等价类** (equivalence class)。

由于 $R$ 和 $P$ 是纯命题，等价 $R(a,b) \simeq P(b)$ 等同于蕴涵 $R(a,b) \to P(b)$ 和 $P(b) \to R(a,b)$。当然，对于任意 $a : A$，我们有规范的等价类 $P_a(b) :\equiv R(a,b)$。

Definition 6.10.5. 我们定义

$$A /\!\!/ R :\equiv \{\, P : A \to \mathsf{Prop} \mid P \text{ 是}R \text{ 的一个等价类} \,\}$$

函数 $q' : A \to A /\!\!/ R$ 定义为 $q'(a) :\equiv P_a$。

Theorem 6.10.6. *对于任意集合 $A$ 上的等价关系 $R$，类型 $A /\!\!/ R$ 与集合商 $A/R$ 等价。*

*证明.* 首先注意，如果 $R(a,b)$，则由于 $R$ 是等价关系，对于任意 $c : A$，我们有 $R(a,c) \Leftrightarrow R(b,c)$。因此，通过一值性 (univalence)，$R(a,c) = R(b,c)$，因此通过函数扩展性 (function extensionality)，$P_a = P_b$，即 $q'(a) = q'(b)$。因此，根据 Lemma 6.10.3，我们有一个从 $A/R$ 到 $A /\!\!/ R$ 的诱导映射 $f$，使得 $f \circ q = q'$。
我们证明 $f$ 是单射且满射，因此是一个等价。满射性直接来自 $q'$ 的满射性，后者本质上是由 $A /\!\!/ R$ 的定义决定的。对于单射性，如果 $f(x) = f(y)$，则为了证明纯命题 $x = y$，通过 $q$ 的满射性，我们可以假设 $x = q(a)$ 和 $y = q(b)$ 对于某些 $a, b : A$。然后，对于任意 $c : A$，$R(a,c) = f(q(a))(c) = f(q(b))(c) = R(b,c)$，特别地，$R(a,b) = R(b,b)$。但是 $R(b,b)$ 是有元素的，因为 $R$ 是等价关系，因此 $R(a,b)$ 也是有元素的。因此 $q(a) = q(b)$，因此 $x = y$。 □

在 §10.1.3 中，我们将给出该定理的另一种证明。注意，与 $A/R$ 不同，构造 $A /\!\!/ R$ 提升了宇宙层次：如果 $A : \mathcal{U}_i$ 且 $R : A \to A \to \mathsf{Prop}_{\mathcal{U}_i}$，则在 $A /\!\!/ R$ 的定义中，我们还必须使用 $\mathsf{Prop}_{\mathcal{U}_i}$ 来包含所有等价类，因此 $A /\!\!/ R : \mathcal{U}_{i+1}$。当然，如果我们假设来自 §3.5 的命题重缩放公理 (propositional resizing axiom)，我们可以避免这个问题。

Remark 6.10.7. 之前的两个构造提供了普遍的商，但在特殊情况下，可能存在更简单的构造。例如，我们可以定义整数 $\mathbb{Z}$ 作为集合商

$$\mathbb{Z} :\equiv (\mathbb{N} \times \mathbb{N})/\sim$$

其中 $\sim$ 是通过以下方式定义的等价关系：

$$(a,b) \sim (c,d) :\equiv (a + d = b + c)$$

换句话说，一个对 $(a,b)$ 代表整数 $a - b$。然而，在这种情况下存在*规范的代表* (canonical representatives)：即形如 $(n,0)$ 或 $(0,n)$ 的代表。

以下引理表明，当出现这种情况时，我们不需要任一一般商的构造。（一个函数 $r : A \to A$ 称为**幂等的** (idempotent)当且仅当 $r \circ r = r$。）

Lemma 6.10.8. 假设 $\sim$ 是集合 $A$ 上的关系，并且存在一个幂等 $r : A \to A$，使得对于所有 $x, y : A$，$(r(x) = r(y)) \simeq (x \sim y)$。（这意味着 $\sim$ 是一个等价关系。）那么类型

$$(A/\sim) :\equiv \left(\sum_{x:A} r(x) = x\right)$$

满足 $A$ 关于 $\sim$ 的集合商的通用性质，因此与之等价。换句话说，有一个映射 $q : A \to (A/\sim)$，使得对于每个集合 $B$，与 $q$ 的预合成诱导一个等价

$$\left((A/\sim) \to B\right) \simeq \left(\sum_{(g:A\to B)} \prod_{(x,y:A)} (x \sim y) \to (g(x) = g(y))\right) \tag{6.10.9}$$

证明. 设 $i : \prod_{(x:A)} r(r(x)) = r(x)$ 见证了 $r$ 的幂等性。映射 $q : A \to (A/\sim)$ 定义为 $q(x) :\equiv (r(x), i(x))$。注意，由于 $A$ 是一个集合，我们有 $q(x) = q(y)$ 当且仅当 $r(x) = r(y)$，因此（根据假设）当且仅当 $x \sim y$。我们通过以下方式定义 (6.10.9) 中从左到右的映射 $e$：

$$e(f) :\equiv (f \circ q, \_)$$

其中下划线 _ 表示以下证明：如果 $x, y : A$ 并且 $x \sim y$，则 $q(x) = q(y)$ 如上所述，因此 $f(q(x)) = f(q(y))$。为了证明 $e$ 是一个等价，考虑由以下方式定义的相反方向的映射 $e'$：

$$e'(g, s)(x, p) :\equiv g(x)$$

给定任意 $f : (A/\sim) \to B$，

$$e'(e(f))(x, p) \equiv f(q(x)) \equiv f(r(x), i(x)) = f(x, p)$$

因为最后的等式成立是因为 $p : r(x) = x$，因此 $(x, p) = (r(x), i(x))$ 因为 $A$ 是一个集合。类似地我们计算

$$e(e'(g, s)) \equiv e(g \circ \mathsf{pr}_1) \equiv (g \circ \mathsf{pr}_1 \circ q, \_)$$

因为 $B$ 是一个集合，我们不必担心 _ 部分，而对于第一个分量，我们有

$$g(\mathsf{pr}_1(q(x))) \equiv g(r(x)) = g(x)$$

其中最后一个等式成立是因为 $r(x) \sim x$，并且 $g$ 尊重 $s$ 所假设的 $\sim$。 □

Corollary 6.10.10. 假设 $p : A \to B$ 是一个集合之间的收缩映射 (retraction)。那么 $B$ 是 $A$ 关于等价关系 $\sim$ 的商，$\sim$ 定义为

$$(a_1 \sim a_2) :\equiv (p(a_1) = p(a_2))$$

证明. 假设 $s : B \to A$ 是 $p$ 的一个截面 (section)。然后 $s \circ p : A \to A$ 是一个满足 Lemma 6.10.8 的 $\sim$ 的幂等，并且 $s$ 诱导了 $B$ 到其固定点集的同构。 □

Remark 6.10.11. Lemma 6.10.8 适用于 $\mathbb{Z}$，使用定义的幂等 $r : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ 定义为

$$r(a, b) = \begin{cases} (a - b, 0) & \text{如果} a \geq b \\ (0, b - a) & \text{否则。} \end{cases}$$

（即使在构造上这个定义也是有效的，因为 $\mathbb{N}$ 上的关系 $\geq$ 是可判定的。）因此，一个非负整数被规范地表示为 $(k, 0)$，一个非正整数表示为 $(0, m)$，其中 $k, m : \mathbb{N}$。这个分案例表示了以下整数的“归纳原理 (induction principle)”。（如往常一样，我们将自然数 $n$ 识别为对应的非负整数，即 $(n, 0) : \mathbb{N} \times \mathbb{N}$ 在 $\mathbb{Z}$ 中的像。）

Lemma 6.10.12. 假设 $P : \mathbb{Z} \to \mathcal{U}$ 是一个类型族，并且我们有

- $d_0 : P(0)$,

- $d_+ : \prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}(n))$，以及
- $d_- : \prod_{(n:\mathbb{N})} P(-n) \to P(-\mathsf{succ}(n))$。

那么我们有 $f : \prod_{(z:\mathbb{Z})} P(z)$ 使得

- $f(0) = d_0$，
- 对于所有 $n : \mathbb{N}$，$f(\mathsf{succ}(n)) = d_+(n, f(n))$，以及
- 对于所有 $n : \mathbb{N}$，$f(-\mathsf{succ}(n)) = d_-(n, f(-n))$。

证明. 为了证明此引理，我们让 $\mathbb{Z}$ 表示 $\sum_{(x:\mathbb{N}\times\mathbb{N})}(r(x) = x)$，其中 $r$ 是上述的幂等。(然后我们可以将结果传递到与 $\mathbb{Z}$ 等价的任何定义。) 让 $q : \mathbb{N} \times \mathbb{N} \to \mathbb{Z}$ 是商映射，定义为 $q(x) = (r(x), i(x))$，如 Lemma 6.10.8 中所述。现在定义 $Q :\equiv P \circ q : \mathbb{N} \times \mathbb{N} \to \mathcal{U}$。通过跨越适当的等式传递给定的数据，我们得到

$$\begin{aligned} d'_0 &: Q(0,0) \\ d'_+ &: \prod_{n:\mathbb{N}} Q(n,0) \to Q(\mathsf{succ}(n),0) \\ d'_- &: \prod_{n:\mathbb{N}} Q(0,n) \to Q(0,\mathsf{succ}(n)) \end{aligned}$$

还注意，由于 $q(n,m) = q(\mathsf{succ}(n),\mathsf{succ}(m))$，我们有一个诱导等价

$$e_{n,m} : Q(n,m) \simeq Q(\mathsf{succ}(n),\mathsf{succ}(m))]$$

xg:$\prod_{(x:\mathbb{N}\times\mathbb{N})} Q(x)$:

$$\begin{aligned} g(0,0) &:\equiv d'_0 \\ g(\mathsf{succ}(n),0) &:\equiv d'_+(n, g(n,0)) \\ g(0,\mathsf{succ}(m)) &:\equiv d'_-(m, g(0,m)) \\ g(\mathsf{succ}(n),\mathsf{succ}(m)) &:\equiv e_{n,m}(g(n,m)) \end{aligned}$$

现在我们有 $\mathsf{pr}_1 : \mathbb{Z} \to \mathbb{N} \times \mathbb{N}$，其属性为 $q \circ \mathsf{pr}_1 = \mathsf{id}$。特别地，因此我们有 $Q \circ \mathsf{pr}_1 = P$，因此有一组等价 $s : \prod_{(z:\mathbb{Z})} Q(\mathsf{pr}_1(z)) \simeq P(z)$。因此，我们可以定义 $f(z) = s(z, g(\mathsf{pr}_1(z)))$ 以获得 $f : \prod_{(z:\mathbb{Z})} P(z)$，并验证所需的等式。 □

我们有时会用模式匹配语法表示从 Lemma 6.10.12 获得的函数 $f : \prod_{(z:\mathbb{Z})} P(z)$，涉及三种情况 $d_0$，$d_+$ 和 $d_-$：

$$\begin{aligned} f(0) &:= d_0 \\ f(\mathsf{succ}(n)) &:= d_+(n, f(n)) \\ f(-\mathsf{succ}(n)) &:= d_-(n, f(-n)) \end{aligned}$$

我们使用 $:=$ 而不是 $:\equiv$，就像我们对高阶归纳类型的路径构造器所做的那样，表示由 Lemma 6.10.12 隐含的“计算”规则只是命题性等式。例如，通过这种方式我们可以定义任意整数 $n$ 的 $n$ 倍连接环 (n-fold concatenation of a loop)。

Corollary 6.10.13. 设 $A$ 是一个具有 $a : A$ 和 $p : a = a$ 的类型。存在一个函数 $\prod_{(n:\mathbb{Z})}(a = a)$，记作 $n \mapsto p^n$，定义为

$$\begin{aligned} p^0 &:= \mathsf{refl}_a \\ p^{n+1} &:= p^n \cdot p && \text{对于 } n \geq 0 \\ p^{n-1} &:= p^n \cdot p^{-1} && \text{对于 } n \leq 0。 \end{aligned}$$

我们将在 §§6.11 and 11.1 中进一步讨论整数。

## 6.11 **代数** (Algebra)

除了构造球面和胞腔复形 (cell complexes) 这样的高维对象外，即使仅在集合上工作，高阶归纳类型 (higher inductive types) 也非常有用。我们在 Lemma 6.9.3 中已经看到了一个例子：它们允许我们构造任何集合图的余极限 (colimit)，而这是在 Chapter 1 的基础类型论中无法实现的。当我们研究具有代数结构的集合时，高阶归纳类型也非常有用。
在本节中，我们将以*群* (groups) 作为一个持续的例子，这对于大多数数学家来说是熟悉的，并展示了基本现象（在后续章节中也将需要它们）。然而，我们所说的大部分内容同样适用于任何形式的代数结构。

Definition 6.11.1. **幺半群** (monoid) 是一个集合 $G$ 以及

- 一个*乘法* (multiplication) 函数 $G \times G \to G$，写作中缀形式 $(x, y) \mapsto x \cdot y$；以及
- 一个*单位元* (unit) 元素 $e : G$；使得
- 对于任意 $x : G$，我们有 $x \cdot e = x$ 且 $e \cdot x = x$；以及
- 对于任意 $x, y, z : G$，我们有 $x \cdot (y \cdot z) = (x \cdot y) \cdot z$。

**群** (group) 是一个幺半群 $G$，并且

- 一个*逆元* (inversion) 函数 $i : G \to G$，写作 $x \mapsto x^{-1}$；使得
- 对于任意 $x : G$ 我们有 $x \cdot x^{-1} = e$ 且 $x^{-1} \cdot x = e$。

Remark 6.11.2. 注意，我们要求群是一个集合。我们可以考虑一种更一般的“∞-群 (∞-group)”它不一定是一个集合，但这会让我们偏离当前的讨论主题。在我们的定义中，我们可以期望所得到的“群论”表现得与集合论数学中的方式类似（除非我们假设 LEM ，否则它将是“构造性的”群论 (constructive group theory)）。

Example 6.11.3. 自然数 $\mathbb{N}$ 在加法下是一个幺半群，单位元是 0，在乘法下也是一个幺半群，单位元是 1。如果我们以显而易见的方式定义整数 $\mathbb{Z}$ 上的算术运算，那么通常它们在加法下是一个群，在乘法下是一个幺半群（当然，它们也是一个环）。例如，如果 $u, v \in \mathbb{Z}$ 分别表示为 $(a, b)$ 和 $(c, d)$，那么 $u + v$ 表示为 $(a + c, b + d)$，$-u$ 表示为 $(b, a)$，而 $uv$ 表示为 $(ac + bd, ad + bc)$。

Example 6.11.4. 我们在 §2.1 中实质上观察到，如果 $(A, a)$ 是一个尖点类型 (pointed type)，那么它的循环空间 (loop space) $\Omega(A, a) :\equiv (a =_A a)$ 具有群的所有结构，除了它通常不是一个集合。它应该是 Remark 6.11.2 中提到的“∞-群”，但我们也可以通过截断 (truncation) 使它成为一个群。具体地，我们定义 $A$ 在点 $a : A$ 处的 **基本群** (fundamental group) 为

$$\pi_1(A, a) :\equiv \|\Omega(A, a)\|_0$$

这继承了一个群结构；例如，通过对路径的连接进行双重截断，定义了乘法 $\pi_1(A, a) \times \pi_1(A, a) \to \pi_1(A, a)$。
更一般地，$(A, a)$ 的 $n$ **阶同伦群** (homotopy group) 定义为 $\pi_n(A, a) :\equiv \|\Omega^n(A, a)\|_0$。因此对于 $n \geq 1$，有 $\pi_n(A, a) = \pi_1(\Omega^{n-1}(A, a))$，所以它也是一个群。（当 $n = 0$ 时，我们有 $\pi_0(A) \equiv \|A\|_0$，它不是一个群。）此外，Eckmann-Hilton 论证 (Theorem 2.1.6) 表明，如果 $n \geq 2$，则 $\pi_n(A, a)$ 是一个*阿贝尔群* (abelian group)，即对于所有 $x, y$ 有 $x \cdot y = y \cdot x$。Chapter 8 将主要研究这些群。

群论中的一个重要概念是由一个集合生成的*自由群* (free group)，或更一般地，由生成元 (generators) 和关系 (relations) 表示的群。众所周知，在类型论中，一些自由代数对象可以使用*普通的*归纳类型 (inductive types) 定义。例如，集合 $A$ 上的自由幺半群可以与 $A$ 元素的*有限列表* (finite lists) 类型 $\mathsf{List}(A)$ 进行标识，后者通过归纳生成：

- 构造函数 $\mathsf{nil} : \mathsf{List}(A)$，以及
- 对于每个 $\ell : \mathsf{List}(A)$ 和 $a : A$，生成一个元素 $\mathsf{cons}(a, \ell) : \mathsf{List}(A)$。

我们有一个显然的包含 $\eta : A \to \mathsf{List}(A)$，定义为 $a \mapsto \mathsf{cons}(a, \mathsf{nil})$。$\mathsf{List}(A)$ 上的幺半群运算是连接 (concatenation)，递归定义为

$$\mathsf{nil} \cdot \ell :\equiv \ell$$
$$\mathsf{cons}(a, \ell_1) \cdot \ell_2 :\equiv \mathsf{cons}(a, \ell_1 \cdot \ell_2)$$

使用 $\mathsf{List}(A)$ 的归纳原理可以很容易地证明 $\mathsf{List}(A)$ 是一个集合，并且列表的连接是结合的 (associative) 并且 $\mathsf{nil}$ 作为单位元。因此，$\mathsf{List}(A)$ 是一个幺半群。

Lemma 6.11.5. *对于任意集合 $A$，类型 $\mathsf{List}(A)$ 是 $A$ 上的自由幺半群。换句话说，对于任意幺半群 $G$，与 $\eta$ 的合成是一个等价性*

$$\hom_{\mathrm{Monoid}}(\mathsf{List}(A), G) \simeq (A \to G),$$

*其中 $\hom_{\mathrm{Monoid}}(-,-)$ 表示幺半群同态 (monoid homomorphisms) 的集合（即保持乘法和单位元的函数）。*

*证明.* 给定 $f : A \to G$，我们通过递归定义 $\bar{f} : \mathsf{List}(A) \to G$：

$$\bar{f}(\mathsf{nil}) :\equiv e$$
$$\bar{f}(\mathsf{cons}(a, \ell)) :\equiv f(a) \cdot \bar{f}(\ell)$$

通过归纳可以很容易地证明 $\bar{f}$ 是一个幺半群同态，并且 $f \mapsto \bar{f}$ 是 $(- \circ \eta)$ 的一个拟逆；参见 Exercise 6.8。 □

这种自由幺半群的构造之所以可能，基本上是因为自由幺半群的元素具有可计算的标准形式（即有限列表）。然而，其他自由（和表示的）代数结构（例如群）的元素通常没有可计算的标准形式。例如，群表示中的词的相等性是算法上不可判定的 (undecidable)。然而，我们仍然可以通过简单地将所有公理化的等式作为路径构造来描述自由代数对象为高阶归纳类型。
例如，设 $A$ 是一个集合，定义一个具有以下生成元的高阶归纳类型 $F(A)$：

- 一个函数 $\eta : A \to F(A)$。
- 一个函数 $m : F(A) \times F(A) \to F(A)$。
- 一个元素 $e : F(A)$。
- 一个函数 $i : F(A) \to F(A)$。
- 对于每个 $x, y, z : F(A)$，有一个等式 $m(x, m(y, z)) = m(m(x, y), z)$。
- 对于每个 $x : F(A)$，有等式 $m(x, e) = x$ 和 $m(e, x) = x$。
- 对于每个 $x : F(A)$，有等式 $m(x, i(x)) = e$ 和 $m(i(x), x) = e$。
- 0-截断构造器：对于任意 $x, y : F(A)$ 和 $p, q : x = y$，我们有 $p = q$。

第一个构造器表示 $A$ 映射到 $F(A)$。接下来的三个给出了 $F(A)$ 的群运算：乘法、单位元和逆元。再往下的三个构造器断言了群的公理：结合律 (associativity)、单位律 (unitality) 和逆元律 (inverses)。最后，最后一个构造器断言 $F(A)$ 是一个集合。
因此，$F(A)$ 是一个群。也可以很容易地证明：

Theorem 6.11.6. *$F(A)$ 是 $A$ 上的自由群。换句话说，对于任意（集合）群 $G$，与 $\eta : A \to F(A)$ 的合成决定了一个等价性*

$$\hom_{\mathrm{Group}}(F(A), G) \simeq (A \to G)$$

*其中 $\hom_{\mathrm{Group}}(-,-)$ 表示两个群之间的群同态 (group homomorphisms) 的集合。*

*证明.* 高阶归纳类型 $F(A)$ 的递归原理精确地说明，如果 $G$ 是一个群并且我们有 $f : A \to G$，那么我们有 $\bar{f} : F(A) \to G$。它的计算规则说明 $\bar{f} \circ \eta \equiv f$，并且 $\bar{f}$ 是一个群同态。因此，$(- \circ \eta) : \hom_{\mathrm{Group}}(F(A), G) \to (A \to G)$ 有一个右逆。使用 $F(A)$ 的归纳原理很容易证明这也是一个左逆。 □

我们不妨退一步考虑我们刚刚完成的工作。我们已经证明，对于任意集合上的自由群的存在不需要给出一个明确的构造。基本上，我们只需要写下它应该满足的普遍性质即可。在集合论中，我们可以通过调用诸如伴随函子定理 (adjoint functor theorem) 之类的黑箱来实现类似的结果；类型论将这样的构造内置于数学基础中。
当然，有时也有一个具体的自由代数结构描述是有用的。在自由群的情况下，我们可以使用商来提供一个。例如，考虑 $\mathsf{List}(A+A)$，在 $A+A$ 中，我们写 $\mathsf{inl}(a)$ 为 $a$，并写 $\mathsf{inr}(a)$ 为 $\hat{a}$（表示 $a$ 的形式逆元）。$\mathsf{List}(A+A)$ 的元素是 $A$ 上的自由群的词 (words)。

Theorem 6.11.7. 设 $A$ 是一个集合，设 $F'(A)$ 是 $\mathsf{List}(A+A)$ 的按以下关系进行的集合商：

$$(\dots, a_1, a_2, \widehat{a_2}, a_3, \dots) = (\dots, a_1, a_3, \dots)$$
$$(\dots, a_1, \widehat{a_2}, a_2, a_3, \dots) = (\dots, a_1, a_3, \dots)$$

那么 $F'(A)$ 也是集合 $A$ 上的自由群。

证明. 首先，我们证明 $F'(A)$ 是一个群。我们已经看到 $\mathsf{List}(A+A)$ 是一个幺半群；我们声称幺半群结构可以下降到商。我们通过双重商递归定义 $F'(A) \times F'(A) \to F'(A)$；只需检查生成的等价关系是否在列表的连接中得到保持。同样，我们通过商归纳来证明结合性和单位律。
为了在 $F'(A)$ 中定义逆元，我们首先通过列表递归定义 $\mathsf{reverse} : \mathsf{List}(B) \to \mathsf{List}(B)$：

$$\mathsf{reverse}(\mathsf{nil}) :\equiv \mathsf{nil},$$
$$\mathsf{reverse}(\mathsf{cons}(b, \ell)) :\equiv \mathsf{reverse}(\ell) \cdot \mathsf{cons}(b, \mathsf{nil})$$

现在我们通过商递归定义 $i : F'(A) \to F'(A)$，作用在列表 $\ell : \mathsf{List}(A+A)$ 上，通过交换 $A$ 的两个副本并反转列表。这样可以保持关系，因此可以下降到商中。通过归纳可以证明，对于 $x : F'(A)$，我们有 $i(x) \cdot x = e$。首先，通过商归纳我们可以假设 $x$ 来自 $\ell : \mathsf{List}(A+A)$，然后我们可以进行列表归纳；如果我们写 $q : \mathsf{List}(A+A) \to F'(A)$ 表示商映射，情况如下：

$$\begin{aligned}
i(q(\mathsf{nil})) \cdot q(\mathsf{nil}) &= q(\mathsf{nil}) \cdot q(\mathsf{nil}) \\
&= q(\mathsf{nil}) \\
i(q(\mathsf{cons}(a, \ell))) \cdot q(\mathsf{cons}(a, \ell)) &= i(q(\ell)) \cdot q(\mathsf{cons}(\hat{a}, \mathsf{nil})) \cdot q(\mathsf{cons}(a, \ell)) \\
&= i(q(\ell)) \cdot q(\mathsf{cons}(\hat{a}, \mathsf{cons}(a, \ell))) \\
&= i(q(\ell)) \cdot q(\ell) \\
&= q(\mathsf{nil}) && \text{(通过归纳假设)}
\end{aligned}$$

（我们省略了一些相当明显的关于列表连接等行为的引理。）
这就完成了对 $F'(A)$ 是一个群的证明。现在，如果 $G$ 是一个具有函数 $f : A \to G$ 的群，我们可以定义 $A+A \to G$ 为 $f$ 在 $A$ 的第一个副本上的作用，$f$ 在 $G$ 的逆映射上的作用。现在，$G$ 是一个幺半群的事实产生了一个幺半群同态 $\mathsf{List}(A+A) \to G$。并且由于 $G$ 是一个群，这个映射保持了关系，因此下降为一个从 $F'(A)$ 到 $G$ 的映射。可以很容易地证明这是一个群同态，并且是唯一一个在 $A$ 上限制为 $f$ 的同态。 □

如果 $A$ 具有可判定相等性（例如，如果我们假设排中律 (excluded middle)），那么通过 Lemma 6.10.8 中的幺等映射可以获得定义 $F'(A)$ 的商。我们将一个词定义为**简化** (reduced)，如果它不包含形如 $(a, \hat{a})$ 或 $(\hat{a}, a)$ 的相邻对。当 $A$ 具有可判定相等性时，很容易定义词的**简化** (reduction)，这是生成适当商的一个幺等映射；我们将细节留给读者。
如果 $A :\equiv \mathbf{1}$，其具有可判定相等性，那么一个简化词必须完全由 $\star$ 组成或完全由 $\hat{\star}$ 组成。因此，$\mathbf{1}$ 上的自由群等价于整数 $\mathbb{Z}$，其中 0 对应于 $\mathsf{nil}$，正整数 $n$ 对应于 $n$ 个 $\star$ 的简化词，负整数 $(-n)$ 对应于 $n$ 个 $\hat{\star}$ 的简化词。当然，也可以直接证明 $\mathbb{Z}$ 具有 $F(\mathbf{1})$ 的普遍性质。

Remark 6.11.8. 在 $F(A)$ 和 $F'(A)$ 的构造及其普遍性质的证明中，我们从未使用 $A$ 是一个集合的假设。因此，我们实际上可以构造任意类型上的自由群。比较普遍性质，我们得出 $F(A) \simeq F(\|A\|_0)$。

我们还可以使用高阶归纳类型来构造代数对象的余极限。例如，假设 $f : G \to H$ 和 $g : G \to K$ 是群同态 (group homomorphisms)。在群范畴中的它们的推积 (pushout)，称为**并合自由积** (amalgamated free product)$H *_G K$，可以构造为由以下部分生成的高阶归纳类型：

- 函数 $h : H \to H *_G K$ 和 $k : K \to H *_G K$。
- 群的运算和公理，如在 $F(A)$ 的定义中。
- 断言 $h$ 和 $k$ 是群同态的公理。
- 对于 $x : G$，我们有 $h(f(x)) = k(g(x))$。
- 0-截断构造器。

另一方面，它也可以显式地构造为 $\mathsf{List}(H + K)$ 的集合商，遵循以下关系：

$$\begin{aligned}
(\dots, x_1, x_2, \dots) &= (\dots, x_1 \cdot x_2, \dots) && \text{对于 } x_1, x_2 : H \\
(\dots, y_1, y_2, \dots) &= (\dots, y_1 \cdot y_2, \dots) && \text{对于 } y_1, y_2 : K \\
(\dots, 1_G, \dots) &= (\dots, \dots) \\
(\dots, 1_H, \dots) &= (\dots, \dots) \\
(\dots, f(x), \dots) &= (\dots, g(x), \dots) && \text{对于 } x : G。
\end{aligned}$$

我们将证明留给读者。在 $G$ 是平凡群 (trivial group) 的特殊情况下，最后一个关系是不必要的，我们得到**自由积** (free product)$H * K$，这是群范畴中的余积 (coproduct)。（不幸的是，这个符号与类型的*连接* (join) 符号冲突，如 §6.8，但上下文通常可以消除歧义。）
注意，通过*表示* (presentations) 定义的群可以看作是余极限的一个特例。假设给定一个集合（或更一般的一个类型）$A$ 和一对函数 $R \rightrightarrows F(A)$。我们将 $R$ 看作是“关系”的类型，两函数为每个关系分配了两个相等的词。例如，在表示 $\langle a \mid a^2 = e \rangle$ 中，我们有 $A :\equiv \mathbf{1}$ 和 $R :\equiv \mathbf{1}$，两个态射 $R \rightrightarrows F(A)$ 分别选取列表 $(a, a)$ 和空列表 $\mathsf{nil}$。然后通过自由群的普遍性质，我们得到一对群同态 $F(R) \rightrightarrows F(A)$。它们在群范畴中的等距 (coequalizer)，可以像推积一样构造，即是由此表示所*表示*的群。

注意，所有这些类型的构造仅适用于*代数理论* (algebraic theories)，即其公理是（全称量化的）等式，涉及给定签名中的变量、常量和运算符。它们也可以修改为适用于所谓的*本质代数理论* (essentially algebraic theories)：这些理论中的操作部分地定义在由先前操作之间的等式指定的域上。它们不适用于例如域 (fields) 的理论，其中“逆运算”部分地定义在由先前操作之间的*不相容性* (apartness) $\#$ 指定的域 $\{\, x \mid x \# 0 \,\}$ 上，参见 Theorem 11.2.4。事实上，众所周知，域的范畴没有初对象 (initial object)。

另一方面，这些构造同样适用于*无穷*代数理论 (infinitary algebraic theories)，其“运算”可以接受无限多个输入。在这种情况下，除非我们假设选择公理 (axiom of choice)，否则可能不存在作为简单商的自由代数或代数余极限的表示。这意味着高阶归纳类型在某些方面显著加强了构造型类型论（不一定是在证明论强度方面，而是在实践能力方面），事实上在某些方面比 Zermelo–Fraenkel 集合论 (Zermelo–Fraenkel set theory)（没有选择公理）更强 [?]。

## 6.12 **展平引理** (The flattening lemma)

正如我们将在 Chapter 8 中看到的，当我们将高阶归纳类型 (higher inductive types) 与单值性公理 (univalence) 结合时，惊人的事情就会发生。这种现象主要来自于以下事实：如果 $W$ 是一个高阶归纳类型，而 $\mathcal{U}$ 是一个类型宇宙 (type universe)，那么我们可以通过使用 $W$ 的递归原理定义一个类型族 $P : W \to \mathcal{U}$。当我们处理 $W$ 的路径构造器 (path constructors) 所对应的递归原理条款时，我们需要提供在 $\mathcal{U}$ 中的路径，而这正是单值性公理起作用的地方。
例如，假设我们有一个类型 $X$ 和一个自等价 (self-equivalence) $e : X \simeq X$。那么我们可以使用 $\mathbb{S}^1$-递归定义一个类型族 $P : \mathbb{S}^1 \to \mathcal{U}$：

$$P(\mathsf{base}) :\equiv X \qquad \text{and} \qquad P(\mathsf{loop}) := \mathsf{ua}(e)$$

因此，类型 $X$ 作为 $P$ 在基点 (basepoint) 上的纤维 $P(\mathsf{base})$ 出现。自等价 $e$ 在 $P$ 中隐藏得稍微深一些，但下面的引理表明，它可以通过沿着 $\mathsf{loop}$ 传输来提取。

Lemma 6.12.1. *给定 $B : A \to \mathcal{U}$ 和 $x, y : A$，存在一个路径 $p : x = y$ 以及一个等价性 $e : B(x) \simeq B(y)$ 使得 $B(p) = \mathsf{ua}(e)$，那么对于任何 $u : B(x)$，我们有*

$$\mathsf{transport}^B(p, u) = e(u)$$

*证明.* 应用 ??，我们有

$$\begin{aligned}\mathsf{transport}^B(p, u) &= \mathsf{idtoeqv}(B(p))(u)\\ &= \mathsf{idtoeqv}(\mathsf{ua}(e))(u)\\ &= e(u)\end{aligned}$$

□

我们之前已经看到过由递归定义的类型族：在 §2.8 and ?? 中，我们用它们来表征（普通）归纳类型的恒等类型。在 Chapter 8 中，我们将使用类似的思想来计算高阶归纳类型的同伦群 (homotopy groups)。

在本节中，我们描述了一个关于这种类型族的一般性引理，它将在后面有用。我们称之为**展平引理** (flattening lemma)：它指出，如果 $P : W \to \mathcal{U}$ 是如上所述递归定义的，那么它的总空间 $\sum_{(x:W)} P(x)$ 等价于一个“展平”的高阶归纳类型，其构造可以从 $W$ 的构造器和 $P$ 的定义中推导出来。（从范畴论的角度来看，$\sum_{(x:W)} P(x)$ 是 $P$ 的“Grothendieck 构造”，而展平引理表达了它作为“松弛 余极限 (colimit)”的普遍性质。由于同伦类型论中的类型（如 $W$）在范畴上对应于 $\infty$-群体（因为所有路径都是可逆的），在这种情况下，松弛余极限与伪余极限相同。）

我们在此证明展平引理的一个一般情况，它直接推导出许多特殊情况，并暗示了证明其他情况的方法。假设我们有 $A, B : \mathcal{U}$ 和 $f, g : B \to A$，并且高阶归纳类型 $W$ 由以下部分生成：

- $\mathsf{c} : A \to W$ 和
- $\mathsf{p} : \prod_{(b:B)} (\mathsf{c}(f(b)) =_W \mathsf{c}(g(b)))$。

因此，$W$ 是 $f$ 和 $g$ 的 **（同伦）余等距** (coequalizer)

- 圆圈 $\mathbb{S}^1$ 可以通过取 $A :\equiv \mathbf{1}$ 和 $B :\equiv \mathbf{1}$ 表示，$f$ 和 $g$ 为恒等映射。
- $j : X \to Y$ 和 $k : X \to Z$ 的推积 (pushout) 可以通过取 $A :\equiv Y + Z$ 和 $B :\equiv X$ 表示，$f :\equiv \mathsf{inl} \circ j$ 和 $g :\equiv \mathsf{inr} \circ k$。

现在假设还有

- $C : A \to \mathcal{U}$ 是一个类型族，并且
- $D : \prod_{(b:B)} C(f(b)) \simeq C(g(b))$ 是一个定义在 $B$ 上的等价性族。

递归地定义一个类型族 $P : W \to \mathcal{U}$ 如下：

$$\begin{aligned}P(\mathsf{c}(a)) &:\equiv C(a)\\ P(\mathsf{p}(b)) &:= \mathsf{ua}(D(b))\end{aligned}$$

设 $\widetilde{W}$ 是由以下部分生成的高阶归纳类型：

- $\widetilde{\mathsf{c}} : \prod_{(a:A)} C(a) \to \widetilde{W}$ 和
- $\widetilde{\mathsf{p}} : \prod_{(b:B)} \prod_{(y:C(f(b)))} (\widetilde{\mathsf{c}}(f(b), y) =_{\widetilde{W}} \widetilde{\mathsf{c}}(g(b), D(b)(y)))$。

展平引理是：

Lemma 6.12.2 (展平引理 (Flattening lemma)). *在上述情况下，我们有*

$$\Big(\sum_{x:W} P(x)\Big) \simeq \widetilde{W}$$

如上所述，这个等价性可以看作是 $\sum_{(x:W)} P(x)$ 作为 $P$ 在 $W$ 上的“松弛 余极限”的普遍性质的表达。它也可以看作是余极限的稳定性与下降 (stability and descent) 性质的一部分，这表征了高阶拓扑斯的性质。
Lemma 6.12.2 的证明占据了本节的其余部分。它有点技术性，初读时可以跳过。但是它也是“证据相关数学 (proof-relevant mathematics)”的一个很好的例子，所以我们建议在第二次阅读时重视它。
这个想法是证明 $\sum_{(x:W)} P(x)$ 具有与 $\widetilde{W}$ 相同的普遍性质。我们首先展示它具有类似于构造器 $\tilde{\mathsf{c}}$ 和 $\tilde{\mathsf{p}}$ 的构造。

Lemma 6.12.3. *存在如下函数*

- $\tilde{c}' : \prod_{(a:A)} C(a) \to \sum_{(x:W)} P(x)$ *和*
- $\tilde{p}' : \prod_{(b:B)} \prod_{(y:C(f(b)))} \left(\tilde{c}'(f(b), y) =_{\sum_{(w:W)} P(w)} \tilde{c}'(g(b), D(b)(y))\right)$。

*证明.* 第一个很简单；定义 $\tilde{c}'(a, x) :\equiv (\mathsf{c}(a), x)$ 并注意到根据定义 $P(\mathsf{c}(a)) \equiv C(a)$。对于第二个，假设给定 $b : B$ 和 $y : C(f(b))$；我们必须给出一个等式

$$(\mathsf{c}(f(b)), y) = (\mathsf{c}(g(b)), D(b)(y))$$

由于我们有 $\mathsf{p}(b) : \mathsf{c}(f(b)) = \mathsf{c}(g(b))$，根据 $\Sigma$-类型的等式，给出一个等式 $\mathsf{p}(b)_*(y) = D(b)(y)$ 即可。但是这由 Lemma 6.12.1 和 $P$ 的定义得出。 □

下面的引理表明，为了定义 $\sum_{(w:W)} P(w)$ 上类型族的截面，只需要给出类似于 $\widetilde{W}$ 情况的数据。

Lemma 6.12.4. *假设* $Q : \left(\sum_{(x:W)} P(x)\right) \to \mathcal{U}$ *是一个类型族，并且我们有*

- $c : \prod_{(a:A)} \prod_{(x:C(a))} Q(\tilde{c}'(a, x))$ *和*
- $p : \prod_{(b:B)} \prod_{(y:C(f(b)))} \left(\tilde{p}'(b, y)_*(c(f(b), y)) = c(g(b), D(b)(y))\right)$。

*那么存在* $k : \prod_{(z:\sum_{(w:W)} P(w))} Q(z)$ *使得* $k(\tilde{c}'(a, x)) \equiv c(a, x)$。

*证明.* 假设给定 $w : W$ 和 $x : P(w)$；我们必须生成一个元素 $k(w, x) : Q(w, x)$。通过对 $w$ 进行归纳，考虑两种情况就足够了。当 $w \equiv \mathsf{c}(a)$ 时，我们有 $x : C(a)$，因此 $c(a, x) : Q(\mathsf{c}(a), x)$ 如预期。（这个定义的部分还确保了所述的计算规则成立。）
现在我们必须证明这个定义是通过 $b : B$ 的 $\mathsf{p}(b)$ 传输保持的。由于我们为所有 $w : W$ 定义的是一个类型为 $\prod_{(x:P(w))} Q(w, x)$ 的函数，根据 Lemma 2.7.7，证明对于任何 $y : C(f(b))$，我们有

$$\mathsf{transport}^Q(\mathsf{pair}^=(\mathsf{p}(b), \mathsf{refl}_{\mathsf{p}(b)_*(y)}), c(f(b), y)) = c(g(b), \mathsf{p}(b)_*(y))$$

设 $q : \mathsf{p}(b)_*(y) = D(b)(y)$ 是从 Lemma 6.12.1 获得的路径。然后我们有

$$\begin{aligned} c(g(b), \mathsf{p}(b)_*(y)) &= \mathsf{transport}^{x \mapsto Q(\tilde{c}'(g(b), x))}(q^{-1}, c(g(b), D(b)(y))) && (\text{通过 } \mathsf{apd}_{x \mapsto c(g(b), x)}(q^{-1})^{-1}) \\ &= \mathsf{transport}^Q(\mathsf{ap}_{x \mapsto \tilde{c}'(g(b), x)}(q^{-1}), c(g(b), D(b)(y))) && (\text{通过 Lemma 2.3.10}) \end{aligned}$$

因此，只需证明

$$\mathsf{transport}^Q\left(\mathsf{pair}^=(\mathsf{p}(b), \mathsf{refl}_{\mathsf{p}(b)_*(y)}),\ c(f(b), y)\right) = \mathsf{transport}^Q\left(\mathsf{ap}_{x \mapsto \tilde{c}'(g(b), x)}(q^{-1}),\ c(g(b), D(b)(y))\right)$$

将右侧传输移动到另一侧，并结合两个传输，这等价于

$$\mathsf{transport}^Q\left(\mathsf{pair}^=(\mathsf{p}(b), \mathsf{refl}_{\mathsf{p}(b)_*(y)}) \cdot \mathsf{ap}_{x \mapsto \tilde{c}'(g(b), x)}(q),\ c(f(b), y)\right) = c(g(b), D(b)(y))$$

但是，我们有

$$\mathsf{pair}^{=}(\mathsf{p}(b), \mathsf{refl}_{\mathsf{p}(b)_*(y)}) \centerdot \mathsf{ap}_{x\mapsto \widetilde{\mathsf{c}}'(g(b),x)}(q) = \\ \mathsf{pair}^{=}(\mathsf{p}(b), \mathsf{refl}_{\mathsf{p}(b)_*(y)}) \centerdot \mathsf{pair}^{=}(\mathsf{refl}_{\mathsf{c}(g(b))}, q) = \mathsf{pair}^{=}(\mathsf{p}(b), q) = \widetilde{\mathsf{p}}'(b,y)$$

因此，通过 $p(b,y)$ 的假设，构造完成了，即

$$\mathsf{transport}^{Q}(\widetilde{\mathsf{p}}'(b,y), c(f(b),y)) = c(g(b), D(b)(y)) \qquad \square$$

Lemma 6.12.4 几乎给出了 $\sum_{(w:W)} P(w)$ 与 $\widetilde{W}$ 相同的归纳原理。缺少的一点是等式 $\mathsf{apd}_k(\widetilde{\mathsf{p}}'(b,y)) = p(b,y)$。为了证明这一点，我们需要分析 Lemma 6.12.4 的证明，这当然是 $k$ 的定义。
应该可以做到这一点，但事实证明，我们只需要非依赖递归原理的计算规则。因此，我们现在给出递归函数的一个稍微简单的直接构造，并证明它的计算规则。

Lemma 6.12.5. *假设 $Q$ 是一个类型，并且我们有*

- $c : \prod_{(a:A)} C(a) \to Q$ *和*
- $p : \prod_{(b:B)} \prod_{(y:C(f(b)))} \Big(c(f(b),y) =_Q c(g(b), D(b)(y))\Big)$。

*那么存在 $k : \big(\sum_{(w:W)} P(w)\big) \to Q$ 使得 $k(\widetilde{\mathsf{c}}'(a,x)) \equiv c(a,x)$。*

*证明.* 如 Lemma 6.12.4 所示，我们通过对 $w : W$ 进行归纳来定义 $k(w,x)$。当 $w \equiv \mathsf{c}(a)$ 时，我们定义 $k(\mathsf{c}(a),x) :\equiv c(a,x)$。现在通过 Lemma 2.7.6，我们只需考虑 $b : B$ 和 $y : C(f(b))$ 的组合路径

$$\mathsf{transport}^{x\mapsto Q}(\mathsf{p}(b), c(f(b),y)) = c(g(b), \mathsf{transport}^{P}(\mathsf{p}(b), y)) \tag{6.12.6}$$

其定义为组合

$$\begin{aligned} \mathsf{transport}^{x\mapsto Q}(\mathsf{p}(b), c(f(b),y)) &= c(f(b),y) && \text{(通过 Lemma 2.3.5)} \\ &= c(g(b), D(b)(y)) && \text{(通过 } p(b,y)\text{)} \\ &= c(g(b), \mathsf{transport}^{P}(\mathsf{p}(b), y)) && \text{(通过 Lemma 6.12.1)} \end{aligned}$$

计算规则 $k(\widetilde{\mathsf{c}}'(a,x)) \equiv c(a,x)$ 如前所述，由定义得出。 $\square$

对于第二个计算规则，我们需要以下引理。

Lemma 6.12.7. *设 $Y : X \to \mathcal{U}$ 为一个类型族，$k : (\sum_{(x:X)} Y(x)) \to Z$ 是通过逐个分量定义的，即 $k(x,y) :\equiv d(x)(y)$ 对于一个柯里化函数 (curried function) $d : \prod_{(x:X)} Y(x) \to Z$。那么对于任何 $s : x_1 =_X x_2$ 和任何 $y_1 : Y(x_1)$ 及 $y_2 : Y(x_2)$，并且有一个路径 $r : s_*(y_1) = y_2$，路径*

$$\mathsf{ap}_k(\mathsf{pair}^{=}(s,r)) : k(x_1,y_1) = k(x_2,y_2)$$

*等于组合路径*

$$\begin{aligned} k(x_1,y_1) &\equiv d(x_1)(y_1) \\ &= \mathsf{transport}^{x\mapsto Z}(s, d(x_1)(y_1)) && \text{(通过 (Lemma 2.3.5)}^{-1}\text{)} \\ &= \mathsf{transport}^{x\mapsto Z}(s, d(x_1)({s^{-1}}_*(s_*(y_1)))) \\ &= \big(\mathsf{transport}^{x\mapsto (Y(x)\to Z)}(s, d(x_1))\big)(s_*(y_1)) && \text{(通过 (2.7.4))} \\ &= d(x_2)(s_*(y_1)) && \text{(通过 } \mathsf{happly}(\mathsf{apd}_d(s))(s_*(y_1))\text{)} \\ &= d(x_2)(y_2) && \text{(通过 } \mathsf{ap}_{d(x_2)}(r)\text{)} \\ &\equiv k(x_2,y_2) \end{aligned}$$

*证明.* 在路径归纳 (path induction) $s$ 和 $r$ 之后，两个等式都简化为反身性。 $\square$

乍一看，Lemma 6.12.7 的表述似乎很复杂，而它的证明却如此简单。复杂性的原因在于确保表述是良类型化的：$\mathsf{ap}_k(\mathsf{pair}^{=}(s,r))$ 和它声称等同的组合路径必须有相同的起点和终点。一旦我们做到了这一点，证明就很容易通过路径归纳得到。

Lemma 6.12.8. *在 Lemma 6.12.5 的情况下，我们有* $\mathsf{ap}_k(\widetilde{\mathsf{p}}'(b,y)) = p(b,y)$。

*证明.* 回想一下 $\widetilde{\mathsf{p}}'(b,y) :\equiv \mathsf{pair}^{=}(\mathsf{p}(b),q)$，其中 $q : \mathsf{p}(b)_*(y) = D(b)(y)$ 来自 Lemma 6.12.1。因此，由于 $k$ 是逐个分量定义的，我们可以通过 Lemma 6.12.7 计算 $\mathsf{ap}_k(\widetilde{\mathsf{p}}'(b,y))$，其定义如下：

$$\begin{aligned} x_1 &:\equiv \mathsf{c}(f(b)) & y_1 &:\equiv y \\ x_2 &:\equiv \mathsf{c}(g(b)) & y_2 &:\equiv D(b)(y) \\ s &:\equiv \mathsf{p}(b) & r &:\equiv q \end{aligned}$$

柯里化函数 $d : \prod_{(w:W)} P(w) \to Q$ 是通过对 $w : W$ 进行归纳定义的；为了应用 Lemma 6.12.7，我们需要了解 $\mathsf{ap}_{d(x_2)}(r)$ 和 $\mathsf{happly}(\mathsf{apd}_d(s), s_*(y_1))$。
对于第一个，由于 $d(\mathsf{c}(a),x) \equiv c(a,x)$，我们有

$$\mathsf{ap}_{d(x_2)}(r) \equiv \mathsf{ap}_{c(g(b),-)}(q)$$

对于第二个，$W$ 的归纳原理的计算规则告诉我们，$\mathsf{apd}_d(\mathsf{p}(b))$ 等于组合 (6.12.6)，通过 Lemma 2.7.6 的等价性传递。因此，Lemma 2.7.6 中给出的计算规则表明 $\mathsf{happly}(\mathsf{apd}_d(\mathsf{p}(b)), \mathsf{p}(b)_*(y))$ 等于组合

$$\begin{aligned} \big(\mathsf{p}(b)_*(c(f(b),-))\big)(\mathsf{p}(b)_*(y)) &= \mathsf{p}(b)_*\Big(c(f(b),\mathsf{p}(b)^{-1}{}_*(\mathsf{p}(b)_*(y)))\Big) && \text{(通过 (2.7.4))} \\ &= \mathsf{p}(b)_*(c(f(b),y)) \\ &= c(f(b),y) && \text{(通过 Lemma 2.3.5)} \\ &= c(g(b),D(b)(y)) && \text{(通过 } p(b,y)\text{)} \\ &= c(g(b),\mathsf{p}(b)_*(y)) && \text{(通过 } \mathsf{ap}_{c(g(b),-)}(q)^{-1}\text{)} \end{aligned}$$

最后，将这些 $\mathsf{ap}_{d(x_2)}(r)$ 和 $\mathsf{happly}(\mathsf{apd}_d(s), s_*(y_1))$ 的值代入 Lemma 6.12.7 中，我们看到所有路径都成对抵消，只剩下 $p(b,y)$。 □

现在我们终于准备好证明展平引理了。

*展平引理 Lemma 6.12.2 的证明.* 我们通过使用 $\widetilde{W}$ 的递归原理，用 $\widetilde{\mathsf{c}}'$ 和 $\widetilde{\mathsf{p}}'$ 作为输入数据来定义 $h : \widetilde{W} \to \sum_{(w:W)} P(w)$。同样地，我们使用 Lemma 6.12.5 的递归原理，用 $\widetilde{\mathsf{c}}$ 和 $\widetilde{\mathsf{p}}$ 作为输入数据来定义 $k : (\sum_{(w:W)} P(w)) \to \widetilde{W}$。
一方面，我们必须证明对于任何 $z : \widetilde{W}$，我们有 $k(h(z)) = z$。通过对 $z$ 进行归纳，只需考虑 $\widetilde{W}$ 的两个构造器即可。但我们有

$$k(h(\widetilde{\mathsf{c}}(a,x))) \equiv k(\widetilde{\mathsf{c}}'(a,x)) \equiv \widetilde{\mathsf{c}}(a,x)$$

根据定义，同样地

$$k(h(\widetilde{\mathsf{p}}(b,y))) = k(\widetilde{\mathsf{p}}'(b,y)) = \widetilde{\mathsf{p}}(b,y)$$

使用 $\widetilde{W}$ 的命题计算规则和 Lemma 6.12.8。
另一方面，我们必须证明对于任何 $z : \sum_{(w:W)} P(w)$，我们有 $h(k(z)) = z$。但这本质上是相同的，使用 Lemma 6.12.4 进行“$\sum_{(w:W)} P(w)$ 的归纳”和相同的计算规则。 □

## 6.13 高阶归纳定义的通用语法 (The general syntax of higher inductive definitions)

在 §5.6中，我们讨论了关于推定的"归纳定义"的条件，使其成为可接受的定义，具体而言，类型在其构造子中的所有归纳出现都是"严格正向的"。在本节中，我们将讨论高阶归纳定义所需的附加条件。找到有效高阶归纳定义的一般语法描述是当前研究的一个领域，到目前为止提出的所有解决方案在本质上都有些技术性；因此，我们只给出一般描述而非精确定义。幸运的是，这些边界情况在实践中似乎从未出现过。

与普通的归纳定义类似，高阶归纳定义是由一系列**构造子**指定的，每个构造子都是一个（依赖的）函数。为简单起见，我们可以要求每个构造子的输入满足与普通归纳类型的构造子的输入相同的条件。特别地，它们可能仅严格正向地包含正在定义的类型。请注意，这排除了如 §6.9中所述的 0-截断定义，其中构造子的输入不仅包含正在定义的归纳类型，还包含其同一类型。也许可以扩展语法以允许这样的定义；但在 §7.3中，我们将给出一个 0-截断的不同构造，其构造子确实满足更严格的条件。

因此，普通归纳定义和高阶归纳定义之间的唯一区别是，构造子的**输出**类型可能不是正在定义的类型（例如 $W$），而是它的一些同一类型，例如 $u =_W v$，或更一般地，是一个迭代的同一类型，例如 $p =_{(u=_W v)} q$。因此，当我们给出高阶归纳定义时，我们不仅要指定每个构造子的输入，还要指定确定路径的源和目标的表达式 $u$ 和 $v$（或 $u$、$v$、$p$ 和 $q$ 等）。

重要的是，这些表达式可以引用 $W$ 的**其他**构造子。例如，在 $\mathbb{S}^1$ 的定义中，构造子 $\mathsf{loop}$ 的 $u$ 和 $v$ 都是先前的构造子 $\mathsf{base}$。为了理解这一点，我们要求高阶归纳类型的构造子按照**顺序**指定，并且允许每个构造子的源和目标表达式 $u$ 和 $v$ 引用先前的构造子，但不能引用后面的构造子。（当然，在实践中，任何归纳定义的构造子都是按某种顺序写下来的，但对于普通归纳类型来说，这个顺序是无关紧要的。）

请注意，这个顺序不一定是"维度"的顺序：原则上，一个一维路径构造子可以引用二维的构造子，因此需要在它之后出现。然而，我们没有给零维的构造子（点构造子）任何引用先前构造子的方式，所以它们最好都先出现。如果我们使用轮辐结构（§6.7）将所有构造子简化为点和一维路径，那么我们可以假设所有点构造子都先出现，然后是所有一维路径构造子——但一维路径构造子之间的顺序仍然很重要。

剩下的问题是，$u$ 和 $v$ 可以是什么样的表达式？我们可能希望它们可以是涉及先前构造子的任何表达式。然而，以下示例表明，这一想法的简单方法是行不通的。

Example 6.13.1. 考虑一组函数 $f : \prod_{(X:\mathcal{U})}(X \to X)$。当然，$f_X$ 可能对所有 $X$ 都是 $\mathsf{id}_X$，但也可能存在其他这样的 $f$。例如，没有什么阻止 $f_{\mathbf{2}} : \mathbf{2} \to \mathbf{2}$ 是非恒等自同构（参见 Exercise 6.9）。

现在假设我们尝试定义一个高阶归纳类型 $K$，其生成元为：

- 两个元素 $a, b : K$，以及
- 一条路径 $\sigma : f_K(a) = f_K(b)$。

$K$ 的归纳原理会是什么样子呢？我们假设一个类型族 $P : K \to \mathcal{U}$，当然我们需要 $x : P(a)$ 和 $y : P(b)$。剩下的数据应该是 $P$ 中生活在 $\sigma$ 之上的依赖路径，它因此必须连接 $P(f_K(a))$ 中的某个元素和 $P(f_K(b))$ 中的某个元素。但这些元素可能是什么呢？我们知道 $P(a)$ 和 $P(b)$ 分别由 $x$ 和 $y$ 占据，但这对 $P(f_K(a))$ 和 $P(f_K(b))$ 没有任何启示。

显然，需要对 $u$ 和 $v$ 进行某种条件限制，以使定义有意义。看起来，就像每个构造子的域被要求是（除其他外）一个**协变函子**，对 $u$ 和 $v$ 的适当条件是它们定义**自然变换**。对这个要求进行精确定义超出了本书的范围，但非正式地说，它意味着 $u$ 和 $v$ 只能涉及所有类型之间的函数所保持的操作。

例如，$u$ 和 $v$ 可以引用路径的串联，如在环面的最终构造子中的情况（§6.6），因为类型理论中的所有函数都保持路径串联（同伦意义上）。然而，它们不能引用像 Example 6.13.1中的函数 $f$ 那样的操作，该操作不一定是自然的：可能存在某个函数 $g : X \to Y$，使得 $f_Y \circ g \neq g \circ f_X$。（同一性蕴含 $f_X$ 必须相对于所有**等价**是自然的，但不一定相对于不是等价的函数。）

自然性的直觉只为何时允许高阶归纳定义提供了一个粗略的指导。即使本书中可能给出这种定义的精确形式的规范，这种规范可能很快就会过时，因为不断有新的扩展理论被探索。例如，在 §6.4中提到

的“依赖 $n$-循环”表示的 $n$ 维球体，以及 Chapter 11中使用的“高阶归纳-递归定义”，都是在本书编写过程中引入的创新。我们鼓励读者进行实验——但要谨慎。

## Notes

高阶归纳类型的概念是 Andrej Bauer、Peter Lumsdaine、Mike Shulman 和 Michael Warren 在 2011 年 Oberwolfach 会议上的讨论中提出的，尽管在早期的一些工作中已经有一些特殊情况的建议。随后，Guillaume Brunerie 和 Dan Licata 对一般理论做出了重要贡献，尤其是找到了在计算机证明助理中表示它们并用它们进行同伦理论的方法（参见 Chapter 8）。

关于高阶归纳类型的语法及其在高阶范畴模型中的语义的一般讨论，见 [?]。与普通归纳类型一样，高阶归纳类型的模型可以通过超限迭代过程构造；一个口号是普通归纳类型描述**自由** Monad，而高阶归纳类型描述 Monad 的**呈示**。路径构造子的引入还涉及模型范畴理论中“右同伦”（使用路径空间定义）和“左同伦”（使用圆柱定义）之间的等价性——这一等价性通常仅在同伦意义上成立，这为路径构造子的命题计算规则提供了语义上的理由。

另一个（暂时的）偏好路径构造子命题计算规则的原因来自现有计算机实现的限制。像 Coq 和已经内置了普通归纳类型，但还没有高阶归纳类型。当然，我们可以通过假设大量公理来引入它们，但这只能得到命题计算规则。然而，Dan Licata 提出了一个技巧，通过私有数据类型实现高阶归纳类型；这为点构造子提供了判断规则，但不适用于路径构造子。

在 §§6.4 and 6.5中使用环空间和悬浮来描述高阶球体的类型论描述，主要归功于 Brunerie 和 Licata；Hou 给出了另一种使用 $n$ 维路径的描述的类型论版本。在 §6.7中，通过轮辐将高阶路径简化为一维路径的描述，归功于 Lumsdaine 和 Shulman。作为高阶归纳类型描述的截断，归功于 Lumsdaine；$(-1)$-截断与 [?]中的“括号类型”密切相关。展平引理首先由 Brunerie 以通用性提出。

在外延类型论中，商类型没有问题，如 NuPRL [?]。它们通常通过转向一个扩展的集合类型系统来添加。然而，商在内涵类型论（同伦类型论的起点）中是一个更棘手的问题，因为不能简单地添加新的命题等式，而不指定它们的行为方式。对这个问题的一些解决方案已经进行了研究 [?, ?, ?]，并且已经考虑了几种不同的商类型概念。使用高阶归纳类型构造集合商为我们采用的方法提供了论据（类似于以前考虑的一些方法），因为它作为一般机制的一个实例出现。我们的构造还没有为与商有关的所有计算问题提供一个新的解决方案，因为我们仍然缺乏对高阶归纳类型的一般计算理解——但它确实意味着对高阶归纳类型的计算解释的持续研究同样适用于商。在同一性公理下，这种商类型在内涵类型论中意味着函数外延性。[?]受到 [?]对精确完备性的研究的启发，证明了这一点；Lemma 6.3.2是这种论证的一个改编版本。

## Exercises

Exercise 6.1. 定义依赖路径的串联，证明依赖函数的应用保持串联，并写出环面 $T^2$ 的精确归纳原理及其计算规则。

Exercise 6.2. 使用 §6.4中给出的 $\mathbb{S}^2$ 的明确定义，证明 $\Sigma\mathbb{S}^1 \simeq \mathbb{S}^2$。

Exercise 6.3. 证明 §6.6中定义的环面 $T^2$ 等价于 $\mathbb{S}^1 \times \mathbb{S}^1$。（警告：路径代数在这方面相当困难。）

Exercise 6.4. 定义依赖 $n$-循环和依赖函数在 $n$-循环上的作用，并写下在 §6.4末尾定义的 $n$-球体的归纳原理。

Exercise 6.5. 使用 §6.4中 $\Omega^n$ 定义的 $\mathbb{S}^n$，证明 $\Sigma\mathbb{S}^n \simeq \mathbb{S}^{n+1}$。

Exercise 6.6. 证明如果类型 $\mathbb{S}^2$ 属于某个宇宙 $\mathcal{U}$ ，则 $\mathcal{U}$ 不是 2 型。

Exercise 6.7. 证明如果 $G$ 是一个幺半群，$x : G$，则 $\sum_{(y:G)}((x \cdot y = e) \times (y \cdot x = e))$ 是一个单纯命题。结论是，使用唯一选择原理 (Corollary 3.9.2)，将群定义为一个幺半群，对于每个 $x : G$，仅有一个 $y : G$ 使得 $x \cdot y = e$ 和 $y \cdot x = e$ 是等价的。

Exercise 6.8. 证明如果 $A$ 是一个集合，那么 $\mathsf{List}(A)$ 是一个幺半群。然后完成 Lemma 6.11.5的证明。

Exercise 6.9. 假设 $\mathsf{LEM}$ ，构造一个函数族 $f : \prod_{(X:\mathcal{U})}(X \to X)$，使得 $f_{\mathbf{2}} : \mathbf{2} \to \mathbf{2}$ 是非恒等自同构。

Exercise 6.10. 证明在 Lemma 6.3.2中构造的映射实际上是 happly 的拟逆，因此一个区间类型意味着完整的函数外延性公理。（您可能需要使用 Exercise 2.16。）

Exercise 6.11. 证明悬浮的通用性原理：

$$\Big(\Sigma A \to B\Big) \simeq \Big(\sum_{(b_n:B)} \sum_{(b_s:B)} (A \to (b_n = b_s))\Big)$$

Exercise 6.12. 证明 $\mathbb{Z} \simeq \mathbb{N} + \mathbf{1} + \mathbb{N}$。证明如果我们将 $\mathbb{Z}$ 定义为 $\mathbb{N} + \mathbf{1} + \mathbb{N}$，那么我们可以得到具有判断计算规则的 Lemma 6.10.12。

Exercise 6.13. 证明我们也可以使用 $\|\mathbf{2}\|$ 而不是 $I$ 来证明 Lemma 6.3.2。

Chapter 7

# 同伦 $n$-类型 (Homotopy $n$-types)

同伦理论 (Homotopy Theory) 中的一个基本概念是**同伦 $n$-类型** (Homotopy $n$-type)：它是一个在维度 $n$ 之上不包含有趣的同伦的空间。例如，一个同伦 0-类型 (Homotopy 0-type) 本质上是一个集合，不包含非平凡路径 (Nontrivial Paths)，而同伦 1-类型 (Homotopy 1-type) 可能包含非平凡路径，但不包含路径之间的非平凡路径。同伦 $n$-类型 (Homotopy $n$-types) 也被称为 **$n$-截断空间** ($n$-truncated spaces)。我们已经在 §3.1 提到了这个概念；我们本章的第一个目标是在同伦类型论 (Homotopy Type Theory) 中给出它的精确定义。

截断性 (Truncatedness) 的对偶概念是连通性 (Connectedness)：一个空间是 **$n$-连通** (n-connected) 的，如果它在维度 $n$ 及其以下没有有趣的同伦。例如，一个空间是 0-连通 (0-connected) 的（也称为 "连通 (Connected)"），如果它只有一个连通分量，而 1-连通 (1-connected) 的（也称为 "单连通 (Simply Connected)"），如果它也没有非平凡循环 (Nontrivial Loops)（尽管它可能有循环之间的非平凡高阶循环 (Higher Loops)）。

截断性 (Truncatedness) 和连通性 (Connectedness) 之间的对偶性最容易通过扩展这两种概念到映射 (Maps) 来看。我们称一个映射为 **$n$-截断** (n-truncated) 或 **$n$-连通** (n-connected) 的，如果它的所有纤维 (Fibers) 都是如此。那么 $n$-连通和 $n$-截断映射 (Maps) 形成一个**正交分解系统** (Orthogonal Factorization System)，即每个映射都唯一地分解为一个 $n$-连通映射和一个 $n$-截断映射。

在 $n = -1$ 的情况下，$n$-截断映射 (n-truncated maps) 是嵌入 (Embeddings)，而 $n$-连通映射 (n-connected maps) 是满射 (Surjections)，如 §4.6 中所定义。因此，$n$-连通分解系统 (n-connected factorization system) 是函数之间标准的图像分解的一个重大推广，它将集合之间的函数分解为一个满射和一个单射。在本章的最后，我们将简要地概述一个更一般的理论：任何类型理论中的**模态性** (Modality) 都会产生一个类似的分解系统。

## 7.1 $n$-类型的定义 (Definition of $n$-types)

如 §§3.1 and 3.11 所述，从零以下两个级别开始定义 $n$-类型是方便的，其中 $(-1)$-类型是纯命题 (Mere Propositions)，$(-2)$-类型是可收缩的 (Contractible)。

Definition 7.1.1. 定义谓词 (Predicate) $\mathsf{is}\text{-}n\text{-}\mathsf{type} : \mathcal{U} \to \mathcal{U}$ 对于 $n \geq -2$，其递归定义如下：

$$\mathsf{is}\text{-}n\text{-}\mathsf{type}(X) :\equiv \begin{cases} \mathsf{isContr}(X) & \text{当} n = -2 \text{ 时,} \\ \prod_{(x,y:X)} \mathsf{is}\text{-}n'\text{-}\mathsf{type}(x =_X y) & \text{当} n = n' + 1 \text{ 时.} \end{cases}$$

我们称 $X$ 是一个 **$n$-类型** (n-type)，有时也称它是 **$n$-截断的** ($n$-truncated)，如果 $\mathsf{is}\text{-}n\text{-}\mathsf{type}(X)$ 是居留的 (Inhabited)。

Remark 7.1.2. 在 Definition 7.1.1 中，$n$ 的取值范围包括所有大于或等于 $-2$ 的整数。我们可以通过定义类型 $\mathbb{Z}_{\geq -2}$ 这样的整数类型来正式理解这一点（该类型的归纳原理与 $\mathbb{N}$ 的归纳原理相同），或者

定义谓词 $\mathsf{is}\text{-}(k-2)\text{-}\mathsf{type}$ 对于 $k:\mathbb{N}$。无论哪种方式，我们都可以通过对 $n$ 进行归纳来证明关于 $n$-类型的定理，其中 $n=-2$ 是基本情况。

Example 7.1.3. 我们在 Lemma 3.11.10 中已经看到，$X$ 是一个 $(-1)$-类型当且仅当它是一个纯命题 (Mere Proposition)。因此，$X$ 是一个 0-类型当且仅当它是一个集合 (Set)。

我们还看到有些类型不是集合 (Example 3.1.9)。然而，到目前为止，我们还没有展示对于任何 $n>0$ 的类型，它们不是 $n$-类型。在 Chapter 8 中，我们将展示 $(n+1)$-球体 $\mathbb{S}^{n+1}$ 不是 $n$-类型。(Kraus 还展示了第 $n$ 层嵌套的单值宇宙 (Nested Univalent Universe) 也不是 $n$-类型，而没有使用任何高阶归纳类型 (Higher Inductive Types))。此外，在 ?? 中，我们将给出一个类型，它不是任何有限数 $n$ 的 $n$-类型。

我们从展示 $n$-类型在某些操作和构造下是封闭的开始，来展开 $n$-类型的一般理论。

Theorem 7.1.4. *令 $p:X\to Y$ 是一个缩回 (Retraction)，并且假设 $X$ 是一个 $n$-类型，对于任何 $n\geq -2$。那么 $Y$ 也是一个 $n$-类型。*

*证明.* 我们通过对 $n$ 进行归纳来证明。基本情况 $n=-2$ 由 Lemma 3.11.7 处理。

对于归纳步骤，假设任何 $n$-类型的缩回也是 $n$-类型，并且 $X$ 是一个 $(n+1)$-类型。令 $y,y':Y$；我们必须证明 $y=y'$ 是一个 $n$-类型。令 $s$ 是 $p$ 的一个截面 (Section)，并令 $\epsilon$ 是一个同伦 $\epsilon:p\circ s\sim 1$。由于 $X$ 是一个 $(n+1)$-类型，$s(y)=_X s(y')$ 是一个 $n$-类型。我们声称 $y=y'$ 是 $s(y)=_X s(y')$ 的一个缩回。对于截面，我们取

$$\mathsf{ap}_s:(y=y')\to(s(y)=s(y'))$$

对于缩回，我们定义 $t:(s(y)=s(y'))\to(y=y')$ 为

$$t(q):\equiv {\epsilon_y}^{-1}\cdot p(q)\cdot\epsilon_{y'}$$

为了证明 $t$ 是 $\mathsf{ap}_s$ 的缩回，我们必须证明

$${\epsilon_y}^{-1}\cdot p(s(r))\cdot\epsilon_{y'}=r$$

对于任意 $r:y=y'$。但这由 Lemma 2.4.3 推出。 □

作为一个直接的推论，我们得到 $n$-类型在等价 (Equivalence) 下的稳定性（这也是由单值性 (Univalence) 直接得到的）：

Corollary 7.1.5. *如果 $X\simeq Y$ 且 $X$ 是一个 $n$-类型，那么 $Y$ 也是。*

还记得在 §4.6 中的嵌入 (Embedding) 的概念。

Theorem 7.1.6. *如果 $f:X\to Y$ 是一个嵌入 (Embedding)，并且 $Y$ 是一个 $n$-类型，对于某个 $n\geq -1$，那么 $X$ 也是。*

*证明.* 令 $x,x':X$；我们必须证明 $x=_X x'$ 是一个 $(n-1)$-类型。但是由于 $f$ 是一个嵌入，我们有 $(x=_X x')\simeq(f(x)=_Y f(x'))$，而后者根据假设是一个 $(n-1)$-类型。 □

注意，当 $n=-2$ 时，这个定理是不成立的：映射 $\mathbf{0}\to\mathbf{1}$ 是一个嵌入，但 $\mathbf{1}$ 是一个 $(-2)$-类型，而 $\mathbf{0}$ 不是。

Theorem 7.1.7. *$n$-类型的层次结构是累积的，在以下意义上：给定一个数 $n\geq -2$，如果 $X$ 是一个 $n$-类型，那么它也是一个 $(n+1)$-类型。*

*证明.* 我们通过对 $n$ 进行归纳来证明。

对于 $n=-2$，我们需要证明一个可收缩类型 (Contractible Type)，例如 $A$，具有可收缩的路径空间 (Path Spaces)。令 $a_0:A$ 为 $A$ 的收缩中心 (Center of Contraction)，并令 $x,y:A$。我们证明 $x=_A y$ 是可收缩的。由于 $A$ 的可收缩性，我们有一条路径 $\mathsf{contr}_x\cdot{\mathsf{contr}_y}^{-1}:x=y$，我们选择它作为 $x=y$ 的收缩中心。给定任何路径 $p:x=y$，我们需要证明 $p=\mathsf{contr}_x\cdot{\mathsf{contr}_y}^{-1}$。通过路径归纳 (Path Induction)，只需证明 $\mathsf{refl}_x=\mathsf{contr}_x\cdot{\mathsf{contr}_x}^{-1}$，这是显然的。

对于归纳步骤，我们需要证明 $x =_X y$ 是一个 $(n+1)$-类型，前提是 $X$ 是一个 $(n+1)$-类型。将归纳假设应用于 $x =_X y$，即可得出所需结论。 □

现在我们展示 $n$-类型在大多数类型构造操作下的保留性。

Theorem 7.1.8. 设 $n \geq -2$，并令 $A : \mathcal{U}$ 和 $B : A \to \mathcal{U}$。如果 $A$ 是一个 $n$-类型，并且对于所有 $a : A$，$B(a)$ 是一个 $n$-类型，那么 $\sum_{(x:A)} B(x)$ 也是一个 $n$-类型。

证明. 我们通过对 $n$ 进行归纳来证明。
对于 $n = -2$，我们选择 $\sum_{(x:A)} B(x)$ 的收缩中心为 $(a_0, b_0)$，其中 $a_0 : A$ 是 $A$ 的收缩中心，$b_0 : B(a_0)$ 是 $B(a_0)$ 的收缩中心。给定 $\sum_{(x:A)} B(x)$ 的任何其他元素 $(a, b)$，我们通过分别对 $A$ 和 $B(a_0)$ 的收缩性提供路径 $(a, b) = (a_0, b_0)$。
对于归纳步骤，假设 $A$ 是一个 $(n+1)$-类型，并且对于任意 $a : A$，$B(a)$ 是一个 $(n+1)$-类型。我们证明 $\sum_{(x:A)} B(x)$ 是一个 $(n+1)$-类型：固定 $\sum_{(x:A)} B(x)$ 中的 $(a_1, b_1)$ 和 $(a_2, b_2)$，我们需要证明 $(a_1, b_1) = (a_2, b_2)$ 是一个 $n$-类型。根据 ??，我们有

$$((a_1, b_1) = (a_2, b_2)) \simeq \sum_{p:a_1 = a_2} (p_*(b_1) =_{B(a_2)} b_2)$$

并且根据等价下 $n$-类型的保留性 (Corollary 7.1.5)，我们只需证明后者是一个 $n$-类型。这可以通过归纳假设得到。 □

作为一个特例，如果 $A$ 和 $B$ 是 $n$-类型，那么 $A \times B$ 也是。还要注意，Theorem 7.1.7 意味着如果 $A$ 是一个 $n$-类型，那么对于任意 $x, y : A$，$x =_A y$ 也是一个 $n$-类型。将此与 Theorem 7.1.8 结合，我们可以看到，对于 $n$-类型的函数 $f : A \to C$ 和 $g : B \to C$，它们的纤维积 (Pullback)

$$A \times_C B :\equiv \sum_{(x:A)} \sum_{(y:B)} (f(x) = g(y))$$

（参见 Exercise 2.11）也是一个 $n$-类型。更一般地说，$n$-类型在所有极限 (Limits) 下是封闭的。

Theorem 7.1.9. 令 $n \geq -2$，并令 $A : \mathcal{U}$ 和 $B : A \to \mathcal{U}$。如果对于所有 $a : A$，$B(a)$ 是一个 $n$-类型，那么 $\prod_{(x:A)} B(x)$ 也是一个 $n$-类型。

证明. 我们通过对 $n$ 进行归纳来证明。对于 $n = -2$，结果就是 Lemma 3.11.6。
对于归纳步骤，假设结果对于 $n$-类型成立，并且每个 $B(a)$ 都是一个 $(n+1)$-类型。令 $f, g : \prod_{(a:A)} B(a)$。我们需要证明 $f = g$ 是一个 $n$-类型。通过函数扩展性 (Function Extensionality) 和等价下 $n$-类型的封闭性，证明 $\prod_{(a:A)}(f(a) =_{B(a)} g(a))$ 是一个 $n$-类型即可。这个结论可通过归纳假设得到。 □

作为上述定理的一个特例，函数空间 $A \to B$ 是一个 $n$-类型，前提是 $B$ 是一个 $n$-类型。我们现在可以推广在 Chapter 2 中关于 $\mathsf{isSet}(A)$ 和 $\mathsf{isProp}(A)$ 是纯命题的观察。

Theorem 7.1.10. 对于任何 $n \geq -2$ 和任何类型 $X$，类型 $\mathsf{is}\text{-}n\text{-}\mathsf{type}(X)$ 是一个纯命题。

证明. 我们通过对 $n$ 进行归纳来证明。
对于基本情况，我们需要证明对于任何 $X$，类型 $\mathsf{isContr}(X)$ 是一个纯命题。这是 Lemma 3.11.4。
对于归纳步骤，我们需要证明

$$\prod_{X:\mathcal{U}} \mathsf{isProp}(\mathsf{is}\text{-}n\text{-}\mathsf{type}(X)) \to \prod_{X:\mathcal{U}} \mathsf{isProp}(\mathsf{is}\text{-}(n+1)\text{-}\mathsf{type}(X))$$

要证明这个推论的结论，我们需要证明对于任何类型 $X$，类型

$$\prod_{x,x':X} \mathsf{is}\text{-}n\text{-}\mathsf{type}(x = x')$$

是一个纯命题。根据 Example 3.6.2 或 Theorem 7.1.9，只需证明对于任何 $x, x' : X$，类型 $\mathsf{is}\text{-}n\text{-}\mathsf{type}(x =_X x')$ 是一个纯命题即可。这可由归纳假设应用于类型 $(x =_X x')$ 得到。 □

最后，我们展示 $n$-类型的类型本身是一个 $(n+1)$-类型。我们将其定义为：

$$n\text{-Type} :\equiv \sum_{X:\mathcal{U}} \mathsf{is}\text{-}n\text{-}\mathsf{type}(X)$$

如有必要，我们可以通过写成 $n\text{-Type}_{\mathcal{U}}$ 来指定宇宙 $\mathcal{U}$。特别地，我们有 $\mathsf{Prop} :\equiv (-1)\text{-Type}$ 和 $\mathsf{Set} :\equiv 0\text{-Type}$，如 Chapter 2 中定义的。注意，由于 $\mathsf{is}\text{-}n\text{-}\mathsf{type}(X)$ 是一个纯命题，根据 Lemma 3.5.1 对于任何 $(X,p),(X',p') : n\text{-Type}$，我们有

$$\begin{aligned}\Big((X,p) =_{n\text{-Type}} (X',p')\Big) &\simeq (X =_{\mathcal{U}} X')\\ &\simeq (X \simeq X')\end{aligned}$$

Theorem 7.1.11. 对于任何 $n \geq -2$，类型 $n$-Type 是一个 $(n+1)$-类型。

证明. 令 $(X,p),(X',p') : n\text{-Type}$；我们需要证明 $(X,p) = (X',p')$ 是一个 $n$-类型。根据上述观察，这个类型等价于 $X \simeq X'$。接下来，我们观察到投影

$$(X \simeq X') \to (X \to X').$$

是一个嵌入，因此如果 $n \geq -1$，根据 Theorem 7.1.6，只需证明 $X \to X'$ 是一个 $n$-类型即可。但是，由于 $n$-类型在箭头类型下是封闭的，这归结为假设 $X'$ 是一个 $n$-类型。
在 $n = -2$ 的情况下，这个论证表明 $X \simeq X'$ 是一个 $(-1)$-类型——但它也是居留的，因为任何两个可收缩类型都等价于 $\mathbf{1}$，因此彼此等价。因此，$X \simeq X'$ 也是一个 $(-2)$-类型。 □

## 7.2 同一性证明的唯一性与 Hedberg 定理 (Uniqueness of identity proofs and Hedberg's theorem)

在 §3.1 中，我们定义了一个类型 $X$ 是集合 (Set)，如果对于所有 $x,y : X$ 和 $p,q : x =_X y$ 我们有 $p = q$。在传统类型论中，这个属性称为**同一性证明的唯一性** (Uniqueness of Identity Proofs, UIP)。我们还看到它等价于上一节中 0-类型的定义。这里是另一个等价的刻画，涉及到 Streicher 的 "公理 K (Axiom K)" [?]：

Theorem 7.2.1. 一个类型 $X$ 是集合 (Set)，当且仅当它满足**公理** K (Axiom K)：对于所有 $x : X$ 和 $p : (x =_A x)$ 我们有 $p = \mathsf{refl}_x$。

证明. 显然，公理 K 是 UIP 的特例。反之，如果 $X$ 满足公理 K，则令 $x,y : X$ 和 $p,q : (x = y)$；我们想要证明 $p = q$。但是对 $q$ 进行归纳 (Induction) 将这个目标精简为公理 K。 □

我们强调，我们并不假设 UIP 或 K 原理作为公理！它们只是某一特定类型可能或可能不满足的属性（它们等价于集合）。回忆 Example 3.1.9，并非所有类型都是集合。
下面的定理是另一个证明类型为集合的有用方法。

Theorem 7.2.2. 假设 $R$ 是一个在类型 $X$ 上的反身纯关系 (Reflexive Mere Relation)，它暗含了同一性 (Identity)。那么 $X$ 是集合，并且对于所有 $x,y : X$，$R(x,y)$ 等价于 $x =_X y$。

证明. 令 $\rho : \prod_{(x:X)} R(x,x)$ 证明 $R$ 的反身性，并令 $f : \prod_{(x,y:X)} R(x,y) \to (x =_X y)$ 证明 $R$ 暗含同一性。首先，定理中的两个陈述是等价的。一方面，如果 $X$ 是一个集合，那么 $x =_X y$ 是一个纯命题，并且由于它在逻辑上等价于纯命题 $R(x,y)$，根据假设，它也必须等价于它。另一方面，如果 $x =_X y$ 等价于 $R(x,y)$，那么与后者一样，对于所有 $x,y : X$，它是一个纯命题，因此 $X$ 是一个集合。
我们给出两种证明这个定理的方法。第一种直接证明 $X$ 是一个集合；第二种直接证明 $R(x,y) \simeq (x = y)$。
**第一种证明**：我们证明 $X$ 是一个集合。这个思路与 Lemma 3.3.4 相同：函数 $f$ 必须在其参数 $x$ 和 $y$ 中是连续的。然而，由于我们必须处理额外的 $R(x,y)$ 类型的参数，符号表达上稍微复杂一些。

首先，对于任何 $x : X$ 和 $p : x =_X x$，考虑 $\mathsf{apd}_{f(x)}(p)$。这是一个从 $f(x,x)$ 到它自身的依赖路径 (Dependent Path)。由于 $f(x,x)$ 仍然是一个函数 $R(x,x) \to (x =_X x)$，根据 Lemma 2.7.6，这给出了对于任何 $r : R(x,x)$ 的路径

$$p_*(f(x,x,r)) = f(x,x,p_*(r))$$

在左侧，我们有同一类型中的传输 (Transport)，即连接。在右侧，我们有 $p_*(r) = r$，因为它们都属于纯命题 $R(x,x)$。因此，替代 $r :\equiv \rho(x)$，我们得到

$$f(x,x,\rho(x)) \cdot p = f(x,x,\rho(x))$$

通过消去得到 $p = \mathsf{refl}_x$。所以 $X$ 满足公理 K，因此是一个集合。

**第二种证明**：我们证明每个 $f(x,y) : R(x,y) \to x =_X y$ 是一个等价 (Equivalence)。根据 Theorem 4.7.7，我们只需证明 $f$ 在总空间 (Total Space) 上诱导了一个等价：

$$\Big(\sum_{y:X} R(x,y)\Big) \simeq \Big(\sum_{y:X} x =_X y\Big)$$

根据 Lemma 3.11.8，右侧的类型是可收缩的，因此只需证明左侧的类型是可收缩的。我们选择对 $(x,\rho(x))$ 的对作为收缩中心。剩下的就是对每个 $y : X$ 和每个 $H : R(x,y)$ 证明

$$(x,\rho(x)) = (y,H)$$

但由于 $R(x,y)$ 是一个纯命题，根据 ??，我们只需证明 $x =_X y$，这是从 $f(H)$ 得到的。 □

Corollary 7.2.3. *如果一个类型 $X$ 具有属性 $\neg\neg(x = y) \to (x = y)$ 对于任何 $x,y : X$，那么 $X$ 是一个集合。*

另一种方便的方法来证明一个类型是集合如下。回忆 §3.4，一个类型 $X$ 被认为具有**可判定等同性** (Decidable Equality)，如果对于所有 $x,y : X$ 我们有

$$(x =_X y) + \neg(x =_X y)$$

这是一种非常强的条件：它意味着当路径存在时，$x = y$ 可以在 $x$ 和 $y$ 中连续（或可计算，或函子化）地选择出来。通过 Theorem 7.2.2 和下面的引理，这最终会导致 $X$ 是一个集合。

Lemma 7.2.4. *对于任何类型 $A$，我们有 $(A + \neg A) \to (\neg\neg A \to A)$。*

*证明.* 这在本质上已经在 Corollary 3.2.7 中证明过了，但我们重复这个论证。假设 $x : A + \neg A$。我们有两个情况需要考虑。如果 $x$ 是某个 $a : A$ 的 $\mathsf{inl}(a)$，那么我们有一个常函数 $\neg\neg A \to A$，它将所有内容映射到 $a$。如果 $x$ 是某个 $t : \neg A$ 的 $\mathsf{inr}(t)$，我们有 $g(t) : \mathbf{0}$ 对于每个 $g : \neg\neg A$。因此，我们可以使用*从错误中推出任何结论*，即 $\mathsf{rec}_{\mathbf{0}}$，对任意 $g : \neg\neg A$ 得到一个 $A$ 的元素。 □

Theorem 7.2.5 (Hedberg). *如果 $X$ 具有可判定等同性 (Decidable Equality)，那么 $X$ 是一个集合。*

*证明.* 如果 $X$ 具有可判定等同性，则对于任何 $x,y : X$，$\neg\neg(x = y) \to (x = y)$ 成立。因此，Hedberg 定理从 Corollary 7.2.3 推出。 □

当然，这个定理与 Corollary 3.2.7 有很强的联系。被 Corollary 3.2.7 否定的 $\mathsf{LEM}_\infty$ 陈述显然意味着每个类型都有可判定等同性，因此是一个集合，这我们知道并非如此。请注意，一致的公理 $\mathsf{LEM}$ 从 §3.4 推出仅意味着每个类型具有*仅仅是可判定等同性* (Merely Decidable Equality)，即对于任何 $A$ 我们有

$$\prod_{a,b:A} (\|a = b\| + \neg\|a = b\|)$$

作为 Theorem 7.2.5 的一个应用实例，回忆在 Example 3.1.4 中我们观察到 $\mathbb{N}$ 是一个集合，通过 ?? 中我们对其等同性类型的刻画。更传统的证明这个定理的方法仅使用 (??) 和 (??)，而不是 ?? 的完整刻画，使用 Theorem 7.2.5 来填补空白。

Theorem 7.2.6. 自然数类型 $\mathbb{N}$ 具有可判定等同性，因此是一个集合。

证明. 令 $x, y : \mathbb{N}$ 被给定；我们通过对 $x$ 进行归纳和对 $y$ 进行情况分析来证明 $(x = y) + \neg(x = y)$。如果 $x \equiv 0$ 且 $y \equiv 0$，我们取 $\mathsf{inl}(\mathsf{refl}_0)$。如果 $x \equiv 0$ 且 $y \equiv \mathsf{succ}(n)$，则根据 (??) 我们得到 $\neg(0 = \mathsf{succ}(n))$。对于归纳步骤，令 $x \equiv \mathsf{succ}(n)$。如果 $y \equiv 0$，我们再次使用 (??)。最后，如果 $y \equiv \mathsf{succ}(m)$，归纳假设给出 $(m = n) + \neg(m = n)$。在第一种情况下，如果 $p : m = n$，那么 $\mathsf{succ}(p) : \mathsf{succ}(m) = \mathsf{succ}(n)$。在第二种情况下，(??) 导致 $\neg(\mathsf{succ}(m) = \mathsf{succ}(n))$。 □

尽管 Hedberg 定理看起来对集合（0-类型）特别适用，“公理 K (Axiom K)” 自然地推广到 $n$-类型。请注意，普通的公理 K（作为类型 $X$ 的属性）陈述为对于所有 $x : X$，环空间 (Loop Space) $\Omega(X, x)$（见 Definition 2.1.8）是可收缩的 (Contractible)。由于 $\Omega(X, x)$ 总是居留的（通过 $\mathsf{refl}_x$），这等价于它是一个纯命题（一个 $(-1)$-类型）。由于 $0 = (-1) + 1$，这提出了以下推广。

Theorem 7.2.7. 对于任何 $n \geq -1$，类型 $X$ 是一个 $(n+1)$-类型，当且仅当对于所有 $x : X$，类型 $\Omega(X, x)$ 是一个 $n$-类型。

在证明此之前，我们先证明一个辅助引理：

Lemma 7.2.8. 给定 $n \geq -1$ 和 $X : \mathcal{U}$。如果给定 $X$ 的任何居留元素就能推出 $X$ 是一个 $n$-类型，那么 $X$ 是一个 $n$-类型。

证明. 令 $f : X \to \mathsf{is}\text{-}n\text{-}\mathsf{type}(X)$ 是给定的映射。我们需要证明对于任何 $x, x' : X$，类型 $x = x'$ 是一个 $(n-1)$-类型。但是 $f(x)$ 表明 $X$ 是一个 $n$-类型，因此它的所有路径空间 (Path Spaces) 都是 $(n-1)$-类型。 □

Proof of Theorem 7.2.7. “如果” 的方向是显然的，因为 $\Omega(X, x) :\equiv (x =_X x)$。反之，为了证明 $X$ 是一个 $(n+1)$-类型，我们需要证明对于任何 $x, x' : X$，类型 $x = x'$ 是一个 $n$-类型。根据 Lemma 7.2.8，只需给出一个映射

$$(x = x') \to \mathsf{is}\text{-}n\text{-}\mathsf{type}(x = x')$$

通过路径归纳 (Path Induction)，只需证明当 $x \equiv x'$ 时，这来自于假设 $\Omega(X, x)$ 是一个 $n$-类型。 □

通过归纳和一些巧妙的刷操作 (Whiskering)，我们可以将 K 属性推广到 $n > 0$。

Theorem 7.2.9. 对于每个 $n \geq -1$，一个类型 $A$ 是一个 $n$-类型，当且仅当对于所有 $a : A$，$\Omega^{n+1}(A, a)$ 是可收缩的。

证明. 回忆 $\Omega^0(A, a) = (A, a)$，$n = -1$ 的情况是 Exercise 3.5。$n = 0$ 的情况是 Theorem 7.2.1。现在我们使用归纳法；假设该陈述对 $n : \mathbb{N}$ 成立。根据 Theorem 7.2.7，$A$ 是一个 $(n+1)$-类型，当且仅当对于所有 $a : A$，$\Omega(A, a)$ 是一个 $n$-类型。根据归纳假设，后者等价于 $\Omega^{n+1}(\Omega(A, a), p)$ 对于所有 $p : \Omega(A, a)$ 是可收缩的。
由于 $\Omega^{n+2}(A, a) :\equiv \Omega^{n+1}(\Omega(A, a), \mathsf{refl}_a)$，并且 $\Omega^{n+1} = \Omega^n \circ \Omega$，它将足以证明 $\Omega(\Omega(A, a), p)$ 等于 $\Omega(\Omega(A, a), \mathsf{refl}_a)$，在类型 $\mathcal{U}_\bullet$ 中。为此，只需给出一个等价

$$g : \Omega(\Omega(A, a), p) \simeq \Omega(\Omega(A, a), \mathsf{refl}_a)$$

它将基点 (Basepoint) $\mathsf{refl}_p$ 映射到基点 $\mathsf{refl}_{\mathsf{refl}_a}$。对于 $q : p = p$，定义 $g(q) : \mathsf{refl}_a = \mathsf{refl}_a$ 为以下复合：

$$\mathsf{refl}_a = p \cdot p^{-1} \stackrel{q}{=} p \cdot p^{-1} = \mathsf{refl}_a$$

其中标有 “$q$” 的路径实际上是 $\mathsf{ap}_{\lambda r.\, r \cdot p^{-1}}(q)$。然后 $g$ 是一个等价，因为它是等价的复合

$$(p = p) \xrightarrow{\mathsf{ap}_{\lambda r.\, r \cdot p^{-1}}} (p \cdot p^{-1} = p \cdot p^{-1}) \xrightarrow{i \cdot - \cdot i^{-1}} (\mathsf{refl}_a = \mathsf{refl}_a)$$

使用 Example 2.4.8 and ??，其中 $i : \mathsf{refl}_a = p \cdot p^{-1}$ 是规范等价。显然，$g(\mathsf{refl}_p) = \mathsf{refl}_{\mathsf{refl}_a}$。 □

## 7.3 **截断** (Truncations)

在 §3.7 中，我们介绍了命题截断 (Propositional Truncation)，它提供了某个类型作为一个纯命题（即 $(-1)$-类型）的 “最佳近似”。在 §6.9 中，我们将此截断构造为一个高阶归纳类型 (Higher Inductive Type)，并给出了将其推广为 0-截断的一种方法。现在，我们将解释一种更好的推广方式，它将任意类型截断为任意 $n \geq -2$ 的 $n$-类型；在经典同伦理论中，这被称为其 $n^{\text{th}}$ Postnikov **截断** (Postnikov section)。

其思想是利用 Theorem 7.2.9，该定理表明 $A$ 是一个 $n$-类型，当且仅当 $\Omega^{n+1}(A,a)$ 对于所有 $a : A$ 是可收缩的，并且利用 Lemma 6.5.4，该引理暗示 $\Omega^{n+1}(A,a) \simeq \mathsf{Map}_*(\mathbb{S}^{n+1},(A,a))$，其中 $\mathbb{S}^{n+1}$ 配备了一些基点 (Basepoint)，我们不妨称其为 $\mathsf{base}$。然而，通过给出路径构造子，可以直接确保 $\mathsf{Map}_*(\mathbb{S}^{n+1},(A,a))$ 的可收缩性。

我们将使用 “轮毂和辐条” 构造 (Hub and Spoke Construction)，如 §6.7 所述。因此，对于 $n \geq -1$，我们将 $\|A\|_n$ 视为由以下构造生成的高阶归纳类型：

- 一个函数 $|-|_n : A \to \|A\|_n$，
- 对于每个 $r : \mathbb{S}^{n+1} \to \|A\|_n$，一个**轮毂** (Hub) 点 $h(r) : \|A\|_n$，以及
- 对于每个 $r : \mathbb{S}^{n+1} \to \|A\|_n$ 和每个 $x : \mathbb{S}^{n+1}$，一个**辐条** (Spoke) 路径 $s_r(x) : r(x) = h(r)$。

这些构造子的存在足以证明：

Lemma 7.3.1. *$\|A\|_n$ 是一个 $n$-类型。*

*证明.* 根据 Theorem 7.2.9，只需证明 $\Omega^{n+1}(\|A\|_n, b)$ 对于所有 $b : \|A\|_n$ 是可收缩的，根据 Lemma 6.5.4 这等价于 $\mathsf{Map}_*(\mathbb{S}^{n+1},(\|A\|_n,b))$。作为后者的收缩中心，我们选择常数为 $b$ 的函数 $c_b : \mathbb{S}^{n+1} \to \|A\|_n$，以及 $\mathsf{refl}_b : c_b(\mathsf{base}) = b$。

现在，$\mathsf{Map}_*(\mathbb{S}^{n+1},(\|A\|_n,b))$ 的任意元素由一个映射 $r : \mathbb{S}^{n+1} \to \|A\|_n$ 以及一个路径 $p : r(\mathsf{base}) = b$ 组成。根据函数外延性 (Function Extensionality)，为了证明 $r = c_b$，只需给出每个 $x : \mathbb{S}^{n+1}$ 的路径 $r(x) = c_b(x) \equiv b$。我们选择它为复合路径 $s_r(x) \cdot s_r(\mathsf{base})^{-1} \cdot p$，其中 $s_r(x)$ 是 $x$ 处的辐条路径。

最后，我们必须证明当沿着这个等式 $r = c_b$ 进行传递时，路径 $p$ 变为 $\mathsf{refl}_b$。通过路径类型中的传递，这意味着我们需要

$$(s_r(\mathsf{base}) \cdot s_r(\mathsf{base})^{-1} \cdot p)^{-1} \cdot p = \mathsf{refl}_b$$

但这是通过路径运算立即得出的。 □

（这种构造对 $n = -2$ 失效，但在这种情况下我们可以简单地定义 $\|A\|_{-2} :\equiv \mathbf{1}$ 对于所有 $A$。从现在起我们假设 $n \geq -1$。）

为了展示 $n$-截断的预期通用性质 (Universal Property)，我们需要归纳原理 (Induction Principle)。我们按照通常的方式从构造子中提取它；它表明，给定 $P : \|A\|_n \to \mathcal{U}$ 以及以下内容：

- 对于每个 $a : A$，一个元素 $g(a) : P(|a|_n)$，
- 对于每个 $r : \mathbb{S}^{n+1} \to \|A\|_n$ 和 $r' : \prod_{(x:\mathbb{S}^{n+1})} P(r(x))$，一个元素 $h'(r,r') : P(h(r))$，
- 对于每个 $r : \mathbb{S}^{n+1} \to \|A\|_n$ 和 $r' : \prod_{(x:\mathbb{S}^{n+1})} P(r(x))$，以及每个 $x : \mathbb{S}^{n+1}$，一个依赖路径 $r'(x) =^P_{s_r(x)} h'(r,r')$，

存在一个截面 $f : \prod_{(x:\|A\|_n)} P(x)$，其中 $f(|a|_n) \equiv g(a)$ 对于所有 $a : A$ 成立。为了使其更有用，我们将其重新表述如下。

Theorem 7.3.2. *对于任意类型族 $P : \|A\|_n \to \mathcal{U}$，使得每个 $P(x)$ 是一个 $n$-类型，并且任意函数 $g : \prod_{(a:A)} P(|a|_n)$，存在一个截面 $f : \prod_{(x:\|A\|_n)} P(x)$，使得 $f(|a|_n) :\equiv g(a)$ 对于所有 $a : A$ 成立。*

*证明.* 只需构造上述列出的第二个和第三个数据，因为 $g$ 的类型与第一个数据完全一致。给定 $r : \mathbb{S}^{n+1} \to \|A\|_n$ 和 $r' : \prod_{(x:\mathbb{S}^{n+1})} P(r(x))$，我们有 $h(r) : \|A\|_n$ 和 $s_r : \prod_{(x:\mathbb{S}^{n+1})}(r(x) = h(r))$。定义 $t : \mathbb{S}^{n+1} \to P(h(r))$ 为 $t(x) :\equiv s_r(x)_*(r'(x))$。然后由于 $P(h(r))$ 是 $n$-截断的，因此存在一个点 $u : P(h(r))$ 和一个收缩 $v : \prod_{(x:\mathbb{S}^{n+1})}(t(x) = u)$。定义 $h'(r,r') :\equiv u$，提供了第二个数据。然后（回忆依赖路径的定义），$v$ 正好具有第三个数据所需的类型。 □

特别地，如果 $E$ 是某个 $n$-类型，我们可以考虑在 $A$ 的每个点上等于 $E$ 的常数类型族 (Constant Family of Types)。因此，每个映射 $f : A \to E$ 都可以扩展为映射 $\mathsf{ext}(f) : \|A\|_n \to E$，定义为 $\mathsf{ext}(f)(|a|_n) :\equiv f(a)$；这是 $\|A\|_n$ 的*递归原理* (Recursion Principle)。
归纳原理还暗示了这种形式的函数的唯一性原理。也就是说，如果 $E$ 是一个 $n$-类型，并且 $g, g' : \|A\|_n \to E$ 满足 $g(|a|_n) = g'(|a|_n)$ 对于每个 $a : A$，那么 $g(x) = g'(x)$ 对于所有 $x : \|A\|_n$ 成立，因为类型 $g(x) = g'(x)$ 是一个 $n$-类型。因此，$g = g'$。（实际上，当 $E$ 是一个 $(n+1)$-类型时，这个唯一性原理在更广泛的情况下也是成立的。）这产生了以下通用性质。

Lemma 7.3.3 (截断的通用性质 (Universal Property of Truncations)). *设 $n \geq -2$, $A : \mathcal{U}$ 和 $B : n\text{-}\mathsf{Type}$。以下映射是一个等价：*

$$\begin{cases} (\|A\|_n \to B) & \longrightarrow & (A \to B) \\ g & \longmapsto & g \circ |-|_n \end{cases}$$

*证明.* 考虑到 $B$ 是 $n$-截断的，任意 $f : A \to B$ 可以扩展为映射 $\mathsf{ext}(f) : \|A\|_n \to B$。映射 $\mathsf{ext}(f) \circ |-|_n$ 等于 $f$，因为对于每个 $a : A$，我们有 $\mathsf{ext}(f)(|a|_n) = f(a)$ 根据定义。而映射 $\mathsf{ext}(g \circ |-|_n)$ 等于 $g$，因为它们都将 $|a|_n$ 映射到 $g(|a|_n)$。 □

在范畴语言中，这意味着 $n$-类型形成了类型范畴 (Category of Types) 的一个*反射子范畴* (Reflective Subcategory)。（为了完全精确地陈述这一点，应使用 $(\infty,1)$-范畴的语言。）特别地，这意味着 $n$-截断是函子性的：给定 $f : A \to B$，将递归原理应用于复合 $A \xrightarrow{f} B \to \|B\|_n$ 产生映射 $\|f\|_n : \|A\|_n \to \|B\|_n$。根据定义，我们有一个同伦

$$\mathsf{nat}_n^f : \prod_{a:A} \|f\|_n(|a|_n) = |f(a)|_n, \tag{7.3.4}$$

表达了映射 $|-|_n$ 的*自然性* (Naturality)。
唯一性暗示了函子性定律 (Functoriality Laws)，例如 $\|g \circ f\|_n = \|g\|_n \circ \|f\|_n$ 和 $\|\mathsf{id}_A\|_n = \mathsf{id}_{\|A\|_n}$，以及随附的相干性定律 (Coherence Laws)。我们还有更高的函子性，例如：

Lemma 7.3.5. *给定 $f, g : A \to B$ 和一个同伦 $h : f \sim g$，存在一个诱导的同伦 $\|h\|_n : \|f\|_n \sim \|g\|_n$，使得复合*

$$|f(a)|_n \overset{\mathsf{nat}_n^f(a)^{-1}}{=\!=} \|f\|_n(|a|_n) \overset{\|h\|_n(|a|_n)}{=\!=} \|g\|_n(|a|_n) \overset{\mathsf{nat}_n^g(a)}{=\!=} |g(a)|_n \tag{7.3.6}$$

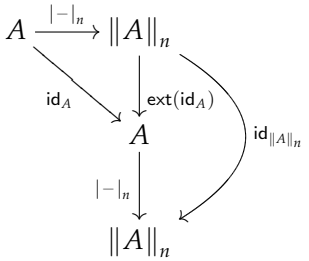
*等于 $\mathsf{ap}_{|-|_n}(h(a))$。*

*证明.* 首先，我们确实有一个同伦，其分量为 $\mathsf{ap}_{|-|_n}(h(a)) : |f(a)|_n = |g(a)|_n$。在两侧复合由 $\|f\|_n$ 和 $\|g\|_n$ 的定义产生的路径 $|f(a)|_n = \|f\|_n(|a|_n)$ 和 $|g(a)|_n = \|g\|_n(|a|_n)$，我们获得了一个同伦 $(\|f\|_n \circ |-|_n) \sim (\|g\|_n \circ |-|_n)$，因此根据函数外延性有一个等式。但是由于 $(- \circ |-|_n)$ 是一个等价，必须有一个路径 $\|f\|_n = \|g\|_n$ 来诱导它，并且函数外延性的相干性定律暗示了 (7.3.6)。 □

关于反射子范畴的以下观察也是标准的。

Corollary 7.3.7. *一个类型 $A$ 是一个 $n$-类型，当且仅当 $|-|_n : A \to \|A\|_n$ 是一个等价。*

*证明.* “如果” 这一点是由等价下的 $n$-类型的封闭性 (Closure of $n$-types under equivalence) 得出的。另一方面，如果 $A$ 是一个 $n$-类型，我们可以定义 $\mathsf{ext}(\mathsf{id}_A) : \|A\|_n \to A$。然后我们有 $\mathsf{ext}(\mathsf{id}_A) \circ |-|_n = \mathsf{id}_A : A \to A$ 根据定义。为了证明 $|-|_n \circ \mathsf{ext}(\mathsf{id}_A) = \mathsf{id}_{\|A\|_n}$，我们只需要证明 $|-|_n \circ \mathsf{ext}(\mathsf{id}_A) \circ |-|_n = \mathsf{id}_{\|A\|_n} \circ |-|_n$。这再次成立：




□

$n$-类型范畴 (Category of $n$-types) 还具有一些其他反射子范畴所不具备的特殊属性。例如，反射器 (Reflector) $\|-\|_n$ 保持有限积 (Finite Products)。

Theorem 7.3.8. *对于任意类型 $A$ 和 $B$，诱导映射 $\|A \times B\|_n \to \|A\|_n \times \|B\|_n$ 是一个等价。*

*证明.* 只需证明 $\|A\|_n \times \|B\|_n$ 具有与 $\|A \times B\|_n$ 相同的通用性质即可。因此，设 $C$ 是一个 $n$-类型；我们有

$$\begin{aligned}(\|A\|_n \times \|B\|_n \to C) &= (\|A\|_n \to (\|B\|_n \to C)) \\ &= (\|A\|_n \to (B \to C)) \\ &= (A \to (B \to C)) \\ &= (A \times B \to C)\end{aligned}$$

使用 $\|B\|_n$ 和 $\|A\|_n$ 的通用性质，以及 $B \to C$ 是一个 $n$-类型，因为 $C$ 是。可以很容易地验证，此等价通过与 $|-|_n \times |-|_n$ 的复合给出，这是所需要的。 □

以下关于依赖和 (Dependent Sum) 的相关事实通常也很有用。

Theorem 7.3.9. *令 $P : A \to \mathcal{U}$ 为一个类型族。则存在等价*

$$\left\|\sum_{x:A} \|P(x)\|_n\right\|_n \simeq \left\|\sum_{x:A} P(x)\right\|_n$$

*证明.* 我们多次使用 $n$-截断的归纳原理来构造函数

$$\varphi : \left\|\sum_{x:A} \|P(x)\|_n\right\|_n \to \left\|\sum_{x:A} P(x)\right\|_n$$
$$\psi : \left\|\sum_{x:A} P(x)\right\|_n \to \left\|\sum_{x:A} \|P(x)\|_n\right\|_n$$

以及将它们作为准逆 (Quasi-inverses) 展示的同伦 $H : \varphi \circ \psi \sim \mathsf{id}$ 和 $K : \psi \circ \varphi \sim \mathsf{id}$。我们通过设定 $\varphi(|(x, |u|_n)|_n) :\equiv |(x,u)|_n$ 来定义 $\varphi$。我们通过设定 $\psi(|(x,u)|_n) :\equiv |(x, |u|_n)|_n$ 来定义 $\psi$。然后我们定义 $H(|(x,u)|_n) :\equiv \mathsf{refl}_{|(x,u)|_n}$ 和 $K(|(x,|u|_n)|_n) :\equiv \mathsf{refl}_{|(x,|u|_n)|_n}$。 □

Corollary 7.3.10. *如果 $A$ 是一个 $n$-类型且 $P : A \to \mathcal{U}$ 是任意类型族，那么*

$$\sum_{a:A} \|P(a)\|_n \simeq \left\|\sum_{a:A} P(a)\right\|_n$$

*证明.* 如果 $A$ 是一个 $n$-类型，则上面的左侧类型已经是一个 $n$-类型，因此等价于其 $n$-截断；因此这是由 Theorem 7.3.9 推导出的。 □

我们可以使用与 §2.8 and ?? 中计算余积 (Coproduct) 和自然数时使用的相同方法（我们将在 Chapter 8 中用来计算同伦群）来描述截断的路径空间 (Path Spaces)。不出所料，$A$ 的 $(n+1)$-截断的路径空间是 $A$ 的路径空间的 $n$-截断。实际上，对于任意 $x, y : A$，存在一个标准映射

$$f : \|x =_A y\|_n \to \left(|x|_{n+1} =_{\|A\|_{n+1}} |y|_{n+1}\right) \tag{7.3.11}$$

定义为

$$f(|p|_n) :\equiv \mathsf{ap}_{|-|_{n+1}}(p)$$

此定义使用了 $\|-\|_n$ 的递归原理，这是正确的，因为 $\|A\|_{n+1}$ 是 $(n+1)$-截断的，因此 $f$ 的余域是 $n$-截断的。

Theorem 7.3.12. *对于任意 $A$ 和 $x, y : A$ 以及 $n \geq -2$，映射 (7.3.11) 是一个等价；因此我们有*

$$\|x =_A y\|_n \simeq \left(|x|_{n+1} =_{\|A\|_{n+1}} |y|_{n+1}\right)$$

**证明**. 证明是一个简单的编码-解码方法的应用 (Encode-decode method)：在以前的情形中，我们不能直接定义映射 (7.3.11) 的准逆 (Quasi-inverse)，因为没有办法对 $|x|_{n+1}$ 和 $|y|_{n+1}$ 之间的等式进行归纳。因此，相反，我们在广义其类型后，再有广义的 $\|A\|_{n+1}$ 的一般元素，而不是 $|x|_{n+1}$ 和 $|y|_{n+1}$。定义 $P : \|A\|_{n+1} \to \|A\|_{n+1} \to n\text{-}\mathsf{Type}$ 为

$$P(|x|_{n+1}, |y|_{n+1}) :\equiv \|x =_A y\|_n$$

此定义是正确的，因为 $\|x =_A y\|_n$ 是 $n$-截断的，而 $n$-Type 是 $(n+1)$-截断的根据 Theorem 7.1.11。现在对于每个 $u, v : \|A\|_{n+1}$，存在一个映射

$$\mathsf{decode} : P(u, v) \to \left(u =_{\|A\|_{n+1}} v\right)$$

对于 $u = |x|_{n+1}$ 和 $v = |y|_{n+1}$ 和 $p : x = y$ 定义为

$$\mathsf{decode}(|p|_n) :\equiv \mathsf{ap}_{|-|_{n+1}}(p)$$

由于 decode 的余域是 $n$-截断的，只需对 $u$ 和 $v$ 进行此形式的定义，然后它就是与以前相同的定义。我们还定义了一个函数

$$r : \prod_{u:\|A\|_{n+1}} P(u, u)$$

通过对 $u$ 的归纳，其中 $r(|x|_{n+1}) :\equiv |\mathsf{refl}_x|_n$。
现在我们可以定义逆映射 (Inverse map)

$$\mathsf{encode} : \left(u =_{\|A\|_{n+1}} v\right) \to P(u, v)$$

通过

$$\mathsf{encode}(p) :\equiv \mathsf{transport}^{v \mapsto P(u,v)}(p, r(u))$$

要证明复合映射

$$\left(u =_{\|A\|_{n+1}} v\right) \xrightarrow{\mathsf{encode}} P(u, v) \xrightarrow{\mathsf{decode}} \left(u =_{\|A\|_{n+1}} v\right)$$

是恒等函数 (Identity function)，通过路径归纳法，只需检查它对于 $\mathsf{refl}_u : u = u$ 的情况，在这种情况下，我们需要知道的是 $\mathsf{decode}(r(u)) = \mathsf{refl}_u$。但是由于这是一个 $(n-1)$-类型，因此也是一个 $(n+1)$-类型，我们可以假设 $u \equiv |x|_{n+1}$，在这种情况下，按照 $r$ 和 decode 的定义，它得以成立。最后，要证明

$$P(u, v) \xrightarrow{\mathsf{decode}} \left(u =_{\|A\|_{n+1}} v\right) \xrightarrow{\mathsf{encode}} P(u, v)$$

是恒等函数，由于该目标再次是一个 $(n-1)$-类型，我们可以假设 $u = |x|_{n+1}$ 和 $v = |y|_{n+1}$ 并且我们正在考虑某些 $p : x = y$ 的 $P(|x|_{n+1}, |y|_{n+1})$。然后我们有

$$\begin{aligned}
\mathsf{encode}(\mathsf{decode}(|p|_n)) &= \mathsf{encode}(\mathsf{ap}_{|-|_{n+1}}(p)) \\
&= \mathsf{transport}^{v \mapsto P(|x|_{n+1}, v)}(\mathsf{ap}_{|-|_{n+1}}(p), |\mathsf{refl}_x|_n) \\
&= \mathsf{transport}^{y \mapsto \|x=y\|_n}(p, |\mathsf{refl}_x|_n) \\
&= \left|\mathsf{transport}^{y \mapsto (x=y)}(p, \mathsf{refl}_x)\right|_n \\
&= |p|_n,
\end{aligned}$$

使用 Lemmas 2.3.10 and 2.3.11。(或者，我们可以在 $p$ 上进行路径归纳；在这种情况下，所需的等式将是判断上的 (Judgmentally)。) 这就完成了 decode 和 encode 是准逆的证明。给出的结果是 $u = |x|_{n+1}$ 和 $v = |y|_{n+1}$ 的特例。 □

Corollary 7.3.13. *令 $n \geq -2$ 并且 $(A, a)$ 是一个基点类型 (Pointed Type)。那么*

$$\left\|\Omega(A, a)\right\|_n = \Omega\left(\|(A, a)\|_{n+1}\right)$$

证明. 这是前述引理的一个特例，其中 $x = y = a$。 □

Corollary 7.3.14. 令 $n \geq -2$ 和 $k \geq 0$ 并且 $(A, a)$ 为一个基点类型。那么

$$\left\|\Omega^k(A,a)\right\|_n = \Omega^k\left(\|(A,a)\|_{n+k}\right)$$

证明. 通过对 $k$ 进行归纳，使用 $\Omega^k$ 的递归定义。 □

我们还观察到 “截断是累积的 (Cumulative)”：如果我们截断为 $n$-类型，然后再截断为 $k$-类型，其中 $k \leq n$，那么我们可能也可以直接截断为 $k$-类型。

Lemma 7.3.15. 令 $k, n \geq -2$ 并且 $k \leq n$ 和 $A : \mathcal{U}$。那么 $\|\|A\|_n\|_k = \|A\|_k$。

证明. 我们定义两个映射 $f : \|\|A\|_n\|_k \to \|A\|_k$ 和 $g : \|A\|_k \to \|\|A\|_n\|_k$ 通过

$$f(||a|_n|_k) :\equiv |a|_k \qquad \text{和} \qquad g(|a|_k) :\equiv ||a|_n|_k$$

映射 $f$ 是良好定义的，因为 $\|A\|_k$ 是 $k$-截断的，并且也是 $n$-截断的（因为 $k \leq n$），映射 $g$ 是良好定义的，因为 $\|\|A\|_n\|_k$ 是 $k$-截断的。
复合 $f \circ g : \|A\|_k \to \|A\|_k$ 满足 $(f \circ g)(|a|_k) = |a|_k$，因此 $f \circ g = \mathsf{id}_{\|A\|_k}$。同样地，我们有 $(g \circ f)(||a|_n|_k) = ||a|_n|_k$，因此 $g \circ f = \mathsf{id}_{\|\|A\|_n\|_k}$。 □

## 7.4 n-类型的余极限 (Colimits of $n$-types)

回忆一下，在 §6.8中，我们使用了高阶归纳类型 (higher inductive types) 来定义类型的推挤 (pushouts)，并证明了它们的泛性质 (universal property)。通常，一个 $n$-类型的（同伦）余极限 (homotopy colimit) 可能不再是 $n$-类型（参见 Exercise 7.2中的极端反例）。然而，如果我们对其进行 $n$-截断 (truncation)，我们将获得一个 $n$-类型，它满足相对于其他 $n$-类型的正确的泛性质。
在本节中，我们将证明这一点，特别是对于推挤（pushouts），这是余极限中最重要和最复杂的情况。回忆 §6.8中的以下定义。

Definition 7.4.1. 一个**广义图** (span)是一个五元组 $\mathscr{D} = (A, B, C, f, g)$，其中 $f : C \to A$ 和 $g : C \to B$。

$$\mathscr{D} = \quad \begin{array}{ccc} C & \xrightarrow{g} & B \\ {\scriptstyle f}\downarrow & & \\ A & & \end{array}$$

Definition 7.4.2. 给定一个广义图 $\mathscr{D} = (A, B, C, f, g)$ 和一个类型 $D$，一个**在以 $D$ 为基底的广义图 $\mathscr{D}$ 下的余锥** (cocone under $\mathscr{D}$ with base $D$) 是一个三元组 $(i, j, h)$ 其中 $i : A \to D$，$j : B \to D$，以及 $h : \prod_{(c:C)} i(f(c)) = j(g(c))$：

$$\begin{array}{ccc} C & \xrightarrow{g} & B \\ {\scriptstyle f}\downarrow & {\scriptstyle h}\swarrow & \downarrow{\scriptstyle j} \\ A & \xrightarrow[i]{} & D \end{array}$$

我们用 $\mathsf{cocone}_{\mathscr{D}}(D)$ 表示所有此类余锥的类型。

余锥的类型是（协变）函子性 (functorial) 的。例如，给定 $D, E$ 和一个映射 $t : D \to E$，有一个映射

$$\left\{ \begin{array}{ccc} \mathsf{cocone}_{\mathscr{D}}(D) & \longrightarrow & \mathsf{cocone}_{\mathscr{D}}(E) \\ c & \longmapsto & t \circ c \end{array} \right.$$

由以下定义：

$$t \circ (i, j, h) = (t \circ i, t \circ j, \mathsf{ap}_t \circ h).$$

并且，给定 $D, E, F$，映射 $t : D \to E$，$u : E \to F$ 和 $c : \mathsf{cocone}_{\mathscr{D}}(D)$，我们有

$$\mathsf{id}_D \circ c = c \tag{7.4.3}$$

$$(u \circ t) \circ c = u \circ (t \circ c). \tag{7.4.4}$$

Definition 7.4.5. 给定一个 $n$-类型的广义图 $\mathscr{D}$，一个 $n$-类型 $D$，以及一个余锥 $c : \mathsf{cocone}_{\mathscr{D}}(D)$，如果对于每个 $n$-类型 $E$，映射

$$\begin{cases} (D \to E) & \longrightarrow & \mathsf{cocone}_{\mathscr{D}}(E) \\ t & \longmapsto & t \circ c \end{cases}$$

是一个等价 (equivalence)，则称对 $(D, c)$ 是一个 $n$-**类型中的推挤** (pushout of $\mathscr{D}$ in $n$-types)。

为了构造 $n$-类型的推挤 (pushouts)，我们需要解释如何反射广义图 (spans) 和余锥 (cocones)。

Definition 7.4.6. 令

$$\mathscr{D} = \begin{array}{ccc} C & \xrightarrow{g} & B \\ {\scriptstyle f}\downarrow & & \\ A & & \end{array}$$

为一个广义图 (span)。我们将以下 $n$-类型的广义图称为 $\|\mathscr{D}\|_n$：

$$\|\mathscr{D}\|_n :\equiv \begin{array}{ccc} \|C\|_n & \xrightarrow{\|g\|_n} & \|B\|_n \\ {\scriptstyle \|f\|_n}\downarrow & & \\ \|A\|_n & & \end{array}$$

Definition 7.4.7. 令 $D : \mathcal{U}$ 和 $c = (i, j, h) : \mathsf{cocone}_{\mathscr{D}}(D)$。我们定义

$$\|c\|_n = (\|i\|_n, \|j\|_n, k) : \mathsf{cocone}_{\|\mathscr{D}\|_n}(\|D\|_n)$$

其中 $k$ 是复合同伦 (composite homotopy)

$$\|i\|_n \circ \|f\|_n \sim \|i \circ f\|_n \sim \|j \circ g\|_n \sim \|j\|_n \circ \|g\|_n$$

使用了 Lemma 7.3.5 和 $\|-\|_n$ 的函子性 (functoriality)。

我们现在观察到，从每个类型到其 $n$-截断的映射组合成了一个广义图的映射，其含义如下。

Definition 7.4.8. 令

$$\mathscr{D} = \begin{array}{ccc} C & \xrightarrow{g} & B \\ {\scriptstyle f}\downarrow & & \\ A & & \end{array} \qquad \text{and} \qquad \mathscr{D}' = \begin{array}{ccc} C' & \xrightarrow{g'} & B' \\ {\scriptstyle f'}\downarrow & & \\ A' & & \end{array}$$

是两个广义图 (spans)。一个**广义图的映射** (map of spans) $\mathscr{D} \to \mathscr{D}'$ 由函数 $\alpha : A \to A'$，$\beta : B \to B'$ 和 $\gamma : C \to C'$ 以及同伦 $\phi : \alpha \circ f \sim f' \circ \gamma$ 和 $\psi : \beta \circ g \sim g' \circ \gamma$ 组成。

因此，对于任意广义图 $\mathscr{D}$，我们有一个广义图的映射 $|-|_n^{\mathscr{D}} : \mathscr{D} \to \|\mathscr{D}\|_n$，由 $|-|_n^A$、$|-|_n^B$、$|-|_n^C$ 以及 (7.3.4)中的自然同伦 $\mathsf{nat}_n^f$ 和 $\mathsf{nat}_n^g$ 组成。

我们还需要知道广义图的映射是如何函子性地 (functorially) 表现的。即，如果 $(\alpha,\beta,\gamma,\phi,\psi):\mathscr{D}\to\mathscr{D}'$ 是广义图的映射，并且 $D$ 是任意类型，那么我们有

$$\begin{cases} \mathsf{cocone}_{\mathscr{D}'}(D) & \longrightarrow & \mathsf{cocone}_{\mathscr{D}}(D) \\ (i,j,h) & \longmapsto & (i\circ\alpha, j\circ\beta, k) \end{cases}$$

其中 $k:\prod_{(z:C)} i(\alpha(f(z)))=j(\beta(g(z)))$ 是复合

$$i(\alpha(f(z)))\overset{\mathsf{ap}_i(\phi)}{=\!=} i(f'(\gamma(z)))\overset{h(\gamma(z))}{=\!=} j(g'(\gamma(z)))\overset{\mathsf{ap}_j(\psi)}{=\!=} j(\beta(g(z))) \tag{7.4.9}$$

我们将此余锥表示为 $(i,j,h)\circ(\alpha,\beta,\gamma,\phi,\psi)$。此外，这种余锥的函子性行为与其他余锥的函子性是相容的：

Lemma 7.4.10. 设 $(\alpha,\beta,\gamma,\phi,\psi):\mathscr{D}\to\mathscr{D}'$ 和 $t:D\to E$，则以下图表是交换的：

$$\begin{array}{ccc} \mathsf{cocone}_{\mathscr{D}'}(D) & \xrightarrow{t\circ -} & \mathsf{cocone}_{\mathscr{D}'}(E) \\ \downarrow & & \downarrow \\ \mathsf{cocone}_{\mathscr{D}}(D) & \xrightarrow[t\circ -]{} & \mathsf{cocone}_{\mathscr{D}}(E) \end{array}$$

证明. 给定 $(i,j,h):\mathsf{cocone}_{\mathscr{D}'}(D)$，注意到两个复合都产生一个余锥，其前两个分量是 $t\circ i\circ\alpha$ 和 $t\circ j\circ\beta$。因此，剩下的就是验证同伦是相同的。对于右上复合，同伦是 (7.4.9)，其中 $(i,j,h)$ 替换为 $(t\circ i, t\circ j, \mathsf{ap}_t\circ h)$：

$$t\,i\,\alpha\,f\,z\overset{\mathsf{ap}_{t\circ i}(\phi)}{=\!=} t\,i\,f'\,\gamma\,z\overset{\mathsf{ap}_t(h(\gamma(z)))}{=\!=} t\,j\,g'\,\gamma\,z\overset{\mathsf{ap}_{t\circ j}(\psi)}{=\!=} t\,j\,\beta\,g\,z$$

（为了简洁，我们省略了函数参数周围的括号。）另一方面，对于左下复合，同伦是应用于 (7.4.9)的 $\mathsf{ap}_t$。由于 $\mathsf{ap}$ 尊重路径的连接，这是等同于

$$t\,i\,\alpha\,f\,z\overset{\mathsf{ap}_t(\mathsf{ap}_i(\phi))}{=\!=} t\,i\,f'\,\gamma\,z\overset{\mathsf{ap}_t(h(\gamma(z)))}{=\!=} t\,j\,g'\,\gamma\,z\overset{\mathsf{ap}_t(\mathsf{ap}_j(\psi))}{=\!=} t\,j\,\beta\,g\,z$$

但是 $\mathsf{ap}_t\circ\mathsf{ap}_i=\mathsf{ap}_{t\circ i}$，$j$ 的情况也是类似的，所以这两个同伦是相等的。 □

最后，请注意，由于我们使用 Lemma 7.3.5定义了 $\|c\|_n:\mathsf{cocone}_{\|\mathscr{D}\|_n}(\|D\|_n)$，附加条件 (7.3.6)意味着

$$|-|_n^D\circ c=\|c\|_n\circ|-|_n^{\mathscr{D}}. \tag{7.4.11}$$

对于任意 $c:\mathsf{cocone}_{\mathscr{D}}(D)$。现在我们可以证明所需的定理。

Theorem 7.4.12. 设 $\mathscr{D}$ 是一个广义图，$(D,c)$ 是其推挤。那么 $(\|D\|_n,\|c\|_n)$ 是 $n$-类型中的 $\|\mathscr{D}\|_n$ 的推挤。

证明. 设 $E$ 是一个 $n$-类型，并考虑下图：

$$\begin{array}{ccc} (\|D\|_n\to E) & \xrightarrow{-\circ|-|_n^D} & (D\to E) \\ {\scriptstyle -\circ\|c\|_n}\downarrow & & \downarrow{\scriptstyle -\circ c} \\ \mathsf{cocone}_{\|\mathscr{D}\|_n}(E) & \xrightarrow{-\circ|-|_n^{\mathscr{D}}} & \mathsf{cocone}_{\mathscr{D}}(E) \\ {\scriptstyle \ell_1}\uparrow & & \uparrow{\scriptstyle \ell_2} \\ (\|A\|_n\to E)\times_{(\|C\|_n\to E)}(\|B\|_n\to E) & \longrightarrow & (A\to E)\times_{(C\to E)}(B\to E) \end{array}$$

上面的水平箭头是等价的，因为 $E$ 是一个 $n$-类型，而 $-\circ c$ 是等价的，因为 $c$ 是一个推挤余锥。因此，通过三者取二性质 (2-out-of-3 property)，为了证明 $-\circ\|c\|_n$ 是等价的，足以证明上方的方形是交换的，并且中间的水平箭头是等价的。要证明上方的方形是交换的，令 $t : \|D\|_n \to E$；那么

$$\begin{aligned}\left(t\circ\|c\|_n\right)\circ|-|_n^{\mathscr{D}} &= t\circ\left(\|c\|_n\circ|-|_n^{\mathscr{D}}\right) && \text{(by Lemma 7.4.10)}\\ &= t\circ\left(|-|_n^{D}\circ c\right) && \text{(by (7.4.11))}\\ &= \left(t\circ|-|_n^{D}\right)\circ c && \text{(by (7.4.4))}\end{aligned}$$

为了证明中间的水平箭头是等价的，考虑下方的方形。两个下方的垂直箭头只是 $\mathsf{happly}$ 的应用：

$$\begin{aligned}\ell_1(i,j,p) &:\equiv (i,j,\mathsf{happly}(p))\\ \ell_2(i,j,p) &:\equiv (i,j,\mathsf{happly}(p))\end{aligned}$$

因此，由函数外延性 (function extensionality) 可知，它们是等价的。最低的水平箭头定义为

$$(i,j,p)\mapsto\left(i\circ|-|_n^A,\ \ j\circ|-|_n^B,\ \ q\right)$$

其中 $q$ 是复合

$$\begin{aligned}i\circ|-|_n^A\circ f &= i\circ\|f\|_n\circ|-|_n^C && \text{(by } \mathsf{funext}(\lambda z.\,\mathsf{ap}_i(\mathsf{nat}_n^f(z)))\text{)}\\ &= j\circ\|g\|_n\circ|-|_n^C && \text{(by } \mathsf{ap}_{-\circ|-|_n^C}(p)\text{)}\\ &= j\circ|-|_n^B\circ g && \text{(by } \mathsf{funext}(\lambda z.\,\mathsf{ap}_j(\mathsf{nat}_n^g(z)))\text{)}\end{aligned}$$

这是一个等价的映射，因为它是由广义图的等价诱导而来的。因此，由三者取二性质 (2-out-of-3)，足以证明下方的方形是交换的。但是，下方的两个复合在前两个分量上定义性地一致，因此足以证明，对于左下角的 $(i,j,p)$ 和 $C$ 中的 $z$，路径

$$\mathsf{happly}(q,z) : i(|f(z)|_n) = j(|g(z)|_n)$$

（其中 $q$ 如上所述）等于复合

$$\begin{aligned}i(|f(z)|_n) &= i(\|f\|_n(|z|_n)) && \text{(by } \mathsf{ap}_i(\mathsf{nat}_n^f(z))\text{)}\\ &= j(\|g\|_n(|z|_n)) && \text{(by } \mathsf{happly}(p,|z|_n)\text{)}\\ &= j(|g(z)|_n) && \text{(by } \mathsf{ap}_j(\mathsf{nat}_n^g(z))\text{)}\end{aligned}$$

然而，由于 $\mathsf{happly}$ 是函子性的，因此检查三个分量路径的等同性就足够了：

$$\begin{aligned}\mathsf{happly}(\mathsf{funext}(\lambda z.\,\mathsf{ap}_i(\mathsf{nat}_n^f(z))),z) &= \mathsf{ap}_i(\mathsf{nat}_n^f(z))\\ \mathsf{happly}(\mathsf{ap}_{-\circ|-|_n^C}(p),z) &= \mathsf{happly}(p,|z|_n)\\ \mathsf{happly}(\mathsf{funext}(\lambda z.\,\mathsf{ap}_j(\mathsf{nat}_n^g(z))),z) &= \mathsf{ap}_j(\mathsf{nat}_n^g(z))\end{aligned}$$

其中第一个和第三个只是 $\mathsf{happly}$ 是 $\mathsf{funext}$ 的拟逆，而第二个是关于 $\mathsf{happly}$ 和预合成 (precomposition) 的简单泛定理。 □

## 7.5 Connectedness (**连通性**)

一个 $n$-类型是指其在 $n$ 维以上没有有趣信息的类型。相对地，一个 $n$-*连通的类型* (n-connected type) 是指其在 $n$ 维以下没有有趣信息的类型。事实证明，研究函数的更一般的概念也是自然的。

Definition 7.5.1. 一个函数 $f : A \to B$ 被称为 $n$-**连通的** ($n$-connected) 如果对所有 $b : B$，类型 $\left\|\mathsf{fib}_f(b)\right\|_n$ 是收缩的 (contractible)：

$$\mathsf{conn}_n(f) :\equiv \prod_{b:B}\mathsf{isContr}(\left\|\mathsf{fib}_f(b)\right\|_n)$$

一个类型 $A$ 被称为 $n$-**连通的** ($n$-connected) 当且仅当唯一的函数 $A \to \mathbf{1}$ 是 $n$-连通的，即 $\|A\|_n$ 是收缩的。

因此，一个函数 $f : A \to B$ 是 $n$-连通的，当且仅当对每个 $b : B$，$\mathsf{fib}_f(b)$ 是 $n$-连通的。当然，每个函数都是 $(-2)$-连通的。在下一个层级上，我们有：

Lemma 7.5.2. *一个函数 $f$ 是 $(-1)$-连通的，当且仅当它在 §4.6的意义下是满射 (surjective)。*

*证明.* 我们定义 $f$ 为满射，如果对所有 $b$，$\left\|\mathsf{fib}_f(b)\right\|_{-1}$ 是可居住的 (inhabited)。但由于它是一个单纯命题 (mere proposition)，可居住性等价于收缩性。 □

因此，$n$-连通性可以被看作是一个强形式的满射性。在范畴论中，$(-1)$-连通性对应于对象上的本质满射性 (essential surjectivity)，而 $n$-连通性对应于对所有 $k \leq n+1$ 的 $k$-态射 (k-morphisms) 的本质满射性。
Lemma 7.5.2还意味着一个类型 $A$ 是 $(-1)$-连通的，当且仅当它是可居住的。当一个类型是 0-连通时，我们可以简单地说它是**连通的** (connected)，当它是 1-连通时，我们说它是**单连通的** (simply connected)。

Remark 7.5.3. 尽管我们对类型的 $n$-连通性的定义与同伦理论中的标准概念一致，但我们对函数的 $n$-连通性概念与经典同伦理论中的一种常见索引方式差一位。虽然我们说一个函数 $f$ 是 $n$-连通的，如果它的所有纤维 (fibers) 是 $n$-连通的，一些经典同伦理论家会称这样的函数为 $(n+1)$-连通的。（这归因于历史上对余纤维 (cofibers) 而非纤维的关注。）

现在我们观察一些关于连通映射 (connected maps) 的闭合性质。

Lemma 7.5.4. *假设 $g$ 是一个 $n$-连通函数 $f$ 的收缩 (retract)。那么 $g$ 是 $n$-连通的。*

*证明.* 这是 Lemma 4.7.3的直接推论。 □

Corollary 7.5.5. *如果 $g$ 与一个 $n$-连通函数 $f$ 是同伦的，那么 $g$ 是 $n$-连通的。*

Lemma 7.5.6. *假设 $f : A \to B$ 是 $n$-连通的。那么 $g : B \to C$ 是 $n$-连通的，当且仅当 $g \circ f$ 是 $n$-连通的。*

*证明.* 对于任意 $c : C$，我们有

$$\begin{aligned}
\left\|\mathsf{fib}_{g\circ f}(c)\right\|_n &\simeq \left\| \sum_{w:\mathsf{fib}_g(c)} \mathsf{fib}_f(\mathsf{pr}_1 w) \right\|_n && \text{(by Exercise 4.4)} \\
&\simeq \left\| \sum_{w:\mathsf{fib}_g(c)} \left\|\mathsf{fib}_f(\mathsf{pr}_1 w)\right\|_n \right\|_n && \text{(by Theorem 7.3.9)} \\
&\simeq \left\|\mathsf{fib}_g(c)\right\|_n && \text{(since } \left\|\mathsf{fib}_f(\mathsf{pr}_1 w)\right\|_n \text{ is contractible)}
\end{aligned}$$

因此，如果并且仅当 $\left\|\mathsf{fib}_{g\circ f}(c)\right\|_n$ 是收缩的，$\left\|\mathsf{fib}_g(c)\right\|_n$ 也是收缩的。 □

重要的是，$n$-连通函数可以等价地被表述为那些满足关于 $n$-类型归纳原理的函数。这一想法将直接引出我们在??中对 Freudenthal 悬吊定理的证明。

Lemma 7.5.7. *对于函数 $f : A \to B$ 和类型族 $P : B \to \mathcal{U}$，考虑以下函数：*

$$\lambda s.\, s \circ f : \left(\prod_{b:B} P(b)\right) \to \left(\prod_{a:A} P(f(a))\right)$$

*对于固定的 $f$ 和 $n \geq -2$，以下条件等价。*

(i) *$f$ 是 $n$-连通的。*
(ii) *对于每个 $P : B \to n\text{-}\mathsf{Type}$，映射 $\lambda s.\, s \circ f$ 是一个等价。*
(iii) *对于每个 $P : B \to n\text{-}\mathsf{Type}$，映射 $\lambda s.\, s \circ f$ 有一个截面 (section)。*

**证明.** 假设 $f$ 是 $n$-连通的，并且令 $P : B \to n\text{-}\mathsf{Type}$。那么我们有以下等价关系：

$$\begin{aligned}
\prod_{b:B} P(b) &\simeq \prod_{b:B} \left( \left\| \mathsf{fib}_f(b) \right\|_n \to P(b) \right) && \text{(since } \left\| \mathsf{fib}_f(b) \right\|_n \text{ is contractible)} \\
&\simeq \prod_{b:B} \left( \mathsf{fib}_f(b) \to P(b) \right) && \text{(since } P(b) \text{ is an } n\text{-type)} \\
&\simeq \prod_{(b:B)} \prod_{(a:A)} \prod_{(p:f(a)=b)} P(b) && \text{(by the left universal property of } \Sigma\text{-types)} \\
&\simeq \prod_{a:A} P(f(a)) && \text{(by the left universal property of path types)}
\end{aligned}$$

我们省略了证明此等价关系确实由 $\lambda s.\, s \circ f$ 给出。因此，(i)⇒(ii)显然，(ii)⇒(iii)也显然成立。为了证明(iii)⇒(i)，考虑类型族

$$P(b) :\equiv \left\| \mathsf{fib}_f(b) \right\|_n$$

那么(iii)给出了一个映射 $c : \prod_{(b:B)} \left\| \mathsf{fib}_f(b) \right\|_n$，使得 $c(f(a)) = \left| (a, \mathsf{refl}_{f(a)}) \right|_n$。为了证明每个 $\left\| \mathsf{fib}_f(b) \right\|_n$ 都是收缩的，我们将找到一个类型为

$$\prod_{(b:B)} \prod_{(w:\left\| \mathsf{fib}_f(b) \right\|_n)} w = c(b)$$

的函数。根据 Theorem 7.3.2，为此目的，我们只需找到一个类型为

$$\prod_{(b:B)} \prod_{(a:A)} \prod_{(p:f(a)=b)} |(a,p)|_n = c(b)$$

的函数。但通过变量重新安排和路径归纳，这等价于类型

$$\prod_{a:A} \left| (a, \mathsf{refl}_{f(a)}) \right|_n = c(f(a))$$

该属性通过我们对 $c(f(a))$ 的选择成立。 □

Corollary 7.5.8. *对于任意 $A$，从类型到其 $n$-截断 (truncation) 的标准函数 $|-|_n : A \to \|A\|_n$ 是 $n$-连通的。*

**证明.** 根据 Theorem 7.3.2及其相关的唯一性原则，Lemma 7.5.7中的条件成立。 □

例如，当 $n = -1$ 时，Corollary 7.5.8表明从一个类型到其命题截断 (propositional truncation) 的映射 $A \to \|A\|$ 是满射的。

Corollary 7.5.9. *一个类型 $A$ 是 $n$-连通的，当且仅当映射*

$$\lambda b.\, \lambda a.\, b : B \to (A \to B)$$

*对于每个 $n$-类型 $B$ 都是一个等价关系。换句话说，“从 $A$ 到 $n$-类型的每个映射都是常数”。*

**证明.** 通过应用 Lemma 7.5.7到以 $\mathbf{1}$ 为终端类型的函数。 □

Lemma 7.5.10. *设 $B$ 是一个 $n$-类型，并且 $f : A \to B$ 是一个函数。那么诱导的函数 $g : \|A\|_n \to B$ 是一个等价关系，当且仅当 $f$ 是 $n$-连通的。*

**证明.** 根据 Corollary 7.5.8，$|-|_n$ 是 $n$-连通的。因此，由于 $f = g \circ |-|_n$，根据 Lemma 7.5.6，如果且仅当 $g$ 是 $n$-连通的，$f$ 才是 $n$-连通的。但由于 $g$ 是一个 $n$-类型之间的函数，其纤维 (fiber) 也是 $n$-类型。因此，$g$ 是 $n$-连通的，当且仅当它是一个等价关系。 □

我们还可以通过包含其基点 (basepoint) 的连通性来表征连通的指点类型 (pointed types)。

Lemma 7.5.11. 设 $A$ 是一个类型，$a_0 : \mathbf{1} \to A$ 是一个基点，且 $n \geq -1$。那么 $A$ 是 $n$-连通的，当且仅当映射 $a_0$ 是 $(n-1)$-连通的。

证明. 首先假设 $a_0 : \mathbf{1} \to A$ 是 $(n-1)$-连通的，并且令 $B$ 是一个 $n$-类型；我们将使用 Corollary 7.5.9。映射 $\lambda b.\lambda a.\, b : B \to (A \to B)$ 有一个回缩 (retraction)，即 $f \mapsto f(a_0)$，因此只需证明它也有一个截面 (section)，即对于任何 $f : A \to B$，存在 $b : B$ 使得 $f = \lambda a.\, b$。我们选择 $b :\equiv f(a_0)$。定义 $P : A \to \mathcal{U}$ 为 $P(a) :\equiv (f(a) = f(a_0))$。然后 $P$ 是一个 $(n-1)$-类型族，并且我们有 $P(a_0)$；因此我们有 $\prod_{(a:A)} P(a)$，因为 $a_0 : \mathbf{1} \to A$ 是 $(n-1)$-连通的。因此，$f = \lambda a.\, f(a_0)$，如所需。
现在假设 $A$ 是 $n$-连通的，并且给定 $P : A \to (n-1)$-$\mathsf{Type}$ 和 $u : P(a_0)$。通过 Lemma 7.5.7，我们将构造一个 $f : \prod_{(a:A)} P(a)$，使得 $f(a_0) = u$。现在，$(n-1)$-$\mathsf{Type}$ 是一个 $n$-类型，并且 $A$ 是 $n$-连通的，因此通过 Corollary 7.5.9，存在一个 $n$-类型 $B$ 使得 $P = \lambda a.\, B$。因此，我们有一组等价关系 $g : \prod_{(a:A)}(P(a) \simeq B)$。定义 $f(a) :\equiv {g_a}^{-1}(g_{a_0}(u))$；那么 $f : \prod_{(a:A)} P(a)$ 并且 $f(a_0) = u$，如所需。 □

特别地，一个指点类型 $(A, a_0)$ 是 0-连通的，当且仅当 $a_0 : \mathbf{1} \to A$ 是满射的，也就是说 $\prod_{(x:A)} \|x = a_0\|$。对于非指点情况的类似结果，请参见 Exercise 7.6。
Lemma 7.5.6的一个有用变体是：

Lemma 7.5.12. 设 $f : A \to B$ 是一个函数，并且 $P : A \to \mathcal{U}$ 和 $Q : B \to \mathcal{U}$ 是类型族。假设 $g : \prod_{(a:A)} P(a) \to Q(f(a))$ 是一个纤维上 (fiberwise) 的 $n$-连通函数族，亦即每个函数 $g_a : P(a) \to Q(f(a))$ 都是 $n$-连通的。如果 $f$ 也是 $n$-连通的，那么函数

$$\varphi : \Big(\sum_{a:A} P(a)\Big) \to \Big(\sum_{b:B} Q(b)\Big)$$
$$\varphi(a,u) :\equiv (f(a), g_a(u))$$

也是 $n$-连通的。反之，如果 $\varphi$ 和每个 $g_a$ 都是 $n$-连通的，并且此外 $Q$ 在纤维上是单射的（即我们有 $\|Q(b)\|$ 对于所有 $b : B$），那么 $f$ 是 $n$-连通的。

证明. 对于任意 $b : B$ 和 $v : Q(b)$，我们有

$$\begin{aligned}
\big\|\mathsf{fib}_\varphi((b,v))\big\|_n &\simeq \Big\| \sum_{(w:\mathsf{fib}_f(b))} \sum_{(u:P(\mathsf{pr}_1(w)))} g_{\mathsf{pr}_1 w}(u) = \mathsf{pr}_2(w)^{-1}{}_*(v) \Big\|_n \\
&\simeq \Big\| \sum_{w:\mathsf{fib}_f(b)} \Big\| \mathsf{fib}_{g(\mathsf{pr}_1 w)}(\mathsf{pr}_2(w)^{-1}{}_*(v)) \Big\|_n \Big\|_n \\
&\simeq \big\|\mathsf{fib}_f(b)\big\|_n
\end{aligned}$$

其中传输 (transportation) 是沿着 $f(p)$ 和 $f(p)^{-1}$ 关于 $Q$ 进行的。因此，如果其中任何一个是可收缩的，那么另一个也是可收缩的。
特别地，如果 $f$ 是 $n$-连通的，那么对于所有的 $b : B$，$\big\|\mathsf{fib}_f(b)\big\|_n$ 是可收缩的，因此对于所有 $(b,v) : \sum_{(b:B)} Q(b)$，$\big\|\mathsf{fib}_\varphi((b,v))\big\|_n$ 也是可收缩的。另一方面，如果 $\varphi$ 是 $n$-连通的，那么对于所有 $(b,v)$，$\big\|\mathsf{fib}_\varphi((b,v))\big\|_n$ 是可收缩的，因此对于任何存在 $v : Q(b)$ 的 $b : B$，$\big\|\mathsf{fib}_f(b)\big\|_n$ 也是可收缩的。最后，由于可收缩性是一个单纯命题，因此我们只需拥有这样的 $v$。 □

Lemma 7.5.12的逆向结论如果 $Q$ 在纤维上不是单射的可能不成立。例如，如果 $P$ 和 $Q$ 都恒为 $\mathbf{0}$，那么 $\varphi$ 和每个 $g_a$ 都是等价关系，但 $f$ 可能是任意的。
另一方面，我们有

Lemma 7.5.13. 设 $P, Q : A \to \mathcal{U}$ 是类型族，并且考虑从 $P$ 到 $Q$ 的纤维转换

$$f : \prod_{a:A} \Big(P(a) \to Q(a)\Big)$$

则诱导的映射 $\mathsf{total}(f) : \sum_{(a:A)} P(a) \to \sum_{(a:A)} Q(a)$ 是 $n$-连通的，当且仅当每个 $f(a)$ 是 $n$-连通的。

当然，“仅当”的方向也是 Lemma 7.5.12的一个特例。

证明. 根据 Theorem 4.7.6，我们有 $\mathsf{fib}_{\mathsf{total}(f)}((x,v)) \simeq \mathsf{fib}_{f(x)}(v)$ 对于每个 $x : A$ 和 $v : Q(x)$。因此，$\left\|\mathsf{fib}_{\mathsf{total}(f)}((x,v))\right\|_n$ 是可收缩的，当且仅当 $\left\|\mathsf{fib}_{f(x)}(v)\right\|_n$ 是可收缩的。 □

关于连通映射的另一个有用事实是它们在 $n$-截断上诱导了等价关系：

Lemma 7.5.14. 如果 $f : A \to B$ 是 $n$-连通的，那么它诱导了一个等价关系 $\|A\|_n \simeq \|B\|_n$。

证明. 设 $c$ 是证明 $f$ 是 $n$-连通的证明。从左到右，我们使用映射 $\|f\|_n : \|A\|_n \to \|B\|_n$。为了定义从右到左的映射，根据截断的通用性质，我们只需给出一个映射 $\mathsf{back} : B \to \|A\|_n$。我们可以如下定义此映射：

$$\mathsf{back}(y) :\equiv \|\mathsf{pr}_1\|_n(\mathsf{pr}_1(c(y)))$$

根据定义，$c(y)$ 的类型是 $\mathsf{isContr}(\left\|\mathsf{fib}_f(y)\right\|_n)$，所以它的第一个分量的类型是 $\left\|\mathsf{fib}_f(y)\right\|_n$，我们可以通过投影从中获得 $\|A\|_n$ 的一个元素。
接下来，我们证明复合是恒等映射。在两个方向上，由于目标是在一个 $n$-截断类型中的路径，因此只需处理构造函数 $|-|_n$。
在一个方向上，我们必须证明对于所有 $x : A$，

$$\|\mathsf{pr}_1\|_n(\mathsf{pr}_1(c(f(x)))) = |x|_n$$

但是 $\left|(x, \mathsf{refl}_{f(x)})\right|_n$ 的类型是 $\left\|\mathsf{fib}_f(f(x))\right\|_n$，并且 $c(f(x))$ 表明该类型是可收缩的，所以

$$\mathsf{pr}_1(c(f(x))) = |(x, \mathsf{refl})|_n$$

将 $\|\mathsf{pr}_1\|_n$ 应用于此等式的两边给出所需的结果。
在另一个方向上，我们必须证明对于所有 $y : B$，

$$\|f\|_n(\|\mathsf{pr}_1\|_n(\mathsf{pr}_1(c(y)))) = |y|_n$$

$\mathsf{pr}_1(c(y))$ 的类型是 $\left\|\mathsf{fib}_f(y)\right\|_n$，并且我们想要的路径基本上是 $\mathsf{fib}_f(y)$ 的第二个分量，但我们需要确保截断结果正确。
一般来说，假设我们给定了 $p : \left\|\sum_{(x:A)} B(x)\right\|_n$ 并且希望证明 $P(\|\mathsf{pr}_1\|_n(p))$。通过截断归纳，只需证明对于所有 $a : A$ 和 $b : B(a)$，$P(|a|_n)$ 成立。在此情况下应用这一原则，只需证明

$$\|f\|_n(|a|_n) = |y|_n$$

给定 $a : A$ 和 $b : f(a) = y$。但左边等于 $|f(a)|_n$，所以将 $|-|_n$ 应用于 $b$ 的两边给出所需的结果。 □

人们可能会猜测这个事实刻画了 $n$-连通映射，但实际上，成为 $n$-连通的条件比这更强。例如，包含映射 $0_{\mathbf{2}} : \mathbf{1} \to \mathbf{2}$ 在 $(-1)$-截断上诱导了一个等价关系，但并不是满射的（即 $(-1)$-连通的）。在??中，我们将看到一般情况下的区别类似于额外的满射性。

## 7.6 正交分解 (Orthogonal Factorization)

在集合论中，满射 (surjection) 和单射 (injection) 组成了一个唯一的分解系统：每个函数都可以实质上唯一地分解为一个满射后接一个单射。我们已经看到，满射可以自然地推广为 $n$-连通 (n-connected) 映射，因此我们自然要探讨这些映射是否也参与到一个分解系统中。下面是关于单射的对应推广。

Definition 7.6.1. 一个函数 $f : A \to B$ 是 **$n$-截断的** ($n$-truncated) 如果对所有 $b : B$，纤维 $\mathsf{fib}_f(b)$ 是一个 $n$-类型 (n-type)。

特别地，$f$ 是 $(-2)$-截断的当且仅当它是一个等价 (equivalence)。并且，$A$ 是一个 $n$-类型当且仅当 $A \to \mathbf{1}$ 是 $n$-截断的。此外，$n$-截断映射也可以递归地定义，类似于 $n$-类型。

Lemma 7.6.2. 对于任何 $n \geq -2$，一个函数 $f : A \to B$ 是 $(n+1)$-截断的当且仅当对所有 $x, y : A$，映射 $\mathsf{ap}_f : (x = y) \to (f(x) = f(y))$ 是 $n$-截断的。特别地，$f$ 是 $(-1)$-截断的当且仅当它是 §4.6 中定义的嵌入。

证明. 注意，对于 $\mathsf{fib}_f(b)$ 中的任意 $(x,p),(y,q)$，我们有

$$\begin{aligned}\big((x,p) = (y,q)\big) &= \sum_{r:x=y} (p = \mathsf{ap}_f(r) \cdot q) \\ &= \sum_{r:x=y} (\mathsf{ap}_f(r) = p \cdot q^{-1}) \\ &= \mathsf{fib}_{\mathsf{ap}_f}(p \cdot q^{-1}).\end{aligned}$$

因此，$f$ 的任何纤维中的任何路径空间都是 $\mathsf{ap}_f$ 的一个纤维。另一方面，通过选择 $b :\equiv f(y)$ 和 $q :\equiv \mathsf{refl}_{f(y)}$ 我们可以看到 $\mathsf{ap}_f$ 的任何纤维都是 $f$ 的某个纤维中的路径空间。结果由此得出，因为如果 $f$ 的所有纤维的路径空间都是 $n$-类型，那么 $f$ 是 $(n+1)$-截断的。 □

现在我们可以通过一种显而易见的方式来构造分解。

Definition 7.6.3. 令 $f : A \to B$ 是一个函数。$f$ 的 **$n$-像** ($n$-image) 定义为

$$\mathsf{im}_n(f) :\equiv \sum_{b:B} \left\|\mathsf{fib}_f(b)\right\|_n.$$

当 $n = -1$ 时，我们简单地写作 $\mathsf{im}(f)$ 并称其为 $f$ 的 **像** (image)。

Lemma 7.6.4. 对于任何函数 $f : A \to B$，规范函数 $\tilde{f} : A \to \mathsf{im}_n(f)$ 是 $n$-连通的。因此，任何函数都可以分解为一个 $n$-连通的函数后接一个 $n$-截断的函数。

证明. 注意 $A \simeq \sum_{(b:B)} \mathsf{fib}_f(b)$。函数 $\tilde{f}$ 是由规范的纤维变换

$$\prod_{b:B} \left(\mathsf{fib}_f(b) \to \left\|\mathsf{fib}_f(b)\right\|_n\right)$$

在总空间上诱导出的函数。由于每个映射 $\mathsf{fib}_f(b) \to \left\|\mathsf{fib}_f(b)\right\|_n$ 通过 Corollary 7.5.8 是 $n$-连通的，$\tilde{f}$ 通过 Lemma 7.5.13 也是 $n$-连通的。最后，投影 $\mathsf{pr}_1 : \mathsf{im}_n(f) \to B$ 是 $n$-截断的，因为它的纤维与 $f$ 的纤维的 $n$-截断是等价的。 □

在下面的引理中，我们设置了一些机制来证明唯一分解定理。

Lemma 7.6.5. 假设我们有一个函数的交换图

$$\begin{array}{ccc} A & \xrightarrow{g_1} & X_1 \\ {\scriptstyle g_2}\downarrow & & \downarrow{\scriptstyle h_1} \\ X_2 & \xrightarrow[h_2]{} & B \end{array}$$

其中 $H : h_1 \circ g_1 \sim h_2 \circ g_2$，$g_1$ 和 $g_2$ 是 $n$-连通的，$h_1$ 和 $h_2$ 是 $n$-截断的。那么存在一个等价

$$E(H,b) : \mathsf{fib}_{h_1}(b) \simeq \mathsf{fib}_{h_2}(b)$$

对于任意 $b : B$，使得对于任何 $a : A$ 我们有一个同一性

$$\overline{E}(H,a) : E(H, h_1(g_1(a)))(g_1(a), \mathsf{refl}_{h_1(g_1(a))}) = (g_2(a), H(a)^{-1})$$

证明. 令 $b : B$。然后我们有以下等价：

$$\begin{aligned}
\mathsf{fib}_{h_1}(b) &\simeq \sum_{w:\mathsf{fib}_{h_1}(b)} \left\|\mathsf{fib}_{g_1}(\mathsf{pr}_1 w)\right\|_n && \text{(因为 } g_1 \text{ 是 } n\text{-连通的)}\\
&\simeq \left\| \sum_{w:\mathsf{fib}_{h_1}(b)} \mathsf{fib}_{g_1}(\mathsf{pr}_1 w)\right\|_n && \text{(由 Corollary 7.3.10，因为 } h_1 \text{ 是 } n\text{-截断的)}\\
&\simeq \left\|\mathsf{fib}_{h_1\circ g_1}(b)\right\|_n && \text{(由 Exercise 4.4)}
\end{aligned}$$

对 $h_2$ 和 $g_2$ 也同样成立。此外，由于我们有一个同伦 $H : h_1 \circ g_1 \sim h_2 \circ g_2$，存在一个显然的等价 $\mathsf{fib}_{h_1\circ g_1}(b) \simeq \mathsf{fib}_{h_2\circ g_2}(b)$。因此我们得到

$$\mathsf{fib}_{h_1}(b) \simeq \mathsf{fib}_{h_2}(b)$$

对于任何 $b : B$。通过分析底层函数，我们得出在应用每个等价 $E$ 组成的映射后，$(g_1(a), \mathsf{refl}_{h_1(g_1(a))})$ 发生的变化。一些同一性是定义上的，但其他的（在下面用 $=$ 标记）是命题上的；将它们组合在一起，我们得到了 $\overline{E}(H, a)$。

$$\begin{aligned}
(g_1(a), \mathsf{refl}_{h_1(g_1(a))}) &\overset{=}{\mapsto} \left((g_1(a), \mathsf{refl}_{h_1(g_1(a))}), \left|(a, \mathsf{refl}_{g_1(a)})\right|_n\right)\\
&\mapsto \left|((g_1(a), \mathsf{refl}_{h_1(g_1(a))}), (a, \mathsf{refl}_{g_1(a)}))\right|_n\\
&\mapsto \left|(a, \mathsf{refl}_{h_1(g_1(a))})\right|_n\\
&\overset{=}{\mapsto} \left|(a, H(a)^{-1})\right|_n\\
&\mapsto \left|((g_2(a), H(a)^{-1}), (a, \mathsf{refl}_{g_2(a)}))\right|_n\\
&\mapsto \left((g_2(a), H(a)^{-1}), \left|(a, \mathsf{refl}_{g_2(a)})\right|_n\right)\\
&\mapsto (g_2(a), H(a)^{-1})
\end{aligned}$$

第一个等式是因为对于一般的 $b$，映射 $\mathsf{fib}_{h_1}(b) \to \sum_{(w:\mathsf{fib}_{h_1}(b))} \left\|\mathsf{fib}_{g_1}(\mathsf{pr}_1 w)\right\|_n$ 插入了通过假设 $g_1$ 是 $n$-截断的来提供的 $\left\|\mathsf{fib}_{g_1}(\mathsf{pr}_1 w)\right\|_n$ 的收缩中心；而在这种情况下，该类型有一个显而易见的居留元 $\left|(a, \mathsf{refl}_{g_1(a)})\right|_n$，通过收缩性必须等于该中心。第二个命题等式是因为等价 $\mathsf{fib}_{h_1\circ g_1}(b) \simeq \mathsf{fib}_{h_2\circ g_2}(b)$ 将第二个分量与 $H(a)^{-1}$ 连接在一起，而我们有 $H(a)^{-1} \cdot \mathsf{refl} = H(a)^{-1}$。读者可以检查其他的等式是定义上的（假设 Exercise 4.4 有合理的解）。 □

结合 Lemmas 7.6.4 and 7.6.5，我们有以下唯一分解结果：

Theorem 7.6.6. 对于每个函数 $f : A \to B$，空间 $\mathsf{fact}_n(f)$ 定义为

$$\sum_{(X:\mathcal{U})} \sum_{(g:A\to X)} \sum_{(h:X\to B)} (h \circ g \sim f) \times \mathsf{conn}_n(g) \times \mathsf{trunc}_n(h)$$

是收缩的。它的收缩中心是由 Lemma 7.6.4 得到的元素

$$(\mathsf{im}_n(f), \tilde{f}, \mathsf{pr}_1, \theta, \varphi, \psi) : \mathsf{fact}_n(f)$$

其中 $\theta : \mathsf{pr}_1 \circ \tilde{f} \sim f$ 是规范的同伦，$\varphi$ 是 Lemma 7.6.4 的证明，$\psi$ 是证明 $\mathsf{pr}_1 : \mathsf{im}_n(f) \to B$ 的纤维是 $n$-截断的显然的证明。

证明. 通过 Lemma 7.6.4 我们知道存在 $\mathsf{fact}_n(f)$ 的一个元素，因此只需证明 $\mathsf{fact}_n(f)$ 是一个纯命题。假设我们有两个 $n$-分解

$$(X_1, g_1, h_1, H_1, \varphi_1, \psi_1) \qquad \text{和} \qquad (X_2, g_2, h_2, H_2, \varphi_2, \psi_2)$$

那么我们有点按连接的同伦

$$H :\equiv (\lambda a.\, H_1(a) \cdot H_2^{-1}(a)) \;:\; (h_1 \circ g_1 \sim h_2 \circ g_2)$$

通过一致性和在 $\Sigma$ 类型、函数类型和路径类型中的路径和运输的特性，证明以下四点即可：

(i) 存在一个等价 $e : X_1 \simeq X_2$，
(ii) 存在一个同伦 $\zeta : e \circ g_1 \sim g_2$，
(iii) 存在一个同伦 $\eta : h_2 \circ e \sim h_1$，
(iv) 对于任意 $a : A$，我们有 $\mathsf{ap}_{h_2}(\zeta(a))^{-1} \cdot \eta(g_1(a)) \cdot H_1(a) = H_2(a)$。

我们按顺序证明这四个断言：

(i) 通过 Lemma 7.6.5，我们有一个纤维等价

$$E(H) : \prod_{b:B} \mathsf{fib}_{h_1}(b) \simeq \mathsf{fib}_{h_2}(b).$$

这诱导了总空间的等价，即我们有

$$\Big(\sum_{b:B} \mathsf{fib}_{h_1}(b)\Big) \simeq \Big(\sum_{b:B} \mathsf{fib}_{h_2}(b)\Big)$$

当然，通过 Lemma 4.8.2，我们也有等价 $X_1 \simeq \sum_{(b:B)} \mathsf{fib}_{h_1}(b)$ 和 $X_2 \simeq \sum_{(b:B)} \mathsf{fib}_{h_2}(b)$。这给了我们等价 $e : X_1 \simeq X_2$；读者可以验证 $e$ 的底层函数由下式给出

$$e(x) \equiv \mathsf{pr}_1(E(H, h_1(x))(x, \mathsf{refl}_{h_1(x)}))$$

(ii) 通过 Lemma 7.6.5，我们可以选择 $\zeta(a) :\equiv \mathsf{ap}_{\mathsf{pr}_1}(\overline{E}(H, a)) : e(g_1(a)) = g_2(a)$。
(iii) 对于每个 $x : X_1$，我们有

$$\mathsf{pr}_2(E(H, h_1(x))(x, \mathsf{refl}_{h_1(x)})) : h_2(e(x)) = h_1(x)$$

给了我们一个同伦 $\eta : h_2 \circ e \sim h_1$。
(iv) 通过纤维中的路径的特性 (Lemma 4.2.5)，来自 Lemma 7.6.5 的路径 $\overline{E}(H, a)$ 给出了 $\eta(g_1(a)) = \mathsf{ap}_{h_2}(\zeta(a)) \cdot H(a)^{-1}$。所需的等式通过替换 $H$ 的定义并重新排列路径得出。 □

通过标准的论证，这产生了以下正交性原则。

Theorem 7.6.7. *设 $e : A \to B$ 是 $n$-连通的，并且 $m : C \to D$ 是 $n$-截断的。则映射*

$$\varphi : (B \to C) \to \sum_{(h:A\to C)} \sum_{(k:B\to D)} (m \circ h \sim k \circ e)$$

*是一个等价。*

*证明简述.* 对于任意的 $(h, k, H)$ 在上面的域中，设 $h = h_2 \circ h_1$ 且 $k = k_2 \circ k_1$，其中 $h_1$ 和 $k_1$ 是 $n$-连通的，而 $h_2$ 和 $k_2$ 是 $n$-截断的。则 $f = (m \circ h_2) \circ h_1$ 和 $f = k_2 \circ (k_1 \circ e)$ 都是 $m \circ h = k \circ e$ 的 $n$-分解。因此，它们之间存在唯一的等价。从中提取 $\mathsf{fib}_\varphi((h, k, H))$ 是收缩的，这是显而易见的。 □

我们最后展示像在拉回下是稳定的。

Lemma 7.6.8. *假设以下方块*

$$\begin{array}{ccc} A & \longrightarrow & C \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[h]{} & D \end{array}$$

*是一个拉回方块，并且令 $b : B$。则 $\mathsf{fib}_f(b) \simeq \mathsf{fib}_g(h(b))$。*

证明. 这来自拉回的粘贴 (Exercise 2.12)，因为下图中的类型 $X$

$$\begin{array}{ccccc} X & \longrightarrow & A & \longrightarrow & C \\ \downarrow & & \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle g} \\ \mathbf{1} & \xrightarrow[b]{} & B & \xrightarrow[h]{} & D \end{array}$$

是左边方块的拉回当且仅当它是外矩形的拉回，而 $\mathsf{fib}_f(b)$ 是左边方块的拉回，$\mathsf{fib}_g(h(b))$ 是外矩形的拉回。 □

Theorem 7.6.9. 考虑函数 $f : A \to B$, $g : C \to D$ 和下图

$$\begin{array}{ccc} A & \longrightarrow & C \\ {\scriptstyle \tilde{f}_n}\downarrow & & \downarrow{\scriptstyle \tilde{g}_n} \\ \mathsf{im}_n(f) & \longrightarrow & \mathsf{im}_n(g) \\ {\scriptstyle \mathsf{pr}_1}\downarrow & & \downarrow{\scriptstyle \mathsf{pr}_1} \\ B & \xrightarrow[h]{} & D \end{array}$$

如果外矩形是一个拉回，那么下方方块也是（因此通过 Exercise 2.12 上方方块也是）。因此，像在拉回下是稳定的。

证明. 假设外方块是一个拉回，我们有以下等价

$$\begin{aligned} &\equiv \sum_{(b:B)} \sum_{(w:\mathsf{im}_n(g))} h(b) = \mathsf{pr}_1 w \\ &\simeq \sum_{(b:B)} \sum_{(d:D)} \sum_{(w:\left\|\mathsf{fib}_g(d)\right\|_n)} h(b) = d \\ &\simeq \sum_{b:B} \left\|\mathsf{fib}_g(h(b))\right\|_n \\ &\simeq \sum_{b:B} \left\|\mathsf{fib}_f(b)\right\|_n && \text{(通过 Lemma 7.6.8)} \\ &\equiv \mathsf{im}_n(f) \end{aligned}$$

□

## 7.7 模态 (Modalities)

几乎所有关于 $n$-类型 ($n$-types) 和连通性 (connectedness) 的理论都可以在更广泛的范围内进行研究。本节内容在本书的其他部分中不会被使用。
我们首先想到的关于推广 $n$-类型理论的方法可能是将 Lemma 7.3.3 作为定义。

Definition 7.7.1. **反射子宇宙** (reflective subuniverse) 是谓词 $P : \mathcal{U} \to \mathsf{Prop}$，使得对于每个 $A : \mathcal{U}$，我们都有一个类型 $\bigcirc A$，使得 $P(\bigcirc A)$ 成立，并且有一个映射 $\eta_A : A \to \bigcirc A$，其性质是对于每个 $P(B)$ 成立的 $B : \mathcal{U}$，以下映射是一个等价：

$$\begin{cases} (\bigcirc A \to B) & \longrightarrow & (A \to B) \\ f & \longmapsto & f \circ \eta_A \end{cases}.$$

我们写作 $\mathcal{U}_P :\equiv \{\, A : \mathcal{U} \mid P(A) \,\}$，因此 $A : \mathcal{U}_P$ 意味着 $A : \mathcal{U}$ 并且 $P(A)$ 成立。我们也用 $\mathsf{rec}_\bigcirc$ 表示上述映射的准逆 (quasi-inverse)。尽管 $\bigcirc$ 的符号可能看起来有些奇怪，但它很快就会变得清晰。
对于任何反射子宇宙，我们可以以通常的方式证明从范畴论 (category theory) 中反射子类 (reflective subcategories) 的所有熟知事实。例如，我们有：

- 一个类型 $A$ 属于 $\mathcal{U}_P$ 当且仅当 $\eta_A : A \to \bigcirc A$ 是一个等价。
- $\mathcal{U}_P$ 在收缩 (retracts) 下封闭。特别地，当 $\eta_A$ 允许一个收缩时，$A$ 就属于 $\mathcal{U}_P$。
- 在适合的逐点一致同伦 (up-to-coherent-homotopy) 意义下，$\bigcirc$ 是一个函子 (functor)，我们可以在需要的任意高级别上将其精确化。
- $\mathcal{U}_P$ 中的类型在所有极限 (limits) 下封闭，例如乘积和拉回 (pullbacks)。特别地，对于任意 $A : \mathcal{U}_P$ 和 $x, y : A$，等式类型 $(x =_A y)$ 也在 $\mathcal{U}_P$ 中，因为它是两个函数 $\mathbf{1} \to A$ 的拉回。
- $\mathcal{U}_P$ 中的余极限 (colimits) 可以通过对类型的普通余极限应用 $\bigcirc$ 来构造。

重要的是，乘积的封闭性也扩展到了“无限乘积”，即依赖函数类型 (dependent function types)。

Theorem 7.7.2. *如果 $B : A \to \mathcal{U}_P$ 是反射子宇宙 $\mathcal{U}_P$ 中的任意类型族 (family of types)，那么 $\prod_{(x:A)} B(x)$ 也在 $\mathcal{U}_P$ 中。*

*证明.* 对于任意 $x : A$，考虑函数 $\mathsf{ev}_x : (\prod_{(x:A)} B(x)) \to B(x)$，定义为 $\mathsf{ev}_x(f) :\equiv f(x)$。由于 $B(x)$ 属于 $P$，这可以扩展为一个函数

$$\mathsf{rec}_{\bigcirc}(\mathsf{ev}_x) : \bigcirc\Big(\prod_{x:A} B(x)\Big) \to B(x).$$

因此，我们可以定义 $h : \bigcirc(\prod_{(x:A)} B(x)) \to \prod_{(x:A)} B(x)$，令 $h(z)(x) :\equiv \mathsf{rec}_{\bigcirc}(\mathsf{ev}_x)(z)$。然后 $h$ 是 $\eta_{\prod_{(x:A)} B(x)}$ 的一个收缩，因此 $\prod_{(x:A)} B(x)$ 属于 $\mathcal{U}_P$。 □

特别地，如果 $B : \mathcal{U}_P$ 并且 $A$ 是任意类型，那么 $(A \to B)$ 属于 $\mathcal{U}_P$ 。在范畴论的语言中，这意味着任何反射子宇宙都是一个**指数理想** (exponential ideal)。这反过来又意味着通过标准的论证，反射器保留有限乘积。

Corollary 7.7.3. *对于任意类型 $A$ 和 $B$ 以及任意反射子宇宙，诱导映射 $\bigcirc(A \times B) \to \bigcirc(A) \times \bigcirc(B)$ 是一个等价。*

*证明.* 只需证明 $\bigcirc(A) \times \bigcirc(B)$ 具有与 $\bigcirc(A \times B)$ 相同的普遍性质即可。根据上述关于 $\mathcal{U}_P$ 中类型在极限下封闭的备注，它属于 $\mathcal{U}_P$。现在令 $C : \mathcal{U}_P$；我们有

$$\begin{aligned}(\bigcirc(A) \times \bigcirc(B) \to C) &= (\bigcirc(A) \to (\bigcirc(B) \to C)) \\ &= (\bigcirc(A) \to (B \to C)) \\ &= (A \to (B \to C)) \\ &= (A \times B \to C)\end{aligned}$$

使用 $\bigcirc(B)$ 和 $\bigcirc(A)$ 的普遍性质，以及 $B \to C$ 属于 $\mathcal{U}_P$ 的事实。很容易验证这个等价是通过与 $\eta_A \times \eta_B$ 复合得到的，如所需。 □

可能看起来很奇怪，所有类型的反射子类 (reflective subcategories) 自动成为一个指数理想 (exponential ideal)，并且有一个保留乘积的反射器。然而，在经典的集合范畴 (category of sets) 中也是如此，原因相同。只是这个事实通常不被提及，因为经典的集合范畴——与同伦类型范畴 (category of homotopy types) 相比——并没有很多有趣的反射子类。
$n$-类型的两个基本性质在一般的反射子宇宙中并不共享：Theorem 7.1.8（在 $\Sigma$ 类型下的封闭性）和 Theorem 7.3.2（截断归纳，truncation induction）。然而，这两个性质的类比是等价的。

Theorem 7.7.4. *对于反射子宇宙 $\mathcal{U}_P$ ，以下条件在逻辑上是等价的：*

(i) *如果 $A : \mathcal{U}_P$ 且 $B : A \to \mathcal{U}_P$，那么 $\sum_{(x:A)} B(x)$ 也属于 $\mathcal{U}_P$ 。*

(ii) *对于每个 $A : \mathcal{U}$、类型族 $B : \bigcirc A \to \mathcal{U}_P$ 以及映射 $g : \prod_{(a:A)} B(\eta(a))$，存在 $f : \prod_{(z:\bigcirc A)} B(z)$，使得对于所有 $a : A$，都有 $f(\eta(a)) = g(a)$。*

证明. 假设条件 (i) 成立。然后，在条件 (ii) 的情况下，类型 $\sum_{(z:\bigcirc A)} B(z)$ 属于 $\mathcal{U}_P$ ，并且我们有 $g' : A \to \sum_{(z:\bigcirc A)} B(z)$，定义为 $g'(a) :\equiv (\eta(a), g(a))$。因此，我们有 $\mathsf{rec}_\bigcirc(g') : \bigcirc A \to \sum_{(z:\bigcirc A)} B(z)$，使得 $\mathsf{rec}_\bigcirc(g')(\eta(a)) = (\eta(a), g(a))$。
现在考虑函数 $\mathsf{pr}_1 \circ \mathsf{rec}_\bigcirc(g') : \bigcirc A \to \bigcirc A$ 和 $\mathsf{id}_{\bigcirc A}$。根据假设，当前置组合 $\eta$ 时，这两个函数相等。因此，根据 $\bigcirc$ 的普遍性质，它们已经相等，即我们有 $p_z : \mathsf{pr}_1(\mathsf{rec}_\bigcirc(g')(z)) = z$ 对于所有 $z$ 都成立。现在我们可以定义 $f(z) :\equiv p_{z*}(\mathsf{pr}_2(\mathsf{rec}_\bigcirc(g')(z)))$, 使用 Definition 7.7.1 中定义的等价性原理，证明 $\mathsf{rec}_\bigcirc(g')(\eta(a)) = (\eta(a), g(a))$ 的第一部分等于 $p_{\eta(a)}$。因此，其第二部分证明 $f(\eta(a)) = g(a)$，如所需。
反过来，假设条件 (ii) 成立，并且 $A : \mathcal{U}_P$ 且 $B : A \to \mathcal{U}_P$。令 $h$ 是复合函数

$$\bigcirc\Big(\sum_{x:A} B(x)\Big) \xrightarrow{\bigcirc(\mathsf{pr}_1)} \bigcirc A \xrightarrow{(\eta_A)^{-1}} A.$$

对于 $z : \sum_{(x:A)} B(x)$，我们有

$$\begin{aligned} h(\eta(z)) &= \eta^{-1}(\bigcirc(\mathsf{pr}_1)(\eta(z))) \\ &= \eta^{-1}(\eta(\mathsf{pr}_1(z))) \\ &= \mathsf{pr}_1(z). \end{aligned}$$

将此路径表示为 $p_z$。现在，如果我们定义 $C : \bigcirc(\sum_{(x:A)} B(x)) \to \mathcal{U}$ 为 $C(w) :\equiv B(h(w))$，我们有

$$g :\equiv \lambda z.\, p_{z*}(\mathsf{pr}_2(z)) \;:\; \prod_{z:\sum_{(x:A)} B(x)} C(\eta(z)).$$

因此，假设条件产生了 $f : \prod_{(w:\bigcirc(\sum_{(x:A)} B(x)))} C(w)$ 使得 $f(\eta(z)) = g(z)$。组合 $h$ 和 $f$ 给出了一个函数 $k : \bigcirc(\sum_{(x:A)} B(x)) \to \sum_{(x:A)} B(x)$ 定义为 $k(w) :\equiv (h(w), f(w))$，而 $p_z$ 和等式 $f(\eta(z)) = g(z)$ 显示 $k$ 是 $\eta_{\sum_{(x:A)} B(x)}$ 的一个收缩。因此，$\sum_{(x:A)} B(x)$ 属于 $\mathcal{U}_P$ 。 □

请注意与 §5.5 中讨论的相似性。反射子宇宙的反射器的普遍性质类似于归纳原则及其唯一性原理，而 Theorem 7.7.4(ii) 类似于相应的归纳原则。与 §5.5 不同的是，这里这两者并不等价，因为我们只能将其消解到 $\mathcal{U}_P$ 中的类型。Theorem 7.7.4 的条件 (i) 修复了这种脱节。
不出所料，如果我们有归纳原则，那么我们可以推导出递归原则。我们还可以推导出它的唯一性原理，只要我们允许自己消解到路径类型。这表明了以下定义。请注意，任何反射子宇宙都可以通过操作 $\bigcirc : \mathcal{U} \to \mathcal{U}$ 和函数 $\eta_A : A \to \bigcirc A$ 来表征，因为我们有 $P(A) = \mathsf{isequiv}(\eta_A)$。

Definition 7.7.5. **模态** (modality) 是一个操作 $\bigcirc : \mathcal{U} \to \mathcal{U}$，它有以下特性：

(i) 对于每种类型 $A$，都有函数 $\eta^\bigcirc_A : A \to \bigcirc(A)$。
(ii) 对于每种类型 $A : \mathcal{U}$ 和每个类型族 $B : \bigcirc(A) \to \mathcal{U}$，有一个函数

$$\mathsf{ind}_\bigcirc : \Big(\prod_{a:A} \bigcirc(B(\eta^\bigcirc_A(a)))\Big) \to \prod_{z:\bigcirc(A)} \bigcirc(B(z)).$$

(iii) 对于每个 $f : \prod_{(a:A)} \bigcirc(B(\eta^\bigcirc_A(a)))$，都有路径 $\mathsf{ind}_\bigcirc(f)(\eta^\bigcirc_A(a)) = f(a)$。
(iv) 对于任何 $z, z' : \bigcirc(A)$，函数 $\eta^\bigcirc_{z=z'} : (z = z') \to \bigcirc(z = z')$ 是一个等价。

我们说 $A$ 对于 $\bigcirc$ 是 **模态的** (modal) ，如果 $\eta^\bigcirc_A : A \to \bigcirc(A)$ 是一个等价，并且我们写作

$$\mathcal{U}_\bigcirc :\equiv \{\, X : \mathcal{U} \mid X \text{ 是 } \bigcirc\text{-模态的} \,\} \tag{7.7.6}$$

表示模态类型的集合。

条件 (ii) 和 (iii) 与 Theorem 7.7.4(ii) 非常相似，但使用 $\bigcirc B(z)$ 而不是假设 $B$ 在 $\mathcal{U}_P$ 中。这允许我们仅使用操作 $\bigcirc$ 来表述条件，而不需要事先给出谓词 $P : \mathcal{U} \to \mathsf{Prop}$。（这不是完全令人满意的，因为我们在条件 (iv) 中仍然不得不提及 $P$。我们不知道 (iv) 是否可以从 (i)–(iii) 推导出来。）然而，通过 Definition 7.7.5(ii) 和 (iii) 可以推出 Theorem 7.7.4(ii) 中的看似更强的性质，因为对于任何 $C : \bigcirc A \to \mathcal{U}_\bigcirc$，我们有 $C(z) \simeq \bigcirc C(z)$，并且我们可以跨越这个等价。
与其他归纳原则一样，这也意味着一个普遍性质。

Theorem 7.7.7. *令 $A$ 为一个类型，令 $B : \bigcirc(A) \to \mathcal{U}_\bigcirc$。则函数*

$$(- \circ \eta^\bigcirc_A) : \Big(\prod_{z:\bigcirc(A)} B(z)\Big) \to \Big(\prod_{a:A} B(\eta^\bigcirc_A(a))\Big)$$

*是一个等价。*

*证明.* 根据定义，操作 $\mathsf{ind}_\bigcirc$ 是 $(- \circ \eta^\bigcirc_A)$ 的右逆。因此，我们只需找到一个同伦

$$\prod_{z:\bigcirc(A)} s(z) = \mathsf{ind}_\bigcirc(s \circ \eta^\bigcirc_A)(z)$$

对于每个 $s : \prod_{(z:\bigcirc(A))} B(z)$，将其展示为左逆。根据假设，每个 $B(z)$ 都是模态的，因此每种类型 $s(z) = R^\bigcirc_X(s \circ \eta^\bigcirc_A)(z)$ 也是模态的。因此，只需找到类型

$$\prod_{a:A} s(\eta^\bigcirc_A(a)) = \mathsf{ind}_\bigcirc(s \circ \eta^\bigcirc_A)(\eta^\bigcirc_A(a))$$

的一个函数，这可以从 Definition 7.7.5(iii) 中得出。 □

特别地，对于每种类型 $A$ 和每个模态类型 $B$，我们有等价 $(\bigcirc A \to B) \simeq (A \to B)$。

Corollary 7.7.8. *对于任何模态 $\bigcirc$，$\bigcirc$-模态类型构成一个满足 Theorem 7.7.4 等价条件的反射子宇宙。*

因此，模态可以与在 $\Sigma$ 类型下封闭的反射子宇宙相识别。*模态*一词来源于 *模态逻辑* (modal logic)，研究可以形成诸如“可能 $A$”（通常写作 $\diamond A$）或“必然 $A$”（通常写作 $\Box A$）的陈述。符号 $\bigcirc$ 在任意模态算子 (modal operator) 中较为常见。在命题即类型 (propositions-as-types) 原则下，模态逻辑中的模态对应于 *类型*上的操作，并且 Definition 7.7.5 似乎是如何定义此类操作的一个合理候选。（更精确地说，我们或许应该称这些为 *幺半模态* (idempotent, monadic) 模态；参见注释部分。）如 §3.10 所述，我们通常可以使用副词来非正式地谈论这些模态，例如“仅仅”（merely）对于命题截断 (propositional truncation) 和“纯粹地”（purely）对于身份模态 (identity modality) （即由 $\bigcirc A :\equiv A$ 定义的那个）。
对于任何模态 $\bigcirc$，我们定义映射 $f : A \to B$ 为 **$\bigcirc$-连通的** ($\bigcirc$-connected)如果 $\bigcirc(\mathsf{fib}_f(b))$ 对于所有 $b : B$ 是收缩的 (contractible)，并且定义为 **$\bigcirc$-截断的** ($\bigcirc$-truncated)如果 $\mathsf{fib}_f(b)$ 对于所有 $b : B$ 是模态的。§§7.5 and 7.6 中的所有理论只要不涉及不同 $n$ 值的 $n$-类型之间的关系，都可以在这个一般情况下原封不动地适用。特别是，我们有一个正交分解系统 (orthogonal factorization system)。
一类重要的模态是不包括 $n$-截断的 (truncation) *左正合模态* (left exact modalities)：它们的函子 $\bigcirc$ 保留拉回 (pullbacks) 以及有限乘积 (finite products)。这些是在初等拓扑斯 (elementary topos) 理论中“Lawvere-Tierney 拓扑”的范畴化 (categorification)，在高阶范畴语义学 (higher-categorical semantics) 中对应于子-$(\infty,1)$-拓扑斯。然而，这超出了本书的讨论范围。
除了 $n$-截断外，还可以在练习中找到其他特定的模态例子。

## Notes

同伦理论 (classical homotopy theory) 中的同伦 $n$-类型的概念相当古老。是 Voevodsky 意识到这个概念可以在同伦类型论 (homotopy type theory) 中递归定义，从契约性 (contractibility) 开始。
“公理 K (Axiom K)”属性由 Thomas Streicher 命名，作为身份类型 (identity types) 的一种性质，它出现在 J 之后，J 是身份类型消解器 (eliminator) 的传统名称。Theorem 7.2.5 归因于 Hedberg [?]；[?] 包含更多信息和推广。

$n$-连通空间和函数的概念在同伦理论中也是经典的，尽管如前所述，我们对函数连通性的索引比经典索引多一个。由 Rezk、Lurie 和其他人在高阶拓扑斯理论中的最新工作强调了由此产生的分解系统的重要性。特别是，本章的结果应该与 [?, §6.5.1] 进行比较。在 ?? 中，$n$-连通映射的理论对于我们的 Freudenthal 悬挂定理 (Freudenthal suspension theorem) 的证明至关重要。
模态算子 在 简单类型论 (simple type theory) 中已经被广泛研究；参见例如 [?]。在依赖类型论 (dependent type theory) 的背景下，[?] 处理了作为模态算子的命题截断 ($(-1)$-truncation) 的特殊情况。这里提出的发展极大地扩展和推广了这项工作，同时还借鉴了拓扑斯理论 (topos theory) 的思想。

通常，模态算子 至少有两种类型：如 $\Diamond$（“可能”）的那些类型，其中 $A \Rightarrow \Diamond A$，以及如 $\Box$（“必然”）的那些类型，其中 $\Box A \Rightarrow A$。当它们也是 幺半的 (idempotent)（即 $\Diamond A = \Diamond\Diamond A$ 或 $\Box A = \Box\Box A$）时，前者可以与反射子类 (reflective subcategories)（或等价地，幺半单子 (idempotent monads)）相识别，后者可以与对反射子类 (coreflective subcategories)（或幺半伴随 (idempotent comonads)）相识别。然而，在依赖类型论中处理共单子类型 (comonadic sort) 更为棘手，因为它们更少稳定于拉回下，因此不能解释为宇宙 $\mathcal{U}$ 上的操作。有时可以绕过这个问题（例如，参见 [?]），但为了简洁，这里我们坚持单子类型。
在计算机科学方面，单子 (monads)（因此模态）用于在函数式编程 (functional programming) 中模拟计算效果 (computational effects) [?]。 当执行结果没有副作用（例如在屏幕上打印消息、播放音乐或通过互联网发送数据）时，计算被称为 纯粹的 (pure)。存在“纯函数式”编程语言 (purely functional programming languages)，例如 Haskell，其中从技术上讲只能编写纯函数：副作用通过对输出类型应用“单子”来表示。例如，类型为 $\mathsf{Int} \to \mathsf{Int}$ 的函数是纯的，而类型为 $\mathsf{Int} \to \mathsf{IO}(\mathsf{Int})$ 的函数在计算结果的过程中可能会执行输入和输出；操作 $\mathsf{IO}$ 是一个单子。（这是我们使用副词“纯粹地 (purely)”来表示身份单子的起源，因为它在计算上对应于没有副作用的纯函数。）我们在本章中考虑的模态都是幺半的，而函数式编程中使用的那些则很少如此，但这些思想仍然密切相关。

## Exercises

Exercise 7.1.

(i) 使用 Theorem 7.2.2 证明，如果对于每种类型 $A$，都有 $\|A\| \to A$，那么每种类型都是集合。
(ii) 证明如果每个满射函数 (surjective function)（纯粹地）分裂，即 $\prod_{(b:B)} \left\|\mathsf{fib}_f(b)\right\| \to \prod_{(b:B)} \mathsf{fib}_f(b)$ 对于每个 $f : A \to B$，那么每种类型都是集合。

Exercise 7.2. 对于这个练习，我们考虑以下一般的余极限概念 (colimit)。定义一个 **图** (graph) $\Gamma$，其由类型 $\Gamma_0$ 和一个家族 $\Gamma_1 : \Gamma_0 \to \Gamma_0 \to \mathcal{U}$ 组成。一个 **图上的图表** (diagram)（类型的）由图 $\Gamma$ 组成，连同对于每个 $x, y : \Gamma_0$ 的函数 $F_{x,y} : \Gamma_1(x,y) \to F(x) \to F(y)$。此类图表的 **余极限** 是由以下类型生成的高阶归纳类型 (higher inductive type) $\mathsf{colim}(F)$：

- 对于每个 $x : \Gamma_0$，一个函数 $\mathsf{inc}_x : F(x) \to \mathsf{colim}(F)$，以及
- 对于每个 $x, y : \Gamma_0$ 和 $\gamma : \Gamma_1(x,y)$ 以及 $a : F(x)$，一个路径 $\mathsf{inc}_y(F_{x,y}(\gamma, a)) = \mathsf{inc}_x(a)$。

当然，还有更多类型的余极限（例如，参见 Exercise 7.16），但这已经足够满足许多目的。

(i) 展示一个图 $\Gamma$，使得 $\Gamma$-图表的余极限可以与 §6.8 中定义的推挤 (pushouts) 相识别。换句话说，每个跨度都应该引发一个 $\Gamma$ 上的图表，其余极限是该跨度的推挤。
(ii) 展示一个图 $\Gamma$ 和图表 $F$，其中 $\Gamma$ 上的图表 $F(x) = \mathbf{1}$ 对于所有 $x$ 都成立，但 $\mathsf{colim}(F) = \mathbb{S}^2$。请注意，$\mathbf{1}$ 是 $(-2)$-类型，而 $\mathbb{S}^2$ 不应被视为任何有限 $n$ 的 $n$-类型。另见 Exercise 7.16。

Exercise 7.3. 证明如果 $A$ 是一个 $n$-类型，并且 $B : A \to n\text{-}\mathsf{Type}$ 是一个 $n$-类型的家族，其中 $n \geq -1$，那么 $W$ 类型 $\mathsf{W}_{(a:A)} B(a)$（参见 §5.3）也是一个 $n$-类型。

Exercise 7.4. 使用 Lemma 7.5.13 将 Lemma 7.5.11 扩展到任何截面-收缩对 (section-retraction pair)。

Exercise 7.5. 证明 Corollary 7.5.9 也可以作为反方向的表征：如果 $B$ 是 $n$-类型，则每个从 $n$-连通类型映射到 $B$ 的映射都是常数映射。理想情况下，您的证明应适用于 §7.7 中的任何模态。

Exercise 7.6. 证明对于 $n \geq -1$，一个类型 $A$ 是 $n$-连通的，如果且仅如果它只是被居住，并且对于所有 $a, b : A$，类型 $a =_A b$ 是 $(n-1)$-连通的。因此，由于每种类型都是 $(-2)$-连通的，$n$-连通性类型可以仅使用命题截断递归定义。（特别地，$A$ 是 0-连通的，如果且仅如果 $\|A\|$ 和 $\prod_{(a,b:A)} \|a = b\|$。）

Exercise 7.7. 对于 $-1 \leq n, m \leq \infty$，令 $\mathsf{LEM}_{n,m}$ 表示陈述

$$\prod_{A:n\text{-}\mathsf{Type}} \|A + \neg A\|_m,$$

其中 $\infty\text{-}\mathsf{Type} :\equiv \mathcal{U}$ 并且 $\|X\|_\infty :\equiv X$。证明：

(i) 如果 $n = -1$ 或 $m = -1$，那么 $\mathsf{LEM}_{n,m}$ 等价于 §3.4 中的 $\mathsf{LEM}$。
(ii) 如果 $n \geq 0$ 并且 $m \geq 0$，那么 $\mathsf{LEM}_{n,m}$ 与一值性 (univalence) 不一致。

Exercise 7.8. 对于 $-1 \leq n, m \leq \infty$，令 $\mathsf{AC}_{n,m}$ 表示陈述

$$\prod_{(X:\mathsf{Set})} \prod_{(Y:X\to n\text{-}\mathsf{Type})} \left(\prod_{x:X} \|Y(x)\|_m\right) \to \left\|\prod_{x:X} Y(x)\right\|_m,$$

与 Exercise 7.7 中的惯例相同。因此，$\mathsf{AC}_{0,-1}$ 是 §3.8 中的选择公理 (axiom of choice)，而 $\mathsf{AC}_{\infty,\infty}$ 是恒等函数。（如果我们像 (3.8.1) 而不是 (3.8.3) 那样制定 $\mathsf{AC}_{n,m}$，那么 $\mathsf{AC}_{\infty,\infty}$ 会像 Theorem 2.9.7 一样。）众所周知，$\mathsf{AC}_{\infty,-1}$ 与一值性是一致的，因为它在 Voevodsky 的单纯模型 (simplicial model) 中成立。

(i) 不使用一值性，证明 $\mathsf{LEM}_{n,\infty}$ 意味着 $\mathsf{AC}_{n,m}$ 对所有 $m$ 成立。（另一方面，在 §10.1.5 中，我们将证明 $\mathsf{AC} = \mathsf{AC}_{0,-1}$ 意味着 $\mathsf{LEM} = \mathsf{LEM}_{-1,-1}$。）
(ii) 当然，如果 $k \leq n$，则 $\mathsf{AC}_{n,m} \Rightarrow \mathsf{AC}_{k,m}$。$\mathsf{AC}_{n,m}$ 之间还有其他含义吗？$\mathsf{AC}_{n,m}$ 与一值性对任何 $m \geq 0$ 和任何 $n$ 一致吗？（这些是开放问题。）

Exercise 7.9. 证明 $\mathsf{AC}_{n,-1}$ 意味着对于任何 $n$-类型 $A$，仅存在一个集合 $B$ 并且一个满射 $B \to A$。

Exercise 7.10. 定义 **$n$-连通选择公理** ($n$-connected axiom of choice) 为以下陈述：

> 如果 $X$ 是一个集合并且 $Y : X \to \mathcal{U}$ 是一个类型家族，使得每个 $Y(x)$ 是 $n$-连通的，那么 $\prod_{(x:X)} Y(x)$ 是 $n$-连通的。

请注意，$(-1)$-连通选择公理是 Exercise 7.8 中的 $\mathsf{AC}_{\infty,-1}$。

(i) 证明 $(-1)$-连通选择公理意味着所有 $n \geq -1$ 的 $n$-连通选择公理。
(ii) $n$-连通选择公理与 $\mathsf{AC}_{n,m}$ 原则之间还有其他含义吗？（这是一个开放问题。）

Exercise 7.11. 证明 $n$-截断模态对于任何 $n \geq -1$ 都不是左正合的。也就是说，展示一个它不能保留的拉回。

Exercise 7.12. 证明 $X \mapsto (\neg\neg X)$ 是一个模态。

Exercise 7.13. 令 $P$ 为一个纯命题 (mere proposition)。

(i) 证明 $X \mapsto (P \to X)$ 是一个左正合模态。这称为 **开模态** (open modality)
(ii) 证明 $X \mapsto P * X$ 是一个左正合模态，其中 $*$ 表示连接 (join)（参见 §6.8）。这称为 **闭模态** (closed modality)与 $P$ 关联。

Exercise 7.14. 令 $f : A \to B$ 为一个映射；类型 $Z$ 是 **$f$-局部的** ($f$-local) 如果 $(- \circ f) : (B \to Z) \to (A \to Z)$ 是一个等价。

(i) 证明 $f$-局部类型构成一个反射子宇宙。您将需要使用高阶归纳类型定义反射器 (localization)。
(ii) 证明如果 $B = \mathbf{1}$，那么该子宇宙是一个模态。

Exercise 7.15. 与 Remark 6.7.1 相反，展示我们可以等效地定义 $\|A\|_n$ 为由函数 $|-|_n : A \to \|A\|_n$ 生成的类型，连同每个 $r : \mathbb{S}^{n+1} \to \|A\|_n$ 和每个 $x : \mathbb{S}^{n+1}$ 的路径 $s_r(x) : r(x) = r(\mathsf{base})$。

Exercise 7.16. 在这个练习中，我们考虑一个比 Exercise 7.2 中略为复杂的余极限概念。定义 **具有复合的图** (graph with composition) $\Gamma$ 为一个图，如 Exercise 7.2 中所述，连同每个 $x,y,z:\Gamma_0$ 的函数 $\Gamma_1(y,z)\to\Gamma_1(x,y)\to\Gamma_1(x,z)$，表示为 $\delta\mapsto\gamma\mapsto\delta\circ\gamma$。（例如，Chapter 9 中的任何预范畴 (precategory) 都是一个具有复合的图。）一个 **图上的图表** (diagram) $F$ 由基础图上的图表组成，连同每个 $x,y,z:\Gamma_0$ 和 $\gamma:\Gamma_1(x,y)$ 以及 $\delta:\Gamma_1(y,z)$ 的同伦 $\mathsf{cmp}_{x,y,z}(\delta,\gamma):F_{y,z}(\delta)\circ F_{x,y}(\gamma)\sim F_{x,z}(\delta\circ\gamma)$。此类图表的 **余极限** 是由以下类型生成的高阶归纳类型 $\mathsf{colim}(F)$：

- 对于每个 $x:\Gamma_0$，一个函数 $\mathsf{inc}_x:F(x)\to\mathsf{colim}(F)$，
- 对于每个 $x,y:\Gamma_0$ 和 $\gamma:\Gamma_1(x,y)$ 以及 $a:F(x)$，一个路径 $\mathsf{glue}_{x,y}(\gamma,a):\mathsf{inc}_y(F_{x,y}(\gamma,a))=\mathsf{inc}_x(a)$，以及
- 对于每个 $x,y,z:\Gamma_0$ 和 $\gamma:\Gamma_1(x,y)$ 以及 $\delta:\Gamma_1(y,z)$ 以及 $a:F(x)$，一个路径

$$\mathsf{inc}_z\big(\mathsf{cmp}_{x,y,z}(\delta,\gamma,a)\big)\cdot\mathsf{glue}_{x,z}(\delta\circ\gamma,a)=\mathsf{glue}_{y,z}(\delta,F_{x,y}(\gamma,a))\cdot\mathsf{glue}_{x,y}(\gamma,a)$$

（这是“第二阶近似”，用于完全同伦理论 (homotopy-theoretic) 的图表和余极限概念，应该涉及所有更高层次的“连贯路径”(coherence paths)。在类型论中定义这种东西是一个重要的开放问题。）
展示一个具有复合的图 $\Gamma$，其中 $\Gamma_0$ 是一个集合并且每个类型 $\Gamma_1(x,y)$ 是纯命题，并且展示一个图表 $F$ 使得 $\Gamma$ 上的图表 $F(x)=\mathbf{1}$ 对于所有 $x$ 成立，但 $\mathsf{colim}(F)=\mathbb{S}^2$。

Exercise 7.17. 比较 Lemmas 7.5.12 and 7.5.13，您可能会推测，如果 $f:A\to B$ 是 $n$-连通的，并且 $g:\prod_{(a:A)}P(a)\to Q(f(a))$ 诱导了一个 $n$-连通的映射 $\left(\sum_{(a:A)}P(a)\right)\to\left(\sum_{(b:B)}Q(b)\right)$，那么 $g$ 是纤维上的 $n$-连通的。举出一个反例证明这不成立。（实际上，当推广到模态时，该属性表征了左正合模态；参见 Exercise 7.13。）

Exercise 7.18. 证明如果 $f:A\to B$ 是 $n$-连通的，那么 $\|f\|_k:\|A\|_k\to\|B\|_k$ 也是 $n$-连通的。

Exercise 7.19. 我们说一个类型 $A$ 是 **范畴连通的** (categorically connected) 如果对于每种类型 $B,C$，通过函数定义的典型映射 $e_{A,B,C}:((A\to B)+(A\to C))\to(A\to B+C)$

$$e_{A,B,C}(\mathsf{inl}(g)):\equiv\lambda x.\,\mathsf{inl}(g(x)),$$
$$e_{A,B,C}(\mathsf{inr}(g)):\equiv\lambda x.\,\mathsf{inr}(g(x))$$

是一个等价。

(i) 证明任何连通类型都是范畴连通的。
(ii) 证明如果且仅如果 LEM 成立，则所有范畴连通类型都是连通的。（提示：考虑 $A:\equiv\Sigma P$ 使得 $\neg\neg P$ 成立。）

# Part II

# Mathematics

Chapter 8

# 同伦理论 (Homotopy Theory)

在本章中，我们将在类型理论中发展一些同伦理论 (homotopy theory)。我们使用了在 Chapter 2中介绍的**综合方法** (synthetic approach) 来进行同伦理论的研究：空间 (spaces)、点 (points)、路径 (paths) 和同伦 (homotopies) 是基本概念，它们分别由类型 (types) 和类型的元素表示，特别是同一性类型 (identity type)。路径和同伦的代数结构由类型上自然的 $\infty$-群 (groupoid) 结构生成，这一结构是由同一性类型的规则生成的。使用在 Chapter 6中引入的高阶归纳类型 (higher inductive types)，我们可以直接通过其泛性质来描述空间。

这种综合方法有几个有趣的方面。首先，它结合了具体模型（如拓扑空间 (topological spaces) 或单纯集合 (simplicial sets)）的优势与用于同伦理论的抽象范畴框架（如 Quillen 模型范畴 (Quillen model categories)）的优势。一方面，我们的证明看起来很基础，并且具体涉及类型中的点、路径和同伦。另一方面，我们的方法仍然抽象化了这些对象的任何具体表示——例如，路径连接的结合性是通过路径归纳法 (path induction) 证明的，而不是通过映射 $[0,1] \to X$ 的重参数化或通过角填充条件 (horn-filling conditions) 证明的。类型理论似乎是研究 $\infty$-群 (groupoid) 抽象同伦理论的非常方便的方法：通过使用同一性类型的规则，我们可以避免许多 $\infty$-群定义中涉及的复杂组合问题，并且只需在任何特定证明中说明所需的结构。

类型理论的抽象性质意味着我们的证明可以自动应用于各种环境中。特别是，如前所述，同伦类型理论 (homotopy type theory) 在 Kan 单纯集合 (Kan simplicial sets) 中有一个解释，这是一种 $\infty$-群同伦理论的模型。因此，我们的证明适用于这个模型，并且通过从单纯集合到拓扑空间的几何实现函子 (geometric realization functor) 转移它们，可以得到经典同伦理论中的对应定理。然而，尽管细节仍在进行中，我们也可以在看起来像 $\infty$-群的其他范畴（例如 $(\infty,1)$-拓扑 (toposes)）中解释类型理论。因此，在类型理论中证明的结果也将在这些环境中成立。这种额外的一般性在普通范畴逻辑中是众所周知的：同一性基础将其扩展到同伦理论中。

其次，我们的综合方法提出了一些新的类型理论方法和证明。我们的一些证明是经典证明的直接转录。其他证明则更具有类型理论的风格，主要包括使用 $\infty$-群操作的计算，这种风格与计算机科学家使用类型理论推理计算机程序非常相似。类型理论允许我们聚焦于同伦理论中不同于其他方法的方面：虽然工具如路径归纳法和高阶归纳类型的泛性质在像 Kan 单纯集合这样的环境中是可用的，但类型理论提升了它们的重要性，因为它们是处理这些类型的**唯一**原始工具。聚焦于这些工具已经导致了对熟悉构造的新描述，例如圆的通用覆盖 (universal cover) 和 Hopf 纤维 (Hopf fibration)，这些描述仅使用高阶归纳类型的递归原则。这些描述非常直接，本章中的许多证明涉及到与这些纤维的计算计算。我们的证明的另一个新方面是它们是构造性的（假设同一性和高阶归纳类型是构造性的）；我们将在??中描述将其应用于球体的同伦群的一个应用。

第三，我们的综合方法非常适合于在证明助手 (proof assistants)（如 Coq 和 Agda ）中进行计算机验证的证明。本章中描述的几乎所有证明都经过了计算机验证，其中许多证明首先在证明助手中给出，然后为本书“非形式化”。计算机验证的证明与这里提出的非正式证明在长度和努力上是可比的，并且在某些情况下，它们甚至更短，更容易完成。

在开始介绍我们的结果之前，我们将简要回顾一些基本概念和同伦理论中的定理，以便那些不熟悉这

些概念的读者。我们还将概述本章中证明的结果。

同伦理论是代数拓扑的一个分支，并使用抽象代数的工具（如群论 (group theory)）来研究空间的性质。同伦理论家研究的一个问题是如何判断两个空间是否相同，其中"相同"意味着**同伦等价** (homotopy equivalence)（连续映射来回组合成同伦意义上的恒等映射——这给了"修正"那些不能完全组合成恒等映射的映射的机会）。判断两个空间是否相同的一种常见方法是计算与空间相关的**代数不变量** (algebraic invariants)，其中包括其**同伦群** (homotopy groups) 和**同调群** (homology) 和**上同调群** (cohomology groups)。同伦等价的空间具有同构的同伦/（上）同调群，因此如果两个空间具有不同的群，那么它们就不等价。因此，这些代数不变量提供了有关空间的全局信息，可用于区分空间，并补充了诸如连续性等概念所提供的局部信息。例如，圆环 (torus) 在局部看起来像 2-球面 (2-sphere)，但它有一个全局差异，因为它有一个洞，这一差异在这两个空间的同伦群中是可见的。

最简单的同伦群的例子是空间的**基本群** (fundamental group)，记为 $\pi_1(X, x_0)$：给定一个空间 $X$ 及其中的一个点 $x_0$，可以构造一个群，其元素是 $x_0$ 处的环路（从 $x_0$ 到 $x_0$ 的连续路径），在同伦意义上进行考虑，群运算由恒等路径（静止）、路径连接和路径反转给出。例如，2-球面的基本群是平凡的，但圆环的基本群不是，这表明球面和圆环不是同伦等价的。直观上，每个球面上的环路都是同伦等价于恒等的，因为它的内部可以填充。而在圆环上穿过圆环洞的环路不是同伦等价于恒等的，因此基本群中有非平凡元素。

**高阶同伦群** (higher homotopy groups) 提供了空间的更多信息。固定空间 $X$ 中的一个点 $x_0$，并考虑恒等路径 $\mathsf{refl}_{x_0}$。那么，$\mathsf{refl}_{x_0}$ 与自身之间同伦的同伦类构成一个群 $\pi_2(X, x_0)$，它告诉我们关于空间二维结构的一些信息。然后，$\pi_3(X, x_0)$ 是同伦之间的同伦类的群，依此类推。代数拓扑的一个基本问题是**计算空间 $X$ 的同伦群**，这意味着给出 $\pi_k(X, x_0)$ 和一些更直接的群的描述（例如，通过乘法表或呈示）之间的群同构。令人惊讶的是，即使对于像球体 (spheres) 这样简单的空间，这个问题也是非常困难的。如 Table 8.1 所示，高阶同伦群的一些模式出现了，但没有一般公式，并且许多球体的同伦群目前仍然未知。

| | $\mathbb{S}^0$ | $\mathbb{S}^1$ | $\mathbb{S}^2$ | $\mathbb{S}^3$ | $\mathbb{S}^4$ | $\mathbb{S}^5$ | $\mathbb{S}^6$ | $\mathbb{S}^7$ | $\mathbb{S}^8$ |
|---|---|---|---|---|---|---|---|---|---|
| $\pi_1$ | 0 | $\mathbb{Z}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\pi_2$ | 0 | 0 | $\mathbb{Z}$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $\pi_3$ | 0 | 0 | $\mathbb{Z}$ | $\mathbb{Z}$ | 0 | 0 | 0 | 0 | 0 |
| $\pi_4$ | 0 | 0 | $\mathbb{Z}_2$ | $\mathbb{Z}_2$ | $\mathbb{Z}$ | 0 | 0 | 0 | 0 |
| $\pi_5$ | 0 | 0 | $\mathbb{Z}_2$ | $\mathbb{Z}_2$ | $\mathbb{Z}_2$ | $\mathbb{Z}$ | 0 | 0 | 0 |
| $\pi_6$ | 0 | 0 | $\mathbb{Z}_{12}$ | $\mathbb{Z}_{12}$ | $\mathbb{Z}_2$ | $\mathbb{Z}_2$ | $\mathbb{Z}$ | 0 | 0 |
| $\pi_7$ | 0 | 0 | $\mathbb{Z}_2$ | $\mathbb{Z}_2$ | $\mathbb{Z}\times\mathbb{Z}_{12}$ | $\mathbb{Z}_2$ | $\mathbb{Z}_2$ | $\mathbb{Z}$ | 0 |
| $\pi_8$ | 0 | 0 | $\mathbb{Z}_2$ | $\mathbb{Z}_2$ | $\mathbb{Z}_2^2$ | $\mathbb{Z}_{24}$ | $\mathbb{Z}_2$ | $\mathbb{Z}_2$ | $\mathbb{Z}$ |
| $\pi_9$ | 0 | 0 | $\mathbb{Z}_3$ | $\mathbb{Z}_3$ | $\mathbb{Z}_2^2$ | $\mathbb{Z}_2$ | $\mathbb{Z}_{24}$ | $\mathbb{Z}_2$ | $\mathbb{Z}_2$ |
| $\pi_{10}$ | 0 | 0 | $\mathbb{Z}_{15}$ | $\mathbb{Z}_{15}$ | $\mathbb{Z}_{24}\times\mathbb{Z}_3$ | $\mathbb{Z}_2$ | 0 | $\mathbb{Z}_{24}$ | $\mathbb{Z}_2$ |
| $\pi_{11}$ | 0 | 0 | $\mathbb{Z}_2$ | $\mathbb{Z}_2$ | $\mathbb{Z}_{15}$ | $\mathbb{Z}_2$ | $\mathbb{Z}$ | 0 | $\mathbb{Z}_{24}$ |
| $\pi_{12}$ | 0 | 0 | $\mathbb{Z}_2^2$ | $\mathbb{Z}_2^2$ | $\mathbb{Z}_2$ | $\mathbb{Z}_{30}$ | $\mathbb{Z}_2$ | 0 | 0 |
| $\pi_{13}$ | 0 | 0 | $\mathbb{Z}_{12}\times\mathbb{Z}_2$ | $\mathbb{Z}_{12}\times\mathbb{Z}_2$ | $\mathbb{Z}_2^3$ | $\mathbb{Z}_2$ | $\mathbb{Z}_{60}$ | $\mathbb{Z}_2$ | 0 |

表 8.1: 球体的同伦群 [?]. $n$ 维球面 $\mathbb{S}^n$ 的第 $k$ 同伦群 $\pi_k$ 同构于表中的群，其中 $\mathbb{Z}$ 是整数的加法群，$\mathbb{Z}_m$ 是阶数为 $m$ 的循环群。

理解这种复杂性的一种方法是通过在 Chapter 2中介绍的空间和 $\infty$-群的对应关系。如在 §6.4中讨论的那样，2-球面由具有一个点和一个二维环的高阶归纳类型 (higher inductive type) 表示。因此，人们可能想知道为什么 $\pi_3(\mathbb{S}^2)$ 是 $\mathbb{Z}$，当类型 $\mathbb{S}^2$ 没有生成 3 维胞元的生成元时。事实证明，$\pi_3(\mathbb{S}^2)$ 的

生成元素是使用 Theorem 2.1.6 中描述的互换律构造的：∞-群的代数结构包括各个层级之间的非平凡相互作用，这些相互作用产生了更高阶同伦群的元素。

类型理论为研究这一结构提供了一个自然的环境，因为我们可以很容易地定义高阶同伦群。回顾 Definition 2.1.8中的定义，对于 $n : \mathbb{N}$，以点类型 $(A, a)$ 的 $n$ 重迭代环空间 递归定义为：

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

这给出了 $n$ 维环的空间（即类型），它本身具有更高阶的同伦。通过截断我们得到 $n$ 维环的集合（这也是在 §6.11中作为示例定义的）：

Definition 8.0.1 (同伦群 (Homotopy Groups)). 对于 $n \geq 1$ 和 $(A, a)$ 一个点类型，我们定义 $A$ 在 $a$ 处的**同伦群** (homotopy groups) 为

$$\pi_n(A, a) :\equiv \left\| \Omega^n(A, a) \right\|_0$$

由于 $n \geq 1$，路径连接和反转操作在 $\Omega^n(A)$ 上诱导出 $\pi_n(A)$ 上的操作，使其成为群的方式非常简单。如果 $n \geq 2$，那么通过 Eckmann–Hilton 论证 (Theorem 2.1.6)，群 $\pi_n(A)$ 是阿贝尔群。将 $\pi_0(A) :\equiv \|A\|_0$ 也是方便的，但这种情况有所不同：它不仅不是一个群，并且定义时不涉及 $A$ 中的任何基点。

这个定义适合于研究同伦群，因为一个类型 $X$ 的（高阶）归纳定义将 $X$ 表现为一个自由类型，类似于一个自由 ∞-群 (groupoid)，并且这种表示决定了 $X$ 的更高同一性类型，但并不显式地描述。同一性类型由生成元（如圆的 loop）和对它们应用的所有群操作（恒等，复合，逆元，结合律，互换律，……）生成。因此，高阶归纳的空间表示允许我们提出问题“$X$ 的同一性类型最终是什么？”尽管回答它可能需要一些重要的数学。这是类型理论中的一个更高维度的推广：即使 $X$ 是一个普通的归纳类型，如自然数或布尔值，刻画 $X$ 的同一性类型也可能需要一些工作。例如，$\mathbf{0}_2$ 和 $\mathbf{1}_2$ 是不同的这个定理并不是从定义直接得出的；见 §2.8。

同一性公理 (univalence axiom) 在计算同伦群时起着关键作用（没有同一性，类型理论可以兼容一种解释，其中所有路径，包括圆上的 loop，都是反射性的）。我们将在下面的圆的基本群的计算中看到这一点：从 $\Omega(\mathbb{S}^1)$ 到 $\mathbb{Z}$ 的映射是通过将圆上的环映射到集合 $\mathbb{Z}$ 的自同构 来定义的，例如 $\mathsf{loop} \cdot \mathsf{loop}^{-1}$ 被送到 $\mathsf{successor} \cdot \mathsf{predecessor}$（其中 successor 和 predecessor 是 $\mathbb{Z}$ 的自同构，被视为宇宙中的路径），然后将自同构应用于 0。通过同一性，宇宙中产生了非平凡的路径，这被用来从高阶归纳类型中的路径中提取信息。

在本章中，我们首先计算一些球体的同伦群，包括 $\pi_k(\mathbb{S}^1)$ (§8.1)，$k < n$ 时的 $\pi_k(\mathbb{S}^n)$ (§8.2 and ??)，通过 Hopf 纤维 (??) 和长正合序列 (long-exact-sequence) 的论证 (??) 计算 $\pi_2(\mathbb{S}^2)$ 和 $\pi_3(\mathbb{S}^2)$，以及通过 Freudenthal 悬挂定理 (??) 计算 $\pi_n(\mathbb{S}^n)$。接着，我们讨论 van Kampen 定理 (??)，它刻画了一个推挽 (pushout) 的基本群，以及 Whitehead 原理的状态（当一个映射在所有同伦群上引起等价时，它是否是一个等价?）(??)。最后，我们简要总结了书中未包括的其他结果，例如 $n \geq 3$ 时的 $\pi_{n+1}(\mathbb{S}^n)$，Blakers–Massey 定理，以及 Eilenberg–Mac Lane 空间的构造 (??)。本章的先决条件包括 Chapters 1, 2, 6 and 7 以及 Chapter 3的部分内容。

## 8.1 $\pi_1(S^1)$

在本节中，我们的目标是证明 $\pi_1(\mathbb{S}^1) = \mathbb{Z}$。实际上，我们将证明环空间（loop space）$\Omega(\mathbb{S}^1)$ 等价于 $\mathbb{Z}$。这是一个更强的陈述，因为根据定义，$\pi_1(\mathbb{S}^1) = \left\|\Omega(\mathbb{S}^1)\right\|_0$；因此，如果 $\Omega(\mathbb{S}^1) = \mathbb{Z}$，那么通过一致性，$\left\|\Omega(\mathbb{S}^1)\right\|_0 = \|\mathbb{Z}\|_0$，并且 $\mathbb{Z}$ 根据定义是一个集合（作为一个集合商；参见 Remarks 6.10.7 and 6.10.11），所以 $\|\mathbb{Z}\|_0 = \mathbb{Z}$。此外，知道 $\Omega(\mathbb{S}^1)$ 是一个集合将意味着 $\pi_n(\mathbb{S}^1)$ 对于 $n > 1$ 是平凡的，因此我们实际上计算了 所有的 $\mathbb{S}^1$ 的同伦群。

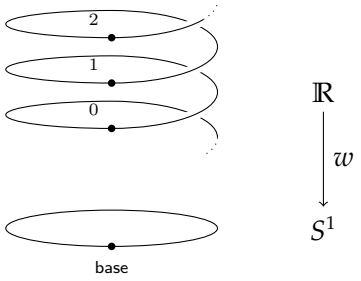



图 8.1: 经典拓扑中的绕线映射 (The winding map in classical topology)

### 8.1.1 开始 (Getting started)

定义 $\Omega(\mathbb{S}^1)$ 和 $\mathbb{Z}$ 之间双向的函数并不困难。通过将 Corollary 6.10.13 专门化为 $\mathsf{loop} : \mathsf{base} = \mathsf{base}$，我们得到了一个函数 $\mathsf{loop}^- : \mathbb{Z} \to (\mathsf{base} = \mathsf{base})$，它（粗略地说）定义如下：

$$\mathsf{loop}^n = \begin{cases} \underbrace{\mathsf{loop} \cdot \mathsf{loop} \cdot \cdots \cdot \mathsf{loop}}_{n} & \text{如果 } n > 0, \\ \underbrace{\mathsf{loop}^{-1} \cdot \mathsf{loop}^{-1} \cdot \cdots \cdot \mathsf{loop}^{-1}}_{-n} & \text{如果 } n < 0, \\ \mathsf{refl}_{\mathsf{base}} & \text{如果 } n = 0. \end{cases}$$

定义一个函数 $g : \Omega(\mathbb{S}^1) \to \mathbb{Z}$ 在另一个方向上有点棘手。注意，后继函数 $\mathsf{succ} : \mathbb{Z} \to \mathbb{Z}$ 是一个等价，因此在宇宙 $\mathcal{U}$ 中诱导出一个路径 $\mathsf{ua}(\mathsf{succ}) : \mathbb{Z} = \mathbb{Z}$。因此，$\mathbb{S}^1$ 的递归原理诱导出一个映射 $c : \mathbb{S}^1 \to \mathcal{U}$，其中 $c(\mathsf{base}) :\equiv \mathbb{Z}$，而 $\mathsf{ap}_c(\mathsf{loop}) := \mathsf{ua}(\mathsf{succ})$。然后我们有 $\mathsf{ap}_c : (\mathsf{base} = \mathsf{base}) \to (\mathbb{Z} = \mathbb{Z})$，并且我们可以定义 $g(p) :\equiv \mathsf{transport}^{X \mapsto X}(\mathsf{ap}_c(p), 0)$。

通过这些定义，我们甚至可以证明对于任何 $n : \mathbb{Z}$，$g(\mathsf{loop}^n) = n$，使用 Lemma 6.10.12 的归纳原理。（我们稍后将证明更一般的情况。）然而，另一个等式 $\mathsf{loop}^{g(p)} = p$ 则要困难得多。显而易见的方法是路径归纳，但路径归纳不适用于像 $p : (\mathsf{base} = \mathsf{base})$ 这样具有 *两端固定*的环路！需要一个新的想法，这可以用经典的同伦理论和类型理论来解释。我们从前者开始。

### 8.1.2 经典证明 (The classical proof)

在经典同伦理论中，有一个标准证明 $\pi_1(\mathbb{S}^1) = \mathbb{Z}$ 使用了通用覆盖空间。我们的证明可以看作是这个证明的类型理论版本，其中覆盖空间在这里出现为纤维，其纤维是集合。回顾一下，在同伦理论中，某个空间 $B$ 上的纤维对应于类型理论中的类型族 $B \to \mathcal{U}$。特别地，对于一个点 $x_0 : B$，类型族 $(x \mapsto (x_0 = x))$ 对应于 *路径纤维* $P_{x_0}B \to B$，其中 $P_{x_0}B$ 的点是从 $x_0$ 开始的 $B$ 中的路径，映射到 $B$ 选择了这种路径的另一端点。这个总空间 $P_{x_0}B$ 是可收缩的，因为我们可以“收缩”任何路径到其初始端点 $x_0$ ——我们已经看到了类型理论版本的这一点，如 Lemma 3.11.8。此外，$x_0$ 之上的纤维是环空间（loop space）$\Omega(B, x_0)$ ——在类型理论中这是显而易见的，因为这是环空间的定义。

现在在经典同伦理论中，$\mathbb{S}^1$ 被视为一个拓扑空间，我们可以如下进行。考虑“绕线”映射 $w : \mathbb{R} \to \mathbb{S}^1$，它看起来像是一个螺旋线投影到圆上（见 Figure 8.1）。这个映射 $w$ 将螺旋线上每一点映射到它“坐在上面”的圆上的点。它是一个纤维，并且每个点的纤维与整数（integers）同构。如果我们提升底部逆时针绕过的路径，我们在螺旋线上升一个级别，增加纤维中的整数。类似地，顺时针绕过底部的路径对应于在螺旋线上下降一级，减少这个计数。这个纤维称为圆的*通用覆盖*（universal cover）。

现在，经典同伦理论中的一个基本事实是 $E_1 \to E_2$ 是 $B$ 上纤维之间的映射，它是 $E_1$ 和 $E_2$ 之间的同伦等价，因而在所有纤维上诱导同伦等价。（我们已经在 Theorem 4.7.7 中看到了类型理论版本的这个事实。）因为 $\mathbb{R}$ 和 $P_{\mathsf{base}}S^1$ 都是可收缩的拓扑空间，所以它们是同伦等价的，因此它们在基点之上的纤维 $\mathbb{Z}$ 和 $\Omega(\mathbb{S}^1)$ 也是同伦等价的。

### 8.1.3 类型理论中的通用覆盖 (The universal cover in type theory)

让我们考虑如何在类型理论中表达前面的证明。我们已经提到 $\mathbb{S}^1$ 的路径纤维由类型族 $(x \mapsto (\mathsf{base} = x))$ 表示。我们也已经看到了一个好的候选通用覆盖 $\mathbb{S}^1$：它正是我们在 §8.1.1 中定义的类型族 $c : \mathbb{S}^1 \to \mathcal{U}$！根据定义，这个家族在 base 上的纤维是 $\mathbb{Z}$，而绕过 loop 的运输效应是增加一个——因此它的行为正如我们从 Figure 8.1 所预期的那样。
然而，由于我们还不知道这个家族的行为是否像一个通用覆盖应该有的那样（例如，它的总空间是否是简单连通的），我们为它使用了一个不同的名字。因此，我们再次重复定义以供参考。

Definition 8.1.1 (Universal Cover of $\mathbb{S}^1$). 通过圆递归定义 $\mathsf{code} : \mathbb{S}^1 \to \mathcal{U}$，其中

$$\mathsf{code}(\mathsf{base}) :\equiv \mathbb{Z}$$
$$\mathsf{ap}_{\mathsf{code}}(\mathsf{loop}) :\equiv \mathsf{ua}(\mathsf{succ})$$

我们简要强调一下这个家族的定义，因为它与经典同伦理论中通常定义覆盖空间的方式是如此不同。要通过圆递归定义一个函数，我们需要在余码中找到一个点和一个环。在这种情况下，余码是 $\mathcal{U}$，我们选择的点是 $\mathbb{Z}$，对应于我们期望的通用覆盖的纤维应该是整数。我们选择的环是后继/前驱（successor/predecessor）的同构，它对应于基座上的环绕上升一级的事实。在这一部分的证明中，等价性是必要的，因为我们需要将 $\mathbb{Z}$ 上的一个 非平凡等价转换为一个恒等式。
我们称这个纤维为"代码"（codes）的纤维，因为它的元素是组合数据，这些数据充当圆上路径的代码：整数 $n$ 为路径编码，该路径在圆上绕行 $n$ 次。
通过这个定义，很容易计算出使用 code 的运输会将 loop 转换为后继函数，将 $\mathsf{loop}^{-1}$ 转换为前驱函数：

Lemma 8.1.2. $\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}, x) = x + 1$ 和 $\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}^{-1}, x) = x - 1$ 。

证明. 对于第一个等式，我们按如下方式计算：

$$\begin{aligned}\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}, x) &= \mathsf{transport}^{A\mapsto A}((\mathsf{code}(\mathsf{loop})), x)\\ &= \mathsf{transport}^{A\mapsto A}(\mathsf{ua}(\mathsf{succ}), x)\end{aligned}$$

$= \mathrm{x} + 1$ 。

第二个等式由第一个得出，因为 $\mathsf{transport}^{B}(p, -)$ 和 $\mathsf{transport}^{B}(p^{-1}, -)$ 总是相互逆转，因此 $\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}^{-1}, -)$ 必须是 succ 的逆函数。 □

我们现在可以看到我们第一个方法的错误之处：我们仅在纤维 $\Omega(\mathbb{S}^1)$ 和 $\mathbb{Z}$ 上定义了 $f$ 和 $g$，而我们应该定义一个整个纤维上的态射（morphism）。在类型理论中，这意味着我们应该定义具有以下类型的函数

$$\prod_{x:\mathbb{S}^1}((\mathsf{base} = x) \to \mathsf{code}(x)) \qquad \text{和/或} \tag{8.1.3}$$
$$\prod_{x:\mathbb{S}^1}(\mathsf{code}(x) \to (\mathsf{base} = x)) \tag{8.1.4}$$

而不仅仅是在 $x$ 为 base 时的特殊情况。这也是类型理论中常见的观察的一个例子：当尝试证明某些归纳类型的特定居民时，通常更容易将陈述泛化，使其涉及 所有此类类型的居民，然后我们可以通过归纳证明。从这个角度来看，$\Omega(\mathbb{S}^1) = \mathbb{Z}$ 的证明符合与在 §2.8 and ?? 中的共产品和自然数的恒等类型的特征化相同的模式。
在这一点上，有两种方法可以完成证明。我们可以继续模仿经典论证，构造 (8.1.3) 或 (8.1.4)（无论哪种都可以），证明总空间之间的同伦等价诱导纤维上的等价，然后证明通用覆盖的总空间是可收缩的。$\Omega(\mathbb{S}^1) = \mathbb{Z}$ 的第一个类型论证遵循了这一模式；我们称之为 同伦理论证明。

然而，后来我们发现有一个替代的证明，它更具类型理论的感觉，更接近于 §2.8 and ?? 中的证明。在这个证明中，我们直接构造了 (8.1.3) 和 (8.1.4)，并通过计算证明它们是互逆的。我们称之为 *编码-解码*证明（encode-decode proof），因为我们将函数 (8.1.3) 和 (8.1.4) 分别称为 *编码*和 *解码*。两个证明都使用了上面给出的相同的覆盖结构。在经典证明中，从总空间的等价诱导出纤维上的等价，而编码-解码证明则明确地构造了一个作为纤维之间映射的逆映射（*解码*）。在经典证明中使用了可收缩性，而在编码-解码证明中使用了路径归纳、圆归纳和整数归纳。这些是用于证明可收缩性的相同工具——实际上，路径归纳 *本质上*是与 $\mathsf{transport}$ 组合的路径纤维的可收缩性——但它们以不同的方式应用。
由于这是一本关于同伦类型论的书，我们首先介绍编码-解码证明。如果一个同伦理论者感到困惑，建议跳到同伦理论证明（§8.1.5）。

### 8.1.4 **编码-解码证明** (The encode-decode proof)

我们首先介绍将路径映射到代码的函数 (8.1.3)：

Definition 8.1.5. 定义 $\mathsf{encode} : \prod_{(x:\mathbb{S}^1)}(\mathsf{base} = x) \to \mathsf{code}(x)$，其中

$$\mathsf{encode}\, p :\equiv \mathsf{transport}^{\mathsf{code}}(p, 0)$$

（我们省略了参数 $x$）。

编码是通过提升路径到通用覆盖中定义的，这确定了一个等价性，然后将结果等价性应用于 0。这个函数有趣的地方在于，它从圆上的环上计算出一个具体的数字，而这个环是使用同伦类型论的抽象群结构表示的。为了获得关于它如何做到这一点的直观理解，请注意，依据上面的引理，$\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}, x)$ 是后继函数，而 $\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}^{-1}, x)$ 是前驱函数。此外，$\mathsf{transport}$ 是函子性的（Chapter 2），所以 $\mathsf{transport}^{\mathsf{code}}(\mathsf{loop} \cdot \mathsf{loop}, -)$ 是

$$(\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}, -)) \circ (\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}, -))$$

依此类推。因此，当 $p$ 是像

$$\mathsf{loop} \cdot \mathsf{loop}^{-1} \cdot \mathsf{loop} \cdot \cdots$$

这样的组合时，$\mathsf{transport}^{\mathsf{code}}(p, -)$ 将计算一个类似于

$$\mathsf{succ} \circ \mathsf{pred} \circ \mathsf{succ} \circ \cdots$$

的函数组合。将这个函数组合应用于 0 将计算路径的*绕线数*（winding number）——路径绕过圆的次数，在逆转被取消之后，标志其方向是正还是负。因此，$\mathsf{encode}$ 的计算行为来自于高级归纳类型和等价性的简化规则，以及 $\mathsf{transport}$ 在组合和逆上的作用。
请注意，实例 $\mathsf{encode}' :\equiv \mathsf{encode}_{\mathsf{base}}$ 的类型为 $(\mathsf{base} = \mathsf{base}) \to \mathbb{Z}$。这将是我们所需等价的一个部分；实际上，它正是我们在 §8.1.1 中定义的函数 $g$。
类似地，函数 (8.1.4) 是 §8.1.1 中函数 $\mathsf{loop}^{-}$ 的推广。

Definition 8.1.6. 通过圆归纳（circle induction）在 $x$ 上定义 $\mathsf{decode} : \prod_{(x:\mathbb{S}^1)} \mathsf{code}(x) \to (\mathsf{base} = x)$。它足以给出一个函数 $\mathsf{code}(\mathsf{base}) \to (\mathsf{base} = \mathsf{base})$，我们使用 $\mathsf{loop}^{-}$，并展示 $\mathsf{loop}^{-}$ 是如何尊重这个环的。

*证明*. 要展示 $\mathsf{loop}^{-}$ 是如何尊重这个环的，足以给出一个从 $\mathsf{loop}^{-}$ 到它本身的路径，该路径位于 $\mathsf{loop}$ 之上。依据依赖路径的定义，这意味着一个路径从

$$\mathsf{transport}^{(x' \mapsto \mathsf{code}(x') \to (\mathsf{base} = x'))}(\mathsf{loop}, \mathsf{loop}^{-})$$

到 $\mathsf{loop}^-$。我们定义这样的路径如下：

$$\begin{aligned}
&\mathsf{transport}^{(x'\mapsto \mathsf{code}(x')\to(\mathsf{base}=x'))}(\mathsf{loop}, \mathsf{loop}^-) \\
&\quad = \mathsf{transport}^{x'\mapsto(\mathsf{base}=x')}(\mathsf{loop}) \circ \mathsf{loop}^- \circ \mathsf{transport}^{\mathsf{code}}(\mathsf{loop}^{-1}) \\
&\quad = (- \cdot \mathsf{loop}) \circ (\mathsf{loop}^-) \circ \mathsf{transport}^{\mathsf{code}}(\mathsf{loop}^{-1}) \\
&\quad = (- \cdot \mathsf{loop}) \circ (\mathsf{loop}^-) \circ \mathsf{pred} \\
&\quad = (n \mapsto \mathsf{loop}^{n-1} \cdot \mathsf{loop})
\end{aligned}$$

在第一行中，我们应用了 transport 的特征，当纤维的外连接是 $\to$ 时，这将 transport 简化为在域和余域类型上与 transport 的预组合和后组合。在第二行中，我们应用了 transport 的特征，当类型族是 $x \mapsto \mathsf{base} = x$ 时，这是路径的后组合。在第三行中，我们使用了 Lemma 8.1.2 中 code 对 $\mathsf{loop}^{-1}$ 的作用。在第四行中，我们只是简化了函数组合。因此，足以证明对于所有 $n$，$\mathsf{loop}^{n-1} \cdot \mathsf{loop} = \mathsf{loop}^n$。这是 Lemma 6.10.12 的一个简单应用，使用了群论的法则。 □

我们现在可以展示 encode 和 decode 是准逆的。以前困难的方向现在变得容易了！

Lemma 8.1.7. *对于所有 $x : \mathbb{S}^1$ 和 $p : \mathsf{base} = x$，$\mathsf{decode}_x(\mathsf{encode}_x(p)) = p$。*

*证明.* 通过路径归纳(path induction),足以证明 $\mathsf{decode}_{\mathsf{base}}(\mathsf{encode}_{\mathsf{base}}(\mathsf{refl}_{\mathsf{base}})) = \mathsf{refl}_{\mathsf{base}}$ 但 $\mathsf{encode}_{\mathsf{base}}(\mathsf{refl}_{\mathsf{base}}) \equiv \mathsf{transport}^{\mathsf{code}}(\mathsf{refl}_{\mathsf{base}}, 0) \equiv 0$ 且 $\mathsf{decode}_{\mathsf{base}}(0) \equiv \mathsf{loop}^0 \equiv \mathsf{refl}_{\mathsf{base}}$。 □

另一个方向并不困难。

Lemma 8.1.8. *对于所有 $x : \mathbb{S}^1$ 和 $c : \mathsf{code}(x)$，我们有 $\mathsf{encode}_x(\mathsf{decode}_x(c)) = c$。*

*证明.* 证明是通过圆归纳（circle induction）进行的。足以证明 base 的情况，因为 loop 的情况是 $\mathbb{Z}$ 中路径之间的路径，这很简单，因为 $\mathbb{Z}$ 是一个集合。
因此，足以证明，对于所有 $n : \mathbb{Z}$，

$$\mathsf{encode}'(\mathsf{loop}^n) = n$$

证明是通过使用 Lemma 6.10.12 的归纳进行的。

- 对于 0 的情况，结果根据定义为真。
- 对于 $n+1$ 的情况，

$$\begin{aligned}
\mathsf{encode}'(\mathsf{loop}^{n+1}) &= \mathsf{encode}'(\mathsf{loop}^n \cdot \mathsf{loop}) && \text{(依据 } \mathsf{loop}^- \text{ 的定义)} \\
&= \mathsf{transport}^{\mathsf{code}}((\mathsf{loop}^n \cdot \mathsf{loop}), 0) && \text{(依据 encode 的定义)} \\
&= \mathsf{transport}^{\mathsf{code}}(\mathsf{loop}, (\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}^n, 0))) && \text{(依据函子性)} \\
&= (\mathsf{transport}^{\mathsf{code}}(\mathsf{loop}^n, 0)) + 1 && \text{(依据 Lemma 8.1.2)} \\
&= n + 1 && \text{(依据归纳假设)}
\end{aligned}$$

- 负数的情况是类似的。 □

最后，我们总结定理。

Theorem 8.1.9. *存在一个等价的族 $\prod_{(x:\mathbb{S}^1)}((\mathsf{base} = x) \simeq \mathsf{code}(x))$。*

*证明.* encode 和 decode 是准逆的，通过 Lemmas 8.1.7 and 8.1.8。 □

在 base 实例化给出

Corollary 8.1.10. $\Omega(\mathbb{S}^1, \mathsf{base}) \simeq \mathbb{Z}$。

一个简单的归纳表明，这个等价性将加法带入组合，因此 $\Omega(\mathbb{S}^1) = \mathbb{Z}$ 作为群体。

Corollary 8.1.11. $\pi_1(\mathbb{S}^1) = \mathbb{Z}$，而 $\pi_n(\mathbb{S}^1) = 0$ 对于 $n > 1$。

证明. 对于 $n = 1$，我们在 Corollary 8.1.10 中概述了证明。对于 $n > 1$，我们有 $\left\|\Omega^n(\mathbb{S}^1)\right\|_0 = \left\|\Omega^{n-1}(\Omega\mathbb{S}^1)\right\|_0 = \left\|\Omega^{n-1}(\mathbb{Z})\right\|_0$。由于 $\mathbb{Z}$ 是一个集合，$\Omega^{n-1}(\mathbb{Z})$ 是可收缩的，因此这是平凡的。 □

### 8.1.5 **同伦理论的证明** (The homotopy-theoretic proof)

在 §8.1.3 中，我们在类型理论中定义了推测的通用覆盖 $\mathsf{code} : \mathbb{S}^1 \to \mathcal{U}$，并在 §8.1.4 中定义了从路径纤维到通用覆盖的映射 $\mathsf{encode} : \prod_{(x:\mathbb{S}^1)}(\mathsf{base} = x) \to \mathsf{code}(x)$。经典证明剩下的内容是证明这个映射在总空间上诱导等价，因为两者都是可收缩的，并从中推导出它在每个纤维上的等价。
在 Lemma 3.11.8 中，我们看到总空间 $\sum_{(x:\mathbb{S}^1)}(\mathsf{base} = x)$ 是可收缩的。对于另一个，我们有：

Lemma 8.1.12. 类型 $\sum_{(x:\mathbb{S}^1)} \mathsf{code}(x)$ 是可收缩的。

证明. 我们应用 Lemma 6.12.2 中的展平引理（flattening lemma），使用以下值：

- $A :\equiv \mathbf{1}$ 和 $B :\equiv \mathbf{1}$，$f$ 和 $g$ 是显然的函数。因此，展平引理中的基高阶归纳类型 $W$ 等价于 $\mathbb{S}^1$。
- $C : A \to \mathcal{U}$ 是常数 $\mathbb{Z}$。
- $D : \prod_{(b:B)}(\mathbb{Z} \simeq \mathbb{Z})$ 是常数 $\mathsf{succ}$。

然后，展平引理中定义的类型族 $P : \mathbb{S}^1 \to \mathcal{U}$ 等价于 $\mathsf{code} : \mathbb{S}^1 \to \mathcal{U}$。因此，展平引理告诉我们，$\sum_{(x:\mathbb{S}^1)} \mathsf{code}(x)$ 等价于一个具有以下生成器的高阶归纳类型，我们将其标记为 $R$：

- 一个函数 $\mathsf{c} : \mathbb{Z} \to R$。
- 对于每个 $z : \mathbb{Z}$，一个路径 $\mathsf{p}_z : \mathsf{c}(z) = \mathsf{c}(\mathsf{succ}(z))$。

我们可以称这个类型为**同伦实数**（homotopical reals）；它与经典证明中的拓扑空间 $\mathbb{R}$ 起着相同的作用。
因此，剩下的就是证明 $R$ 是可收缩的。作为收缩的中心，我们选择 $\mathsf{c}(0)$；我们现在必须证明 $x = \mathsf{c}(0)$ 对于所有 $x : R$。我们通过在 $R$ 上的归纳来做到这一点。首先，当 $x$ 是 $\mathsf{c}(z)$ 时，我们必须给出一个路径 $q_z : \mathsf{c}(0) = \mathsf{c}(z)$，我们可以通过对 $z : \mathbb{Z}$ 的归纳来做到这一点，使用 Lemma 6.10.12：

$$\begin{aligned} q_0 &:\equiv \mathsf{refl}_{\mathsf{c}(0)} \\ q_{n+1} &:\equiv q_n \cdot \mathsf{p}_n && \text{对于 } n \geq 0 \\ q_{n-1} &:\equiv q_n \cdot \mathsf{p}_{n-1}{}^{-1} && \text{对于 } n \leq 0 \end{aligned}$$

其次，我们必须证明对于任何 $z : \mathbb{Z}$，路径 $q_z$ 在 $\mathsf{p}_z$ 上的运输是 $q_{z+1}$。通过路径的运输，这意味着我们需要 $q_z \cdot \mathsf{p}_z = q_{z+1}$。这通过 $z$ 的归纳很容易做到，使用 $q_z$ 的定义。这完成了 $R$ 是可收缩的证明，因此 $\sum_{(x:\mathbb{S}^1)} \mathsf{code}(x)$ 也是可收缩的。 □

Corollary 8.1.13. 由 $\mathsf{encode}$ 诱导的映射：

$$\sum_{(x:\mathbb{S}^1)}(\mathsf{base} = x) \to \sum_{(x:\mathbb{S}^1)} \mathsf{code}(x)$$

是一个等价。

证明. 两种类型都是可收缩的。 □

Theorem 8.1.14. $\Omega(\mathbb{S}^1, \mathsf{base}) \simeq \mathbb{Z}$。

证明. 将 Theorem 4.7.7 应用于 $\mathsf{encode}$，使用 Corollary 8.1.13。 □

本质上，这两个证明没有太大不同：编码-解码的证明可以看作是同伦理论证明的“简化”或“解包”。每个都有其优势；两者之间的相互作用是该主题的一部分兴趣所在。

### 8.1.6 通用覆盖作为恒等系统 (The universal cover as an identity system)

请注意，纤维 $\mathsf{code} : \mathbb{S}^1 \to \mathcal{U}$ 与 $0 : \mathsf{code}(\mathsf{base})$ 一起是 Definition 5.8.1 中所述的*指向谓词*（pointed predicate）。从这个角度来看，我们可以看到，在 §8.1.4 中的编码-解码证明由证明 code 满足 Theorem 5.8.2(iii) 组成，而在 §8.1.5 中的同伦理论证明则由证明它满足 Theorem 5.8.2(iv) 组成。这暗示了一种第三种方法。

Theorem 8.1.15. *对于* $\mathsf{base} : \mathbb{S}^1$，*对偶* $(\mathsf{code}, 0)$ *是 Definition 5.8.1 中所定义的恒等系统*（identity system）。

*证明.* 令 $D : \prod_{(x:\mathbb{S}^1)} \mathsf{code}(x) \to \mathcal{U}$ 和 $d : D(\mathsf{base}, 0)$ 已给出；我们要定义一个函数 $f : \prod_{(x:\mathbb{S}^1)} \prod_{(c:\mathsf{code}(x))} D(x, c)$。通过圆归纳，足以指定 $f(\mathsf{base}) : \prod_{(c:\mathsf{code}(\mathsf{base}))} D(\mathsf{base}, c)$ 并验证 $\mathsf{loop}_*(f(\mathsf{base})) = f(\mathsf{base})$。
当然，$\mathsf{code}(\mathsf{base}) \equiv \mathbb{Z}$。通过 Lemma 8.1.2 和对 $n$ 的归纳，我们可以为任何整数 $n$ 获得一个路径 $p_n : \mathsf{transport}^{\mathsf{code}}(\mathsf{loop}^n, 0) = n$。因此，通过 $\Sigma$-类型的路径，我们在 $\sum_{(x:\mathbb{S}^1)} \mathsf{code}(x)$ 中有一个路径 $\mathsf{pair}^{=}(\mathsf{loop}^n, p_n) : (\mathsf{base}, 0) = (\mathsf{base}, n)$。在与 $D$ 相关联的纤维 $\widehat{D} : (\sum_{(x:\mathbb{S}^1)} \mathsf{code}(x)) \to \mathcal{U}$ 中沿着这个路径运输 $d$，我们获得 $D(\mathsf{base}, n)$ 的一个元素。我们定义这个元素为 $f(\mathsf{base})(n)$：

$$f(\mathsf{base})(n) :\equiv \mathsf{transport}^{\widehat{D}}(\mathsf{pair}^{=}(\mathsf{loop}^n, p_n), d)$$

现在我们需要 $\mathsf{transport}^{\lambda x.\, \prod_{(c:\mathsf{code}(x))} D(x,c)}(\mathsf{loop}, f(\mathsf{base})) = f(\mathsf{base})$。通过 Lemma 2.7.7，这意味着我们需要证明对于任何 $n : \mathbb{Z}$，

$$\mathsf{transport}^{\widehat{D}}(\mathsf{pair}^{=}(\mathsf{loop}, \mathsf{refl}_{\mathsf{loop}_*(n)}), f(\mathsf{base})(n)) =_{D(\mathsf{base}, \mathsf{loop}_*(n))} f(\mathsf{base})(\mathsf{loop}_*(n))$$

现在我们有一个路径 $q : \mathsf{loop}_*(n) = n + 1$，所以沿着这个路径运输，足以证明

$$\mathsf{transport}^{D(\mathsf{base})}(q, \mathsf{transport}^{\widehat{D}}(\mathsf{pair}^{=}(\mathsf{loop}, \mathsf{refl}_{\mathsf{loop}_*(n)}), f(\mathsf{base})(n))) \\ =_{D(\mathsf{base}, n+1)} \mathsf{transport}^{D(\mathsf{base})}(q, f(\mathsf{base})(\mathsf{loop}_*(n)))$$

通过关于运输和依赖应用的几个引理，这等效于

$$\mathsf{transport}^{\widehat{D}}(\mathsf{pair}^{=}(\mathsf{loop}, q), f(\mathsf{base})(n)) =_{D(\mathsf{base}, n+1)} f(\mathsf{base})(n+1)$$

然而，展开 $f(\mathsf{base})$ 的定义，我们有

$$\begin{aligned} \mathsf{transport}^{\widehat{D}}(\mathsf{pair}^{=}(\mathsf{loop}, q), f(\mathsf{base})(n)) &= \mathsf{transport}^{\widehat{D}}(\mathsf{pair}^{=}(\mathsf{loop}, q), \mathsf{transport}^{\widehat{D}}(\mathsf{pair}^{=}(\mathsf{loop}^n, p_n), d)) \\ &= \mathsf{transport}^{\widehat{D}}(\mathsf{pair}^{=}(\mathsf{loop}^n, p_n) \cdot \mathsf{pair}^{=}(\mathsf{loop}, q), d) \\ &= \mathsf{transport}^{\widehat{D}}(\mathsf{pair}^{=}(\mathsf{loop}^{n+1}, p_{n+1}), d) \\ &= f(\mathsf{base})(n+1) \end{aligned}$$

我们使用了运输的函子性，$\Sigma$-类型中的组合的特征（这是读者的练习），以及一个将 $p_n$ 和 $q$ 关联到 $p_{n+1}$ 的引理，我们将留给读者陈述和证明。
这完成了函数 $f : \prod_{(x:\mathbb{S}^1)} \prod_{(c:\mathsf{code}(x))} D(x, c)$ 的构造。由于

$$f(\mathsf{base}, 0) \equiv \mathsf{pair}^{=}(\mathsf{loop}^0, p_0)_*(d) = \mathsf{refl}_{\mathsf{base}*}(d) = d$$

我们已经证明了 $(\mathsf{code}, 0)$ 是一个恒等系统。 □

Corollary 8.1.16. *对于任何* $x : \mathbb{S}^1$，*我们有* $(\mathsf{base} = x) \simeq \mathsf{code}(x)$。

*证明.* 通过 Theorem 5.8.2。 □

当然，这个证明也包含了前两个证明的基本元素。大致上，我们可以说它统一了 Definition 8.1.6 and Lemma 8.1.8 的证明，在一般情况下只执行一次所需的归纳论证。

Remark 8.1.17. 请注意，所有上述证明 $\pi_1(\mathbb{S}^1) \simeq \mathbb{Z}$ 都在本质上使用了等价性公理。这是不可避免的：等价性或类似的东西是*必要的*，以证明 $\pi_1(\mathbb{S}^1) \simeq \mathbb{Z}$。在没有等价性的情况下，可以假设"所有类型都是集合"的陈述（又称为"恒等证明的唯一性"或"Axiom K"，如 §7.2 中讨论的），该陈述则意味着 $\pi_1(\mathbb{S}^1) \simeq \mathbf{1}$。实际上，$\pi_1(\mathbb{S}^1)$ 的（非）平凡性正好检测所有类型是否为集合：Lemma 6.4.1 的证明反过来表明，如果 $\mathsf{loop} = \mathsf{refl}_{\mathsf{base}}$，则所有类型都是集合。

## 8.2 **悬挂的连通性** (Connectedness of suspensions)

回顾在 §7.5 中提到的，如果一个类型 $A$ 被称为 $n$-**连通的** ($n$-connected)，则 $\|A\|_n$ 是可收缩的 (contractible)。本节的目的是证明悬挂操作 (suspension operation) 会增加连通性，如 §6.5 中所述。

Theorem 8.2.1. *如果 $A$ 是 $n$-连通的，那么 $A$ 的悬挂* (suspension) *是 $(n+1)$-连通的。*

*证明.* 我们在 §6.8 中提到，$A$ 的悬挂是推挤 (pushout) $\mathbf{1} \sqcup^A \mathbf{1}$，因此我们需要证明以下类型是可收缩的：

$$\left\|\mathbf{1} \sqcup^A \mathbf{1}\right\|_{n+1}$$

根据 Theorem 7.4.12，我们知道 $\left\|\mathbf{1} \sqcup^A \mathbf{1}\right\|_{n+1}$ 是 $(n+1)$-$\mathsf{Type}$ 中以下图表的推挤：

$$\begin{array}{ccc} \|A\|_{n+1} & \longrightarrow & \|\mathbf{1}\|_{n+1} \\ \downarrow & & \\ \|\mathbf{1}\|_{n+1} & & \end{array}$$

鉴于 $\|\mathbf{1}\|_{n+1} = \mathbf{1}$，$\left\|\mathbf{1} \sqcup^A \mathbf{1}\right\|_{n+1}$ 的类型在 $(n+1)$-$\mathsf{Type}$ 中也是以下图表的推挤（因为两个图表是相等的）：

$$\mathscr{D} = \begin{array}{ccc} \|A\|_{n+1} & \longrightarrow & \mathbf{1} \\ \downarrow & & \\ \mathbf{1} & & \end{array}$$

我们现在将证明在 $(n+1)$-$\mathsf{Type}$ 中 $\mathbf{1}$ 也是 $\mathscr{D}$ 的推挤。设 $E$ 是一个 $(n+1)$-截断的类型；我们需要证明以下映射是一个等价 (equivalence)：

$$\begin{cases} (\mathbf{1} \to E) & \longrightarrow & \mathsf{cocone}_{\mathscr{D}}(E) \\ y & \longmapsto & (y, y, \lambda u.\, \mathsf{refl}_{y(\star)}) \end{cases}$$

我们回顾一下，$\mathsf{cocone}_{\mathscr{D}}(E)$ 是以下类型：

$$\sum_{(f:\mathbf{1}\to E)} \sum_{(g:\mathbf{1}\to E)} (\|A\|_{n+1} \to (f(\star) =_E g(\star)))$$

映射 $\begin{cases} (\mathbf{1} \to E) & \longrightarrow & E \\ f & \longmapsto & f(\star) \end{cases}$ 是一个等价映射，因此我们也有

$$\mathsf{cocone}_{\mathscr{D}}(E) = \sum_{(x:E)} \sum_{(y:E)} (\|A\|_{n+1} \to (x =_E y))$$

现在 $A$ 是 $n$-连通的，因此 $\|A\|_{n+1}$ 也是 $n$-连通的，因为 $\left\|\|A\|_{n+1}\right\|_n = \|A\|_n = \mathbf{1}$，而 $(x =_E y)$ 是 $n$-截断的，因为 $E$ 是 $(n+1)$-截断的。因此，根据 Corollary 7.5.9，以下映射是一个等价映射：

$$\begin{cases} (x =_E y) & \longrightarrow & (\|A\|_{n+1} \to (x =_E y)) \\ p & \longmapsto & \lambda z.\, p \end{cases}$$

因此我们有

$$\mathsf{cocone}_{\mathscr{D}}(E) = \sum_{(x:E)} \sum_{(y:E)} (x =_E y)$$

但是以下映射是一个等价映射：

$$\begin{cases} E & \longrightarrow & \sum_{(x:E)} \sum_{(y:E)} (x =_E y) \\ x & \longmapsto & (x, x, \mathsf{refl}_x) \end{cases}$$

因此

$$\mathsf{cocone}_{\mathscr{D}}(E) = E$$

最后我们得到一个等价映射

$$(\mathbf{1} \to E) \simeq \mathsf{cocone}_{\mathscr{D}}(E)$$

我们现在可以展开定义以得到此映射的明确表达式，并且我们可以轻松地看到，这正是我们在开始时所得到的映射。
因此我们证明了 $\mathbf{1}$ 是 $(n+1)$-$\mathsf{Type}$ 中 $\mathscr{D}$ 的推挤。使用推挤的唯一性，我们得到 $\left\|\mathbf{1} \sqcup^A \mathbf{1}\right\|_{n+1} = \mathbf{1}$，这证明了 $A$ 的悬挂是 $(n+1)$-连通的。 □

Corollary 8.2.2. *对于所有 $n : \mathbb{N}$，球体 $\mathbb{S}^n$ 是 $(n-1)$-连通的。*

*证明.* 我们通过归纳法证明这一点。对于 $n = 0$，我们必须证明 $\mathbb{S}^0$ 仅仅是居住的 (inhabited)，这是显然的。设 $n : \mathbb{N}$ 使得 $\mathbb{S}^n$ 是 $(n-1)$-连通的。根据定义，$\mathbb{S}^{n+1}$ 是 $\mathbb{S}^n$ 的悬挂，因此根据前面的引理，$\mathbb{S}^{n+1}$ 是 $n$-连通的。 □

Chapter 9

# 范畴论 (Category theory)

在数学的各个分支中，范畴论可能是最不适合基于集合论基础的学科之一。一个问题是，大多数范畴论的内容在弱于等同性的“相同”概念下是不变的，例如范畴中的同构或者范畴之间的等价，而集合论无法捕捉这一点。但是，对于类型论中的等同性，无效性公理 (univalence axiom) 通过将等同性与等价性等同解决了类似的问题。因此，在无效性基础 (univalent foundations) 中，考虑一种“范畴”概念是合理的，在这种范畴中，对象的等同性与同构被以类似的方式识别。
忽略大小问题，在基于集合的数学中，范畴由一个对象的集合 $A_0$ 和对于每个 $x, y \in A_0$ 的集合 $\hom_A(x,y)$ 组成。在无效性基础下，一个“朴素的”范畴定义只是模仿这个结构，用一个类型的对象和类型的态射。如果我们允许这些类型包含任意的高阶同伦，则应当引入更高的连贯条件，从而导向某种 $(\infty,1)$-范畴的概念，但目前我们的目标更加温和。我们只考虑 1-范畴，因此我们将 $\hom_A(x,y)$ 的类型限制为集合，即 0-类型。如果我们不施加进一步的条件，我们将这种概念称为*预范畴*。
如果我们添加条件，使得对象类型 $A_0$ 是一个集合，那么我们得到一个行为与传统集合论定义相似的定义。我们称这种概念为*严格范畴* (strict category)。或者，我们可以要求无效性公理的广义版本，将 $(x =_{A_0} y)$ 与 $x$ 到 $y$ 的同构类型 $\mathsf{iso}(x,y)$ 识别。由于我们认为后一种选择通常是“正确的”定义，因此我们简单地称它为*范畴*。
一个关于这三种范畴概念的区别的好例子是“每个满忠实且本质满的函子是范畴等价”的命题，在经典基于集合的范畴论中，该命题等价于选择公理。

(i) 对于严格范畴，这一命题仍然等价于选择公理。
(ii) 对于预范畴，没有一致的选择公理可以使其成立。
(iii) 对于范畴，它可以*不使用*任何选择公理的情况下被证明。

我们将在本章中证明最后一个命题，并讨论范畴的其他令人愉快的性质，例如等价范畴是相等的（作为范畴类型的元素）。我们还将描述一种将预范畴 $A$“饱和”成范畴 $\widehat{A}$ 的通用方式，我们称之为它的 Rezk *完成*，尽管它也可以合理地称为*栈完成*（见注释）。
Rezk 完成还进一步阐明了范畴等价的概念。例如，函子 $A \to \widehat{A}$ 始终是满忠实且本质满的，因此是“弱等价”。因此，一个预范畴是范畴的必要和充分条件是它“视”所有满忠实且本质满的函子为等价；因此我们的“范畴”概念已经内在于“满忠实且本质满函子”的概念之中。
我们假定读者对经典范畴论有一定的基本了解。请记住，每当我们写到 $\mathcal{U}$ 时，它表示某个类型的宇宙，但在不同情况下可能是不同的宇宙；我们所说的一切都对任何一致的宇宙层次的选择成立。我们将使用 Chapters 1 and 2 中的同伦类型论的基本概念以及 Chapter 3 中的命题截断，但不会涉及 Part I 的大部分内容，除非我们将在第二种 Rezk 完成的构造中使用高阶归纳类型。

## 9.1 范畴与预范畴 (Categories and precategories)

在经典数学中，有许多等价的范畴定义。在我们的情况下，由于我们有依赖类型，因此将箭头作为对象的类型族是很自然的。这与范畴论中 Hom-类型的使用方式相匹配：我们在比较两个箭头之前从未

考虑过它们的域和陪域是否一致。此外，显然，对于 1-范畴理论，所有的 Hom-类型都应该是集合。这引导我们到以下定义。

Definition 9.1.1. 一个**预范畴** (precategory)$A$ 包含以下内容。

(i) 一个类型 $A_0$，其元素称为**对象** (objects)。我们记 $a : A$ 表示 $a : A_0$。
(ii) 对于每个 $a, b : A$，有一个集合 $\hom_A(a, b)$，其元素称为**箭头** (arrows) 或**态射** (morphisms)。
(iii) 对于每个 $a : A$，有一个态射 $1_a : \hom_A(a, a)$，称为**恒等态射** (identity morphism)。
(iv) 对于每个 $a, b, c : A$，有一个函数

$$\hom_A(b, c) \to \hom_A(a, b) \to \hom_A(a, c)$$

称为**复合** (composition)，记为中缀形式 $g \mapsto f \mapsto g \circ f$，有时简单记为 $gf$。
(v) 对于每个 $a, b : A$ 和 $f : \hom_A(a, b)$，我们有 $f = 1_b \circ f$ 和 $f = f \circ 1_a$。
(vi) 对于每个 $a, b, c, d : A$ 和

$$f : \hom_A(a, b), \qquad g : \hom_A(b, c), \qquad h : \hom_A(c, d),$$

我们有 $h \circ (g \circ f) = (h \circ g) \circ f$。

预范畴的一个问题是，对于对象 $a, b : A$，我们有两种可能不同的"相同"概念。一方面，我们有类型 $(a =_{A_0} b)$。但另一方面，有标准的范畴论中的同构 (isomorphism) 的概念。

Definition 9.1.2. 态射 $f : \hom_A(a, b)$ 是一个**同构** (isomorphism)如果存在态射 $g : \hom_A(b, a)$ 使得 $g \circ f = 1_a$ 和 $f \circ g = 1_b$。我们记 $a \cong b$ 表示这种同构的类型。

Lemma 9.1.3. *对于任意 $f : \hom_A(a, b)$，类型"$f$ 是同构"是一个单命题。因此，对于任意 $a, b : A$，类型 $a \cong b$ 是一个集合。*

*证明.* 假设给定 $g : \hom_A(b, a)$ 和 $\eta : (1_a = g \circ f)$ 以及 $\epsilon : (f \circ g = 1_b)$，类似地有 $g'$，$\eta'$ 和 $\epsilon'$。我们必须证明 $(g, \eta, \epsilon) = (g', \eta', \epsilon')$。但由于所有同态集都是集合，因此它们的等同性类型是单命题，所以只需证明 $g = g'$。为此我们有

$$g' = 1_a \circ g' = (g \circ f) \circ g' = g \circ (f \circ g') = g \circ 1_b = g$$

使用 $\eta$ 和 $\epsilon'$。 □

如果 $f : a \cong b$，我们记 $f^{-1}$ 表示它的逆，由 Lemma 9.1.3 唯一确定。
在预范畴中，这两种相同性概念之间的唯一关系如下。

Lemma 9.1.4 (idtoiso). *如果 $A$ 是一个预范畴，$a, b : A$，则*

$$(a = b) \to (a \cong b)$$

。

*证明.* 通过等同性归纳，我们可以假设 $a$ 和 $b$ 是相同的。但是在这种情况下，我们有 $1_a : \hom_A(a, a)$，它显然是一个同构。 □

显然，这种情况类似于促使我们引入无效性公理的问题。实际上，我们有以下例子：

Example 9.1.5. 存在一个预范畴 $\mathcal{S}et$ ，其对象的类型是 $\mathsf{Set}$ ，并且 $\hom_{\mathcal{S}et}(A, B) :\equiv (A \to B)$。恒等态射是恒等函数，复合是函数复合。对于这个预范畴，Lemma 9.1.4 等同于（限制到集合的）来自 ?? 中的 $\mathsf{idtoeqv}$。
当然，更准确地说，我们应该称这个范畴为 $\mathcal{S}et_{\mathcal{U}}$，因为它的对象仅是相对于某个宇宙 $\mathcal{U}$ 的小集合。
因此，引入以下定义是合乎逻辑的。

Definition 9.1.6. 一个**范畴** (category)是一个预范畴，使得对于所有 $a, b : A$，Lemma 9.1.4 中的函数 $\mathsf{idtoiso}_{a,b}$ 是一个等价。

特别是，在一个范畴中，如果 $a \cong b$，那么 $a = b$。

Example 9.1.7. 无效性公理立即蕴含 $\mathcal{S}et$ 是一个范畴。还可以通过无效性公理证明，任何集合层次的结构（如群、环、拓扑空间等）的预范畴都是一个范畴；见 §9.8。

我们还注意到以下内容。

Lemma 9.1.8. *在一个范畴中，对象的类型是一个 1-类型。*

*证明.* 证明对于任意 $a, b : A$，类型 $a = b$ 是一个集合即可。但 $a = b$ 等价于 $a \cong b$，而后者是一个集合。 □

我们记 $\mathsf{isotoid}$ 为 Lemma 9.1.4 中从 $(a \cong b)$ 到 $(a = b)$ 的逆。以下关系非常重要。

Lemma 9.1.9. *对于 $p : a = a'$ 和 $q : b = b'$ 以及 $f : \hom_A(a, b)$，我们有*

$$(p, q)_*(f) = \mathsf{idtoiso}(q) \circ f \circ \mathsf{idtoiso}(p)^{-1} \tag{9.1.10}$$

*证明.* 通过归纳法，我们可以假设 $p$ 和 $q$ 分别是 $\mathsf{refl}_a$ 和 $\mathsf{refl}_b$。在这种情况下，(9.1.10) 的左边简化为 $f$。但根据定义，$\mathsf{idtoiso}(\mathsf{refl}_a)$ 是 $1_a$，$\mathsf{idtoiso}(\mathsf{refl}_b)$ 是 $1_b$，因此 (9.1.10) 的右边为 $1_b \circ f \circ 1_a$，这等于 $f$。 □

类似地，我们可以证明

$$\mathsf{idtoiso}(p^{-1}) = (\mathsf{idtoiso}(p))^{-1} \tag{9.1.11}$$

$$\mathsf{idtoiso}(p \cdot q) = \mathsf{idtoiso}(q) \circ \mathsf{idtoiso}(p) \tag{9.1.12}$$

$$\mathsf{isotoid}(f \circ e) = \mathsf{isotoid}(e) \cdot \mathsf{isotoid}(f) \tag{9.1.13}$$

等等。

Example 9.1.14. 一个预范畴，其中每个集合 $\hom_A(a, b)$ 是单命题等价于一个类型 $A_0$，配有一个反身 $(a \le a)$ 和传递 $(a \le b$ 且 $b \le c$，则 $a \le c)$ 的单命题关系“$\le$”。我们称这种结构为**预序** (preorder)。在预序中，证据 $f : a \le b$ 是一个同构当且仅当存在某种证据 $g : b \le a$。因此，$a \cong b$ 是单命题，即 $a \le b$ 且 $b \le a$。因此，一个预序 $A$ 是范畴当且仅当 (1) 每个类型 $a = b$ 是单命题，(2) 对于任何 $a, b : A_0$ 存在一个函数 $(a \cong b) \to (a = b)$。换句话说，$A_0$ 必须是一个集合，且 $\le$ 必须是反对称的（如果 $a \le b$ 且 $b \le a$，则 $a = b$）。我们称这种结构为 (**偏序**) (partial) order 或**偏序集** (poset)。

Example 9.1.15. 如果 $A$ 是一个范畴，那么 $A_0$ 是一个集合当且仅当对于任意 $a, b : A_0$，类型 $a \cong b$ 是单命题。给定 $A$ 是一个范畴，这等价于说每个自同构在 $A$ 中都是恒等箭头。另一方面，如果 $A$ 是一个预范畴且 $A_0$ 是一个集合，那么 $A$ 是范畴当且仅当它是骨架 (skeletal)（任何两个同构的对象是相等的）*并且*每个自同构是恒等箭头。这种类型的范畴有时被称为 gaunt 范畴 [?]。对于我们的范畴，没有真正的“骨架性”概念，除非认为 Definition 9.1.6 本身是这样的。

Example 9.1.16. 对于任意 1-类型 $X$，存在一个范畴，其对象类型为 $X$，且 $\hom(x, y) :\equiv (x = y)$。如果 $X$ 是一个集合，我们称其为**离散范畴** (discrete category)。通常我们称其为**群组** (groupoid)（见 Exercise 9.6）。

Example 9.1.17. 对于*任意*类型 $X$，存在一个预范畴，其对象类型为 $X$，且 $\hom(x, y) :\equiv \|x = y\|_0$。复合运算

$$\|y = z\|_0 \to \|x = y\|_0 \to \|x = z\|_0$$

通过同伦连接从复合 $(y = z) \to (x = y) \to (x = z)$ 归纳定义。我们称其为 $X$ 的**基本预群组** (fundamental pregroupoid)。（实际上，我们在 ?? 中已经遇到了它；另见 Exercise 9.11。）

Example 9.1.18. 存在一个预范畴，其对象类型是 $\mathcal{U}$，且 $\hom(X, Y) :\equiv \|X \to Y\|_0$，复合通过同伦连接从普通复合 $(Y \to Z) \to (X \to Y) \to (X \to Z)$ 归纳定义。我们称其为**类型的同伦预范畴** (homotopy precategory of types)。

Example 9.1.19. 令 $\mathcal{R}el$ 为以下预范畴：

- 其对象是集合。
- $\hom_{\mathcal{R}el}(X,Y) = X \to Y \to \mathsf{Prop}$。
- 对于集合 $X$，我们有 $1_X(x,x') :\equiv (x = x')$。
- 对于 $R : \hom_{\mathcal{R}el}(X,Y)$ 和 $S : \hom_{\mathcal{R}el}(Y,Z)$，它们的复合定义为

$$(S \circ R)(x,z) :\equiv \left\| \sum_{y:Y} R(x,y) \times S(y,z) \right\|$$

假设 $R : \hom_{\mathcal{R}el}(X,Y)$ 是一个同构，其逆为 $S$。我们观察到以下几点。

(i) 如果 $R(x,y)$ 且 $S(y',x)$，则 $(R \circ S)(y',y)$，从而 $y' = y$。类似地，如果 $R(x,y)$ 且 $S(y,x')$，则 $x = x'$。

(ii) 对于任意 $x$，我们有 $x = x$，因此 $(S \circ R)(x,x)$。因此，仅存在某个 $y : Y$ 使得 $R(x,y)$ 且 $S(y,x)$。

(iii) 假设 $R(x,y)$。根据 (ii)，仅存在某个 $y'$ 使得 $R(x,y')$ 且 $S(y',x)$。但随后根据 (i)，仅 $y' = y$，因此由于 $Y$ 是集合，所以 $y' = y$。因此，通过沿此等同性传递 $S(y',x)$，我们有 $S(y,x)$。结论是 $R(x,y) \to S(y,x)$。类似地，$S(y,x) \to R(x,y)$。

(iv) 如果 $R(x,y)$ 且 $R(x,y')$，则根据 (iii)，$S(y',x)$，因此根据 (i)，$y = y'$。因此，对于任意 $x$，至多存在一个 $y$ 使得 $R(x,y)$。根据 (ii)，仅存在这样的 $y$，因此存在这样的 $y$。

结论是，如果 $R : \hom_{\mathcal{R}el}(X,Y)$ 是同构，则对于每个 $x : X$，存在且仅存在一个 $y : Y$ 使得 $R(x,y)$，反之亦然。因此，存在一个函数 $f : X \to Y$，将每个 $x$ 映射到此 $y$，该函数是等价的；因此 $X = Y$。经过更多的推导，我们可以得出 $\mathcal{R}el$ 是一个范畴。

我们现在可能将自己限制为仅考虑范畴而非预范畴。相反，我们将对预范畴和范畴进行许多概念的开发，以强调范畴相比于预范畴以及经典数学中的普通范畴如何更加良好。

我们还将在 §§9.6–9.7 中看到，在稍微异质的背景下，某些类型的预范畴除了范畴之外还有其他用途，每种预范畴以不同的方式“修复”对象的等同性。这进一步强调了预范畴的“预”性：它们是从中可以定义多种重要范畴结构的原材料。

## 9.2 函子与变换 (Functors and transformations)

以下定义非常明显，无需修改。

Definition 9.2.1. 设 $A$ 和 $B$ 是前范畴 (precategories)。一个**函子** (functor) $F : A \to B$ 由以下部分组成：

(i) 一个函数 $F_0 : A_0 \to B_0$，通常也记作 $F$。

(ii) 对每个 $a, b : A$，一个函数 $F_{a,b} : \hom_A(a,b) \to \hom_B(Fa, Fb)$，通常也记作 $F$。

(iii) 对每个 $a : A$，我们有 $F(1_a) = 1_{Fa}$。

(iv) 对每个 $a, b, c : A$ 以及 $f : \hom_A(a,b)$ 和 $g : \hom_A(b,c)$，我们有

$$F(g \circ f) = Fg \circ Ff$$

注意，通过同一性归纳 (induction on identity)，函子还保持 $\mathsf{idtoiso}$。

Definition 9.2.2. 对于函子 $F, G : A \to B$，一个**自然变换** (natural transformation) $\gamma : F \to G$ 由以下部分组成：

(i) 对每个 $a : A$，一个态射 $\gamma_a : \hom_B(Fa, Ga)$ (称为"分量")。
(ii) 对每个 $a, b : A$ 和 $f : \hom_A(a, b)$，我们有 $Gf \circ \gamma_a = \gamma_b \circ Ff$ (称为"自然性公理")。

由于每个类型 $\hom_B(Fa, Gb)$ 是一个集合，因此它的同一性类型 (identity type) 是一个纯命题 (mere proposition)。因此，自然性公理是一个纯命题，所以自然变换的同一性由它们的分量的同一性决定。特别地，对于任意 $F$ 和 $G$，从 $F$ 到 $G$ 的自然变换的类型也是一个集合。
同样，函子的同一性由函数 $A_0 \to B_0$ 的同一性（以及在此基础上运输的同源集合上的相应函数）决定。

Definition 9.2.3. 对于前范畴 $A, B$，存在一个称为**函子前范畴** (functor precategory) 的前范畴 $B^A$，其定义如下：

- $(B^A)_0$ 是从 $A$ 到 $B$ 的函子的类型。
- $\hom_{B^A}(F, G)$ 是从 $F$ 到 $G$ 的自然变换的类型。

证明. 我们定义 $(1_F)_a :\equiv 1_{Fa}$。自然性遵循前范畴的单位公理。对于 $\gamma : F \to G$ 和 $\delta : G \to H$，我们定义 $(\delta \circ \gamma)_a :\equiv \delta_a \circ \gamma_a$。自然性遵循结合性。同样，$B^A$ 的单位和结合性公理也遵循 $B$ 的单位和结合性公理。 □

Lemma 9.2.4. *一个自然变换 $\gamma : F \to G$ 如果且仅如果每个 $\gamma_a$ 在 $B$ 中是一个同构，那么它在 $B^A$ 中是一个同构。*

证明. 如果 $\gamma$ 是一个同构，那么我们有 $\delta : G \to F$ 作为它的逆变换。根据在 $B^A$ 中的合成定义，$(\delta\gamma)_a \equiv \delta_a\gamma_a$，同样地 $(\gamma\delta)_a \equiv \gamma_a\delta_a$。因此，$\delta\gamma = 1_F$ 和 $\gamma\delta = 1_G$ 意味着 $\delta_a\gamma_a = 1_{Fa}$ 和 $\gamma_a\delta_a = 1_{Ga}$，因此 $\gamma_a$ 是一个同构。
反过来，假设每个 $\gamma_a$ 是一个同构，设它的逆为 $\delta_a$。我们定义一个自然变换 $\delta : G \to F$，其分量为 $\delta_a$；对于自然性公理，我们有

$$Ff \circ \delta_a = \delta_b \circ \gamma_b \circ Ff \circ \delta_a = \delta_b \circ Gf \circ \gamma_a \circ \delta_a = \delta_b \circ Gf.$$

现在，由于自然变换的合成和同一性是由它们的分量决定的，我们有 $\gamma\delta = 1_G$ 和 $\delta\gamma = 1_F$。 □

以下结果是基础性的。

Theorem 9.2.5. *如果 $A$ 是一个前范畴而 $B$ 是一个范畴* (category)，*那么 $B^A$ 是一个范畴。*

证明. 令 $F, G : A \to B$；我们必须证明 $\mathsf{idtoiso} : (F = G) \to (F \cong G)$ 是一个等价。
为了给它一个逆，我们假设 $\gamma : F \cong G$ 是一个自然同构。然后对于任意 $a : A$，我们有一个同构 $\gamma_a : Fa \cong Ga$，因此一个同一性 $\mathsf{isotoid}(\gamma_a) : Fa = Ga$。通过函数扩展性 (function extensionality)，我们有一个同一性 $\bar{\gamma} : F_0 =_{(A_0 \to B_0)} G_0$。
现在，由于函子的最后两个公理是纯命题，为了证明 $F = G$，我们只需证明对于任意 $a, b : A$，函数

$$F_{a,b} : \hom_A(a, b) \to \hom_B(Fa, Fb) \qquad \text{和}$$
$$G_{a,b} : \hom_A(a, b) \to \hom_B(Ga, Gb)$$

在沿着 $\bar{\gamma}$ 运输时变得相等。通过函数扩展性的计算，当应用于 $a$ 时，$\bar{\gamma}$ 变得等于 $\mathsf{isotoid}(\gamma_a)$。但通过 Lemma 9.1.9，沿 $\mathsf{isotoid}(\gamma_a)$ 和 $\mathsf{isotoid}(\gamma_b)$ 运输 $Ff : \hom_B(Fa, Fb)$ 等于复合 $\gamma_b \circ Ff \circ (\gamma_a)^{-1}$，这通过 $\gamma$ 的自然性等于 $Gf$。
这完成了从 $(F \cong G) \to (F = G)$ 的函数的定义。现在考虑复合

$$(F = G) \to (F \cong G) \to (F = G).$$

由于同源集 (hom-sets) 是集合，它们的同一性类型是纯命题，因此为了证明两个同一性 $p, q : F = G$ 是相等的，只需证明 $p =_{F_0 = G_0} q$。但在 $\bar{\gamma}$ 的定义中，如果 $\gamma$ 是 $\mathsf{idtoiso}(p)$ 的形式，那么 $\gamma_a$ 将等于

$\mathsf{idtoiso}(p_a)$（这可以通过对 $p$ 归纳轻松证明）。因此，$\mathsf{isotoid}(\gamma_a)$ 将等于 $p_a$，所以通过函数扩展性，我们将有 $\bar{\gamma} = p$，这正是我们需要的。
最后，考虑复合

$$(F \cong G) \to (F = G) \to (F \cong G).$$

由于自然变换的同一性可以通过分量来测试，只需证明对于每个 $a$，我们有 $\mathsf{idtoiso}(\bar{\gamma})_a = \gamma_a$。但如上所述，我们有 $\mathsf{idtoiso}(\bar{\gamma})_a = \mathsf{idtoiso}((\bar{\gamma})_a)$，而 $(\bar{\gamma})_a = \mathsf{isotoid}(\gamma_a)$ 通过函数扩展性计算得到。由于 $\mathsf{isotoid}$ 和 $\mathsf{idtoiso}$ 是逆函数，我们有 $\mathsf{idtoiso}(\bar{\gamma})_a = \gamma_a$，如所希望的那样。 □

特别地，范畴（相对于前范畴）之间的自然同构函子是相等的。

我们现在定义所有常见的函子和自然变换的合成方式。

Definition 9.2.6. 对于函子 $F : A \to B$ 和 $G : B \to C$，它们的复合 $G \circ F : A \to C$ 定义如下：

- 复合 $(G_0 \circ F_0) : A_0 \to C_0$
- 对每个 $a, b : A$，复合

$$(G_{Fa,Fb} \circ F_{a,b}) : \hom_A(a, b) \to \hom_C(GFa, GFb)$$

公理很容易验证。

Definition 9.2.7. 对于函子 $F : A \to B$ 和 $G, H : B \to C$ 以及自然变换 $\gamma : G \to H$，复合 $(\gamma F) : GF \to HF$ 定义如下：

- 对每个 $a : A$，分量为 $\gamma_{Fa}$。

自然性很容易验证。同样，对于上述的 $\gamma$ 和 $K : C \to D$，复合 $(K\gamma) : KG \to KH$ 定义如下：

- 对每个 $b : B$，分量为 $K(\gamma_b)$。

Lemma 9.2.8. *对于函子 $F, G : A \to B$ 和 $H, K : B \to C$ 以及自然变换 $\gamma : F \to G$ 和 $\delta : H \to K$，我们有*

$$(\delta G)(H\gamma) = (K\gamma)(\delta F)$$

*证明.* 分量检查足够了：在 $a : A$ 处我们有

$$\begin{aligned}
((\delta G)(H\gamma))_a &\equiv (\delta G)_a (H\gamma)_a \\
&\equiv \delta_{Ga} \circ H(\gamma_a) \\
&= K(\gamma_a) \circ \delta_{Fa} && \text{(通过 } \delta \text{ 的自然性)} \\
&\equiv (K\gamma)_a \circ (\delta F)_a \\
&\equiv ((K\gamma)(\delta F))_a
\end{aligned}$$

□

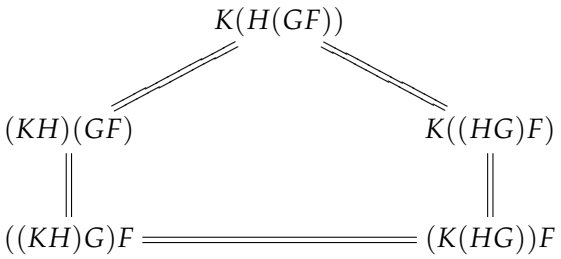
在经典的范畴论中，通常定义 $\gamma : F \to G$ 和 $\delta : H \to K$ 的“水平合成 (horizontal composite)”为 $(\delta G)(H\gamma)$ 和 $(\gamma\delta)(\delta F)$ 的公共值。我们将避免这样做，因为虽然它们相等，但这两个变换并不是*定义上*相等的。这也带来了这样的结果，即我们可以明确地使用符号 $\circ$（或并列）进行所有种类的合成：只有一种方式可以合成两个自然变换（而不是在任一侧将自然变换与函子合成）。

Lemma 9.2.9. *函子合成是结合的：* $H(GF) = (HG)F$。

*证明.* 由于函数的合成是结合的，这在作用于对象和同态时立刻就可以得到。由于同源集是集合，其余的数据是自动的。 □

Lemma 9.2.9 中的等式同样不是定义上的。（函数合成是定义上的结合，但进入函子的公理也必须合成，这打破了定义上的结合性。）因此，我们还需要知道关于结合性的*一致性* (coherence)。

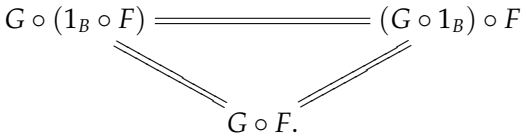
Lemma 9.2.10. Lemma 9.2.9 是一致的，即以下五边形 (pentagon) 的等式是可交换的：




证明. 正如在 Lemma 9.2.9 中一样，这在作用于对象时是显而易见的，其余的是自动的。 □

我们将从此滥用符号，通过书写 $H \circ G \circ F$ 或 $HGF$ 来表示 $H(GF)$ 或 $(HG)F$，在必要时沿 Lemma 9.2.9 运输。对于单位我们也有类似的一致性结果。

Lemma 9.2.11. 对于函子 $F : A \to B$，我们有等式 $(1_B \circ F) = F$ 和 $(F \circ 1_A) = F$，并且给定 $G : B \to C$ 时，以下三角形的等式是可交换的：




请参阅 Exercises 9.4 and 9.5 以进一步发展这些思想。

## 9.3 伴随 (Adjunctions)

伴随函子的定义是直接的；主要的有趣方面来源于证明相关性。

Definition 9.3.1. 一个函子 $F : A \to B$ 是一个**左伴随** (left adjoint)，如果存在

- 一个函子 $G : B \to A$。
- 一个自然变换 $\eta : 1_A \to GF$ (称为**单元** (unit))。
- 一个自然变换 $\epsilon : FG \to 1_B$ (称为**余单元** (counit))。
- $(\epsilon F)(F\eta) = 1_F$。
- $(G\epsilon)(\eta G) = 1_G$。

最后两个等式称为**三角等式** (triangle identities) 或**锯齿等式** (zigzag identities)。我们将留给读者定义右伴随 (right adjoints) 的类似定义。

Lemma 9.3.2. 如果 $A$ 是一个范畴（但 $B$ 可能只是一个前范畴），那么类型“$F$ 是一个左伴随”是一个纯命题。

证明. 假设我们给定 $(G, \eta, \epsilon)$ 以及三角等式，并且也有 $(G', \eta', \epsilon')$。定义 $\gamma : G \to G'$ 为 $(G'\epsilon)(\eta' G)$，并且定义 $\delta : G' \to G$ 为 $(G\epsilon')(\eta G')$。那么

$$\begin{aligned}
\delta\gamma &= (G\epsilon')(\eta G')(G'\epsilon)(\eta' G) \\
&= (G\epsilon')(GFG'\epsilon)(\eta G'FG)(\eta' G) \\
&= (G\epsilon)(G\epsilon' FG)(GF\eta' G)(\eta G) \\
&= (G\epsilon)(\eta G) \\
&= 1_G
\end{aligned}$$

使用 Lemma 9.2.8 和三角等式。类似地，我们证明 $\gamma\delta = 1_{G'}$，因此 $\gamma$ 是一个自然同构 $G \cong G'$。通过 Theorem 9.2.5，我们有一个同一性 $G = G'$。
现在我们需要知道当 $\eta$ 和 $\epsilon$ 沿此同一性运输时，它们变得等于 $\eta'$ 和 $\epsilon'$。通过 Lemma 9.1.9，此运输通过与 $\gamma$ 或 $\delta$ 适当地复合来给出。对于 $\eta$，这产生了

$$(G'\epsilon F)(\eta' GF)\eta = (G'\epsilon F)(G'F\eta)\eta' = \eta'$$

使用 Lemma 9.2.8 和三角等式。对于 $\epsilon$ 的情况是类似的。最后，三角等式自动正确地运输，因为同源集是集合。 □

在 §9.5 中我们将给出 Lemma 9.3.2 的另一个证明。

## 9.4 等价 (Equivalences)

在范畴论 (category theory) 中，通常将范畴的等价 (equivalence of categories) 定义为一个函子 $F : A \to B$，使得存在一个函子 $G : B \to A$ 以及自然同构 $FG \cong 1_B$ 和 $GF \cong 1_A$。然而，与作为伴随 (adjunction) 的性质不同，如果不进行截断，这将不是一个纯命题，原因与拟逆元 (quasi-inverses) 的类型行为不良相同（参见 §4.1）。如同在 §4.2 中那样，我们可以通过使用通常的伴随等价 (adjoint equivalence) 的概念来避免这个问题。

Definition 9.4.1. 一个函子 $F : A \to B$ 是一个**范畴的等价** (equivalence of (pre)categories)，如果它是一个左伴随，并且其 $\eta$ 和 $\epsilon$ 是同构。我们记 $A \simeq B$ 为从 $A$ 到 $B$ 的范畴等价的类型。

根据 Lemmas 9.1.3 and 9.3.2，如果 $A$ 是一个范畴，那么“$F$ 是前范畴的等价”这个类型是一个纯命题。

Lemma 9.4.2. 如果对于 $F : A \to B$ 存在 $G : B \to A$ 和同构 $GF \cong 1_A$ 和 $FG \cong 1_B$，那么 $F$ 是前范畴的等价。

证明. 类似于 Theorem 4.2.3 中关于类型等价的证明。 □

Definition 9.4.3. 我们说一个函子 $F : A \to B$ 是**忠实的** (faithful)，如果对于所有 $a, b : A$，函数

$$F_{a,b} : \hom_A(a,b) \to \hom_B(Fa, Fb)$$

是单射 (injective) 的，并且**满的** (full)，如果对于所有 $a, b : A$ 这个函数是满射 (surjective) 的。如果它同时是两者（因此每个 $F_{a,b}$ 是一个等价），我们说 $F$ 是**完全忠实的** (fully faithful)。

Definition 9.4.4. 我们说一个函子 $F : A \to B$ 是**分裂本质满射的** (split essentially surjective)，如果对于所有 $b : B$，存在一个 $a : A$，使得 $Fa \cong b$。

Lemma 9.4.5. 对于任意前范畴 $A$ 和 $B$ 以及函子 $F : A \to B$，以下类型是等价的。

(i) $F$ 是前范畴的等价。
(ii) $F$ 是完全忠实并且分裂本质满射的。

证明. 假设 $F$ 是前范畴的等价，并指定了 $G, \eta, \epsilon$。那么我们有函数

$$\begin{aligned}\hom_B(Fa, Fb) &\to \hom_A(a,b),\\ g &\mapsto {\eta_b}^{-1} \circ G(g) \circ \eta_a\end{aligned}$$

对于 $f : \hom_A(a,b)$，我们有

$${\eta_b}^{-1} \circ G(F(f)) \circ \eta_a = {\eta_b}^{-1} \circ \eta_b \circ f = f$$

而对于 $g : \hom_B(Fa, Fb)$，我们有

$$\begin{aligned} F({\eta_b}^{-1} \circ G(g) \circ \eta_a) &= F({\eta_b}^{-1}) \circ F(G(g)) \circ F(\eta_a) \\ &= \epsilon_{Fb} \circ F(G(g)) \circ F(\eta_a) \\ &= g \circ \epsilon_{Fa} \circ F(\eta_a) \\ &= g \end{aligned}$$

使用 $\epsilon$ 的自然性 (naturality)，以及三角等式两次。因此，$F_{a,b}$ 是一个等价，因此 $F$ 是完全忠实的。最后，对于任意 $b : B$，我们有 $Gb : A$ 和 $\epsilon_b : FGb \cong b$。
另一方面，假设 $F$ 是完全忠实并且分裂本质满射的。定义 $G_0 : B_0 \to A_0$ 通过将 $b : B$ 发送到指定的本质分裂给出的 $a : A$，并写 $\epsilon_b$ 为同样指定的同构 $FGb \cong b$。
现在对于任意 $g : \hom_B(b, b')$，定义 $G(g) : \hom_A(Gb, Gb')$ 为唯一的态射，使得 $F(G(g)) = (\epsilon_{b'})^{-1} \circ g \circ \epsilon_b$（由于 $F$ 是完全忠实的，该态射存在）。最后，对于 $a : A$ 定义 $\eta_a : \hom_A(a, GFa)$ 为唯一的态射，使得 $F\eta_a = {\epsilon_{Fa}}^{-1}$。很容易验证 $G$ 是一个函子，并且 $(G, \eta, \epsilon)$ 展示了 $F$ 作为一个前范畴的等价。
现在考虑复合 (i)→(ii)→(i)。我们显然恢复了相同的函数 $G_0 : B_0 \to A_0$。对于同态集上的 $G$ 的作用，我们必须证明对于 $g : \hom_B(b, b')$，$G(g)$ 是（必然唯一的）态射，使得 $F(G(g)) = (\epsilon_{b'})^{-1} \circ g \circ \epsilon_b$。但这个方程通过假设的 $\epsilon$ 的自然性成立。我们同样显然恢复了 $\epsilon$，而 $\eta$ 是通过 $F\eta_a = {\epsilon_{Fa}}^{-1}$（这是伴随等价结构中假定的三角等式之一）唯一确定的。因此，这个复合等于恒等式。
最后，考虑另一个复合 (ii)→(i)→(ii)。由于完全忠实是一个纯命题，足以观察到我们为每个 $b : B$ 恢复了相同的 $a : A$ 和同构 $Fa \cong b$。但这是显然的，因为我们使用了这个函数和同构来定义 $G_0$ 和 $\epsilon$ 在 (i) 中，这反过来又正是我们用来再次恢复 (ii) 的内容。因此，两个方向上的复合都等于恒等式，因此我们有一个等价 (i) $\simeq$ (ii)。 □

然而，如果 $A$ 不是一个范畴，那么 Lemma 9.4.5 中的任意类型可能不一定是一个纯命题。这提示我们也考虑以下概念。

Definition 9.4.6. 一个函子 $F : A \to B$ 是**本质满射** (essentially surjective) 的，如果对于所有 $b : B$，仅仅存在一个 $a : A$ 使得 $Fa \cong b$。我们说 $F$ 是一个**弱等价** (weak equivalence)，如果它是完全忠实并且本质满射的。

作为一个弱等价始终是一个纯命题。然而，对于范畴之间的函子，等价与弱等价没有区别。

Lemma 9.4.7. *如果 $F : A \to B$ 是完全忠实的并且 $A$ 是一个范畴，那么对于任何 $b : B$ 类型 $\sum_{(a:A)}(Fa \cong b)$ 是一个纯命题。因此，一个范畴之间的函子如果且仅如果它是一个弱等价，则是一个等价。*

*证明.* 假设给定 $(a, f)$ 和 $(a', f')$ 在 $\sum_{(a:A)}(Fa \cong b)$ 中。然后 ${f'}^{-1} \circ f$ 是一个同构 $Fa \cong Fa'$。由于 $F$ 是完全忠实的，我们有 $g : a \cong a'$，其中 $Fg = {f'}^{-1} \circ f$。由于 $A$ 是一个范畴，我们有 $p : a = a'$，其中 $\mathsf{idtoiso}(p) = g$。现在 $Fg = {f'}^{-1} \circ f$ 意味着 $((F_0)(p))_*(f) = f'$，因此（根据依赖对类型中相等性的特征）$(a, f) = (a', f')$。
因此，对于域为范畴的完全忠实函子，本质满射等价于分裂本质满射，因此，弱等价等价于等价。 □

这是我们范畴论相对于基于集合的方法的一个重要优势。在一个纯集合论定义的范畴中，“每个完全忠实并且本质满射的函子都是一个范畴等价”这一命题等价于选择公理 AC 。在这里，我们免费获得了它，作为选择唯一性原则 (principle of unique choice) 的范畴论版本 (§3.9)。（实际上，这个性质刻画了前范畴中的范畴；参见 §9.9。）
另一方面，以下对范畴等价的刻画可能更有用。

Definition 9.4.8. 一个函子 $F : A \to B$ 是一个**范畴的同构** (isomorphism of (pre)categories)，如果 $F$ 是完全忠实的并且 $F_0 : A_0 \to B_0$ 是一个类型等价。

这个定义是我们的一般规则的一个例外（参见 §2.4），即只在集合和类集合样的对象上使用“同构”一词。然而，在这里它确实带有适当的含义，因为对于一般的前范畴，同构比等价更强。
注意，作为前范畴的同构始终是一个纯命题。我们记 $A \cong B$ 为从 $A$ 到 $B$ 的前范畴同构的类型。

Lemma 9.4.9. 对于前范畴 $A$ 和 $B$ 以及函子 $F : A \to B$，以下命题是等价的。

(i) $F$ 是前范畴的同构。

(ii) 存在 $G : B \to A$ 和 $\eta : 1_A = GF$ 以及 $\epsilon : FG = 1_B$，使得

$$\mathsf{ap}_{(\lambda H.FH)}(\eta) = \mathsf{ap}_{(\lambda K.KF)}(\epsilon^{-1}) \tag{9.4.10}$$

(iii) 仅仅存在 $G : B \to A$ 和 $\eta : 1_A = GF$ 以及 $\epsilon : FG = 1_B$。

注意，如果 $B_0$ 不是一个 1-类型，那么 (9.4.10) 可能不是一个纯命题。

证明. 首先注意，由于同源集是集合，函子之间相等性的相等由其对象部分唯一确定。因此，根据函数外延性 (function extensionality)， (9.4.10) 等价于

$$(F_0)(\eta_0)_a = (\epsilon_0)^{-1}{}_{F_0 a} \tag{9.4.11}$$

对于所有 $a : A_0$。注意这正是 $G_0$，$\eta_0$ 和 $\epsilon_0$ 作为证明 $F_0$ 是一个半伴随等价 (half adjoint equivalence) 的三角等式。

现在假设 (i)。令 $G_0 : B_0 \to A_0$ 为 $F_0$ 的逆元，具有 $\eta_0 : \mathsf{id}_{A_0} = G_0F_0$ 和 $\epsilon_0 : F_0G_0 = \mathsf{id}_{B_0}$ 满足三角等式，这正是 (9.4.11)。现在定义 $G_{b,b'} : \hom_B(b,b') \to \hom_A(G_0b, G_0b')$ 为

$$G_{b,b'}(g) :\equiv (F_{G_0b,G_0b'})^{-1}\Big(\mathsf{idtoiso}((\epsilon_0)^{-1}{}_{b'}) \circ g \circ \mathsf{idtoiso}((\epsilon_0)_b)\Big)$$

(使用假设 $F$ 是完全忠实的)。由于 idtoiso 取逆元到逆元，且复合到组合，并且 $F$ 是一个函子，因此 $G$ 是一个函子。

按定义，我们有 $(GF)_0 \equiv G_0F_0$，通过 $\eta_0$ 等于 $\mathsf{id}_{A_0}$。为了获得 $1_A = GF$，我们需要证明当沿着 $\eta_0$ 运输时，同源集的恒等函数变得等于复合 $G_{Fa,Fa'} \circ F_{a,a'}$。换句话说，对于任意 $f : \hom_A(a,a')$，我们必须有

$$\mathsf{idtoiso}((\eta_0)_{a'}) \circ f \circ \mathsf{idtoiso}((\eta_0)^{-1}{}_a) = (F_{GFa,GFa'})^{-1}\Big(\mathsf{idtoiso}((\epsilon_0)^{-1}{}_{Fa'}) \circ F_{a,a'}(f) \circ \mathsf{idtoiso}((\epsilon_0)_{Fa})\Big)$$

但这等价于

$$(F_{GFa,GFa'})\Big(\mathsf{idtoiso}((\eta_0)_{a'}) \circ f \circ \mathsf{idtoiso}((\eta_0)^{-1}{}_a)\Big) = \mathsf{idtoiso}((\epsilon_0)^{-1}{}_{Fa'}) \circ F_{a,a'}(f) \circ \mathsf{idtoiso}((\epsilon_0)_{Fa})$$

这跟随于 $F$ 的函子性，$F$ 保持 idtoiso 的事实，以及 (9.4.11)。因此我们有 $\eta : 1_A = GF$。

另一方面，我们有 $(FG)_0 \equiv F_0G_0$，通过 $\epsilon_0$ 等于 $\mathsf{id}_{B_0}$。为了获得 $FG = 1_B$，我们需要证明当沿着 $\epsilon_0$ 运输时，同源集的恒等函数变得等于复合 $F_{Gb,Gb'} \circ G_{b,b'}$。也就是说，对于任意 $g : \hom_B(b,b')$，我们必须有

$$F_{Gb,Gb'}\Big((F_{Gb,Gb'})^{-1}\Big(\mathsf{idtoiso}((\epsilon_0)^{-1}{}_{b'}) \circ g \circ \mathsf{idtoiso}((\epsilon_0)_b)\Big)\Big) = \mathsf{idtoiso}((\epsilon_0{}^{-1})_{b'}) \circ g \circ \mathsf{idtoiso}((\epsilon_0)_b)$$

但这仅仅是 $(F_{Gb,Gb'})^{-1}$ 是 $F_{Gb,Gb'}$ 的逆元的事实。我们已经提到 (9.4.10) 等价于 (9.4.11)，所以 (ii) 成立。

反过来，假设 (ii)；那么 $G$ 的对象部分，$\eta$ 和 $\epsilon$ 以及 (9.4.11) 的对象部分表明 $F_0$ 是一个类型等价。而对于 $a, a' : A_0$，我们定义 $\overline{G}_{a,a'} : \hom_B(Fa, Fa') \to \hom_A(a,a')$ 为

$$\overline{G}_{a,a'}(g) :\equiv \mathsf{idtoiso}(\eta^{-1})_{a'} \circ G(g) \circ \mathsf{idtoiso}(\eta)_a \tag{9.4.12}$$

根据 $\mathsf{idtoiso}(\eta)$ 的自然性，对于任意 $f : \hom_A(a, a')$，我们有

$$\begin{aligned}\overline{G}_{a,a'}(F_{a,a'}(f)) &= \mathsf{idtoiso}(\eta^{-1})_{a'} \circ G(F(f)) \circ \mathsf{idtoiso}(\eta)_a \\ &= \mathsf{idtoiso}(\eta^{-1})_{a'} \circ \mathsf{idtoiso}(\eta)_{a'} \circ f \\ &= f\end{aligned}$$

另一方面，对于 $g : \hom_B(Fa, Fa')$，我们有

$$\begin{aligned}F_{a,a'}(\overline{G}_{a,a'}(g)) &= F(\mathsf{idtoiso}(\eta^{-1})_{a'}) \circ F(G(g)) \circ F(\mathsf{idtoiso}(\eta)_a) \\ &= \mathsf{idtoiso}(\epsilon)_{Fa'} \circ F(G(g)) \circ \mathsf{idtoiso}(\epsilon^{-1})_{Fa} \\ &= \mathsf{idtoiso}(\epsilon)_{Fa'} \circ \mathsf{idtoiso}(\epsilon^{-1})_{Fa'} \circ g \\ &= g\end{aligned}$$

（这里需要一些关于 idtoiso 和 whiskering 兼容性的引理，我们留给读者去陈述和证明。）因此，$F_{a,a'}$ 是一个等价，所以 $F$ 是完全忠实的；即 (i) 成立。

现在复合 (i)→(ii)→(i) 等于恒等式，因为 (i) 是一个纯命题。另一方面，跟踪以上构造，我们看到复合 (ii)→(i)→(ii) 本质上保留了 $G_0$，$\eta_0$，$\epsilon_0$ 的对象部分以及 (9.4.10) 的对象部分。而在后面三种情况下，对象部分就是全部，因为同源集是集合。

因此，足以证明我们恢复了 $G$ 在同源集上的作用。换句话说，我们必须证明如果 $g : \hom_B(b, b')$，那么

$$G_{b,b'}(g) = \overline{G}_{G_0 b, G_0 b'}\left(\mathsf{idtoiso}((\epsilon_0)^{-1}{}_{b'}) \circ g \circ \mathsf{idtoiso}((\epsilon_0)_b)\right)$$

其中 $\overline{G}$ 定义为 (9.4.12)。然而，这跟随于 $G$ 的函子性和另一三角等式，我们在 Chapter 4 中看到这等价于 (9.4.11)。

现在，由于 (i) 是一个纯命题，(ii) 也是如此，所以足以证明它们逻辑上等价于 (iii)。当然，(ii)→(iii)，所以让我们假设 (iii)。由于 (i) 是一个纯命题，我们可以假设给定 $G$，$\eta$ 和 $\epsilon$。然后 $G_0$ 连同 $\eta$ 和 $\epsilon$ 表明 $F_0$ 是一个等价。此外，我们还有自然同构 $\mathsf{idtoiso}(\eta) : 1_A \cong GF$ 和 $\mathsf{idtoiso}(\epsilon) : FG \cong 1_B$，因此根据 Lemma 9.4.2，$F$ 是一个前范畴的等价，特别是完全忠实的。 □

从 Lemma 9.4.9(ii) 和函子范畴中的 idtoiso，我们可以立即得出任何前范畴的同构都是等价的。对于前范畴，反过来可能会失败。

Example 9.4.13. 设 $X$ 是一个类型且 $x_0 : X$ 是一个元素，并令 $X_{\mathrm{ch}}$ 表示 $X$ 上的混乱 (chaotic)或不分离 (indiscrete)前范畴。按照定义，我们有 $(X_{\mathrm{ch}})_0 :\equiv X$，并且对于所有 $x, x'$，$\hom_{X_{\mathrm{ch}}}(x, x') :\equiv \mathbf{1}$。然后唯一的函子 $X_{\mathrm{ch}} \to \mathbf{1}$ 是一个前范畴的等价，但除非 $X$ 是收缩的 (contractible)，否则它不是同构。这个例子也表明，一个前范畴可以等价于一个范畴而不自身是一个范畴。当然，如果一个前范畴是同构的到一个范畴，那么它自身必须是一个范畴。

然而，对于范畴，这两个概念是一致的。

Lemma 9.4.14. 对于范畴 $A$ 和 $B$，一个函子 $F : A \to B$ 如果且仅如果它是范畴的同构，则是范畴的等价。

证明. 由于两者都是纯命题，所以足以证明它们逻辑上等价。因此，首先假设 $F$ 是范畴的等价，并给定了 $(G, \eta, \epsilon)$。我们已经看到 $F$ 是完全忠实的。根据 Theorem 9.2.5，自然同构 $\eta$ 和 $\epsilon$ 产生了恒等式 $1_A = GF$ 和 $FG = 1_B$，因此特别是 $\mathsf{id}_A = G_0 \circ F_0$ 和 $F_0 \circ G_0 = \mathsf{id}_B$ 的恒等式。因此，$F_0$ 是一个类型等价。

反过来，假设 $F$ 是完全忠实的并且 $F_0$ 是一个类型等价，假设其逆元是 $G_0$。那么对于每个 $b : B$，我们有 $G_0 b : A$ 并且有一个恒等式 $FGb = b$，因此有一个同构 $FGb \cong b$。因此，根据 Lemma 9.4.5，$F$ 是一个范畴的等价。 □

当然，还有第三个相似的概念用于（前）范畴：相等性。然而，一致性公理 (univalence axiom) 表明它与同构一致。

Lemma 9.4.15. 如果 $A$ 和 $B$ 是前范畴，那么函数

$$(A = B) \to (A \cong B)$$

(通过从恒等函子归纳定义) 是一个类型等价。

证明. 正如通常对于依赖和类型，给出一个 $A = B$ 的元素等价于给出

- 一个恒等式 $P_0 : A_0 = B_0$，
- 对于每个 $a, b : A_0$，一个恒等式

$$P_{a,b} : \hom_A(a,b) = \hom_B(P_{0*}(a), P_{0*}(b)),$$

- 恒等式 $(P_{a,a})_*(1_a) = 1_{P_{0*}(a)}$ 和 $(P_{a,c})_*(gf) = (P_{b,c})_*(g) \circ (P_{a,b})_*(f)$

(再次使用同源集的身份类型是纯命题的事实。) 然而，根据一致性，这等价于给出

- 一个类型的等价 $F_0 : A_0 \simeq B_0$，
- 对于每个 $a, b : A_0$，一个类型的等价

$$F_{a,b} : \hom_A(a,b) \simeq \hom_B(F_0(a), F_0(b)),$$

- 和恒等式 $F_{a,a}(1_a) = 1_{F_0(a)}$ 和 $F_{a,c}(gf) = F_{b,c}(g) \circ F_{a,b}(f)$。

但这正是一个函子 $F : A \to B$，它是一个范畴的同构。并且通过对恒等式的归纳，这个等价 $(A = B) \simeq (A \cong B)$ 等于通过归纳获得的那个。 □

因此，对于范畴，相等性也等同于等价。我们可以将此解释为，范畴、函子和自然变换不仅形成了一个前 2-范畴 (pre-2-category)，还形成了一个 2-范畴（参见 Exercise 9.4）。

Theorem 9.4.16. 如果 $A$ 和 $B$ 是范畴，那么函数

$$(A = B) \to (A \simeq B)$$

(通过从恒等函子归纳定义) 是一个类型等价。

证明. 根据 Lemmas 9.4.14 and 9.4.15。 □

因此，范畴的类型是一个 2-类型。因为 $A \simeq B$ 是函子类型的一个子类型，而函子是范畴的对象，所以它是一个 1-类型；因此 $A = B$ 的恒等类型也是 1-类型。

## 9.5 Yoneda **引理** (The Yoneda Lemma)

回顾一下，我们有一个范畴 $\mathcal{S}et$ ，其对象是集合，其态射是函数。现在我们展示每个前范畴都有一个取值为 $\mathcal{S}et$ 的 hom-函子。首先我们需要定义（前）范畴的对偶与积。

Definition 9.5.1. 对于一个前范畴 $A$, 其 **对偶** (opposite) (前) 范畴的对偶前范畴的对偶范畴的对偶$A^{\mathrm{op}}$ 是一个具有相同对象类型的前范畴，其中 $\hom_{A^{\mathrm{op}}}(a,b) :\equiv \hom_A(b,a)$，而其恒等元和复合则继承自 $A$。

Definition 9.5.2. 对于前范畴 $A$ 和 $B$，它们的 **积** (product) 前范畴的积范畴的积(前) 范畴的积$A \times B$ 是一个前范畴，其 $(A \times B)_0 :\equiv A_0 \times B_0$ 且

$$\hom_{A\times B}((a,b),(a',b')) :\equiv \hom_A(a,a') \times \hom_B(b,b')$$

恒等元定义为 $1_{(a,b)} :\equiv (1_a, 1_b)$，复合定义为 $(g,g')(f,f') :\equiv ((gf),(g'f'))$

Lemma 9.5.3. *对于前范畴 $A, B, C$，下列类型是等价的。*

(i) *函子 $A \times B \to C$。*
(ii) *函子 $A \to C^B$。*

*证明.* 给定 $F : A \times B \to C$，对于任意 $a : A$，我们显然有一个函子 $F_a : B \to C$。这给出了一个函数 $A_0 \to (C^B)_0$。接下来，对于任意 $f : \hom_A(a, a')$，我们对于任意 $b : B$ 都有态射 $F_{(a,b),(a',b)}(f, 1_b) : F_a(b) \to F_{a'}(b)$。这些是自然变换 $F_a \to F_{a'}$ 的分量。在 $a$ 上的函子性易于验证，因此我们有一个函子 $\hat{F} : A \to C^B$。
反过来，假设给定 $G : A \to C^B$。那么对于任意 $a : A$ 和 $b : B$，我们有对象 $G(a)(b) : C$，给出一个函数 $A_0 \times B_0 \to C_0$。对于 $f : \hom_A(a, a')$ 和 $g : \hom_B(b, b')$，我们有态射

$$G(a')_{b,b'}(g) \circ G_{a,a'}(f)_b = G_{a,a'}(f)_{b'} \circ G(a)_{b,b'}(g)$$

在 $\hom_C(G(a)(b), G(a')(b'))$ 中。函子性再次容易验证，因此我们有一个函子 $\check{G} : A \times B \to C$。
最后，这些操作是互逆的也很明显。 □

现在对于任何前范畴 $A$，我们有一个 hom-函子 hom-函子

$$\hom_A : A^{\mathrm{op}} \times A \to \mathcal{S}et$$

它将一个对 $(a, b) : (A^{\mathrm{op}})_0 \times A_0 \equiv A_0 \times A_0$ 映射为集合 $\hom_A(a, b)$。对于一个态射 $(f, f') : \hom_{A^{\mathrm{op}} \times A}((a, b), (a', b'))$，根据定义我们有 $f : \hom_A(a', a)$ 和 $f' : \hom_A(b, b')$，因此我们可以定义

$$\begin{aligned}(\hom_A)_{(a,b),(a',b')}(f, f') &:\equiv (g \mapsto (f' g f)) \\ &: \hom_A(a, b) \to \hom_A(a', b')\end{aligned}$$

函子性易于验证。
因此根据 Lemma 9.5.3，我们有一个诱导的函子 $\mathbf{y} : A \to \mathcal{S}et^{A^{\mathrm{op}}}$，我们称之为 Yoneda **嵌入** (Yoneda embedding)。Yoneda 嵌入嵌入!Yoneda

Theorem 9.5.4 (Yoneda 引理). Yoneda *引理对于任何前范畴 $A$，任意 $a : A$ 和任意函子 $F : \mathcal{S}et^{A^{\mathrm{op}}}$，我们有一个同构*

$$\hom_{\mathcal{S}et^{A^{\mathrm{op}}}}(\mathbf{y}a, F) \cong Fa \tag{9.5.5}$$

*此外，这在 $a$ 和 $F$ 中都是自然的。*

*证明.* 给定一个自然变换 $\alpha : \mathbf{y}a \to F$，我们可以考虑分量 $\alpha_a : \mathbf{y}a(a) \to Fa$。由于 $\mathbf{y}a(a) \equiv \hom_A(a, a)$，我们有 $1_a : \mathbf{y}a(a)$，因此 $\alpha_a(1_a) : Fa$。这给出了 (9.5.5) 中从左到右的一个函数 $(\alpha \mapsto \alpha_a(1_a))$。
反过来，给定 $x : Fa$，我们通过定义 $\alpha : \mathbf{y}a \to F$

$$\alpha_{a'}(f) :\equiv F_{a,a'}(f)(x)$$

自然性易于验证，因此这给出了 (9.5.5) 中从右到左的一个函数。
为了证明这些是互逆的，首先假设给定 $x : Fa$。那么按照上面的方法定义的 $\alpha$ 满足 $\alpha_a(1_a) = F_{a,a}(1_a)(x) = 1_{Fa}(x) = x$。另一方面，如果假设给定 $\alpha : \mathbf{y}a \to F$ 并按照上面的方法定义 $x$，那么对于任意 $f : \hom_A(a', a)$ 我们有

$$\begin{aligned}\alpha_{a'}(f) &= \alpha_{a'}(\mathbf{y}a_{a,a'}(f)(1_a)) \\ &= (\alpha_{a'} \circ \mathbf{y}a_{a,a'}(f))(1_a) \\ &= (F_{a,a'}(f) \circ \alpha_a)(1_a) \\ &= F_{a,a'}(f)(\alpha_a(1_a)) \\ &= F_{a,a'}(f)(x)\end{aligned}$$

因此，两个复合都等于恒等式。我们将自然性的证明留给读者。 □

Corollary 9.5.6. Yoneda 嵌入 $\mathbf{y} : A \to \mathcal{S}et^{A^{\mathrm{op}}}$ 是完全忠实的。

证明. 根据 Theorem 9.5.4，我们有

$$\hom_{\mathcal{S}et^{A^{\mathrm{op}}}}(\mathbf{y}a, \mathbf{y}b) \cong \mathbf{y}b(a) \equiv \hom_A(a, b)$$

验证这个同构实际上是 $\mathbf{y}$ 在 hom 集上的作用是容易的。 □

Corollary 9.5.7. 如果 $A$ 是一个范畴，那么 $\mathbf{y}_0 : A_0 \to (\mathcal{S}et^{A^{\mathrm{op}}})_0$ 是一个嵌入。特别地，如果 $\mathbf{y}a = \mathbf{y}b$，那么 $a = b$。

证明. 根据 Corollary 9.5.6，$\mathbf{y}$ 在同构集上诱导了一个同构。由于 $A$ 和 $\mathcal{S}et^{A^{\mathrm{op}}}$ 是范畴且 $\mathbf{y}$ 是一个函子，这等价于在身份类型上的同构，这就是嵌入的定义。 □

Definition 9.5.8. 一个函子 $F : \mathcal{S}et^{A^{\mathrm{op}}}$ 被称为 **可表示的** (representable) 可表示函子可表示函子如果存在 $a : A$ 并且有一个同构 $\mathbf{y}a \cong F$。

Theorem 9.5.9. 如果 $A$ 是一个范畴，那么“$F$ 是可表示的”类型是一个纯命题。

证明. 根据定义“$F$ 是可表示的”恰好是 $\mathbf{y}_0$ 在 $F$ 上的纤维。由于 Corollary 9.5.7 中的 $\mathbf{y}_0$ 是一个嵌入，因此这个纤维是一个纯命题。 □

特别地，在一个范畴中，同一个函子的任何两个表示都是相等的。我们可以利用这一点给出 Lemma 9.3.2 的另一种证明。首先我们给出一种用可表示性描述伴随关系的特征。

Lemma 9.5.10. 对于任意前范畴 $A$ 和 $B$ 以及一个函子 $F : A \to B$，下列类型是等价的。

(i) $F$ 是一个左伴随 (left adjoint)伴随函子。
(ii) 对于每个 $b : B$，从 $A^{\mathrm{op}}$ 到 $\mathcal{S}et$ 的函子 $(a \mapsto \hom_B(Fa, b))$ 是可表示的可表示函子。

证明. 类型 (ii) 的一个元素由一个函数 $G_0 : B_0 \to A_0$ 以及对于每个 $a : A$ 和 $b : B$ 的一个同构

$$\gamma_{a,b} : \hom_B(Fa, b) \cong \hom_A(a, G_0 b)$$

构成，满足 $\gamma_{a,b}(g \circ Ff) = \gamma_{a',b}(g) \circ f$ 对于 $f : \hom_A(a, a')$。
有了这个，对于 $a : A$，我们定义 $\eta_a :\equiv \gamma_{a,Fa}(1_{Fa})$，对于 $b : B$，我们定义 $\epsilon_b :\equiv (\gamma_{Gb,b})^{-1}(1_{Gb})$。现在对于 $g : \hom_B(b, b')$ 我们定义

$$G_{b,b'}(g) :\equiv \gamma_{Gb,b'}(g \circ \epsilon_b)$$

验证 $G$ 是一个函子且 $\eta$ 和 $\epsilon$ 是满足三角恒等式的自然变换的过程与经典情形完全相同，并且由于它们都是纯命题，我们不会关心它们的具体值。因此，我们有一个从 (ii) 到 (i) 的函数。
反过来，如果 $F$ 是一个左伴随，我们当然有 $G_0$ 指定的，我们可以将 $\gamma_{a,b}$ 取为复合

$$\hom_B(Fa, b) \xrightarrow{G_{Fa,b}} \hom_A(GFa, Gb) \xrightarrow{(-\circ\eta_a)} \hom_A(a, Gb)$$

由于 $\eta$ 是自然的，因此这显然是自然的，并且它有一个逆元给出

$$\hom_A(a, Gb) \xrightarrow{F_{a,Gb}} \hom_B(Fa, FGb) \xrightarrow{(\epsilon_b\circ-)} \hom_A(Fa, b)$$

(通过三角恒等式)。因此我们也有 (i) 到 (ii)。
对于复合 (ii) 到 (i) 再到 (ii)，显然函数 $G_0$ 是保持的，因此只需检查我们是否得到了 $\gamma$。但是新定义的 $\gamma$ 被定义为将 $f : \hom_B(Fa, b)$ 映射为

$$\begin{aligned} G(f) \circ \eta_a &\equiv \gamma_{GFa,b}(f \circ \epsilon_{Fa}) \circ \eta_a \\ &= \gamma_{GFa,b}(f \circ \epsilon_{Fa} \circ F\eta_a) \\ &= \gamma_{GFa,b}(f) \end{aligned}$$

所以它与旧的 $\gamma$ 一致。
最后，对于 (i) 到 (ii) 再到 (i)，我们肯定会得到对象上的函子 $G$。新的 $G_{b,b'} : \hom_B(b,b') \to \hom_A(Gb,Gb')$ 被定义为将 $g$ 映射为

$$\begin{aligned}\gamma_{Gb,b'}(g \circ \epsilon_b) &\equiv G(g \circ \epsilon_b) \circ \eta_{Gb} \\ &= G(g) \circ G\epsilon_b \circ \eta_{Gb} \\ &= G(g)\end{aligned}$$

所以它与旧的 $G$ 一致。新的 $\eta_a$ 被定义为 $\gamma_{a,Fa}(1_{Fa}) \equiv G(1_{Fa}) \circ \eta_a$，因此它等于旧的 $\eta_a$。最后，新的 $\epsilon_b$ 被定义为 $(\gamma_{Gb,b})^{-1}(1_{Gb}) \equiv \epsilon_b \circ F(1_{Gb})$，这也等于旧的 $\epsilon_b$。 □

Corollary 9.5.11. [Lemma 9.3.2] 如果 $A$ 是一个范畴且 $F : A \to B$，那么类型“$F$ 是一个左伴随”是一个纯命题。

证明. 根据 Theorem 9.5.9，如果 $A$ 是一个范畴，那么 Lemma 9.5.10 中的类型 (ii) 是一个纯命题。 □

## 9.6 严格范畴 (Strict categories)

Definition 9.6.1. 一个**严格范畴** (strict category) 是一个其对象类型为集合的预范畴。

根据数学中“红鲱鱼原则 (red herring principle)”，严格范畴不一定是一个范畴。实际上，当且仅当一个范畴是消瘦范畴 (gaunt category) 时，它才是一个严格范畴 (Example 9.1.15)。大多数情况下，范畴论研究的是范畴，而非严格范畴，但有时人们会考虑严格范畴。其主要优点在于，严格范畴比等价关系有更严格的“相同”概念，即同构（或者等价地，通过 Lemma 9.4.15，即等同关系）。
以下是严格范畴的一个来源。
Example 9.6.2. 设 $A$ 是一个预范畴，$x : A$ 是一个对象。则存在一个预范畴 $\mathsf{mono}(A,x)$，如下定义：

- 其对象由一个对象 $y : A$ 和一个从 $y$ 到 $x$ 的单态射 $m : \hom_A(y,x)$ 组成。(通常，$m : \hom_A(y,x)$ 是一个**单态射** (monomorphism)（或称为**单态** (monic)）当且仅当 $(m \circ f = m \circ g) \Rightarrow (f = g)$。)
- 从 $(y,m)$ 到 $(z,n)$ 的态射是 $A$ 中从 $y$ 到 $z$ 的任意态射（不必尊重 $m$ 和 $n$）。

在 $\mathsf{mono}(A,x)$ 中对象的等同 $(y,m) = (z,n)$ 包含对象的等同 $p : y = z$ 和等同 $p_*(m) = n$，根据 Lemma 9.1.9，这等价于等同 $m = n \circ \mathsf{idtoiso}(p)$。由于态射集是集合，这种等同性的类型只是一个单纯命题。但是由于 $m$ 和 $n$ 是单态射，使得 $m = n \circ f$ 的态射 $f$ 的类型也是一个单纯命题。因此，如果 $A$ 是一个范畴，那么 $(y,m) = (z,n)$ 只是一个单纯命题，因此 $\mathsf{mono}(A,x)$ 是一个严格范畴。
这个例子可以对偶化，并以各种方式推广。以下是严格范畴的一个有趣应用。
Example 9.6.3. 设 $E/F$ 是有限伽罗瓦扩展 (Galois extension) 的域，$G$ 是其伽罗瓦群 (Galois group)。则存在一个严格范畴，其对象是中间域 $F \subseteq K \subseteq E$，其态射是固定 $F$ 不动的域同态（但不必与 $E$ 中的包含映射可交换）。另一个严格范畴的对象是子群 $H \subseteq G$，其态射是 $G$ 集合 $G/H \to G/K$ 的态射。伽罗瓦理论的基本定理 表明这两个预范畴是同构的（不仅仅是等价的）。

## 9.7 †-范畴 (†-categories)

还值得一提的是一种有用的预范畴，其对象类型不是集合，但它也不是一个范畴。

Definition 9.7.1. 一个 **†-预范畴** (†-precategory) 是一个预范畴 $A$，同时具备以下结构：

(i) 对于每个 $x,y : A$，存在一个函数 $(-)^\dagger : \hom_A(x,y) \to \hom_A(y,x)$。
(ii) 对于所有 $x : A$，我们有 $(1_x)^\dagger = 1_x$。
(iii) 对于所有 $f,g$，我们有 $(g \circ f)^\dagger = f^\dagger \circ g^\dagger$。

(iv) 对于所有 $f$，我们有 $(f^\dagger)^\dagger = f$。

Definition 9.7.2. 在 †-预范畴中的态射 $f : \mathsf{hom}_A(x,y)$ 是**酉态射** (unitary) 如果 $f^\dagger \circ f = 1_x$ 并且 $f \circ f^\dagger = 1_y$。

当然，每个酉态射都是同构的，并且酉态射的性质是一个单纯命题。因此，对于每个 $x, y : A$，我们有从 $x$ 到 $y$ 的酉同构的集合，我们记为 $(x \cong^\dagger y)$。

Lemma 9.7.3. *如果 $p : (x = y)$，则 $\mathsf{idtoiso}(p)$ 是酉态射。*

*证明.* 通过归纳法，我们可以假设 $p$ 是 $\mathsf{refl}_x$。但此时 $(1_x)^\dagger \circ 1_x = 1_x \circ 1_x = 1_x$，类似地。 □

Definition 9.7.4. 一个 †-**范畴** (†-category) 是一个 †-预范畴，使得对于所有 $x, y : A$，函数

$$(x = y) \to (x \cong^\dagger y)$$

从 Lemma 9.7.3 是一个等价。

Example 9.7.5. 从 Example 9.1.19 中的关系范畴 ($\mathcal{R}el$) 通过定义 $(R^\dagger)(y,x) :\equiv R(x,y)$ 变成一个 †-预范畴。关系范畴 ($\mathcal{R}el$) 是一个范畴的证明实际上表明每个同构都是酉的；因此 $\mathcal{R}el$ 也是一个 †-范畴。

Example 9.7.6. 任何群胚 (groupoid) 通过定义 $f^\dagger :\equiv f^{-1}$ 变成一个 †-范畴。

Example 9.7.7. 令 $\mathcal{H}ilb$ 是以下预范畴：

- 其对象是有限维度的带内积 $\langle -, - \rangle$ 的向量空间。
- 其态射是任意的线性映射。

根据标准的线性代数，任意有限维度内积空间之间的线性映射 $f : V \to W$ 都有一个唯一确定的伴随映射 $f^\dagger : W \to V$，其特征是 $\langle fv, w \rangle = \langle v, f^\dagger w \rangle$。通过这种方式，$\mathcal{H}ilb$ 变成了一个 †-预范畴。此外，当且仅当线性同构是**等距同构** (isometry) 时，它是酉态射，即 $\langle fv, fw \rangle = \langle v, w \rangle$。由此可以推断 $\mathcal{H}ilb$ 是一个 †-范畴，尽管它不是一个范畴（不是每个线性同构都是酉态射）。

关于 †-范畴的经典理论已经有了相当多的发展。人们很早就观察到，对于 †-范畴的对象，酉同构，而不是任意的同构，是“相同”的正确概念，这在范畴论学者中引起了一些困惑。同伦类型论通过将 †-范畴与严格范畴一样视为一种不同的预范畴，解决了这一问题。

## 9.8 **结构同一性原理** (The structure identity principle)

*结构同一性原理* (structure identity principle) 是一个非正式的原理，表示同构结构是相同的。我们旨在证明一个可以应用于广泛结构概念的一般抽象结果，其中结构可以是多类的，甚至是依赖类的，无穷的，甚至是高阶的。
最简单的单类结构由没有附加结构的类型组成。单值性公理以一种强烈的形式表达了这种结构概念的结构同一性原理：对于类型 $A, B$，从 $(A = B) \to (A \simeq B)$ 的典型函数是一个等价。
我们从一个预范畴 $X$ 开始。在我们应用于单类一阶结构时，$X$ 将是 $\mathcal{U}$-小集的范畴 ($\mathcal{S}et$

Definition 9.8.1. 一个**结构概念** (notion of structure) $(P, H)$ 在 $X$ 上包括以下内容。

(i) 一个类型族 $P : X_0 \to \mathcal{U}$。对于每个 $x : X_0$，$Px$ 的元素称为 $(P,H)$-**结构** (structures on $x$)。
(ii) 对于 $x, y : X_0$，$f : \mathsf{hom}_X(x,y)$ 和 $\alpha : Px$，$\beta : Py$，一个单纯命题

$$H_{\alpha\beta}(f)$$

如果 $H_{\alpha\beta}(f)$ 为真，我们称 $f$ 是从 $\alpha$ 到 $\beta$ 的 $(P,H)$-**同态** (homomorphism)。
(iii) 对于 $x : X_0$ 和 $\alpha : Px$，我们有 $H_{\alpha\alpha}(1_x)$。

(iv) 对于 $x, y, z : X_0$ 和 $\alpha : Px$， $\beta : Py$， $\gamma : Pz$，如果 $f : \hom_X(x, y)$ 和 $g : \hom_X(y, z)$，我们有

$$H_{\alpha\beta}(f) \to H_{\beta\gamma}(g) \to H_{\alpha\gamma}(g \circ f)$$

当 $(P, H)$ 是一个结构概念时，对于 $\alpha, \beta : Px$ 我们定义

$$(\alpha \leq_x \beta) :\equiv H_{\alpha\beta}(1_x)$$

根据 (iii) 和 (iv)，这是一个预序 (Example 9.1.14)，其对象类型为 $Px$。如果 $(P, H)$ 是一个**标准的结构概念** (standard notion of structure)，则对于 $X$ 上的所有 $x$，这个预序实际上是一个偏序。

请注意，对于标准的结构概念，每个类型 $Px$ 实际上必须是一个集合。我们现在定义，对于任何 $(P, H)$ 的结构概念，$(P, H)$-**结构的预范畴**，$A = \mathsf{Str}_{(P,H)}(X)$。

- $A$ 的对象类型是类型 $A_0 :\equiv \sum_{(x:X_0)} Px$。如果 $a \equiv (x, \alpha) : A_0$，我们可以写作 $|a| :\equiv x$。
- 对于 $(x, \alpha) : A_0$ 和 $(y, \beta) : A_0$，我们定义

$$\hom_A((x, \alpha), (y, \beta)) :\equiv \{\, f : x \to y \mid H_{\alpha\beta}(f) \,\}$$

合成和身份继承自 $X$；条件 (iii) 和 (iv) 确保这些提升到 $A$。

Theorem 9.8.2 (结构同一性原理 (Structure identity principle)). 如果 $X$ 是一个范畴，并且 $(P, H)$ 是 $X$ 上的一个标准结构概念，则预范畴 $\mathsf{Str}_{(P,H)}(X)$ 是一个范畴。

证明. 根据依赖对类型的等同性定义，给出等同性 $(x, \alpha) = (y, \beta)$ 包括以下内容：

- 一个等同性 $p : x = y$，以及
- 一个等同性 $p_*(\alpha) = \beta$。

由于 $P$ 是集合值的，后者只是一个单纯命题。另一方面，很容易看出 $\mathsf{Str}_{(P,H)}(X)$ 中的一个同构 $(x, \alpha) \cong (y, \beta)$ 包括以下内容：

- $X$ 中 $x \cong y$ 的一个同构 $f$，使得
- $H_{\alpha\beta}(f)$ 和 $H_{\beta\alpha}(f^{-1})$ 成立。

当然，第二项也是一个单纯命题。并且由于 $X$ 是一个范畴，函数 $(x = y) \to (x \cong y)$ 是一个等价。因此，只需证明对于任意 $p : x = y$ 和任意 $(\alpha : Px)$, $(\beta : Py)$，我们有 $p_*(\alpha) = \beta$ 当且仅当 $H_{\alpha\beta}(\mathsf{idtoiso}(p))$ 和 $H_{\beta\alpha}(\mathsf{idtoiso}(p)^{-1})$ 同时成立。
“仅当”方向仅是 $\mathsf{Str}_{(P,H)}(X)$ 范畴的函数 $\mathsf{idtoiso}$ 的存在性。对于“如果”方向，通过对 $p$ 进行归纳，我们可以假设 $y \equiv x$ 且 $p \equiv \mathsf{refl}_x$。然而，在这种情况下 $\mathsf{idtoiso}(p) \equiv 1_x$ 因此 $\mathsf{idtoiso}(p)^{-1} = 1_x$。因此，$\alpha \leq_x \beta$ 和 $\beta \leq_x \alpha$，这意味着 $\alpha = \beta$，因为 $(P, H)$ 是一个标准的结构概念。 □

作为一个例子，这种方法提供了一种表达 Theorem 9.2.5 证明的替代方法。

Example 9.8.3. 设 $A$ 是一个预范畴，$B$ 是一个范畴。存在一个预范畴 $B^{A_0}$，其对象是从 $A_0$ 到 $B_0$ 的函数，其态射集是从 $F_0 : A_0 \to B_0$ 到 $G_0 : A_0 \to B_0$ 的 $\gamma : \hom_{B^{A_0}}(F_0, G_0)$ 是一个同构，当且仅当每个分量 $\gamma_a$ 是同构，因此我们有 $(F_0 \cong G_0) \simeq \prod_{(a:A_0)}(F_0 a \cong G_0 a)$。此外，$B^{A_0}$ 的 $\mathsf{idtoiso} : (F_0 = G_0) \to (F_0 \cong G_0)$ 映射等同于以下复合函数

$$(F_0 = G_0) \longrightarrow \prod_{a:A_0} (F_0 a = G_0 a) \longrightarrow \prod_{a:A_0} (F_0 a \cong G_0 a) \longrightarrow (F_0 \cong G_0)$$

其中第一个映射是函数外延性给出的等价，第二个因为它是等价的依赖积（因为 $B$ 是一个范畴），第三个如上所述。因此，$B^{A_0}$ 是一个范畴。
现在我们在 $B^{A_0}$ 上定义一个结构概念，其中 $P(F_0)$ 是扩展 $F_0$ 为函子的操作的类型 $F : \prod_{(a,a':A_0)} \hom_A(a, a') \to \hom_B(F_0 a, F_0 a')$（即保持合成和身份）。这是一个集合，因为每个 $\hom_B(-, -)$ 是这样。给定这样的 $F$

和 $G$，如果它形成了一个自然变换，我们定义 $\gamma : \hom_{B^{A_0}}(F_0, G_0)$ 为同态。在 Definition 9.2.3 中，我们基本上验证了这是一种结构概念。此外，如果 $F$ 和 $F'$ 都是 $F_0$ 上的结构，并且恒等是从 $F$ 到 $F'$ 的自然变换，那么对于任何 $f : \hom_A(a, a')$，我们有 $F'f = F'f \circ 1_{F_0 a} = 1_{F_0 a} \circ Ff = Ff$。应用函数外延性，我们得出结论 $F = F'$。因此，我们有一个标准结构概念，因此根据 Theorem 9.8.2，预范畴 $B^A$ 是一个范畴。

作为另一个例子，我们考虑一阶签名的结构范畴。我们定义一个**一阶签名** (first-order signature)，$\Omega$，其由函数符号 $\Omega_0$ 和关系符号 $\Omega_1$ 组成，$\omega : \Omega_0$ 和 $\omega : \Omega_1$，每个符号都有一个为集合的元数 $|\omega|$。一个 **$\Omega$-结构** ($\Omega$-structure)$a$ 由一个集合 $|a|$ 以及为每个函数符号 $\omega$ 分配的一个 $|\omega|$ 元函数 $\omega^a : |a|^{|\omega|} \to |a|$ 和为每个关系符号 $\omega$ 分配的一个 $|\omega|$ 元关系 $\omega^a$ 组成，赋予每个 $x : |a|^{|\omega|}$ 一个单纯命题 $\omega^a x$。并且对于给定的 $\Omega$-结构 $a, b$，如果它保持结构，即对于每个签名符号 $\omega$ 和每个 $x : |a|^{|\omega|}$，

(i) 如果 $\omega : \Omega_0$，则 $f(\omega^a x) = \omega^b(f \circ x)$，并且
(ii) 如果 $\omega : \Omega_1$，则 $\omega^a x \to \omega^b(f \circ x)$。

请注意，每个 $x : |a|^{|\omega|}$ 是一个函数 $x : |\omega| \to |a|$ 使得 $f \circ x : b^\omega$。
现在我们假设给定一个（单值的）宇宙 $\mathcal{U}$ 和一个 $\mathcal{U}$-小的签名 $\Omega$；即 $|\Omega|$ 是一个 $\mathcal{U}$-小集合，并且对于每个 $\omega : |\Omega|$，集合 $|\omega|$ 是 $\mathcal{U}$-小集合。然后我们有 $\mathcal{U}$-小集合的范畴 $\mathcal{S}et_\mathcal{U}$。我们希望定义 $\mathcal{S}et_\mathcal{U}$ 上的 $\mathcal{U}$-小 $\Omega$-结构的预范畴，并使用 Theorem 9.8.2 来证明它是一个范畴。
我们使用一阶签名 $\Omega$ 给我们一个 $\mathcal{S}et_\mathcal{U}$ 上的标准结构概念 $(P, H)$。

Definition 9.8.4.

(i) 对于每个 $\mathcal{U}$-小集合 $x$ 定义

$$Px :\equiv P_0 x \times P_1 x$$

这里

$$P_0 x :\equiv \prod_{\omega:\Omega_0} x^{|\omega|} \to x, \text{ 和}$$
$$P_1 x :\equiv \prod_{\omega:\Omega_1} x^{|\omega|} \to \mathsf{Prop}_\mathcal{U},$$

(ii) 对于 $\mathcal{U}$-小集合 $x, y$ 和 $\alpha : P^\omega x$，$\beta : P^\omega y$，$f : x \to y$，定义

$$H_{\alpha\beta}(f) :\equiv H_{0,\alpha\beta}(f) \wedge H_{1,\alpha\beta}(f)$$

这里

$$H_{0,\alpha\beta}(f) :\equiv \forall(\omega : \Omega_0).\forall(u : x^{|\omega|}).f(\alpha u) = \beta(f \circ u), \text{ 和}$$
$$H_{1,\alpha\beta}(f) :\equiv \forall(\omega : \Omega_1).\forall(u : x^{|\omega|}).\alpha u \to \beta(f \circ u)$$

现在例行公事地检查 $(P, H)$ 是 $\mathcal{S}et_\mathcal{U}$ 上的标准结构概念，因此我们可以使用 Theorem 9.8.2 得出 $\mathcal{U}$-小 $\Omega$-结构的预范畴 $Str_{(P,H)}(\mathcal{S}et_\mathcal{U})$ 是一个范畴。只需观察到这本质上与 $\mathcal{S}et_\mathcal{U}$ 上的 $\mathcal{U}$-小 $\Omega$-结构的预范畴相同。

## 9.9 Rezk **完备** (The Rezk completion)

在本节中，我们将给出一种将预范畴替换为范畴的通用方法。实际上，我们将给出两种方法。这两种方法都依赖于这样一个事实：“范畴将弱等价视为等价”。
为了证明这一点，我们首先引入一些完全标准的范畴论引理，这些引理经过仔细表述，以确保我们正确使用了 $\|-\|_{-1}$ 的消元器。如果我们想要避免选择公理，在经典范畴论中也需要同样谨慎地处理：任何时候我们想要定义一个函数，我们都需要以某种方式唯一地描述其值。

Lemma 9.9.1. 如果 $A, B, C$ 是预范畴，并且 $H : A \to B$ 是一个本质上满的函子，那么 $(- \circ H) : C^B \to C^A$ 是忠实的。

证明. 设 $F, G : B \to C$，并且 $\gamma, \delta : F \to G$ 满足 $\gamma H = \delta H$；我们必须证明 $\gamma = \delta$。因此设 $b : B$；我们想要证明 $\gamma_b = \delta_b$。这是一个单纯命题，因此由于 $H$ 本质上满，我们可以假设给定 $a : A$ 和同构 $f : Ha \cong b$。但是现在我们有

$$\gamma_b = G(f) \circ \gamma_{Ha} \circ F(f^{-1}) = G(f) \circ \delta_{Ha} \circ F(f^{-1}) = \delta_b \qquad \square$$

Lemma 9.9.2. 如果 $A, B, C$ 是预范畴，并且 $H : A \to B$ 是本质上满且充满的，那么 $(- \circ H) : C^B \to C^A$ 是充分忠实的。

证明. 现在还需要证明充满性。因此，设 $F, G : B \to C$ 和 $\gamma : FH \to GH$。我们声称对于任意 $b : B$，类型

$$\sum_{(g : \hom_C(Fb, Gb))} \prod_{(a:A)} \prod_{(f : Ha \cong b)} (\gamma_a = Gf^{-1} \circ g \circ Ff) \tag{9.9.3}$$

是可约的。由于可约性是一个单纯命题，并且 $H$ 是本质上满的，我们可以假设给定 $a_0 : A$ 和 $h : Ha_0 \cong b$。

现在令 $g :\equiv Gh \circ \gamma_{a_0} \circ Fh^{-1}$。那么给定任意其他的 $a : A$ 和 $f : Ha \cong b$，我们必须证明 $\gamma_a = Gf^{-1} \circ g \circ Ff$。由于 $H$ 是充满的，存在一个态射 $k : \hom_A(a, a_0)$ 使得 $Hk = h^{-1} \circ f$。并且由于我们的目标是一个单纯命题，我们可以假设给定某个这样的 $k$。然后我们有

$$\begin{aligned} \gamma_a &= GHk^{-1} \circ \gamma_{a_0} \circ FHk \\ &= Gf^{-1} \circ Gh \circ \gamma_{a_0} \circ Fh^{-1} \circ Ff \\ &= Gf^{-1} \circ g \circ Ff. \end{aligned}$$

因此，(9.9.3) 是可居住的。现在还需要证明它是一个单纯命题。设 $g, g' : \hom_C(Fb, Gb)$，并且对于所有 $a : A$ 和 $f : Ha \cong b$，我们有 $(\gamma_a = Gf^{-1} \circ g \circ Ff)$ 和 $(\gamma_a = Gf^{-1} \circ g' \circ Ff)$。依赖积类型是单纯命题，因此我们只需要证明 $g = g'$。但这是一个单纯命题，因此我们可以假设 $a_0 : A$ 和 $h : Ha_0 \cong b$，在这种情况下我们有

$$g = Gh \circ \gamma_{a_0} \circ Fh^{-1} = g'$$

这证明了对于所有 $b : B$，(9.9.3) 是可约的。现在我们通过为每个 $b$ 取 (9.9.3) 中唯一的 $g$ 来定义 $\delta : F \to G$。为了证明这是自然的，假设给定 $f : \hom_B(b, b')$；我们必须证明 $Gf \circ \delta_b = \delta_{b'} \circ Ff$。和以前一样，我们可以假设 $a : A$ 和 $h : Ha \cong b$，以及 $a' : A$ 和 $h' : Ha' \cong b'$。由于 $H$ 既是充满的也是本质上满的，我们还可以假设 $k : \hom_A(a, a')$ 并且 $Hk = h'^{-1} \circ f \circ h$。

由于 $\gamma$ 是自然的，$GHk \circ \gamma_a = \gamma_{a'} \circ FHk$。使用 $\delta$ 的定义，我们有

$$\begin{aligned} Gf \circ \delta_b &= Gf \circ Gh \circ \gamma_a \circ Fh^{-1} \\ &= Gh' \circ GHk \circ \gamma_a \circ Fh^{-1} \\ &= Gh' \circ \gamma_{a'} \circ FHk \circ Fh^{-1} \\ &= Gh' \circ \gamma_{a'} \circ Fh'^{-1} \circ Ff \\ &= \delta_{b'} \circ Ff. \end{aligned}$$

因此，$\delta$ 是自然的。最后，对于任意 $a : A$，将 $\delta_{Ha}$ 的定义应用于 $a$ 和 $1_a$，我们得到 $\gamma_a = \delta_{Ha}$。因此，$\delta \circ H = \gamma$。 $\square$

定理的其余部分几乎完全按照相同的思路进行，唯一不同的是在一个关键步骤中插入了范畴的性质，我们在下面用斜体强调了这一点。这是在定义一个到*对象*的函数时不使用选择公理的步骤，因此我们必须小心定义对象“唯一确定”的含义。在经典范畴论中，我们只能说这个对象在唯一同构意义上是确定的，但在集合论基础中，这样的唯一性并不足以在不引入选择公理的情况下定义一个函数。然而，在单值基础中，如果 $C$ 是一个范畴，那么同构就是等同，并且我们有适当的唯一性（即存在于一个可约空间中）。

Theorem 9.9.4. 如果 $A, B$ 是预范畴，$C$ 是一个范畴，并且 $H : A \to B$ 是一个弱等价，那么 $(- \circ H) : C^B \to C^A$ 是一个同构。

证明. 由 Theorem 9.2.5，$C^B$ 和 $C^A$ 是范畴。因此，根据 Lemma 9.4.14，我们只需证明 $(- \circ H)$ 是一个等价。但是，由于我们从前面两个引理中知道它是充分忠实的，根据 Lemma 9.4.7，我们只需证明它是本质上满的。因此，假设 $F : A \to C$；我们希望仅存在一个 $G : B \to C$ 使得 $GH \cong F$。
对于每个 $b : B$，令 $X_b$ 为一个类型，其元素由以下组成：

(i) 一个元素 $c : C$；以及
(ii) 对于每个 $a : A$ 和 $h : Ha \cong b$，一个同构 $k_{a,h} : Fa \cong c$；满足
(iii) 对于 (ii) 中的每个 $(a, h)$ 和 $(a', h')$ 以及每个满足 $h' \circ Hf = h$ 的态射 $f : \hom_A(a, a')$，我们有 $k_{a',h'} \circ Ff = k_{a,h}$。

我们声称对于任意 $b : B$，类型 $X_b$ 是可约的。由于这是一个单纯命题，我们可以假设给定 $a_0 : A$ 和 $h_0 : Ha_0 \cong b$。令 $c^0 :\equiv Fa_0$。接下来，给定 $a : A$ 和 $h : Ha \cong b$，由于 $H$ 是充分忠实的，存在一个唯一的同构 $g_{a,h} : a \to a_0$，使得 $Hg_{a,h} = {h_0}^{-1} \circ h$；定义 $k^0_{a,h} :\equiv Fg_{a,h}$。最后，如果 $h' \circ Hf = h$，那么 ${h_0}^{-1} \circ h' \circ Hf = {h_0}^{-1} \circ h$，因此 $g_{a',h'} \circ f = g_{a,h}$，从而 $k^0_{a',h'} \circ Ff = k^0_{a,h}$。因此，$X_b$ 是可居住的。
现在假设给定另一个 $(c^1, k^1) : X_b$。那么 $k^1_{a_0,h_0} : c^0 \equiv Fa_0 \cong c^1$。由于 $C$ 是一个范畴，我们有 $p : c^0 = c^1$ 并且 $\mathsf{idtoiso}(p) = k^1_{a_0,h_0}$。并且对于任意 $a : A$ 和 $h : Ha \cong b$，通过 (iii)，对于 $(c^1, k^1)$ 和 $f :\equiv g_{a,h}$，我们有

$$k^1_{a,h} = k^1_{a_0,h_0} \circ k^0_{a,h} = p_*\left(k^0_{a,h}\right)$$

这提供了等同性 $(c^0, k^0) = (c^1, k^1)$ 所需的数据，从而完成了 $X_b$ 可约性的证明。
现在，由于对于每个 $b$，$X_b$ 是可约的，类型 $\prod_{(b:B)} X_b$ 也是可约的。特别是，它是可居住的，因此我们有一个函数，将每个 $b : B$ 映射到一个 $c$ 和一个 $k$。定义 $G_0(b)$ 为这个 $c$；这给出了一个函数 $G_0 : B_0 \to C_0$。
接下来我们需要定义 $G$ 在态射上的作用。对于每个 $b, b' : B$ 和 $f : \hom_B(b, b')$，令 $Y_f$ 为一个类型，其元素由以下组成：

(iv) 一个态射 $g : \hom_C(Gb, Gb')$，使得
(v) 对于每个 $a : A$ 和 $h : Ha \cong b$，以及每个 $a' : A$ 和 $h' : Ha' \cong b'$，以及任意 $\ell : \hom_A(a, a')$，我们有

$$(h' \circ H\ell = f \circ h) \to (k_{a',h'} \circ F\ell = g \circ k_{a,h})$$

我们声称对于任意 $b, b'$ 和 $f$，类型 $Y_f$ 是可约的。由于这是一个单纯命题，我们可以假设给定 $a_0 : A$ 和 $h_0 : Ha_0 \cong b$，以及每个 $a'_0 : A$ 和 $h'_0 : Ha'_0 \cong b'$。然后由于 $H$ 是充分忠实的，存在一个唯一的 $\ell_0 : \hom_A(a_0, a'_0)$ 使得 $h'_0 \circ H\ell_0 = f \circ h_0$。定义 $g_0 :\equiv k_{a'_0,h'_0} \circ F\ell_0 \circ (k_{a_0,h_0})^{-1}$。
现在对于任意 $a, h, a', h'$，和 $\ell$ 使得 $(h' \circ H\ell = f \circ h)$，我们有 $h^{-1} \circ h_0 : Ha_0 \cong Ha$，因此存在一个唯一的 $m : a_0 \cong a$ 使得 $Hm = h^{-1} \circ h_0$，因此 $h \circ Hm = h_0$。类似地，我们有一个唯一的 $m' : a'_0 \cong a'$ 使得 $h' \circ Hm' = h'_0$。现在通过 (iii)，我们有 $k_{a,h} \circ Fm = k_{a_0,h_0}$ 和 $k_{a',h'} \circ Fm' = k_{a'_0,h'_0}$。我们还有

$$\begin{aligned} Hm' \circ H\ell_0 &= (h')^{-1} \circ h'_0 \circ H\ell_0 \\ &= (h')^{-1} \circ f \circ h_0 \\ &= (h')^{-1} \circ f \circ h \circ h^{-1} \circ h_0 \\ &= H\ell \circ Hm \end{aligned}$$

因此，由于 $H$ 是充分忠实的，$m' \circ \ell_0 = \ell \circ m$。最后，我们可以计算出

$$\begin{aligned} g_0 \circ k_{a,h} &= k_{a'_0,h'_0} \circ F\ell_0 \circ (k_{a_0,h_0})^{-1} \circ k_{a,h} \\ &= k_{a'_0,h'_0} \circ F\ell_0 \circ Fm^{-1} \\ &= k_{a'_0,h'_0} \circ (Fm')^{-1} \circ F\ell \\ &= k_{a',h'} \circ F\ell \end{aligned}$$

这完成了 $Y_f$ 可居住性的证明。为了证明它是可约的，由于态射集是集合，因此只需取另一个 $g_1 : \hom_C(Gb, Gb')$ 满足 (v) 并证明 $g_0 = g_1$。然而，我们仍然保留了指定的 $a_0, h_0, a_0', h_0', \ell_0$，并且 (v) 表明 $g_0$ 和 $g_1$ 必须都等于 $k_{a_0',h_0'} \circ F\ell_0 \circ (k_{a_0,h_0})^{-1}$。
这完成了对于每个 $b, b' : B$ 和 $f : \hom_B(b, b')$，$Y_f$ 是可约的证明。因此，有一个函数将每个 $f$ 映射到其唯一的居住者；将此函数记为 $G_{b,b'} : \hom_B(b, b') \to \hom_C(Gb, Gb')$。$G$ 是一个函子的证明是直截了当的；在每种情况下，我们都可以选择 $a, h$ 并应用 (v)。
最后，对于任意 $a_0 : A$，定义 $c :\equiv Fa_0$ 并且 $k_{a,h} :\equiv Fg$，其中 $g : \hom_A(a, a_0)$ 是与 $Hg = h$ 唯一同构的同构，给出了 $X_{Ha_0}$ 的一个元素。因此，它等于指定的一个；因此 $GHa = Fa$。类似地，对于 $f : \hom_A(a_0, a_0')$，我们可以通过沿着这些等同性传递来定义 $Y_{Hf}$ 的一个元素，因此它必须等于指定的一个。因此，我们有 $GH = F$，从而 $GH \cong F$ 如预期。 □

因此，如果一个预范畴 $A$ 允许一个弱等价函子 $A \to \widehat{A}$ 映射到一个范畴，那么它就是范畴的“反射”：从 $A$ 到范畴的任何函子都将通过 $\widehat{A}$ 本质上唯一地分解。我们现在给出两个此类弱等价的构造。

Theorem 9.9.5. *对于任何预范畴 $A$，存在一个范畴 $\widehat{A}$ 和一个弱等价 $A \to \widehat{A}$。*

*第一个证明.* 令 $\widehat{A}_0 :\equiv \left\{ F : \mathcal{S}et^{A^{\mathrm{op}}} \;\middle|\; \exists (a : A).\, (\mathbf{y}a \cong F) \right\}$，其态射集继承自 $\mathcal{S}et^{A^{\mathrm{op}}}$。然后，包含映射 $\widehat{A} \to \mathcal{S}et^{A^{\mathrm{op}}}$ 是充分忠实的，并且在对象上是嵌入的。由于 $\mathcal{S}et^{A^{\mathrm{op}}}$ 是一个范畴（由 Theorem 9.2.5，因为 $\mathcal{S}et$ 是由单值性公理给出的），$\widehat{A}$ 也是一个范畴。
令 $A \to \widehat{A}$ 为 Yoneda 嵌入。由 Corollary 9.5.6，这条映射是充分忠实的，并且根据 $\widehat{A}_0$ 的定义本质上满。因此它是一个弱等价。 □

这个证明非常简洁，但它的缺点是它增加了宇宙级别。如果 $A$ 是 $\mathcal{U}$ 宇宙中的一个范畴，那么在这个证明中 $\mathcal{S}et$ 必须至少与 $\mathcal{S}et_{\mathcal{U}}$ 一样大。然后 $\mathcal{S}et_{\mathcal{U}}$ 和 $(\mathcal{S}et_{\mathcal{U}})^{A^{\mathrm{op}}}$ 本身不是 $\mathcal{U}$ 中的范畴，而只是在更高的宇宙中是范畴，并且*先验地* $\widehat{A}$ 也是如此。人们可以想象一个可以处理这个问题的重置公理，但也可以使用高阶归纳类型直接构造。

*第二个证明.* 我们定义一个高阶归纳类型 $\widehat{A}_0$，其具有以下构造函数：

- 一个函数 $i : A_0 \to \widehat{A}_0$。
- 对于每个 $a, b : A$ 和 $e : a \cong b$，一个等同性 $je : ia = ib$。
- 对于每个 $a : A$，一个等同性 $j(1_a) = \mathsf{refl}_{ia}$。
- 对于每个 $(a, b, c : A)$，$(f : a \cong b)$，和 $(g : b \cong c)$，一个等同性 $j(g \circ f) = j(f) \cdot j(g)$。
- 1-截断：对于所有 $x, y : \widehat{A}_0$ 和 $p, q : x = y$ 以及 $r, s : p = q$，一个等同性 $r = s$。

注意，对于任何 $a, b : A$ 和 $p : a = b$，我们有 $j(\mathsf{idtoiso}(p)) = i(p)$。这通过对 $p$ 的路径归纳得出，并且通过第三个构造函数给出。
类型 $\widehat{A}_0$ 将是 $\widehat{A}$ 的对象类型；我们现在构建其余的结构。（以下证明是那种可以从计算机证明助手的帮助中获益匪浅的：它是广泛而浅显的，涉及许多简短的情况需要考虑，大部分工作是写下需要检查的内容。）

*步骤 1:* 我们通过对 $\widehat{A}_0$ 进行双重归纳定义一个族 $\hom_{\widehat{A}} : \widehat{A}_0 \to \widehat{A}_0 \to \mathsf{Set}$。由于 $\mathsf{Set}$ 是一个 1-类型，我们可以忽略 1-截断构造函数。当 $x$ 和 $y$ 是 $ia$ 和 $ib$ 时，我们取 $\hom_{\widehat{A}}(ia, ib) :\equiv \hom_A(a, b)$。还需要考虑其他可能的构造对。
让我们首先保持 $x = ia$ 固定。如果 $y$ 随着等同性 $je : ib = ib'$ 变化，对于某些 $e : b \cong b'$，我们需要一个等同性 $\hom_A(a, b) = \hom_A(a, b')$。通过单值性，给出一个等价 $\hom_A(a, b) \simeq \hom_A(a, b')$ 就足够了。我们取这个函数 $(e \circ -) : \hom_A(a, b) \to \hom_A(a, b')$。要证明这是一个等价，我们将其逆定义为 $(e^{-1} \circ -)$，并通过 $e$ 的逆是 $e$ 在 $A$ 中的逆的事实来证明其逆性。
当 $y$ 随着等同性 $j(1_b) = \mathsf{refl}_{ib}$ 变化时，我们需要一个等同性 $(1_b \circ -) = \mathsf{refl}_{\hom_A(a,b)}$；这由预范畴的身份公理 $1_b \circ g = g$ 推导而来。类似地，当 $y$ 随着等同性 $j(g \circ f) = j(f) \cdot j(g)$ 变化时，我们需要一个等同性 $((g \circ f) \circ -) = (g \circ (f \circ -))$，这由结合律推导而来。

现在我们考虑 $x$ 的其他构造。假设 $x$ 随着等同性 $j(e) : ia = ia'$ 变化，对于某些 $e : a \cong a'$；我们再次必须处理 $y$ 的所有构造。如果 $y$ 是 $ib$，那么我们需要一个等同性 $\hom_A(a,b) = \hom_A(a',b)$。通过单值性，这可能来自一个等价，并且我们可以使用 $(- \circ e^{-1})$，逆为 $(- \circ e)$。
仍然是 $x$ 随着 $j(e)$ 变化，假设现在 $y$ 也随 $j(f)$ 变化，对于某些 $f : b \cong b'$。那么我们需要知道两个级联的等同性

$$\hom_A(a,b) = \hom_A(a',b) = \hom_A(a',b') \qquad \text{和}$$
$$\hom_A(a,b) = \hom_A(a,b') = \hom_A(a',b')$$

是相同的。这由结合律得出：$(f \circ -) \circ e^{-1} = f \circ (- \circ e^{-1})$。$y$ 的其他两个构造是平凡的，因为它们是集合中的 2-重等同性。
对于 $x$ 的接下来的两个构造，$y$ 的所有构造除了第一个都是平凡的。当 $x$ 随着 $j(1_a) = \mathsf{refl}_{ia}$ 变化，并且 $y$ 是 $ib$ 时，我们再次使用身份公理。类似地，当 $x$ 随着 $j(g \circ f) = j(f) \cdot j(g)$ 变化时，我们再次使用结合律。这完成了 $\hom_{\widehat{A}} : \widehat{A}_0 \to \widehat{A}_0 \to \mathsf{Set}$ 的构造。

步骤 2：我们通过对 $\widehat{A}_0$ 进行归纳来给出 $\widehat{A}$ 上的预范畴结构。我们现在正在向集合（$\widehat{A}$ 的态射集）消元，因此除了前两个构造外，所有构造都很容易处理。
对于恒等式，如果 $x$ 是 $ia$，那么我们有 $\hom_{\widehat{A}}(x,x) \equiv \hom_A(a,a)$ 并且我们定义 $1_x :\equiv 1_{ia}$。如果 $x$ 随着 $je$ 变化，对于 $e : a \cong a'$，我们必须证明 $\mathsf{transport}^{x \mapsto \hom_{\widehat{A}}(x,x)}(je, 1_{ia}) = 1_{ia'}$。但根据 $\hom_{\widehat{A}}$ 的定义，沿着 $je$ 传递等同于与 $e$ 和 $e^{-1}$ 复合，并且我们有 $e \circ 1_{ia} \circ e^{-1} = 1_{ia'}$。
对于复合，如果 $x, y, z$ 分别是 $ia, ib, ic$，那么 $\hom_{\widehat{A}}$ 简化为 $\hom_A$，我们可以定义 $\widehat{A}$ 中的复合作为 $A$ 中的复合。当 $x$、$y$ 或 $z$ 随着 $je$ 变化时，我们验证以下等同性：

$$\begin{aligned} e \circ (g \circ f) &= (e \circ g) \circ f, \\ g \circ f &= (g \circ e^{-1}) \circ (e \circ f), \\ (g \circ f) \circ e^{-1} &= g \circ (f \circ e^{-1}) \end{aligned}$$

最后，结合性和单性公理是单纯命题，因此除了第一个构造外，所有构造都是平凡的。但是在这种情况下，我们有 $A$ 中的相应公理。

步骤 3：我们展示 $\widehat{A}$ 是一个范畴。即我们必须证明对于所有 $x, y : \widehat{A}$，函数 $\mathsf{idtoiso} : (x = y) \to (x \cong y)$ 是一个等价。首先我们定义，对于所有 $x, y : \widehat{A}$，一个函数 $k_{x,y} : (x \cong y) \to (x = y)$，通过归纳。如前所述，由于我们的目标是一个集合，除了前两个构造外，所有构造都是平凡的。
当 $x$ 和 $y$ 分别是 $ia$ 和 $ib$ 时，我们有 $\hom_{\widehat{A}}(ia, ib) \equiv \hom_A(a,b)$，其复合和恒等式也继承了，因此 $(ia \cong ib)$ 等价于 $(a \cong b)$。但现在我们有构造 $j : (a \cong b) \to (ia = ib)$。
接下来，如果 $y$ 随着 $j(e)$ 变化，对于某些 $e : b \cong b'$，我们必须证明对于 $f : a \cong b$，我们有 $j(j(e)_*(f)) = j(f) \cdot j(e)$。但是通过 $\hom_{\widehat{A}}$ 在等同性上的定义，沿着 $j(e)$ 传递等同于与 $e$ 后复合，因此这个等同性由 $\widehat{A}_0$ 的最后一个构造得出。当 $x$ 随着 $j(e)$ 变化对于 $e : a \cong a'$ 时，其余情况类似。这完成了 $k : \prod_{(x,y:\widehat{A}_0)} (x \cong y) \to (x = y)$ 的定义。
现在我们必须展示的一件事是，如果 $p : x = y$，那么 $k(\mathsf{idtoiso}(p)) = p$。通过对 $p$ 的归纳，我们可以假设它是 $\mathsf{refl}_x$，因此 $\mathsf{idtoiso}(p) \equiv 1_x$。现在我们通过对 $x : \widehat{A}_0$ 的归纳进行论证，并且由于我们的目标是一个单纯命题（因为 $\widehat{A}_0$ 是一个 1-类型），除了第一个构造外，所有构造都是平凡的。但是如果 $x$ 是 $ia$，那么 $k(1_{ia}) \equiv j(1_a)$，它等于 $\mathsf{refl}_{ia}$ 由 $\widehat{A}_0$ 的第三个构造得出。
为了完成 $\widehat{A}$ 是一个范畴的证明，我们必须证明如果 $f : x \cong y$，那么 $\mathsf{idtoiso}(k(f)) = f$。通过归纳我们可以假设 $x$ 和 $y$ 分别是 $ia$ 和 $ib$，在这种情况下 $f$ 必须来自一个同构 $g : a \cong b$ 并且我们有 $k(f) \equiv j(g)$。然而，对于任何 $p$ 我们有 $\mathsf{idtoiso}(p) = p_*(1)$，特别地 $\mathsf{idtoiso}(j(g)) = j(g)_*(1_{ia})$。并且通过 $\hom_{\widehat{A}}$ 在等同性上的定义，这个传递等同于将 $1_{ia}$ 与等价 $g$ 复合，因此等于 $g$。
注意这一步与在 §2.8, ??, and Chapter 8 中使用的编码-解码方法的相似性。再次，我们通过递归地定义一族代码（这里是 $(x,y) \mapsto (x \cong y)$）和编码与解码函数，来描述一个高阶归纳类型的身份类型（这里是 $\widehat{A}_0$），并在 $\widehat{A}_0$ 和路径上进行归纳。

步骤 4：我们定义一个弱等价 $I : A \to \widehat{A}$。我们取 $I_0 :\equiv i : A_0 \to \widehat{A}_0$，并且通过 $\hom_{\widehat{A}}$ 的构造，我们有函数 $I_{a,b} : \hom_A(a,b) \to \hom_{\widehat{A}}(Ia, Ib)$ 形成函子 $I : A \to \widehat{A}$。这个函子通过构造是充分忠实的，因

此剩下的就是证明它本质上是满的。即对于所有 $x : \widehat{A}$ 我们希望仅存在一个 $a : A$ 使得 $Ia \cong x$。如前所述，我们通过对 $x$ 的归纳进行论证，并且由于目标是一个单纯命题，除了第一个构造外，所有构造都是平凡的。但是如果 $x$ 是 $ia$，那么当然我们有 $a : A$ 并且 $Ia \equiv ia$，因此 $Ia \cong ia$。（请注意，如果我们试图证明 $I$ 是分裂本质上满的，我们将陷入困境，因为我们对 $A_0$ 中的等同性一无所知，因此无法处理进一步的构造。） □

我们将构造 $A \mapsto \widehat{A}$ 称为 Rezk **完备** (Rezk completion)，尽管也有一个（来自高阶拓扑语义的）论点称其为**堆栈完备** (stack completion)。
我们已经看到，大多数在实践中出现的预范畴都是范畴，因为它们是从 $\mathcal{S}et$ 构造的，而 $\mathcal{S}et$ 由于单值性公理而是一个范畴。然而，在某些情况下，为了获得一个范畴，Rezk 完备是必要的。

Example 9.9.6. 回想 Example 9.1.17，对于任意类型 $X$，存在一个预群范畴，其对象类型为 $X$，且 $\hom(x, y) :\equiv \|x = y\|_0$。其 Rezk 完备是 $X$ 的基本群范畴 (fundamental groupoid)。回想群范畴等价于 1-类型，不难将此群范畴识别为 $\|X\|_1$。

Example 9.9.7. 回想 Example 9.1.18，存在一个预范畴，其对象类型为 $\mathcal{U}$ 且 $\hom(X, Y) :\equiv \|X \to Y\|_0$。其 Rezk 完备可以称为**类型的同伦范畴** (homotopy category of types)。其对象类型可以识别为 $\|\mathcal{U}\|_1$（见 Exercise 9.9）。

Rezk 完备还允许我们表明，“范畴”的概念是由“预范畴的弱等价”的概念所决定的。因此，后者不可避免地决定了前者。

Theorem 9.9.8. *一个预范畴 $C$ 是一个范畴，当且仅当对于每个预范畴的弱等价 $H : A \to B$，诱导的函子 $(- \circ H) : C^B \to C^A$ 是一个预范畴的同构。*

*证明.* “只有如果”是 Theorem 9.9.4。在另一个方向上，令 $H$ 为 $I : A \to \widehat{A}$。然后由于 $(- \circ I)_0$ 是等价的，存在 $R : \widehat{A} \to A$ 使得 $RI = 1_A$。因此 $IRI = I$，但再次由于 $(- \circ I)_0$ 是等价的，这意味着 $IR = 1_{\widehat{A}}$。根据 Lemma 9.4.9(iii)，$I$ 是预范畴的同构。但是由于 $\widehat{A}$ 是一个范畴，因此 $A$ 也是一个范畴。 □

## Notes

范畴的最初定义当然是在集合论基础上的，因此一个范畴的对象集形成一个集合（对于大范畴，则是一个类）。随着时间的推移，人们逐渐认识到，所有“范畴论”中的对象性质在同构下都是不变的，并且范畴中对象的等同性通常不是一个非常有用的概念。许多作者 [?, ?, ?, ?] 发现，依赖类型逻辑能够在不引入对象等同性概念的情况下，形式化范畴的定义，并且在这种逻辑中可以证明的陈述恰恰是那些在同构下不变的“范畴论”陈述。

尽管大多数范畴论似乎在对象的同构和范畴的等价下不变，但也有一些有趣的例外，这引发了关于“范畴论”意味着什么的哲学讨论。例如，Example 9.6.3 被彼得 · 梅 (Peter May) 于 2010 年 5 月在范畴邮件列表上提出，作为一个在两个范畴（按照集合论中的通常定义）是同构而不仅仅是等价时显得重要的案例。†-范畴的情况也对那些倡导同构不变版本的范畴论的人来说有些困惑，因为 †-范畴中对象之间的“正确”相同性概念并不是普通的同构，而是*酉同构* (unitary isomorphism)。

满足“饱和”或“单值性”原则的范畴，如 Definition 9.1.6所述，最早由 Hofmann 和 Streicher [?] 考虑。这一条件随后在几年后几乎同时由 Voevodsky、Shulman 和其他人独立提出，并由 Ahrens 和 Kapulkin [?] 形式化。该框架将所有上述例子置于统一的背景中：一些预范畴是范畴，另一些是严格范畴，等等。一个关于“同构蕴含等同”的一般定理（假设单值性公理）由 Coquand 和 Danielsson 证明；§9.8 中结构等同性原则的表述归功于 Aczel。

独立于关于范畴论的哲学考虑，Rezk [?] 发现，在定义 $(\infty,1)$-范畴的概念时，使用一个不仅包含态射空间的对象集，而且包含所有等价和同伦的对象空间，是非常方便的。这种方式产生了一种非常良好表现的 $(\infty,1)$-范畴模型，称为*完备* Segal *空间* (complete Segal spaces)。这种模型的一个特别好的方面是与 Lemma 9.4.14的类似性：一个完备 Segal 空间的映射当且仅当它是一个逐层等价的 Simplicial 空间时，才是等价的。

在 Voevodsky 的单值基础的 Simplicial 集模型中，我们的预范畴类似于 Rezk 的“Segal 空间”的截断类比，而我们的范畴则对应于他的“完备 Segal 空间”。严格范畴则对应于（弱化和截断版本的）所谓的“Segal 范畴”。已知 Segal 范畴和完备 Segal 空间是 $(\infty,1)$-范畴的等价模型（见例如 [?]），因此在 Simplicial 集模型中，范畴和严格范畴产生了“等价”的范畴理论——尽管如我们所见，前者仍然有许多优势。然而，在更一般的高阶拓扑语义学中，一个严格范畴对应于相应的 1-拓扑中（传统意义上的）内部范畴，而一个范畴对应于*堆栈* (stack)。后者通常是相对于拓扑的更合适的“范畴”形式。

在 Rezk 的上下文中，我们称之为“Rezk 完备”的东西对应于完备 Segal 空间模型范畴中的纤维化替代。由于这是通过一个超限归纳论证建立的，它最接近我们的第二个构造作为一个高阶归纳类型。然而，在同伦类型论的高阶拓扑模型中，Rezk 完备对应于*堆栈完备* (stack completion)，它可以通过超限归纳 [?] 或使用 Yoneda 嵌入 [?] 构造。

## Exercises

Exercise 9.1. 对于一个预范畴 $A$ 和 $a : A$，定义**切片预范畴** (slice precategory) $A/a$。证明如果 $A$ 是一个范畴，那么 $A/a$ 也是一个范畴。

Exercise 9.2. 对于任意集合 $X$，证明切片范畴 $\mathcal{S}et/X$ 等价于函子范畴 $\mathcal{S}et^X$，其中在后者情况下，我们将 $X$ 视为一个离散范畴。

Exercise 9.3. 证明一个函子是范畴的等价，当且仅当它是*右伴随* (right adjoint)，且其单位和余单位是同构。

Exercise 9.4. 定义**预 2-范畴** (pre-2-category) 的概念。证明如 §9.2 中定义的预范畴、函子和自然变换构成一个预 2-范畴。同样，通过用满足类似一致性条件的自然同构替换 Lemmas 9.2.9 and 9.2.11 中的等式，定义**预双范畴** (pre-bicategory)。定义一个从预 2-范畴到预双范畴的函数，并证明当其同态预范畴是范畴时，它的限制和共限制成为一个等价。

Exercise 9.5. 定义 2-**范畴** (2-category) 为一个满足类似于 Definition 9.1.6的条件的预 2-范畴。验证范畴的预 2-范畴 $\mathcal{C}at$ 是一个 2-范畴。本章的内容在多大程度上可以在一个任意的 2-范畴中进行内部处理?

Exercise 9.6. 定义一个 2-范畴，其对象是 1-类型，其态射是函数，其 2-态射是同伦。证明它在适当的意义下等价于由*群范畴* (groupoids)（每个箭头都是同构的范畴）构成的 $\mathcal{C}at$ 的全子 2-范畴。

Exercise 9.7. 回想一个*严格范畴* (strict category) 是一个对象类型为集合的预范畴。证明严格范畴的预 2-范畴等价于以下预 2-范畴。

- 其对象是范畴 $A$，并配备一个满射 $p_A : A'_0 \to A_0$，其中 $A'_0$ 是一个集合。
- 其态射是函子 $F : A \to B$，并配备一个函数 $F'_0 : A'_0 \to B'_0$ 使得 $p_B \circ F'_0 = F_0 \circ p_A$。
- 其 2-态射只是自然变换。

Exercise 9.8. 定义 †-范畴的预 2-范畴，其中其同态预范畴上具有 † 结构。证明两个 †-范畴在适当的意义下当且仅当它们是“酉等价”的，才是相等的。

Exercise 9.9. 证明一个函数 $X \to Y$ 是等价的当且仅当其在 Example 9.9.7 的同伦范畴中的像是一个同构。证明这个范畴的对象类型是 $\|\mathcal{U}\|_1$。

Exercise 9.10. 构造一个 †-预范畴的 †-Rezk 完备，使其成为一个 †-范畴，并赋予其一个适当的泛性质。

Exercise 9.11. 使用 Examples 9.1.17 and 9.9.6 中的基本（预）群范畴和 §9.9 中的 Rezk 完备，给出范坎彭定理 (??) 的另一种证明。

Exercise 9.12. 设 $X$ 和 $Y$ 是集合，$p : Y \to X$ 是一个满射。

(i) 对于任意预范畴 $A$，定义相对于 $p$ 的 **下降数据** (descent data) 范畴 $\mathrm{Desc}(A, p)$。
(ii) 证明任意预范畴 $A$ 对于 $p$ 都是一个**预堆栈** (prestack)，即典范函子 $A^X \to \mathrm{Desc}(A, p)$ 是完全忠实的。
(iii) 证明如果 $A$ 是一个范畴，则对于 $p$，它是一个**堆栈** (stack)，即 $A^X \to \mathrm{Desc}(A, p)$ 是一个等价。
(iv) 证明命题“每个严格范畴对于每个集合满射都是堆栈”与选择公理等价。

Chapter 10

# 集合论 (Set theory)

我们将集合理解为具有特别简单的同伦特征的类型，参见 §3.1。这种理解与 Zermelo–Fraenkel集合论中的集合完全不同，后者形成了一个包含复杂嵌套隶属关系的累积层次结构。对于许多数学目的，同伦理论中的集合与 Zermelo–Fraenkel 集合一样好用，但它们之间存在重要差异。

我们在本章开始于 §10.1，展示了范畴 $\mathcal{S}et$ 具有通常集合范畴的（大部分）性质。在构造性、预测性、单值性基础上，它是一个 “ΠW-前拓扑（pretopos）”；而如果我们假设命题的缩放 (§3.5)，它是一个初等拓扑（elementary topos），如果我们假设 LEM 和 AC，那么它是 Lawvere 的集合范畴的初等理论的模型。这足以确保在同伦类型论中的集合行为类似于大多数数学家在集合论之外使用的集合。

在本章的其余部分，我们研究了一些传统上属于“集合论”的主题。在 §§10.2–10.4中，我们研究了基数和序数。这些通常在集合论中使用全局隶属关系来定义，但我们将看到，单值性公理使得一种同样方便、更具“结构性”的方法成为可能。

最后，在 §10.5中，我们考虑在同伦类型论内部构建一个具有二元隶属关系的类似于 Zermelo–Fraenkel 集合论的累积层次结构的可能性。这结合了高阶归纳类型与代数集合论领域的思想。

在本章中，我们将经常使用 §3.7中描述的传统逻辑符号。除了 Chapters 2 and 3的基本理论外，我们还使用了 §§6.8 and 6.10中关于余极限和商集的高阶归纳类型，以及 Chapter 7中关于截断的某些理论，特别是在 §7.6中提到的因子分解系统在 $n = -1$ 的情况。在 §10.3中，我们使用了一个归纳族 (§5.7) 来描述良序性，在 §10.5中，我们使用了一个更复杂的高阶归纳类型来呈现累积层次结构。

## 10.1 集合范畴 (The category of sets)

回顾在 Chapter 9中，我们定义了范畴 0-类型（在某个宇宙）及其之间的映射组成，并且观察到它是一个范畴（不仅仅是一个预范畴）。我们将依次考虑。

### 10.1.1 极限和余极限 (Limits and colimits)

由于集合在积下是封闭的，Theorem 2.9.2中的积的泛性质立即表明。实际上，无穷积也同样容易从等价式中得出：

$$\Big(X \to \prod_{a:A} B(a)\Big) \simeq \Big(\prod_{a:A} (X \to B(a))\Big).$$

我们在 Exercise 2.11中看到，$f : A \to C$ 和 $g : B \to C$ 的拉回可以定义为 $\sum_{(a:A)} \sum_{(b:B)} f(a) = g(b)$；如果 $A, B, C$ 都是集合，则它是一个集合并继承了正确的泛性质。因此，完备范畴。

由于集合在 $+$ 下是封闭的并包含 $\mathbf{0}$ ,。同样，由于 $\sum_{(a:A)} B(a)$ 是一个集合，当且仅当 $A$ 和每个 $B(a)$ 是集合时，它在 $B$ 的余积。最后，我们在 §7.4中证明了 $n$-类型中的推出存在，这特别包括了 $\mathcal{S}et$ 。因此，余完备的。

### 10.1.2 像 (Images)

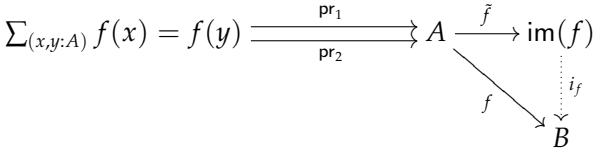
接下来，我们展示**正则范畴** (regular category)，即：

(i) 。
(ii) 任何函数 $f : A \to B$ 的核对偶 $\mathsf{pr}_1, \mathsf{pr}_2 : (\sum_{(x,y:A)} f(x) = f(y)) \to A$ 具有余等化子。
(iii) 正则满态射的拉回再次是正则满态射。

回想一个**正则满态射** (regular epimorphism) 是某对映射的余等化子。因此在(iii)中，余等化子的拉回需要再次是余等化子，但不一定是被拉回对偶的。
$f : A \to B$ 的核对偶的余等化子的明显候选者是 $f$ 的像，如 §7.6中定义的那样。回想我们定义了 $\mathsf{im}(f) :\equiv \sum_{(b:B)} \left\| \mathsf{fib}_f(b) \right\|$，并且定义了 $\tilde{f} : A \to \mathsf{im}(f)$ 和 $i_f : \mathsf{im}(f) \to B$，如下所示：

$$\tilde{f} :\equiv \lambda a.\, \big(f(a),\, \big|(a, \mathsf{refl}_{f(a)})\big|\big)$$
$$i_f :\equiv \mathsf{pr}_1$$

它们构成了一个图：




回想一个函数 $f : A \to B$ 称为满射 (surjective)，如果 $\forall(b : B).\, \left\| \mathsf{fib}_f(b) \right\|$，或者等价地 $\forall(b : B).\, \exists(a : A).\, f(a) = b$。我们还说过，两个集合之间的函数 $f : A \to B$ 称为单射 (injective)，如果 $\forall(a, a' : A).\, (f(a) = f(a')) \Rightarrow (a = a')$，或者等价地，如果它的每个纤维是一个简单命题。由于这些是在 Chapter 7意义上的 $(-1)$-连通和 $(-1)$-截断映射，一般理论表明，上述 $\tilde{f}$ 是满射而 $i_f$ 是单射，并且这种因子分解在拉回下是稳定的。
我们现在将单射性和满射性与适当的范畴理论概念进行比较。首先我们观察到范畴中的单态射和满态射有一个略微更强的等价公式。

Lemma 10.1.1. *对于范畴 $A$ 中的一个态射 $f : \hom_A(a, b)$，以下条件是等价的。*

(i) *$f$ 是一个单态射* (monomorphism)*：对于所有 $x : A$ 和 $g, h : \hom_A(x, a)$，如果 $f \circ g = f \circ h$，则 $g = h$。*
(ii) *（如果 $A$ 有拉回）对角线映射 $a \to a \times_b a$ 是一个同构。*
(iii) *对于所有 $x : A$ 和 $k : \hom_A(x, b)$，类型 $\sum_{(h:\hom_A(x,a))} (k = f \circ h)$ 是一个简单命题。*
(iv) *对于所有 $x : A$ 和 $g : \hom_A(x, a)$，类型 $\sum_{(h:\hom_A(x,a))} (f \circ g = f \circ h)$ 是一个收缩的。*

*证明.* 条件 (i)和 (ii)的等价性是标准范畴论。现在考虑 $\hom_A(x, a)$ 和 $\hom_A(x, b)$ 之间的函数 $(f \circ -)$。条件 (i)表示它是单射，而 (iii)表示它的纤维是简单命题；因此它们是等价的。 (iii)通过取 $k :\equiv f \circ g$ 并记住被占用的简单命题是收缩的来隐含 (iv)。最后， (iv)隐含 (i)，因为如果 $p : f \circ g = f \circ h$，那么 $(g, \mathsf{refl})$ 和 $(h, p)$ 都包含在 (iv)中的类型中，因此是相等的，所以 $g = h$。 □

Lemma 10.1.2. *集合之间的一个函数 $f : A \to B$ 是单射的当且仅当它是。*

*证明.* 留给读者。 □

当然，**满态射** (epimorphism) 是在对偶范畴中的单态射。我们现在展示，在，满态射正是满射，同时也正是余等化子（正则满态射）。
两个集合 $A$ 和 $B$ 之间的 $f, g : A \to B$ 的余等化子在 $\mathcal{S}et$ 中定义为一般（同伦）余等化子的 0-截断。为了清楚起见，我们可以将其称为**集合余等化子** (set-coequalizer)。它的泛性质方便地表达如下。

Lemma 10.1.3. 设 $f,g : A \to B$ 为两个集合 $A$ 和 $B$ 之间的函数。集合余等化子 $c_{f,g} : B \to Q$ 具有如下性质：对于任意集合 $C$ 和任意 $h : B \to C$，满足 $h \circ f = h \circ g$，类型

$$\sum_{k:Q\to C} (k \circ c_{f,g} = h)$$

是收缩的。

Lemma 10.1.4. 对于集合之间的任意函数 $f : A \to B$，以下条件是等价的：

(i) $f$ 是一个满态射。

(ii) 考虑推出图

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ \downarrow & & \downarrow{\scriptstyle \iota} \\ \mathbf{1} & \xrightarrow[t]{} & C_f \end{array}$$

在 $\mathcal{S}et$ 中定义了映射锥。那么类型 $C_f$ 是收缩的。

(iii) $f$ 是满射。

证明. 设 $f : A \to B$ 为一个集合之间的函数，假设它是一个满态射；我们证明 $C_f$ 是收缩的。$C_f$ 的构造器 $\mathbf{1} \to C_f$ 给了我们一个元素 $t : C_f$。我们需要证明

$$\prod_{x:C_f} x = t.$$

请注意 $x = t$ 是一个简单命题，因此我们可以对 $C_f$ 进行归纳。当然，当 $x$ 为 $t$ 时，我们有 $\mathsf{refl}_t : t = t$，所以足以找到

$$I_0 : \prod_{b:B} \iota(b) = t$$
$$I_1 : \prod_{a:A} \alpha_1(a)^{-1} \cdot I_0(f(a)) = \mathsf{refl}_t$$

其中 $\iota : B \to C_f$ 和 $\alpha_1 : \prod_{(a:A)} \iota(f(a)) = t$ 是 $C_f$ 的其他构造器。请注意 $\alpha_1$ 是 $\iota \circ f$ 到 $\mathsf{const}_t \circ f$ 的一个同伦，因此我们可以找到元素

$$(\iota, \mathsf{refl}_{\iota\circ f}), (\mathsf{const}_t, \alpha_1) : \sum_{h:B\to C_f} \iota \circ f \sim h \circ f.$$

通过 Lemma 10.1.1(iv)的对偶（以及函数扩展性），我们有一条路径

$$\gamma : (\iota, \mathsf{refl}_{\iota\circ f}) = (\mathsf{const}_t, \alpha_1).$$

因此，我们可以定义 $I_0(b) :\equiv \mathsf{happly}(\mathsf{ap}_{\mathsf{pr}_1}(\gamma), b) : \iota(b) = t$。我们还有

$$\mathsf{ap}_{\mathsf{pr}_2}(\gamma) : \mathsf{ap}_{\mathsf{pr}_1}(\gamma)_*\left(\mathsf{refl}_{\iota\circ f}\right) = \alpha_1$$

此传输涉及 $f$ 的前置，它与 $\mathsf{happly}$ 一起工作。因此，从路径类型中的传输我们得到 $I_0(f(a)) = \alpha_1(a)$，对于任何 $a : A$，这给了我们 $I_1$。

现在假设 $C_f$ 是收缩的；我们证明 $f$ 是满射。我们首先通过 $C_f$ 上的递归构造一个类型族 $P : C_f \to \mathsf{Prop}$，这是有效的，因为。在点构造器上，我们定义

$$P(t) :\equiv \mathbf{1}$$
$$P(\iota(b)) :\equiv \left\|\mathsf{fib}_f(b)\right\|.$$

为了完成 $P$ 的构造，我们还需要为所有 $a : A$ 给出一条路径 $\left\|\mathsf{fib}_f(f(a))\right\| =_{\mathsf{Prop}} \mathbf{1}$ 然而，$\left\|\mathsf{fib}_f(f(a))\right\|$ 是由 $(f(a), \mathsf{refl}_{f(a)})$ 居住的。由于它是一个简单命题，这意味着它是收缩的——因此是等价的，因此与。这完成了 $P$ 的定义。现在，由于 $C_f$ 被假设为收缩的，因此 $P(x)$ 对于任何 $x : C_f$ 都是等价于 $P(t)$ 的。特别是，$P(\iota(b)) \equiv \left\|\mathsf{fib}_f(b)\right\|$ 等价于 $P(t) \equiv \mathbf{1}$，对于每个 $b : B$，因此是收缩的。因此，$f$ 是满射。

最后，假设 $f : A \to B$ 是满射，并考虑一个集合 $C$ 和两个函数 $g, h : B \to C$，它们具有 $g \circ f = h \circ f$ 的性质。由于 $f$ 被假设为满射，因此对于所有 $b : B$，类型 $\left\|\mathsf{fib}_f(b)\right\|$ 是收缩的。因此我们有以下等价：

$$\begin{aligned}
\prod_{b:B} (g(b) = h(b)) &\simeq \prod_{b:B} \left( \left\|\mathsf{fib}_f(b)\right\| \to (g(b) = h(b)) \right) \\
&\simeq \prod_{b:B} \left( \mathsf{fib}_f(b) \to (g(b) = h(b)) \right) \\
&\simeq \prod_{(b:B)} \prod_{(a:A)} \prod_{(p:f(a)=b)} g(b) = h(b) \\
&\simeq \prod_{a:A} g(f(a)) = h(f(a))
\end{aligned}$$

使用在第二行中的事实，即 $g(b) = h(b)$ 是一个简单命题，因为 $C$ 是一个集合。但根据假设，有该类型的一个元素。 □

Theorem 10.1.5. *范畴 $\mathcal{S}et$ 是正则的。此外，集合之间的满射是正则满态射。*

*证明.* 这是范畴论中的一个标准引理，即范畴是正则的，只要它承认有限极限和稳定于拉回的正交因子分解系统 $(\mathcal{E}, \mathcal{M})$，其中 $\mathcal{M}$ 是单态射，在这种情况下，$\mathcal{E}$ 自动由正则满态射组成。(参见例如 [?, A1.3.4])。因子分解系统的存在性在 Theorem 7.6.6中得到证明。 □

Lemma 10.1.6. *在，正则满态射的拉回是正则满态射。*

*证明.* 我们在 Theorem 7.6.9中展示了，$n$-连通函数的拉回是 $n$-连通的。通过 Theorem 10.1.5，当 $n = -1$ 时应用这一结论就足够了。 □

,我们有了"像"运算作用于子集。也就是说,给定 $f : A \to B$,任何子集 $P : \mathcal{P}(A)$(即谓词 $P : A \to \mathsf{Prop}$)都有一个**像** (image)，它是 $B$ 的一个子集。这可以直接定义为 $\{ y : B \mid \exists (x : A). f(x) = y \wedge P(x) \}$，或间接地定义为复合函数

$$\{ x : A \mid P(x) \} \to A \xrightarrow{f} B]$$

$\{ f(x) \mid P(x) \}$ 来表示 $P$ 的像。

### 10.1.3 商 (Quotients)

现在我们知道 $\mathcal{S}et$ 是正则的，要表明 $\mathcal{S}et$ 是精确的，我们需要证明每个等价关系都是有效的。换句话说，给定等价关系 $R : A \to A \to \mathsf{Prop}$，存在一个对偶的余等化子 $c_R$，并且 $\mathsf{pr}_1$ 和 $\mathsf{pr}_2$ 形成 $c_R$ 的核对偶。

我们已经在 §6.10中看到了两个构造集合按等价关系 $R : A \to A \to \mathsf{Prop}$ 的商的方法。第一个可以描述为

$$\mathsf{pr}_1, \mathsf{pr}_2 : \left( \sum_{x,y:A} R(x,y) \right) \to A]$$

Definition 10.1.7. 一个关系 $R : A \to A \to \mathsf{Prop}$ 被称为**有效的** (effective)，如果方框

$$\begin{array}{ccc}
\sum_{(x,y:A)} R(x,y) & \xrightarrow{\mathsf{pr}_1} & A \\
{\scriptstyle \mathsf{pr}_2}\downarrow & & \downarrow{\scriptstyle c_R} \\
A & \xrightarrow[c_R]{} & A/R
\end{array}$$

是一个拉回。

由于 $c_R$ 和它本身的标准拉回是 $\sum_{(x,y:A)}(c_R(x) = c_R(y))$，通过 Theorem 4.7.7，这相当于要求 $c_R(x) = c_R(y)$ 的典型转换是一个纤维等价。

Lemma 10.1.8. *假设 $(A,R)$ 是一个等价关系。那么对于任何 $x,y:A$，有一个等价关系*

$$(c_R(x) = c_R(y)) \simeq R(x,y)$$

*换句话说，等价关系是有效的。*

*证明.* 我们首先通过对 $A/R$ 进行双重归纳将 $R$ 扩展为一个关系 $\widetilde{R} : A/R \to A/R \to \mathsf{Prop}$，然后我们将证明它与 $A/R$ 上的恒等类型等价。我们定义 $\widetilde{R}(c_R(x), c_R(y)) :\equiv R(x,y)$。对于 $r : R(x,x')$ 和 $s : R(y,y')$，$R$ 的传递性和对称性给出了 $R(x,y)$ 到 $R(x',y')$ 的一个等价关系。这完成了 $\widetilde{R}$ 的定义。现在要证明对于每个 $w,w' : A/R$，$\widetilde{R}(w,w') \simeq (w = w')$。方向 $(w = w') \to \widetilde{R}(w,w')$ 通过传输一次我们证明了 $\widetilde{R}$ 是反射的，这是一种简单的归纳。另一个方向 $R(w,w') \to (w = w')$ 是一个简单命题，因此由于 $c_R : A \to A/R$ 是满射，只需要假设 $w$ 和 $w'$ 是 $c_R(x)$ 和 $c_R(y)$ 形式的。但在这种情况下，我们有典型映射 $R(c_R(x), c_R(y)) :\equiv R(x,y) \to (c_R(x) = c_R(y))$。（再次注意到编码解码方法的出现。）□

第二个商的构造是作为 $R$ 的等价类的集合（它的幂集的子集）：

$$A /\!\!/ R :\equiv \{\, P : A \to \mathsf{Prop} \mid P \text{ is an equivalence class of } R \,\}]$$

AR
注意，如果我们将 $R$ 视为 $A$ 到 $A \to \mathsf{Prop}$ 的函数，那么 $A /\!\!/ R$ 等价于 §10.1.2中构造的 $\mathsf{im}(R)$。现在在 Theorem 10.1.5中我们已经证明了图像是余等化子。特别是，我们立即得到余等化子图

$$\textstyle\sum_{(x,y:A)} R(x) = R(y) \;\underset{\mathsf{pr}_2}{\overset{\mathsf{pr}_1}{\rightrightarrows}}\; A \longrightarrow A /\!\!/ R$$

我们可以用这个来给出另一个证明，即任何等价关系是有效的，并且两个商的定义是一致的。

Theorem 10.1.9. *对于任意两个集合之间的函数 $f : A \to B$，关系 $\ker(f) : A \to A \to \mathsf{Prop}$ 由 $\ker(f,x,y) :\equiv (f(x) = f(y))$ 给出是有效的。*

*证明.* 我们将使用 $\mathsf{pr}_1, \mathsf{pr}_2 : (\sum_{(x,y:A)} f(x) = f(y)) \to A$ 的余等化子 $\mathsf{im}(f)$。注意，函数

$$c_f :\equiv \lambda a.\left(f(a), \left\|(a, \mathsf{refl}_{f(a)})\right\|\right) : A \to \mathsf{im}(f)$$

的核对偶由两个投影

$$\mathsf{pr}_1, \mathsf{pr}_2 : \Big(\sum_{x,y:A} c_f(x) = c_f(y)\Big) \to A$$

对于任何 $x,y:A$，我们有等价

$$\begin{aligned}(c_f(x) = c_f(y)) &\simeq \Big(\sum_{p:f(x)=f(y)} p_*(\left\|(x, \mathsf{refl}_{f(x)})\right\|) = \left\|(y, \mathsf{refl}_{f(y)})\right\|\Big)\\ &\simeq (f(x) = f(y))\end{aligned}$$

其中最后一个等价关系成立，因为 $\left\|\mathsf{fib}_f(b)\right\|$ 对于任何 $b : B$ 来说是一个简单命题。因此，我们得到

$$\Big(\sum_{x,y:A} c_f(x) = c_f(y)\Big) \simeq \Big(\sum_{x,y:A} f(x) = f(y)\Big)$$

并且我们可以得出结论，对于任何函数 $f$，$\ker f$ 是一个有效的关系。□

Theorem 10.1.10. *等价关系是有效的，并且 $A/R \simeq A /\!\!/ R$。*

*证明.* 我们需要分析余等化子图

$$\sum_{(x,y:A)} R(x) = R(y) \underset{\mathsf{pr}_2}{\overset{\mathsf{pr}_1}{\rightrightarrows}} A \longrightarrow A /\!\!/ R$$

通过单值化公理，类型 $R(x) = R(y)$ 等价于从 $R(x)$ 到 $R(y)$ 的同伦类型，并且进一步等价于 $\prod_{(z:A)} R(x,z) \simeq R(y,z)$。由于 $R$ 是一个等价关系，后者空间等价于 $R(x,y)$。总之，我们得到 $(R(x) = R(y)) \simeq R(x,y)$，因此 $R$ 是有效的，因为它等价于一个有效的关系。此外，图

$$\sum_{(x,y:A)} R(x,y) \underset{\mathsf{pr}_2}{\overset{\mathsf{pr}_1}{\rightrightarrows}} A \longrightarrow A /\!\!/ R$$

是一个余等化子图。由于余等化子是等价的，因此可以得出 $A/R \simeq A /\!\!/ R$。 □

我们通过提到商的第三种可能的构造来结束本节。考虑一个以 $A$ 为对象的预范畴，其同态集为 $R$；此预范畴的 Rezk 完成（参见 §9.9）的对象类型将是该等价关系的商。读者可以检查详细信息。

### 10.1.4 $\mathcal{S}et$ 是一个 ΠW-前拓扑范畴

所谓的 ΠW-*前拓扑范畴* 是一种局部笛卡尔闭范畴 具有不相交的有限上积，有效等价关系，以及多项式自函子的初始代数的范畴——这被认为是一种“预测性”的拓扑概念，即“预测性集合”的范畴，可用于构造数学 就像通常的集合范畴对于经典数学 的用途一样。
通常，在构造性类型论中，求助于“集合体”——一种精确补全——的外部构造来获得具有此类闭包性质的范畴。特别是，良好行为的商在数学中通常涉及（非构造性）幂集的许多构造中是必需的。值得注意的是，统一基础通过更高归纳类型（higher inductive types）提供了这些内部构造，而无需此类外部构造。这代表了我们方法的强大优势，我们将在后续示例中看到。

Theorem 10.1.11. *范畴 $\mathcal{S}et$ 是一个 ΠW-前拓扑范畴。*

*证明.* 我们有一个初始对象 $\mathbf{0}$ 和有限、不相交的上积 $A + B$。这些在拉回时保持稳定，原因很简单，因为拉回有一个右伴随。事实上，$\mathcal{S}et$ 是局部笛卡尔闭的，因为对于集合之间的任何映射 $f : A \to B$，使用“纤维化替换”$\sum_{(a:A)} f(a) = b$ 等价于 $A$（在 $B$ 上），并且我们有该替换的依赖函数类型。我们刚刚展示了 $\mathcal{S}et$ 是正则的（Theorem 10.1.5），并且商是有效的（Lemma 10.1.8）。因此，我们有一个局部笛卡尔闭前拓扑范畴。最后，由于 $n$-类型在 Exercise 7.3中通过多项式自函子的初始代数（Theorem 5.4.7）构成，我们看到 $\mathcal{S}et$ 是一个 ΠW-前拓扑范畴。 □

人们自然会想知道，有什么（如果有的话）阻止 $\mathcal{S}et$ 成为一个（基本）拓扑? 除了已经提到的结构，拓扑还具有一个*子对象分类器*：这是一个指示对象，用于分类（等价类的）单态射（monomorphisms）。实际上，在具有子对象分类器的情况下，事情变得稍微简单一些：仅需要笛卡尔闭包即可获得余积。在同伦类型论中，单值化公理表明，类型 $\mathsf{Prop} :\equiv \sum_{(X:\mathcal{U})} \mathsf{isProp}(X)$ 确实分类单态射（通过类似于 §4.8的论证），但通常它与周围的宇宙 $\mathcal{U}$ 一样大。因此，它在某种意义上是一个“集合”，因为它是一个 0-类型，但它不是“小的”，因为它不是 $\mathcal{U}$ 的对象，因此不是范畴 $\mathcal{S}et$ 的对象。然而，如果我们假设一种适当形式的命题缩放（见 §3.5），那么我们可以找到 $\mathsf{Prop}$ 的一个小版本，使得 $\mathcal{S}et$ 成为一个基本的拓扑范畴。

Theorem 10.1.12. *如果存在一个类型 $\Omega : \mathcal{U}$，包含所有简单命题，那么范畴 $\mathcal{S}et_{\mathcal{U}}$ 是一个基本拓扑范畴。*

一个足够的条件是排中律，在“简单命题”形式中，我们称之为 $\mathsf{LEM}$ ；因为在这种情况下，我们有 $\mathsf{Prop} = \mathbf{2}$，这是“小的”，并且可以分类所有的简单命题。此外，拓扑理论中一个众所周知的充分条件是选择公理，这是经典 集合论中经常假设的公理。在下一节中，我们将简要探讨这些条件在我们环境下的关系。

### 10.1.5 选择公理蕴含排中律

我们从以下引理开始。

Lemma 10.1.13. *如果 $A$ 是一个简单命题，那么其悬挂 $\Sigma(A)$ 是一个集合，并且 $A$ 等价于 $\mathsf{N} =_{\Sigma(A)} \mathsf{S}$。*

*证明.* 为了证明 $\Sigma(A)$ 是一个集合，我们定义一个族 $P : \Sigma(A) \to \Sigma(A) \to \mathcal{U}$，使得对于每个 $x, y : \Sigma(A)$，$P(x,y)$ 是一个简单命题，并且它等价于 $\Sigma(A)$ 上的同一类型 $\mathsf{Id}_{\Sigma(A)}$。我们做以下定义：

$$P(\mathsf{N},\mathsf{N}) :\equiv \mathbf{1} \qquad P(\mathsf{S},\mathsf{N}) :\equiv A$$
$$P(\mathsf{N},\mathsf{S}) :\equiv A \qquad P(\mathsf{S},\mathsf{S}) :\equiv \mathbf{1}$$

我们需要检查定义是否保持路径。对于任何 $a : A$，有一个子午线 $\mathsf{merid}(a) : \mathsf{N} = \mathsf{S}$，因此我们应该有

$$P(\mathsf{N},\mathsf{N}) = P(\mathsf{N},\mathsf{S}) = P(\mathsf{S},\mathsf{N}) = P(\mathsf{S},\mathsf{S})$$

但由于 $A$ 由 $a$ 占据，它等价于 $\mathbf{1}$，因此我们有

$$P(\mathsf{N},\mathsf{N}) \simeq P(\mathsf{N},\mathsf{S}) \simeq P(\mathsf{S},\mathsf{N}) \simeq P(\mathsf{S},\mathsf{S})$$

单值化公理将这些转换为所需的相等性。此外，$P(x,y)$ 对于所有 $x, y : \Sigma(A)$ 是一个简单命题，这通过对 $x$ 和 $y$ 的归纳以及简单命题是一个简单命题这一事实来证明。
注意，$P$ 是一个自反关系。因此，我们可以应用 Theorem 7.2.2，因此只需构造 $\tau : \prod_{(x,y:\Sigma(A))} P(x,y) \to (x = y)$。我们通过双重归纳来实现。当 $x$ 是 $\mathsf{N}$ 时，我们定义 $\tau(\mathsf{N})$ 为

$$\tau(\mathsf{N},\mathsf{N},u) :\equiv \mathsf{refl}_{\mathsf{N}} \qquad \text{以及} \qquad \tau(\mathsf{N},\mathsf{S},a) :\equiv \mathsf{merid}(a)$$

如果 $A$ 由 $a$ 占据，那么 $\mathsf{merid}(a) : \mathsf{N} = \mathsf{S}$，因此我们还需要 $\mathsf{merid}(a)_*(\tau(\mathsf{N},\mathsf{N})) = \tau(\mathsf{N},\mathsf{S})$ 通过函数外延性，我们使用以下事实来完成这一步：

$$\mathsf{merid}(a)_*(\tau(\mathsf{N},\mathsf{N},x)) = \tau(\mathsf{N},\mathsf{N},x) \cdot \mathsf{merid}(a)^{-1} \equiv$$
$$\mathsf{refl}_{\mathsf{N}} \cdot \mathsf{merid}(a) = \mathsf{merid}(a) = \mathsf{merid}(x) \equiv \tau(\mathsf{N},\mathsf{S},x)$$

以对称的方式，我们可以通过以下方式定义 $\tau(\mathsf{S})$：

$$\tau(\mathsf{S},\mathsf{N},a) :\equiv \mathsf{merid}(a)^{-1} \qquad \text{以及} \qquad \tau(\mathsf{S},\mathsf{S},u) :\equiv \mathsf{refl}_{\mathsf{S}}$$

为了完成 $\tau$ 的构造，我们需要检查 $\mathsf{merid}(a)_*(\tau(\mathsf{N})) = \tau(\mathsf{S})$，对于任何 $a : A$。验证过程与上述类似，通过对 $\tau$ 的第二个参数的归纳来进行。
因此，通过 Theorem 7.2.2 我们得出 $\Sigma(A)$ 是一个集合，并且对于所有 $x, y : \Sigma(A)$ 有 $P(x,y) \simeq (x = y)$。取 $x :\equiv \mathsf{N}$ 和 $y :\equiv \mathsf{S}$ 即可得到所需的 $A \simeq (\mathsf{N} =_{\Sigma(A)} \mathsf{S})$。 □

Theorem 10.1.14 (Diaconescu 定理). *选择公理蕴含排中律。*

*证明.* 我们使用在 Lemma 3.8.2 中给出的选择公理的等价形式。考虑一个简单命题 $A$。定义一个函数 $f : \mathbf{2} \to \Sigma(A)$，其定义为 $f(0_2) :\equiv \mathsf{N}$ 和 $f(1_2) :\equiv \mathsf{S}$。这个函数是满射的。实际上，我们有 $(0_2, \mathsf{refl}_{\mathsf{N}}) : \mathsf{fib}_f(\mathsf{N})$ 和 $(1_2, \mathsf{refl}_{\mathsf{S}}) : \mathsf{fib}_f(\mathsf{S})$。由于 $\left\|\mathsf{fib}_f(x)\right\|$ 是一个简单命题，通过归纳法可以得出所需的满射性。
根据 Lemma 10.1.13，悬挂 $\Sigma(A)$ 是一个集合，因此根据选择公理，存在一个从 $\Sigma(A)$ 到 $\mathbf{2}$ 的截面 $g : \Sigma(A) \to \mathbf{2}$。由于 $\mathbf{2}$ 上的相等性是可判定的，我们得到

$$(g(f(0_2)) = g(f(1_2))) + \neg(g(f(0_2)) = g(f(1_2))),$$

并且，由于 $g$ 是 $f$ 的一个截面，因此是单射，

$$(f(0_2) = f(1_2)) + \neg(f(0_2) = f(1_2))$$

最后，由于 $(f(0_2) = f(1_2)) = (\mathsf{N} = \mathsf{S}) = A$ 根据 Lemma 10.1.13，我们有 $A + \neg A$。 □

Theorem 10.1.15. 如果选择公理成立，则类别 $\mathcal{S}et$ 是一个带选择的良好定点布尔初等拓扑。

证明. 由于 AC 蕴含 LEM ，通过 Theorem 10.1.12 以及后面的注释，我们可以得到一个带选择的布尔初等拓扑。我们将良好定点性的证明留给读者作为练习 (Exercise 10.3)。 □

Remark 10.1.16. 定理中提到的关于范畴的条件被称为 Lawvere 用于"集合范畴的初等理论"的公理 [?]。

## 10.2 基数 (Cardinal numbers)

Definition 10.2.1. **基数类型** (type of cardinal numbers) 是集合类型 (Set) 的 0-截断：

$$\mathsf{Card} :\equiv \|\mathsf{Set}\|_0$$

因此，一个 **基数** (cardinal number)，或称 **基数** (cardinal)，是 $\mathsf{Card} \equiv \|\mathsf{Set}\|_0$ 的一个元素。技术上，当然每个宇宙 $\mathcal{U}$ 都有一个单独的基数类型 $\mathsf{Card}_{\mathcal{U}}$ 。

和通常的截断一样，如果 $A$ 是一个集合，那么 $|A|_0$ 表示其在从集合类型到 0-截断 $\|\mathsf{Set}\|_0 \equiv \mathsf{Card}$ 的标准映射下的像；我们称 $|A|_0$ 为 $A$ 的 **基数** (cardinality)。根据定义，Card 是一个集合。它还继承了来自 Set 的半环结构。

Definition 10.2.2. **基数加法** (cardinal addition) 的运算定义为截断上的归纳：

$$(- + -) : \mathsf{Card} \to \mathsf{Card} \to \mathsf{Card}$$

定义为：

$$|A|_0 + |B|_0 :\equiv |A + B|_0.$$

证明. 由于 $\mathsf{Card} \to \mathsf{Card}$ 是一个集合，要为所有 $\alpha : \mathsf{Card}$ 定义 $(\alpha + -) : \mathsf{Card} \to \mathsf{Card}$，通过归纳，只需要假设 $\alpha$ 是某个 $A : \mathsf{Set}$ 的 $|A|_0$。现在我们想要定义 $(|A|_0 + -) : \mathsf{Card} \to \mathsf{Card}$，即我们想要为所有 $\beta : \mathsf{Card}$ 定义 $|A|_0 + \beta : \mathsf{Card}$。然而，由于 Card 是一个集合，通过归纳，只需要假设 $\beta$ 是某个 $B : \mathsf{Set}$ 的 $|B|_0$。但是现在我们可以定义 $|A|_0 + |B|_0$ 为 $|A + B|_0$。 □

Definition 10.2.3. 类似地，**基数乘法** (cardinal multiplication) 的运算定义为截断上的归纳：

$$(- \cdot -) : \mathsf{Card} \to \mathsf{Card} \to \mathsf{Card}$$

定义为：

$$|A|_0 \cdot |B|_0 :\equiv |A \times B|_0$$

Lemma 10.2.4. Card 是一个交换半环 (commutative semiring)，即对于 $\alpha, \beta, \gamma : \mathsf{Card}$ 我们有以下性质。

$$(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$$
$$\alpha + 0 = \alpha$$
$$\alpha + \beta = \beta + \alpha$$
$$(\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma)$$
$$\alpha \cdot 1 = \alpha$$
$$\alpha \cdot \beta = \beta \cdot \alpha$$
$$\alpha \cdot (\beta + \gamma) = \alpha \cdot \beta + \alpha \cdot \gamma$$

其中 $0 :\equiv |\mathbf{0}|_0$ 且 $1 :\equiv |\mathbf{1}|_0$。

证明. 我们证明乘法的交换性，即 $\alpha \cdot \beta = \beta \cdot \alpha$；其他性质的证明完全类似。由于 Card 是一个集合，类型 $\alpha \cdot \beta = \beta \cdot \alpha$ 是一个纯命题，并且特别地是一个集合。因此，通过截断上的归纳，只需要假设 $\alpha$ 和 $\beta$ 分别是某些 $A, B : \mathsf{Set}$ 的 $|A|_0$ 和 $|B|_0$。现在 $|A|_0 \cdot |B|_0 \equiv |A \times B|_0$ 和 $|B|_0 \cdot |A|_0 \equiv |B \times A|_0$，所以只需要证明 $A \times B = B \times A$。最后，通过同一性原理 (univalence)，只需要给出一个等价 $A \times B \simeq B \times A$。这很简单：取 $(a, b) \mapsto (b, a)$ 及其显然的逆映射即可。 □

Definition 10.2.5. **基数指数** (cardinal exponentiation) 的运算同样定义为截断上的归纳：

$$|A|_0^{|B|_0} :\equiv |B \to A|_0.$$

Lemma 10.2.6. 对于 $\alpha, \beta, \gamma : \mathsf{Card}$ 我们有以下性质：

$$\begin{aligned}
\alpha^0 &= 1 \\
1^\alpha &= 1 \\
\alpha^1 &= \alpha \\
\alpha^{\beta+\gamma} &= \alpha^\beta \cdot \alpha^\gamma \\
\alpha^{\beta\cdot\gamma} &= (\alpha^\beta)^\gamma \\
(\alpha \cdot \beta)^\gamma &= \alpha^\gamma \cdot \beta^\gamma
\end{aligned}$$

证明. 证明与 Lemma 10.2.4 类似。 □

Definition 10.2.7. **基数不等式** (cardinal inequality) 的关系定义为截断上的归纳：

$$|A|_0 \le |B|_0 :\equiv \|\mathsf{inj}(A, B)\|$$

其中 $\mathsf{inj}(A, B)$ 是从 $A$ 到 $B$ 的单射 (injections) 的类型。换句话说，$|A|_0 \le |B|_0$ 意味着从 $A$ 到 $B$ 仅仅存在一个单射。

Lemma 10.2.8. 基数不等式是一个预序 (preorder)，即对于 $\alpha, \beta : \mathsf{Card}$ 我们有：

$$\begin{aligned}
&\alpha \le \alpha \\
&(\alpha \le \beta) \to (\beta \le \gamma) \to (\alpha \le \gamma)
\end{aligned}$$

证明. 同前，通过截断上的归纳。例如，类型 $(\alpha \le \beta) \to (\beta \le \gamma) \to (\alpha \le \gamma)$ 是一个纯命题，因此通过 0-截断上的归纳，我们可以假设 $\alpha$、$\beta$ 和 $\gamma$ 分别是 $|A|_0$、$|B|_0$ 和 $|C|_0$。现在，由于 $|A|_0 \le |C|_0$ 是一个纯命题，通过 $(-1)$-截断上的归纳，我们可以假设给定了从 $A$ 到 $B$ 的单射 $f$ 和从 $B$ 到 $C$ 的单射 $g$。但是，$g \circ f$ 是从 $A$ 到 $C$ 的一个单射，因此 $|A|_0 \le |C|_0$ 成立。反身性 (reflexivity) 更容易证明。 □

我们还可以证明基数不等式与半环运算的兼容性。

Lemma 10.2.9. 考虑以下命题：

(i) 存在从 $A$ 到 $B$ 的一个单射。
(ii) 存在从 $B$ 到 $A$ 的一个满射。

那么，在假设排中律 (excluded middle) 的情况下：

- 给定 $a_0 : A$，我们有 (i)→(ii)。
- 因此，如果 $A$ 是居留 (merely inhabited) 的，我们有 (i) → 仅仅存在 (ii)。
- 假设选择公理 (axiom of choice)，我们有 (ii) → 仅仅存在 (i)。

证明. 如果 $f : A \to B$ 是一个单射，则定义 $g : B \to A$ 如下。由于 $f$ 是单射，因此 $f$ 在 $b$ 处的纤维是一个纯命题。因此，根据排中律，要么存在 $a : A$ 满足 $f(a) = b$，要么不存在。在第一种情况下，定义 $g(b) :\equiv a$；否则设定 $g(b) :\equiv a_0$。那么对于任何 $a : A$，我们有 $a = g(f(a))$，所以 $g$ 是满射。第二条是通过截断上的归纳得到的。对于第三条，如果 $g : B \to A$ 是满射，那么根据选择公理，仅仅存在一个函数 $f : A \to B$ 满足对所有的 $a$ 有 $g(f(a)) = a$。但此时 $f$ 必然是单射。 □

Theorem 10.2.10 (Schroeder–Bernstein (施罗德–贝尔斯坦定理)). 在假设排中律的情况下，对于集合 $A$ 和 $B$ 我们有：

$$\mathsf{inj}(A, B) \to \mathsf{inj}(B, A) \to (A \cong B)$$

证明. 常见的"来回反复 (back-and-forth)"论证在此仍然适用。注意，这实际上构造了一个同构 $A \cong B$（假设排中律，以便我们可以决定给定元素是属于循环、无限链、从 $A$ 开始的链，还是从 $B$ 开始的链）。 □

Corollary 10.2.11. 在假设排中律的情况下，基数不等式是一个偏序 (partial order)，即对于 $\alpha, \beta : \mathsf{Card}$ 我们有：

$$(\alpha \leq \beta) \to (\beta \leq \alpha) \to (\alpha = \beta).$$

证明. 由于 $\alpha = \beta$ 是一个纯命题，通过截断上的归纳，我们可以假设 $\alpha$ 和 $\beta$ 分别是 $|A|_0$ 和 $|B|_0$，并且我们有从 $A$ 到 $B$ 的单射 $f$ 和从 $B$ 到 $A$ 的单射 $g$。但根据施罗德–贝尔斯坦定理，我们得到了一个同构 $A \cong B$，因此得到了一个等式 $|A|_0 = |B|_0$。 □

最后，我们可以重现康托尔 (Cantor) 定理，证明对于每个基数，都存在一个更大的基数。

Theorem 10.2.12 (康托尔定理 (Cantor's theorem)). 对于 $A : \mathsf{Set}$，不存在满射 $A \to (A \to \mathbf{2})$。

证明. 假设 $f : A \to (A \to \mathbf{2})$ 是任意一个函数，定义 $g : A \to \mathbf{2}$ 为 $g(a) :\equiv \neg f(a)(a)$。如果 $g = f(a_0)$，那么 $g(a_0) = f(a_0)(a_0)$ 但 $g(a_0) = \neg f(a_0)(a_0)$，这就矛盾了。因此，$f$ 不是满射。 □

Corollary 10.2.13. 在假设排中律的情况下，对于任意 $\alpha : \mathsf{Card}$，存在一个基数 $\beta$ 使得 $\alpha \leq \beta$ 且 $\alpha \neq \beta$。

证明. 令 $\beta = 2^\alpha$。现在我们想要证明一个纯命题，所以通过归纳我们可以假设 $\alpha$ 是 $|A|_0$，因此 $\beta \equiv |A \to \mathbf{2}|_0$。在排中律的帮助下，我们定义了一个函数 $f : A \to (A \to \mathbf{2})$，定义为：

$$f(a)(a') :\equiv \begin{cases} 1_\mathbf{2} & a = a' \\ 0_\mathbf{2} & a \neq a'. \end{cases}$$

如果 $f(a) = f(a')$，那么 $f(a')(a) = f(a)(a) = 1_\mathbf{2}$，所以 $a = a'$；因此 $f$ 是单射。因此，$\alpha \equiv |A|_0 \leq |A \to \mathbf{2}|_0 \equiv 2^\alpha$。
另一方面，如果 $2^\alpha \leq \alpha$，那么我们将有一个单射 $(A \to \mathbf{2}) \to A$。根据 Lemma 10.2.9，由于我们有 $(\lambda x.\, 0_\mathbf{2}) : A \to \mathbf{2}$ 和排中律，那么就会有一个从 $A$ 到 $(A \to \mathbf{2})$ 的满射，这与康托尔定理相矛盾。 □

## 10.3 序数 (Ordinal numbers)

Definition 10.3.1. 设 $A$ 为一个集合，且

$$(- < -) : A \to A \to \mathsf{Prop}$$

是 $A$ 上的一个二元关系。我们通过归纳定义 $a : A$ 的元素对 $<$ 是 **可达** (accessible) 的：

- 如果对所有 $b < a$ 的 $b$ 都是可达的，则 $a$ 是可达的。

我们写作 $\mathsf{acc}(a)$ 表示 $a$ 是可达的。

乍一看，这样的归纳定义似乎无法成立，但如果 $a$ 具有这样一个性质，即不存在 $b$ 使得 $b < a$，那么 $a$ 是可达的，这实际上是成立的。
注意，这是一个类型族的归纳定义，类似于在 §5.7 中考虑的向量类型。更确切地说，它只有一个构造器，记为 $\mathsf{acc}_<$，其类型为

$$\mathsf{acc}_< : \prod_{a:A} \Big( \prod_{b:A} (b < a) \to \mathsf{acc}(b) \Big) \to \mathsf{acc}(a).$$

$\mathsf{acc}$ 的归纳原理表明，对于任意 $P : \prod_{(a:A)} \mathsf{acc}(a) \to \mathcal{U}$，如果我们有

$$f : \prod_{(a:A)} \prod_{(h:\prod_{(b:A)} (b<a) \to \mathsf{acc}(b))} \Big( \prod_{(b:A)} \prod_{(l:b<a)} P(b, h(b,l)) \Big) \to P(a, \mathsf{acc}_<(a,h)),$$

那么我们就有通过归纳定义的 $g : \prod_{(a:A)} \prod_{(c:\mathsf{acc}(a))} P(a,c)$，其中

$$g(a, \mathsf{acc}_<(a,h)) \equiv f(a,\, h,\, \lambda b.\, \lambda l.\, g(b, h(b,l))).$$

这个公式很繁琐，但通常我们只在 $P : A \to \mathcal{U}$ 只依赖于 $A$ 时使用它的简化形式。在这种情况下，$f$ 的第二和第三个参数可以合并，因此我们要证明的是

$$f : \prod_{a:A} \Big( \prod_{b:A} (b < a) \to \mathsf{acc}(b) \times P(b) \Big) \to P(a).$$

也就是说，我们假设每个 $b < a$ 都是可达的，且 $g(b) : P(b)$ 已定义，然后从这些定义 $g(a) : P(a)$。省略 $P$ 的第二个参数是通过以下引理来证明的，这也是我们唯一一次使用归纳原理的更一般形式。

Lemma 10.3.2. 可达性 (Accessibility) 是一个纯属性。

证明. 我们必须证明，对于任意 $a : A$ 和 $s_1, s_2 : \mathsf{acc}(a)$，有 $s_1 = s_2$。我们通过 $s_1$ 的归纳来证明这一点，其归纳假设为

$$P_1(a, s_1) :\equiv \prod_{s_2:\mathsf{acc}(a)} (s_1 = s_2)$$

因此，我们必须证明，对于任意 $a : A$ 和 $h_1 : \prod_{(b:A)} (b < a) \to \mathsf{acc}(b)$ 以及

$$k_1 : \prod_{(b:A)} \prod_{(l:b<a)} \prod_{(t:\mathsf{acc}(b))} h_1(b,l) = t,$$

我们有 $\mathsf{acc}_<(a,h) = s_2$ 对于任意 $s_2 : \mathsf{acc}(a)$。我们将这个陈述视为 $\prod_{(a:A)} \prod_{(s_2:\mathsf{acc}(a))} P_2(a, s_2)$，其中

$$P_2(a, s_2) :\equiv \prod_{(h_1:\cdots)} \prod_{(k_1:\cdots)} (\mathsf{acc}_<(a, h_1) = s_2);$$

因此，我们可以通过 $s_2$ 的归纳来证明它。因此，我们假设 $h_2 : \prod_{(b:A)} (b < a) \to \mathsf{acc}(b)$，并且 $k_2$ 具有一个复杂但无关紧要的类型，并且必须证明，对于任意类型为上述类型的 $h_1$ 和 $k_1$，我们有 $\mathsf{acc}_<(a, h_1) = \mathsf{acc}_<(a, h_2)$。根据函数外延性 (function extensionality)，只需证明 $h_1(b,l) = h_2(b,l)$ 对于所有 $b : A$ 和 $l : b < a$。这是由 $k_1$ 推出的。 □

Definition 10.3.3. 集合 $A$ 上的二元关系 $<$ 是 **良基** (well-founded) 的 如果 $A$ 的每个元素都是可达的。

良基性的意义在于，对于 $P : A \to \mathcal{U}$，我们可以使用 acc 的归纳原理得出 $\prod_{(a:A)} \mathsf{acc}(a) \to P(a)$，然后应用良基性得出 $\prod_{(a:A)} P(a)$。换句话说，如果从 $\forall (b : A).\, (b < a) \to P(b)$ 我们可以证明 $P(a)$，那么 $\forall (a : A).\, P(a)$ 成立。这称为 **良基归纳** (well-founded induction)。

Lemma 10.3.4. 良基性是一个纯属性。

证明. $<$ 的良基性是类型 $\prod_{(a:A)} \mathsf{acc}(a)$，它是一个纯命题，因为每个 $\mathsf{acc}(a)$ 是一个纯命题。 □

Example 10.3.5. 也许最为熟悉的良基关系是自然数 $\mathbb{N}$ 上的通常的严格排序。要证明这是良基的，我们必须证明对于每个 $n : \mathbb{N}$, $n$ 是可达的。这只是从 $\mathbb{N}$ 上的普通归纳得出的“强归纳”(strong induction) 的常规证明。
具体来说，我们通过对 $n : \mathbb{N}$ 的归纳证明 $k \le n$ 的所有 $k$ 都是可达的。基本情况只是 0 是可达的，这在逻辑上成立，因为没有任何元素严格小于 0。对于归纳步骤，我们假设 $k \le n$ 的所有 $k$ 都是可达的，这就是说，对于所有 $k < n+1$，因此根据定义 $n+1$ 也是可达的。
$\mathbb{N}$ 上的一个不同关系也可以是良基的，即设定仅 $n < \mathsf{succ}(n)$ 对于所有 $n : \mathbb{N}$。这个关系的良基性几乎正是 $\mathbb{N}$ 的普通归纳原理。

Example 10.3.6. 设 $A : \mathsf{Set}$ 且 $B : A \to \mathsf{Set}$ 是集合的一个族。回忆 §5.3 中的 $W$-类型 $\mathsf{W}_{(a:A)} B(a)$ 是由单个构造器归纳生成的

- $\mathsf{sup} : \prod_{(a:A)} (B(a) \to \mathsf{W}_{(x:A)} B(x)) \to \mathsf{W}_{(x:A)} B(x)$

我们通过其第二个参数递归地定义 $\mathsf{W}_{(x:A)}B(x)$ 上的关系 $<$：

- 对于任意 $a : A$ 和 $f : B(a) \to \mathsf{W}_{(x:A)}B(x)$，我们定义 $w < \mathsf{sup}(a, f)$ 表示仅存在一个 $b : B(a)$ 使得 $w = f(b)$。

现在我们使用 $\mathsf{W}_{(x:A)}B(x)$ 的通常归纳原理证明对该关系的每个 $w : \mathsf{W}_{(x:A)}B(x)$ 是可达的。这意味着我们假设给定了 $a : A$ 和 $f : B(a) \to \mathsf{W}_{(x:A)}B(x)$，以及一个提升 $f' : \prod_{(b:B(a))} \mathsf{acc}(f(b))$。但根据 $<$ 的定义，对于所有 $w < \mathsf{sup}(a, f)$，我们有 $\mathsf{acc}(w)$；因此 $\mathsf{sup}(a, f)$ 是可达的。

良基性允许我们通过递归定义函数，并通过归纳证明陈述，例如以下内容。回忆 §3.5 中 $\mathcal{P}(B)$ 表示幂集 $\mathcal{P}(B) :\equiv (B \to \mathsf{Prop})$。

Lemma 10.3.7. *假设 $B$ 是一个集合，并且我们有一个函数*

$$g : \mathcal{P}(B) \to B$$

*那么，如果 $<$ 是 $A$ 上的一个良基关系，那么存在一个函数 $f : A \to B$，使得对所有 $a : A$ 我们有*

$$f(a) = g\Big( \{\, f(a') \mid a' < a \,\} \Big)$$

（我们使用了 §10.1.2 中关于子集的像的记法。）

**证明.** 我们首先定义，对于每个 $a : A$ 和 $s : \mathsf{acc}(a)$，一个元素 $\bar{f}(a, s) : B$。根据归纳假设，我们假设 $s$ 是一个函数，将每个 $a' < a$ 分配给一个证据 $s(a') : \mathsf{acc}(a')$，并且对每个这样的 $a'$，我们有一个元素 $\bar{f}(a', s(a')) : B$。在这种情况下，我们定义

$$\bar{f}(a, s) :\equiv g\Big( \{\, \bar{f}(a', s(a')) \mid a' < a \,\} \Big)$$

现在，由于 $<$ 是良基的，我们有一个函数 $w : \prod_{(a:A)} \mathsf{acc}(a)$。因此，我们可以定义 $f(a) :\equiv \bar{f}(a, w(a))$。 □

在经典逻辑中，良基性有一个更为人所熟知的重新表述。在以下内容中，我们说一个子集 $B : \mathcal{P}(A)$ 是**非空的** (nonempty)，如果它不等于空子集 $(\lambda x.\, \bot) : \mathcal{P}(X)$。我们留给读者验证，在假设排中律的前提下，这与单纯的可居住性是等价的，即满足条件 $\exists(x : A).\, x \in B$。

Lemma 10.3.8. *假设排中律，$<$ 是良基的当且仅当每个非空子集 $B : \mathcal{P}(A)$ 都仅有一个最小元素。*

**证明.** 首先假设 $<$ 是良基的，并假设 $B \subseteq A$ 是一个没有最小元素的子集。也就是说，对于任何 $a : A$ 使得 $a \in B$，仅存在一个 $b : A$ 使得 $b < a$ 且 $b \in B$。

我们断言，对于任意 $a : A$ 和 $s : \mathsf{acc}(a)$，我们有 $a \notin B$。通过归纳，我们可以假设 $s$ 是一个函数，将每个 $a' < a$ 分配给一个证明 $s(a') : \mathsf{acc}(a')$，并且对每个这样的 $a'$，我们有 $a' \notin B$。如果 $a \in B$，那么根据假设，必然存在一个 $b < a$ 且 $b \in B$，这与假设矛盾。因此，$a \notin B$；这完成了归纳。由于 $<$ 是良基的，我们有 $a \notin B$ 对所有 $a : A$ 成立，即 $B$ 是空的。

现在假设每个非空子集都仅有一个最小元素。设 $B = \{\, a : A \mid \neg\mathsf{acc}(a) \,\}$。那么，如果 $B$ 是非空的，它仅有一个最小元素。因此，仅存在一个 $a : A$ 使得 $a \in B$，并且对所有 $b < a$，我们有 $\mathsf{acc}(b)$。但是根据定义（以及对截断的归纳），$a$ 是仅可达的，因此是可达的，这与 $a \in B$ 矛盾。因此，$B$ 是空的，所以 $<$ 是良基的。 □

Definition 10.3.9. 集合 $A$ 上的良基关系 $<$ 是**外延** (extensional) 的 如果对于任意 $a, b : A$，我们有

$$\Big( \forall(c : A).\, (c < a) \Leftrightarrow (c < b) \Big) \to (a = b)$$

注意，由于 $A$ 是一个集合，外延性是一个纯属性。这种“外延性”的概念与函数外延性 (function extensionality) 无关，也与等式类型的外延性无关。相反，它是经典集合论中外延公理的“局部”对应。

Theorem 10.3.10. *外延良基关系的类型是一个集合。*

证明. 根据公理 (univalence axiom)，如果 $(A,<)$ 是外延和良基的，并且 $f:(A,<)\cong(A,<)$，那么我们必须证明 $f=\mathsf{id}_A$。我们通过对 $<$ 进行归纳来证明对于所有 $a:A$，$f(a)=a$。归纳假设是，对于所有 $a'<a$，我们有 $f(a')=a'$。
现在，由于 $A$ 是外延的，为了得出 $f(a)=a$，我们只需证明

$$\forall(c:A).\,(c<f(a))\Leftrightarrow(c<a)$$

但是，由于 $f$ 是一个自同构，我们有 $(c<a)\Leftrightarrow(f(c)<f(a))$。但 $c<a$ 意味着根据归纳假设 $f(c)=c$，因此 $(c<a)\to(c<f(a))$。另一方面，如果 $c<f(a)$，那么 $f^{-1}(c)<a$，因此 $c=f(f^{-1}(c))=f^{-1}(c)$ 再次根据归纳假设；因此 $c<a$。因此，我们有 $(c<a)(c<f(a))$ 对于任意 $c:A$，所以 $f(a)=a$。 □

Definition 10.3.11. 如果 $(A,<)$ 和 $(B,<)$ 是外延良基的关系，$f:A\to B$ 是一个**模拟** (simulation) 如果满足以下条件：

(i) 如果 $a<a'$，那么 $f(a)<f(a')$，并且
(ii) 对于所有 $a:A$ 和 $b:B$，如果 $b<f(a)$，那么仅存在一个 $a'<a$ 使得 $f(a')=b$。

Lemma 10.3.12. 任何模拟都是单射的。

证明. 我们通过双重良基归纳证明，对于任意 $a,b:A$，如果 $f(a)=f(b)$，那么 $a=b$。归纳假设对于 $a:A$ 表示，对于任意 $a'<a$ 和任意 $b:B$，如果 $f(a')=f(b)$，那么 $a=b$。内部归纳假设对于 $b:A$ 表示，对于任意 $b'<b$，如果 $f(a)=f(b')$，那么 $a=b'$。
假设 $f(a)=f(b)$；我们必须证明 $a=b$。根据外延性，为了证明 $f(a)=b$，我们只需证明 $\forall(c:A).\,(c<a)\Leftrightarrow(c<b)$。
但是，由于 $f$ 是一个模拟，如果 $c<a$，那么我们有 $f(c)<f(a)$ 根据 Definition 10.3.11(i)。因此 $f(c)<f(b)$，所以根据 Definition 10.3.11(ii)，仅存在一个 $c':A$ 使得 $c'<b$ 且 $f(c)=f(c')$。根据归纳假设 $a$，我们有 $c=c'$，因此 $c<b$。对称的论证也是对称的。 □

特别地，这意味着在 Definition 10.3.11(ii) 中可以去掉“仅”一词而不改变意义。

Corollary 10.3.13. 如果 $f:A\to B$ 是一个模拟，那么对于所有 $a:A$ 和 $b:B$，如果 $b<f(a)$，那么必然存在一个 $a'<a$ 使得 $f(a')=b$。

证明. 由于 $f$ 是单射的，$\sum_{(a:A)}(f(a)=b)$ 是一个纯命题。 □

我们说一个子集 $C:\mathcal{P}(B)$ 是**初始段** (initial segment) ，如果 $c\in C$ 并且 $b<c$，那么 $b\in C$。模拟的像必须是一个初始段，而任何初始段的包含是一个模拟。因此，根据公理 (univalence axiom)，每个 $A\to B$ 的模拟等同于 $B$ 的某个初始段的包含。

Theorem 10.3.14. 对于一个集合 $A$，设 $P(A)$ 为 $A$ 上的外延良基关系的类型。如果 $<_A:P(A)$ 和 $<_B:P(B)$ 并且 $f:A\to B$，令 $H_{<_A<_B}(f)$ 是 $f$ 是一个模拟的纯命题。那么 $(P,H)$ 是一个标准的结构概念，属于 $\mathcal{S}et$ 中的结构，见 §9.8。

证明. 我们留给读者验证，恒等是模拟，并且模拟的复合也是模拟。因此，我们有了一个结构的概念。对于标准性，我们必须证明如果 $<$ 和 $\prec$ 是两个集合 $A$ 上的外延良基关系，并且 $\mathsf{id}_A$ 是双向模拟，那么 $<$ 和 $\prec$ 是相等的。由于外延性和良基性是纯命题，对于这一点我们仅需有 $\forall(a,b:A).\,(a<b)\Leftrightarrow(a\prec b)$。但这由 Definition 10.3.11(i) 对于 $\mathsf{id}_A$ 来证明。 □

Corollary 10.3.15. 有一个范畴，其对象是带有外延良基关系的集合，其态射是模拟。

实际上，这个范畴是一个偏序集。

Lemma 10.3.16. 对于外延和良基的 $(A,<)$ 和 $(B,<)$，至多存在一个从 $A$ 到 $B$ 的模拟。

证明. 假设 $f, g : A \to B$ 是模拟。由于是模拟是一个纯命题，我们只需证明 $\forall(a : A).\,(f(a) = g(a))$。通过对 $<$ 进行归纳，我们可以假设对于所有 $a' < a$，$f(a') = g(a')$。由于 $B$ 的外延性，为了证明 $f(a) = g(a)$，我们只需有 $\forall(b : B).\,(b < f(a)) \Leftrightarrow (b < g(a))$。
但由于 $f$ 是一个模拟，如果 $b < f(a)$，那么我们有 $a' < a$ 使得 $f(a') = b$。根据归纳假设，我们也有 $g(a') = b$，因此 $b < g(a)$。对称的论证也是对称的。 □

因此，如果 $A$ 和 $B$ 都装备有外延和良基的关系，我们可以写作 $A \leq B$ 表示存在一个从 $A$ 到 $B$ 的模拟。Corollary 10.3.15 意味着如果 $A \leq B$ 并且 $B \leq A$，那么 $A = B$。

Definition 10.3.17. 一个**序数** (ordinal) 是一个集合 $A$，具有一个外延良基关系，并且该关系是*传递的*，即满足 $\forall(a, b, c : A).\,(a < b) \to (b < c) \to (a < c)$。

Example 10.3.18. 当然，$\mathbb{N}$ 上的通常严格序是传递的。它也容易看出是外延的；因此它是一个序数。如常所示，我们用 $\omega$ 来表示这个序数。
令 $\mathsf{Ord}$ 表示序数的类型。根据之前的结果，$\mathsf{Ord}$ 是一个集合，并且具有一个自然的偏序关系。我们现在证明 $\mathsf{Ord}$ 还承认一个良基关系。
如果 $A$ 是一个序数，并且 $a : A$，令 $A_{/a} :\equiv \{\, b : A \mid b < a \,\}$ 表示初始段。注意，如果 $A_{/a} = A_{/b}$ 作为序数，那么该同构必然尊重它们在 $A$ 中的包含（因为模拟构成一个偏序集），因此它们作为 $A$ 的子集是相等的。因此，由于 $A$ 是外延的，$a = b$。因此函数 $a \mapsto A_{/a}$ 是一个从 $A$ 到 $\mathsf{Ord}$ 的单射。

Definition 10.3.19. 对于序数 $A$ 和 $B$，一个模拟 $f : A \to B$ 称为**有界的** (bounded) 如果存在 $b : B$ 使得 $A = B_{/b}$。

上述评论意味着当它存在时，这样的 $b$ 是唯一的，因此有界性是一个纯命题。
我们写作 $A < B$ 表示存在一个从 $A$ 到 $B$ 的有界模拟。由于模拟是唯一的，$A < B$ 也是一个纯命题。

Theorem 10.3.20. $(\mathsf{Ord}, <)$ *是一个序数*。

更精确地说，这个定理表明，序数类型 $\mathsf{Ord}_{\mathcal{U}_i}(\mathsf{Ord}_{\mathcal{U}_i}, <) : \mathsf{Ord}_{\mathcal{U}_{i+1}}$。

证明. 设 $A$ 是一个序数；我们首先证明对于所有 $a : A$，$A_{/a}$ 是可达的（在 $\mathsf{Ord}$ 中）。通过对 $A$ 进行良基归纳，假设对于所有 $b < a$，$A_{/b}$ 是可达的。根据可达性的定义，我们必须证明对于所有 $B < A_{/a}$，$B$ 是可达的。但是，如果 $B < A_{/a}$，那么存在一些 $b < a$ 使得 $B = (A_{/a})_{/b} = A_{/b}$，这是通过归纳假设可达的。因此，$A_{/a}$ 对所有 $a : A$ 是可达的。
现在为了证明 $A$ 在 $\mathsf{Ord}$ 中是可达的，根据定义，我们必须证明对于所有 $B < A$，$B$ 是可达的。但如前所述，$B < A$ 意味着 $B = A_{/a}$ 对于某个 $a : A$，它是我们刚刚证明的可达的。因此，$\mathsf{Ord}$ 是良基的。
对于外延性，假设 $A$ 和 $B$ 是序数，并且 $\prod_{(C:\mathsf{Ord})}(C < A) \Leftrightarrow (C < B)$. 那么对于每个 $a : A$，由于 $A_{/a} < A$，我们有 $A_{/a} < B$，因此存在 $b : B$ 使得 $A_{/a} = B_{/b}$。定义 $f : A \to B$ 将每个 $a$ 映射到相应的 $b$；很容易验证 $f$ 是一个同构。因此 $A \cong B$，因此根据公理 $A = B$。
最后，很容易看出 $<$ 是传递的。 □

将 $\mathsf{Ord}$ 视为一个序数通常非常方便，但它也有一些陷阱。例如，考虑下面的引理，我们注意到如何使用宇宙。

Lemma 10.3.21. *令 $\mathcal{U}$ 为一个宇宙。对于任何 $A : \mathsf{Ord}_{\mathcal{U}}$，存在一个 $B : \mathsf{Ord}_{\mathcal{U}}$ 使得 $A < B$*。

证明. 令 $B = A + \mathbf{1}$，其中元素 $\star : \mathbf{1}$ 大于 $A$ 的所有元素。那么 $B$ 是一个序数，并且很容易看出 $A \cong B_{/\star}$。 □

在 Lemma 10.3.21 的证明中构造的序数 $B$ 被称为 $A$ 的**后继** (successor)。
这个引理说明了在使用“典型含糊 (typically ambiguous)”的风格来使用 $\mathcal{U}$ 表示一个任意、未指定的宇宙时可能出现的陷阱。考虑下面的另一种证明方法。

*另一个* Lemma 10.3.21 *的证明*. 注意 $C < A$ 当且仅当 $C = A_{/a}$ 对于某个 $a : A$。这给出了一个同构 $A \cong \mathsf{Ord}_{/A}$，因此 $A < \mathsf{Ord}$。因此我们可以取 $B :\equiv \mathsf{Ord}$。 □

第二种证明方法在我们以典型含糊的风格陈述 Lemma 10.3.21 时是有效的。但是得到的引理将不那么有用，因为第二个证明将限制引理陈述中的第二个“Ord ”与第一个相比指的是更高的宇宙级别。第一个证明允许两个宇宙是相同的。
类似的评论适用于下一个引理，它可以通过观察到对于任何 $A : \mathsf{Ord}$，$A \leq \mathsf{Ord}$ 来证明。

Lemma 10.3.22. *令 $\mathcal{U}$ 为一个宇宙。对于任何 $X : \mathcal{U}$ 和 $F : X \to \mathsf{Ord}_{\mathcal{U}}$，存在一个 $B : \mathsf{Ord}_{\mathcal{U}}$ 使得 $Fx \leq B$ 对于所有 $x : X$。*

*证明.* 令 $B$ 为等价关系 $\sim$ 在 $\sum_{(x:X)} Fx$ 上的商集（见 Remark 6.10.1）定义如下：

$$(x, y) \sim (x', y') :\equiv \Big( (Fx)_{/y} \cong (Fx')_{/y'} \Big).$$

定义 $(x, y) < (x', y')$ 如果 $(Fx)_{/y} < (Fx')_{/y'}$。这显然传递到商集，并且可以看出使得 $B$ 成为一个序数。此外，对于每个 $x : X$，诱导的映射 $Fx \to B$ 是一个模拟。 □

## 10.4 经典良序 (Classical well-orderings)

现在我们来展示我们定义的序数 (ordinal) 与更为熟悉的经典良序 (classical well-orderings) 之间的等价性。

Lemma 10.4.1. *在假设排中律的情况下，每一个序数 (ordinal) 都是三分性的 (trichotomous)：*

$$\forall (a, b : A).\ (a < b) \vee (a = b) \vee (b < a).$$

*证明.* 通过对 $a$ 进行归纳，我们可以假设对于每一个 $a' < a$ 和每一个 $b' : A$，我们有 $(a' < b') \vee (a' = b') \vee (b' < a')$。现在通过对 $b$ 进行归纳，我们可以假设对于每一个 $b' < b$，我们有 $(a < b') \vee (a = b') \vee (b' < a)$。
根据排中律，存在一个 $b' < b$ 使得 $a < b'$，或者存在一个 $b' < b$ 使得 $a = b'$，或者对于所有的 $b' < b$ 我们有 $b' < a$。在第一种情况下，仅 $a < b$ 通过传递性成立，因此 $a < b$ 作为一个命题成立。同样地，在第二种情况下，$a < b$ 通过传递成立。因此，假设 $\forall (b' : A).\ (b' < b) \to (b' < a)$。
现在类似地，要么存在 $a' < a$ 使得 $b < a'$，要么存在 $a' < a$ 使得 $a' = b$，要么对于每个 $a' < a$ 我们有 $a' < b$。在第一种和第二种情况下，$b < a$，因此我们可以假设 $\forall (a' : A).\ (a' < a) \to (a' < b)$。然而，通过外延性 (extensionality)，我们的两个假设现在意味着 $a = b$。 □

Lemma 10.4.2. *一个良基关系 (well-founded relation) 不包含循环，也就是说*

$$\forall (n : \mathbb{N}).\ \forall (a : \mathbb{N}_n \to A).\ \neg \Big( (a_0 < a_1) \wedge \cdots \wedge (a_{n-1} < a_n) \wedge (a_n < a_0) \Big).$$

*证明.* 我们通过对 $a : A$ 进行归纳证明不存在包含 $a$ 的循环。因此，假设通过归纳，对于所有的 $a' < a$，不存在包含 $a'$ 的循环。但是在任何包含 $a$ 的循环中，都存在一些元素小于 $a$ 并包含在相同的循环中。 □

特别是，一个良基关系必须是**反自反的** (irreflexive)，即对于所有的 $a$，$\neg (a < a)$。

Theorem 10.4.3. *在假设排中律的情况下，$(A, <)$ 是一个序数 (ordinal) 当且仅当每个非空子集 $B \subseteq A$ 都有一个最小元素。*

*证明.* 如果 $A$ 是一个序数，那么根据 Lemma 10.3.8，每个非空子集仅存在一个最小元素。但是三分性 (trichotomy) 意味着任何最小元素都是一个最小元素。此外，最小元素存在时是唯一的，因此仅有一个最小元素等同于存在一个最小元素。
反之，如果每个非空子集都有一个最小元素，那么根据 Lemma 10.3.8，$A$ 是良基的。我们也有三分性，因为对于任何 $a, b$，子集 $\{ a, b \} :\equiv \{ x : A \mid x = a \vee x = b \}$ 仅有一个最小元素，这个元素必定是 $a$ 或者 $b$。这意味着传递性 (transitivity)，因为如果 $a < b$ 并且 $b < c$，那么 $a = c$ 或者 $c < a$ 将产生一个循环。类似地，它也意味着外延性，因为如果 $\forall (c : A).\ (c < a) \Leftrightarrow (c < b)$，那么 $a < b$ 意味着 (取 $c$ 为 $a$) $a < a$，这将产生一个循环，反之亦然；因此 $a = b$。 □

在经典数学中，Theorem 10.4.3 中的特征被视为*良序* (well-ordering) 的定义，其中*序数* (ordinals) 是同构类的良序的一个典型代表。在我们的语境中，结构同一性原理 (structure identity principle) 意味着不需要寻找这样的代表：任何一个良序与另一个都是同样好的。
我们现在继续讨论选择公理 (axiom of choice) 的后果。对于任何集合 $X$，让 $\mathcal{P}_+(X)$ 表示 $X$ 的仅有居住的子集类型：

$$\mathcal{P}_+(X) :\equiv \{\, Y : \mathcal{P}(X) \mid \exists (x : X).\, x \in Y \,\}.$$

在假设排中律的情况下，这等价于 $X$ 的*非空* (nonempty) 子集的类型，并且我们有 $\mathcal{P}(X) \simeq (\mathcal{P}_+(X)) + \mathbf{1}$。

Theorem 10.4.4. *在假设排中律的情况下，以下条件等价：*

(i) *对于每个集合 $X$，仅存在一个函数 $f : \mathcal{P}_+(X) \to X$ 使得对于所有 $Y : \mathcal{P}_+(X)$ 有 $f(Y) \in Y$。*
(ii) *每个集合仅具有序数结构。*

当然，(i) 是选择公理的一种标准经典版本；参见 Exercise 10.10。

*证明.* 一个方向是容易的：假设 (ii)。因为我们要证明仅作为命题的 (i)，我们可以假设 $A$ 是一个序数。但随后我们可以定义 $f(B)$ 为 $B$ 的最小元素。
现在假设 (i)。同样，由于 (ii) 是一个命题，我们可以假设给定这样的一个 $f$。我们将 $f$ 扩展为一个函数

$$\bar{f} : \mathcal{P}(X) \simeq (\mathcal{P}_+(X)) + \mathbf{1} \longrightarrow X + \mathbf{1}$$

以显而易见的方式。现在对于任何一个序数 $A$，我们可以通过良基递归定义 $g_A : A \to X + \mathbf{1}$：

$$g_A(a) :\equiv \bar{f}\Big(X \setminus \{\, g_A(b) \mid (b < a) \wedge (g_A(b) \in X) \,\}\Big)$$

(将 $X$ 视为 $X + \mathbf{1}$ 的一个子集显然)。
设 $A' :\equiv \{\, a : A \mid g_A(a) \in X \,\}$ 为 $g_A$ 的预像；然后我们声明 $g'_A : A' \to X$ 是单射 (injective)。因为如果 $a, a' : A$ 且 $a \neq a'$，那么根据三分性，我们可以假设 $a' < a$。因此 $g_A(a') \in \{\, g_A(b) \mid b < a \,\}$，所以由于对于所有 $Y$ 有 $f(Y) \in Y$，我们有 $g_A(a) \neq g_A(a')$。
此外，$A'$ 是 $A$ 的一个初始片段 (initial segment)。因为 $g_A(a)$ 落在 $\mathbf{1}$ 中当且仅当 $\{\, g_A(b) \mid b < a \,\} = X$，并且如果这是成立的，那么对于任何 $a' > a$ 这也是成立的。因此，$A'$ 本身是一个序数。
最后，由于 $\mathsf{Ord}$ 是一个序数，我们可以取 $A :\equiv \mathsf{Ord}$。设 $X'$ 为 $g'_{\mathsf{Ord}} : \mathsf{Ord}' \to X$ 的像；然后 $g'_{\mathsf{Ord}}$ 的逆函数给出了一个单射 $H : X' \to \mathsf{Ord}$。根据 Lemma 10.3.22，存在一个序数 $C$ 使得对于所有 $x : X'$，$Hx \leq C$。然后根据 Lemma 10.3.21，存在一个更大的序数 $D$ 使得 $C < D$，因此对于所有 $x : X'$，$Hx < D$。现在我们有

$$\begin{aligned} g_{\mathsf{Ord}}(D) &= \bar{f}\Big(X \setminus \Big\{\, g_{\mathsf{Ord}}(B) \;\Big|\; B < D \wedge (g_{\mathsf{Ord}}(B) \in X) \,\Big\}\Big) \\ &= \bar{f}\Big(X \setminus \Big\{\, g_{\mathsf{Ord}}(B) \;\Big|\; g_{\mathsf{Ord}}(B) \in X \,\Big\}\Big) \end{aligned}$$

因为如果 $B : \mathsf{Ord}$ 并且 $(g_{\mathsf{Ord}}(B) \in X)$，那么 $B = Hx$ 对于某些 $x : X'$，因此 $B < D$。现在如果

$$\Big\{\, g_{\mathsf{Ord}}(B) \;\Big|\; g_{\mathsf{Ord}}(B) \in X \,\Big\}$$

不是全部的 $X$，那么 $g_{\mathsf{Ord}}(D)$ 将在 $X$ 中但不在这个子集中，这将与 $D$ 自己作为 $B$ 的一个可能值相矛盾。所以这个集合必须是全部的 $X$，因此 $g'_{\mathsf{Ord}}$ 是满射 (surjective) 的同时也是单射的。因此，我们可以将 $\mathsf{Ord}$' 上的序数结构传递给 $X$。 □

Remark 10.4.5. 如果我们使用了错误的 Lemma 10.3.21 或 Lemma 10.3.22 的证明，那么 Theorem 10.4.4 的结果证明将是无效的：将无法一致地分配宇宙级别。如现在所示，我们需要命题调整 (propositional resizing)（这由 $\mathsf{LEM}$ 导出）来确保 $X'$ 处于与 $X$ 相同的宇宙中（直到等价）。

Corollary 10.4.6. *在假设选择公理的情况下，函数* $\mathsf{Ord} \to \mathsf{Set}$（*它忽略了序结构*）*是一个满射。*

注意，Ord 是一个集合，而 Set 是一个 1 类型 (1-type)。一般来说，没有理由让一个 1 类型接受来自集合的任何满射。即使选择公理似乎也不意味着每一个 1 类型都如此（尽管参见 Exercise 7.9），但它很容易意味着对从 Set 构造出的 1 类型（例如在 §9.8 中的结构的类别对象类型）是如此。以下的推论也适用于这样的类别。

Corollary 10.4.7. *在假设* AC *的情况下，*$\mathcal{S}et$ *接受来自严格类别的弱等价函子。*

*证明.* 设 $X_0 :\equiv \mathsf{Ord}$，并且对于 $A, B : X_0$，设 $\hom_X(A,B) :\equiv (A \to B)$。那么 $X$ 是一个严格类别，因为 Ord 是一个集合，并且上述满射 $X_0 \to \mathsf{Set}$ 扩展为一个弱等价函子 $X \to \mathcal{S}et$。 □

现在回顾 §10.2，我们有进一步的满射 $|-|_0 : \mathsf{Set} \to \mathsf{Card}$，因此有一个复合满射 $\mathsf{Ord} \to \mathsf{Card}$ 将每个序数映射到其基数 (cardinality)。

Theorem 10.4.8. *在假设* AC *的情况下，满射* $\mathsf{Ord} \to \mathsf{Card}$ *有一个截面* (section)。

*证明.* 有一个简单且错误的证明：因为 Ord 和 Card 都是集合，AC 意味着任何满射之间仅有一个截面。然而，我们实际上有一个规范的指定截面：因为 Ord 是一个序数，每个非空子集都有一个唯一指定的最小元素。因此，我们可以将每个基数映射到对应纤维中的最小元素。 □

在集合论中，传统上将基数与其在 Ord 中的像相识别：具有该基数的最小序数。
这意味着 Card 也可以规范地接受一个序数结构：实际上，它与 Ord 同构。具体来说，我们通过良基递归定义了一个函数 $\aleph : \mathsf{Ord} \to \mathsf{Ord}$，使得 $\aleph(A)$ 是具有大于 $\aleph(A_{/a})$ 的基数的最小序数，对于所有 $a : A$。然后（假设 AC），$\aleph$ 的像正好是 Card 的像。

## 10.5 累积层次 (The cumulative hierarchy)

我们可以将给定宇宙 $\mathcal{U}$ 中的所有集合的累积层次 $V$ 定义为一个高阶归纳类型 (higher inductive type)，使得 $V$ 再次是一个集合（在更大的宇宙 $\mathcal{U}'$ 中），配备一个二元"成员"关系 $x \in y$，它满足集合论的通常定律。

Definition 10.5.1. 相对于类型宇宙 $\mathcal{U}$ 的**累积层次** (cumulative hierarchy) $V$ 是由以下构造函数生成的高阶归纳类型：

(i) 对于每个 $A : \mathcal{U}$ 和 $f : A \to V$，有一个元素 $\mathsf{set}(A,f) : V$。
(ii) 对于所有 $A, B : \mathcal{U}$，$f : A \to V$ 和 $g : B \to V$，满足

$$\big(\forall(a:A).\,\exists(b:B).\,f(a) =_V g(b)\big) \wedge \big(\forall(b:B).\,\exists(a:A).\,f(a) =_V g(b)\big) \tag{10.5.2}$$

存在一个路径 $\mathsf{set}(A,f) =_V \mathsf{set}(B,g)$。
(iii) 0-截断构造函数：对于所有 $x, y : V$ 和 $p, q : x = y$，我们有 $p = q$。

在集合论的语言中，$\mathsf{set}(A,f)$ 可以理解为（经典集合论意义上的）集合，它是 $A$ 在 $f$ 下的像，即 $\{\, f(a) \mid a \in A \,\}$。然而，我们将避免这种表示法，因为它会与我们的子类型 (subtypes) 表示法发生冲突（但参见 (10.5.3) 和 Definition 10.5.7）。
层次 $V$ 是从空映射 $\mathsf{rec}_{\mathbf{0}}(V) : \mathbf{0} \to V$ 启动的，它给出了空集 $\emptyset = \mathsf{set}(\mathbf{0}, \mathsf{rec}_{\mathbf{0}}(V))$。然后通过 $\mathbf{1} \to V$ 的定义 $\star \mapsto \emptyset$ 进入 $V$，以此类推。（该定义也可以调整为包括任意的"原子"(atoms) 或"无元素" (urelements) 集合，通过添加额外的点构造函数。）类型 $V$ 存在于与基础宇宙 $\mathcal{U}$ 相同的宇宙中。
$V$ 的第二个构造函数的形式与我们以前见过的任何东西都不同：它不仅涉及 $V$ 中的路径（在 §6.9 中我们声称这些路径稍有问题），而且还涉及它们的和的截断。显然，它不符合在 §6.13 中描述的一般方案，因此其归纳原理是什么可能并不明显。幸运的是，就像我们在 §6.9 中对 0-截断的第一次定义一样，它可以通过辅助高阶归纳类型重新表达。我们将细节留给读者去思考（参见 Exercise 10.11）。
最后，$V$ 的归纳原理（用模式匹配语言书写）表明，给定 $P : V \to \mathsf{Set}$，为了构造 $h : \prod_{(x:V)} P(x)$，只需给出以下内容即可。

(i) 对于任意 $f : A \to V$，构造 $h(\mathsf{set}(A,f))$，假设给定了所有 $a : A$ 的 $h(f(a))$。

(ii) 验证如果 $f : A \to V$ 和 $g : B \to V$ 满足 (10.5.2)，则 $h(\mathsf{set}(A,f)) =^P_q h(\mathsf{set}(B,g))$，其中 $q$ 是从 $V$ 的第二个构造函数和 (10.5.2) 中生成的路径，假设 $h(f(a))$ 和 $h(g(b))$ 已经为所有 $a : A$ 和 $b : B$ 定义，并且满足以下条件：

$$\begin{aligned} &\big(\forall(a:A).\,\exists(b:B).\,\exists(p : f(a) = g(b)).\,h(f(a)) =^P_p h(g(b))\big) \\ \wedge\quad &\big(\forall(b:B).\,\exists(a:A).\,\exists(p : f(a) = g(b)).\,h(f(a)) =^P_p h(g(b))\big) \end{aligned}$$

第二个条款检查正在定义的映射是否尊重在 (10.5.2) 中引入的路径。像往常一样，当我们使用模式匹配来陈述高阶归纳原理时，它可能看起来像是一种同义反复，但实际上并非如此。关键在于"$h(f(a))$"本质上是一个形式符号，我们不能窥视其内部，$h(\mathsf{set}(A,f))$ 必须根据其定义。因此，在第二个条款中，我们在适当的时候假设这些形式符号的等价性，并验证由第一条款的构造产生的元素是否也相等。当然，如果 $P$ 是一个命题族，那么第二条款是自动成立的。

注意，通过归纳，对于每个 $v : V$，仅存在 $A : \mathcal{U}$ 和 $f : A \to V$ 使得 $v = \mathsf{set}(A,f)$。因此，定义 $V$ 上的**成员关系** (membership relation) $x \in v$ 是合理的：

$$(x \in \mathsf{set}(A,f)) :\equiv (\exists(a : A).\,x = f(a)).$$

要证明该定义是有效的，我们必须使用 $V$ 的递归原理。因此，假设我们有一个通过 (10.5.2) 构造的路径 $\mathsf{set}(A,f) = \mathsf{set}(B,g)$。如果 $x \in \mathsf{set}(A,f)$，则仅有一个 $a : A$ 使得 $x = f(a)$，但是根据 (10.5.2) 仅有一个 $b : B$ 使得 $f(a) = g(b)$，因此 $x = g(b)$ 并且 $x \in \mathsf{set}(B,g)$。反之亦然。

**子集关系** (subset relation) $x \subseteq y$ 在 $V$ 上的定义与通常一样

$$(x \subseteq y) :\equiv \forall(z : V).\,z \in x \Rightarrow z \in y.$$

**类** (class) 可以被认为是 $V$ 上的一个命题谓词。我们可以说一个类 $C : V \to \mathsf{Prop}$ 是一个 $V$-**集合** ($V$-set)，如果仅存在 $v : V$ 使得

$$\forall(x : V).\,C(x)x \in v.$$

我们还可以使用类的常规表示法，它与我们的标准子类型 (subtypes) 表示法相匹配：

$$\{\, x \mid C(x) \,\} :\equiv \lambda x.\,C(x). \tag{10.5.3}$$

一个类 $C : V \to \mathsf{Prop}$ 将被称为 $\mathcal{U}$-**小** ($\mathcal{U}$-small)，如果其所有值 $C(x)$ 都处于 $\mathcal{U}$ 中，特别是 $C : V \to \mathsf{Prop}_\mathcal{U}$。由于 $V$ 存在于与基础宇宙 $\mathcal{U}$ 相同的宇宙 $\mathcal{U}'$ 中，因此对于任何 $v, w : V$，它们的同一性类型 $v =_V w$ 也是如此。为了在没有命题调整的情况下获得良好表现，我们可以定义一个 $\mathcal{U}$-小的"调整"身份关系，我们可以通过归纳如下定义。

Definition 10.5.4. 定义**双模拟** (bisimulation)关系

$$\sim : V \times V \longrightarrow \mathsf{Prop}_\mathcal{U}$$

通过在 $V$ 上进行双重归纳，其中对于 $\mathsf{set}(A,f)$ 和 $\mathsf{set}(B,g)$ 我们有：

$$\mathsf{set}(A,f) \sim \mathsf{set}(B,g) :\equiv \big(\forall(a:A).\,\exists(b:B).\,f(a) \sim g(b)\big) \wedge \big(\forall(b:B).\,\exists(a:A).\,f(a) \sim g(b)\big).$$

要验证该定义的正确性，我们只需检查它是否尊重通过 (10.5.2) 构造的路径，但这是显而易见的，并且 $\mathsf{Prop}_\mathcal{U}$ 是一个集合，这也是显而易见的。请注意，$u \sim v$ 是在构造中处于 $\mathsf{Prop}_\mathcal{U}$ 的。

Lemma 10.5.5. *对于任意 $u, v : V$ 我们有 $(u =_V v) = (u \sim v)$。*

*证明.* 一个简单的归纳法表明 $\sim$ 是自反的，因此通过传递性，我们有 $(u =_V v) \to (u \sim v)$。因此，剩下的要证明的是 $(u \sim v) \to (u =_V v)$。通过对 $u$ 和 $v$ 的归纳，我们可以假设它们分别是 $\mathsf{set}(A,f)$ 和 $\mathsf{set}(B,g)$。（我们可以忽略 $V$ 的路径构造子，因为 $(u \sim v) \to (u =_V v)$ 是一个单纯命题。）然后根据定义，$\mathsf{set}(A,f) \sim \mathsf{set}(B,g)$ 意味着 $(\forall(a : A).\,\exists(b : B).\,f(a) \sim g(b))$，反之亦然。但是根据归纳假设，我们可以得出 $(\forall(a : A).\,\exists(b : B).\,f(a) = g(b))$，反之亦然。所以根据 $V$ 的路径构造子，我们有 $\mathsf{set}(A,f) =_V \mathsf{set}(B,g)$。 □

有人可能会认为我们可以省略 $V$ 的 0-截断构造子，并通过将 Theorem 7.2.2 应用于仿真来证明 $V$ 是 0-截断的。然而，在 Lemma 10.5.5 的证明中，我们使用了 $V$ 是 0-截断的事实，得出 $(u \sim v) \to (u =_V v)$ 是一个单纯命题，因此在归纳中我们可以假设 $u$ 和 $v$ 是 $\mathsf{set}(A,f)$ 和 $\mathsf{set}(B,g)$。
现在我们可以使用调整大小的恒等关系来得到以下有用的原理。

Lemma 10.5.6. *对于每个 $u : V$，存在一个给定的 $A_u : \mathcal{U}$ 和单射 $m_u : A_u \rightarrowtail V$，使得 $u = \mathsf{set}(A_u, m_u)$。*

*证明.* 取任意表示 $u = \mathsf{set}(A,f)$ 并将 $f : A \to V$ 分解为一个满射后跟一个单射：

$$f = m_u \circ e_u : A \twoheadrightarrow A_u \rightarrowtail V.$$

显然，如果 $A_u$ 仍在 $\mathcal{U}$ 中，则 $u = \mathsf{set}(A_u, m_u)$ 成立，这在 $e_u : A \twoheadrightarrow A_u$ 的核在 $\mathcal{U}$ 中时成立。但是 $e_u : A \twoheadrightarrow A_u$ 的核是 $f : A \to V$ 上的恒等的拉回，我们刚刚证明它在等价的意义下是 $\mathcal{U}$-小的。现在，从 $u : V$ 构造出 $m_u : A_u \rightarrowtail V$ 和 $u = \mathsf{set}(A_u, m_u)$ 的对 $(A_u, m_u)$ 是在 $V$ 上等价的唯一的，因此通过单一选择原理 (3.9.2)，存在一个映射 $c : V \to \sum_{(A:\mathcal{U})}(A \to V)$，使得 $c(u) = (A_u, m_u)$，其中 $m_u : A_u \rightarrowtail V$ 并且 $u = \mathsf{set}(c(u))$，如所述。 □

Definition 10.5.7. 对于 $u : V$，刚刚构造的单射表示 $m_u : A_u \rightarrowtail V$ 使得 $u = \mathsf{set}(A_u, m_u)$ 可以称为 $u$ 的**成员类型** (type of members)，记为 $m_u : [u] \rightarrowtail V$，或者简写为 $[u] \rightarrowtail V$。我们可以将 $[u]$ 视为“由 $u$ 的成员组成的 $V$ 的子类”。

Theorem 10.5.8. *以下关于 $(V, \in)$ 成立：*

(i) *外延性* (extensionality):
$$\forall (x, y : V).\, x \subseteq y \wedge y \subseteq x \Leftrightarrow x = y.$$
(ii) *空集* (empty set): *对于所有 $x : V$，我们有 $\neg(x \in \emptyset)$。*
(iii) *配对* (pairing): *对于所有 $u, v : V$，类 $\{u, v\} :\equiv \{\, x \mid x = u \vee x = v \,\}$ 是一个 $V$-集合。*
(iv) *无穷* (infinity): *存在 $v : V$ 使得 $\emptyset \in v$ 且 $x \in v$ 意味着 $x \cup \{x\} \in v$。*
(v) *并集* (union): *对于所有 $v : V$，类 $\cup v :\equiv \{\, x \mid \exists (u : V).\, x \in u \in v \,\}$ 是一个 $V$-集合。*
(vi) *函数集* (function set): *对于所有 $u, v : V$，类 $v^u :\equiv \{\, x \mid x : u \to v \,\}$ 是一个 $V$-集合。*[1]
(vii) *$\in$-归纳* (induction): *如果 $C : V \to \mathsf{Prop}$ 是一个类，并且当对所有 $x \in a$ 都有 $C(x)$ 时 $C(a)$ 成立，那么对于所有 $v : V$，$C(v)$ 成立。*
(viii) *替代* (replacement): *给定任何 $r : V \to V$ 和 $x : V$，类*
$$\{\, y \mid \exists (z : V).\, z \in x \wedge y = r(z) \,\}$$
*是一个 $V$-集合。*
(ix) *分离* (separation): *给定任何 $a : V$ 和 $\mathcal{U}$-小 $C : V \to \mathsf{Prop}_{\mathcal{U}}$，类*
$$\{\, x \mid x \in a \wedge C(x) \,\}$$
*是一个 $V$-集合。*

*证明* (Sketch of proof).

(i) *外延性* (Extensionality): 如果 $\mathsf{set}(A,f) \subseteq \mathsf{set}(B,g)$，那么对于每个 $a : A$，有 $f(a) \in \mathsf{set}(B,g)$，因此对于每个 $a : A$，仅存在 $b : B$ 使得 $f(a) = g(b)$。假设 $\mathsf{set}(B,g) \subseteq \mathsf{set}(A,f)$ 得到了 (10.5.2) 的另一半，因此 $\mathsf{set}(A,f) = \mathsf{set}(B,g)$。
(ii) *空集* (Empty set): 假设 $x \in \emptyset = \mathsf{set}(\mathbf{0}, \mathsf{rec}_{\mathbf{0}}(V))$。那么 $\exists (a : \mathbf{0}).\, x = \mathsf{rec}_{\mathbf{0}}(V, a)$，这是荒谬的。
(iii) *配对* (Pairing): 给定 $u$ 和 $v$，令 $w = \mathsf{set}(\mathbf{2}, \mathsf{rec}_{\mathbf{2}}(V, u, v))$。

[1] 这里 $x : u \to v$ 的意思是 $x$ 是一组适当的有序对，根据集合论中编码函数的常规方式。

(iv) 无穷 (Infinity): 取 $w = \mathsf{set}(\mathbb{N}, I)$，其中 $I : \mathbb{N} \to V$ 由递归 $I(0) :\equiv \emptyset$ 和 $I(n+1) :\equiv I(n) \cup \{I(n)\}$ 给出。

(v) 并集 (Union): 取任意 $v : V$ 和任意表示 $f : A \to V$，使得 $v = \mathsf{set}(A, f)$。然后令 $\tilde{A} :\equiv \sum_{(a:A)} [fa]$，其中 $m_{fa} : [fa] \rightarrowtail V$ 是来自 Definition 10.5.7 的成员类型。显然 $\tilde{A}$ 是 $\mathcal{U}$-小的，我们定义 $\cup v :\equiv \mathsf{set}(\tilde{A}, \lambda x.\, m_{f(\mathsf{pr}_1(x))}(\mathsf{pr}_2(x)))$。

(vi) 函数集 (Function set): 给定 $u, v : V$，取成员类型 $[u] \rightarrowtail V$ 和 $[v] \rightarrowtail V$ 以及函数类型 $[u] \to [v]$。我们想要定义一个映射

$$r : ([u] \to [v]) \longrightarrow V$$

其中"$r(f) = \{\, (x, f(x)) \mid x : [u] \,\}$"，但为了使其有意义，我们首先必须定义有序对 $(x, y)$，然后我们取映射 $r' : x \mapsto (x, f(x))$，然后我们可以定义 $r(f) :\equiv \mathsf{set}([u], r')$。但有序对可以根据通常的无序配对定义。

(vii) $\in$-归纳 ($\in$-induction): 令 $C : V \to \mathsf{Prop}$ 为一个类，假设当对所有 $x \in a$ 都有 $C(x)$ 时 $C(a)$ 成立，并且取任意 $v = \mathsf{set}(B, g)$。为了通过归纳证明 $C(v)$，假设对所有 $b : B$，有 $C(g(b))$。对于每个 $x \in v$，仅存在 $b : B$，使得 $x = g(b)$，因此 $C(x)$ 成立。因此 $C(v)$ 成立。

(viii) 替代 (Replacement): 令 $C$ 表示上述的类。声明"$C$ 是一个 $V$-集合"是一个单纯命题，因此我们可以通过如下归纳进行推导。假设 $x$ 是 $\mathsf{set}(A, f)$，我们断言 $w :\equiv \mathsf{set}(A, r \circ f)$ 是我们寻找的集合。如果 $C(y)$ 成立，那么仅存在 $z : V$ 和 $a : A$ 使得 $z = f(a)$ 且 $y = r(z)$，因此 $y \in w$。反之，如果 $y \in w$，那么仅存在 $a : A$ 使得 $y = r(f(a))$，因此如果我们取 $z :\equiv f(a)$，我们可以看到 $C(y)$ 成立。

(ix) 令我们称一个类 $C : V \to \mathsf{Prop}$ 是**可分离的** (separable)，如果对于任何 $a : V$，类

$$a \cap C :\equiv \{\, x \mid x \in a \wedge C(x) \,\}$$

是一个 $V$-集合。我们需要证明任何 $\mathcal{U}$-小的 $C : V \to \mathsf{Prop}_{\mathcal{U}}$ 是可分离的。事实上，给定 $a = \mathsf{set}(A, f)$，令 $A' = \sum_{(x:A)} C(fx)$，并取 $f' = f \circ i$，其中 $i : A' \to A$ 是明显的包含映射。然后我们可以取 $a' = \mathsf{set}(A', f')$，并且有 $x \in a \wedge C(x) \Leftrightarrow x \in a'$，如所述。我们需要假设 $C$ 的值域在 $\mathcal{U}$ 中，以便 $A' = \sum_{(x:A)} C(fx)$ 在 $\mathcal{U}$ 中。 □

我们还可以使用一个严格的句法标准来判断可分离性，以便从类的表达式中读出它是否产生了一个 $V$-集合。一个熟悉的条件是 $\Delta_0$，这意味着表达式是由等式 $x =_V y$ 和隶属关系 $x \in y$ 通过仅限于单纯命题的连接词 $\neg$，$\wedge$，$\vee$，$\Rightarrow$ 和量词 $\forall$，$\exists$ （即所谓的**有界量词** (bounded quantifiers)）构建的。

Corollary 10.5.9. 如果类 $C : V \to \mathsf{Prop}$ 是上述意义上的 $\Delta_0$，那么它是可分离的。

证明. 记住我们有一个 $\mathcal{U}$-小的 $\sim$ 恒等关系 $x = y$。由于 $x \in y$ 是根据 $x = y$ 定义的，我们也有一个 $\mathcal{U}$-小的隶属关系的调整大小

$$x \mathrel{\widetilde{\in}} \mathsf{set}(A, f) :\equiv \exists (a : A).\, x \sim f(a)$$

现在，设 $\Phi$ 为 $C$ 的 $\Delta_0$ 表达式，因此作为类 $\Phi = C$（严格来说，我们应该区分表达式和它们的意义，但我们将模糊它们之间的差异）。令 $\widetilde{\Phi}$ 为将所有出现的 $=$ 和 $\in$ 替换为它们的调整后的等价物 $\sim$ 和 $\mathrel{\widetilde{\in}}$ 的结果。显然，$\widetilde{\Phi}$ 也表达了 $C$，就类 $x : V$ 而言，$\widetilde{\Phi}(x) \Leftrightarrow C(x)$，因此根据同一性原理，$\widetilde{\Phi} = C$。现在只需要证明 $\widetilde{\Phi}$ 是 $\mathcal{U}$-小的，因为这样它将通过定理成为可分离的。
我们通过对表达式的构造进行归纳来证明 $\widetilde{\Phi}$ 是 $\mathcal{U}$-小的。基础情况是 $x \sim y$ 和 $x \mathrel{\widetilde{\in}} y$，它们已经被调整大小为 $\mathcal{U}$。显然，$\mathcal{U}$ 在单纯命题操作（和 $(-1)$-截断）下是封闭的，因此剩下的只是检查有界量词 $\exists (x \in a)$ 和 $\forall (y \in b)$。根据定义，

$$\exists (x \in a) P(x) :\equiv \left\| \sum_{x:V} (x \mathrel{\widetilde{\in}} a \wedge P(x)) \right\|,$$
$$\forall (y \in b) P(x) :\equiv \prod_{x:V} (x \mathrel{\widetilde{\in}} a \to P(x))$$

我们考虑 $\left\|\sum_{(x:V)}(x \mathrel{\widetilde{\in}} a \wedge P(x))\right\|$。尽管主体 $(x \mathrel{\widetilde{\in}} a \wedge P(x))$ 是 $\mathcal{U}$-小的，因为根据归纳假设，$P(x)$ 也是如此，但是对 $V$ 的量化不一定保持在 $\mathcal{U}$ 内。然而，在当前情况下，我们可以用对 $a$ 的成员类型 $[a] \rightarrowtail V$ 的量化替代它，并且很容易证明

$$\sum_{x:V}(x \mathrel{\widetilde{\in}} a \wedge P(x)) = \sum_{x:[a]} P(x)$$

右侧确实保持在 $\mathcal{U}$ 内，因为 $[a]$ 和 $P(x)$ 都在 $\mathcal{U}$ 内。$\prod_{(x:V)}(x \mathrel{\widetilde{\in}} a \to P(x))$ 的情况类似，使用 $\prod_{(x:V)}(x \mathrel{\widetilde{\in}} a \to P(x)) = \prod_{(x:[a])} P(x)$。 □

我们已经证明在具有一个宇宙 $\mathcal{U}$ 的类型论中，累积层次 $V$ 是"构造性集合论"的一个模型，具有许多标准公理。然而，据我们所知，它缺少 强收集 (strong collection)和 子集收集 (subset collection)公理，这些公理包含在 Constructive Zermelo–Fraenkel Set Theory [?] 中。在类型论中，强收集可能由于其他原因而成立。我们不知道这些公理是否在我们的模型 $(V, \in)$ 中成立，但这似乎不太可能。由于 $V$ 是系统内部的一个更高阶归纳类型，而不是一个外部构造，这在某些方面与先前的解释有所不同。

最后，考虑在我们的类型论中添加集合的选择公理，即 §10.1.5 中的 AC。这会导致 LEM 成立，由 Theorem 10.1.14 可知，因此 Set 是一个带有子对象分类器 $\mathbf{2}$ 的拓扑，由 Theorem 10.1.12 得出。在这种情况下，我们有 $\mathsf{Prop} = \mathbf{2} : \mathcal{U}$，因此所有类都是可分离的。因此我们得出：

Lemma 10.5.10. 在具有 AC 的类型论中，**完全分离** (full separation)对于 $V$ 成立：给定任何类 $C : V \to \mathsf{Prop}$ 和 $a : V$，类 $a \cap C$ 是一个 $V$-集合。

Theorem 10.5.11. 在具有 AC 和一个宇宙 $\mathcal{U}$ 的类型论中，累积层次 $V$ 是带选择的 Zermelo–Fraenkel 集合论 (ZFC) 的一个模型。

证明. 我们已经有 Theorem 10.5.8 中列出的所有公理，加上完全分离，因此我们只需要证明对于所有 $a : V$，存在幂集 $\mathcal{P}(a) : V$。但是由于我们有 LEM，这些仅仅是函数类型 $\mathcal{P}(a) = (a \to \mathbf{2})$。因此 $V$ 是 Zermelo–Fraenkel 集合论 ZF 的一个模型。我们将选择公理的验证作为一个简单的练习留给读者。 □

## Notes

关于集合范畴的基本性质可以追溯到早期的初等拓扑论的日子。§10.1.5 中提到的 集合范畴的初等理论 (Elementary theory of the category of sets) 是由 Lawvere 在 [?] 中引入的，作为集合论的范畴论公理化。[?] 中介绍了 ΠW-预拓扑的概念，作为初等拓扑的一个预判版本；另请参阅 [?]。

§10.1 中关于集合范畴的处理大致遵循了 [?] 中的处理方式。Lemma 10.1.4 中的结果，即上同态是满射在经典数学中是众所周知的，但作为一个预判性的证明，它并不像看起来那样简单。[?] 中的证明使用了幂集操作（这是不可判定的），尽管它也可以被视为一个在更高宇宙 $\mathcal{U}_{i+1}$ 中的映射是满射的预判性证明。Wilander [?] 给出了集合论的一个预判性证明。我们的证明类似于 Wilander 的证明，但通过使用推出和同一性避免了集合论。

Theorem 10.1.14 中从 AC 到 LEM 的推导是 Diaconescu [?] 在拓扑论中的一个定理在同伦类型论中的改编，Bishop [?, Problem 2] 早在其问题中就提出了这一问题。

关于序数的直觉主义理论，请参见 [?, ?] 以及 [?]。在类型论中，通过归纳原理定义良基性，包括可达性的归纳谓词，已经在 [?, ?, ?] 中进行了研究，尽管这一想法可以追溯到 Gentzen 关于算术一致性的证明 [?]。

代数集合论的思想启发了我们在 §10.5 中对累积层次的展开，这一思想源于 [?]，但其起源于早期的 [?] 的工作。

## Exercises

Exercise 10.1. 按照 $\mathcal{S}et$ 的模式，我们想要构造一个所有类型和它们之间的映射的范畴 $\mathcal{T}ype$（在给定的宇宙 $\mathcal{U}$ 中）。然而，为了使其成为 §9.1 中意义上的范畴，我们必须首先定义 $\hom(X,Y) :\equiv \|X \to Y\|_0$，其组合通过从普通组合 $(Y \to Z) \to (X \to Y) \to (X \to Z)$ 的截断归纳来定义。这在 Example 9.1.18 中定义为**类型的同伦前范畴**。然而，它仍然不是一个范畴，而只是一个前范畴（它的对象类型 $\mathcal{U}$ 甚至不是一个 0-类型）。通过 Rezk 完全性（见 Example 9.9.7），其类型可以与 $\|\mathcal{U}\|_1$ 识别。证明所得的范畴 $\mathcal{T}ype$，不像 $\mathcal{S}et$，不是一个预拓扑。

Exercise 10.2. 证明，如果在 $\mathcal{S}et$ 范畴中每个满射都有一个截面，那么选择公理成立。

Exercise 10.3. 证明：在 LEM 成立的情况下，范畴 $\mathcal{S}et$ 是良定点的，这意味着以下陈述成立：对于任何 $f,g : A \to B$，如果 $f \neq g$，则存在一个函数 $a : 1 \to A$ 使得 $f(a) \neq g(a)$。证明切片范畴 $\mathcal{S}et/\mathbf{2}$ 由 $A \to \mathbf{2}$ 的函数和交换三角形组成，不具有此属性。（提示：$\mathcal{S}et/\mathbf{2}$ 中的终对象是恒等函数 $\mathbf{2} \to \mathbf{2}$，因此在这个范畴中，可能存在没有元素 $1 \to X$ 的对象 $X$。）

Exercise 10.4. 证明：如果 $(A, <_A)$ 和 $(B, <_B)$ 是良基的，外延的，或序数，那么 $A + B$ 也是良基的，外延的，或序数，其中 $<$ 定义为

$$\begin{aligned}
(a < a') &:\equiv (a <_A a') && \text{对于} a, a' : A \\
(b < b') &:\equiv (b <_B b') && \text{对于} b, b' : B \\
(a < b) &:\equiv \mathbf{1} && \text{对于} (a : A), (b : B) \\
(b < a) &:\equiv \mathbf{0} && \text{对于} (a : A), (b : B)
\end{aligned}$$

Exercise 10.5. 证明：如果 $(A, <_A)$ 和 $(B, <_B)$ 是良基的，外延的，或序数，那么 $A \times B$ 也是良基的，外延的，或序数，其中 $<$ 定义为

$$((a,b) < (a',b')) :\equiv (a <_A a') \vee ((a = a') \wedge (b <_B b'))$$

Exercise 10.6. 定义序数的通常代数运算，并证明它们满足通常的性质。

Exercise 10.7. 请注意 $\mathbf{2}$ 是一个序数，在显然的关系 $<$ 下，只有 $0_{\mathbf{2}} < 1_{\mathbf{2}}$ 成立。

(i) 定义 Prop 上的一个关系 $<$，使其成为一个序数。
(ii) 证明 $\mathbf{2} =_{\mathsf{Ord}} \mathsf{Prop}$ 当且仅当 LEM 成立。

Exercise 10.8. 回想一下，当我们将 $\mathbb{N}$ 视为序数时，我们将其记为 $\omega$；因此我们也有序数 $\omega + 1$。另一方面，定义

$$\mathbb{N}_\infty :\equiv \left\{ a : \mathbb{N} \to \mathbf{2} \;\middle|\; \forall (n : \mathbb{N}).\, (a_n \leq a_{\mathsf{succ}(n)}) \right\}$$

其中 $\leq$ 表示 $\mathbf{2}$ 上的显然部分顺序，其中 $0_{\mathbf{2}} \leq 1_{\mathbf{2}}$。

(i) 定义 $\mathbb{N}_\infty$ 上的一个关系 $<$，使其成为一个序数。
(ii) 证明 $\omega + 1 =_{\mathsf{Ord}} \mathbb{N}_\infty$ 当且仅当有限全知原理 (11.5.8) 成立。

Exercise 10.9. 证明：如果 $(A, <)$ 是良基的且外延的，并且 $A : \mathcal{U}$，那么存在一个仿真 $A \to V$，其中 $(V, \in)$ 是根据宇宙 $\mathcal{U}$ 构建的累积层次 §10.5。

Exercise 10.10. 证明 Theorem 10.4.4(i) 与选择公理 (3.8.1) 等价。

Exercise 10.11. 给定类型 $A$ 和 $B$，定义**双全关系** (bitotal relation)为 $R : A \to B \to \mathsf{Prop}$，使得

$$\Big(\forall (a : A).\, \exists (b : B).\, R(a,b)\Big) \wedge \Big(\forall (b : B).\, \exists (a : A).\, R(a,b)\Big)$$

对于这样的 $A, B, R$，令 $A \sqcup^R B$ 为以下更高阶归纳类型生成的类型：

- $i : A \to A \sqcup^R B$

- $j : B \to A \sqcup^R B$
- 对于每个 $a : A$ 和 $b : B$，如果 $R(a,b)$ 成立，则存在路径 $i(a) = j(b)$。

证明：累积层次 $V$ 可以通过以下更直接的构造生成，并且所得的归纳原理就是在 §10.5 中给出的原理。

- 对于每个 $A : \mathcal{U}$ 和 $f : A \to V$，存在一个元素 $\mathsf{set}(A,f) : V$。
- 对于任何 $A, B : \mathcal{U}$ 和双全关系 $R : A \to B \to \mathsf{Prop}$，以及任何映射 $h : A \sqcup^R B \to V$，存在路径 $\mathsf{set}(A, h \circ i) = \mathsf{set}(B, h \circ j)$。

- 0-截断构造子。

Exercise 10.12. 在 Constructive Zermelo–Fraenkel Set Theory 中，**强收集公理** (axiom of strong collection)的形式如下：

$$\Big(\forall(x \in v).\, \exists(y).\, R(x,y)\Big) \Rightarrow \exists(w).\, \big[\big(\forall(x \in v).\, \exists(y \in w).\, R(x,y)\big) \wedge \big(\forall(y \in w).\, \exists(x \in v).\, R(x,y)\big)\big]$$

它在累积层次 $V$ 中成立吗？（我们不知道答案。）

Exercise 10.13. 验证：如果我们假设 $\mathsf{AC}$，那么累积层次 $V$ 满足通常的集合论选择公理，其形式如下：

$$\forall(x : V).\, \Big((\forall(y \in x).\, \exists(z : V).\, z \in y) \Rightarrow \exists(c \in (\cup x)^x).\, \forall(y \in x).\, c(y) \in y\Big)$$

Exercise 10.14. 假设命题重设，证明存在一个单纯谓词 $\mathsf{isPlump} : \mathsf{Ord} \to \mathsf{Prop}$，使得对于任何 $A : \mathsf{Ord}$，我们有

$$\mathsf{isPlump}(A) = \Big(\forall(B < A).\, \mathsf{isPlump}(B)\Big) \wedge \Big(\forall(C, B : \mathsf{Ord}).\, C \leq B < A \wedge \mathsf{isPlump}(C) \Rightarrow C < A\Big)$$

请注意，$\mathsf{isPlump}$ 不能通过对 $\mathsf{Ord}$ 的简单良基归纳来定义；你必须使用不同的良基关系。我们说一个序数 $A$ 是 **丰满的** (plump) [?, ?] 如果 $\mathsf{isPlump}(A)$ 成立。

Exercise 10.15. 证明：$\mathsf{LEM}$ 与“所有序数都是丰满的”这一陈述等价。

Exercise 10.16. 定义序数 $A$ 的**丰满后继** (plump successor)为

$$t(A) :\equiv \{\, B : \mathsf{Ord} \mid (B \leq A) \wedge \mathsf{isPlump}(B) \,\}$$

(i) 根据定义，$t(A)$ 属于更高的宇宙。证明：假设命题重设，它等于与 $A$ 在同一宇宙中的一个序数。
(ii) 再次假设命题重设，证明如果 $A$ 是丰满的（见 Exercise 10.14），那么 $t(A)$ 也是丰满的。

Exercise 10.17. 相对于一个宇宙 $\mathcal{U}_i$，一个 ZF-**代数** (ZF-algebra) [?]是一个偏序集（见 Example 9.1.14）$V : \mathcal{U}_{i+1}$，它具有所有由类型 $\mathcal{U}_i$ 索引的上确界，并且配备了一个“后继”函数 $s : V \to V$（不一定以任何方式尊重 $\leq$）。

(i) 证明：累积层次 $(V_{\mathcal{U}_i}, \subseteq, s)$ 是初始 ZF-代数，其中 $s(x)$ 是单元集 $\{\, x \,\}$。
(ii) 证明：$(\mathsf{Ord}_{\mathcal{U}_i}, \leq, s)$ 是具有 $x \leq s(x)$ 对于所有 $x$ 成立的初始 ZF-代数，其中 $s(A) = A + \mathbf{1}$ 是 Lemma 10.3.21 中的后继。
(iii) 假设命题重设，证明 $\Big(\{\, A : \mathsf{Ord}_{\mathcal{U}_i} \mid \mathsf{isPlump}(A) \,\}, \leq, t\Big)$ 是具有 $(x \leq y) \Rightarrow (t(x) \leq t(y))$ 对于所有 $x, y$ 成立的初始 ZF-代数，其中 $t$ 是 Exercise 10.16 中的丰满后继。

Exercise 10.18. 对于一个范畴 $A$，如果存在一个态射 $g : \hom_A(b,a)$ 使得 $g \circ f = 1_a$，则态射 $f : \hom_A(a,b)$ 称为**分裂单态射** (split monomorphism)。(这样的 $g$ 称为 $f$ 的**重** traction。) 证明以下陈述逻辑等价。

(i) $\mathsf{LEM}$。
(ii) 对于范畴 $\mathcal{S}et$ 中的每对集合 $A$ 和 $B$，如果 $A$ 是非空的，那么对于 $\mathcal{S}et$ 中的每个单态射 $f : A \to B$，$f$ 也是 $\mathcal{S}et$ 中的分裂单态射。

Chapter 11

# 实数 (Real numbers)

任何称得上是数学基础的理论最终都必须解决实数的构造问题，这个问题在数学分析中被理解为完备的阿基米德有序域。完备性有两种定义。一种由柯西提出，要求实数在柯西序列的极限下是封闭的，而 Dedekind 提出的更强的定义要求实数在 Dedekind 分割下是封闭的。这两种完备性定义分别引出了两种构造实数的方法，我们将在 §11.2和 §11.3中分别进行研究。在 Theorems 11.2.14 and 11.3.50中，我们从泛性质的角度对这两种构造进行了刻画：Dedekind 实数是最终的阿基米德有序域，而柯西实数是初始的柯西完备阿基米德有序域。

在传统的构造性数学中，实数总是需要某些妥协。例如，Dedekind 实数在幂集或其他形式的不可判定性存在时效果较好，而柯西实数在可数选择的存在下效果较好。然而，我们给出了柯西实数的新构造，将其作为一种更高阶的归纳-归纳类型，这种构造似乎是第三种可能性，它既不需要幂集也不需要可数选择。

在 §11.4中，我们比较了这两种实数构造。柯西实数包含在 Dedekind 实数中。如果排中律或可数选择成立，它们是一致的，但通常情况下，这种包含可能是严格的。

在 §11.5中，我们讨论了闭区间 $[0,1]$ 的三种紧致性定义。我们首先证明 $[0,1]$ 在度量意义上是紧致的，即它是完备且全有界的，并且度量紧致空间上的一致连续映射表现得如预期那样。相对而言，Bolzano–Weierstraß 性质，即每个序列都有一个收敛的子序列，意味着有限全知原理，这是排中律的一种形式。最后，我们讨论了 Heine–Borel 紧致性。有限子覆盖性质的一个朴素的表述是不可行的，但证明相关的归纳覆盖是可行的。本节基本上属于标准的构造性分析。

在同伦类型论中，实数和分析的发展可以很容易地与经典数学兼容。通过假设排中律和选择公理，我们得到了标准的经典分析：Dedekind 实数和柯西实数一致，关于 Dedekind 实数的不可判定性性质的基础问题消失了，并且区间是尽可能紧致的。

我们在 §11.6中通过构造 Conway 的超现实数作为一种更高阶的归纳-归纳类型来结束本章；这种构造在一元类型论中比在经典集合论中更自然。

除了 Chapters 2 and 3中的基本理论外，如上所述，我们还使用了“更高阶的归纳-归纳类型”来构造柯西实数和超现实数：这些结合了 Chapter 6中的思想和 §5.7中提到的归纳-归纳类型的概念。我们还经常使用 §3.7中描述的传统逻辑符号，以及（在 §10.1中证明的）我们的“集合”表现如预期的事实。

请注意，圆的通用覆盖的总空间，在 §8.1.5中扮演了类似于“实数”在经典代数拓扑中的角色，但并不是我们寻找的实数类型。该类型是可缩的，因此等价于单类型，因此它不能被赋予非平凡的代数结构。

## 11.1 有理数域 (The field of rational numbers)

我们首先构造有理数 $\mathbb{Q}$ ，因为实数可以被视为 $\mathbb{Q}$ 的完备化。一个专家可能会指出，$\mathbb{Q}$ 可以被任何近似域替代，即 $\mathbb{Q}$ 的一个子环，其中存在任意精确的近似逆元 。一个例子是二进制有理数环，这些有理数的形式为 $n/2^k$。如果我们在计算机上实现构造性数学，近似域会更适合，但我们将这种精细留给那些关心 $\pi$ 的位数的人。

我们在 §6.10中构造了整数 $\mathbb{Z}$ ，作为 $\mathbb{N} \times \mathbb{N}$ 的商，并观察到该商由幂等生成。在 §6.11中我们看到

$\mathbb{Z}$ 是在 $\mathbf{1}$ 上的自由群；我们可以类似地证明它是 $\mathbf{0}$ 上的自由交换环。有理数域 $\mathbb{Q}$ 是沿着相同的思路构造的，即为商

$$\mathbb{Q} :\equiv (\mathbb{Z} \times \mathbb{N})/\approx$$

其中

$$(u,a) \approx (v,b) :\equiv (u(b+1) = v(a+1))$$

换句话说，一个对 $(u,a)$ 代表有理数 $u/(1+a)$。这里不存在除以零的问题，因为我们巧妙地在分母 $a$ 上加了一。这里我们也有一个规范的代表选择，即最简分数。因此我们可以应用 Lemma 6.10.8来获得集合 $\mathbb{Q}$ ，它再次具有可判定的等式。

我们不打算写下 $\mathbb{Q}$ 上的算术运算，因为我们相信我们的读者知道如何计算分数，即使是在分母上加一的情况下。让我们只是记录结论，即有理数域 $\mathbb{Q}$ 的构造完全没有问题，它具有可判定的等式和可判定的顺序。它也可以被刻画为初始有序域。

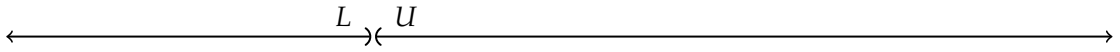
最后，我们将 $\mathbb{Q}_+ :\equiv \{\, q : \mathbb{Q} \mid q > 0 \,\}$ 记作正有理数的类型。

## 11.2 Dedekind **实数** (Dedekind reals)

让我们首先回顾一下 Dedekind 构造的基本思想。我们使用双侧的 Dedekind 分割，而不是经常使用的单侧版本，因为对称性使构造更优雅，并且它在构造性数学和经典数学中都能工作。一个 Dedekind *分割*由一对 $(L,U)$ 组成，其中 $L,U \subseteq \mathbb{Q}$，分别称为 *下分割* (lower cut) 和 *上分割* (upper cut)，满足：

(i) *非空* (inhabited): 存在 $q \in L$ 和 $r \in U$，
(ii) *圆整* (rounded): $q \in L \Leftrightarrow \exists(r \in \mathbb{Q}).\, q < r \wedge r \in L$ 和 $r \in U \Leftrightarrow \exists(q \in \mathbb{Q}).\, q \in U \wedge q < r$，
(iii) *不相交* (disjoint): $\neg(q \in L \wedge q \in U)$，并且
(iv) *确定性* (located): $q < r \Rightarrow q \in L \vee r \in U$。

从左到右阅读圆整性条件告诉我们分割是*开集* (open)，而从右到左它们分别是*下集* (lower) 和 *上集* (upper)。确定性条件表明 $L$ 和 $U$ 之间没有大间隙。由于分割始终是开集，它们永远不会包含“中间的点”，即使它是有理数。一个典型的 Dedekind 分割如下图所示：




我们可能会天真地将非正式定义翻译成类型论，认为分割是一对映射 $L,U : \mathbb{Q} \to \mathsf{Prop}$。但是我们在§3.5中看到 $\mathsf{Prop}$ 是 $\mathsf{Prop}_{\mathcal{U}_i}$ 的一种含糊的符号，其中 $\mathcal{U}_i$ 是一个宇宙。一旦我们使用特定的 $\mathcal{U}_i$ 来定义分割，实数类型将位于下一个宇宙 $\mathcal{U}_{i+1}$ 中，实数的性质位于更高的宇宙 $\mathcal{U}_{i+2}$ 中，实数子集的性质位于更高的宇宙 $\mathcal{U}_{i+3}$ 中，依此类推。原则上，我们应该能够跟踪宇宙层级，特别是在证明助手的帮助下，但这样做只会让我们负担更多的繁琐工作，因此我们更愿意避免。我们因此将作一个简化的假设，即单一命题类型 $\Omega$ 足以满足我们所有的需求。

事实上，Dedekind 实数的构造对逻辑操作相当有弹性。我们可以有几种方法来理解使用单一类型 $\Omega$ 的含义：

(i) 我们可以将 $\Omega$ 识别为含糊的 $\mathsf{Prop}$ 并跟踪所有出现在定义和构造中的宇宙。
(ii) 我们可以假设命题重设公理，如 §3.5所述，这本质上将所有 $\mathsf{Prop}_{\mathcal{U}_i}$ 的层级折叠为最低层级，我们称之为 $\Omega$。
(iii) 对于一个不关心类型论宇宙的复杂性或计算的经典数学家，他可以简单地假设对于单纯命题成立的排中律 (3.4.1)，这样 $\Omega \equiv \mathbf{2}$。这不仅消除了对 $\mathsf{Prop}$ 的层级问题，还将我们所做的一切转化为标准的经典实数构造。
(iv) 在另一个极端，人们可能会要求一个使构造工作所需的最小要求。定义一个命题为 Dedekind 分割的条件只使用合取，析取和对 $\mathbb{Q}$ 的存在量词，$\mathbb{Q}$ 是一个可数集。因此我们可以将 $\Omega$ 视为初始 $\sigma$-*框架* (sigma-frame)，即一个具有可数上确界的格，其中二元下确界分布在可数上确界上。(初始 $\sigma$-框架不能是两点格 $\mathbf{2}$，因为 $\mathbf{2}$ 不闭合于可数上确界，除非我们假设排中律。) 这将导致 $\Omega$ 的构造作为一种更高阶的归纳-归纳类型，但在 §11.3中进行这种实验已经足够了。

在所有上述情况下，$\Omega$ 是一个集合。事不宜迟，我们将非正式定义翻译成类型论。在本章中，我们使用了来自 Definition 3.7.1中的逻辑符号。

Definition 11.2.1. 一个 Dedekind **分割** (Dedekind cut) 是一对仅仅命题的 $L : \mathbb{Q} \to \Omega$ 和 $U : \mathbb{Q} \to \Omega$，满足：

(i) 非空 (inhabited) (即有界): $\exists(q : \mathbb{Q}).\, L(q)$ 和 $\exists(r : \mathbb{Q}).\, U(r)$，
(ii) 圆整 (rounded): 对于所有 $q, r : \mathbb{Q}$，

$$L(q) \Leftrightarrow \exists(r : \mathbb{Q}).\, (q < r) \wedge L(r) \qquad \text{并且}$$
$$U(r) \Leftrightarrow \exists(q : \mathbb{Q}).\, (q < r) \wedge U(q)$$

(iii) 不相交 (disjoint): 对于所有 $q : \mathbb{Q}$，$\neg(L(q) \wedge U(q))$，
(iv) 确定性 (located): 对于所有 $q, r : \mathbb{Q}$，$(q < r) \Rightarrow L(q) \vee U(r)$。

我们用 $\mathsf{isCut}(L, U)$ 表示这些条件的合取。定义 Dedekind **实数** (Dedekind reals) 的类型为

$$\mathbb{R}_{\mathsf{d}} :\equiv \{\, (L, U) : (\mathbb{Q} \to \Omega) \times (\mathbb{Q} \to \Omega) \mid \mathsf{isCut}(L, U) \,\}$$

显然，$\mathsf{isCut}(L, U)$ 是一个仅仅命题，并且由于 $\mathbb{Q} \to \Omega$ 是一个集合，Dedekind 实数也形成了一个集合。参见 Exercises 11.2 to 11.4，了解 Dedekind 分割的变体，它们导致了扩展实数、下实数和上实数以及区间域。
存在一个嵌入 $\mathbb{Q} \to \mathbb{R}_{\mathsf{d}}$，它将每个有理数 $q : \mathbb{Q}$ 关联到分割 $(L_q, U_q)$，其中

$$L_q(r) :\equiv (r < q) \qquad \text{并且} \qquad U_q(r) :\equiv (q < r)$$

我们将简单地用 $q$ 表示与有理数相关联的分割 $(L_q, U_q)$。

### 11.2.1 Dedekind **实数的代数结构** (The algebraic structure of Dedekind reals)

在直觉主义逻辑中，Dedekind 实数的代数和序理论结构的构造如往常一样进行。我们不打算详细讨论，而是指出经典和直觉主义设置之间的差异。用 $L_x$ 和 $U_x$ 表示实数 $x : \mathbb{R}_{\mathsf{d}}$ 的下分割和上分割，我们定义加法为

$$L_{x+y}(q) :\equiv \exists(r, s : \mathbb{Q}).\, L_x(r) \wedge L_y(s) \wedge q = r + s,$$
$$U_{x+y}(q) :\equiv \exists(r, s : \mathbb{Q}).\, U_x(r) \wedge U_y(s) \wedge q = r + s$$

并定义加法逆元为

$$L_{-x}(q) :\equiv \exists(r : \mathbb{Q}).\, U_x(r) \wedge q = -r,$$
$$U_{-x}(q) :\equiv \exists(r : \mathbb{Q}).\, L_x(r) \wedge q = -r$$

通过这些操作，$(\mathbb{R}_{\mathsf{d}}, 0, +, -)$ 是一个阿贝尔群。乘法稍微复杂一点：

$$L_{x\cdot y}(q) :\equiv \exists(a, b, c, d : \mathbb{Q}).\, L_x(a) \wedge U_x(b) \wedge L_y(c) \wedge U_y(d) \wedge q < \min(a \cdot c, a \cdot d, b \cdot c, b \cdot d)$$
$$U_{x\cdot y}(q) :\equiv \exists(a, b, c, d : \mathbb{Q}).\, L_x(a) \wedge U_x(b) \wedge L_y(c) \wedge U_y(d) \wedge \max(a \cdot c, a \cdot d, b \cdot c, b \cdot d) < q$$

这些公式与区间算术中区间的乘法有关，其中区间 $[a, b]$ 和 $[c, d]$ 的端点是有理数，乘积为区间

$$[a, b] \cdot [c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$$

例如，下分割的公式可以解释为当存在包含 $x$ 和 $y$ 的区间 $[a, b]$ 和 $[c, d]$ 时，$q < x \cdot y$ 说明 $q$ 位于 $[a, b] \cdot [c, d]$ 的左侧。通常将满足 $L_x(a)$ 和 $U_x(b)$ 的区间 $[a, b]$ 视为 $x$ 的近似是有用的，参见 Exercise 11.4。

现在我们有了一个具有单位元的交换环，单位元为 $(\mathbb{R}_{\mathsf{d}}, 0, 1, +, -, \cdot)$。为了处理乘法逆元，我们首先引入顺序。定义 $\leq$ 和 $<$ 为

$$\begin{aligned}(x \leq y) &:\equiv \forall(q:\mathbb{Q}).\, L_x(q) \Rightarrow L_y(q)\\ (x < y) &:\equiv \exists(q:\mathbb{Q}).\, U_x(q) \wedge L_y(q)\end{aligned}$$

Lemma 11.2.2. *对于所有 $x:\mathbb{R}_{\mathsf{d}}$ 和 $q:\mathbb{Q}$，$L_x(q) \Leftrightarrow (q < x)$ 并且 $U_x(q) \Leftrightarrow (x < q)$。*

*证明.* 如果 $L_x(q)$，则通过圆整性，存在仅仅一个 $r > q$ 使得 $L_x(r)$，并且由于 $U_q(r)$，因此得出 $q < x$。反之，如果 $q < x$，则存在 $r:\mathbb{Q}$ 使得 $U_q(r)$ 并且 $L_x(r)$，因此由于 $L_x$ 是下集，因此 $L_x(q)$。其余证明对称成立。 □

关系 $\leq$ 是偏序，并且 $<$ 是传递的和反自反的。线性性

$$(x < y) \vee (y \leq x)$$

在假设排中律时成立，但在不假设排中律的情况下，我们得到弱线性性

$$(x < y) \Rightarrow (x < z) \vee (z < y) \tag{11.2.3}$$

乍一看，(11.2.3) 可能不清楚与线性顺序的关系。但如果我们令 $x \equiv u - \epsilon$ 和 $y \equiv u + \epsilon$，其中 $\epsilon > 0$，那么我们得到

$$(u - \epsilon < z) \vee (z < u + \epsilon)$$

这是"加上一个小数误差"的线性性，即，由于不合理地期望我们可以实际用无限精度计算，我们不应感到惊讶，我们只能在计算的任何有限精度范围内决定 $<$。
要看出 (11.2.3) 成立，假设 $x < y$。那么仅仅存在 $q:\mathbb{Q}$ 使得 $U_x(q)$ 并且 $L_y(q)$。通过圆整性，存在仅仅 $r, s:\mathbb{Q}$ 使得 $r < q < s$，$U_x(r)$ 并且 $L_y(s)$。然后，通过确定性 $L_z(r)$ 或 $U_z(s)$。在第一种情况下，我们得到 $x < z$，在第二种情况下，得到 $z < y$。
在经典情况下，乘法逆元存在于所有不等于零的数中。然而，在没有排中律的情况下，需要一个更强的条件。称 $x, y:\mathbb{R}_{\mathsf{d}}$ 彼此**相异** (apart)，记作 $x \# y$，当 $(x < y) \vee (y < x)$ 时：

$$(x \# y) :\equiv (x < y) \vee (y < x)$$

如果 $x \# y$，则 $\neg(x = y)$。如果假设排中律，反过来也是成立的，但在构造性数学中不可证。事实上，如果 $\neg(x = y)$ 蕴含 $x \# y$，那么排中律的一小部分将成立；参见 Exercise 11.10。

Theorem 11.2.4. *当且仅当一个实数与 0 相异时，它是可逆的。*

Remark 11.2.5. 我们观察到一个实数是可逆的，当且仅当它是仅仅可逆的。实际上，在任何环中都是如此，因为一个环是一个集合，如果存在，乘法逆元是唯一的。参见 Corollary 3.9.2后的讨论。

*证明.* 假设 $x \cdot y = 1$。那么仅仅存在 $a, b, c, d : \mathbb{Q}$ 使得 $a < x < b$，$c < y < d$ 并且 $0 < \min(ac, ad, bc, bd)$。由于 $0 < ac$ 和 $0 < bc$，可得 $a$，$b$ 和 $c$ 要么全为正数，要么全为负数。因此，要么 $0 < a < x$，要么 $x < b < 0$，因此 $x \# 0$。
反之，如果 $x \# 0$，则 $x > 0$ 或 $x < 0$。如果 $x > 0$，我们定义 $x^{-1}$ 如下：

$$\begin{aligned}L_{x^{-1}}(q) &:\equiv (q > 0) \Rightarrow \exists(r:\mathbb{Q}).\, U_x(r) \wedge (qr < 1)\\ U_{x^{-1}}(q) &:\equiv (q > 0) \wedge \exists(r:\mathbb{Q}).\, L_x(r) \wedge (qr > 1)\end{aligned}$$

如果 $x < 0$，则我们通过以下方式定义它：

$$\begin{aligned}L_{x^{-1}}(q) &:\equiv (q < 0) \wedge \exists(r:\mathbb{Q}).\, U_x(r) \wedge (qr > 1)\\ U_{x^{-1}}(q) &:\equiv (q < 0) \Rightarrow \exists(r:\mathbb{Q}).\, L_x(r) \wedge (qr < 1)\end{aligned}$$

□

阿基米德原理可以用几种方式表达。我们认为最具启发性的是形式，它说 $\mathbb{Q}$ 在 $\mathbb{R}_{\mathsf{d}}$ 中是稠密的。

Theorem 11.2.6 (阿基米德原理对于 $\mathbb{R}_{\mathsf{d}}$). *对于所有 $x,y:\mathbb{R}_{\mathsf{d}}$，如果 $x<y$，则仅仅存在 $q:\mathbb{Q}$ 使得 $x<q<y$。*

*证明.* 根据 $<$ 的定义。 □

在处理 Dedekind 实数的完备性之前，让我们准确地描述它们具有的代数结构。在以下定义中，我们的目标不是最小公理化，而是一个有用的结构和性质。

Definition 11.2.7. 一个**有序域** (ordered field) 是一个集合 $F$，它与常数 $0$，$1$，运算 $+$，$-$，$\cdot$，$\min$，$\max$，和仅仅关系 $\leq$，$<$，$\#$ 使得：

(i) $(F,0,1,+,-,\cdot)$ 是一个交换环，带有单位元；
(ii) 当且仅当 $x\#0$ 时，$x:F$ 是可逆的；
(iii) $(F,\leq,\min,\max)$ 是一个格；
(iv) 严格顺序 $<$ 是传递的，反自反的，并且弱线性 $(x<y\Rightarrow x<z\vee z<y)$；
(v) 相异性 $\#$ 是反自反的，对称的，并且是对传递的 $(x\#y\Rightarrow x\#z\vee y\#z)$；
(vi) 对于所有 $x,y,z:F$：

$$\begin{aligned} x\leq y &\Leftrightarrow \neg(y<x) & x<y\leq z &\Rightarrow x<z\\ x\#y &\Leftrightarrow (x<y)\vee(y<x) & x\leq y<z &\Rightarrow x<z\\ x\leq y &\Leftrightarrow x+z\leq y+z & x\leq y\wedge 0\leq z &\Rightarrow xz\leq yz\\ x<y &\Leftrightarrow x+z<y+z & 0<z &\Rightarrow (x<y\Leftrightarrow xz<yz)\\ 0<x+y &\Rightarrow 0<x\vee 0<y & 0&<1 \end{aligned}$$

每个这样的域都有一个规范的嵌入 $\mathbb{Q}\to F$。一个有序域是 **阿基米德** (archimedean) 当对于所有 $x,y:F$，如果 $x<y$，则仅仅存在 $q:\mathbb{Q}$ 使得 $x<q<y$。

Theorem 11.2.8. Dedekind *实数形成了一个有序的阿基米德域。*

*证明.* 我们省略了证明，因为我们已经展示的内容使得该定理看起来是合理的。 □

### 11.2.2 Dedekind 实数是柯西完备的 (Dedekind reals are Cauchy complete)

回顾一下，$x:\mathbb{N}\to\mathbb{Q}$ 是一个*柯西序列* (Cauchy sequence)当它满足

$$\prod_{(\epsilon:\mathbb{Q}_+)}\sum_{(n:\mathbb{N})}\prod_{(m,k\geq n)}|x_m-x_k|<\epsilon \tag{11.2.9}$$

请注意，我们并*没有*截断内部存在的存在量词，因为我们实际上希望计算收敛速度——没有误差估计的近似几乎没有什么有用的信息。通过 Theorem 2.9.7, (11.2.9)产生一个函数 $M:\mathbb{Q}_+\to\mathbb{N}$, 称为*收敛模* (modulus of convergence), 使得 $m,k\geq M(\epsilon)$ 时, $|x_m-x_k|<\epsilon$。由此我们得到 $|x_{M(\delta/2)}-x_{M(\epsilon/2)}|<\delta+\epsilon$ 对于所有 $\delta,\epsilon:\mathbb{Q}_+$。事实上，映射 $(\epsilon\mapsto x_{M(\epsilon/2)}):\mathbb{Q}_+\to\mathbb{Q}$ 携带的极限信息与原始的柯西条件 (11.2.9) 相同。我们将处理这些近似函数而不是柯西序列。

Definition 11.2.10. 一个**柯西近似** (Cauchy approximation) 是一个映射 $x:\mathbb{Q}_+\to\mathbb{R}_{\mathsf{d}}$，满足

$$\forall(\delta,\epsilon:\mathbb{Q}_+).\,|x_\delta-x_\epsilon|<\delta+\epsilon \tag{11.2.11}$$

一个柯西近似 $x:\mathbb{Q}_+\to\mathbb{R}_{\mathsf{d}}$ 的**极限** (limit) 是一个实数 $\ell:\mathbb{R}_{\mathsf{d}}$，使得

$$\forall(\epsilon,\theta:\mathbb{Q}_+).\,|x_\epsilon-\ell|<\epsilon+\theta$$

Theorem 11.2.12. $\mathbb{R}_{\mathsf{d}}$ *中的每一个柯西近似都有一个极限。*

证明. 请注意，我们正在展示极限的存在，而不是仅仅存在。给定一个柯西近似 $x : \mathbb{Q}_+ \to \mathbb{R}_\mathsf{d}$，定义

$$L_y(q) :\equiv \exists(\epsilon, \theta : \mathbb{Q}_+).\, L_{x_\epsilon}(q + \epsilon + \theta)$$
$$U_y(q) :\equiv \exists(\epsilon, \theta : \mathbb{Q}_+).\, U_{x_\epsilon}(q - \epsilon - \theta)$$

很明显 $L_y$ 和 $U_y$ 是非空的，圆整的，不相交的。为了证明确定性，考虑任意的 $q, r : \mathbb{Q}$ 满足 $q < r$。存在 $\epsilon : \mathbb{Q}_+$ 满足 $5\epsilon < r - q$。由于 $q + 2\epsilon < r - 2\epsilon$ 仅仅 $L_{x_\epsilon}(q + 2\epsilon)$ 或 $U_{x_\epsilon}(r - 2\epsilon)$。在第一种情况下，我们有 $L_y(q)$，在第二种情况下我们有 $U_y(r)$。
为了表明 $y$ 是 $x$ 的极限，考虑任意的 $\epsilon, \theta : \mathbb{Q}_+$。由于 $\mathbb{Q}$ 在 $\mathbb{R}_\mathsf{d}$ 中是稠密的，存在仅仅的 $q, r : \mathbb{Q}$ 满足

$$x_\epsilon - \epsilon - \theta/2 < q < x_\epsilon - \epsilon - \theta/4 < x_\epsilon <$$

$\mathrm{x}_\epsilon + \epsilon + \theta/4 < r < x_\epsilon + \epsilon + \theta/2$ 因此 $q < y < r$。现在要么 $y < x_\epsilon + \theta/2$，要么 $x_\epsilon - \theta/2 < y$。在第一种情况下，我们有

$$x_\epsilon - \epsilon - \theta/2 < q < y < x_\epsilon + \theta/2$$

在第二种情况下

$$x_\epsilon - \theta/2 < y < r < x_\epsilon + \epsilon + \theta/2$$

无论哪种情况，都得出 $|y - x_\epsilon| < \epsilon + \theta$。 □

为了完整性，我们记录了经典的表达方式。

Corollary 11.2.13. 假设 $x : \mathbb{N} \to \mathbb{R}_\mathsf{d}$ 满足柯西条件 (11.2.9)。那么存在 $y : \mathbb{R}_\mathsf{d}$ 满足

$$\prod_{(\epsilon:\mathbb{Q}_+)} \sum_{(n:\mathbb{N})} \prod_{(m \geq n)} |x_m - y| < \epsilon$$

证明. 通过 Theorem 2.9.7存在 $M : \mathbb{Q}_+ \to \mathbb{N}$ 使得 $\bar{x}(\epsilon) :\equiv x_{M(\epsilon/2)}$ 是一个柯西近似。令 $y$ 为其极限，该极限通过 Theorem 11.2.12 存在。给定任意 $\epsilon : \mathbb{Q}_+$，令 $n :\equiv M(\epsilon/4)$ 并观察到，对于任何 $m \geq n$，

$$|x_m - y| \leq |x_m - x_n| + |x_n - y| = |x_m - x_n| + |\bar{x}(\epsilon/2) - y| < \epsilon/4 + \epsilon/2 + \epsilon/4 = \epsilon$$ □

### 11.2.3 Dedekind 实数是 Dedekind 完备的 (Dedekind reals are Dedekind complete)

我们得到了 $\mathbb{R}_\mathsf{d}$ 作为 $\mathbb{Q}$ 上的 Dedekind 分割的类型。但是我们也可以从任意的阿基米德有序域 $F$ 开始，并构造 $F$ 上的 Dedekind 分割。这些将再次形成一个阿基米德有序域 $\bar{F}$，称为 Dedekind **完备的** $F$，，其中 $F$ 被包含为一个子域。如果我们将此构造应用于 $\mathbb{R}_\mathsf{d}$，我们会得到更多的实数吗? 答案是否定的。事实上，我们将证明一个更强的结果：$\mathbb{R}_\mathsf{d}$ 是最终的。
称一个有序域 $F$ 是**对 $\Omega$ 可接受的** (admissible for $\Omega$) 当严格顺序 $<$ 在 $F$ 上是一个映射 $< : F \to F \to \Omega$。

Theorem 11.2.14. 每一个对 $\Omega$ 可接受的阿基米德有序域都是 $\mathbb{R}_\mathsf{d}$ 的一个子域。

证明. 令 $F$ 为一个阿基米德有序域。对于每一个 $x : F$ 定义 $L_x, U_x : \mathbb{Q} \to \Omega$ 如下

$$L_x(q) :\equiv (q < x) \qquad \text{并且} \qquad U_x(q) :\equiv (x < q)$$

(我们刚刚使用了 $F$ 是对 $\Omega$ 可接受的这个假设。) 然后 $(L_x, U_x)$ 是一个 Dedekind 分割。事实上，这些分割是非空且圆整的，因为 $F$ 是阿基米德的，$<$ 是传递的，不相交是因为 $<$ 是反自反的，并且确定性是因为 $<$ 是弱线性顺序。令 $e : F \to \mathbb{R}_\mathsf{d}$ 为映射 $e(x) :\equiv (L_x, U_x)$。
我们声称 $e$ 是一个保持并反映顺序的域嵌入。首先，注意 $e(q) = q$ 对于一个有理数 $q$。其次，我们有等价关系，对于所有 $x, y : F$，

$$x < y \Leftrightarrow (\exists(q : \mathbb{Q}).\, x < q < y) \Leftrightarrow (\exists(q : \mathbb{Q}).\, U_x(q) \wedge L_y(q)) \Leftrightarrow e(x) < e(y)$$

因此 $e$ 确实保持并反映顺序。$e(x+y) = e(x) + e(y)$ 成立是因为对于所有 $q : \mathbb{Q}$，

$$q < x + y \Leftrightarrow \exists(r, s : \mathbb{Q}).\, r < x \wedge s < y \wedge q = r + s$$

从右到左的蕴含是显然的。对于另一方向，如果 $q < x + y$，则仅仅存在 $r : \mathbb{Q}$ 使得 $q - y < r < x$，通过取 $s :\equiv q - r$ 我们得到所需的 $r$ 和 $s$。我们将保留 $e$ 的乘法的证明作为练习。 □

为了证明 $\mathbb{R}_{\mathsf{d}}$ 上的 Dedekind 分割不会给我们带来任何新内容，我们需要再证明一个引理。

Lemma 11.2.15. *如果 $F$ 对 $\Omega$ 是可接受的，那么它的 Dedekind 完备也是如此。*

*证明.* 令 $\bar{F}$ 为 $F$ 的 Dedekind 完备。$\bar{F}$ 上的严格顺序定义为

$$((L, U) < (L', U')) :\equiv \exists(q : \mathbb{Q}).\, U(q) \wedge L'(q)$$

由于 $U(q)$ 和 $L'(q)$ 是 $\Omega$ 的元素，该引理成立只要 $\Omega$ 在合取和可数存在下是封闭的，这是我们一开始就假设的。 □

Corollary 11.2.16. Dedekind *实数是* Dedekind *完备的：对于每一个实数值的* Dedekind *分割* $(L, U)$，*存在唯一的* $x : \mathbb{R}_{\mathsf{d}}$ *使得* $L(y) = (y < x)$ *并且* $U(y) = (x < y)$ *对于所有* $y : \mathbb{R}_{\mathsf{d}}$ *成立。*

*证明.* 通过 Lemma 11.2.15，$\mathbb{R}_{\mathsf{d}}$ 的 Dedekind 完备 $\bar{\mathbb{R}}_{\mathsf{d}}$ 对 $\Omega$ 是可接受的，因此通过 Theorem 11.2.14 我们有一个嵌入 $\bar{\mathbb{R}}_{\mathsf{d}} \to \mathbb{R}_{\mathsf{d}}$，以及一个嵌入 $\mathbb{R}_{\mathsf{d}} \to \bar{\mathbb{R}}_{\mathsf{d}}$。但这些嵌入必须是同构，因为它们的组合是保持顺序的域同态，它们固定了稠密子域 $\mathbb{Q}$，这意味着它们是恒等映射。推论现在立即得出，因为 $\bar{\mathbb{R}}_{\mathsf{d}} \to \mathbb{R}_{\mathsf{d}}$ 是一个同构。 □

## 11.3 **柯西实数** (Cauchy Reals)

柯西实数是通过极限方式构造 $\mathbb{Q}$ 的完备化 (completion)。在经典的柯西实数构造中，我们考虑所有柯西序列 (Cauchy sequences) 的集合 $\mathcal{C}$，然后形成适当的商 $\mathcal{C}/{\approx}$。为了证明 $\mathcal{C}/{\approx}$ 是柯西完备的，我们考虑一个柯西序列 $x : \mathbb{N} \to \mathcal{C}/{\approx}$，将其提升为序列 $\bar{x} : \mathbb{N} \to \mathcal{C}$，并使用 $\bar{x}$ 构造 $x$ 的极限。然而，将 $x$ 提升为 $\bar{x}$ 的过程使用了可数选择公理 (axiom of countable choice) 或排中律 (law of excluded middle)，我们可能希望避免这种情况。任何最后一步是商的实数构造都存在这个问题。构造性数学中有三种常见的解决方案：

(i) 假装实数是一个集合 $(\mathcal{C}, \approx)$，即附带有重合 (coincidence) 关系的柯西序列 $\mathcal{C}$。实数序列就简化为表示它们的柯西序列。
(ii) 接受可数选择公理的诱惑。毕竟，该公理在大多数基于计算观点的构造性数学模型中是有效的，比如可实现性模型。
(iii) 宣布柯西实数不够完美，转而构造 Dedekind 实数。在某些上下文中，这样的判断是完全有效的，比如在 sheaf-theoretic 模型的构造性数学中。然而，正如我们在 §11.2 中看到的，构造性 Dedekind 实数也有自己的问题。

然而，使用更高阶归纳类型 (higher inductive types)，有第四种解决方案，这种方法优于以上任一种，甚至对经典数学家也是有趣的。其核心思想是柯西实数应该是 $\mathbb{Q}$ 生成的 *自由完备度量空间* (free complete metric space)。通常，构造自由结构需要多次将结构操作应用于生成元。例如，自由群上的元素不仅仅是元素 $X$ 的二元乘积和逆元，还需要通过迭代乘积和逆元的构造形成单词。因此，我们可能自然地期望同样的过程适用于柯西完备化，相关的"操作"是"取柯西序列的极限"。(在这种情况下，即使在无穷次操作后仍然会有新的柯西序列，因此迭代必须是超限的。)
上述论证表明，如果排中律或可数选择成立，那么柯西完备化是非常特殊的：在构造空间的完备化时，只需要在"一步"操作后停止。这可以类比于自由单子和自由群可以通过（化简的）单词给出明确描述。然而，我们在 §6.11 中看到，更高阶归纳类型允许我们*直接*构造自由结构，而不管是否存在明确的描述。在本节中，我们将展示对于柯西实数也是如此（类似技术可以构造任何度量空间的柯西完备化；参见 Exercise 11.9）。具体而言，更高阶归纳类型允许我们*同时*添加柯西序列的极限并商取重合关系，从而可以避免将实数序列提升为代表序列的问题。

### 11.3.1 **柯西实数的构造** (Construction of Cauchy Reals)

作为一个更高阶归纳类型，柯西实数 $\mathbb{R}_{\mathsf{c}}$ 的构造比 §6.11 中讨论的自由代数结构更为复杂。我们打算包括一个"取极限"的构造器，其输入是一个实数的柯西序列，但"实数柯西序列"的概念依赖于度量"距离"的方法。当然，两个实数之间的距离将是另一个实数，从而导致一个潜在的问题。
然而，我们实际上所需要的不是一般意义上的"距离"，而是表达"两个实数之间的距离小于 $\epsilon$"的方式，对于任意 $\epsilon : \mathbb{Q}_+$。这可以通过一组二元关系 $\sim_\epsilon : \mathbb{R}_{\mathsf{c}} \to \mathbb{R}_{\mathsf{c}} \to \mathsf{Prop}$ 来表示。$\sim_\epsilon$ 的意图是表达 $|x-y| < \epsilon$，但由于我们尚未定义减法、绝对值或不等式（毕竟，我们刚刚定义 $\mathbb{R}_{\mathsf{c}}$），我们必须在定义 $\mathbb{R}_{\mathsf{c}}$ 的同时定义这些关系 $\sim_\epsilon$。由于 $\sim_\epsilon$ 是由两个 $\mathbb{R}_{\mathsf{c}}$ 和一个 $\mathbb{Q}_+$ 索引的类型族，我们不能使用普通的归纳（higher）定义；相反，我们必须使用 *更高阶归纳归纳定义* (higher inductive-inductive definition)。
§5.7 中提到的普通归纳归纳定义允许我们通过同时归纳定义一个类型及其索引的类型族。当然，"更高阶"版本允许类型和族具有路径构造器以及点构造器。我们不会尝试制定任何一般的更高阶归纳归纳定义理论，但希望我们对 $\mathbb{R}_{\mathsf{c}}$ 和 $\sim_\epsilon$ 的描述能够使这一思想变得清晰。

Remark 11.3.1. 我们可能还考虑一种 *更高阶归纳递归定义*，其中 $\sim_\epsilon$ 是使用 $\mathbb{R}_{\mathsf{c}}$ 的 *递归*原则定义的，同时与 $\mathbb{R}_{\mathsf{c}}$ 的 *归纳*定义一起进行。然而，我们选择归纳归纳路线有两个原因。首先，更高阶归纳递归定义在同伦语义中更难以证明。其次，更重要的是，归纳归纳定义产生了更强大的归纳原则，这是我们在发展柯西实数的基本理论时所需要的。

最后，正如我们在 §11.2.2 中讨论柯西完备性的 Dedekind 实数时所做的那样，我们将使用 *柯西逼近* (Definition 11.2.10) 而不是柯西序列。当然，我们的柯西逼近现在将由柯西实数构成，而不是 Dedekind 实数或有理数。

Definition 11.3.2. 让 $\mathbb{R}_{\mathsf{c}}$ 和关系 $\sim : \mathbb{Q}_+ \times \mathbb{R}_{\mathsf{c}} \times \mathbb{R}_{\mathsf{c}} \to \mathcal{U}$ 成为以下更高阶归纳归纳类型族。柯西实数的类型 $\mathbb{R}_{\mathsf{c}}$ 由以下构造生成：

- *有理点* (rational points): 对于任意 $q : \mathbb{Q}$，存在一个实数 $\mathsf{rat}(q)$。
- *极限点* (limit points): 对于任意 $x : \mathbb{Q}_+ \to \mathbb{R}_{\mathsf{c}}$，如果满足

$$\forall(\delta, \epsilon : \mathbb{Q}_+).\, x_\delta \sim_{\delta+\epsilon} x_\epsilon \tag{11.3.3}$$

  则存在一个点 $\mathsf{lim}(x) : \mathbb{R}_{\mathsf{c}}$。我们称 $x$ 为一个 **柯西逼近** (Cauchy approximation)。
- *路径* (paths): 对于 $u, v : \mathbb{R}_{\mathsf{c}}$，如果满足

$$\forall(\epsilon : \mathbb{Q}_+).\, u \sim_\epsilon v \tag{11.3.4}$$

  则存在一个路径 $\mathsf{eq}_{\mathbb{R}_{\mathsf{c}}}(u, v) : u =_{\mathbb{R}_{\mathsf{c}}} v$。

同时，类型族 $\sim : \mathbb{R}_{\mathsf{c}} \to \mathbb{R}_{\mathsf{c}} \to \mathbb{Q}_+ \to \mathcal{U}$ 由以下构造生成。其中 $q$ 和 $r$ 表示有理数；$\delta$、$\epsilon$ 和 $\eta$ 表示正有理数；$u$ 和 $v$ 表示柯西实数；$x$ 和 $y$ 表示柯西逼近：

- 对于任意 $q, r, \epsilon$，如果 $-\epsilon < q - r < \epsilon$，则 $\mathsf{rat}(q) \sim_\epsilon \mathsf{rat}(r)$，
- 对于任意 $q, y, \epsilon, \delta$，如果 $\mathsf{rat}(q) \sim_{\epsilon-\delta} y_\delta$，则 $\mathsf{rat}(q) \sim_\epsilon \mathsf{lim}(y)$，
- 对于任意 $x, r, \epsilon, \delta$，如果 $x_\delta \sim_{\epsilon-\delta} \mathsf{rat}(r)$，则 $\mathsf{lim}(x) \sim_\epsilon \mathsf{rat}(r)$，
- 对于任意 $x, y, \epsilon, \delta, \eta$，如果 $x_\delta \sim_{\epsilon-\delta-\eta} y_\eta$，则 $\mathsf{lim}(x) \sim_\epsilon \mathsf{lim}(y)$，
- 对于任意 $u, v, \epsilon$，如果 $\xi, \zeta : u \sim_\epsilon v$，则 $\xi = \zeta$（命题截断）。

$\mathbb{R}_{\mathsf{c}}$ 的第一个构造器表示任何有理数都可以看作实数。第二个构造器表示从任何柯西逼近到一个实数的极限，我们可以得到一个新的实数。第三个构造器表达了如果两个柯西逼近重合，那么它们的极限相等。
$\sim$ 的前四个构造器指定了何时两个有理数接近，何时一个有理数接近一个极限，以及何时两个极限接近。对于两个有理数的情况，这只是有理数 $\epsilon$-接近的通常概念，而其他情况可以通过注意每个逼近 $x_\delta$ 应该在极限 $\mathsf{lim}(x)$ 的 $\delta$ 范围内来推导。
我们提醒自己关于证明相关的内容：一个通过 $\mathsf{lim}$ 获得的实数不仅由一个柯西逼近 $x$ 表示，还需要一个证明 $p$ 证明 (11.3.3)，所以我们在技术上应该写 $\mathsf{lim}(x, p)$ 而不是简单地写 $\mathsf{lim}(x)$。类似地，$\mathsf{eq}_{\mathbb{R}_{\mathsf{c}}}$ 和

(11.3.4) 的情况也类似，但我们将简单地写 $\mathsf{eq}_{\mathbb{R}_c} : u = v$ 而不是 $\mathsf{eq}_{\mathbb{R}_c}(u, v, p) : u = v$。这些符号的滥用通过忽略命题并省略容易猜测的信息来减轻其影响。同样，$\sim_\epsilon$ 的最后一个构造器解释了我们为何将其他四个构造器留作无名。

我们立即可以用许多实数填充 $\mathbb{R}_c$。因为假设 $x : \mathbb{N} \to \mathbb{Q}$ 是一个传统的有理数柯西序列，并且设 $M : \mathbb{Q}_+ \to \mathbb{N}$ 是其收敛模数。那么 $\mathsf{rat} \circ x \circ M : \mathbb{Q}_+ \to \mathbb{R}_c$ 是一个柯西逼近，使用 $\sim$ 的第一个构造器产生所需的证明。因此，$\lim(\mathsf{rat} \circ x \circ m)$ 是一个实数。各种著名的实数如 $\sqrt{2}$、$\pi$、$e$ 等都是这种有理数柯西序列的极限。

### 11.3.2 柯西实数的归纳和递归 (Induction and Recursion on Cauchy Reals)

当然，要对 $\mathbb{R}_c$ 做有用的事情，我们需要给出其归纳原则。每当我们归纳定义两个或多个对象时，$\mathbb{R}_c$ 和 $\sim$ 的基本归纳原则都需要同时对两者进行归纳。因此，我们应期望其说的是，假设有两个类型族分别位于 $\mathbb{R}_c$ 和 $\sim$ 之上，并具有与每个构造器对应的数据，那么就存在这两个类型族的截面函数。然而，由于 $\sim$ 依赖于两个 $\mathbb{R}_c$ 和一个 $\mathbb{Q}_+$，因此这些族的确切依赖关系有点微妙。归纳原则将适用于任意一对类型族：

$$\begin{aligned} &A : \mathbb{R}_c \to \mathcal{U} \\ &B : \prod_{x,y:\mathbb{R}_c} A(x) \to A(y) \to \prod_{\epsilon:\mathbb{Q}_+} (x \sim_\epsilon y) \to \mathcal{U}. \end{aligned}$$

$A$ 的类型显而易见，但 $B$ 的类型需要仔细考虑。由于 $B$ 必须依赖于 $\sim$，但 $\sim$ 依赖于两个 $\mathbb{R}_c$ 和一个 $\mathbb{Q}_+$，所以很明显 $B$ 也必须依赖于变量 $x, y : \mathbb{R}_c$ 和 $\epsilon : \mathbb{Q}_+$ 以及 $(x \sim_\epsilon y)$ 的一个元素。稍微不太明显的是，$B$ 还必须依赖于 $A(x)$ 和 $A(y)$。

如果我们考虑非依赖型情况（递归原则），可能更容易理解，这时 $A$ 是一个简单类型（而不是类型族）。在这种情况下，我们期望 $B$ 不依赖于 $x, y : \mathbb{R}_c$ 或 $(x \sim_\epsilon y)$。但是，递归原则（及其相关的唯一性原则）应该表明 $\mathbb{R}_c$ 和 $\sim_\epsilon$ 是某种范畴中的“初始对象”，因此在这种情况下 $A$ 和 $B$ 的依赖结构应与 $\mathbb{R}_c$ 和 $\sim_\epsilon$ 的依赖结构一致：即我们应该有 $B : A \to A \to \mathbb{Q}_+ \to \mathcal{U}$。结合这一观察，并考虑到在依赖型情况下 $B$ 必须依赖于 $x, y : \mathbb{R}_c$ 和 $(x \sim_\epsilon y)$，最终导致了上述 $B$ 的类型。

将 $B$ 看作 $\epsilon$ 索引的关系族是有帮助的，这些关系位于类型 $A(x)$ 和 $A(y)$ 之间。这样，我们可以将 $B(x, y, a, b, \epsilon, \xi)$ 写为 $(x, a) \frown_\epsilon^\xi (y, b)$。由于 $\xi : x \sim_\epsilon y$ 是唯一的（如果存在），我们通常省略 $\xi$，写作 $(x, a) \frown_\epsilon (y, b)$；只要我们记住这一关系仅在 $x \sim_\epsilon y$ 时定义，这样的省略是无害的。我们有时还可以进一步简化，写作 $a \frown_\epsilon b$，其中 $x$ 和 $y$ 从 $a$ 和 $b$ 的类型中推断，但有时为了清晰起见需要包含它们。

现在，给定一个类型族 $A : \mathbb{R}_c \to \mathcal{U}$ 和一个关系族 $\frown$，归纳原则的假设包括以下数据，每个对应 $\mathbb{R}_c$ 或 $\sim$ 的一个构造器：

- 对于任意 $q : \mathbb{Q}$，有一个元素 $f_q : A(\mathsf{rat}(q))$。
- 对于任意柯西逼近 $x$，以及任何 $a : \prod_{(\epsilon:\mathbb{Q}_+)} A(x_\epsilon)$，满足

$$\forall(\delta, \epsilon : \mathbb{Q}_+).\, (x_\delta, a_\delta) \frown_{\delta+\epsilon} (x_\epsilon, a_\epsilon), \tag{11.3.5}$$

  的元素 $f_{x,a} : A(\lim(x))$。我们称这样的 $a$ 为 **依赖柯西逼近** (dependent Cauchy approximation)。在 $x$ 上。
- 对于 $u, v : \mathbb{R}_c$，如果 $h : \forall(\epsilon : \mathbb{Q}_+).\, u \sim_\epsilon v$，以及所有 $a : A(u)$ 和 $b : A(v)$ 满足 $\forall(\epsilon : \mathbb{Q}_+).\, (u, a) \frown_\epsilon (v, b)$，则有 $a =^A_{\mathsf{eq}_{\mathbb{R}_c}(u,v)} b$。
- 对于 $q, r : \mathbb{Q}$ 和 $\epsilon : \mathbb{Q}_+$，如果 $-\epsilon < q - r < \epsilon$，则 $(\mathsf{rat}(q), f_q) \frown_\epsilon (\mathsf{rat}(r), f_r)$
- 对于 $q : \mathbb{Q}$ 和 $\delta, \epsilon : \mathbb{Q}_+$ 以及 $y$ 柯西逼近，和 $b$ 依赖柯西逼近 $y$，如果 $\mathsf{rat}(q) \sim_{\epsilon-\delta} y_\delta$，那么

$$(\mathsf{rat}(q), f_q) \frown_{\epsilon-\delta} (y_\delta, b_\delta) \;\Rightarrow\; (\mathsf{rat}(q), f_q) \frown_\epsilon (\lim(y), f_{y,b})$$

- 类似地，对于 $r : \mathbb{Q}$ 和 $\delta, \epsilon : \mathbb{Q}_+$ 和 $x$ 柯西逼近，和 $a$ 依赖柯西逼近 $x$，如果 $x_\delta \sim_{\epsilon-\delta} \mathsf{rat}(r)$，那么

$$(x_\delta, a_\delta) \frown_{\epsilon-\delta} (\mathsf{rat}(r), f_r) \;\Rightarrow\; (\lim(x), f_{x,a}) \frown_\epsilon (\mathsf{rat}(q), f_r)$$

- 对于 $\epsilon,\delta,\eta:\mathbb{Q}_+$ 和 $x,y$ 柯西逼近，和 $a$ 和 $b$ 依赖柯西逼近 $x$ 和 $y$，如果我们有 $x_\delta \sim_{\epsilon-\delta-\eta} y_\eta$，那么
$$(x_\delta, a_\delta) \frown_{\epsilon-\delta-\eta} (y_\eta, b_\eta) \;\Rightarrow\; (\mathsf{lim}(x), f_{x,a}) \frown_\epsilon (\mathsf{lim}(y), f_{y,b})$$
- 对于 $\epsilon:\mathbb{Q}_+$ 和 $x,y:\mathbb{R}_\mathsf{c}$ 和 $\xi,\zeta: x\sim_\epsilon y$，和 $a:A(x)$ 和 $b:A(y)$，$(x,a)\frown^\xi_\epsilon (y,b)$ 和 $(x,a)\frown^\zeta_\epsilon (y,b)$ 中的任意两个元素在 $\xi=\zeta$ 上依赖相等。注意，这与要求 $\frown$ 取值于单纯命题是等价的。

在这些假设下，我们得到函数
$$f:\prod_{x:\mathbb{R}_\mathsf{c}} A(x)$$
$$g:\prod_{(x,y:\mathbb{R}_\mathsf{c})}\prod_{(\epsilon:\mathbb{Q}_+)}\prod_{(\xi:x\sim_\epsilon y)} (x,f(x))\frown^\xi_\epsilon (y,f(y))$$

其计算如预期：
$$f(\mathsf{rat}(q)) :\equiv f_q, \tag{11.3.6}$$
$$f(\mathsf{lim}(x)) :\equiv f_{x,(f,g)[x]} \tag{11.3.7}$$

这里 $(f,g)[x]$ 表示将 $f$ 和 $g$ 应用于柯西逼近 $x$ 以获得 $x$ 上的依赖柯西逼近的结果。也就是说，我们定义 $(f,g)[x]_\epsilon :\equiv f(x_\epsilon):A(x_\epsilon)$，然后对于任何 $\epsilon,\delta:\mathbb{Q}_+$，我们有 $g(x_\epsilon,x_\delta,\epsilon+\delta,\xi)$ 来证明 $(f,g)[x]$ 是依赖柯西逼近，其中 $\xi: x_\epsilon\sim_{\epsilon+\delta} x_\delta$ 来源于 $x$ 是柯西逼近的假设。
我们不会再使用这种符号表示，所以不用记住它。通常我们使用模式匹配约定，其中 $f$ 由类似于 (11.3.6) 和 (11.3.7) 的方程定义，其中 (11.3.7) 的右侧可能涉及符号 $f(x_\epsilon)$ 和假设它们形成依赖柯西逼近。
然而，这个归纳原则确实仍然相当复杂。为了帮助理解它，我们观察到它包含了两个关于 $\mathbb{R}_\mathsf{c}$ 和 $\sim$ 的单独的归纳原则的特殊情况。首先，假设只给定一个类型族 $A:\mathbb{R}_\mathsf{c}\to\mathcal{U}$，并将 $\frown$ 定义为常量 $\mathbf{1}$。那么许多必要的数据就变得微不足道，我们剩下：

- 对于任意 $q:\mathbb{Q}$，一个元素 $f_q:A(\mathsf{rat}(q))$，
- 对于任意柯西逼近 $x$，和任意 $a:\prod_{(\epsilon:\mathbb{Q}_+)}A(x_\epsilon)$，一个元素 $f_{x,a}:A(\mathsf{lim}(x))$，
- 对于 $u,v:\mathbb{R}_\mathsf{c}$ 和 $h:\forall(\epsilon:\mathbb{Q}_+).u\sim_\epsilon v$，以及 $a:A(u)$ 和 $b:A(v)$，我们有 $a=^A_{\mathsf{eq}_{\mathbb{R}_\mathsf{c}}(u,v)} b$。

在这些数据的基础上，归纳原则生成一个函数 $f:\prod_{(x:\mathbb{R}_\mathsf{c})}A(x)$，使得
$$f(\mathsf{rat}(q)) :\equiv f_q,$$
$$f(\mathsf{lim}(x)) :\equiv f_{x,f(x)}$$

我们称此原则为 **$\mathbb{R}_\mathsf{c}$-归纳** ($\mathbb{R}_\mathsf{c}$-induction)；它基本上表明，如果我们认为 $\sim_\epsilon$ 是已知的，那么 $\mathbb{R}_\mathsf{c}$ 是通过其构造器归纳生成的。
注意，如果 $A$ 是单纯命题，那么 $\mathbb{R}_\mathsf{c}$-归纳中的第三个假设是自动成立的（我们将很快看到这些是等价的命题）。因此，我们可以通过简单地证明对有理数和柯西逼近的极限来证明实数的单纯命题。以下是一个例子。

Lemma 11.3.8. *对于任意 $u:\mathbb{R}_\mathsf{c}$ 和 $\epsilon:\mathbb{Q}_+$，我们有 $u\sim_\epsilon u$。*

*证明.* 定义 $A(u):\equiv\forall(\epsilon:\mathbb{Q}_+).(u\sim_\epsilon u)$。由于这是一个单纯命题（根据 $\sim$ 的最后一个构造器），根据 $\mathbb{R}_\mathsf{c}$-归纳，我们只需要证明当 $u$ 是 $\mathsf{rat}(q)$ 和 $u$ 是 $\mathsf{lim}(x)$ 时的情况。在第一个情况下，我们显然有 $|q-q|<\epsilon$ 对于任意 $\epsilon$，因此根据 $\sim$ 的第一个构造器，$\mathsf{rat}(q)\sim_\epsilon\mathsf{rat}(q)$ 是成立的。在第二种情况下，我们可以假设归纳地 $x_\delta\sim_\epsilon x_\delta$ 对于所有 $\delta,\epsilon:\mathbb{Q}_+$ 成立。然后特别地，我们有 $x_{\epsilon/3}\sim_{\epsilon/3}x_{\epsilon/3}$，因此根据 $\sim$ 的第四个构造器，$\mathsf{lim}(x)\sim_\epsilon\mathsf{lim}(x)$ 成立。 □

从 Lemma 11.3.8，我们推断，如果类型族 $A:\mathbb{R}_\mathsf{c}\to\mathcal{U}$ 是一个单纯命题，那么直接应用 $\mathbb{R}_\mathsf{c}$-归纳只有在这种情况下才有可能成功。为了证明这一点，固定 $u:\mathbb{R}_\mathsf{c}$。将 $v$ 设为 $u$，$\mathbb{R}_\mathsf{c}$-归纳的第三个假设告诉我们，对于任意 $a:A(u)$，我们有 $a=^A_{\mathsf{eq}_{\mathbb{R}_\mathsf{c}}(u,u)}a$。再给定一个点 $b:A(u)$，我们也得到 $a=^A_{\mathsf{eq}_{\mathbb{R}_\mathsf{c}}(u,u)}b$。根据依赖路径类型的定义，我们得出这些路径组合的结论是 $a=b$，即 $A(u)$ 中的所有点都是相等的。

Theorem 11.3.9. $\mathbb{R}_c$ 是一个集合 (set)。

证明. 我们刚刚证明了 $P(u,v) :\equiv \forall(\epsilon : \mathbb{Q}_+).(u \sim_\epsilon v)$ 是自反的。由于它暗含了恒等性，$\mathbb{R}_c$ 的路径构造器表明结果可由 Theorem 7.2.2 推导。 □

我们还可以证明，尽管 $\mathbb{R}_c$ 可能不是有理数柯西序列集合的商，但它仍然是实数柯西序列集合的商。（当然，这不是 $\mathbb{R}_c$ 的有效定义，但它是一个有用的性质。）我们将柯西逼近的类型定义为

$$\mathcal{C} :\equiv \{\, x : \mathbb{Q}_+ \to \mathbb{R}_c \mid \forall(\epsilon, \delta : \mathbb{Q}_+).\, x_\delta \sim_{\delta+\epsilon} x_\epsilon \,\}.$$

$\mathbb{R}_c$ 的第二个构造器提供了函数 $\mathsf{lim} : \mathcal{C} \to \mathbb{R}_c$。

Lemma 11.3.10. 每个实数 (real) 仅仅是一个极限点：$\forall(u : \mathbb{R}_c).\exists(x : \mathcal{C}).\, u = \mathsf{lim}(x)$。换句话说，$\mathsf{lim} : \mathcal{C} \to \mathbb{R}_c$ 是满射。

证明. 通过 $\mathbb{R}_c$-归纳，我们可以对 $u$ 进行分情况讨论。当然，如果 $u$ 是一个极限 $\mathsf{lim}(x)$，那么该命题是显然成立的。所以假设 $u$ 是一个有理点 $\mathsf{rat}(q)$；我们声称 $u$ 等于 $\mathsf{lim}(\lambda\epsilon.\,\mathsf{rat}(q))$。通过 $\mathbb{R}_c$ 的路径构造器，只需证明 $\mathsf{rat}(q) \sim_\epsilon \mathsf{lim}(\lambda\epsilon.\,\mathsf{rat}(q))$ 对所有 $\epsilon : \mathbb{Q}_+$ 成立。而根据 $\sim$ 的第二个构造器，为此只需找到 $\delta : \mathbb{Q}_+$ 使得 $\mathsf{rat}(q) \sim_{\epsilon-\delta} \mathsf{rat}(q)$。但根据 $\sim$ 的第一个构造器，我们可以取任意 $\delta : \mathbb{Q}_+$ 使得 $\delta < \epsilon$。 □

Lemma 11.3.11. 如果 $A$ 是一个集合并且 $f : \mathcal{C} \to A$ 满足 Cauchy 近似的重合性，即

$$\forall(x, y : \mathcal{C}).\, \mathsf{lim}(x) = \mathsf{lim}(y) \Rightarrow f(x) = f(y),$$

那么 $f$ 唯一地通过 $\mathsf{lim} : \mathcal{C} \to \mathbb{R}_c$ 因子化。

证明. 由于 lim 是满射，根据 Theorem 10.1.5，$\mathbb{R}_c$ 是 $\mathcal{C}$ 通过 lim 的核对偶 的商集。这正是引理的表述。 □

对于归纳原理的第二个特例，假设我们将 $A$ 取为常量 $\mathbf{1}$。在这种情况下，$\frown$ 只是对 $\epsilon$ 近似对实数对的 $\epsilon$ 索引的关系族，因此我们可以写 $u \frown_\epsilon v$ 代替 $(u,\star) \frown_\epsilon (v,\star)$。那么所需的数据简化为如下内容，其中 $q, r$ 表示有理数，$\epsilon, \delta, \eta$ 是正有理数，而 $x, y$ 是 Cauchy 近似：

- 如果 $-\epsilon < q - r < \epsilon$，那么 $\mathsf{rat}(q) \frown_\epsilon \mathsf{rat}(r)$，
- 如果 $\mathsf{rat}(q) \sim_{\epsilon-\delta} y_\delta$ 并且 $\mathsf{rat}(q) \frown_{\epsilon-\delta} y_\delta$，那么 $\mathsf{rat}(q) \frown_\epsilon \mathsf{lim}(y)$，
- 如果 $x_\delta \sim_{\epsilon-\delta} \mathsf{rat}(r)$ 并且 $x_\delta \frown_{\epsilon-\delta} \mathsf{rat}(r)$，那么 $\mathsf{lim}(y) \frown_\epsilon \mathsf{rat}(q)$，
- 如果 $x_\delta \sim_{\epsilon-\delta-\eta} y_\eta$ 并且 $x_\delta \frown_{\epsilon-\delta-\eta} y_\eta$，那么 $\mathsf{lim}(x) \frown_\epsilon \mathsf{lim}(y)$。

由此得出的结论是 $\forall(u, v : \mathbb{R}_c).\forall(\epsilon : \mathbb{Q}_+).(u \sim_\epsilon v) \to (u \frown_\epsilon v)$。我们称这个原理为 $\sim$-**归纳**；它本质上表明，如果我们将 $\mathbb{R}_c$ 视为已知，那么 $\sim_\epsilon$ 由它的构造函数归纳生成（作为类型族）。例如，我们可以使用它来证明 $\sim$ 是对称的。

Lemma 11.3.12. 对于任意的 $u, v : \mathbb{R}_c$ 和 $\epsilon : \mathbb{Q}_+$，我们有 $(u \sim_\epsilon v) = (v \sim_\epsilon u)$。

证明. 由于两者都是单纯命题，通过对称性我们只需证明一个方向的蕴涵。因此，令 $(u \frown_\epsilon v) :\equiv (v \sim_\epsilon u)$。通过 $\sim$-归纳，我们可以将问题归结为 $u \sim_\epsilon v$ 由 $\sim$ 的四个有趣构造函数之一导出。在第一种情况下，当 $u$ 和 $v$ 都是有理数时，结果是显然的（我们可以再次应用第一个构造函数）。在其他三种情况下，归纳假设（以及 $\mathbb{Q}$ 中加法的交换性）正好得出 $\sim$ 的另一个构造函数的输入（第二个和第三个构造函数互换，而第四个保持不变）。 □

一般归纳原理，我们可以称之为 $(\mathbb{R}_c, \sim)$-**归纳**，因此是一种联合的 $\mathbb{R}_c$-归纳和 $\sim$-归纳。例如，考虑它的非依赖版本，我们称之为 $(\mathbb{R}_c, \sim)$-**递归**，这是我们最常用的。普通的 $\mathbb{R}_c$-递归告诉我们，要定义一个函数 $f : \mathbb{R}_c \to A$，只需：

(i) 对每个 $q : \mathbb{Q}$ 构造 $f(\mathsf{rat}(q)) : A$，

(ii) 对每个 Cauchy 近似 $x : \mathbb{Q}_+ \to \mathbb{R}_\mathsf{c}$，构造 $f(x) : A$，假设已为所有 $\epsilon : \mathbb{Q}_+$ 定义了 $f(x_\epsilon)$，
(iii) 证明对于所有满足 $\forall(\epsilon : \mathbb{Q}_+).\, u \sim_\epsilon v$ 的 $u, v : \mathbb{R}_\mathsf{c}$，有 $f(u) = f(v)$。

然而，在不了解 $f$ 如何作用于 $\epsilon$ 近似的 Cauchy 实数的情况下，通常很难证明 (iii)。$(\mathbb{R}_\mathsf{c}, \sim)$-递归的增强原理弥补了这一缺陷，使我们能够指定一个 *任意的* "$f$ 作用于 $\epsilon$ 近似的 Cauchy 实数的方式"，然后我们可以通过与 $f$ 的定义同时归纳来证明这一点。这就是关系族 $\frown$。由于 $A$ 独立于 $\mathbb{R}_\mathsf{c}$，为简单起见，我们可以假设 $\frown$ 仅依赖于 $A$ 和 $\mathbb{Q}_+$，因此在写 $a \frown_\epsilon b$ 时没有歧义，而不是 $(u, a) \frown_\epsilon (v, b)$。在这种情况下，通过 $(\mathbb{R}_\mathsf{c}, \sim)$-递归定义函数 $f : \mathbb{R}_\mathsf{c} \to A$ 需要以下情况（我们现在使用模式匹配约定来写）。

- 对于每个 $q : \mathbb{Q}$，构造 $f(\mathsf{rat}(q)) : A$。
- 对于每个 Cauchy 近似 $x : \mathbb{Q}_+ \to \mathbb{R}_\mathsf{c}$，构造 $f(\mathsf{lim}(x)) : A$，假设归纳地已经为所有 $\epsilon : \mathbb{Q}_+$ 定义了 $f(x_\epsilon)$ 并形成一个 "相对于 $\frown$ 的 Cauchy 近似"，即 $\forall(\epsilon, \delta : \mathbb{Q}_+).\, (f(x_\epsilon) \frown_{\epsilon+\delta} f(x_\delta))$。
- 证明这些关系 $\frown$ 是*分离的*，即对于任意 $a, b : A$，$(\forall(\epsilon : \mathbb{Q}_+).\, a \frown_\epsilon b) \Rightarrow (a = b)$
- 证明如果 $-\epsilon < q - r < \epsilon$ 对于 $q, r : \mathbb{Q}$，那么 $f(\mathsf{rat}(q)) \frown_\epsilon f(\mathsf{rat}(r))$。
- 对于任意 $q : \mathbb{Q}$ 和任意 Cauchy 近似 $y$，证明 $f(\mathsf{rat}(q)) \frown_\epsilon f(\mathsf{lim}(y))$，假设归纳地 $\mathsf{rat}(q) \sim_{\epsilon-\delta} y_\delta$ 并且 $f(\mathsf{rat}(q)) \frown_{\epsilon-\delta} f(y_\delta)$ 对于某个 $\delta : \mathbb{Q}_+$，并且 $\eta \mapsto f(x_\eta)$ 是相对于 $\frown$ 的 Cauchy 近似。
- 对于任意 Cauchy 近似 $x$ 和任意 $r : \mathbb{Q}$，证明 $f(\mathsf{lim}(x)) \frown_\epsilon f(\mathsf{rat}(r))$，假设归纳地 $x_\delta \sim_{\epsilon-\delta} \mathsf{rat}(r)$ 并且 $f(x_\delta) \frown_{\epsilon-\delta} f(\mathsf{rat}(r))$ 对于某个 $\delta : \mathbb{Q}_+$，并且 $\eta \mapsto f(x_\eta)$ 是相对于 $\frown$ 的 Cauchy 近似。
- 对于任意 Cauchy 近似 $x, y$，证明 $f(\mathsf{lim}(x)) \frown_\epsilon f(\mathsf{lim}(y))$，假设归纳地 $x_\delta \sim_{\epsilon-\delta-\eta} y_\eta$ 并且 $f(x_\delta) \frown_{\epsilon-\delta-\eta} f(y_\eta)$ 对于某个 $\delta, \eta : \mathbb{Q}_+$，并且 $\theta \mapsto f(x_\theta)$ 和 $\theta \mapsto f(y_\theta)$ 是相对于 $\frown$ 的 Cauchy 近似。

注意，在最后四个证明中，我们可以自由使用在前两个数据中给出的 $f(\mathsf{rat}(q))$ 和 $f(\mathsf{lim}(x))$ 的具体定义。然而，分离性的证明必须适用于 $A$ 的*任意*两个元素，而与 $f$ 无关：它是一种对关系族 $\frown$ 的 "可接纳性" 条件。因此，我们通常首先验证它，在定义 $\frown$ 之后立即进行，然后再继续定义 $f(\mathsf{rat}(q))$ 和 $f(\mathsf{lim}(x))$。
在上述假设下，$(\mathbb{R}_\mathsf{c}, \sim)$-递归产生一个函数 $f : \mathbb{R}_\mathsf{c} \to A$，使得 $f(\mathsf{rat}(q))$ 和 $f(\mathsf{lim}(x))$ 在判断上等于在前两个子句中给出的定义。此外，我们还可以得出

$$\forall(u, v : \mathbb{R}_\mathsf{c}).\, \forall(\epsilon : \mathbb{Q}_+).\, (u \sim_\epsilon v) \to (f(u) \frown_\epsilon f(v)). \tag{11.3.13}$$

作为一个典型的例子，$(\mathbb{R}_\mathsf{c}, \sim)$-递归允许我们将定义在 $\mathbb{Q}$ 上的函数扩展到所有 $\mathbb{R}_\mathsf{c}$ 上，只要它们足够连续。

Definition 11.3.14. 函数 $f : \mathbb{Q} \to \mathbb{R}_\mathsf{c}$ 是 Lipschitz 的，如果存在 $L : \mathbb{Q}_+$（Lipschitz **常数**），使得

$$|q - r| < \epsilon \Rightarrow (f(q) \sim_{L\epsilon} f(r))$$

对于所有 $\epsilon : \mathbb{Q}_+$ 和 $q, r : \mathbb{Q}$。类似地，$g : \mathbb{R}_\mathsf{c} \to \mathbb{R}_\mathsf{c}$ 是 Lipschitz 的，如果存在 $L : \mathbb{Q}_+$ 使得

$$(u \sim_\epsilon v) \Rightarrow (g(u) \sim_{L\epsilon} g(v))$$

对于所有 $\epsilon : \mathbb{Q}_+$ 和 $u, v : \mathbb{R}_\mathsf{c}$。

特别地，注意到通过 $\sim$ 的第一个构造函数，如果 $f : \mathbb{Q} \to \mathbb{Q}$ 在显然的意义上是 Lipschitz 的，那么复合函数 $\mathbb{Q} \xrightarrow{f} \mathbb{Q} \to \mathbb{R}_\mathsf{c}$ 也是如此。

Lemma 11.3.15. *假设 $f : \mathbb{Q} \to \mathbb{R}_\mathsf{c}$ 是常数为 $L : \mathbb{Q}_+$ 的 Lipschitz 函数。那么存在一个 Lipschitz 映射 $\bar{f} : \mathbb{R}_\mathsf{c} \to \mathbb{R}_\mathsf{c}$，其常数也是 $L$，并且对于所有 $q : \mathbb{Q}$，有 $\bar{f}(\mathsf{rat}(q)) \equiv f(q)$。*

*证明.* 我们通过 $(\mathbb{R}_\mathsf{c}, \sim)$-递归定义 $\bar{f}$，其余象为 $A :\equiv \mathbb{R}_\mathsf{c}$。我们定义关系 $\frown : \mathbb{R}_\mathsf{c} \to \mathbb{R}_\mathsf{c} \to \mathbb{Q}_+ \to \mathsf{Prop}$ 为

$$(u \frown_\epsilon v) :\equiv (u \sim_{L\epsilon} v)$$

对于 $q : \mathbb{Q}$，我们定义

$$\bar{f}(\mathsf{rat}(q)) :\equiv \mathsf{rat}(f(q))$$

对于 Cauchy 近似 $x : \mathbb{Q}_+ \to \mathbb{R}_c$，我们定义

$$\bar{f}(\mathsf{lim}(x)) :\equiv \mathsf{lim}(\lambda\epsilon.\, \bar{f}(x_{\epsilon/L}))$$

为了使这有意义，我们必须验证 $y :\equiv \lambda\epsilon.\, \bar{f}(x_{\epsilon/L})$ 是一个 Cauchy 近似。然而，这一步的归纳假设是，对于任何 $\delta, \epsilon : \mathbb{Q}_+$，我们有 $\bar{f}(x_\delta) \frown_{\delta+\epsilon} \bar{f}(x_\epsilon)$，即 $\bar{f}(x_\delta) \sim_{L\delta+L\epsilon} \bar{f}(x_\epsilon)$。因此我们有

$$y_\delta \equiv f(x_{\delta/L}) \sim_{\delta+\epsilon} f(x_{\epsilon/L}) \equiv y_\epsilon$$

为了证明分离性，我们简单地观察到 $\forall(\epsilon : \mathbb{Q}_+).\, a \frown_\epsilon b$ 意味着 $\forall(\epsilon : \mathbb{Q}_+).\, a \sim_{L\epsilon} b$，这意味着 $\forall(\epsilon : \mathbb{Q}_+).\, a \sim_\epsilon b$，从而 $a = b$。

为了完成 $(\mathbb{R}_c, \sim)$-递归，还需要验证 $\frown$ 上的四个条件。这基本上相当于证明 $\bar{f}$ 对 $\sim$ 的所有四个构造函数都是 Lipschitz 的。

(i) 当 $u$ 是 $\mathsf{rat}(q)$ 并且 $v$ 是 $\mathsf{rat}(r)$ 时，如果 $-\epsilon < |q - r| < \epsilon$，假设 $f$ 是 Lipschitz 的得出 $f(q) \sim_{L\epsilon} f(r)$，因此 $\bar{f}(\mathsf{rat}(q)) \frown_\epsilon \bar{f}(\mathsf{rat}(r))$ 通过定义。

(ii) 当 $u$ 是 $\mathsf{lim}(x)$ 并且 $v$ 是 $\mathsf{rat}(q)$，如果 $x_\eta \sim_{\epsilon-\eta} \mathsf{rat}(q)$，那么归纳假设是 $\bar{f}(x_\eta) \sim_{L\epsilon-L\eta} \mathsf{rat}(f(q))$，这证明了 $\bar{f}(\mathsf{lim}(x)) \sim_{L\epsilon} \bar{f}(\mathsf{rat}(q))$

(iii) 当 $u$ 是有理数而 $v$ 是极限时，对称情况基本相同。

(iv) 当 $u$ 是 $\mathsf{lim}(x)$ 并且 $v$ 是 $\mathsf{lim}(y)$，其中 $\delta, \eta : \mathbb{Q}_+$ 使得 $x_\delta \sim_{\epsilon-\delta-\eta} y_\eta$，归纳假设是 $\bar{f}(x_\delta) \sim_{L\epsilon-L\delta-L\eta} \bar{f}(y_\eta)$，这证明了 $\bar{f}(\mathsf{lim}(x)) \sim_{L\epsilon} \bar{f}(\mathsf{lim}(y))$ 通过 $\sim$ 的第四个构造函数。

这完成了 $(\mathbb{R}_c, \sim)$-递归，从而完成了 $\bar{f}$ 的构造。所需的等式 $\bar{f}(\mathsf{rat}(q)) \equiv f(q)$ 正是 $(\mathbb{R}_c, \sim)$-递归的第一个计算规则，附加条件 (11.3.13) 正是表明 $\bar{f}$ 是常数为 $L$ 的 Lipschitz 的。 □

在这一点上，如果没有对 $\sim$ 的更好表征，我们已经做到了尽可能多的事情。我们在 $\sim$ 的构造函数中指定了我们希望何种形式的 Cauchy 实数在 $\epsilon$-近似时是 $\epsilon$-close。然而，我们如何知道在所生成的归纳-归纳类型族中，这些是 *唯一的*证明此事实的见证者？我们已经看到，归纳类型族（如恒等类型，参见 §5.8）和更高的归纳类型有包含 “超出它们的内容” 的倾向，因此这不是一个空洞的问题。

为了更精确地表征 $\sim$，我们将定义一个关系族 $\approx_\epsilon$ 在 $\mathbb{R}_c$ 上*递归*地计算，使其能够在构造函数上计算，并证明这个关系族等价于 $\sim_\epsilon$。

Theorem 11.3.16. *存在一个关系族* $\approx : \mathbb{R}_c \to \mathbb{R}_c \to \mathbb{Q}_+ \to \mathsf{Prop}$，*使得*

$$(\mathsf{rat}(q) \approx_\epsilon \mathsf{rat}(r)) :\equiv (-\epsilon < q - r < \epsilon) \tag{11.3.17}$$

$$(\mathsf{rat}(q) \approx_\epsilon \mathsf{lim}(y)) :\equiv \exists(\delta : \mathbb{Q}_+).\, \mathsf{rat}(q) \approx_{\epsilon-\delta} y_\delta \tag{11.3.18}$$

$$(\mathsf{lim}(x) \approx_\epsilon \mathsf{rat}(r)) :\equiv \exists(\delta : \mathbb{Q}_+).\, x_\delta \approx_{\epsilon-\delta} \mathsf{rat}(r) \tag{11.3.19}$$

$$(\mathsf{lim}(x) \approx_\epsilon \mathsf{lim}(y)) :\equiv \exists(\delta, \eta : \mathbb{Q}_+).\, x_\delta \approx_{\epsilon-\delta-\eta} y_\eta \tag{11.3.20}$$

*此外，我们有*

$$(u \approx_\epsilon v) \Leftrightarrow \exists(\theta : \mathbb{Q}_+).\, (u \approx_{\epsilon-\theta} v) \tag{11.3.21}$$

$$(u \approx_\epsilon v) \to (v \sim_\delta w) \to (u \approx_{\epsilon+\delta} w) \tag{11.3.22}$$

$$(u \sim_\epsilon v) \to (v \approx_\delta w) \to (u \approx_{\epsilon+\delta} w) \tag{11.3.23}$$

附加条件 (11.3.21)–(11.3.23) 证明了使得递归定义能够进行。条件 (11.3.21) 被称为**圆滑的**。从右向左阅读得出 $\approx$ 的 **单调性**，

$$(\delta < \epsilon) \wedge (u \approx_\delta v) \Rightarrow (u \approx_\epsilon v)$$

而从左向右阅读则得出 $\approx$ 的 **开放性**，

$$(u \approx_\epsilon v) \Rightarrow \exists(\delta : \mathbb{Q}_+).\, (\delta < \epsilon) \wedge (u \approx_\delta v)$$

条件 (11.3.22) 和 (11.3.23) 是三角不等式的形式，表明 $\approx$ 在两侧是 $\sim$ 的 “模”。

证明. 我们将通过双 $(\mathbb{R}_{\mathsf{c}}, \sim)$-递归定义 $\approx : \mathbb{R}_{\mathsf{c}} \to \mathbb{R}_{\mathsf{c}} \to \mathbb{Q}_+ \to \mathsf{Prop}$。首先我们将应用 $(\mathbb{R}_{\mathsf{c}}, \sim)$-递归，其余象是 $\mathbb{R}_{\mathsf{c}} \to \mathbb{Q}_+ \to \mathsf{Prop}$ 的子集，包含那些圆滑并满足一种适当形式的三角不等式的谓词族。将这些谓词视为二元关系的一半，我们将它们写为 $(u, \epsilon) \mapsto (\diamondsuit \approx_\epsilon u)$，其中符号 $\diamondsuit$ 指整个关系。现在我们可以精确地写出 $A$

$$\begin{aligned} A :\equiv \Bigg\{ \diamondsuit : \mathbb{R}_{\mathsf{c}} \to \mathbb{Q}_+ \to \mathsf{Prop} \;\Bigg|\; & \Big(\forall(u : \mathbb{R}_{\mathsf{c}}).\, \forall(\epsilon : \mathbb{Q}_+).\, \big((\diamondsuit \approx_\epsilon u) \Leftrightarrow \exists(\theta : \mathbb{Q}_+).\, (\diamondsuit \approx_{\epsilon-\theta} u)\big)\Big) \\ & \wedge \Big(\forall(u, v : \mathbb{R}_{\mathsf{c}}).\, \forall(\eta, \epsilon : \mathbb{Q}_+).\, (u \sim_\epsilon v) \to \\ & \qquad \big((\diamondsuit \approx_\eta u) \to (\diamondsuit \approx_{\eta+\epsilon} v)\big) \wedge \big((\diamondsuit \approx_\eta v) \to (\diamondsuit \approx_{\eta+\epsilon} u)\big)\Big) \Bigg\} \end{aligned}$$

与通常的子集一样，我们将使用相同的符号来表示 $A$ 的一个元素及其第一个分量 $\diamondsuit$。作为 $(\mathbb{R}_{\mathsf{c}}, \sim)$-递归所需的关系族，我们考虑以下内容，这将确保三角不等式的另一种形式：

$$(\diamondsuit \frown_\epsilon \heartsuit) :\equiv \forall(u : \mathbb{R}_{\mathsf{c}}).\, \forall(\eta : \mathbb{Q}_+).\, ((\diamondsuit \approx_\eta u) \to (\heartsuit \approx_{\epsilon+\eta} u)) \wedge ((\heartsuit \approx_\eta u) \to (\diamondsuit \approx_{\epsilon+\eta} u))$$

我们观察到这些关系是分离的。因为假设 $\forall(\epsilon : \mathbb{Q}_+).\, (\diamondsuit \frown_\epsilon \heartsuit)$ 要证明 $\diamondsuit = \heartsuit$，只需证明对于所有 $u : \mathbb{R}_{\mathsf{c}}$，$(\diamondsuit \approx_\epsilon u) \Leftrightarrow (\heartsuit \approx_\epsilon u)$。但 $\diamondsuit \approx_\epsilon u$ 意味着对于某个 $\theta$，$\diamondsuit \approx_{\epsilon-\theta} u$，根据圆滑性，并且 $\diamondsuit \frown_\epsilon \heartsuit$ 意味着 $\heartsuit \approx_\epsilon u$；反之亦然。
现在，递归原则所需的前两个数据如下。

- 对于任何 $q : \mathbb{Q}$，我们必须给出 $A$ 的一个元素，我们将其记作 $(\mathsf{rat}(q) \approx_{(-)} -)$。
- 对于任何 Cauchy 近似 $x$，如果我们假设定义了函数 $\mathbb{Q}_+ \to A$，我们将其记作 $\epsilon \mapsto (x_\epsilon \approx_{(-)} -)$，其具有以下属性：
$$\forall(u : \mathbb{R}_{\mathsf{c}}).\, \forall(\delta, \epsilon, \eta : \mathbb{Q}_+).\, (x_\delta \approx_\eta u) \to (x_\epsilon \approx_{\eta+\delta+\epsilon} u) \tag{11.3.24}$$
我们必须给出 $A$ 的一个元素，我们将其写作 $(\mathsf{lim}(x) \approx_{(-)} -)$。

在这两种情况下，我们通过使用嵌套 $(\mathbb{R}_{\mathsf{c}}, \sim)$-递归给出所需定义，其余象为由圆滑的命题构成的 $\mathbb{Q}_+ \to \mathsf{Prop}$ 的子集。将这些命题视为二元关系的零一半，我们将它们写作 $\epsilon \mapsto (\bullet \approx_\epsilon \triangle)$，符号 $\triangle$ 指整个关系族。现在我们可以精确地写出这些内递归的余象：

$$C :\equiv \Big\{ \triangle : \mathbb{Q}_+ \to \mathsf{Prop} \;\Big|\; \forall(\epsilon : \mathbb{Q}_+).\, \Big((\bullet \approx_\epsilon \triangle) \Leftrightarrow \exists(\theta : \mathbb{Q}_+).\, (\bullet \approx_{\epsilon-\theta} \triangle)\Big) \Big\}$$

我们将所需的关系族定义为三角不等式的剩余部分：

$$(\triangle \smile_\epsilon \square) :\equiv \forall(\eta : \mathbb{Q}_+).\, ((\bullet \approx_\eta \triangle) \to (\bullet \approx_{\epsilon+\eta} \square)) \wedge ((\bullet \approx_\eta \square) \to (\bullet \approx_{\epsilon+\eta} \triangle))$$

通过与 $\frown$ 类似的论证，这些关系是分离的，使用 $C$ 的所有元素的圆滑性。
注意，如果这样的内递归成功，它将生成一个谓词族 $\diamondsuit : \mathbb{R}_{\mathsf{c}} \to \mathbb{Q}_+ \to \mathsf{Prop}$，这些谓词族是圆滑的（因为它们在 $\mathbb{Q}_+ \to \mathsf{Prop}$ 中的象属于 $C$）并且满足

$$\forall(u, v : \mathbb{R}_{\mathsf{c}}).\, \forall(\epsilon : \mathbb{Q}_+).\, (u \sim_\epsilon v) \to \big((\diamondsuit \approx_{(-)} u) \smile_\epsilon (\diamondsuit \approx_{(-)} u)\big)$$

展开 $\smile$ 的定义，这正是使得 $\diamondsuit$ 属于 $A$ 的第三个条件；因此，这正是我们需要的。
此时我们可以给出定义 (11.3.17)–(11.3.20)，作为两个内递归的前两个子句，分别对应于有理点和极限。在每种情况下，我们必须验证关系是圆滑的，因此属于 $C$。在有理-有理的情况下 (11.3.17)，这是显然的，而在其他情况下则来自归纳假设。（在 (11.3.18) 中，相关的归纳假设是 $(\mathsf{rat}(q) \approx_{(-)} y_\delta) : C$，而在 (11.3.19) 和 (11.3.20) 中，它是 $(x_\delta \approx_{(-)} -) : A$。）
余下的子递归数据包括证明 (11.3.17)–(11.3.20) 满足相对于 $\sim$ 构造函数的右侧三角不等式。有八种情况——每个子递归中的四个——对应于 $u$、$v$ 和 $w$ 在 (11.3.22) 中可以选择为有理点或极限的八种方式。首先考虑 $u$ 是 $\mathsf{rat}(q)$ 的情况。

(i) 假设 $\mathsf{rat}(q) \approx_\phi \mathsf{rat}(r)$ 并且 $-\epsilon < |r-s| < \epsilon$，我们必须证明 $\mathsf{rat}(q) \approx_{\phi+\epsilon} \mathsf{rat}(s)$。但根据 $\approx$ 的定义，这简化为有理数的三角不等式。

(ii) 我们假设 $\phi, \epsilon, \delta : \mathbb{Q}_+$ 使得 $\mathsf{rat}(q) \approx_\phi \mathsf{rat}(r)$ 并且 $\mathsf{rat}(r) \sim_{\epsilon-\delta} y_\delta$，并且归纳地

$$\forall(\psi : \mathbb{Q}_+).\, (\mathsf{rat}(q) \approx_\psi \mathsf{rat}(r)) \to (\mathsf{rat}(q) \approx_{\psi+\epsilon-\delta} y_\delta) \tag{11.3.25}$$

我们还假设 $\psi, \delta \mapsto (\mathsf{rat}(q) \approx_\psi y_\delta)$ 是相对于 $\smile$ 的 Cauchy 近似，即

$$\forall(\psi, \xi zeta : \mathbb{Q}_+).\, (\mathsf{rat}(q) \approx_\psi y_\xi) \to (\mathsf{rat}(q) \approx_{\psi+\xi+\zeta} y_\zeta) \tag{11.3.26}$$

尽管在这种情况下我们不需要这种假设。事实上，(11.3.25) 直接给出了 $\mathsf{rat}(q) \approx_{\phi+\epsilon-\delta} y_\delta$，从而通过 $\approx$ 的定义 $\mathsf{rat}(q) \approx_{\phi+\epsilon} \mathsf{lim}(y)$。

(iii) 我们假设 $\phi, \epsilon, \delta : \mathbb{Q}_+$ 使得 $\mathsf{rat}(q) \approx_\phi \mathsf{lim}(y)$ 并且 $y_\delta \sim_{\epsilon-\delta} \mathsf{rat}(r)$，并且归纳地

$$\forall(\psi : \mathbb{Q}_+).\, (\mathsf{rat}(q) \approx_\psi y_\delta) \to (\mathsf{rat}(q) \approx_{\psi+\epsilon-\delta} \mathsf{rat}(r)) \tag{11.3.27}$$

$$\forall(\psi, \xi, \zeta : \mathbb{Q}_+).\, (\mathsf{rat}(q) \approx_\psi y_\xi) \to (\mathsf{rat}(q) \approx_{\psi+\xi+\zeta} y_\zeta) \tag{11.3.28}$$

根据定义，$\mathsf{rat}(q) \approx_\phi \mathsf{lim}(y)$ 意味着我们有 $\theta : \mathbb{Q}_+$ 使得 $\mathsf{rat}(q) \approx_{\phi-\theta} y_\theta$。根据假设 (11.3.28)，因此我们还得出 $\mathsf{rat}(q) \approx_{\phi+\delta} y_\delta$，然后根据 (11.3.27) 得出 $\mathsf{rat}(q) \approx_{\phi+\epsilon} \mathsf{rat}(r)$，如所需。

(iv) 我们假设 $\phi, \epsilon, \delta eta : \mathbb{Q}_+$ 使得 $\mathsf{rat}(q) \approx_\phi \mathsf{lim}(y)$ 并且 $y_\delta \sim_{\epsilon-\delta-\eta} z_\eta$，并且归纳地

$$\forall(\psi : \mathbb{Q}_+).\, (\mathsf{rat}(q) \approx_\psi y_\delta) \to (\mathsf{rat}(q) \approx_{\psi+\epsilon-\delta-\eta} z_\eta) \tag{11.3.29}$$

$$\forall(\psi, \xi, \zeta : \mathbb{Q}_+).\, (\mathsf{rat}(q) \approx_\psi y_\xi) \to (\mathsf{rat}(q) \approx_{\psi+\xi+\zeta} y_\zeta) \tag{11.3.30}$$

$$\forall(\psi, \xi, \zeta : \mathbb{Q}_+).\, (\mathsf{rat}(q) \approx_\psi z_\xi) \to (\mathsf{rat}(q) \approx_{\psi+\xi+\zeta} z_\zeta) \tag{11.3.31}$$

同样地，$\mathsf{rat}(q) \approx_\phi \mathsf{lim}(y)$ 意味着我们有 $\xi : \mathbb{Q}_+$ 使得 $\mathsf{rat}(q) \approx_{\phi-\xi} y_\xi$，而 (11.3.30) 然后意味着 $\mathsf{rat}(q) \approx_{\phi+\delta} y_\delta$ 并且 (11.3.29) 意味着 $\mathsf{rat}(q) \approx_{\phi+\epsilon-\eta} z_\eta$。但根据 $\approx$ 的定义，这意味着 $\mathsf{rat}(q) \approx_{\phi+\epsilon} \mathsf{lim}(z)$ 如所需。

现在我们继续讨论 $u$ 是 $\mathsf{lim}(x)$ 的情况，其中 $x$ 是 Cauchy 近似。在这种情况下，$(\mathsf{lim}(x) \approx_{(-)} -) : A$ 的定义的背景归纳假设是我们有 $(x_\delta \approx_{(-)} -) : A$，因此除了是圆滑的外，它们还满足右侧三角不等式。

(v) 假设 $\mathsf{lim}(x) \approx_\phi \mathsf{rat}(r)$ 并且 $-\epsilon < |r-s| < \epsilon$，我们必须证明 $\mathsf{lim}(x) \approx_{\phi+\epsilon} \mathsf{rat}(s)$。根据 $\approx$ 的定义，前者意味着 $x_\delta \approx_{\phi-\delta} \mathsf{rat}(r)$，因此上述三角不等式得出 $x_\delta \approx_{\epsilon+\phi-\delta} \mathsf{rat}(s)$，从而 $\mathsf{lim}(x) \approx_{\phi+\epsilon} \mathsf{rat}(s)$ 如所需。

(vi) 我们假设 $\phi, \epsilon delta : \mathbb{Q}_+$ 使得 $\mathsf{lim}(x) \approx_\phi \mathsf{rat}(r)$ 并且 $\mathsf{rat}(r) \sim_{\epsilon-\delta} y_\delta$，并且有两个不需要的归纳假设。根据定义，我们有 $\eta : \mathbb{Q}_+$ 使得 $x_\eta \approx_{\phi-\eta} \mathsf{rat}(r)$，因此归纳三角不等式得出 $x_\eta \approx_{\phi+\epsilon-\eta-\delta} y_\delta$。然后 $\approx$ 的定义立即得出 $\mathsf{lim}(x) \approx_{\phi+\epsilon} \mathsf{lim}(y)$。

(vii) 我们假设 $\phi, \epsilon delta : \mathbb{Q}_+$ 使得 $\mathsf{lim}(x) \approx_\phi \mathsf{lim}(y)$ 并且 $y_\delta \sim_{\epsilon-\delta} \mathsf{rat}(r)$，并且有两个不需要的归纳假设。根据定义，我们有 $\xi theta : \mathbb{Q}_+$ 使得 $x_\xi \approx_{\phi-\xi-\theta} y_\theta$。由于 $y$ 是 Cauchy 近似，我们有 $y_\theta \sim_{\theta+\delta} y_\delta$，因此归纳三角不等式得出 $x_\xi \approx_{\phi+\delta-\xi} y_\delta$ 然后 $x_\xi \sim_{\phi+\epsilon-\xi} \mathsf{rat}(r)$。然后 $\approx$ 的定义得出 $\mathsf{lim}(x) \approx_{\phi+\epsilon} \mathsf{rat}(r)$，如所需。

(viii) 最后，我们假设 $\phi, \epsilon deltaeta : \mathbb{Q}_+$ 使得 $\mathsf{lim}(x) \approx_\phi \mathsf{lim}(y)$ 并且 $y_\delta \sim_{\epsilon-\delta-\eta} z_\eta$。然后如前所述，我们有 $\xi theta : \mathbb{Q}_+$ 使得 $x_\xi \approx_{\phi-\xi-\theta} y_\theta$，并且如前所述的两个三角不等式足够了。

这完成了两个内递归，从而完成了 $(\mathsf{lim}(x) \approx_{(-)} -) : A$ 的定义。由于所有都是 $A$ 的元素，它们是圆滑的，并且满足相对于 $\sim$ 的右侧三角不等式。余下的任务是验证与 $\frown$ 相关的条件，即这些关系满足相对于 $\sim$ 构造函数的左侧三角不等式。四种情况对应于在 (11.3.23) 中为 $u$ 和 $v$ 选择有理点或极限的四种选择，并且由于它们都是纯命题，我们可以应用 $\mathbb{R}_\mathsf{c}$ 归纳，并假设 $w$ 也是有理点或极限。这产生了另外八种情况，其证明基本与前述证明相同；因此我们不再让读者经历它们。 □

我们现在可以证明：

Theorem 11.3.32. *对于任意的 $u, v : \mathbb{R}_\mathsf{c}$ 和 $\epsilon : \mathbb{Q}_+$，我们有 $(u \sim_\epsilon v) = (u \approx_\epsilon v)$。*

证明. 由于两者都是纯命题，足以证明双向蕴涵。对于左到右方向，我们使用 $\sim$-归纳，应用于 $C(u,v,\epsilon) :\equiv (u \approx_\epsilon v)$。因此，足以考虑 $\sim$ 的四个构造函数。在每种情况下，$u$ 和 $v$ 专门化为有理点或极限，因此 $\approx$ 的定义计算并且归纳假设总是适用。
对于右到左方向，我们使用 $\mathbb{R}_c$-归纳来假设 $u$ 和 $v$ 是有理点或极限，允许 $\approx$ 计算。但现在 $\approx$ 的定义以及归纳假设提供了 $\sim$ 构造函数所需的数据。 □

略微夸张地说，可以将 $\approx$ 称为 $\sim$ 的 "编码"，上述证明的两个方向分别是 encode 和 decode 。根据 $\approx$ 的定义，从 Theorem 11.3.32 我们得出等价关系

$$\begin{aligned}(\mathsf{rat}(q) \sim_\epsilon \mathsf{rat}(r)) &= (-\epsilon < q - r < \epsilon)\\ (\mathsf{rat}(q) \sim_\epsilon \mathsf{lim}(y)) &= \exists(\delta : \mathbb{Q}_+).\, \mathsf{rat}(q) \sim_{\epsilon-\delta} y_\delta\\ (\mathsf{lim}(x) \sim_\epsilon \mathsf{rat}(r)) &= \exists(\delta : \mathbb{Q}_+).\, x_\delta \sim_{\epsilon-\delta} \mathsf{rat}(r)\\ (\mathsf{lim}(x) \sim_\epsilon \mathsf{lim}(y)) &= \exists(\delta,\eta : \mathbb{Q}_+).\, x_\delta \sim_{\epsilon-\delta-\eta} y_\eta\end{aligned}$$

我们的证明还提供了以下附加信息。

Corollary 11.3.33. $\sim$ 是圆滑的并且满足三角不等式:

$$(u \sim_\epsilon v) \simeq \exists(\theta : \mathbb{Q}_+).\, u \sim_{\epsilon-\theta} v \tag{11.3.34}$$

$$(u \sim_\epsilon v) \to (v \sim_\delta w) \to (u \sim_{\epsilon+\delta} w) \tag{11.3.35}$$

有了三角不等式，我们可以证明 Cauchy 近似的 "极限" 实际上表现得像极限。

Lemma 11.3.36. 对于任意的 $u : \mathbb{R}_c$，Cauchy 近似 $y$，和 $\epsilon,\delta : \mathbb{Q}_+$，如果 $u \sim_\epsilon y_\delta$ 那么 $u \sim_{\epsilon+\delta} \mathsf{lim}(y)$。

证明. 我们对 $u$ 使用 $\mathbb{R}_c$-归纳。如果 $u$ 是 $\mathsf{rat}(q)$，那么这正是 $\sim$ 的第二个构造函数。现在假设 $u$ 是 $\mathsf{lim}(x)$，并且每个 $x_\eta$ 具有以下性质：对于任意 $y,\epsilon,\delta$，如果 $x_\eta \sim_\epsilon y_\delta$ 那么 $x_\eta \sim_{\epsilon+\delta} \mathsf{lim}(y)$。特别地，对于 $y :\equiv x$ 和 $\delta :\equiv \eta$ 在此假设下，我们得出对于任意 $\eta,\theta$，$x_\eta \sim_{\eta+\theta} \mathsf{lim}(x)$。
现在令 $y,\epsilon,\delta$ 任意，并假设 $\mathsf{lim}(x) \sim_\epsilon y_\delta$。通过圆滑性，存在一个 $\theta$ 使得 $\mathsf{lim}(x) \sim_{\epsilon-\theta} y_\delta$。然后，根据上述观察，对于任何 $\eta$，我们有 $x_\eta \sim_{\eta+\theta/2} \mathsf{lim}(x)$，因此通过三角不等式得出 $x_\eta \sim_{\epsilon+\eta-\theta/2} y_\delta$。因此，$\sim$ 的第四个构造函数得出 $\mathsf{lim}(x) \sim_{\epsilon+2\eta+\delta-\theta/2} \mathsf{lim}(y)$。因此，如果我们选择 $\eta :\equiv \theta/4$，则结果成立。 □

Lemma 11.3.37. 对于任意的 Cauchy 近似 $y$ 和任意的 $\delta,\eta : \mathbb{Q}_+$ 我们有 $y_\delta \sim_{\delta+\eta} \mathsf{lim}(y)$。

证明. 在前一个引理中取 $u :\equiv y_\delta$ 和 $\epsilon :\equiv \eta$。 □

Remark 11.3.38. 我们可能期望有 $y_\delta \sim_\delta \mathsf{lim}(y)$，但在某些例子中这会失败。例如，考虑定义为 $x_\epsilon :\equiv \epsilon$ 的 $x$。它的极限显然是 0，但我们没有 $|\epsilon - 0| < \epsilon$，只有 $\leq$。
作为一个应用，Lemma 11.3.37 使我们能够证明 Lemma 11.3.15 中 Lipschitz 函数的扩展是唯一的。

Lemma 11.3.39. 令 $f,g : \mathbb{R}_c \to \mathbb{R}_c$ 是连续的，意思是

$$\forall(u : \mathbb{R}_c).\forall(\epsilon : \mathbb{Q}_+).\exists(\delta : \mathbb{Q}_+).\forall(v : \mathbb{R}_c).\,(u \sim_\delta v) \to (f(u) \sim_\epsilon f(v))$$

并且类似地对于 $g$。如果对于所有 $q : \mathbb{Q}$ 有 $f(\mathsf{rat}(q)) = g(\mathsf{rat}(q))$，那么 $f = g$。

证明. 我们通过 $\mathbb{R}_c$-归纳证明对于所有 $u$ 有 $f(u) = g(u)$。有理数情况只是假设。因此，假设对于所有 $\delta$ 有 $f(x_\delta) = g(x_\delta)$。我们将证明对于所有 $\epsilon$，$f(\mathsf{lim}(x)) \sim_\epsilon g(\mathsf{lim}(x))$，因此 $\mathbb{R}_c$ 的路径构造函数适用。
由于 $f$ 和 $g$ 是连续的，存在 $\theta,\eta$ 使得对于所有 $v$，我们有

$$(\mathsf{lim}(x) \sim_\theta v) \to (f(\mathsf{lim}(x)) \sim_{\epsilon/2} f(v))$$

$$(\mathsf{lim}(x) \sim_\eta v) \to (g(\mathsf{lim}(x)) \sim_{\epsilon/2} g(v))$$

选择 $\delta < \min(\theta,\eta)$，通过 Lemma 11.3.37 我们有 $\mathsf{lim}(x) \sim_\theta y_\delta$ 和 $\mathsf{lim}(x) \sim_\eta y_\delta$。因此

$$f(\mathsf{lim}(x)) \sim_{\epsilon/2} f(y_\delta) = g(y_\delta) \sim_{\epsilon/2} g(\mathsf{lim}(x))$$

因此通过三角不等式 $f(\mathsf{lim}(x)) \sim_\epsilon g(\mathsf{lim}(x))$。 □

### 11.3.3 Cauchy 实数的代数结构

我们首先定义加法结构 $(\mathbb{R}_c, 0, +, -)$。显然，加法单位元素 0 就是 $\mathsf{rat}(0)$，而加法逆元 $-:\mathbb{R}_c \to \mathbb{R}_c$ 是通过加法逆元 $-:\mathbb{Q}\to\mathbb{Q}$ 的扩展得到的，使用 Lemma 11.3.15 Lipschitz 常数为 1。对于加法我们必须做更多工作。

Lemma 11.3.40. *假设 $f:\mathbb{Q}\times\mathbb{Q}\to\mathbb{Q}$ 满足，对于所有 $q,r,s:\mathbb{Q}$，*

$$|f(q,s)-f(r,s)| \le |q-r| \qquad 和 \qquad |f(q,r)-f(q,s)| \le |r-s|$$

*那么存在一个函数 $\bar{f}:\mathbb{R}_c\times\mathbb{R}_c\to\mathbb{R}_c$，使得对于所有 $q,r:\mathbb{Q}$，有 $\bar{f}(\mathsf{rat}(q),\mathsf{rat}(r)) = f(q,r)$。此外，对于所有 $u,v,w:\mathbb{R}_c$ 和 $q:\mathbb{Q}_+$，*

$$u \sim_\epsilon v \Rightarrow \bar{f}(u,w) \sim_\epsilon \bar{f}(v,w) \quad 和 \quad v \sim_\epsilon w \Rightarrow \bar{f}(u,v) \sim_\epsilon \bar{f}(u,w)$$

*证明.* 我们使用 $(\mathbb{R}_c,\sim)$-递归来构造 $\bar{f}$ 的柯里化形式，作为映射 $\mathbb{R}_c\to A$，其中 $A$ 是非扩展实值函数的空间：

$$A :\equiv \{\, h:\mathbb{R}_c\to\mathbb{R}_c \mid \forall(\epsilon:\mathbb{Q}_+).\forall(u,v:\mathbb{R}_c).\, u\sim_\epsilon v \Rightarrow h(u)\sim_\epsilon h(v) \,\}.$$

我们还需要一个合适的 $\frown_\epsilon$ 在 $A$ 上，我们定义为

$$(h \frown_\epsilon k) :\equiv \forall(u:\mathbb{R}_c).h(u)\sim_\epsilon k(u)$$

显然，如果 $\forall(\epsilon:\mathbb{Q}_+).h\frown_\epsilon k$ 那么对所有 $u:\mathbb{R}_c$ 有 $h(u)=k(u)$，因此 $\frown$ 是分离的。
对于基准情况，我们定义 $\bar{f}(\mathsf{rat}(q)):A$，其中 $q:\mathbb{Q}$，作为 Lipschitz 映射 $\lambda r.f(q,r)$ 从 $\mathbb{Q}\to\mathbb{Q}$ 到 $\mathbb{R}_c\to\mathbb{R}_c$ 的扩展，使用 Lipschitz 常数为 1 的 Lemma 11.3.15 构造。接下来，对于一个 Cauchy 近似 $x$，我们定义 $\bar{f}(\lim(x)):\mathbb{R}_c\to\mathbb{R}_c$ 为

$$\bar{f}(\lim(x))(v) :\equiv \lim(\lambda\epsilon.\bar{f}(x_\epsilon)(v))$$

为了使这个定义有效，$\lambda\epsilon.\bar{f}(x_\epsilon)(v)$ 应该是一个 Cauchy 近似，因此考虑任意 $\delta,\epsilon:\mathbb{Q}$。然后根据假设 $\bar{f}(x_\delta)\frown_{\delta+\epsilon}\bar{f}(x_\epsilon)$，因此 $\bar{f}(x_\delta)(v)\sim_{\delta+\epsilon}\bar{f}(x_\epsilon)(v)$。此外，$\bar{f}(\lim(x))$ 是非扩展的，因为根据归纳假设 $\bar{f}(x_\epsilon)$ 是这样的。事实上，如果 $u\sim_\epsilon v$，那么对于所有 $\epsilon:\mathbb{Q}$，

$$\bar{f}(x_{\epsilon/3})(u)\sim_{\epsilon/3}\bar{f}(x_{\epsilon/3})(v)$$

因此通过 $\sim$ 的第四个构造函数 $\bar{f}(\lim(x))(u)\sim_\epsilon\bar{f}(\lim(x))(v)$。
我们还需要检查四个条件，我们来说明其中一个。假设 $\epsilon:\mathbb{Q}_+$ 并且对于某个 $\delta:\mathbb{Q}_+$ 有 $\mathsf{rat}(q)\sim_{\epsilon-\delta} y_\delta$ 和 $\bar{f}(\mathsf{rat}(q))\frown_{\epsilon-\delta}\bar{f}(y_\delta)$。为了证明 $\bar{f}(\mathsf{rat}(q))\frown_\epsilon\bar{f}(\lim(y))$，考虑任意 $v:\mathbb{R}_c$ 并观察到

$$\bar{f}(\mathsf{rat}(q))(v)\sim_{\epsilon-\delta}\bar{f}(y_\delta)(v)$$

因此，通过 $\sim$ 的第二个构造函数，我们有 $\bar{f}(\mathsf{rat}(q))(v)\sim_\epsilon\bar{f}(\lim(y))(v)$ 如所需。 □

我们可以将 Lemma 11.3.40 应用于任何在每个变量中分别是非扩展的二元有理函数。加法就是这样的函数，因此我们得到了 $+:\mathbb{R}_c\times\mathbb{R}_c\to\mathbb{R}_c$。此外，只要我们要求它在每个变量中是非扩展的，这个扩展就是唯一的，并且与一元情况下相同，有理数上的恒等式扩展为实数上的恒等式。由于非扩展映射的复合再次是非扩展的，我们可以得出加法满足通常的性质，例如交换律和结合律。因此，$(\mathbb{R}_c,0,+,-)$ 是一个交换群。
我们还可以将 Lemma 11.3.40 应用于函数 $\min:\mathbb{Q}\times\mathbb{Q}\to\mathbb{Q}$ 和 $\max:\mathbb{Q}\times\mathbb{Q}\to\mathbb{Q}$，这将 $\mathbb{R}_c$ 变成一个格。$\mathbb{R}_c$ 上的偏序 $\le$ 以 max 的形式定义为

$$(u\le v) :\equiv (\max(u,v)=v)$$

关系 $\le$ 是偏序，因为它在 $\mathbb{Q}$ 上是这样的，并且偏序的公理可以用 min 和 max 表示为方程，因此它们转移到 $\mathbb{R}_c$。

另一个通过相同方法扩展到 $\mathbb{R}_c$ 的函数是绝对值 $|-|$。同样，它具有预期的性质，因为它们从 $\mathbb{Q}$ 转移到 $\mathbb{R}_c$。
从 $\leq$ 我们得到严格顺序 $<$，其定义为

$$(u < v) :\equiv \exists(q,r:\mathbb{Q}).\,(u \leq \mathsf{rat}(q)) \wedge (q < r) \wedge (\mathsf{rat}(r) \leq v)$$

即，当仅存在一对有理数 $q < r$ 使得 $x \leq \mathsf{rat}(q)$ 且 $\mathsf{rat}(r) \leq v$ 时，$u < v$ 成立。不难检查 $<$ 是反身和传递的，并且具有有序域预期的其他性质。阿基米德原则直接来自于 $<$ 的定义。

Theorem 11.3.41 (RC 的阿基米德原理). *对于每个 $u,v:\mathbb{R}_c$ 使得 $u < v$，仅存在 $q:\mathbb{Q}$ 使得 $u < q < v$。*

*证明.* 从 $u < v$ 我们仅得到 $r,s:\mathbb{Q}$ 使得 $u \leq r < s \leq v$，我们可以取 $q :\equiv (r+s)/2$。 □

我们现在有足够的结构来用标准概念表达 $u \sim_\epsilon v$。

Lemma 11.3.42. *如果 $q:\mathbb{Q}$ 和 $u:\mathbb{R}_c$ 满足 $u \leq \mathsf{rat}(q)$，那么对于任何 $v:\mathbb{R}_c$ 和 $\epsilon:\mathbb{Q}_+$，如果 $u \sim_\epsilon v$ 那么 $v \leq \mathsf{rat}(q+\epsilon)$。*

*证明.* 注意函数 $\max(\mathsf{rat}(q),-):\mathbb{R}_c \to \mathbb{R}_c$ 是 Lipschitz 的，常数为 1。首先考虑 $u = \mathsf{rat}(r)$ 为有理数的情况。对于此，我们使用归纳法 $v$。如果 $v$ 是有理数，那么该陈述是显然的。如果 $v$ 是 $\mathsf{lim}(y)$，我们假设归纳地，对于任何 $\epsilon,\delta$，如果 $\mathsf{rat}(r) \sim_\epsilon y_\delta$ 那么 $y_\delta \leq \mathsf{rat}(q+\epsilon)$，即 $\max(\mathsf{rat}(q+\epsilon),y_\delta) = \mathsf{rat}(q+\epsilon)$。现在假设 $\epsilon$ 并且 $\mathsf{rat}(r) \sim_\epsilon \mathsf{lim}(y)$，我们有 $\theta$ 使得 $\mathsf{rat}(r) \sim_{\epsilon-\theta} \mathsf{lim}(y)$，因此 $\mathsf{rat}(r) \sim_\epsilon y_\delta$ 只要 $\delta < \theta$。因此，归纳假设给出了 $\max(\mathsf{rat}(q+\epsilon),y_\delta) = \mathsf{rat}(q+\epsilon)$ 对于这样的 $\delta$。但根据定义，

$$\max(\mathsf{rat}(q+\epsilon),\mathsf{lim}(y)) \equiv \mathsf{lim}(\lambda\delta.\,\max(\mathsf{rat}(q+\epsilon),y_\delta))$$

由于最终常数 Cauchy 近似的极限是那个常数，我们有

$$\max(\mathsf{rat}(q+\epsilon),\mathsf{lim}(y)) = \mathsf{rat}(q+\epsilon)$$

因此 $\mathsf{lim}(y) \leq \mathsf{rat}(q+\epsilon)$。
现在考虑一个一般的 $u:\mathbb{R}_c$。由于 $u \leq \mathsf{rat}(q)$ 意味着 $\max(\mathsf{rat}(q),u) = \mathsf{rat}(q)$，假设 $u \sim_\epsilon v$ 和 $\max(\mathsf{rat}(q),-)$ 的 Lipschitz 性质意味着 $\max(\mathsf{rat}(q),v) \sim_\epsilon \mathsf{rat}(q)$。因此，$\mathsf{rat}(q) \leq \mathsf{rat}(q)$ 的第一个情况意味着 $\max(\mathsf{rat}(q),v) \leq \mathsf{rat}(q+\epsilon)$，因此通过 $\leq$ 的传递性 $v \leq \mathsf{rat}(q+\epsilon)$。 □

Lemma 11.3.43. *假设 $q:\mathbb{Q}$ 和 $u:\mathbb{R}_c$ 满足 $u < \mathsf{rat}(q)$。那么：*

(i) *对于任何 $v:\mathbb{R}_c$ 和 $\epsilon:\mathbb{Q}_+$，如果 $u \sim_\epsilon v$ 那么 $v < \mathsf{rat}(q+\epsilon)$。*
(ii) *存在 $\epsilon:\mathbb{Q}_+$ 使得对于任何 $v:\mathbb{R}_c$，如果 $u \sim_\epsilon v$ 我们有 $v < \mathsf{rat}(q)$。*

*证明.* 根据定义，$u < \mathsf{rat}(q)$ 意味着存在 $r:\mathbb{Q}$ 使得 $r < q$ 并且 $u \leq \mathsf{rat}(r)$。然后根据 Lemma 11.3.42，对于任何 $\epsilon$，如果 $u \sim_\epsilon v$ 那么 $v \leq \mathsf{rat}(r+\epsilon)$。结论 (i) 立即得出，因为 $r+\epsilon < q+\epsilon$，而对于 (ii) 我们可以取任何 $\epsilon < q-r$。 □

我们现在可以证明辅助关系 $\sim$ 确实如我们所想的那样。

Theorem 11.3.44. *对于所有 $u,v:\mathbb{R}_c$ 和 $\epsilon:\mathbb{Q}_+$，$(u \sim_\epsilon v) \simeq (|u-v| < \mathsf{rat}(\epsilon))$。*

*证明.* 减法和绝对值的 Lipschitz 性质意味着如果 $u \sim_\epsilon v$，那么 $|u-v| \sim_\epsilon |u-u| = 0$。因此，对于从左到右的方向，足以证明如果 $u \sim_\epsilon 0$，那么 $|u| < \mathsf{rat}(\epsilon)$。我们通过 $\mathbb{R}_c$-归纳法证明 $u$。
如果 $u$ 是有理数，那么该陈述立即得出，因为绝对值和顺序扩展了 $\mathbb{Q}_+$ 上的标准。如果 $u$ 是 $\mathsf{lim}(x)$，那么通过圆滑性我们有 $\theta:\mathbb{Q}_+$，使得 $\mathsf{lim}(x) \sim_{\epsilon-\theta} 0$。因此，通过三角不等式，我们有 $x_{\theta/3} \sim_{\epsilon-2\theta/3} 0$，因此归纳假设得出 $|x_{\theta/3}| < \mathsf{rat}(\epsilon-2\theta/3)$。但是 $x_{\theta/3} \sim_{2\theta/3} \mathsf{lim}(x)$，因此根据 Lipschitz 性质 $|x_{\theta/3}| \sim_{2\theta/3} |\mathsf{lim}(x)|$，因此 Lemma 11.3.43(i) 意味着 $|\mathsf{lim}(x)| < \mathsf{rat}(\epsilon)$。
另一个方向上，我们使用 $\mathbb{R}_c$-归纳法证明 $u$ 和 $v$。如果两者都是有理数，这是 $\sim$ 的第一个构造函数。

如果 $u$ 是 $\mathsf{rat}(q)$ 而 $v$ 是 $\mathsf{lim}(y)$，我们归纳假设，对于任何 $\epsilon, \delta$，如果 $|\mathsf{rat}(q) - y_\delta| < \mathsf{rat}(\epsilon)$ 那么 $\mathsf{rat}(q) \sim_\epsilon y_\delta$。固定一个 $\epsilon$ 使得 $|\mathsf{rat}(q) - \mathsf{lim}(y)| < \mathsf{rat}(\epsilon)$。由于 $\mathbb{Q}$ 在 $\mathbb{R}_\mathsf{c}$ 中是序致密的，存在 $\theta < \epsilon$ 使得 $|\mathsf{rat}(q) - \mathsf{lim}(y)| < \mathsf{rat}(\theta)$。现在对于任何 $\delta, \eta$，我们有 $\mathsf{lim}(y) \sim_{2\delta} y_\delta$，因此根据 Lipschitz 性质

$$|\mathsf{rat}(q) - \mathsf{lim}(y)| \sim_{\delta+\eta} |\mathsf{rat}(q) - y_\delta|$$

因此，根据 Lemma 11.3.43(i)，我们有 $|\mathsf{rat}(q) - y_\delta| < \mathsf{rat}(\theta + 2\delta)$。因此，根据归纳假设，$\mathsf{rat}(q) \sim_{\theta+2\delta} y_\delta$，因此通过三角不等式 $\mathsf{rat}(q) \sim_{\theta+4\delta} \mathsf{lim}(y)$。因此，足以选择 $\delta :\equiv (\epsilon - \theta)/4$。
剩下的两个情况完全类似。 □

接下来，我们想为 $\mathbb{R}_\mathsf{c}$ 配备乘法结构。对于每个 $q : \mathbb{Q}$，映射 $r \mapsto q \cdot r$ 是 Lipschitz 的，常数[1] 为 $|q| + 1$，因此我们可以将其扩展到实数上的乘法。因此 $\mathbb{R}_\mathsf{c}$ 是一个 $\mathbb{Q}$ 上的向量空间。通常，我们可以将实数乘法定义为

$$u \cdot v :\equiv \tfrac{1}{2} \cdot ((u+v)^2 - u^2 - v^2) \tag{11.3.45}$$

因此我们只需要平方 $u \mapsto u^2$ 作为一个映射 $\mathbb{R}_\mathsf{c} \to \mathbb{R}_\mathsf{c}$。平方不是一个 Lipschitz 映射，但它在每个有界域上是 Lipschitz 的，这使得我们可以将其拼接在一起。定义开区间和闭区间

$$[u, v] :\equiv \{\, x : \mathbb{R}_\mathsf{c} \mid u \le x \le v \,\} \qquad \text{和} \qquad (u, v) :\equiv \{\, x : \mathbb{R}_\mathsf{c} \mid u < x < v \,\}$$

尽管技术上 $[u, v]$ 或 $(u, v)$ 的元素是一个 Cauchy 实数，连同一个证明，因为后者居住在一个无关紧要的命题中，所以它是无趣的。因此，与子集类型常见的情况一样，我们通常仅当 $x : \mathbb{R}_\mathsf{c}$ 满足 $u \le x \le v$ 时写作 $x : [u, v]$，同样。

Theorem 11.3.46. *存在一个唯一的函数 $(-)^2 : \mathbb{R}_\mathsf{c} \to \mathbb{R}_\mathsf{c}$ 扩展了有理数的平方 $q \mapsto q^2$ 并满足*

$$\forall (n : \mathbb{N}). \forall (u, v : [-n, n]). |u^2 - v^2| \le 2 \cdot n \cdot |u - v|$$

*证明.* 我们首先观察到，对于每个 $u : \mathbb{R}_\mathsf{c}$ 仅存在 $n : \mathbb{N}$ 使得 $-n \le u \le n$，参见 Exercise 11.7，因此映射

$$e : \left(\sum_{n:\mathbb{N}} [-n, n]\right) \to \mathbb{R}_\mathsf{c} \qquad e(n, x) :\equiv x$$

是满射。接下来，对于每个 $n : \mathbb{N}$，平方映射

$$s_n : \{\, q : \mathbb{Q} \mid -n \le q \le n \,\} \to \mathbb{Q} \qquad s_n(q) :\equiv q^2$$

是 Lipschitz 的，常数为 $2n$，因此我们可以使用 Lemma 11.3.15 将其扩展到 Lipschitz 常数为 $2n$ 的映射 $\bar{s}_n : [-n, n] \to \mathbb{R}_\mathsf{c}$，参见 Exercise 11.8 了解详细信息。这些映射 $\bar{s}_n$ 是兼容的：如果 $m < n$ 对于某些 $m, n : \mathbb{N}$，那么 $s_n$ 限制到 $[-m, m]$ 必须与 $s_m$ 一致，因为两者都是 Lipschitz 的，因此在 Lemma 11.3.39 的意义上是连续的。因此，根据 Theorem 10.1.5，映射

$$\left(\sum_{n:\mathbb{N}} [-n, n]\right) \to \mathbb{R}_\mathsf{c} \qquad (n, x) \mapsto s_n(x)$$

唯一地分解为 $\mathbb{R}_\mathsf{c}$ 给我们所需的函数。 □

此时我们得到了实数的环结构和阿基米德顺序。要将 $\mathbb{R}_\mathsf{c}$ 作为一个阿基米德有序域，我们仍然需要逆元。

Theorem 11.3.47. *Cauchy 实数是可逆的，当且仅当它远离零。*

[1] 我们将 Lipschitz 常数定义为 正有理数。

证明. 首先，假设 $u : \mathbb{R}_c$ 有一个逆元 $v : \mathbb{R}_c$ 根据阿基米德原则存在 $q : \mathbb{Q}$ 使得 $|v| < q$。那么 $1 = |uv| < |u| \cdot v < |u| \cdot q$，因此 $|u| > 1/q$，也就是说 $u \# 0$。
相反，我们通过拼接函数来构造逆映射

$$(-)^{-1} : \{ u : \mathbb{R}_c \mid u \# 0 \} \to \mathbb{R}_c$$

只给出主要步骤。对于每个 $q : \mathbb{Q}$，定义

$$[q, \infty) :\equiv \{ u : \mathbb{R}_c \mid q \le u \} \qquad (-\infty, q] :\equiv \{ u : \mathbb{R}_c \mid u \le -q \}$$

然后，当 $q$ 取遍 $\mathbb{Q}_+$ 时，类型 $(-\infty, q]$ 和 $[q, \infty)$ 共同覆盖 $\{ u : \mathbb{R}_c \mid u \# 0 \}$。在每个 $[q, \infty)$ 和 $(-\infty, q]$ 上，逆函数通过应用 Lemma 11.3.15 与 Lipschitz 常数 $1/q^2$ 得到。最后，Theorem 10.1.5 保证逆函数唯一地分解为 $\{ u : \mathbb{R}_c \mid uapart0 \}$。 □

我们用定理总结 $\mathbb{R}_c$ 的代数结构。

Theorem 11.3.48. Cauchy 实数组成一个阿基米德有序域。

### 11.3.4 Cauchy 实数是 Cauchy 完备的

我们通过将 $\mathbb{Q}$ 在 Cauchy 近似的极限下闭合来构造 $\mathbb{R}_c$，所以 $\mathbb{R}_c$ 应该是 Cauchy 完备的。根据 Theorem 11.3.44，Cauchy 近似 $x : \mathbb{Q}_+ \to \mathbb{R}_c$ 的定义与 $\mathbb{R}_c$ 构造中的定义一致，并且与 Definition 11.2.10（适用于 $\mathbb{R}_c$）中的 Cauchy 近似一致。
因此，给定一个 Cauchy 近似 $x : \mathbb{Q}_+ \to \mathbb{R}_c$，我们很自然地期望 $\lim(x)$ 是它的极限，极限的概念定义如 Definition 11.2.10。但事实如此，通过 Theorem 11.3.44 和 Lemma 11.3.37。我们已经证明了：

Theorem 11.3.49. 每个 $\mathbb{R}_c$ 中的 Cauchy 近似都有一个极限。

一个阿基米德有序域中每个 Cauchy 近似都有一个极限的域称为 Cauchy **完备**。Cauchy 实数是最小的此类域。

Theorem 11.3.50. Cauchy 实数嵌入到每个 Cauchy 完备的阿基米德有序域中。

证明. 假设 $F$ 是一个 Cauchy 完备的阿基米德有序域。因为极限是唯一的，所以存在一个运算符 $\lim$，它将 $F$ 中的 Cauchy 近似映射到它们的极限。我们定义嵌入 $e : \mathbb{R}_c \to F$ 通过 $(\mathbb{R}_c, \sim)$-递归作为

$$e(\mathsf{rat}(q)) :\equiv q \qquad e(\lim(x)) :\equiv \lim(e \circ x)$$

在 $F$ 上合适的 $\frown$ 是

$$(a \frown_\epsilon b) :\equiv |a - b| < \epsilon$$

这是一个分离的关系，因为 $F$ 是阿基米德的。对于 $(\mathbb{R}_c, \sim)$-递归的其余条款很容易验证。还需要检查 $e$ 是一个固定有理数的有序域的嵌入。 □

## 11.4 柯西实数与戴德金实数的比较 (Comparison of Cauchy and Dedekind reals)

我们来讨论一下柯西实数 $(\mathbb{R}_c)$ 和戴德金实数 $(\mathbb{R}_d)$ 之间的关系。根据 Theorem 11.3.48，$\mathbb{R}_c$ 是阿基米德有序域 (archimedean ordered field)。它对于 $\Omega$ 也是可容许的 (admissible)，这一点很容易验证。（如果 $\Omega$ 是初始 $\sigma$-框架 (initial $\sigma$-frame)，则只需简单的归纳法即可验证；在其他情况下，这一点是直接显然的。）因此，根据 Theorem 11.2.14，有一个有序域的嵌入 (embedding of ordered fields)：

$$\mathbb{R}_c \to \mathbb{R}_d$$

这个嵌入固定了有理数 (rational numbers)。（我们也可以从 Theorems 11.2.12 and 11.3.50 中得出这一结论。）通常情况下，如果没有进一步的假设，我们并不期望 $\mathbb{R}_c$ 和 $\mathbb{R}_d$ 会相等。

Lemma 11.4.1. *如果对于每个 $x : \mathbb{R}_{\mathsf{d}}$，仅存在一个满足以下条件的 $c$：*

$$c : \prod_{q,r:\mathbb{Q}} (q < r) \to (q < x) + (x < r) \tag{11.4.2}$$

*那么柯西实数 (Cauchy reals) 和戴德金实数 (Dedekind reals) 是一致的。*

*证明.* 请注意，(11.4.2) 中的类型是 (11.2.3) 的非截断版本，后者表明 $<$ 是一个弱线性序 (weak linear order)。我们已经知道 $\mathbb{R}_{\mathsf{c}}$ 嵌入了 $\mathbb{R}_{\mathsf{d}}$，因此只需证明每个戴德金实数 (Dedekind real) 都仅是一个有理数柯西序列 (Cauchy sequence) 的极限即可。
考虑任意 $x : \mathbb{R}_{\mathsf{d}}$。根据假设，存在一个如引理陈述中所描述的 $c$，并且根据切分 (cuts) 的居留性 (inhabitation)，存在有理数 $a, b : \mathbb{Q}$ 满足 $a < x < b$。我们通过递归构造一个序列 $f : \mathbb{N} \to \{ (q,r) \in \mathbb{Q} \times \mathbb{Q} \mid q < r \}$：

(i) 设 $f(0) :\equiv (a,b)$。
(ii) 假设 $f(n)$ 已经定义为 $(q_n, r_n)$，且 $q_n < r_n$。定义 $s :\equiv (2q_n + r_n)/3$ 和 $t :\equiv (q_n + 2r_n)/3$。然后 $c(s,t)$ 决定了 $s < x$ 还是 $x < t$。如果决定 $s < x$，那么我们设置 $f(n+1) :\equiv (s, r_n)$，否则设置 $f(n+1) :\equiv (q_n, t)$。

让我们用 $(q_n, r_n)$ 表示序列 $f$ 的第 $n$ 项。那么很容易看出，对于所有 $n : \mathbb{N}$，都有 $q_n < x < r_n$ 并且 $|q_n - r_n| \leq (2/3)^n \cdot |q_0 - r_0|$。因此，$q_0, q_1, \ldots$ 和 $r_0, r_1, \ldots$ 都是收敛于戴德金实数 $x$ 的柯西序列 (Cauchy sequences)。我们已经证明，对于每个 $x : \mathbb{R}_{\mathsf{d}}$，仅存在一个收敛于 $x$ 的柯西序列。 □

该引理表明，无论是可数选择公理 (countable choice) 还是排中律 (excluded middle) 都足以保证 $\mathbb{R}_{\mathsf{c}}$ 和 $\mathbb{R}_{\mathsf{d}}$ 的一致性。

Corollary 11.4.3. *如果排中律 (excluded middle) 或者可数选择公理 (countable choice) 成立，那么 $\mathbb{R}_{\mathsf{c}}$ 和 $\mathbb{R}_{\mathsf{d}}$ 是等价的。*

*证明.* 如果排中律成立，那么 $(x < y) \to (x < z) + (z < y)$ 可以被证明：要么 $x < z$，要么 $\neg(x < z)$。在前一种情况下，我们完成证明；而在后一种情况下，我们得出 $z < y$，因为 $z \leq x < y$。因此，我们得到 (11.4.2)，可以应用 Lemma 11.4.1。
假设可数选择公理成立。集合 $S = \{ (q,r) \in \mathbb{Q} \times \mathbb{Q} \mid q < r \}$ 与 $\mathbb{N}$ 是等价的，因此我们可以将可数选择公理应用于 $x$ 的定位 (located)，即：

$$\forall((q,r) : S).\, (q < x) \vee (x < r)$$

请注意，$(q < x) \vee (x < r)$ 可以表示为一个存在量词语句 (existential statement) $\exists(b : \mathbf{2}).\, (b = 0_{\mathbf{2}} \to q < x) \wedge (b = 1_{\mathbf{2}} \to x < r)$。选择函数的柯里形式 (curried form) 就是 (11.4.2)，因此 Lemma 11.4.1 再次适用。 □

## 11.5 区间的紧致性 (Compactness of the interval)

我们已经指出，我们对于实数的构造与经典逻辑 (classical logic) 完全兼容。因此，假设排中律 (law of excluded middle) (3.4.1) 和选择公理 (axiom of choice) (3.8.1) 成立，我们可以发展经典分析 (classical analysis)，这实际上相当于复制任何标准的分析书籍。
然而，对于任何对计算感兴趣的人，例如数值分析学家，应该对在一个计算上有意义的环境中发展分析感到好奇。构造性环境中的分析是可能的，这一点已经由 [?] 证明了。作为经典分析和构造性分析之间差异和相似性的一个例子，我们将简要讨论一个主题——闭区间 $[0,1]$ 的紧致性 (compactness) 以及围绕这一概念的一些定理。
在构造性数学中，经典上等价的概念常常会分裂，紧致性也不例外。最常用的三种紧致性概念是：

(i) **度量紧致性** (metrically compact): “柯西完备 (Cauchy complete) 和全有界 (totally bounded)”，

(ii) **波尔查诺-魏尔斯特拉斯紧致性** (Bolzano–Weierstraß compact): "每个序列都有一个收敛的子序列",
(iii) **海涅-博雷尔紧致性** (Heine–Borel compact): "每个开覆盖 (open cover) 都有一个有限子覆盖 (finite subcover)"。

这些在经典数学中都是等价的。让我们看看它们在同伦类型论 (homotopy type theory) 中的表现。我们可以使用戴德金实数 (Dedekind reals) 或柯西实数 (Cauchy reals)，因此我们将实数记作 $\mathbb{R}$。首先我们回顾一些基本定义。

Definition 11.5.1. **度量空间** (metric space) $(M,d)$ 是一个集合 $M$，其上有一个映射 $d : M \times M \to \mathbb{R}$，满足对所有 $x,y,z : M$，

$$d(x,y) \geq 0, \qquad d(x,y) = d(y,x),$$
$$d(x,y) = 0 \Leftrightarrow x = y, \qquad d(x,z) \leq d(x,y) + d(y,z).$$

Definition 11.5.2. **柯西逼近** (Cauchy approximation) 在 $M$ 中是一个序列 $x : \mathbb{Q}_+ \to M$，满足

$$\forall(\delta,\epsilon).\, d(x_\delta, x_\epsilon) < \delta + \epsilon.$$

一个柯西逼近 $x : \mathbb{Q}_+ \to M$ 的 **极限** (limit) 是一个点 $\ell : M$，满足

$$\forall(\epsilon,\theta : \mathbb{Q}_+).\, d(x_\epsilon, \ell) < \epsilon + \theta.$$

一个 **完备度量空间** (complete metric space) 是其中每个柯西逼近都有极限的空间。

Definition 11.5.3. 对于一个正有理数 $\epsilon$，一个 $\epsilon$-**网** ($\epsilon$-net) 在度量空间 $(M,d)$ 中是一个元素

$$\sum_{(n:\mathbb{N})} \sum_{(x_1,\ldots,x_n:M)} \forall(y : M).\, \exists(k \leq n).\, d(x_k, y) < \epsilon.$$

换句话说，这是一个有限的点序列 $x_1,\ldots,x_n$，使得 $M$ 中的每个点仅在某个 $x_k$ 的 $\epsilon$ 内。
一个度量空间 $(M,d)$ 是 **全有界** (totally bounded) 的，当它有所有大小的 $\epsilon$-网时：

$$\prod_{(\epsilon:\mathbb{Q}_+)} \sum_{(n:\mathbb{N})} \sum_{(x_1,\ldots,x_n:M)} \forall(y : M).\, \exists(k \leq n).\, d(x_k, y) < \epsilon.$$

Remark 11.5.4. 在全有界性的定义中，我们使用了不严格的记法 $\sum_{(n:\mathbb{N})} \sum_{(x_1,\ldots,x_n:M)}$。正式来说，我们应该写成 $\sum_{(x:\mathsf{List}(M))}$，其中 $\mathsf{List}(M)$ 是来自 §5.1 的有限列表的归纳类型 (inductive type of finite lists)。然而，这会使得表达其余部分的陈述更加麻烦。

注意，在全有界性的定义中，我们要求纯粹存在 (pure existence) 的 $\epsilon$-网，而不是仅仅存在 (mere existence)。这样我们得到一个函数，该函数为每个 $\epsilon : \mathbb{Q}_+$ 分配一个特定的 $\epsilon$-网。这样的函数可以称为"全有界性的模数 (modulus of total boundedness)"。通常，在将经典的度量概念移植到同伦类型论时，我们应该谨慎地使用命题截断 (propositional truncation)，通常这样做是为了避免要求从 $\mathbb{R}$ 到 $\mathbb{Q}$ 或 $\mathbb{N}$ 的非常数映射。例如，以下是均匀连续性的"正确"定义。

Definition 11.5.5. 在度量空间上的映射 $f : M \to \mathbb{R}$ 是 **均匀连续** (uniformly continuous) 的，当

$$\prod_{(\epsilon:\mathbb{Q}_+)} \sum_{(\delta:\mathbb{Q}_+)} \forall(x,y : M).\, d(x,y) < \delta \Rightarrow |f(x) - f(y)| < \epsilon.$$

特别地，一个均匀连续映射有一个均匀连续性的模数 (modulus of uniform continuity)，这是一个为每个 $\epsilon$ 分配相应 $\delta$ 的函数。

让我们证明 $[0,1]$ 在第一种意义上是紧致的。

Theorem 11.5.6. *闭区间 $[0,1]$ 是完备的且全有界的。*

证明. 给定 $\epsilon : \mathbb{Q}_+$，存在 $k : \mathbb{N}$ 使得 $2/k < \epsilon$，所以我们可以取 $\epsilon$-网 $x_i = i/k$，其中 $i = 0, \dots, k$。这是一个 $\epsilon$-网，因为对于每个 $y : [0,1]$，仅存在某个 $i$，使得 $0 \le i \le k$ 且 $(i-1)/k < y < (i+1)/k$，因此 $|y - x_i| < 2/k < \epsilon$。
对于 $[0,1]$ 的完备性，考虑一个柯西逼近 $x : \mathbb{Q}_+ \to [0,1]$ 并让 $\ell$ 是它在 $\mathbb{R}$ 中的极限。由于 max 和 min 是 Lipschitz 映射，由 $r(x) :\equiv \max(0, \min(1, x))$ 定义的从 $\mathbb{R}$ 到 $[0,1]$ 的缩回 (retraction) $r$ 与柯西逼近的极限交换，因此

$$r(\ell) = r(\lim x) = \lim(r \circ x) = \lim x = \ell,$$

这意味着 $0 \le \ell \le 1$，这是我们所要求的。 □

因此，我们在同伦类型论中至少有一种好的紧致性概念。不幸的是，它仅限于度量空间，因为全有界性是一个度量概念。我们很快将考虑其他两个概念，但首先我们证明在全有界空间上的均匀连续映射具有 **上确界** (supremum)，即一个小于或等于所有其他上界的上界。

Theorem 11.5.7. *在一个全有界度量空间 $(M,d)$ 上的均匀连续映射 $f : M \to \mathbb{R}$ 有一个上确界 $m : \mathbb{R}$。对于每个 $\epsilon : \mathbb{Q}_+$，存在 $u : M$，使得 $|m - f(u)| < \epsilon$。*

证明. 设 $h : \mathbb{Q}_+ \to \mathbb{Q}_+$ 是 $f$ 的均匀连续性模数 (modulus of uniform continuity)。我们如下定义一个逼近 $x : \mathbb{Q}_+ \to \mathbb{R}$：对于任意 $\epsilon : \mathbb{Q}$，$M$ 的全有界性给出了一个 $h(\epsilon)$-网 $y_0, \dots, y_n$。定义

$$x_\epsilon :\equiv \max(f(y_0), \dots, f(y_n))$$

我们声称 $x$ 是一个柯西逼近。考虑任意 $\epsilon, \eta : \mathbb{Q}$，使得

$$x_\epsilon \equiv \max(f(y_0), \dots, f(y_n)) \quad \text{和} \quad x_\eta \equiv \max(f(z_0), \dots, f(z_m))$$

对于某个 $h(\epsilon)$-网 $y_0, \dots, y_n$ 和 $h(\eta)$-网 $z_0, \dots, z_m$。每个 $z_i$ 仅与某个 $y_j$ 的 $h(\epsilon)$-近，因此 $|f(z_i) - f(y_j)| < \epsilon$，我们可以得出

$$f(z_i) < \epsilon + f(y_j) \le \epsilon + x_\epsilon,$$

因此 $x_\eta < \epsilon + x_\epsilon$。对称地，我们得到 $x_\eta < \eta + x_\eta$，因此 $|x_\eta - x_\epsilon| < \eta + \epsilon$。
我们声称 $m :\equiv \lim x$ 是 $f$ 的上确界。为了证明 $f(x) \le m$ 对所有 $x : M$ 成立，只需证明 $\neg(m < f(x))$。假设相反的情况，即 $m < f(x)$。存在 $\epsilon : \mathbb{Q}_+$ 使得 $m + \epsilon < f(x)$。但是现在仅对定义 $x_\epsilon$ 的某个 $y_i$，我们得到 $|f(x) - f(y_i)| < \epsilon$，因此 $m < f(x) - \epsilon < f(y_i) \le m$，这与假设矛盾。
最后，我们通过证明 $m$ 满足定理的第二部分来结束证明，因为它自动是一个最小的上界。给定任意 $\epsilon : \mathbb{Q}_+$，一方面 $|m - f(x_{\epsilon/2})| < 3\epsilon/4$，另一方面 $|f(x_{\epsilon/2}) - f(y_i)| < \epsilon/4$ 仅对定义 $x_{\epsilon/2}$ 的某个 $y_i$ 成立，因此通过取 $u :\equiv y_i$，我们通过三角不等式 (triangle inequality) 得到 $|m - f(u)| < \epsilon$。 □

现在，如果在 Theorem 11.5.7 中我们也知道 $M$ 是完备的，我们可以希望将均匀连续性的假设减弱为连续性 (continuity)，并将结论加强为存在一个点在该点上达到上确界。通常的证明这些改进依赖于以下事实：在一个完备的全有界空间中

(i) 连续性意味着均匀连续性，
(ii) 每个序列都有一个收敛的子序列。

第一个陈述很容易从海涅-博雷尔紧致性推导出来，第二个则只是波尔查诺-魏尔斯特拉斯紧致性。不幸的是，这两者都有些问题。首先我们证明波尔查诺-魏尔斯特拉斯紧致性蕴含了一个排中律的实例，称为**全知原理的有限形式** (limited principle of omniscience)：对于每个 $\alpha : \mathbb{N} \to \mathbf{2}$，

$$\Big(\sum_{n:\mathbb{N}} \alpha(n) = 1_2\Big) + \Big(\prod_{n:\mathbb{N}} \alpha(n) = 0_2\Big). \tag{11.5.8}$$

从计算的角度来看，我们不会期望这个原理成立，因为它要求我们决定一个函数是否有无穷多个值为 $0_2$。

Theorem 11.5.9. *波尔查诺-魏尔斯特拉斯紧致性 (Bolzano–Weierstraß compactness) 对于 $[0,1]$ 蕴含了全知原理的有限形式。*

证明. 给定任意 $\alpha : \mathbb{N} \to \mathbf{2}$，定义序列 (sequence) $x : \mathbb{N} \to [0,1]$ 为

$$x_n :\equiv \begin{cases} 0 & \text{如果对所有 } k < n,\ \alpha(k) = 0_{\mathbf{2}}, \\ 1 & \text{如果对某个 } k < n,\ \alpha(k) = 1_{\mathbf{2}}. \end{cases}$$

如果波尔查诺-魏尔斯特拉斯性质成立，存在一个严格递增的 $f : \mathbb{N} \to \mathbb{N}$，使得 $x \circ f$ 是一个柯西序列 (Cauchy sequence)。对于一个足够大的 $n : \mathbb{N}$，第 $n$ 项 $x_{f(n)}$ 距离其极限小于 $1/6$。要么 $x_{f(n)} < 2/3$，要么 $x_{f(n)} > 1/3$。如果 $x_{f(n)} < 2/3$，则 $x_n$ 收敛到 0，因此 $\prod_{(n:\mathbb{N})} \alpha(n) = 0_{\mathbf{2}}$。如果 $x_{f(n)} > 1/3$，则 $x_{f(n)} = 1$，因此 $\sum_{(n:\mathbb{N})} \alpha(n) = 1_{\mathbf{2}}$。 □

虽然我们可能不会太在意波尔查诺-魏尔斯特拉斯紧致性，但没有海涅-博雷尔紧致性似乎更难以接受，正如经典数学和布劳威尔直觉主义 (Brouwer's Intuitionism) 都接受了它一样。由于我们不想深入一般拓扑学，我们将使用基本开集 (basic open sets) 来工作。在 $\mathbb{R}$ 的情况下，这些是具有有理数端点的开区间。一个由类型 $I$ 索引的此类区间的族将是一个映射

$$\mathcal{F} : I \to \{\, (q,r) : \mathbb{Q} \times \mathbb{Q} \mid q < r \,\},$$

其想法是，一个有理数对 $(q,r)$ 与 $q < r$ 确定类型 $\{\, x : \mathbb{R} \mid q < x < r \,\}$。允许退化区间稍微更方便一些，因此我们取一个 **基本区间族** (family of basic intervals) 为一个映射

$$\mathcal{F} : I \to \mathbb{Q} \times \mathbb{Q}$$

要非常精确，一个族是一个依赖对 $(I, \mathcal{F})$，而不仅仅是 $\mathcal{F}$。一个 **有限基本区间族** (finite family of basic intervals) 是由 $\{\, m : \mathbb{N} \mid m < n \,\}$ 对某个 $n : \mathbb{N}$ 索引的族。我们通常用有限列表 $[(q_0, r_0), \ldots, (q_{n-1}, r_{n-1})]$ 来表示它。最后，一个 $(I, \mathcal{F})$ 的 **有限子族** (finite subfamily) 是由索引列表 $[i_1, \ldots, i_n]$ 给出的，这些索引决定了有限族 $[\mathcal{F}(i_1), \ldots, \mathcal{F}(i_n)]$。
只要我们知道对 $(q,r)$ 和对应的区间 $\{\, x : \mathbb{R} \mid q < x < r \,\}$ 之间的区别，我们可以安全地对两者使用相同的符号 $(q,r)$。区间的交集 (intersections) 和包含 (inclusions) 可以用它们的端点表示：

$$\begin{aligned} (q,r) \cap (s,t) &:\equiv (\max(q,s), \min(r,t)) \\ (q,r) \subseteq (s,t) &:\equiv (q < r \Rightarrow s \leq q < r \leq t) \end{aligned}$$

我们说 $(I, \lambda i.\,(q_i, r_i))$ (**逐点**) **覆盖** $[a,b]$ ((pointwise) covers $[a,b]$)，当

$$\forall (x : [a,b]).\, \exists (i : I).\, q_i < x < r_i \tag{11.5.10}$$

$[0,1]$ 的 **海涅-博雷尔紧致性** (Heine–Borel compactness)表示每个覆盖 $[0,1]$ 的区间族都有一个有限子族仍然覆盖 $[0,1]$。

Theorem 11.5.11. 如果排中律成立，那么 $[0,1]$ 是海涅-博雷尔紧致的。

证明. 假设为了达到矛盾，一个族 $(I, \lambda i.\,(a_i, b_i))$ 覆盖 $[0,1]$，但没有有限子族覆盖它。我们构造一系列闭区间 $[q_n, r_n]$，这些区间是嵌套的，它们的大小收缩到 0，且其中没有一个被 $(I, \lambda i.\,(a_i, b_i))$ 的有限子族覆盖。
我们设 $[q_0, r_0] :\equiv [0,1]$。假设 $[q_n, r_n]$ 已经构造，设 $s :\equiv (2q_n + r_n)/3$ 和 $t :\equiv (q_n + 2r_n)/3$。$[q_n, t]$ 和 $[s, r_n]$ 都被 $(I, \lambda i.\,(a_i, b_i))$ 覆盖，但它们不能同时有一个有限子覆盖，否则 $[q_n, r_n]$ 也会有一个有限子覆盖。要么 $[q_n, t]$ 有一个有限子覆盖，要么它没有。如果有，我们设 $[q_{n+1}, r_{n+1}] :\equiv [s, r_n]$，否则我们设 $[q_{n+1}, r_{n+1}] :\equiv [q_n, t]$。
序列 $q_0, q_1, \ldots$ 和 $r_0, r_1, \ldots$ 都是柯西的，它们收敛到 $[0,1]$ 中的一个点 $x$，该点包含在每个 $[q_n, r_n]$ 中。仅存在 $i : I$ 使得 $a_i < x < b_i$。由于区间 $[q_n, r_n]$ 的大小收缩到零，存在 $n : \mathbb{N}$ 使得 $a_i < q_n \leq x \leq r_n < b_i$，但这意味着 $[q_n, r_n]$ 被单个区间 $(a_i, b_i)$ 覆盖，而同时它没有有限子覆盖。这是矛盾。 □

没有排中律，或布劳威尔直觉主义 (Brouwerian Intuitionism) 的一点点帮助，我们似乎陷入了困境。然而，在构造性环境中，$[0,1]$ 的海涅-博雷尔紧致性 可以得到恢复，并且仍然与经典数学兼容！为此，我们需要重新审视覆盖的概念。(11.5.10) 的问题在于截断的存在使得一个空间可以以任何随意的方式被覆盖，从计算的角度来看，我们几乎没有希望仅提取一个有限子覆盖。通过移除截断，我们得到

$$\prod_{(x:[0,1])} \sum_{(i:I)} q_i < x < r_i, \tag{11.5.12}$$

这可能有所帮助，但它对覆盖的要求太高了。使用这个定义，我们甚至无法证明 $(0,3)$ 和 $(2,5)$ 覆盖 $[1,4]$，因为这相当于展示一个从 $[1,4]$ 到 $\mathbf{2}$ 的非常数映射，参见 Exercise 11.6。在这里，我们可以从“无点拓扑 (pointfree topology)”(即位置理论 (locale theory))中吸取教训：覆盖的概念应该用开集来表达，而不涉及点。这样一种对空间的“整体”观点将允许我们分析覆盖的概念，我们将能够恢复海涅-博雷尔紧致性。位置理论使用幂集 (power sets)，我们可以通过假设命题调整 (propositional resizing) 获得；但我们可以从位置理论的预测性 (predicative) 近亲中偷取想法，这被称为“形式拓扑 (formal topology)”。

假设我们有一个族 $(I,\mathcal{F})$ 和一个区间 $(a,b)$。我们如何表达 $(a,b)$ 被该族覆盖的事实，而不涉及点呢？这里有一种方法：如果 $(a,b)$ 等于某个 $\mathcal{F}(i)$，那么它就被该族覆盖了。还有一种方法：如果 $(a,b)$ 被另一个族 $(J,\mathcal{G})$ 覆盖，而每个 $\mathcal{G}(j)$ 都被 $(I,\mathcal{F})$ 覆盖，那么 $(a,b)$ 就被 $(I,\mathcal{F})$ 覆盖。注意，我们正在列出可以用来推导出 $(I,\mathcal{F})$ 覆盖 $(a,b)$ 的规则。我们应该找到足够好的规则并将它们转化为一个归纳定义。

Definition 11.5.13. **归纳覆盖** $\lhd$ (inductive cover $\lhd$)是一个单纯关系 (mere relation)

$$\lhd : (\mathbb{Q}\times\mathbb{Q}) \to \left(\sum_{I:\mathcal{U}} (I \to \mathbb{Q}\times\mathbb{Q})\right) \to \mathsf{Prop}$$

通过以下规则进行归纳定义，其中 $q,r,s,t$ 是有理数，$(I,\mathcal{F})$ 和 $(J,\mathcal{G})$ 是基本区间族：

(i) 自反性 (reflexivity):对于所有 $i:I$，$\mathcal{F}(i) \lhd (I,\mathcal{F})$，
(ii) 传递性 (transitivity):如果 $(q,r) \lhd (J,\mathcal{G})$ 并且 $\forall(j:J).\,\mathcal{G}(j) \lhd (I,\mathcal{F})$，那么 $(q,r) \lhd (I,\mathcal{F})$，
(iii) 单调性 (monotonicity):如果 $(q,r) \subseteq (s,t)$ 并且 $(s,t) \lhd (I,\mathcal{F})$，那么 $(q,r) \lhd (I,\mathcal{F})$，
(iv) 局部化 (localization):如果 $(q,r) \lhd (I,\mathcal{F})$，那么 $(q,r) \cap (s,t) \lhd (I, \lambda i.\,(\mathcal{F}(i) \cap (s,t)))$。
(v) 如果 $q<s<t<r$，那么 $(q,r) \lhd [(q,t),(r,s)]$，
(vi) $(q,r) \lhd (\{\,(s,t) : \mathbb{Q}\times\mathbb{Q} \mid q<s<t<r\,\}, \lambda u.\,u)$。

该定义应被视为一个高阶归纳类型 (higher-inductive type)，其中列出的规则是点构造器 (point constructors)，并且该类型是 $(-1)$ 截断的。前四个条款是一般性的，应该是直观的。最后两条是特定于实数的：一条说，如果它们重叠，则一个区间可以被两个区间覆盖，而另一条说，一个区间可以从内部覆盖。如果 $r \le q$，则根据最后一条规则，$(q,r)$ 被空族覆盖。

归纳覆盖享有海涅-博雷尔性质，其证明需要一个引理。

Lemma 11.5.14. *假设 $q<s<t<r$ 并且 $(q,r) \lhd (I,\mathcal{F})$。那么仅存在 $(I,\mathcal{F})$ 的一个有限子族，它归纳覆盖 $(s,t)$。*

*证明.* 我们通过归纳证明 $(q,r) \lhd (I,\mathcal{F})$。有六种情况：

(i) 自反性：如果 $(q,r) = \mathcal{F}(i)$，那么根据单调性 $(s,t)$ 被有限子族 $[\mathcal{F}(i)]$ 覆盖。
(ii) 传递性：假设 $(q,r) \lhd (J,\mathcal{G})$ 并且 $\forall(j:J).\,\mathcal{G}(j) \lhd (I,\mathcal{F})$。根据归纳假设，仅存在 $[\mathcal{G}(j_1),\dots,\mathcal{G}(j_n)]$ 覆盖 $(s,t)$。再次根据归纳假设，它们中的每一个都被 $(I,\mathcal{F})$ 的一个有限子族覆盖，并且我们可以将这些子族收集成一个有限子族覆盖 $(s,t)$。
(iii) 单调性：如果 $(q,r) \subseteq (u,v)$ 并且 $(u,v) \lhd (I,\mathcal{F})$，那么我们可以应用归纳假设到 $(u,v) \lhd (I,\mathcal{F})$，因为 $u<s<t<v$。
(iv) 局部化：假设 $(q',r') \lhd (I,\mathcal{F})$ 并且 $(q,r) = (q',r') \cap (a,b)$。因为 $q'<s<t<r'$，根据归纳假设，存在一个有限子覆盖 $[\mathcal{F}(i_1),\dots,\mathcal{F}(i_n)]$ 覆盖 $(s,t)$。我们还知道 $a<s<t<b$，因此 $(s,t) = (s,t)\cap(a,b)$ 被 $[\mathcal{F}(i_1)\cap(a,b),\dots,\mathcal{F}(i_n)\cap(a,b)]$ 覆盖，这是 $(I,\lambda i.\,(\mathcal{F}(i)\cap(a,b)))$ 的一个有限子族。

(v) 如果 $(q,r) \lhd [(q,v),(u,r)]$ 对于某个 $q < u < v < r$，那么根据单调性 $(s,t) \lhd [(q,v),(u,r)]$。
(vi) 最后，通过自反性 $(s,t) \lhd (\{\, (u,v) : \mathbb{Q} \times \mathbb{Q} \mid q < u < v < r \,\}, \lambda z.\, z)$。 □

说 $(I,\mathcal{F})$ 归纳覆盖 $[a,b]$ 当仅存在 $\epsilon : \mathbb{Q}_+$，使得 $(a-\epsilon, b+\epsilon) \lhd (I,\mathcal{F})$。

Corollary 11.5.15. 闭区间对归纳覆盖是海涅-博雷尔紧致的。

证明. 假设 $[a,b]$ 被 $(I,\mathcal{F})$ 归纳覆盖，因此仅存在 $\epsilon : \mathbb{Q}_+$，使得 $(a-\epsilon, b+\epsilon) \lhd (I,\mathcal{F})$。根据 Lemma 11.5.14 存在一个有限子覆盖 $(a-\epsilon/2, b+\epsilon/2)$，因此是 $[a,b]$ 的有限子覆盖。 □

形式拓扑的经验 (Experience from formal topology) 表明，归纳覆盖的规则对于构造性的无点拓扑 (pointfree topology) 发展是足够的。但我们也可以提供自己的证据，证明它们是一个合理的概念。

Theorem 11.5.16.

(i) 一个归纳覆盖也是一个逐点覆盖。
(ii) 假设排中律，一个逐点覆盖也是一个归纳覆盖。

证明.

(i) 考虑一个基本区间族 $(I,\mathcal{F})$，其中我们写 $(q_i,r_i) :\equiv \mathcal{F}(i)$，一个由 $(I,\mathcal{F})$ 逐点覆盖的区间 $(a,b)$ 和 $x$，使得 $a < x < b$。我们通过 $(a,b) \lhd (I,\mathcal{F})$ 的归纳证明，仅存在 $i : I$，使得 $q_i < x < r_i$。大多数情况都很明显，所以我们只展示两个。如果 $(a,b) \lhd (I,\mathcal{F})$ 是通过自反性覆盖的，那么仅存在某个 $i : I$，使得 $(a,b) = (q_i,r_i)$，因此 $q_i < x < r_i$。如果 $(a,b) \lhd (I,\mathcal{F})$ 是通过 $(J, \lambda j.\,(s_j,t_j))$ 的传递性覆盖的，那么根据归纳假设，仅存在 $j : J$，使得 $s_j < x < t_j$，然后因为 $(s_j,t_j) \lhd (I,\mathcal{F})$，再次根据归纳假设，仅存在 $i : I$，使得 $q_i < x < r_i$。其他情况同样激动人心。
(ii) 假设 $(I, \lambda i.\,(q_i,r_i))$ 逐点覆盖 $(a,b)$。根据 Item (vi) 和 Definition 11.5.13 的定义，只需证明 $(I, \lambda i.\,(q_i,r_i))$ 归纳覆盖 $(c,d)$，只要 $a < c < d < b$，所以考虑这样的 $c$ 和 $d$。根据 Theorem 11.5.11，存在一个有限族 $[i_1,\ldots,i_n]$，它已经逐点覆盖 $[c,d]$，因此 $(c,d)$。设 $\epsilon : \mathbb{Q}_+$ 为 $[c,d]$ 的 $[(q_{i_1},r_{i_1}),\ldots,(q_{i_n},r_{i_n})]$ 的 Lebesgue 数 (Lebesgue number)，如 Exercise 11.12 中所示。存在一个正数 $k : \mathbb{N}$，使得 $2(d-c)/k < \min(1,\epsilon)$。对于 $0 \le i \le k$，设

$$c_k :\equiv ((k-i)c + id)/k$$

区间 $(c_0,c_2)$、$(c_1,c_3)$、……、$(c_{k-2},c_k)$ 通过反复使用传递性和 Item (v) 归纳覆盖 $(c,d)$。由于它们的宽度低于 $\epsilon$，因此每个都包含在某个 $(q_i,r_i)$ 中，并且我们可以使用传递性和单调性得出 $(I, \lambda i.\,(q_i,r_i))$ 归纳覆盖 $(c,d)$。 □

前述定理的结果是，就经典数学而言，逐点覆盖和归纳覆盖之间没有区别。特别是，由于在同伦类型论中假设排中律是自洽的，我们不能展示一个归纳覆盖无法逐点覆盖。或者换句话说，逐点覆盖和归纳覆盖之间的区别不在于它们覆盖什么，而在于它们证明它们覆盖的内容。
我们可以写另一本书来继续讨论这些内容，但让我们在这里停止，希望我们已经提供了充分的理由来证明分析可以在同伦类型论中进行发展。好奇的读者应参阅 Exercise 11.13，以了解中值定理 (intermediate value theorem) 的构造性版本。

## 11.6 超现实数 (The surreal numbers)

在本节中，我们将讨论另一种高阶归纳-归纳类型的例子，它汇集了我们的许多线索：Conway 的超现实数领域 No [?]。超现实数是（Dedekind）实数（参见 §11.2）和序数（参见 §10.3）的自然公共推广。Conway 在具有排中律和选择公理的经典数学中，定义了一个超现实数为一对超现实数集合，记为 $\{\, L \mid R \,\}$，其中 $L$ 中的每个元素都严格小于 $R$ 中的每个元素。这显然看起来像是一个归纳定义，但将其视为归纳定义存在三个问题。
首先，该定义需要超现实数之间的（严格）不等关系，因此该关系必须与超现实数类型 No 同时定义。（Conway 通过首先定义 游戏 来避免这个问题，游戏类似于超现实数，但省略了对 $L$ 和 $R$ 的兼容

性条件。）与 Cauchy 实数的 $\sim$ 关系一样，这种同时定义可以先验地是归纳-归纳或归纳-递归的。我们将选择将其做成归纳-归纳类型，原因与我们为 $\sim$ 做出该选择的原因相同。

此外，我们将分别定义超现实数的严格不等关系 $<$ 和非严格不等关系 $\leq$（并且互相归纳定义）。Conway 用 $\leq$ 来定义 $<$，这种方法在经典数学中是合理的，但在构造性数学中却不合理。此外，$<$ 的负定义将使其作为高阶归纳类型构造器的假设无法接受（参见 §5.6）。

其次，Conway 说在 $\{ L \mid R \}$ 中的 $L$ 和 $R$ 应该是“超现实数集合”，但将其直观地理解为谓词 $\mathsf{No} \to \mathsf{Prop}$ 并不正性，因此不能作为归纳构造器的输入。然而，这也不是 Conway 真正想表达的类型论翻译，因为在集合论中，超现实数形成一个适当的类，而 $L$ 和 $R$ 是实际的（小）集合，而不是 $\mathsf{No}$ 的任意子类。在类型论中，这意味着 $\mathsf{No}$ 将相对于一个宇宙 $\mathcal{U}$ 定义，但它本身将属于下一个更高的宇宙 $\mathcal{U}'$，如序数和基数的集合 $\mathsf{Ord}$ 和 $\mathsf{Card}$ 、累积层次结构 $V$，甚至在没有命题调整的情况下的 Dedekind 实数。然后我们将要求超现实数的“集合”$L$ 和 $R$ 是 $\mathcal{U}$-小的，因此用一些 $\mathcal{U}$-小类型索引的超现实数族来表示它们是自然的。（这与我们在 §10.5 中对累积层次结构所做的完全相同。）也就是说，超现实数的构造器将具有类型

$$\prod_{\mathcal{L},\mathcal{R}:\mathcal{U}} (\mathcal{L} \to \mathsf{No}) \to (\mathcal{R} \to \mathsf{No}) \to (\text{some condition}) \to \mathsf{No}$$

并且确实是严格正性的。

最后，在给出 $\mathsf{No}$ 及其排序的相互定义之后，Conway 宣布如果 $x \leq y$ 且 $y \leq x$，那么两个超现实数 $x$ 和 $y$ 是*相等的*。这自然被理解为通过等价关系对“预超现实数”集合进行商。然而，在没有选择公理的情况下，这样的商在构造 Cauchy 实数的通常过程中提出了同样的问题：它将不再是超现实数族 $L$ 和 $R$ 可以产生一个新的超现实数 $\{ L \mid R \}$，因为我们无法必然地“提升”$L$ 和 $R$ 到预超现实数族。当然，我们可以像对 Cauchy 实数一样，通过使用*高阶*归纳-归纳定义来解决这个问题。

Definition 11.6.1. **超现实数**的类型 $\mathsf{No}$, 以及关系 $< : \mathsf{No} \to \mathsf{No} \to \mathcal{U}$ 和 $\leq : \mathsf{No} \to \mathsf{No} \to \mathcal{U}$，定义为如下的高阶归纳-归纳类型。类型 $\mathsf{No}$ 具有以下构造器。

- 对于任何 $\mathcal{L}, \mathcal{R} : \mathcal{U}$ 和函数 $\mathcal{L} \to \mathsf{No}$ 和 $\mathcal{R} \to \mathsf{No}$，其值分别记为 $x^L$ 和 $x^R$，其中 $L : \mathcal{L}$ 和 $R : \mathcal{R}$，如果 $\forall (L : \mathcal{L}). \forall (R : \mathcal{R}). x^L < x^R$，则存在一个超现实数 $x$。
- 对于任何 $x, y : \mathsf{No}$，如果 $x \leq y$ 且 $y \leq x$，则我们有 $\mathsf{eq}_{\mathsf{No}}(x, y) : x = y$。

我们将第一个构造器的输入称为**分割** (cut)。如果 $x$ 是由一个分割构造的超现实数，那么符号 $x^L$ 将隐含地假设 $L : \mathcal{L}$，类似地，$x^R$ 将假设 $R : \mathcal{R}$。通过这种方式，我们通常可以避免命名索引类型 $\mathcal{L}$ 和 $\mathcal{R}$，这在讨论许多不同的分割时非常方便。按照 Conway 的说法，我们称 $x^L$ 为 $x$ 的*左选项* (left option)，$x^R$ 为*右选项* (right option)。

路径构造器意味着不同的分割可以定义相同的超现实数。因此，除非我们也知道 $x$ 是由一个特定的分割定义的，否则讲一个任意超现实数 $x$ 的左选项或右选项是没有意义的。因此，在下面的内容中，我们会说“给定定义超现实数 $x$ 的一个分割”，而不是“给定一个超现实数 $x$”。

关系 $\leq$ 具有以下构造器。

- 给定定义两个超现实数 $x$ 和 $y$ 的分割，如果对所有 $L$，$x^L < y$，并且对所有 $R$，$x < y^R$，则 $x \leq y$。
- 命题截断：对于任何 $x, y : \mathsf{No}$，如果 $p, q : x \leq y$，则 $p = q$。

关系 $<$ 具有以下构造器。

- 给定定义两个超现实数 $x$ 和 $y$ 的分割，如果存在一个 $L$ 使得 $x \leq y^L$，则 $x < y$。
- 给定定义两个超现实数 $x$ 和 $y$ 的分割，如果存在一个 $R$ 使得 $x^R \leq y$，则 $x < y$。
- 命题截断：对于任何 $x, y : \mathsf{No}$，如果 $p, q : x < y$，则 $p = q$。

我们将此与 Conway 的定义进行比较：

- 如果 $L, R$ 是任意两个数集，并且 $L$ 的任何成员都不大于 $R$ 的任何成员，那么存在一个数 $\{ L \mid R \}$。所有的数都是这样构造的。
- $x \geq y$ 当且仅当（没有 $x^R \leq y$ 且 $x \leq$ 没有 $y^L$）。
- $x = y$ 当且仅当（$x \geq y$ 且 $y \geq x$）。
- $x > y$ 当且仅当（$x \geq y$ 且 $y \not\geq x$）。

在 $<$ 的定义中包括 $x \geq y$ 是不必要的，如果所有对象都是 [超现实] 数而不是“游戏”。因此，Conway 的 $<$ 只是其 $\geq$ 的否定，因此 $\{ L \mid R \}$ 成为超现实数的条件与我们的相同。通过取 negating Conway 的 $\leq$ 并取消双重否定，我们得出了我们对 $<$ 的定义，然后我们可以在没有否定的情况下重新定义他的 $\leq$。

我们可以立即用许多超现实数填充 $\mathsf{No}$。像 Conway 一样，我们写作

$$\{\, x, y, z, \dots \,|\, u, v, w, \dots \,\}$$

表示由一个分割定义的超现实数，其中 $\mathcal{L} \to \mathsf{No}$ 和 $\mathcal{R} \to \mathsf{No}$ 是由 $x, y, z, \dots$ 和 $u, v, w, \dots$ 描述的族。当然，如果 $\mathcal{L}$ 或 $\mathcal{R}$ 是 $\mathbf{0}$，我们会在记号中省略相应部分。存在一个不幸的冲突，即与子集的标准记号 $\{\, x : A \mid P(x) \,\}$ 冲突，但我们在本节中不会使用后者。

- 我们递归地定义 $\iota_{\mathbb{N}} : \mathbb{N} \to \mathsf{No}$ 为

$$\begin{aligned} \iota_{\mathbb{N}}(0) &:\equiv \{\, |\, \}, \\ \iota_{\mathbb{N}}(\mathsf{succ}(n)) &:\equiv \{\, \iota_{\mathbb{N}}(n) \,|\, \} \end{aligned}$$

  即，$\iota_{\mathbb{N}}(0)$ 由 $\mathbf{0} \to \mathsf{No}$ 和 $\mathbf{0} \to \mathsf{No}$ 组成的分割定义。类似地，$\iota_{\mathbb{N}}(\mathsf{succ}(n))$ 由 $\mathbf{1} \to \mathsf{No}$ 定义（选择 $\iota_{\mathbb{N}}(n)$）和 $\mathbf{0} \to \mathsf{No}$。
- 类似地，我们使用符号情况递归原理（Lemma 6.10.12）定义 $\iota_{\mathbb{Z}} : \mathbb{Z} \to \mathsf{No}$：

$$\begin{aligned} \iota_{\mathbb{Z}}(0) &:\equiv \{\, |\, \}, \\ \iota_{\mathbb{Z}}(n+1) &:\equiv \{\, \iota_{\mathbb{Z}}(n) \,|\, \} && \text{当 } n \geq 0 \text{ 时}, \\ \iota_{\mathbb{Z}}(n-1) &:\equiv \{\, |\, \iota_{\mathbb{Z}}(n) \,\} && \text{当 } n \leq 0 \text{ 时}。 \end{aligned}$$

- 通过**二进制有理数** 我们指的是一个 $(a, n)$ 的对，其中 $a : \mathbb{Z}$ 且 $n : \mathbb{N}$，并且如果 $n > 0$，则 $a$ 是奇数。我们将其记为 $a/2^n$，并将其与相应的有理数等同。如果 $\mathbb{Q}_D$ 表示二进制有理数的集合，我们通过对 $n$ 进行归纳定义 $\iota_{\mathbb{Q}_D} : \mathbb{Q}_D \to \mathsf{No}$：

$$\begin{aligned} \iota_{\mathbb{Q}_D}(a/2^0) &:\equiv \iota_{\mathbb{Z}}(a) \\ \iota_{\mathbb{Q}_D}(a/2^n) &:\equiv \{\, \iota_{\mathbb{Q}_D}(a/2^n - 1/2^n) \,|\, \iota_{\mathbb{Q}_D}(a/2^n + 1/2^n) \,\}, \quad \text{当 } n > 0 \text{ 时}。 \end{aligned}$$

  这里我们使用了这样一个事实，即如果 $n > 0$ 且 $a$ 是奇数，那么 $a/2^n \pm 1/2^n$ 是一个分母比 $a/2^n$ 更小的二进制有理数。
- 我们定义 $\iota_{\mathbb{R}_{\mathsf{d}}} : \mathbb{R}_{\mathsf{d}} \to \mathsf{No}$，其中 $\mathbb{R}_{\mathsf{d}}$ 是（任意版本的）Dedekind 实数，参见 §11.2，定义为

$$\iota_{\mathbb{R}_{\mathsf{d}}}(x) :\equiv \{\, q \in \mathbb{Q}_D \text{ 满足} q < x \,|\, q \in \mathbb{Q}_D \text{ 满足} x < q \,\}$$

  与前面的情况不同，当我们将二进制有理数视为 Dedekind 实数时，显然这扩展了 $\iota_{\mathbb{Q}_D}$。这可以通过简单性定理（Theorem 11.6.2）得出。
- 回忆 §10.3 中定义的*序数类型* $\mathsf{Ord}$，它由 $<$ 关系良好排序，其中 $A < B$ 意味着 $A = B_{/b}$ 对某些 $b : B$ 成立。我们通过序数上的良基递归（Lemma 10.3.7）定义 $\iota_{\mathsf{Ord}} : \mathsf{Ord} \to \mathsf{No}$：

$$\iota_{\mathsf{Ord}}(A) :\equiv \{\, \iota_{\mathsf{Ord}}(A_{/a}) \text{ 对所有} a : A \,|\, \}$$

  它也将从简单性定理中得出，$\iota_{\mathsf{Ord}}$ 限制在有限序数上时与 $\iota_{\mathbb{N}}$ 一致。（然而，我们提醒读者，与上述例子不同，$\iota_{\mathsf{Ord}}$ 在不限制为较小的序数类时在构造上不可注入；参见 Exercises 11.16 and 11.17。）
- 以下是从 Conway 中选取的一些有趣例子：

$$\begin{aligned} \omega &:\equiv \{\, 0, 1, 2, 3, \dots \,|\, \} \qquad \text{（也是一个序数）} \\ -\omega &:\equiv \{\, |\, \dots, -3, -2, -1, 0 \,\} \\ 1/\omega &:\equiv \{\, 0 \,|\, 1, \tfrac{1}{2}, \tfrac{1}{4}, \tfrac{1}{8}, \dots \,\} \\ \omega - 1 &:\equiv \{\, 0, 1, 2, 3, \dots \,|\, \omega \,\} \\ \omega/2 &:\equiv \{\, 0, 1, 2, 3, \dots \,|\, \dots, \omega - 2, \omega - 1, \omega \,\} \end{aligned}$$

在识别由不同分割表示的超现实数时，以下简单的观察是有用的。

Theorem 11.6.2 (Conway 的简单性定理). 假设 $x$ 和 $z$ 是由分割定义的超现实数，并且满足以下条件。
- 对于所有的 $L$ 和 $R$，$x^L < z < x^R$。
- 对于 $z$ 的每个左选项 $z^L$，存在一个左选项 $x^{L'}$ 使得 $z^L \leq x^{L'}$。
- 对于 $z$ 的每个右选项 $z^R$，存在一个右选项 $x^{R'}$ 使得 $x^{R'} \leq z^R$。

则 $x = z$。

证明. 应用 $\mathsf{No}$ 的路径构造器，我们必须证明 $x \leq z$ 和 $z \leq x$。首先要求对所有 $L$，$x^L < z$，这已假定，并且对所有 $R$，$x < z^R$。但根据假设，对于任何 $z^R$，都有一个 $x^{R'}$ 使得 $x^{R'} \leq z^R$，因此 $x < z^R$ 如所需。因此 $x \leq z$；$z \leq x$ 的证明是对称的。 □

然而，为了进一步了解超现实数，我们需要它们的归纳原理。$(\mathsf{No}, \leq, <)$ 的相互归纳原理适用于三种类型的族：

$$\begin{aligned} A &: \mathsf{No} \to \mathcal{U} \\ B &: \prod_{(x,y:\mathsf{No})} \prod_{(a:A(x))} \prod_{(b:A(y))} (x \leq y) \to \mathcal{U} \\ C &: \prod_{(x,y:\mathsf{No})} \prod_{(a:A(x))} \prod_{(b:A(y))} (x < y) \to \mathcal{U} \end{aligned}$$

与 Cauchy 实数的归纳原理一样，考虑 $B$ 和 $C$ 作为类型 $A(x)$ 和 $A(y)$ 之间的关系族是有帮助的。因此，我们将 $B(x,y,a,b,\xi)$ 写为 $(x,a) \trianglelefteq^{\xi} (y,b)$，将 $C(x,y,a,b,\xi)$ 写为 $(x,a) \triangleleft^{\xi} (y,b)$。同样，我们通常省略 $\xi$，因为它居住在一个简单的命题中，因此并不有趣，并且我们可能经常省略 $x$ 和 $y$，简单地写作 $a \trianglelefteq b$ 或 $a \triangleleft b$。有了这些符号，归纳原理的假设如下。

- 对于定义超现实数 $x$ 的任何分割，以及

  (i) 对于每个 $L$，一个元素 $a^L : A(x^L)$，以及
  (ii) 对于每个 $R$，一个元素 $a^R : A(x^R)$，使得
  (iii) 对于所有 $L$ 和 $R$ 我们有 $(x^L, a^L) \triangleleft (x^R, a^R)$

  有一个指定的元素 $f_a : A(x)$。我们称这些数据为定义 $x$ 的分割上的**依赖分割** (dependent cut)。
- 对于 $x, y : \mathsf{No}$ 及 $a : A(x)$ 和 $b : A(y)$，如果 $x \leq y$ 且 $y \leq x$ 且 $(x,a) \trianglelefteq (y,b)$ 并且 $(y,b) \trianglelefteq (x,a)$，则 $a =^{A}_{\mathsf{eq}_{\mathsf{No}}} b$。
- 给定定义两个超现实数 $x$ 和 $y$ 的分割，以及 $x$ 上的依赖分割 $a$ 和 $y$ 上的依赖分割 $b$，如果对于所有 $L$，我们有 $x^L < y$ 且 $(x^L, a^L) \triangleleft (y, f_b)$，并且对于所有 $R$，我们有 $x < y^R$ 且 $(x, f_a) \triangleleft (y^R, b^R)$，则 $(x, f_a) \trianglelefteq (y, f_b)$。
- $\trianglelefteq$ 的值是简单命题。
- 给定定义两个超现实数 $x$ 和 $y$ 的分割，$x$ 上的依赖分割 $a$ 和 $y$ 上的依赖分割 $b$，以及一个 $L_0$，使得 $x \leq y^{L_0}$ 且 $(x, f_a) \trianglelefteq (y^{L_0}, b^{L_0})$，则我们有 $(x, f_a) \triangleleft (y, f_b)$。
- 给定定义两个超现实数 $x$ 和 $y$ 的分割，$x$ 上的依赖分割 $a$ 和 $y$ 上的依赖分割 $b$，以及一个 $R_0$，使得 $x^{R_0} \leq y$ 以及 $(x^{R_0}, a^{R_0}), \trianglelefteq (y, f_b)$，则我们有 $(x, f_a) \triangleleft (y, f_b)$。
- $\triangleleft$ 的值是简单命题。

在这些假设下，我们推导出一个函数 $f : \prod_{(x:\mathsf{No})} A(x)$，使得

$$\begin{aligned} f(x) &\equiv f_{f[x]} \\ (x \leq y) &\Rightarrow (x, f(x)) \trianglelefteq (y, f(y)) \\ (x < y) &\Rightarrow (x, f(x)) \triangleleft (y, f(y)) \end{aligned} \tag{11.6.3}$$

在点构造器的计算规则 (11.6.3) 中，$x$ 是由一个分割定义的超现实数，$f[x]$ 表示通过应用 $f$（并使用 $f$ 取 $<$ 到 $\triangleleft$ 的事实）定义的 $x$ 上的依赖分割。通常，我们将使用模式匹配符号，其中对分割 $\{\, x^L \mid x^R \,\}$ 的 $f$ 的定义可以使用符号 $f(x^L)$ 和 $f(x^R)$ 并假设它们形成依赖分割。

与 Cauchy 实数一样，我们有一些通过将 $A$，$\trianglelefteq$ 和 $\triangleleft$ 简化而得到的特殊情况。取 $\trianglelefteq$ 和 $\triangleleft$ 为常数 $\mathbf{1}$，我们有 $\mathsf{No}$-**归纳**，为简单起见，我们仅为简单属性状态：

- 给定 $P : \mathsf{No} \to \mathsf{Prop}$，如果 $P(x)$ 在由分割定义的超现实数 $x$ 且对所有 $L$ 和 $R$ 满足 $P(x^L)$ 和 $P(x^R)$ 时成立，则 $P(x)$ 对所有 $x : \mathsf{No}$ 成立。

这应该与 Conway 的评论进行比较：

> 一般来说，当我们希望为所有数 $x$ 建立一个命题 $P(x)$ 时，我们将通过从所有命题 $P(x^L)$ 和 $P(x^R)$ 的真理中推导出 $P(x)$ 来进行归纳证明。我们认为短语"所有数都是这样构造的"证明了这一过程的合法性。

通过 No-归纳，我们可以证明

Theorem 11.6.4 (Conway 定理 0).

(i) *对于任何 $x : \mathsf{No}$，我们有 $x \le x$。*
(ii) *对于由分割定义的任何 $x : \mathsf{No}$，我们有 $x^L < x$ 和 $x < x^R$ 对所有 $L$ 和 $R$ 成立。*

*证明.* 首先注意，如果 $x \le x$，那么每当 $x$ 作为某个分割 $y$ 的左选项出现时，我们有 $x < y$，通过 $<$ 的第一个构造器，每当 $x$ 作为 $y$ 的右选项出现时，我们有 $y < x$。特别是，(i)⇒(ii)。
我们通过 No-归纳来证明 (i)。因此，假设 $x$ 是由一个分割定义的，使得对所有 $L$ 和 $R$，$x^L \le x^L$ 和 $x^R \le x^R$ 成立。但通过上面的观察，这些假设意味着 $x^L < x$ 和 $x < x^R$ 对所有 $L$ 和 $R$ 成立，通过 $\le$ 的构造器，我们得到 $x \le x$。 □

Corollary 11.6.5. *No 是一个 0-类型。*

*证明.* 简单关系 $R(x,y) :\equiv (x \le y) \wedge (y \le x)$ 通过 No 的路径构造器意味着恒等，并且通过 Theorem 11.6.4(i) 包含对角线。因此，Theorem 7.2.2 适用。 □

相比之下，Conway 的定理 1（$\le$ 的传递性）用我们的定义稍难建立；参见 Corollary 11.6.17。
我们还需要联合递归原理，$(\mathsf{No}, \le, <)$-**递归**。方便起见，将其表示如下。假设 $A$ 是一个带有关系 $\trianglelefteq : A \to A \to \mathsf{Prop}$ 和 $\lhd : A \to A \to \mathsf{Prop}$ 的类型。然后我们可以通过执行以下操作来定义 $f : \mathsf{No} \to A$。

(i) 对于由一个分割定义的任意 $x$，假设 $f(x^L)$ 和 $f(x^R)$ 已定义，使得 $f(x^L) \lhd f(x^R)$ 对所有 $L$ 和 $R$ 成立，我们必须定义 $f(x)$。（我们将其称为递归的*主要条款*）。
(ii) 证明 $\trianglelefteq$ 是*反对称的*：如果 $a \trianglelefteq b$ 且 $b \trianglelefteq a$，则 $a = b$。
(iii) 对于由分割定义的 $x, y$，假设对所有 $L$，$x^L < y$，对所有 $R$，$x < y^R$，并假设归纳地 $f(x^L) \lhd f(y)$ 对所有 $L$ 成立，$f(x) \lhd f(y^R)$ 对所有 $R$ 成立，并且 $f(x^L) \lhd f(x^R)$ 和 $f(y^L) \lhd f(y^R)$ 对所有 $L$ 和 $R$ 成立，必须证明 $f(x) \trianglelefteq f(y)$。
(iv) 对于由分割定义的 $x, y$ 和一个 $L_0$ 使得 $x \le y^{L_0}$，假设归纳地 $f(x) \trianglelefteq f(y^{L_0})$，并且 $f(x^L) \lhd f(x^R)$ 和 $f(y^L) \lhd f(y^R)$ 对所有 $L$ 和 $R$ 成立，我们必须证明 $f(x) \lhd f(y)$。
(v) 对于由分割定义的 $x, y$ 和一个 $R_0$ 使得 $x^{R_0} \le y$，假设归纳地 $f(x^{R_0}) \trianglelefteq f(y)$，并且 $f(x^L) \lhd f(x^R)$ 和 $f(y^L) \lhd f(y^R)$ 对所有 $L$ 和 $R$ 成立，我们必须证明 $f(x) \lhd f(y)$。

最后三条可以更简洁地描述为我们必须证明 $f$（如在第一个条款中定义）将 $\le$ 映射为 $\trianglelefteq$，并且 $<$ 映射为 $\lhd$。我们将通过说 $f$ *保持不等性*来引用这些属性。此外，在证明 $f$ 保持不等性的过程中，我们可以假设构造器的输入是不等性 $<$ 或 $\le$ 的特定实例，并且我们也可以使用 $f$ 保持构造器输入中出现的所有不等性的归纳假设。
如果我们在上述 (i)–(v) 中取得成功，那么我们将得到 $f : \mathsf{No} \to A$，它在分割上的计算如 (i) 所规定，并且保持所有不等性：

$$\forall (x, y : \mathsf{No}).\ \Big((x \le y) \to (f(x) \trianglelefteq f(y))\Big) \wedge \Big((x < y) \to (f(x) \lhd f(y))\Big)$$

就像 Cauchy 实数的 $(\mathbb{R}_c, \sim)$-递归一样，这种递归原理在定义 No 上的函数时仍然是必不可少的，因为我们无法先定义一个"预超现实数"上的函数，然后再证明它符合等同性的概念。

Example 11.6.6. 我们定义*否定*函数 $\mathsf{No} \to \mathsf{No}$。我们将联合递归原理应用于 $A :\equiv \mathsf{No}$，其中 $(x \trianglelefteq y) :\equiv (y \le x)$，$(x \lhd y) :\equiv (y < x)$。显然，这个 $\trianglelefteq$ 是反对称的。
对于递归的主要条款，我们假设 $x$ 是由一个分割定义的，并且 $-x^L$ 和 $-x^R$ 已定义，使得 $-x^L \lhd -x^R$ 对所有 $L$ 和 $R$ 成立。根据定义，这意味着对所有 $L$ 和 $R$，$-x^R < -x^L$，因此我们可以通过分割

$\{ -x^R \mid -x^L \}$ 定义 $-x$。这个符号，跟随 Conway，指的是一个分割，其中左选项由 $x$ 的右选项的索引类型 $\mathcal{R}$ 索引，而右选项由左选项的索引类型 $\mathcal{L}$ 索引，并且相应的族 $\mathcal{R} \to \mathsf{No}$ 和 $\mathcal{L} \to \mathsf{No}$ 通过与否定组合定义。

现在我们必须验证 $f$ 保持不等性。

- 对于 $x \le y$，我们可以假设 $x^L < y$ 对所有 $L$ 和 $x < y^R$ 对所有 $R$ 成立，并证明 $-y \le -x$。但根据归纳假设，我们可以假设 $-y < -x^L$ 和 $-y^R < -x$，这给出了所需的结果，依据 $-y$、$-x$ 的定义和 $\le$ 的构造器。
- 对于 $x < y$，在它从某个 $x \le y^{L_0}$ 产生的第一种情况，我们可以归纳地假设 $-y^{L_0} \le -x$，在这种情况下，$-y < -x$ 通过 $<$ 的构造器得出。
- 类似地，如果 $x < y$ 是由 $x^{R_0} \le y$ 产生的，那么归纳假设是 $-y \le -x^R$，因此再次得到 $-y < -x$。

要做更多的事情，然而，我们需要像对 Cauchy 实数一样更明确地描述 $\le$ 和 $<$ 的关系，如在 Theorem 11.3.32 中那样。同样，在那里，为了使归纳进行下去，我们必须同时证明这些关系的一些基本性质。

Theorem 11.6.7. *存在关系 $\preceq : \mathsf{No} \to \mathsf{No} \to \mathsf{Prop}$ 和 $\prec : \mathsf{No} \to \mathsf{No} \to \mathsf{Prop}$，使得如果 $x$ 和 $y$ 是由分割定义的超现实数，则*

$$(x \preceq y) :\equiv \big(\forall(L).\, x^L \prec y\big) \wedge \big(\forall(R).\, x \prec y^R\big)$$
$$(x \prec y) :\equiv \big(\exists(L).\, x \preceq y^L\big) \vee \big(\exists(R).\, x^R \preceq y\big)$$

*此外，我们有*

$$(x \prec y) \to (x \preceq y) \tag{11.6.8}$$

*并且所有合理的传递性质使得 $\prec$ 和 $\preceq$ 成为 $\le$ 和 $<$ 上的“双模”：*

$$\begin{array}{ll} (x \le y) \to (y \preceq z) \to (x \preceq z) & (x \preceq y) \to (y \le z) \to (x \preceq z) \\ (x \le y) \to (y \prec z) \to (x \prec z) & (x \preceq y) \to (y < z) \to (x \prec z) \\ (x < y) \to (y \preceq z) \to (x \prec z) & (x \prec y) \to (y \le z) \to (x \prec z) \end{array} \tag{11.6.9}$$

*证明.* 我们通过对 $x, y$ 进行双重 $(\mathsf{No}, \le, <)$-归纳来定义 $\preceq$ 和 $\prec$。第一个归纳是一个简单的递归，其余域是 $(\mathsf{No} \to \mathsf{Prop}) \times (\mathsf{No} \to \mathsf{Prop})$ 的子集 $A$，其中包含的命题对满足“右传递性”条件，即 (11.6.8) 和 (11.6.9) 的右列，并将 $(x \preceq -)$ 和 $(x \prec -)$ 替换为给定的两个命题。正如在 Theorem 11.3.16 的证明中，我们将这些命题视为二元关系的一半，将它们写作 $y \mapsto (\diamondsuit \preceq y)$ 和 $y \mapsto (\diamondsuit \prec y)$，其中 $\diamondsuit$ 表示命题对。我们给 $A$ 配备以下两种关系：

$$(\diamondsuit \trianglelefteq \heartsuit) :\equiv \forall(y : \mathsf{No}).\, \Big((\heartsuit \preceq y) \to (\diamondsuit \preceq y)\Big) \wedge \Big((\heartsuit \prec y) \to (\diamondsuit \prec y)\Big)$$
$$(\diamondsuit \triangleleft \heartsuit) :\equiv \forall(y : \mathsf{No}).\, \Big((\heartsuit \preceq y) \to (\diamondsuit \prec y)\Big)$$

注意 $\trianglelefteq$ 是反对称的，因为如果 $\diamondsuit \trianglelefteq \heartsuit$ 和 $\heartsuit \trianglelefteq \diamondsuit$，则 $(\heartsuit \preceq y) \Leftrightarrow (\diamondsuit \preceq y)$ 和 $(\heartsuit \prec y) \Leftrightarrow (\diamondsuit \prec y)$ 对所有 $y$ 成立，因此根据简单命题的一致性和函数扩展性，$\diamondsuit = \heartsuit$。此外，将函数 $\mathsf{No} \to A$ 保持不等性是说，当它被视为一对二元关系时，它满足“左传递性”（(11.6.9) 的左列）。

现在对于递归的主要条款，我们假设 $x$ 是由一个分割定义的，并且对所有 $L$ 和 $R$，关系 $(x^L \prec -)$，$(x^R \prec -)$，$(x^L \preceq -)$ 和 $(x^R \preceq -)$ 已定义，严格的关系暗示非严格的关系，并满足右传递性，并且

$$\forall(L, R).\, \forall(y : \mathsf{No}).\, \Big((x^R \preceq y) \to (x^L \prec y)\Big) \tag{11.6.10}$$

现在我们必须定义对所有 $y$ 的 $(x \prec y)$ 和 $(x \preceq y)$。在与 Theorem 11.3.16 相反的情况下，而不是嵌套递归，我们使用嵌套归纳，以便能够使用归纳通过 $x^L < x$ 和 $x < x^R$ 的不等关系。定义 $A' : \mathsf{No} \to \mathcal{U}$，其中 $A'(y)$ 是 $\mathsf{Prop} \times \mathsf{Prop}$ 的子集 $A'$，由两个简单命题组成，分别表示 $\triangle \preceq y$ 和 $\triangle \prec y$（用 $\triangle : A'(y)$ 表示），并且

$$(\triangle \prec y) \to (\triangle \preceq y) \tag{11.6.11}$$
$$\forall(L).\, (\triangle \preceq y) \to (x^L \prec y) \tag{11.6.12}$$
$$\forall(R).\, (x^R \preceq y) \to (\triangle \prec y) \tag{11.6.13}$$

使用与 $\trianglelefteq$ 和 $\triangleleft$ 类似的符号，我们为 $A'$ 配备了由以下定义的两个关系，针对 $\triangle : A'(y)$ 和 $\square : A'(z)$：

$$(\triangle \sqsubseteq \square) :\equiv \Big((\triangle \preceq y) \to (\square \preceq z)\Big) \wedge \Big((\triangle \prec y) \to (\square \prec z)\Big)$$
$$(\triangle \sqsubset \square) :\equiv \Big((\triangle \preceq y) \to (\square \prec z)\Big)$$

再次地，$\sqsubseteq$ 显然是反对称的。此外，映射到 $A'(y)$ 的函数 $\prod_{(y:\mathsf{No})} A'(y)$ 保持不等性，是指 $x^L < x$ 和 $x < x^R$ 的不等关系在左侧保持，$y^L < y^R$ 的不等关系在右侧保持。因此，这种内部归纳将提供我们需要的来完成外部递归的主要条款。

对于内部归纳的主要条款，我们还假设 $y$ 是由一个分割定义的，并且 $(x \prec y^L)$，$(x \prec y^R)$，$(x \preceq y^L)$ 和 $(x \preceq y^R)$ 的性质已定义，对所有 $L$ 和 $R$ 满足，严格的关系暗示非严格的关系，并且左侧的 $x^L < x$ 和 $x < x^R$ 的传递性以及右侧的 $y^L < y^R$ 的传递性。我们现在可以给出定理陈述中指定的定义：

$$(x \preceq y) :\equiv (\forall(L).\, x^L \prec y) \wedge (\forall(R).\, x \prec y^R), \tag{11.6.14}$$
$$(x \prec y) :\equiv (\exists(L).\, x \preceq y^L) \vee (\exists(R).\, x^R \preceq y). \tag{11.6.15}$$

为了定义 $A'(y)$ 的一个元素，我们首先必须证明 $(x \prec y) \to (x \preceq y)$。假设 $x \prec y$ 有两种情况。一方面，如果存在 $L_0$ 使得 $x \preceq y^{L_0}$，则根据右边的传递性并考虑 $y^{L_0} < y^R$，我们得到对于所有的 $R$，$x \prec y^R$。此外，根据左边的传递性并考虑 $x^L < x$，我们有对于任何 $L$，$x^L \prec y^{L_0}$，因此根据右边的传递性，我们有 $x^L \prec y$。因此，$x \preceq y$。

另一方面，如果存在 $R_0$ 使得 $x^{R_0} \preceq y$，那么根据 (11.6.10)，我们有对于所有的 $L$，$x^L \prec y$。根据左边和右边的传递性并考虑 $x < x^{R_0}$ 和 $y < y^R$，我们对于任何 $R$ 都有 $x \prec y^R$。因此，$x \preceq y$。

我们还需要证明这些定义在左边对于 $x^L < x$ 和 $x < x^R$ 是传递的。但是如果 $x \preceq y$，那么根据定义，所有的 $L$ 都有 $x^L \prec y$；如果 $x^R \preceq y$，那么根据定义 $x \prec y$。

因此，(11.6.14) 和 (11.6.15) 确实定义了 $A'(y)$ 的一个元素。我们现在必须验证这个定义是否保持了不等式，作为到 $A'$ 的一个依赖函数，即这些关系在右边是否是传递的。请记住，在每种情况下，我们可以归纳假设它们在右边对于由不等式构造器产生的所有不等式是传递的。

- 假设 $x \preceq y$ 且 $y \leq z$，后者来自 $y^L < z$ 和 $y < z^R$ 对于所有的 $L$ 和 $R$。然后，应用到 $y < z^R$ 的内递归的归纳假设，对于任何 $R$，我们得到 $x \prec z^R$。此外，根据定义 $x \preceq y$ 意味着对于任何 $L$，$x^L \prec y$，所以通过外递归的归纳假设，我们得到 $x^L \prec z$。因此，$x \preceq z$。
- 假设 $x \preceq y$ 且 $y < z$。首先，假设 $y < z$ 来自 $y \leq z^{L_0}$。然后应用到 $y \leq z^{L_0}$ 的内递归的归纳假设，我们得到 $x \preceq z^{L_0}$，因此 $x \prec z$。
  其次，假设 $y < z$ 来自 $y^{R_0} \leq z$。然后根据定义，$x \preceq y$ 意味着 $x \prec y^{R_0}$，然后对于 $y^{R_0} \leq z$ 的内递归的归纳假设，我们得到 $x \prec z$。
- 假设 $x \prec y$ 且 $y \leq z$，后者来自 $y^L < z$ 和 $y < z^R$ 对于所有的 $L$ 和 $R$。根据定义，$x \prec y$ 仅意味着存在 $R_0$ 使得 $x^{R_0} \preceq y$ 或存在 $L_0$ 使得 $x \preceq y^{L_0}$。如果 $x^{R_0} \preceq y$，那么外递归的归纳假设得出 $x^{R_0} \preceq z$，因此 $x \prec z$。如果 $x \preceq y^{L_0}$，那么对于 $y^{L_0} < z$ 的内递归的归纳假设（该假设由 $y \leq z$ 的构造器成立）得出 $x \prec z$。

这完成了内递归。因此，对于由分割定义的任何 $x$，我们有 $(x \prec -)$ 和 $(x \preceq -)$ 由 (11.6.14) 和 (11.6.15) 定义，并在右边是传递的。

要完成外递归，我们需要验证这些定义在左边也是传递的。在对 $z$ 进行 $\mathsf{No}$-归纳后，我们得到了三个基本上与上述右边传递性相同的情况。因此，我们省略它们。 □

Theorem 11.6.16. *对于任何 $x, y : \mathsf{No}$，我们有 $(x < y) = (x \prec y)$ 和 $(x \leq y) = (x \preceq y)$。*

*证明.* 从左到右，我们使用 $(\mathsf{No}, \leq, <)$-归纳，其中 $A(x) :\equiv \mathbf{1}$，$\preceq$ 和 $\prec$ 提供了关系 $\trianglelefteq$ 和 $\triangleleft$。在所有构造器的情况下，$x$ 和 $y$ 是通过分割定义的，因此 $\preceq$ 和 $\prec$ 的定义得到评估，归纳假设适用。

从右到左，我们使用 $\mathsf{No}$-归纳假设 $x$ 和 $y$ 是通过分割定义的。但现在 $\preceq$ 和 $\prec$ 的定义，以及归纳假设，正好提供了 $\leq$ 和 $<$ 的相关构造器所需的数据。 □

Corollary 11.6.17. *在 $\mathsf{No}$ 上的关系 $\leq$ 和 $<$ 满足*

$$\forall(x, y : \mathsf{No}).\, (x < y) \to (x \leq y)$$

并且是传递的：

$$(x \le y) \to (y \le z) \to (x \le z)$$
$$(x \le y) \to (y < z) \to (x < z)$$
$$(x < y) \to (y \le z) \to (x < z).$$

与柯西实数相似，联合 $(\mathsf{No}, \le, <)$-递归原则在定义所有 No 上的操作时仍然是必要的。

Example 11.6.18. 我们通过对第一个参数进行递归，然后对第二个参数进行归纳来定义 $+ : \mathsf{No} \to \mathsf{No} \to \mathsf{No}$。对于外递归，我们将余域设为由函数 $g$ 组成的 $\mathsf{No} \to \mathsf{No}$ 的子集，使得对于所有 $x, y$，$(x < y) \to (g(x) < g(y))$ 和 $(x \le y) \to (g(x) \le g(y))$ 成立。对于这样的 $g, h$，我们定义 $(g \trianglelefteq h) :\equiv \forall (x : \mathsf{No}).\, g(x) \le h(x)$ 和 $(g \triangleleft h) :\equiv \forall (x : \mathsf{No}).\, g(x) < h(x)$。显然，$\trianglelefteq$ 是反对称的。

对于递归的主要条款，我们假设 $x$ 是通过分割定义的，函数 $(x^L + -)$ 和 $(x^R + -)$ 已定义、保持不等式，并满足 $x^L + y < x^R + y$，然后我们定义 $(x + -)$。与 Theorem 11.6.7 中一样，我们不使用内递归，而是通过进入 $A : \mathsf{No} \to \mathcal{U}$ 的家族进行内归纳，其中 $A(y)$ 是那些满足每个 $x^L + y < z$ 和每个 $x^R + y > z$ 的 $z : \mathsf{No}$ 的子集。我们给 $A$ 配备由 No 诱导的不等式 $\le$ 和 $<$，因此反对称性是显而易见的。对于内递归的主要条款，我们还假设 $y$ 是通过分割定义的，并且每个 $x + y^L$ 和 $x + y^R$ 都已定义并满足 $x^L + y^L < x + y^L$，$x^L + y^R < x + y^R$，$x + y^L < x^R + y^L$ 和 $x + y^R < x^R + y^R$（这些来自施加在 $A(y)$ 元素上的附加条件），并且 $x + y^L < x + y^R$（因为 $x + y^L$ 和 $x + y^R$ 是 $A(y)$ 的依赖分割）。现在我们给出康威的定义：

$$x + y :\equiv \{\, x^L + y, x + y^L \mid x^R + y, x + y^R \,\}.$$

换句话说，$x + y$ 的左选项是形式为 $x^L + y$ 的所有数字，其中 $x^L$ 是左选项，或者 $x + y^L$，其中 $y^L$ 是左选项。我们必须证明这些左选项中的每一个都小于这些右选项中的每一个：

- 根据外归纳假设，$x^L + y < x^R + y$。
- 根据 $(x^L + -)$ 保持不等式的事实，$x^L + y < x^L + y^R < x + y^R$，而第二个不等式由于 $x + y^R : A(y^R)$ 成立。
- 根据 $x + y^L : A(y^L)$ 和 $(x^R + -)$ 保持不等式的事实，$x + y^L < x^R + y^L < x^R + y$。
- 根据内归纳假设（特别是我们有一个依赖分割的事实），$x + y^L < x + y^R$。

我们还必须证明这样定义的 $x + y$ 位于 $A(y)$ 中，即 $x^L + y < x + y$ 和 $x + y < x^R + y$；但这由 Theorem 11.6.4(ii) 成立。

接下来我们必须验证 $(x + -)$ 的定义是否保持不等式：

- 如果 $y \le z$ 来自知道 $y^L < z$ 和 $y < z^R$ 对于所有的 $L$ 和 $R$，那么内归纳假设得出 $x + y^L < x + z$ 和 $x + y < x + z^R$，而外归纳假设得出 $x^L + y \le x^L + z$ 和 $x^R + y \le x^R + z$。此外，由于 $x^R + y$ 是 $x + y$ 的一个右选项，我们有 $x + y < x^R + y$。类似地，我们发现 $x^L + z$ 是 $x + z$ 的一个左选项，因此 $x^L + z < x + z$。因此，使用传递性，我们有 $x^L + y < x + z$ 和 $x + y < x^R + z$；所以我们可以通过 $\le$ 的构造器得出 $x + y \le x + z$。
- 如果 $y < z$ 来自 $L_0$ 使得 $y \le z^{L_0}$，那么归纳得出 $x + y \le x + z^{L_0}$，因此 $x + y < x + z$，因为 $x + z^{L_0}$ 是 $x + z$ 的一个右选项。
- 类似地，如果 $y < z$ 来自 $y^{R_0} \le z$，那么 $x + y < x + z$ 因为 $x + y^{R_0} \le x + z$。

这完成了内递归。对于外递归，我们必须验证 $+$ 在左边也保持不等式。在 No-归纳后，这个过程与之前的完全相同。

在 [?] 的零部分附录中，康威讨论了如何在 ZFC 集合论中形式化超现实数：通过沿着序数迭代并转向每个等价类的最低等级的代表集，或者通过用“符号扩展”来表示数字。然后他指出

> 这些构造的奇异复杂性更多地告诉我们有关 ZF 中形式化的性质，而不是我们数字系统的性质…

并继续倡导一种“允许的构造类型”的一般理论，该理论应包括

(i) 可以以任何合理的构造方式从早期对象创建新对象。

(ii) 在创建对象之间的平等可以是任何所需的等价关系。

条件 (i) 可以自然地理解为证明*归纳定义*的一般原则是合理的，例如在 §§5.6 and 5.7 中提出的那些原则。特别是，构造器的严格正性条件可以被视为"合理构造"的形式化。然后条件 (ii) 建议我们应将其扩展到各种*高阶归纳定义*，其中我们可以引入路径构造器以使对象以任何合理的方式相等。例如，在下一段中康威说：

> …我们还可以自由创建一个新对象 $(x,y)$ 并将其称为 $x$ 和 $y$ 的有序对。我们还可以创建一个与 $(x,y)$ 不同但与之共存的有序对 $[x,y]$…如果我们想将 $(x,y)$ 变成无序对，我们可以通过等价关系 $(x,y)=(z,t)$ 来定义平等，当且仅当 $x=z, y=t$ 或者 $x=t, y=z$。

引入具有新名称的新对象的自由，生成特定形式的构造器，正是我们在归纳定义理论中所拥有的自由。正如我们在 §5.2 中的自然数 $\mathbb{N}$ 和 $\mathbb{N}'$ 的两个副本，如果我们写下一个与笛卡尔积类型 $A \times B$ 相同的定义，我们将获得一个不同的积类型 $A \times' B$，其规范元素我们可以自由地写作 $[x,y]$。并且我们可以通过添加合适的路径构造器使其中一个成为无序对的类型。
当然，康威的观点不是抱怨 ZF，特别是，而是反对所有的基础理论：

> …这个提议不是作为 ZF 的任何特定理论的替代品… 提议的是我们赋予自己创造这些种类的任意数学理论的自由，但证明一个元定理，该定理一劳永逸地确保任何这种理论都可以用任何标准的基础理论形式化。

有人可能会回应说，实际上，单一性的基础不是康威所设想的"标准基础理论"之一，而是我们可以表达我们创造新理论的能力的*元理论*，并且关于它我们可以证明康威的元定理。例如，超现实数是康威心中的"数学理论"之一，我们已经看到它们可以在单一性基础内构造并得到证明。同样，康威早些时候提到

> …集合论将是这样的一种理论，集合通过对应于通常公理的过程从早期集合中构造出来，平等关系就是具有相同成员的关系。

这个描述与 §10.5 中累积层次的高阶归纳构造非常吻合。康威的元定理将对应于我们多次提到的事实，即我们可以在 ZFC 内构造单一性基础的模型（这超出了本书的范围）。
然而，单一性基础本身如此丰富和强大，以至于仅将其归为构造类似集合的理论的元理论是愚蠢的。我们已经看到，即使在集合（0-类型）的层面，单一性基础中的高阶归纳类型通过它们的通用属性（§6.11）直接构造对象，例如柯西完备性的构造性理论（§11.3）。但最重要的是，直接在基础系统中建模同伦论和范畴论的潜力（Chapters 8 and 9）赋予单一性基础以任何集合论基础都无法比拟的优势。

## Notes

定义 Dedekind 实数上的代数运算，特别是乘法，既有点棘手又有点繁琐。有几种方法可以启动算术：每种方法都有其优势，但它们似乎都需要一些技术工作。例如，Richman [?] 首先在正切割上定义了 Dedekind 实数的乘法，然后将其代数扩展到所有 Dedekind 切割，而 Conway [?] 则观察到，超现实数的乘法定义对于 Dedekind 实数来说效果很好。
我们对 Dedekind 实数的处理借鉴了 [?] 的许多想法，在该文献中，Dedekind 实数是在抽象 Stone 对偶性的背景下构造的。这是一种（受限的）简单类型的 $\lambda$-演算，其中有一个区分的对象 $\Sigma$ 用于分类开集，并且通过对偶也分类闭集。在 [?] 中，您还可以找到算术运算基本性质的详细证明。
$\mathbb{R}_c$ 是最小的柯西完备阿基米德有序域这一事实，如在 Theorem 11.3.50 中证明的那样，表明我们的柯西实数可能与 Escardó-Simpson 实数 [?] 一致。检查这是否真的如此将很有趣。Escardó-Simpson 实数或更准确地说是相应的闭区间的概念很有趣，因为它可以在任何具有有限积的范畴中表述。
在通过"正规扩展公理"增强的构造性集合论中，您还可以尝试通过在柯西序列的极限下进行闭合并进行超限迭代来定义柯西完备性。检查此构造是否与我们的构造一致也将很有趣。
众所周知，柯西实数与 Dedekind 实数的一致性需要依赖选择，但不太为人所知的是，可数选择就足够了。请记住**依赖选择** 声明，对于 $A$ 上的总关系 $R$，我们意味着 $\forall(x:A).\exists(y:A).R(x,y)$，并且对于任何 $a:A$，仅存在 $f:\mathbb{N}\to A$ 使得 $f(0)=a$ 并且对于所有 $n:\mathbb{N}$，$R(f(n),f(n+1))$ 成立。我们的 Corollary 11.4.3 使用了将依赖选择的应用转换为使用可数选择的典型技巧。即，我们使用可数选择来预先做出所有可能出现的选择，然后使用选择函数来避免依赖选择。
在构造性背景下，各种紧致性概念之间的复杂关系在 [?] 中有所讨论。Palmgren [?] 对逐点分析和点自由拓扑进行了良好的比较。

超现实数是由 [?] 定义的，使用了一种归纳定义的形式，但没有在任何基础系统中显式证明。出于这个原因，一些后来的作者倾向于使用符号扩展或其他更明确的表示，这些表示可以更明显地编码到集合论中。在类型论中表示它们的想法最早由 Hancock 提出，而 Setzer 和 Forsberg [?] 注意到超现实数及其不等式关系 $<$ 和 $\leq$ 自然地形成了一个归纳-归纳定义。这里提出的高阶归纳-归纳版本，内置了超现实数的正确平等概念，是新的。

## Exercises

Exercise 11.1. 给出 Dedekind 实数的替代定义，首先定义平方，然后使用 Eq. (11.3.45)。检查是否获得了一个交换环。

Exercise 11.2. 假设我们去除了 Definition 11.2.1 中的有界性条件 (i)。那么我们得到**扩展实数** 它们包含 $-\infty :\equiv (\mathbf{0}, \mathbb{Q})$ 和 $\infty :\equiv (\mathbb{Q}, \mathbf{0})$。哪些切割上算术运算的定义仍然适用于扩展实数？我们得到什么代数结构？

Exercise 11.3. 通过考虑单侧切割，我们得到下和上 Dedekind 实数，分别。例如，下实数由谓词 $L : \mathbb{Q} \to \Omega$ 给出

(i) 非空： $\exists(q : \mathbb{Q}).\, L(q)$ 和
(ii) 舍入： $L(q) = \exists(r : \mathbb{Q}).\, q < r \wedge L(r)$。

（我们还可以要求 $\exists(r : \mathbb{Q}).\, \neg L(r)$ 以排除切割 $\infty :\equiv \mathbb{Q}$。）您可以在下实数上定义哪些算术运算？特别是，发生了什么加法逆运算？

Exercise 11.4. 假设我们去除了 Definition 11.2.1 中的定点性条件。那么我们得到**区间域** $\mathbb{I}$ 因为切割允许存在“间隙”，即只是区间。通过定义偏序 $\sqsubseteq$ 在 $\mathbb{I}$ 上通过

$$((L, U) \sqsubseteq (L', U')) :\equiv (\forall(q : \mathbb{Q}).\, L(q) \Rightarrow L'(q)) \wedge (\forall(q : \mathbb{Q}).\, U(q) \Rightarrow U'(q))$$

什么是相对于 $\mathbb{I}$ 的 $\mathbb{I}$ 的最大元素？定义“端点”操作，它将区间域的元素分配到它的下端点和上端点。端点是实数、下实数还是上实数（参见 Exercise 11.3）？哪些切割上的算术运算定义仍然适用于区间域？

Exercise 11.5. 显示对于所有 $x, y : \mathbb{R}_{\mathsf{d}}$，

$$\neg(x < y) \Rightarrow y \leq x$$

和

$$(x \leq y) \simeq \left(\prod_{\epsilon:\mathbb{Q}_+} x < y + \epsilon\right)$$

$\neg(x \leq y)$ 是否意味着 $y < x$？

Exercise 11.6.

(i) 假设排中律，构造一个非常数映射 $\mathbb{R}_{\mathsf{d}} \to \mathbb{Z}$。
(ii) 假设 $f : \mathbb{R}_{\mathsf{d}} \to \mathbb{Z}$ 是一个映射使得 $f(0) = 0$ 并且对于所有 $x > 0$，$f(x) \neq 0$。从中推导出有限全知原理 (11.5.8)。

Exercise 11.7. 显示在一个有序域 $F$ 中，$\mathbb{Q}$ 的密度和传统的阿基米德公理是等价的：

$$(\forall(x, y : F).\, x < y \Rightarrow \exists(q : \mathbb{Q}).\, x < q < y) \Leftrightarrow (\forall(x : F).\, \exists(k : \mathbb{Z}).\, x < k)$$

Exercise 11.8. 假设 $a, b : \mathbb{Q}$ 和 $f : \{\, q : \mathbb{Q} \mid a \leq q \leq b \,\} \to \mathbb{R}_{\mathsf{c}}$ 是常数为 $L$ 的 Lipschitz 映射。显示存在一个唯一的延拓 $\bar{f} : [a, b] \to \mathbb{R}_{\mathsf{c}}$ 的 Lipschitz 映射常数 $L$。提示：不要重新执行 Lemma 11.3.15 对于封闭区间，请注意有一个缩回 $r : \mathbb{R}_{\mathsf{c}} \to [-n, n]$ 并应用 Lemma 11.3.15 对 $f \circ r$。

Exercise 11.9. 将 $\mathbb{R}_{\mathsf{c}}$ 的构造推广到任何度量空间的柯西完备性。首先，考虑哪种实数概念最自然作为度量空间距离的余域。它重要吗？接下来，详细研究两种构造：

(i) 按照柯西实数的构造，将度量空间的完备性定义为在柯西序列的极限下闭合的归纳-归纳类型。
(ii) 使用以下构造，由 Lawvere [?] 和 Richman [?] 提出，其中度量空间 $(M, d)$ 的完备性由**位置**的类型给出。位置是一个函数 $f : M \to \mathbb{R}$ 使得

(a) $f(x) \geq |f(y) - d(x,y)|$ 对于所有 $x, y : M$，并且
(b) $\inf_{x \in M} f(x) = 0$，我们意味着 $\forall(\epsilon : \mathbb{Q}_+).\, \exists(x : M).\, |f(x)| < \epsilon$ 和 $\forall(x : M).\, f(x) \geq 0$。

这个想法是 $f$ 看起来像是在测量距离从一个点。

最后，证明度量完备性的以下通用属性：从度量空间到柯西完备度量空间的局部一致连续映射可以唯一地延拓到完备性上的局部一致连续映射。（我们说映射是**局部一致连续的** 如果它在开放球上是一致连续的。）

Exercise 11.10. **马尔可夫原理** 表示对于所有 $f : \mathbb{N} \to \mathbf{2}$，

$$(\neg\neg\exists(n : \mathbb{N}).\, f(n) = 1_{\mathbf{2}}) \Rightarrow \exists(n : \mathbb{N}).\, f(n) = 1_{\mathbf{2}}$$

这是双重否定律的一个特例 (3.4.2)。显示 $\forall(x, y : \mathbb{R}_{\mathsf{d}}).\, x \neq y \Rightarrow x \# y$ 意味着马尔可夫原理。是反过来也成立?

Exercise 11.11. 验证以下“无零因子”性质是否适用于实数：$xy \# 0 \Leftrightarrow x \# 0 \wedge y \# 0$。

Exercise 11.12. 假设 $(q_1, r_1), \ldots, (q_n, r_n)$ 逐点覆盖 $(a, b)$。那么有 $\epsilon : \mathbb{Q}_+$ 使得每当 $a < x < y < b$ 并且 $|x - y| < \epsilon$ 然后仅存在 $i$ 使得 $q_i < x < r_i$ 并且 $q_i < y < r_i$。这样的 $\epsilon$ 被称为**勒贝格数** 对于给定的覆盖。

Exercise 11.13. 证明中值定理的以下近似版本：

*如果 $f : [0,1] \to \mathbb{R}$ 是一致连续的并且 $f(0) < 0 < f(1)$ 然后对于每个 $\epsilon : \mathbb{Q}_+$ 仅存在 $x : [0,1]$ 使得 $|f(x)| < \epsilon$。*

提示：不要尝试使用二分法，因为它会导致选择公理。相反，用分段线性图逼近 $f$。您如何构造分段线性图?

Exercise 11.14. 检查是否可以使用 §11.6 的高阶归纳-归纳超现实数来完成 [?] 中的所有操作。

Exercise 11.15. 回忆在 278 页定义的函数 $\iota_{\mathbb{R}_{\mathsf{d}}} : \mathbb{R}_{\mathsf{d}} \to \mathsf{No}$。

(i) 显示 $\iota_{\mathbb{R}_{\mathsf{d}}}$ 是单射的。
(ii) 有显然的扩展 $\iota_{\mathbb{R}_{\mathsf{d}}}$ 到扩展实数（Exercise 11.2）和区间域（Exercise 11.4）。它们是单射的吗?

Exercise 11.16. 显示在 278 页定义的函数 $\iota_{\mathsf{Ord}} : \mathsf{Ord} \to \mathsf{No}$ 是单射的，当且仅当 LEM 成立。

Exercise 11.17. 定义一个类型 $\mathsf{POrd}$，配备二元关系 $\leq$ 和 $<$，通过模仿 No 的定义，但只使用左选项。

(i) 构造一个映射 $j : \mathsf{POrd} \to \mathsf{No}$ 并显示它是嵌入的。
(ii) 显示 $\mathsf{POrd}$ 是一个序数（在下一个更高的宇宙中，如 Ord ）在关系 $<$ 下。
(iii) 假设命题缩放，显示 $\mathsf{POrd}$ 等价于子集

$$\{\, A : \mathsf{Ord} \mid \mathsf{isPlump}(A) \,\}$$

来自 Exercise 10.14 的 Ord 。结论 $\iota_{\mathsf{Ord}} : \mathsf{Ord} \to \mathsf{No}$ 是单射的，当限制为饱满序数时。

在没有命题缩放的情况下，我们仍然可以将 $\mathsf{POrd}$（或它们在 No 中的图像）称为**饱满序数**。

Exercise 11.18. 定义一个超现实数为**伪序数** 如果它等于一个没有右选项的分割 $\{\, x^L \mid \,\}$（但其左选项本身可能有右选项）。显示声明“每个伪序数都是饱满序数”当且仅当等价于 LEM 。

Exercise 11.19. 注意 Theorem 11.6.7 和 Example 11.6.18 都使用类似的模式来定义一个函数 $\mathsf{No} \to \mathsf{No} \to B$：一个外 No-递归，其余域是从 No 到 $B$ 的保序函数的集合，接着是内 No-归纳到一个家族 $A : \mathsf{No} \to \mathcal{U}$，其中 $A(y)$ 是 $B$ 的子集，确保不等式 $x^L < x$ 和 $x < x^R$ 也被保留。制定并证明一个“双 No-递归”的通用原则，它推广了这些证明。

# Appendix

附录 A

# 形式类型论 (Formal Type Theory)

正如可以在不显式使用策梅洛-弗兰克尔集合论的公理的情况下在集合论中发展数学一样，在本书中，我们在不明确提及同伦类型论形式系统的情况下，在单值基础上发展了数学。然而，拥有对同伦类型论作为一个形式系统的精确定义是很重要的，例如，为了

- 陈述并证明其元理论性质，包括逻辑一致性，
- 构造模型，例如在单纯集合、模型范畴、更高拓扑斯等中，
- 将其在 Coq 或 Agda 等证明助手中实现。

即使是同伦类型论的逻辑一致性，即在空上下文中没有术语 $a : \mathbf{0}$，也是不明显的：如果我们错误地选择了一个定义，使得 $\mathbf{0} \simeq \mathbf{1}$，那么单值性将暗示 $\mathbf{0}$ 有一个元素，因为 $\mathbf{1}$ 有一个元素。我们的 $\mathbf{S}^1$ 作为一个高阶归纳类型的定义是否能够表现得像普通的圆，也是不明显的。
在处理这些问题之前，我们必须明确类型论的两个方面。回想一下，在引言中我们提到，类型论包含一组规则，用来规定何时判断 $a : A$ 和 $a \equiv a' : A$ 成立——例如，积的规则是：当 $a : A$ 和 $b : B$ 时，$(a, b) : A \times B$。要使这一点精确，我们首先必须精确定义术语的语法——这些判断所涉及的对象 $a, a', A, \ldots$；然后，我们必须精确定义判断及其推理规则——即如何从其他判断中推导出判断。
在本附录中，我们将介绍 Martin-Löf 类型论及构成同伦类型论的扩展的两种表述。第一种表述 (Appendix A.1) 描述了作为无类型 $\lambda$-演算扩展的术语的语法和判断的形式，而推理规则则保持非正式状态。第二种表述 (Appendix A.2) 以自然演绎的方式递归地定义了术语、判断和推理规则，这在许多类型论文献中是常见的做法。

## 预备知识 (Preliminaries)

在 Chapter 1 中，我们介绍了类型论的两个基本**判断** (judgments)。第一个判断 $a : A$ 断言一个术语 $a$ 具有类型 $A$。第二个判断 $a \equiv b : A$ 断言两个术语 $a$ 和 $b$ 在类型 $A$ 下是**判断等同的** (judgmentally equal)。这些判断是由 Appendix A.2 中描述的一组推理规则递归定义的。
构造类型 $A$ 的元素 $a$ 就是推导出 $a : A$；在本书中，我们给出了描述 $a$ 构造的非正式论证，但在正式场合中，必须具体指定一个术语 $a$ 以及 $a : A$ 的完整推导。
然而，本书中的类型论表述与本附录中的表述之间的主要区别在于，这里判断是在显式的**上下文** (context)中进行表述的，或假设的列表，形式为

$$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n.$$

上下文中的元素 $x_i : A_i$ 表示变量 $x_i$ 拥有类型 $A_i$ 的假设。上下文中出现的变量 $x_1, \ldots, x_n$ 必须是不同的。我们用字母 $\Gamma$ 和 $\Delta$ 来表示上下文。

上下文 $\Gamma$ 中的判断 $a : A$ 记作

$$\Gamma \vdash a : A$$

表示在 $\Gamma$ 列出的假设下 $a : A$。当假设列表为空时，我们仅写作

$$\vdash a : A$$

或

$$\cdot \vdash a : A$$

其中 $\cdot$ 表示空上下文。相同的适用于等同性判断

$$\Gamma \vdash a \equiv b : A$$

然而，这些判断只有在**上下文良构** (well-formed)时才有意义，这一概念由我们最后的判断表达

$$(x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n)\ \mathsf{ctx}$$

表示在上下文 $x_1 : A_1, x_2 : A_2, \ldots, x_{i-1} : A_{i-1}$ 中，每个 $A_i$ 都是一个类型。因此，特别地，如果 $\Gamma \vdash a : A$ 且 $\Gamma\ \mathsf{ctx}$，那么我们知道每个 $A_i$ 仅包含变量 $x_1, \ldots, x_{i-1}$，而 $a$ 和 $A$ 仅包含变量 $x_1, \ldots, x_n$。
在非正式的数学表述中，上下文是隐含的。在证明的每个阶段，数学家都知道哪些变量是可用的，以及它们的类型是什么，要么是基于历史惯例（$n$ 通常是一个数，$f$ 是一个函数，等等），要么是因为变量是通过类似“设 $x$ 为一个实数”这样的句子明确引入的。我们将在 Appendices A.2.4 and A.2.5 中讨论使用显式上下文的一些好处。
我们用 $B[a/x]$ 表示术语 $a$ 代替术语 $B$ 中自由出现的变量 $x$ 的**替换** (substitution)，同时可能会进行避免捕获的绑定变量重命名，如 §1.2 中所讨论。替换的一般形式

$$B[a_1, \ldots, a_n / x_1, \ldots, x_n]$$

将表达式 $a_1, \ldots, a_n$ 同时替换为变量 $x_1, \ldots, x_n$。
**在表达式 $B$ 中绑定变量** $x$ (bind a variable $x$ in an expression $B$)意味着将两者结合到一个更大的表达式中，称为**抽象** (abstraction)，其目的是表达 $x$ 是“局部的”于 $B$，即不应与其他地方出现的 $x$ 混淆。绑定变量对程序员来说是熟悉的，但对数学家来说却不那么熟悉。绑定有各种记法，例如 $x \mapsto B$、$\lambda x.B$ 和 $x\,.B$，具体情况视场景而定。我们可以用 $C[a]$ 表示在抽象表达式中将术语 $a$ 替换为变量的替换，即我们可以定义 $(x.B)[a]$ 为 $B[a/x]$。如 §1.2 中所讨论的，改变一个表达式内的绑定变量名（“$\alpha$-转换”）不会改变表达式。因此，非常精确地说，一个表达式是一个语法形式的等价类，它们在绑定变量名上有所不同。
我们还可以认为判断

$$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n \vdash a : A$$

中的每个变量 $x_i$ 都绑定在其**作用域** (scope)中，该作用域由表达式 $A_{i+1}, \ldots, A_n$、$a$ 和 $A$ 组成。

## A.1 第一次表述 (The First Presentation)

我们的类型论中的对象和类型可以使用以下语法作为术语来表示，这是一种扩展了 $\lambda$-演算的语法，包含了*变量* $x, x', \ldots$、*原始常量* (primitive constants) $c, c', \ldots$、*定义常量* (defined constants) $f, f', \ldots$ 以及术语形成操作：

$$t ::= x \mid \lambda x.t \mid t(t') \mid c \mid f$$

此处使用的符号表示术语 $t$ 要么是变量 $x$，要么是形式为 $\lambda x.t$ 的表达式，其中 $x$ 是一个变量，$t$ 是一个术语；或者它是形式为 $t(t')$ 的表达式，其中 $t$ 和 $t'$ 都是术语；或者它是一个原始常量 $c$；或者它是一个定义常量 $f$。语法标记'$\lambda$'、'('、')' 和',' 是为了指导人的眼睛的标点符号。
我们使用 $t(t_1, \ldots, t_n)$ 作为重复应用 $t(t_1)(t_2)\ldots(t_n)$ 的缩写。我们还可以使用*中缀* (infix)表示法，当 $\star$ 是一个原始或定义的常量时，写作 $t_1 \star t_2$ 表示 $\star(t_1, t_2)$。

每个定义常量有零个、一个或多个**定义方程** (defining equations)。定义常量有两种类型。显式定义常量 $f$ 具有一个定义方程

$$f(x_1,\ldots,x_n) :\equiv t,$$

其中 $t$ 不涉及 $f$。例如，我们可以引入显式定义常量 $\circ$，其定义方程为

$$\circ(x,y)(z) :\equiv x(y(z)),$$

并使用中缀符号 $x \circ y$ 表示 $\circ(x,y)$。当然，这只是函数的复合。
第二种定义常量用于指定一个（参数化的）映射 $f(x_1,\ldots,x_n,x)$，其中 $x$ 范围为元素由零个或多个原始常量生成的类型。对于每个这样的原始常量 $c$，都有一个形式为

$$f(x_1,\ldots,x_n,c(y_1,\ldots,y_m)) :\equiv t,$$

的定义方程，其中 $f$ 可以出现在 $t$ 中，但只能以确定方程定义了一个完全定义的函数的方式出现。这类定义函数的范例是自然数上的原始递归定义的函数。我们可以称这种函数的定义为**完全递归定义** (total recursive definition)。在计算机科学和逻辑学中，这种在递归数据类型上定义函数的方式被称为**结构递归定义** (definition by structural recursion)。
**术语的可转换性** (Convertibility)$t \downarrow t'$ 是由常量的定义方程、计算规则生成的等价关系

$$(\lambda x.\, t)(u) :\equiv t[u/x],$$

以及使其成为应用和 $\lambda$-抽象的同余的规则生成的等价关系：

- 如果 $t \downarrow t'$ 且 $s \downarrow s'$，则 $t(s) \downarrow t'(s')$，并且
- 如果 $t \downarrow t'$，则 $(\lambda x.\, t) \downarrow (\lambda x.\, t')$。

等同性判断 $t \equiv u : A$ 可以通过以下单一规则得出：

- 如果 $t : A$，$u : A$，并且 $t \downarrow u$，那么 $t \equiv u : A$。

判断等同性是一个等价关系。
注意，本节中使用的类型论与正文中使用的类型论在不包含函数的判断唯一性原则 $f \equiv (\lambda x.\, f(x))$ 方面有所不同。这种等同性要求判断等同性对所涉及术语的类型敏感，因为这种等同性只有在 $f$ 被认为是一个函数时才有意义，而在本节中的可转换关系与类型无关。Appendix A.2中的第二种表述包含了唯一性原则。

### A.1.1 类型宇宙 (Type Universes)

我们假设有一个**宇宙** (universes)的层次结构，由原始常量表示

$$\mathcal{U}_0, \quad \mathcal{U}_1, \quad \mathcal{U}_2, \quad \ldots$$

关于宇宙的前两个规则表明它们形成了一个累积的类型层次结构：

- $\mathcal{U}_m : \mathcal{U}_n$ 对于 $m < n$，
- 如果 $A : \mathcal{U}_m$ 且 $m \leq n$，那么 $A : \mathcal{U}_n$，

第三个规则表示宇宙中的一个对象可以作为类型并在判断中出现在冒号的右边：

- 如果 $\Gamma \vdash A : \mathcal{U}_n$，并且 $x$ 是一个新变量，[1]那么 $\vdash (\Gamma, x : A)\ \mathsf{ctx}$。

在正文中，类型 $A$ 和 $B$ 之间的等同性判断 $A \equiv B : \mathcal{U}_n$ 通常简写为 $A \equiv B$。这是一种典型的歧义，因为我们可以随时切换到更大的宇宙，但这不会影响判断的有效性。
以下转换规则允许我们在类型判断中用一个类型替换为与之等同的类型：

- 如果 $a : A$ 且 $A \equiv B$，那么 $a : B$。

[1]“新”表示它没有出现在 $\Gamma$ 或 $A$ 中。

### A.1.2 依赖函数类型 (Π-types)

我们引入一个原始常量 $c_\Pi$，但将 $c_\Pi(A, \lambda x.B)$ 写作 $\prod_{(x:A)} B$。通过以下规则引入了关于此类表达式和形式为 $\lambda x.b$ 的表达式的判断：

- 如果 $\Gamma \vdash A : \mathcal{U}_n$ 且 $\Gamma, x : A \vdash B : \mathcal{U}_n$，那么 $\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}_n$
- 如果 $\Gamma, x : A \vdash b : B$，那么 $\Gamma \vdash (\lambda x.b) : (\prod_{(x:A)} B)$
- 如果 $\Gamma \vdash g : \prod_{(x:A)} B$ 且 $\Gamma \vdash t : A$，那么 $\Gamma \vdash g(t) : B[t/x]$

如果 $x$ 在 $B$ 中不是自由的，我们将 $\prod_{(x:A)} B$ 简写为非依赖函数类型 $A \to B$，并推导出以下规则：

- 如果 $\Gamma \vdash g : A \to B$ 且 $\Gamma \vdash t : A$，那么 $\Gamma \vdash g(t) : B$

使用非依赖函数类型并隐式保留上下文 $\Gamma$，可以将上述规则写成以下替代风格，我们将在附录的本节其余部分中使用：

- 如果 $A : \mathcal{U}_n$ 且 $B : A \to \mathcal{U}_n$，那么 $\prod_{(x:A)} B(x) : \mathcal{U}_n$
- 如果 $x : A \vdash b : B(x)$ 那么 $\lambda x.b : \prod_{(x:A)} B(x)$
- 如果 $g : \prod_{(x:A)} B(x)$ 且 $t : A$ 那么 $g(t) : B(t)$

### A.1.3 依赖对类型 (Σ-types)

我们引入原始常量 $c_\Sigma$ 和 $c_{\mathsf{pair}}$。形式为 $c_\Sigma(A, \lambda a.B)$ 的表达式写作 $\sum_{(a:A)} B$，形式为 $c_{\mathsf{pair}}(a, b)$ 的表达式写作 $(a, b)$。如果 $x$ 在 $B$ 中不是自由的，我们将 $\sum_{(x:A)} B$ 写作 $A \times B$。
通过以下规则引入了关于此类表达式的判断：

- 如果 $A : \mathcal{U}_n$ 且 $B : A \to \mathcal{U}_n$，那么 $\sum_{(x:A)} B(x) : \mathcal{U}_n$
- 此外，如果 $a : A$ 且 $b : B(a)$，那么 $(a, b) : \sum_{(x:A)} B(x)$

如果我们有如上的 $A$ 和 $B$，$C : (\sum_{(x:A)} B(x)) \to \mathcal{U}_m$，并且

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

我们可以引入一个定义常量

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

其定义方程为

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

注意，$C$、$d$、$x$ 和 $y$ 可能包含一些额外的隐式参数 $x_1, \ldots, x_n$，如果它们是在某个非空上下文中获得的；因此，完全显式的递归模式为

$$f(x_1, \ldots, x_n, (x(x_1, \ldots, x_n), y(x_1, \ldots, x_n))) :\equiv d(x_1, \ldots, x_n, (x(x_1, \ldots, x_n), y(x_1, \ldots, x_n)))$$

### A.1.4 余积类型 (Coproduct Types)

我们引入原始常量 $c_+$、$c_{\mathsf{inl}}$ 和 $c_{\mathsf{inr}}$。我们用 $A + B$ 代替 $c_+(A, B)$，用 $\mathsf{inl}(a)$ 代替 $c_{\mathsf{inl}}(a)$，用 $\mathsf{inr}(a)$ 代替 $c_{\mathsf{inr}}(a)$：

- 如果 $A, B : \mathcal{U}_n$，则 $A + B : \mathcal{U}_n$
- 此外，$\mathsf{inl} : A \to A + B$ 和 $\mathsf{inr} : B \to A + B$

如果我们有上述的 $A$ 和 $B$，$C : A + B \to \mathcal{U}_m$，$d : \prod_{(x:A)} C(\mathsf{inl}(x))$ 和 $e : \prod_{(y:B)} C(\mathsf{inr}(y))$，那么我们可以引入一个定义常量 $f : \prod_{(z:A+B)} C(z)$，其定义方程为

$$f(\mathsf{inl}(x)) :\equiv d(x) \qquad \text{和} \qquad f(\mathsf{inr}(y)) :\equiv e(y)$$

### A.1.5 有限类型 (The Finite Types)

我们引入原始常量 $\star$、$\mathbf{0}$ 和 $\mathbf{1}$，满足以下规则：

- $\mathbf{0} : \mathcal{U}_0$，$\mathbf{1} : \mathcal{U}_0$
- $\star : \mathbf{1}$

给定 $C : \mathbf{0} \to \mathcal{U}_n$，我们可以引入一个定义常量 $f : \prod_{(x:\mathbf{0})} C(x)$，没有定义方程。
给定 $C : \mathbf{1} \to \mathcal{U}_n$ 和 $d : C(\star)$，我们可以引入一个定义常量 $f : \prod_{(x:\mathbf{1})} C(x)$，其定义方程为 $f(\star) :\equiv d$。

### A.1.6 自然数 (Natural Numbers)

自然数的类型通过引入原始常量 $\mathbb{N}$、0 和 $\mathsf{succ}$ 获得，并满足以下规则：

- $\mathbb{N} : \mathcal{U}_0$，
- $0 : \mathbb{N}$，
- $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$。

此外，我们可以通过原始递归定义函数。如果我们有 $C : \mathbb{N} \to \mathcal{U}_k$，当我们有

$$d : C(0)$$
$$e : \prod_{(x:\mathbb{N})} (C(x) \to C(\mathsf{succ}(x)))$$

时，我们可以引入定义常量 $f : \prod_{(x:\mathbb{N})} C(x)$，其定义方程为

$$f(0) :\equiv d \text{和} \qquad f(\mathsf{succ}(x)) :\equiv e(x, f(x))$$

### A.1.7 $W$-类型 ($W$-Types)

对于 $W$-类型，我们引入原始常量 $c_{\mathsf{W}}$ 和 $c_{\mathsf{sup}}$。形式为 $c_{\mathsf{W}}(A, \lambda x.\, B)$ 的表达式写作 $\mathsf{W}_{(x:A)} B$，形式为 $c_{\mathsf{sup}}(x, u)$ 的表达式写作 $\mathsf{sup}(x, u)$：

- 如果 $A : \mathcal{U}_n$ 和 $B : A \to \mathcal{U}_n$，则 $\mathsf{W}_{(x:A)} B(x) : \mathcal{U}_n$
- 此外，如果 $a : A$ 和 $u : B(a) \to \mathsf{W}_{(x:A)} B(x)$，则 $\mathsf{sup}(a, u) : \mathsf{W}_{(x:A)} B(x)$。

在这里我们也可以通过完全递归定义函数。如果我们有上述的 $A$ 和 $B$，以及 $C : (\mathsf{W}_{(x:A)} B(x)) \to \mathcal{U}_m$，那么我们可以引入一个定义常量 $f : \prod_{(z:\mathsf{W}_{(x:A)} B(x))} C(z)$，当我们有

$$d : \prod_{(a:A)} \prod_{(u:B(a) \to \mathsf{W}_{(x:A)} B(x))} ((\prod_{(y:B(a))} C(u(y))) \to C(\mathsf{sup}(a, u)))$$

时，定义方程为

$$f(\mathsf{sup}(a, u)) :\equiv d(a, u, f \circ u)$$

### A.1.8 等同类型 (Identity Types)

我们引入原始常量 $c_=$ 和 $c_{\mathsf{refl}}$。当 $a : A$ 被理解时，我们将 $a =_A b$ 写作 $c_=(A, a, b)$，将 $\mathsf{refl}_a$ 写作 $c_{\mathsf{refl}}(A, a)$：

- 如果 $A : \mathcal{U}_n$，$a : A$ 和 $b : A$，那么 $a =_A b : \mathcal{U}_n$。
- 如果 $a : A$，那么 $\mathsf{refl}_a : a =_A a$。

给定 $a : A$，如果 $y : A, z : a =_A y \vdash C : \mathcal{U}_m$ 和 $\vdash d : C[a, \mathsf{refl}_a / y, z]$，那么我们可以引入一个定义常量

$$f : \prod_{(y:A)} \prod_{(z:a =_A y)} C$$

其定义方程为

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

## A.2 第二次表述 (The Second Presentation)

在本节中，有三种类型的判断：

$$\Gamma \ \mathsf{ctx} \qquad\qquad \Gamma \vdash a : A \qquad\qquad \Gamma \vdash a \equiv a' : A$$

我们通过提供推理规则来指定它们。一个典型的**推理规则** (inference rule)形式如下：

$$\frac{\mathcal{J}_1 \qquad \cdots \qquad \mathcal{J}_k}{\mathcal{J}}\ \text{Name}$$

它表示我们可以得出**结论** (conclusion)$\mathcal{J}$，前提是我们已经得出了**前提** (hypotheses)$\mathcal{J}_1, \ldots, \mathcal{J}_k$。（注意，由于这些是判断而不是类型，它们不是内部于类型论的假设，正如 §1.1中提到的；它们是推理系统中的假设，即元理论中的假设。）在右边我们写下规则的 Name，并且在应用该规则之前可能需要检查一些额外的附加条件。
判断的**推导** (derivation)是一棵由这种推理规则构建的树，树的根节点是判断。例如，使用下面给出的规则，以下是 $\cdot \vdash \lambda x.\, x : \mathbf{1} \to \mathbf{1}$ 的推导过程。

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\cdot\ \mathsf{ctx}}\ \textsf{ctx-emp}}{\vdash \mathbf{1} : \mathcal{U}_0}\ \mathbf{1}\text{-form}}{x{:}\mathbf{1}\ \mathsf{ctx}}\ \textsf{ctx-ext}}{x{:}\mathbf{1} \vdash x : \mathbf{1}}\ \textsf{Vble}}{\cdot \vdash \lambda x.\, x : \mathbf{1} \to \mathbf{1}}\ \Pi\text{-intro}$$

### A.2.1 上下文 (Contexts)

上下文是一个列表：

$$x_1{:}A_1, x_2{:}A_2, \ldots, x_n{:}A_n$$

它表示不同的变量$x_1, \ldots, x_n$ 分别假设具有类型 $A_1, \ldots, A_n$。该列表可以为空。我们用字母 $\Gamma$ 和 $\Delta$ 来缩写上下文，并且可以通过将它们并置来形成更大的上下文。
判断 $\Gamma\ \mathsf{ctx}$ 正式表示 $\Gamma$ 是一个良构上下文，并且受以下推理规则的约束：

$$\frac{}{\cdot\ \mathsf{ctx}}\ \textsf{ctx-emp} \qquad\qquad \frac{x_1{:}A_1, \ldots, x_{n-1}{:}A_{n-1} \vdash A_n : \mathcal{U}_i}{(x_1{:}A_1, \ldots, x_n{:}A_n)\ \mathsf{ctx}}\ \textsf{ctx-ext}$$

第二个规则的附加条件是：变量 $x_n$ 必须与变量 $x_1, \ldots, x_{n-1}$ 不同。注意，ctx-ext 的前提和结论是不同形式的判断：前提表示在变量 $x_1, \ldots, x_{n-1}$ 的上下文中，表达式 $A_n$ 具有类型 $\mathcal{U}_i$；而结论表示扩展后的上下文 $(x_1{:}A_1, \ldots, x_n{:}A_n)$ 是良构的。
这是系统的元理论性质，即如果可以推导出任何形式为 $\Gamma \vdash a : A$ 或 $\Gamma \vdash a \equiv a' : A$ 的判断，那么也可以推导出上下文 $\Gamma$ 是良构的判断 $\Gamma\ \mathsf{ctx}$。所有规则的前提都被选择为包含足够的良构性假设，以使该性质可证明，但没有更多的假设。例如，不需要在 ctx-ext 中假设 $(x_1{:}A_1, \ldots, x_{n-1}{:}A_{n-1})$ 是良构的，因为这将从其前提的可推导性中得出；但在下一节中的 Vble 规则中必须假设上下文是良构的。这种选择只是精确制定类型论的许多可能方法之一，但详细研究这些问题超出了本附录的范围。

### A.2.2 结构规则 (Structural Rules)

上下文中持有假设的事实通过以下规则表达，该规则表示我们可以推导出上下文中列出的那些类型判断：

$$\frac{(x_1{:}A_1, \ldots, x_n{:}A_n)\ \mathsf{ctx}}{x_1{:}A_1, \ldots, x_n{:}A_n \vdash x_i : A_i}\ \textsf{Vble}$$

与 ctx-ext 一样，规则 Vble 的前提和结论是不同形式的判断，只不过现在它们是反过来的：我们从一个良构的上下文开始，并推导出一个类型判断。

以下重要原则，称为**替换** (substitution)和**弱化** (weakening)，不需要显式假设。相反，可以通过对所有可能推导的结构进行归纳证明，每当这些规则的前提是可推导的，其结论也是可推导的。[2] 对于类型判断，这些原则表现为：

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x{:}A, \Delta \vdash b : B}{\Gamma, \Delta[a/x] \vdash b[a/x] : B[a/x]}\ \mathsf{Subst}_1 \qquad \frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, \Delta \vdash b : B}{\Gamma, x{:}A, \Delta \vdash b : B}\ \mathsf{Wkg}_1$$

对于判断等同性，它们表现为：

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x{:}A, \Delta \vdash b \equiv c : B}{\Gamma, \Delta[a/x] \vdash b[a/x] \equiv c[a/x] : B[a/x]}\ \mathsf{Subst}_2 \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma, x{:}A, \Delta \vdash c : C}{\Gamma, \Delta[a/x] \vdash c[a/x] \equiv c[b/x] : C[a/x]}\ \mathsf{Subst}_3$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, \Delta \vdash b \equiv c : B}{\Gamma, x{:}A, \Delta \vdash b \equiv c : B}\ \mathsf{Wkg}_2$$

除了为每个类型构造器给出的判断等同性规则外，我们还假设判断等同性是一个由类型支持的等价关系。

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A}{\Gamma \vdash b \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash b \equiv c : A}{\Gamma \vdash a \equiv c : A}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a : B} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a \equiv b : B}$$

最后，我们假设判断等同性是一个由类型支持的同余关系，即每个类型和术语构造器在其所有参数中保持判断等同性。例如，与 Π-，我们假设规则

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash B : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash b \equiv b' : B}{\Gamma \vdash \lambda x.\, b \equiv \lambda x.\, b' : \prod_{(x:A)} B}\ \Pi\text{-intro-eq}$$

完成依赖函数类型的情况，另外两条相似的规则为 Π-form-eq 和 Π-elim-eq。这些局部原则（在每个类型上）一起蕴含了上述的全局同余原则 $\mathsf{Subst}_2$ 和 $\mathsf{Subst}_3$。为了简洁起见，我们将省略这些局部规则。

### A.2.3 **类型宇宙** (Type Universes)

我们假设一个无限层级的类型宇宙：

$$\mathcal{U}_0, \quad \mathcal{U}_1, \quad \mathcal{U}_2, \quad \ldots$$

每个宇宙都包含在下一个宇宙中，$\mathcal{U}_i$ 中的任何类型也在 $\mathcal{U}_{i+1}$ 中：

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\ \mathcal{U}\text{-intro} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_{i+1}}\ \mathcal{U}\text{-cumul}$$

我们将设置类型论的规则，使得 $\Gamma \vdash a : A$ 蕴含 $\Gamma \vdash A : \mathcal{U}_i$ 对某个 $i$ 成立。换句话说，如果 $A$ 扮演类型的角色，那么它位于某个宇宙中。我们系统的另一个性质是 $\Gamma \vdash a \equiv b : A$ 蕴含 $\Gamma \vdash a : A$ 和 $\Gamma \vdash b : A$。

[2] 这些规则被称为**可容许的规则** (admissible)。

### A.2.4 **依赖函数类型** (Π-types)

在 §1.2中，我们引入了非依赖函数 $A \to B$，以定义类型族作为函数 $\lambda(x:A).B : A \to \mathcal{U}_i$，进而产生依赖函数类型 $\prod_{(x:A)} B$。但通过显式上下文，我们可以将 $\lambda(x:A).B : A \to \mathcal{U}_i$ 替换为判断

$$x{:}A \vdash B : \mathcal{U}_i$$

因此，我们可以直接定义依赖函数，而不需要参考非依赖函数。这样我们遵循了每个类型构造器都应独立于其他类型构造器引入的总体原则。实际上，从现在起，每个类型构造器将通过以下方式系统地引入：

- **构造规则** (formation rule)，说明何时可以应用类型构造器；
- 一些**引入规则** (introduction rules)，说明如何构造该类型的元素；
- **消去规则** (elimination rules)，或归纳原理，说明如何使用该类型的元素；
- **计算规则** (computation rules)，这些是判断等同性，解释当对引入规则的结果应用消去规则时会发生什么；
- 可选的**唯一性原则** (uniqueness principles)，这些是判断等同性，解释如何通过对该类型的元素应用消去规则来唯一地确定它们。

（另见 Remark 1.5.1。）
对于依赖函数类型，这些规则为：

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash B : \mathcal{U}_i}{\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}_i}\ \Pi\text{-form} \qquad \frac{\Gamma, x{:}A \vdash b : B}{\Gamma \vdash \lambda(x:A).b : \prod_{(x:A)} B}\ \Pi\text{-intro}$$

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a/x]}\ \Pi\text{-elim} \qquad \frac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash (\lambda(x:A).b)(a) \equiv b[a/x] : B[a/x]}\ \Pi\text{-comp}$$

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B}{\Gamma \vdash f \equiv (\lambda x.f(x)) : \prod_{(x:A)} B}\ \Pi\text{-uniq}$$

表达式 $\lambda(x:A).b$ 在 $b$ 中绑定 $x$ 的自由出现，同样 $\prod_{(x:A)} B$ 也在 $B$ 中绑定 $x$。
当 $x$ 在 $B$ 中不自由出现，从而 $B$ 不依赖于 $A$ 时，我们得到一个特例，即普通函数类型 $A \to B :\equiv \prod_{(x:A)} B$。我们将其视为 $\to$ 的**定义**。
我们可以将表达式 $\lambda(x:A).b$ 缩写为 $\lambda x.b$，理解为在类型检查之前，应该适当地填入省略的类型 $A$。

### A.2.5 **依赖对类型** (Σ-types)

在 §1.6中，我们需要 $\to$ 和 $\prod$ 类型，以定义 $\sum$ 的引入和消去规则；与 $\prod$ 一样，上下文允许我们独立地陈述 $\sum$ 的规则。回顾一下，对于正类型如 $\Sigma$，其消去规则称为**归纳** (induction)，用 $\mathsf{ind}$ 表示。

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash B : \mathcal{U}_i}{\Gamma \vdash \sum_{(x:A)} B : \mathcal{U}_i}\ \Sigma\text{-form} \qquad \frac{\Gamma, x{:}A \vdash B : \mathcal{U}_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a,b) : \sum_{(x:A)} B}\ \Sigma\text{-intro}$$

$$\frac{\Gamma, z{:}\sum_{(x:A)} B \vdash C : \mathcal{U}_i \qquad \Gamma, x{:}A, y{:}B \vdash g : C[(x,y)/z] \qquad \Gamma \vdash p : \sum_{(x:A)} B}{\Gamma \vdash \mathsf{ind}_{\sum_{(x:A)} B}(z.C, x.y.g, p) : C[p/z]}\ \Sigma\text{-elim}$$

$$\frac{\begin{array}{c}\Gamma, z{:}\sum_{(x:A)} B \vdash C : \mathcal{U}_i \qquad \Gamma, x{:}A, y{:}B \vdash g : C[(x,y)/z] \\ \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]\end{array}}{\Gamma \vdash \mathsf{ind}_{\sum_{(x:A)} B}(z.C, x.y.g, (a,b)) \equiv g[a,b/x,y] : C[(a,b)/z]}\ \Sigma\text{-comp}$$

表达式 $\sum_{(x:A)} B$ 在 $B$ 中绑定 $x$ 的自由出现。此外，由于 $\mathsf{ind}_{\sum_{(x:A)} B}$ 有一些超出 $\Gamma$ 的自由变量参与的参数，我们在 $C$ 中绑定 $z$，在 $g$ 中绑定 $x$ 和 $y$。这些绑定写作 $z.C$ 和 $x.y.g$，以指示被绑定变量的名称。特别地，我们将 $\mathsf{ind}_{\sum_{(x:A)} B}$ 视为一个原语，其中的两个参数包含绑定变量；这表面上类似于，但不同于，$\mathsf{ind}_{\sum_{(x:A)} B}$ 是一个以函数作为参数的函数。
当 $B$ 不包含 $x$ 的自由出现时，我们得到一个特例，即笛卡尔积 $A \times B :\equiv \sum_{(x:A)} B$。我们将其视为笛卡尔积的**定义**。
注意，我们没有为 $\Sigma$-类型假定一个判断的唯一性原则，即使我们可以这样做；参见??以获取对应命题唯一性原则的证明。

### A.2.6 余积类型 (Coproduct Types)

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash A + B : \mathcal{U}_i}\ +\text{-form}$$

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{inl}(a) : A + B}\ +\text{-intro}_1 \qquad \frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i \qquad \Gamma \vdash b : B}{\Gamma \vdash \mathsf{inr}(b) : A + B}\ +\text{-intro}_2$$

$$\frac{\begin{array}{c}\Gamma, z{:}(A+B) \vdash C : \mathcal{U}_i \\ \Gamma, x{:}A \vdash c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y{:}B \vdash d : C[\mathsf{inr}(y)/z] \\ \Gamma \vdash e : A + B\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}(z.C, x.c, y.d, e) : C[e/z]}\ +\text{-elim}$$

$$\frac{\begin{array}{c}\Gamma, z{:}(A+B) \vdash C : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y{:}B \vdash d : C[\mathsf{inr}(y)/z] \\ \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}(z.C, x.c, y.d, \mathsf{inl}(a)) \equiv c[a/x] : C[\mathsf{inl}(a)/z]}\ +\text{-comp}_1$$

$$\frac{\begin{array}{c}\Gamma, z{:}(A+B) \vdash C : \mathcal{U}_i \qquad \Gamma, x{:}A \vdash c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y{:}B \vdash d : C[\mathsf{inr}(y)/z] \\ \Gamma \vdash b : B\end{array}}{\Gamma \vdash \mathsf{ind}_{A+B}(z.C, x.c, y.d, \mathsf{inr}(b)) \equiv d[b/y] : C[\mathsf{inr}(b)/z]}\ +\text{-comp}_2$$

在 $\mathsf{ind}_{A+B}$ 中，$z$ 在 $C$ 中绑定，$x$ 在 $c$ 中绑定，$y$ 在 $d$ 中绑定。

### A.2.7 空类型 $\mathbf{0}$ (The Empty Type $\mathbf{0}$)

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbf{0} : \mathcal{U}_i}\ \mathbf{0}\text{-form} \qquad \frac{\Gamma, x{:}\mathbf{0} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash a : \mathbf{0}}{\Gamma \vdash \mathsf{ind}_{\mathbf{0}}(x.C, a) : C[a/x]}\ \mathbf{0}\text{-elim}$$

在 $\mathsf{ind}_{\mathbf{0}}$ 中，$x$ 在 $C$ 中绑定。空类型没有引入规则和计算规则。

### A.2.8 单位类型 $\mathbf{1}$ (The Unit Type $\mathbf{1}$)

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbf{1} : \mathcal{U}_i}\ \mathbf{1}\text{-form} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \star : \mathbf{1}}\ \mathbf{1}\text{-intro} \qquad \frac{\Gamma, x{:}\mathbf{1} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c : C[\star/x] \qquad \Gamma \vdash a : \mathbf{1}}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}(x.C, c, a) : C[a/x]}\ \mathbf{1}\text{-elim}$$

$$\frac{\Gamma, x{:}\mathbf{1} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c : C[\star/x]}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}(x.C, c, \star) \equiv c : C[\star/x]}\ \mathbf{1}\text{-comp}$$

在 $\mathsf{ind}_{\mathbf{1}}$ 中，变量 $x$ 在 $C$ 中绑定。
注意，我们没有为单位类型假设判断的唯一性原则；参见 §1.5以获取对应命题唯一性陈述的证明。

### A.2.9 **自然数类型** (The Natural Number Type)

我们给出自然数的规则，遵循 §1.9。

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbb{N} : \mathcal{U}_i}\ \mathbb{N}\text{-form} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash 0 : \mathbb{N}}\ \mathbb{N}\text{-intro}_1 \qquad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{succ}(n) : \mathbb{N}}\ \mathbb{N}\text{-intro}_2$$

$$\frac{\Gamma, x{:}\mathbb{N} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c_0 : C[0/x] \qquad \Gamma, x{:}\mathbb{N}, y{:}C \vdash c_s : C[\mathsf{succ}(x)/x] \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, n) : C[n/x]}\ \mathbb{N}\text{-elim}$$

$$\frac{\Gamma, x{:}\mathbb{N} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c_0 : C[0/x] \qquad \Gamma, x{:}\mathbb{N}, y{:}C \vdash c_s : C[\mathsf{succ}(x)/x]}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, 0) \equiv c_0 : C[0/x]}\ \mathbb{N}\text{-comp}_1$$

$$\frac{\Gamma, x{:}\mathbb{N} \vdash C : \mathcal{U}_i \qquad \Gamma \vdash c_0 : C[0/x] \qquad \Gamma, x{:}\mathbb{N}, y{:}C \vdash c_s : C[\mathsf{succ}(x)/x] \qquad \Gamma \vdash n : \mathbb{N}}{\begin{array}{c}\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, \mathsf{succ}(n)) \\ \equiv c_s[n, \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, n)/x, y] : C[\mathsf{succ}(n)/x]\end{array}}\ \mathbb{N}\text{-comp}_2$$

在 $\mathsf{ind}_{\mathbb{N}}$ 中，$x$ 在 $C$ 中绑定，$x$ 和 $y$ 在 $c_s$ 中绑定。
其他归纳定义的类型遵循相同的通用方案。

### A.2.10 **恒等类型** (Identity Types)

这里的表述对应于 §1.12中恒等类型的（非基）路径归纳原理。

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash a =_A b : \mathcal{U}_i}\ =\text{-form} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : a =_A a}\ =\text{-intro}$$

$$\frac{\begin{array}{c}\Gamma, x{:}A, y{:}A, p{:}x =_A y \vdash C : \mathcal{U}_i \\ \Gamma, z{:}A \vdash c : C[z, z, \mathsf{refl}_z/x, y, p]\end{array} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma \vdash p' : a =_A b}{\Gamma \vdash \mathsf{ind}_{=_A}(x.y.p.C, z.c, a, b, p') : C[a, b, p'/x, y, p]}\ =\text{-elim}$$

在 $\mathsf{ind}_{=_A}$ 中，$x$，$y$，和 $p$ 在 $C$ 中绑定，$z$ 在 $c$ 中绑定。

### A.2.11 **定义** (Definitions)

尽管我们列出的规则已经允许我们直接构造所需的一切，但我们仍然希望能够使用命名常量，如 $\mathsf{isequiv}$，作为便利手段。非正式地，我们可以将这些常量视为简写，但在形式化中情况稍微复杂一些。
例如，考虑函数组合，它将 $f : A \to B$ 和 $g : B \to C$ 映射到 $g \circ f : A \to C$。有些出乎意料的是，为了使这种形式化工作，$\circ$ 必须不仅接受 $f$ 和 $g$，还要接受它们的类型 $A$，$B$，$C$ 作为参数：

$$\circ :\equiv \lambda(A{:}\mathcal{U}_i).\lambda(B{:}\mathcal{U}_i).\lambda(C{:}\mathcal{U}_i).\lambda(g{:}B \to C).\lambda(f{:}A \to B).\lambda(x{:}A).g(f(x))$$

从实际角度看，我们不希望每次应用 $\circ$ 时都注释上 $A$，$B$ 和 $C$，因为它们通常可以很容易地从上下文中推断出来。我们希望简单地写 $g \circ f$。然后，严格来说，$g \circ f$ 并不是 $\lambda(x{:}A).g(f(x))$ 的简写，因为它涉及我们希望省略的附加**隐含参数** (implicit arguments)。
隐含参数的推断、典型模糊性（§1.3）、确保符号只定义一次等，统称为**精炼** (elaboration)。精炼必须在检查推导之前进行，因此通常不被作为核心类型论的一部分来介绍。然而，几乎不可能使用不进行精炼的类型论实现；参见 [?, ?]获取进一步讨论。

## A.3 **同伦类型论** (Homotopy Type Theory)

在本节中，我们陈述了同伦类型论的附加公理，这些公理将它与标准的 Martin-Löf 类型论区分开来：函数外延性、等价性公理（univalence axiom）以及高阶归纳类型。我们将以第二种形式的方式 (Appendix A.2) 来陈述它们，尽管第一种形式 (Appendix A.1) 也同样适用。

### A.3.1 函数外延性与等价性 (Function Extensionality and Univalence)

引入不涉及新语法或判断等式的公理（如函数外延性和等价性）有两种基本方式：要么添加一个原语常量来充当公理的实例，要么通过假设一个变量来占据公理的位置并证明所有依赖于该公理的定理，参见 §1.1。虽然这两者本质上是等价的，但我们选择前者，因为我们认为同伦类型论的公理是核心理论的重要组成部分。

Axiom 2.7.3 通过引入一个常量 $\mathsf{funext}$ 来形式化，该常量断言 $\mathsf{happly}$ 是一个等价：

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B \qquad \Gamma \vdash g : \prod_{(x:A)} B}{\Gamma \vdash \mathsf{funext}(f,g) : \mathsf{isequiv}(\mathsf{happly}_{f,g})}\ \Pi\text{-ext}$$

$\mathsf{happly}$ 和 $\mathsf{isequiv}$ 的定义可以分别在 (2.7.2)和 §4.5中找到。

??也以类似的方式形式化：

$$\frac{\Gamma \vdash A : \mathcal{U}_i \qquad \Gamma \vdash B : \mathcal{U}_i}{\Gamma \vdash \mathsf{univalence}(A,B) : \mathsf{isequiv}(\mathsf{idtoeqv}_{A,B})}\ \mathcal{U}_i\text{-univ}$$

$\mathsf{idtoeqv}$ 的定义可以在 (??)中找到。

### A.3.2 圆 (The Circle)

这里我们给出一个基本的高阶归纳类型的例子；其他类型遵循相同的一般方案，尽管有所展开。

注意，以下规则并未完全遵循 Appendix A.2中普通归纳类型的模式：这些规则涉及到映射的传递和函子性 (§2.2) 概念，并且第二个计算规则是命题等式，而不是判断等式。这些差异在 §6.2中进行了讨论。

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbb{S}^1 : \mathcal{U}_i}\ \mathbb{S}^1\text{-form} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathsf{base} : \mathbb{S}^1}\ \mathbb{S}^1\text{-intro}_1 \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}}\ \mathbb{S}^1\text{-intro}_2$$

$$\frac{\Gamma, x{:}\mathbb{S}^1 \vdash C : \mathcal{U}_i \qquad \Gamma \vdash b : C[\mathsf{base}/x] \qquad \Gamma \vdash \ell : b =^{C}_{\mathsf{loop}} b \qquad \Gamma \vdash p : \mathbb{S}^1}{\Gamma \vdash \mathsf{ind}_{\mathbb{S}^1}(x.C, b, \ell, p) : C[p/x]}\ \mathbb{S}^1\text{-elim}$$

$$\frac{\Gamma, x{:}\mathbb{S}^1 \vdash C : \mathcal{U}_i \qquad \Gamma \vdash b : C[\mathsf{base}/x] \qquad \Gamma \vdash \ell : b =^{C}_{\mathsf{loop}} b}{\Gamma \vdash \mathsf{ind}_{\mathbb{S}^1}(x.C, b, \ell, \mathsf{base}) \equiv b : C[\mathsf{base}/x]}\ \mathbb{S}^1\text{-comp}_1$$

$$\frac{\Gamma, x{:}\mathbb{S}^1 \vdash C : \mathcal{U}_i \qquad \Gamma \vdash b : C[\mathsf{base}/x] \qquad \Gamma \vdash \ell : b =^{C}_{\mathsf{loop}} b}{\Gamma \vdash \mathbb{S}^1\text{-}\mathsf{loopcomp} : \mathsf{apd}_{(\lambda y.\, \mathsf{ind}_{\mathbb{S}^1}(x.C,b,\ell,y))}(\mathsf{loop}) = \ell}\ \mathbb{S}^1\text{-comp}_2$$

在 $\mathsf{ind}_{\mathbb{S}^1}$ 中，$x$ 在 $C$ 中绑定。关于依赖路径 (dependent paths) 的记号 $b =^{C}_{\mathsf{loop}} b$ 在 §6.2中介绍。

## A.4 基本元理论 (Basic Metatheory)

本节讨论了 Appendix A.1中呈现的类型论的元理论性质，类似的结果也适用于 Appendix A.2。当我们添加 Appendix A.3中的特性时，确定这些性质是否仍然成立很快会引出一些开放问题，正如本节末尾所讨论的那样。

回想一下，Appendix A.1将类型论的项定义为无类型 $\lambda$-演算的扩展。$\lambda$-演算有其自身的计算概念，即计算规则：

$$(\lambda x.\, t)(u) :\equiv t[u/x]$$

该规则与定义常量的定义方程一起形成**重写规则** (rewriting rules)，这些规则确定了重写系统的简化步骤。这些步骤提供了计算的概念，因为每个规则都有一个自然的方向：通过计算函数的参数来简化 $(\lambda x.\, t)(u)$。

此外，该系统是合流的 (confluent)，即，如果 $a$ 在若干步骤中简化为 $a'$ 和 $a''$，则存在某个 $b$，使得 $a'$ 和 $a''$ 最终都简化为 $b$。因此，我们可以定义 $t \downarrow u$ 表示 $t$ 和 $u$ 简化为相同的项。
（在 Appendix A.2中的情况类似：尽管我们在那里将计算规则表示为无方向的等式 $\equiv$，但我们可以通过说将消除子应用于引入形式的结果简化为其等式而非相反来给出操作语义。）
使用类型论的标准技术，可以证明 Appendix A.1中的系统具有以下性质：

Theorem A.4.1. *如果 $A : \mathcal{U}$ 且 $A \downarrow A'$，则 $A' : \mathcal{U}$。如果 $t : A$ 且 $t \downarrow t'$，则 $t' : A$。*

我们说一个项是**可归约的** (normalizable)（分别是**强可归约的** (strongly normalizable)），如果从该项开始的某些（分别是所有）重写步骤序列终止。

Theorem A.4.2. *如果 $A : \mathcal{U}$，则 $A$ 是强可归约的。如果 $t : A$，则 $A$ 和 $t$ 都是强可归约的。*

我们说一个项处于**规范形式** (normal form)如果它不能被进一步简化，并且我们说一个项是**闭合的** (closed)如果其中没有自由变量出现在其中。闭合的规范类型必须是原语类型，即，形式为 $c(\vec{v})$ 的某个原语常量（其中闭合规范项列表 $\vec{v}$ 可能为空，此时可以省略，例如 $\mathbb{N}$）。事实上，我们可以明确描述所有规范形式：

Lemma A.4.3. *规范形式的项可以通过以下语法描述：*

$$v ::= k \mid \lambda x.\, v \mid c(\vec{v}) \mid f(\vec{v})$$
$$k ::= x \mid k(v) \mid f(\vec{v})(k)$$

*其中 $f(\vec{v})$ 表示已定义函数 $f$ 的部分应用。特别地，规范形式的类型是 $k$ 或 $c(\vec{v})$ 形式。*

Theorem A.4.4. *如果 $A$ 处于规范形式，则判断 $A : \mathcal{U}$ 是可判定的。如果 $A : \mathcal{U}$ 且 $t$ 处于规范形式，则判断 $t : A$ 是可判定的。*

(of the system in Appendix A.1) follows immediately: if we had $a : \mathbf{0}$ in the empty context, then by Theorems A.4.1 and A.4.2, $a$ simplifies to a normal term $a' : \mathbf{0}$. But by Lemma A.4.3 no such term exists.

Corollary A.4.5. The system in Appendix A.1 is logically consistent.

Similarly, we have the canonicity property that if $a : \mathbb{N}$ in the empty context, then $a$ simplifies to a normal term $\mathsf{succ}^k(0)$ for some numeral $k$.

Corollary A.4.6. The system in Appendix A.1 has the canonicity property.

Finally, if $a, A$ are in normal form, it is decidable whether $a : A$; in other words, because type-checking amounts to verifying the correctness of a proof, this means we can always "recognize a correct proof when we see one".

Corollary A.4.7. The property of being a proof in the system in Appendix A.1 is decidable.

然而，以上结果并不适用于扩展了同伦类型论特性的系统（即，通过 Appendix A.3扩展的上述系统），因为等价性公理和高阶归纳类型构造器的出现从未简化，打破了 Lemma A.4.3。能否通过简化这些常量的应用以恢复规范性仍然是一个开放问题。我们也没有描述所有允许的高阶归纳类型的模式，也不确定如何正确制定它们的规则（例如，高阶构造器的计算规则是否应该是判断等式）。
Martin-Löf 类型论扩展到等价性和高阶归纳类型的系统的一致性可以通过发明一种适当的归约过程来证明，但目前这些系统一致性的唯一证明是通过语义模型实现的——对于等价性，一个由 Voevodsky 提出的 Kan 复形模型 [?]，而对于高阶归纳类型，一个由 Lumsdaine 和 Shulman 提出的模型 [?]。
其他元理论问题以及我们当前结果的总结，在本书引言的"构造性"和"开放问题"部分中有更详细的讨论。

## Notes

引入原语常量的规则体系，以及消除和计算规则（定义常量），受到 Gentzen 自然演绎的启发。对存在量词消除规则的增强可能性在 [?] 中提到。对析取的公理增强出现在 [?] 中，对荒谬消除和等式类型的增强出现在 [?] 中。$W$-类型首次出现在 [?] 中。它们推广了 [?] 引入的树的概念。
自然数和序数的原始递归的广义形式出现在 [?] 中。这激发了 Gödel 系统 $T$ 的诞生，[?]，该系统由 [?] 进行了分析，他使用了 [?] 中的术语"定义等式"（definitional equality）来表示归约：如果两个项通

过一系列应用归约规则归约到一个共同的项，则它们被称为**判断等同的**（judgmentally equal）。这种术语也被 de Bruijn 在其关于 AUTOMATH 的呈现中使用 [?]。
我们的第二种形式呈现包含了相当标准的内涵 Martin-Löf 类型论，并且增加了一些在同伦类型论中必需的特性。与 [?] 的参考呈现相比，本书中的类型论有以下几个非关键性差异：

- Russell 风格的宇宙（universes），即 [?]中的概念；以及
- 对 $\Pi$ 类型的判断 $\eta$ 等式和函数外延性；

以及几个对于同伦类型论至关重要的特性：

- 等价性公理；以及
- 高阶归纳类型。

作为一种便利手段，本书主要通过**模式匹配**（pattern matching）定义函数。事实上，可以像在 Appendix A.1中那样形式化模式匹配的概念。然而，标准的类型理论呈现方式（在 Appendix A.2中采用）是为每个类型构造引入一个单一的**依赖消除子**（dependent eliminator），由此可以从该类型定义出函数。这种方法在语法和语义上都更容易形式化，因为它等同于类型构造的通用属性。这两种方法是等价的；参见 §1.10以获取更详细的讨论。

# 符号索引 (Index of symbols)

## 引言摘录:

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多个不同领域的元素。它基于最近发现的 同伦论 (Homotopy Theory) 和 类型论 (Type Theory) 之间的联系。它涉及的主题范围广泛，从球面同伦群、类型检查算法到弱 ∞-群 (Weak ∞-Groupoids) 的定义。

同伦类型论为数学基础带来了新的思想。一方面，有 Voevodsky 的微妙而美丽的 同值性公理 (Univalence Axiom)。同值性公理特别意味着同构结构可以被视为相同，这一原则虽然与传统基础的"官方"教义不符，却一直被数学家们在实际工作中愉快地使用。另一方面，我们有 高阶归纳类型 (Higher Inductive Types)，它们为同伦论中的一些基本空间和构造提供了直接的逻辑描述：球面、圆柱体、截断、局部化等。这些思想在经典的集合论基础中无法直接捕捉，但在同伦类型论中结合后，它们形成了一种全新的"同伦类型逻辑 (Logic of Homotopy Types)"。

这表明了一种数学基础的新概念，具有内在的同伦内容，一种数学对象的"不变量"概念——以及便捷的计算机实现，可以作为数学家工作的实用辅助工具。这就是 同值性基础 (Univalent Foundations) 计划。

本书旨在首次系统性地阐述同值性基础的基本内容，并收集这种新型推理风格的实例——但不要求读者掌握或学习任何形式逻辑，也不需要使用任何计算机证明助手。我们相信，同值性基础最终将成为集合论的可行替代方案，成为大多数数学家进行非形式化数学时的"隐式基础"。

在 HomotopyTypeTheory.org 获取本书的免费副本。