目录

方体类型论中的胞腔复形

邱皓晨
学号：16300180068
专业：数学与应用数学
指导老师：王国祯

摘要 同伦类型论是基于类型论和同伦理论之间的联系而建立的一种新理论，它基于构造主义的哲学思想，是一种潜在的数学基础性理论。方体类型论是同伦类型论的一个模型。本文试图运用方体类型论的思想，形式化同伦理论中关于映射同伦类和 CW 复形的概念。
关键字: 同伦类型论，方体类型论，映射同伦类，胞腔复形.

Abstract Cubical type theory is a newly-emerged model of homotopy type theory, which is a potential mathematical foundation based on constructivism. This paper aims to explore a variety of approaches to formalize notions of homotopy and homology theory, especially the CW complex and cellular homology.
Keywords: homotopy type theory, cubical type theory, homotopy classes of based maps, CW complex, Agda.

第一章　总论

第 1 节　导语

同伦类型论是基于类型论和同伦理论之间的联系而建立的一种新理论：对于已有的马丁-洛夫（Martin-Löf，瑞典逻辑学家）类型论来说，它引入了同伦论中的道路来刻画等价概念；对于数学来说，它的新特性是计算机能够理解并辅助证明，因此能用它来形式化数学概念；同时，同伦类型论基于构造主义的哲学思想，每一个对象和证明都需要具体构造出来，这使其成为一种潜在的数学基础性理论. 这方面的开创性工作来自《同伦类型论之书》[1]，这个项目用同伦类型论形式化了集合论、范畴论、同伦理论和实数理论的基础.

同伦类型论中定义的道路类型具有归纳公理：只要一个函数在常闭道上有定义，那么它就是整个道路类型上良定的函数. 这依据的是“所有道路都自由同伦于常闭道”的观点. 因此，同伦类型论实际上是通过有别于点集拓扑中定义开集的方式来刻画连续性. 进一步地，同伦类型论中的高维归纳类型将拓扑空间表示为含有非平凡道路的类型，它从整体的角度形式化同伦概念，有希望让我们对同伦理论有更深的理解. 通过同伦类型论研究同伦论的研究路径被称为整体同伦论.

方体类型论是同伦类型论中的一个模型，自创立之后得到了较多关注. 它的特点有二：其一是将道路表示为单位区间到类型宇宙的函数，这更接近传统的拓扑学. 在点集拓扑中，连续性的载体是开集，在同伦类型论中是道路的自由同伦，而在方体类型论中则是“沿着一条道路移动时能获得的连续性质”.“所有道路都自由同伦于常闭道”的观点，虽然相当简洁优美，却无法表现定端同伦. 因此，同伦类型论在定义具有非平凡道路的高维归纳类型（例如圆周）时，必须辅以相应的归纳公理，指明它上面的函数应该是怎么样的，在这个公理中蕴涵非平凡道路的性质. 而在方体类型论中，由于道路是由一个个对象组成的，避免了同伦类型论中道路和它所处的类型中的对象之间的割裂，所以它定义高维归纳类型时只需要给出高维道路的构造即可.

其二，方体类型论使用了范畴论中方体集的概念来描述类型宇宙的结构，赋予了道路类型更丰富的语义内涵. 除了方体类型论之外，单形类型论也是同伦类型论的另一个模型，它们的区别就是一个用正方形，一个用三角形：正方形虽然因为某些性质不好，在同调理论中较少使用，但在同伦理论中却大放异彩. 所谓“沿着一条道路移动时能获得的连续性质”，能很方便地用高维方体来构造. 因此，我们认为方体类型论更适合于整体同伦论的研究.

本选题试图通过尝试正在发展中的类型论的诸多未确定的手段，形式化同伦和同调理论中的一些数学对象并试图形式证明一些相关定理. 首先，本文将介绍同伦类型论和方体类型论的思想、基本概念、语法；其次，我们用方体类型论的模型形式化球面的自映射同伦类，并证明它是一个同构于整数的群；接着，我们利用这个结果定义映射度；最后，形式化 CW 复形及其胞腔同调. 本文所采用的模型导致的技术路线与同伦类型论中的形式化完全不同，代码也要简洁得多[1].

第 2 节　同伦类型论

2.1.　同伦类型论的基本设定

定义 1.1.　*若 A 是一个类型（type），那么 $a : A$ 声明了“a 是 A 中的一个对象”。*

需要注意的是，一个声明（在一些文献例如《同伦类型论之书》[1] 中，它们被称为判定，但我们认为，程序语言用词声明更适合这个语境）无关是与非，因为它们是被定义为成立的。因此，$a : A$ 与 $a \in A$ 有本质上的区别。例如，$\mathbf{2}$ 是布尔类型，那么

$$x : \mathbf{2}$$

的含义与程序语言中一个变量的声明

$$\text{boolean } x;$$

大体相同。

定义 1.2.　*如果 A 是一个类型, 那么 $a \equiv b : A$ 声明了 a 与 b 被定义为类型 A 中相同的对象.*

a 与 b 在定义的意义上相同, 意味着它们是指向同一个对象的“指针”.

在证明论中, 类型被视为命题, 而一个声明 $a : A$ 被视为命题 A 成立的假设. 如果 A 中的一个对象被具体的构造了出来, 那么它就是 A 的一个证明.

2.2.　类型的构造

在类型论中，我们遵循一个通用的模式来引入新类型。这个模式包含一系列规则, 它们类似于游戏规则 (另一方面, 公理类似于游戏的初始状态)。

首先, **类型签名** (或称形成规则，formation rule) 说明如何构造这样的新类型. 一个类型签名可以构造一系列的新类型，它们以其它已有类型或已有类型中已构造的对象为依据或指标。

[1] 本文详尽代码见https://hcqiu.github.io/.

其次, **构造子** (或称引入规则，introduction rule) 是一系列规则，说明如何构造这个类型中的元素. 它们同样可以接受其它已有类型或已构造的对象为参数。

定义 1.3. **函数类型.** 类型签名: 若 A 与 B 是类型, 则 $A \to B$ 表示**函数类型**, 它的定义域是 A, 值域是 B. 函数类型有一个构造子, 被称为 **λ-抽象**:

$$(\lambda x.\Phi) : A \to B, \tag{1.1}$$

在这里 $x : A$, $\Phi : B$ 是一个 x 的表达式.

例 1.4. 设 A 与 B 是两个类型, 并且 $y : B$, 那么

$$(\lambda x.y) : A \to B \tag{1.2}$$

是一个常值函数.

定义 1.5. $\mathcal{U}$ 表示一个**宇宙**, 它的元素是一些类型, 并且不包含自身（这是为了避免一些逻辑悖论）.

类型论中的宇宙构成了一个包含关系的线序序列 $\{\mathcal{U}_i\}$. 通常不指明下标，这时的宇宙级别需根据语境判断, 一般为 $\mathcal{U}_0$.

定义 1.6. 设 A 是一个类型, $\mathcal{U}$ 是一个宇宙, 则函数类型 $A \to \mathcal{U}$ 被称为**类型族**.

定义 1.7. **Π-类型（或称依赖函数类型）.** 类型签名: 设 $A : \mathcal{U}$, $B : A \to \mathcal{U}$, 我们有 $\prod_{x:A} B(x)$.

构造子:

$$(\lambda x.\Phi) : \prod_{x:A} B(x), \tag{1.3}$$

这里 $x : A$, $\Phi : B(x)$ 是一个 x 的表达式.

注 1.1 两个符号 Π 并列的表达式可以被视为值域是另一个 Π-类型的 Π-类型, 出现多个符号 Π 时照此递归处理. 另外, 容易证明 Π 的顺序可以在良定义（后续的类型的签名中的参数，已经出现在前面的语境中）的前提下任意调换，不改变其代表的类型（在等价意义下）.

定义 1.8. **积类型.** 类型签名: 设 $A, B : \mathcal{U}$, 我们有 $A \times B$.

构造子:

$$(a, b) : A \times B, \tag{1.4}$$

其中 $a : A$，$b : B$.

定义 1.9. **单位类型.** 构造子:

$$\star : \mathbf{1}. \tag{1.5}$$

定义 1.10.　**Σ-类型 (或称依赖乘积类型).** 类型签名: 设 $A:\mathcal{U}$, $B:A\to\mathcal{U}$, 则我们有 $\sum_{x:A}B(x)$.

构造子:

$$(a,b):\sum_{x:A}B(x), \tag{1.6}$$

其中 $a:A$, $b:B(x)$.

定义 1.11.　**余积类型.** 签名: 设 $A,B:\mathcal{U}$, 则我们有 $A+B$.

余积类型有两个构造子: 其一是左包含

$$\mathsf{inl}(a):A+B, \tag{1.7}$$

这里 $a:A$. 其二是右包含

$$\mathsf{inr}(b):A+B, \tag{1.8}$$

这里 $b:B$.

定义 1.12.　**空类型 $\mathbf{0}$.** 签名: 我们有 $\mathbf{0}$ (不依赖于语境).

构造子: 无.

定义 1.13.　**道路类型.** 签名: 设 $a,b:A$, 我们有 $a=_A b$, 在证明论中它代表命题“a 和 b 相等”, 在同伦论中它代表“空间 A 中连接 a 与 b 的道路空间”.

道路类型 $a=_A a$ 有一个构造子:

$$\mathsf{refl}(a),\ \text{其中}\,\mathsf{refl}:\prod_{x:A}(x=_A x), \tag{1.9}$$

在这里 $a:A$.

注 1.2　(1.9) 中的 refl 是一条规则, 它规定每一个点都有一条到自身的常闭道.

2.3.　归纳规则

归纳规则告诉我们该怎么使用一个类型中的对象. 在这里,“使用”的意思是一个以该类型为定义域的函数. 一般地, 一个类型中的对象不仅仅有构造子所构造的对象, 但是, 归纳规则会把定义在这些已知对象上的函数延拓到整个类型上 (这就是“归纳”的含义: 处理未知对象时, 归纳为已知对象的情况), 从而我们常常可以通过把值域取为道路类型, 而把原类型中所有可能的对象都和构造子所定义的对象等同起来.

我们来看乘积空间的归纳规则:

$$\mathsf{ind}_{A\times B}:\prod_{C:A\times B\to\mathcal{U}}\left(\prod_{(x:A)}\prod_{(y:B)}C\left((x,y)\right)\right)\to\prod_{z:A\times B}C(z) \tag{1.10}$$

它表明，只要在所有的有序对上定义一个依赖函数 $d:\left(\prod_{(x:A)}\prod_{(y:B)}C\left((x,y)\right)\right)$，那么我们就有了一个定义在 $A\times B$ 上的依赖函数.

对于道路类型和高维归纳类型，同伦类型论也有相应的归纳规则，但在方体类型论中都可以规避它们，因此不再赘述.

2.4. 同伦类型论中的证明

我们的目标是利用类型论来构建一个形式证明的体系，而同伦类型论很好地完成了这个任务. 如前所述，类型 A 被当作一个命题，$a:A$ 则是假设或证明，$a=_A b$ 是等价的证明，$\mathbf{1}$ 是真命题 (因其有一个证明 $\star$)，$\mathbf{0}$ 是假命题 (因为它的构造子是空的，所以它不可能有证明), $A\times B$ 是 $A\wedge B$(因为给出的 $A\times B$ 一个对象 x，等价于给出 A 和 B 的对象 $pr_1(x)$ 和 $pr_2(x)$), $A+B$ 是 $A\vee B$ (给出 $A+B$ 的一个对象，等价于给出 A 或 B 的一个对象)，$A\to B$ 则是“A 能推出 B”(给出一个 A 的证明以及一个由 A 到 B 的函数，显然我们就能获得一个 B 的证明)，而 $A\to\mathbf{0}$ 则是“$\neg A$”(因为 A 的一个证明能推出假命题的一个证明，所以 A 不可能有证明).

我们用一个例子说明这套语言的威力[2]：命题“如果 A，那么 (非 (非 A))”，可以被翻译为

$$A\to((A\to\mathbf{0})\to\mathbf{0}) \tag{1.11}$$

为了证明这个命题，我们给出上述函数类型的一个对象

$$f:A\to(A\to\mathbf{0})\to\mathbf{0} \tag{1.12}$$

$$f\ a\ g=g\ a \tag{1.13}$$

我们逐字解释一下它们的含义：首先，(1.12) 比 (1.11) 少了一个括号，但它们的含义是一样的，都表示一个含有两个自变量的函数类型；其次，a 是 A 的任意对象，g 是 $(A\to\mathbf{0})$ 的任意对象，它们输入 f 的过程也省略了括号. 我们定义它们输入 f 后的结果是 $g\ a$，这里再次省略括号. 根据定义，$g\ a$ 的结果是 $\mathbf{0}$ 中的对象. 这就完成了命题的证明.

第 3 节　方体类型论

在下面的介绍中，我们遵循严格的数理逻辑的语言，但采用 Agda 语言的符号，以与后续的代码兼容.

[2]这是《同伦类型论之书》[1] 的习题 1.12(ii).

3.1. 方体类型论中的道路

方体类型论的道路不再是一个抽象的类型，而是被定义为区间到类型的函数，这更接近拓扑学的设定，也简化了相关问题的处理方式. 但这种更具体的模型会否导致丧失一般性等其它问题，还需要更多实验.

为了解释清楚道路的具体情况，我们正式引入类型论的推演系统 (judgmental structure)[3]:

推理规则 (inference rule) 是在推演系统中被给定 (postulated) 的如下形式的表达式：

$$\frac{\mathcal{H}_1 \ \mathcal{H}_2 \ \cdots \ \mathcal{H}_n}{\mathcal{C}} \tag{1.14}$$

横线表明其下面的表达式可由其上面的推出.

从公式集 Γ 到公式 α 的一个推演是一个有穷的公式序列，每一个公式都能由前面的公式以及 Γ 中的公式按推理规则推出，最后一个公式是 α. 如果存在前述推演，则称 $\Gamma \vdash \alpha$.

一个语境 (context) 是如下形式的变量声明序列：

$$x_1 : A_1,\ x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}), \tag{1.15}$$

其中对于每一个被声明的变量 x_k 所在的类型 $A_k(x_1, \ldots, x_{k-1})$，都存在一个能够推出

$$x_1 : A_1,\ x_2 : A_2(x_1), \ldots, x_{k-1} : A_k(x_1, \ldots, x_{k-2}) \vdash A_k(x_1, \ldots, x_{k-1}) : \mathrm{Type}, \tag{1.16}$$

的推理规则，这里 $A_k(x_1, \ldots, x_{k-1})$ 表示 A_k 可能依赖于前面出现的 $k-1$ 个对象 (这是依赖类型论的特性)，例如

$$x_1 : \mathrm{Type},\ x_2 : x_1,\ x_3 : x_1,\ x_4 : x_2 =_{x_1} x_3. \tag{1.17}$$

其中 x_4 所在的类型之所以是良定义的，是因为推理规则 (1.21) (以及类型论通用的变量规则).

回到方体. 方体类型论中区间类型的形成规则和引入规则由以下推理规则给出：

$$\frac{}{\Gamma \vdash \mathbb{I} : \mathrm{Type}} \tag{1.18}$$

$$\frac{}{\Gamma \vdash i0, i1 : \mathbb{I}} \tag{1.19}$$

它们说明区间类型是不依赖任何语境的基础类型 (同第一节介绍的单位类型、空类型一样)，并且给出了区间中的两个对象，即起点和终点. 给出这两条规则后，区间类型在空语境下也是良定义的，因此能作为变量声明序列的第一条 (故区间类型是闭类型、两个端点是闭项)、能被加入任何语境中 (故类型论的推演系统被扩张为包含区间及其端点).

[3]细节可参考https://hott.github.io/HoTT-2019/images/hott-intro-rijke.pdf.

为给出方体类型论的道路类型，向推演系统中添加的推理规则包括：

类型签名：

$$\frac{\Gamma,\ i : \mathrm{I} \vdash A(i) \quad \Gamma \vdash a : A\ i0 \quad \Gamma \vdash b : A\ i1}{\Gamma \vdash \mathrm{PathP}\ A\ a\ b : \mathrm{Type}} \tag{1.20}$$

它的意思是，当我们有一个以 I 为指标集的类型族 A，以及类型 $A\ i0$ 中的对象 a、$A\ i1$ 中的对象 b 后，我们就有一个从 a 到 b 的道路类型. 它里面的对象是跨越了不同类型的道路, 称为*异类道路*.

构造子：

$$\frac{\Gamma,\ i : \mathrm{I} \vdash A(i) \quad \Gamma,\ i : \mathrm{I} \vdash a(i) : A(i)}{\Gamma \vdash \lambda\ i\ \to\ a\ i : \mathrm{PathP}\ A\ (a\ i0)\ (a\ i1)} \tag{1.21}$$

指明了构造一条道路的方式.

使用规则 (elimination rule)：

$$\frac{\Gamma \vdash p : \mathrm{PathP}\ A\ a\ b \quad \Gamma \vdash i : \mathrm{I}}{\Gamma \vdash p\ i : A\ i} \tag{1.22}$$

它告诉我们如何使用一条道路：将时刻 i 代入一条异类道路，得到在此时刻该道路所在的类型中的一个对象.

端点规则：

$$\frac{\Gamma \vdash p : \mathrm{PathP}\ A\ a\ b}{\Gamma \vdash p\ i0 = a : A\ i0} \tag{1.23}$$

$$\frac{\Gamma \vdash p : \mathrm{PathP}\ A\ a\ b}{\Gamma \vdash p\ i1 = b : A\ i1} \tag{1.24}$$

它们是说，由于在道路类型的签名中已经指明了两端的位置，因此在相应的端点，这个类型中所有的道路一定处于相应的位置.

3.2. 方体的语义

为了明白类型论的符号语言背后的含义，我们需要明确它的语义内涵. 语义学 (semantics) 指的是用元逻辑来研究符号逻辑，在这里我们立足的元逻辑是范畴论.

定义 1.14.　*笛卡尔方体范畴 $\square$ 的对象是有限集，态射为 $\mathrm{Hom}_{\square}(I, J)$*

一个*笛卡尔方体集* (cartesian cubical set) 是一个反变函子 $\Gamma : \widehat{\square}$. 直观上，可以把 Γ 当成一个空间，$\Gamma(\{i_1, \ldots, i_n\})$ 当成从 n 维方体到 Γ 的连续映射 (称为奇异 n-方体) 的集合，其中每一个 i_k 是这个方体的一个坐标.

定义米田嵌入 (Yoneda embedding)

$$\mathbf{y} : \square \to \widehat{\square} \tag{1.25}$$

$$I \mapsto \mathrm{Hom}_{\square}(-, I) \tag{1.26}$$

方体类型论的区间类型的语义就是 $\mathbf{y}(\{i\})$，它被称为可表示 1-方体，记为 $\mathbb{I}$. 可表示方体集是方体集范畴的一个子范畴，它的一个重要性质是对笛卡儿积封闭，即 $\mathbb{I}^n \simeq \mathbf{y}(\{i_1, \ldots, i_n\})$. 因此由米田引理 (Yoneda lemma)，“Γ 的奇异 n-方体”和“反变函子 $\mathbb{I}^n$ 和 Γ 之间的自然变换”一一对应，这可从下面的交换图看出：

$$id \longmapsto n\text{-cube} \tag{1.27}$$

$$\begin{array}{ccc} \mathrm{Hom}_{\square}(\{i_1, \ldots, i_n\}, \{i_1, \ldots, i_n\}) & \longrightarrow & \Gamma(\{i_1, \ldots, i_n\}) \\ \downarrow & & \downarrow \\ \mathrm{Hom}_{\square}(I, \{i_1, \ldots, i_n\}) & \longrightarrow & \Gamma(I) \end{array}$$

也就是说，一般的方体集被可表示方体集到它的映射完全决定. 这说明上面的直观是合理的. 进一步地，这也说明类型论中的表达式

$$i_1 : \mathbb{I}, \ldots, i_n : \mathbb{I} \vdash A : \mathcal{U} \tag{1.28}$$

与

$$A : \mathbb{I}^n \to \mathcal{U} \tag{1.29}$$

的对应是合理的.

3.3. 道路的连续性质

所谓“沿着道路移动时的连续性质”，指的是纤维化的道路提升，在方体类型论中就是“对于一条异类道路，在已知起点的情况下，获得它的终点”. 但在道路类型中，这样的移动无法保证定端，于是科恩引进了 kan 复合的构造，在移动高维道路时固定它的一些面.

kan 复合的一般形式是

$$\begin{aligned} &hcomp\ (\lambda\ k \to \lambda\ \{(j_0 = \epsilon_0) \to a_0(k, j_1, \cdots, j_n), \cdots, (j_n = \epsilon_n) \to a_n(k, \cdots, j_{n-1})\}) \\ &\qquad b\ (j_0, \cdots, j_n) \end{aligned}$$

其中 $\epsilon_0, \cdots, \epsilon_n$ 为 $i0$ 或 $i1$，b 为 $k = 0$ 时的底部方体 (即移动的起始点)，必须在边缘上满足花括号内给定的要求.

例 1.15. 证明 refl 是道路复合的右幺元：

$$\begin{aligned} &\mathsf{rUnit} : \{A : \mathsf{Type}\}\ \{a\ ,\ b : A\}\ (p : \mathsf{Path}\ A\ a\ b) \to \\ &\qquad\qquad \mathsf{Path}\ (\mathsf{Path}\ A\ a\ b)\ (p \cdot \mathsf{refl})\ p \end{aligned}$$

```
rUnit p k j = hcomp (λ i → λ { (j = i0) → p i0 ;
                               (j = i1) → p (i1 ) ;
                               (k = i1) → p ( j) })
                    (p (j ))
```

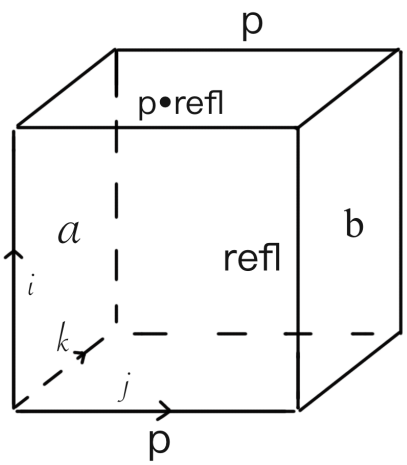

图 1.1: rUnit

第二章　方体类型论中的胞腔复形及胞腔同调

我们的目标是在方体类型论中形式化 CW 复形，以及计算它们的胞腔同调. 先看一个简单的例子：

```
{-# OPTIONS --cubical --safe #-}
open import Cubical.Foundations.Prelude
open import Cubical.Foundations.Structure
data RP² : Type where
  0-cell : RP²
  1-cell : 0-cell ≡ 0-cell
  2-cell : Path (0-cell ≡ 0-cell) (1-cell · 1-cell) refl
```

这里我们定义了一个类型 $\mathbb{R}P^2$，它的签名为 $\mathbb{R}P^2 : \mathrm{Type}$. 它有一个零维胞腔和一个一维胞腔，一维胞腔即为从零维胞腔到自身的道路. 它还有一个二维胞腔，它是道路的道路，连接 1-cell · 1-cell，以及零维胞腔处的常闭道. 直观上，二维胞腔是一个圆盘，边缘在一维胞腔上缠绕了两次.

这段代码中引用的两个包来自 Cubical Agda 标准库[1]，包含了前一章所述的基本构造. 本文所有代码仅依赖于这两个包.

第 1 节　球面的映射同伦类

1.1.　高维球面

定义高维球面有几种不同的方式. 最简洁的一种是将球面表示成一个 0 维胞腔，贴上一个 n 维胞腔. 代码如下：

```
data §¹ : Type where
  0-cell : §¹
  1-cell : 0-cell ≡ 0-cell

data §² : Type where
```

[1] 见https://github.com/agda/cubical.

```
  0-cell : §²
  2-cell : Path (0-cell ≡ 0-cell) refl refl
```

$\mathbb{S}^2$ 的二维胞腔是连接常闭道和常闭道的道路. 想象一个闭道，在 0 时刻为一个点，随后在运动中扩张成一个圆圈，最后在 1 时刻又收缩为一个点. 它划过的轨迹就形成了二维球面.

这样定义的好处是能够非常方便地按坐标顺序指明球面上的一个点：

```
rot§² : §² → §²
rot§² 0-cell = 0-cell
rot§² (2-cell i j) = 2-cell j i
```

这也说明，这样定义的球面是对称的，这是同伦类型论的定义中不具有的优势. 在同伦类型论中，“道路”和“道路的道路”之间有不可跨越的天堑.

一般地，n 维球面如下定义：

```
data § (n : ℕ) : Type  where
  0-cell : § n
  _-cell : typ ((Ω^ n) ( § n , 0-cell ))
```

其中，(𝕊 n , 0-cell) 表示带基点的 n 维球面；Ω^n 表示带基点类型的 n 维闭道空间 (loop space)；typ 表示带基点类型的类型，pt 表示带基点类型的基点. 它们的定义如下[2]：

```
data ℕ : Type  where
  zero : ℕ
  suc : ℕ → ℕ

{-# BUILTIN NATURAL ℕ #-}

Pointed : (l : Level) → Type (l-suc l)
Pointed l = TypeWithStr l (λ x → x)

pt : ∀ {l} (A· : Pointed l) → typ A·
pt = str

Pointed  = Pointed l-zero

{- Pointed functions -}
_→·_ : ∀{l l'} → (A : Pointed l) (B : Pointed l') → Type (l-max l l')
```

[2]这段代码来自 Cubical Agda 库.

```
_→·_ A B = Σ[ f ∈ (typ A → typ B) ] f (pt A) ≡ pt B

{- loop space of a pointed type -}
Ω : {l : Level} → Pointed l → Pointed l
Ω (_ , a) = ((a ≡ a) , refl)

{- n-fold loop space of a pointed type -}
Ω^_ : ∀ {l} → ℕ → Pointed l → Pointed l
(Ω^ 0) p = p
(Ω^ (suc n)) p = Ω ((Ω^ n) p)
```

这样的定义美则美矣，但因为在同伦和同调理论的构造中，含有“悬挂”(suspension)操作的序列经常出现，因此使用递归定义会更加方便:

```
data Sus (A : Type ) : Type  where
  north : Sus A
  south : Sus A
  merid : (a : A) → north ≡ south

data Bool : Type  where
  pos : Bool
  neg : Bool

_-primSphere : ℕ → Type
(0) -primSphere = Bool
(suc n) -primSphere = Sus ((n) -primSphere)
```

1.2. 高维同伦群的可交换性

在同伦类型论中，同伦群被定义为闭道空间的 0-截断 (n-截断的构造类似 E-M 空间，不断在一个类型上沿着非平凡 $n+1$ 维球面上贴上 $n+2$ 维胞腔，从而消去高维结构)，用这个定义可以证明二维及以上的同伦群是可交换的. 但这个证明用到了道路空间中常闭道的计算规则 (computational rule)，这在方体类型论中是没有的 (在方体类型论中，道路类型的归纳规则是被计算出来的，因此对应于常闭道的计算规则仅在同伦意义下成立)，这使得采取相似路径证明方体类型论中的结论变得非常复杂. 因此，我们采用更加拓扑的方式来证明这个结论.

高维同伦群可交换，本质上是因为更多的维度提供了更多的腾挪空间. 我们定义二维球面的旋转半圈的映射为：

```
2-sphere-exc : 2 -primSphere → 2 -primSphere
2-sphere-exc north = north
2-sphere-exc south = south
2-sphere-exc (merid north i) = (merid south i)
2-sphere-exc (merid south i) = (merid north i)
2-sphere-exc (merid (merid neg i) j) = (merid (merid pos (~ i)) j)
2-sphere-exc (merid (merid pos i) j) = (merid (merid neg (~ i)) j)
```

我们希望这个映射是和恒同映射同伦的. 然而，在方体类型论中无法直接将球面平滑地旋转 π，因此我们采取分步的方法：

先定义中间的状态，它是将其中一个半球拉伸为整个球面，另一个半球缩为 merid north 这一条线：

```
2-sphere-stretchTo : 2 -primSphere → 2 -primSphere
2-sphere-stretchTo north = north
2-sphere-stretchTo south = south
2-sphere-stretchTo (merid north i) = (merid north i)
2-sphere-stretchTo (merid south i) = (merid north i)
2-sphere-stretchTo (merid (merid neg i) j) = (merid (((merid neg) · (sym (merid pos))) i) j)
2-sphere-stretchTo (merid (merid pos i) j) = (merid north j)
```

然后，写出恒同映射到这个中间映射的同伦：

```
2-sphere-stretch-1 : (x : 2 -primSphere) → ( x ≡ 2-sphere-stretchTo x)
2-sphere-stretch-1 north = refl
2-sphere-stretch-1 south = refl
2-sphere-stretch-1 (merid north i) = refl
2-sphere-stretch-1 (merid south i) j = (merid (sym (merid pos) j) i)
2-sphere-stretch-1 (merid (merid neg i) j) k = merid (
                                    hcomp (λ h → λ {(i = i0) → north ;
                                                    (i = i1) → sym (merid pos) (k ∧ h) ;
                                                    (k = i0) → merid neg i })
                                          ( merid neg i))
                                                j
2-sphere-stretch-1 (merid (merid pos i) j) k = merid (sym (merid pos) ((~ i) ∨ k)) j
```

第二步是将之前被拉伸的半球压缩，之前被压缩的半球拉伸恢复为半球，但此时它们的位置已经交换了：

```
2-sphere-stretch-2 : (x : 2 -primSphere) → (2-sphere-stretchTo x ≡ 2-sphere-exc x)
2-sphere-stretch-2 north = refl
2-sphere-stretch-2 south = refl
2-sphere-stretch-2 (merid north i) j = merid (merid neg j) i
2-sphere-stretch-2 (merid south i) = refl
2-sphere-stretch-2 (merid (merid neg i) j) k = merid (
                                    hcomp (λ h → λ {(i = i0) → merid neg k ;
                                                    (i = i1) → merid pos (~ h) ;
                                                    (k = i1) → sym (merid pos) (i ∧ h) })
                                                    ( merid neg (i ∨ k) ))
                                                      j
2-sphere-stretch-2 (merid (merid pos i) j) k = merid (merid neg ((~ i) ∧ k)) j
```

接着我们将这两步组合起来，得到二维球面的保持基点的旋转：

```
2-sphere- rot : (x : 2 -primSphere) → (x ≡ 2-sphere-exc x)
2-sphere- rot north = refl
2-sphere- rot south = refl
2-sphere- rot (merid north i) s = merid ((refl · (λ j → (merid neg j))) s) i
2-sphere- rot (merid south i) s = merid (((λ j → (sym (merid pos) j)) · refl) s) i
2-sphere- rot (merid (merid neg i) j) s = merid (((λ k → (
                                    hcomp (λ h → λ {(i = i0) → north ;
                                                    (i = i1) → sym (merid pos) (k ∧ h) ;
                                                    (k = i0) → merid neg i })
                                                ( merid neg i))) ·
                                                    (λ k → (
                                    hcomp (λ h → λ {(i = i0) → merid neg k ;
                                                    (i = i1) → merid pos (~ h) ;
                                                    (k = i1) → sym (merid pos) (i ∧ h) })
                                                ( merid neg (i ∨ k) )))) s)
                                                      j
2-sphere- rot (merid (merid pos i) j) s = merid (((λ k →
                (sym (merid pos) ((~ i) ∨ k))) · (λ k → (merid neg ((~ i) ∧ k)))) s ) j
```

有了球面的旋转后，利用悬挂函子和闭道函子互为伴随 (adjoint) 的性质

$$[\Sigma A, B] \simeq [A, \Omega B] \tag{2.1}$$

我们可以证明二维同伦群是可交换的，再用自然数上的归纳得到高维同伦群是可交换的.

1.3. 悬挂和闭道函子互为伴随

在证明 (2.1) 时, 考虑到不同文献中的后续构造，我们实际上有三种等价的方式可以选择：悬挂、缩合悬挂 (reduced suspension)、同圆周的缩合积 (smash product). 我们尝试了第三种，但由于方体类型论中的缩合积仅是在同伦的意义下缩合 (即贴胞腔)，结果非常复杂：

```
data Smash (A B : Pointed ) : Type  where
  basel : Smash A B
  baser : Smash A B
  _∧_ : (x : typ A) → (y : typ B) → Smash A B
  gluel : (a : typ A) → a ∧ (pt B) ≡ basel
  gluer : (b : typ B) → (pt A) ∧ b ≡ baser

SmashPt : (A B : Pointed ) → Pointed
SmashPt A B = (Smash A B , basel)

basedCircle : Pointed
basedCircle = (§¹ , 0-cell)

sus→loop : {A , B : Pointed } → ((SmashPt A basedCircle) →· B) → (A →· Ω B)
sus→loop {A = A} {B = B} f = ( (λ (a : typ A) →
          ( sym ((cong (fst f) (gluel a)) · (snd f)) )
            ·· (λ i → ( (fst f) ( a ∧ (1-cell i))))
            ·· ((cong (fst f) (gluel a)) · (snd f)) ) ,
                    (( λ j → ( λ i →
                    hcomp (λ k → λ {(i = i0) →
                    hfill (λ h → λ { (k = i1) → (pt B) ;
                        (k = i0) → (cong (fst f) (gluer (0-cell)) h) })
                        (inS ( (sym ((cong (fst f) (gluel (pt A)))
                        · (snd f))) (~ k) )) j ;
                                    (i = i1) →
                    hfill (λ h → λ { (k = i1) → (pt B) ;
                        (k = i0) → (cong (fst f) (gluer (0-cell)) h) })
                        (inS ( ((cong (fst f) (gluel (pt A)))
```

```
                · (snd f)) k )) j })
    (cong (fst f) (gluer (1-cell i)) j)))
    · ( λ j → ( λ i →
    hcomp (λ k → λ {(i = i0) → (λ s → (
    hcomp (λ h → λ { (s = i1) → (pt B) ;
            ( s = i0) → (cong (fst f) (gluer (0-cell)) h) })
  ( ((cong (fst f) (gluel (pt A))) · (snd f)) s) )) (k ∧ (~ j)) ;
                        (i = i1) → (λ s → (
    hcomp (λ h → λ { (s = i1) → (pt B) ;
            ( s = i0) → (cong (fst f) (gluer (0-cell)) h) })
  ( ((cong (fst f) (gluel (pt A))) · (snd f)) s) )) (k ∧ (~ j)) ;
            (j = i1) → (fst f) baser })
  ((fst f) baser) )))  )
```

使用了六个立方体来构造两个函子间的自然变换 (natural transformation)，纯属自讨苦吃，于是我们直接采用最简单的第一种构造，即非缩合的悬挂：

```
susp : (A : Pointed ) → Pointed
susp A = (Sus (typ A) , north)

compSym : {A : Type } {a , b : A} (p : Path A a b) →
                                (Path (Path A a a) (p · (sym p)) refl)
compSym {a = a} p j i = hcomp (λ k → λ {(i = i0) → a ;
                                       (i = i1) → (sym p) (j ∨ k) ;
                                       (j = i1) → (a) })
                              (p (i ∧ (~ j)))

susp→loop : {A : Pointed } {B : Pointed } → (susp A →· B) → (A →· (Ω B))
susp→loop {A = A} {B = B} f = g , p
  where
    g : (typ A) → typ (Ω B)
    g a = ((sym (snd f)) · (cong (fst f) (merid a))) ·
          (sym ((sym (snd f)) · (cong (fst f) (merid (pt A)))))

    p : Path ((pt B) ≡ (pt B)) (g (pt A)) refl
    p = compSym {typ B} {pt B} {pt B}
                ((sym (snd f)) · (cong (fst f) (merid (pt A))))
```

```
loop→susp : {A : Pointed } {B : Pointed } → (A →· (Ω B)) → (susp A →· B)
loop→susp {A = A} {B = B} f = g , refl
  where
    g : typ (susp A) → typ B
    g north = pt B
    g south = pt B
    g (merid a i) = ((fst f) a) i
```

其中，susp 是 Sus 的带基点版本，以北极为基点；compSym 是连接 $p \cdot p^{-1}$ 与 refl 的道路；susp→loop 的定义中，由于保持基点的映射只是在同伦意义下 (即基点的像可连接到基点)，因此需要先用 snd f 连接；走到 (typ f) south 后，再通过 A 的基点的像的路径 (倒着) 走回 pt B (见Figure 2.1)；此时 pt A 的像自然是一个零伦的闭道，这是通过 compSym 见证的. 反过来，loop→susp 的定义是显然的. 由这样的直观易得这两个映射互为同伦逆.

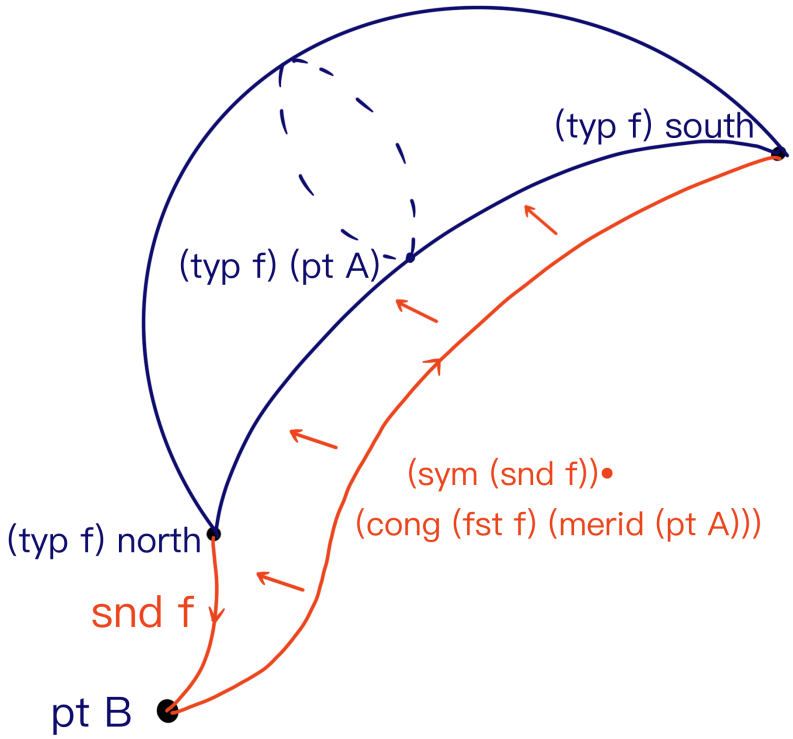

图 2.1: susp→loop

1.4. 映射同伦类构成一个群

由球面的悬挂结构，我们可以将其中段捏为一个点，从而赋予以其为定义域的保基点映射一个群结构.

```
data ∨₂susp (A : Type ) : Type  where
  north : ∨₂susp A
  south : ∨₂susp A
  middle : ∨₂susp A
  nmerid : (a : A) → north ≡ middle
  smerid : (a : A) → middle ≡ south

σ : {A : Type } → Sus A → ∨₂susp A
σ north = north
σ south = south
σ (merid a i) = ((nmerid a) · (smerid a)) i

_prim⋆_ : {A : Pointed } {B : Pointed } →
          (susp A →· B) → (susp A →· B) → ( ∨₂susp (typ A) → typ B)
_prim⋆_ {A = A} {B = B} f g north = pt B
_prim⋆_ {A = A} {B = B} f g middle = pt B
_prim⋆_ {A = A} {B = B} f g south = pt B
_prim⋆_ {A = A} {B = B} f g (nmerid a i) = ((sym (snd f)) · (cong (fst f) (merid a))
                                          · (cong (fst f) (sym (merid (pt A)))) · (snd f)) i
_prim⋆_ {A = A} {B = B} f g (smerid a i) = ((sym (snd g)) · (cong (fst g) (merid a))
                                          · (cong (fst g) (sym (merid (pt A)))) · (snd g)) i

_⋆_ : {A : Pointed } {B : Pointed } → (susp A →· B) → (susp A →· B) → (susp A →· B)
_⋆_ {A = A} {B = B} f g = ((λ (x : typ (susp A)) → ((_prim⋆_ {A} {B} f g) (σ x))) , refl)
```

在定义 prim⋆ 时，为了使得其在捏合点上良定义，我们使用了和上一节一样的“迂回”方法.

接着验证乘法的良定性：

```
⋆-wellDef : {A : Pointed } {B : Pointed } {a , b , c , d : (susp A →· B)}
          → (f : a ≡ b) → (g : c ≡ d) → ((_⋆_ {A} {B} a c) ≡ (_⋆_ {A} {B} b d))
⋆-wellDef {A = A} {B = B} f g i = (_⋆_ {A} {B} (f i) (g i))
```

随后我们定义了幺元和逆元：

```
id⋆ : {A : Pointed } {B : Pointed } → (susp A →· B)
id⋆ {B = B} = (λ x → (pt B)) , refl
```

```
inv⋆ : {A : Pointed } {B : Pointed } → (susp A →· B) → (susp A →· B)
inv⋆ {A = A} {B = B} f = g , ((sym (cong (fst f) (merid (pt A)))) · (snd f))
  where
  g : (typ (susp A))→ typ B
  g north = fst f south
  g south = fst f north
  g (merid a i) = (fst f) (merid a (~ i))
```

第 2 节　CW 复形及胞腔同调

2.1.　映射度的定义

有了上一节的准备工作，我们可以具体地定义映射度了，它是一个从球面 (非保持基点的) 自映射类型到整数类型的函数：

$$deg : (\mathbb{S}^n \to \mathbb{S}^n) \to \mathbb{Z} \tag{2.2}$$

先看 $n = 1$ 的情况，此时圆周之间的映射度即为缠绕数 (winding number). 我们利用方体类型论的可计算性单叶公理 (univalence axiom) 来计算缠绕数，详见 [2].

假设 $n = k$ 的情况已经定义好了，我们利用定理 (2.1) 以及《同伦类型论之书》[1] 的定理 8.6.4 (Cubical Agda 库中给出了该定理的方体类型论版本的证明代码)，完成 $n = k + 1$ 情况的定义.

2.2.　CW 复形

在方体类型论中，有限维 CW 复形及其胞腔同调的定义与 [3] 中同伦类型论版本的相关定义类似，只需要把推出 (pushout) 换成方体类型论中的推出即可.

致谢

感谢王国祯老师，吕志老师，傅吉祥老师在我学习代数拓扑过程中给予的指导和帮助. 感谢 github 平台为本文及其它形式化数学方面的成果托管代码.

参考文献

[1] Univalent Foundations Program, Homotopy Type Theory: Univalent Foundations of Mathematics, https://homotopytypetheory.org/book, Institute for Advanced Study，2013.

[2] Mörtberg, Anders and Pujet, Loïc, "Cubical synthetic homotopy theory", in *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP'20)*, 2020.

[3] Ulrik Buchholtz, Kuen-Bang Hou (Favonia), "Cellular Cohomology in Homotopy Type Theory", in *33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, 2018.