直觉类型论

Per Martin-Löf

*Giovanni Sambin*的一系列讲座笔记
于*1980*年6月在帕多瓦举行

目录

前言

这些讲座是在1980年6月在帕多瓦的Consiglio Nazionale delleRicerche的Laboratorio per Ricerche di Dinamica dei Sistemi e di Elettronica Biomedica举行的。我要感谢该实验室的Enrico Pagello博士给予我这个机会。观众由哲学家、数学家和计算机科学家组成。因此，我试图说一些可能对这三个类别中的每一个都有兴趣的事情。基本上是相同的讲座，尽管形式上有所改进和更高级，后来在同一年的会议上以构造性集合论和类型论的形式给出，该会议由Helmut Schwichtenberg教授组织在慕尼黑在1980年9月29日至10月3日期间，我对他的邀请表示感谢。

与在帕多瓦给出的讲座相比，慕尼黑讲座的主要改进之一是采用了系统化的高层（德语Stufe）符号，使我能够简单地书写

$$\Pi(A,\, B),\, \Sigma(A,\, B),\, \mathcal{W}(A,\, B),\, \lambda(b),$$
$$\mathsf{E}(c,\, d),\, \mathsf{D}(c,\, d,\, e),\, \mathsf{R}(c,\, d,\, e),\, \mathsf{T}(c,\, d)$$

而不是

$$(\Pi\, x \in A)\ B\ (x)\ ,\ (\Sigma\, x \in A)\ B\ (x)\ ,\ (\mathcal{W}\, x \in A)\ B\ (x)\ ,\ (\lambda x)\ b\ (x)\ ,\ \mathsf{E}\ (c,\ (x,\, y)\ d\ (x,\, y))\ ,\ \mathsf{D}\ (c,\ (x)\ d\ (x)\ ,\ (y)\ e\ (y))\ ,\ \mathsf{R}\ (c,\, d,\ (x,\, y)\ e\ (x,\, y))\ ,\ \mathsf{T}\ (c,\ (x,\, y,\, z)\ d\ (x,\, y,\, z))$$

分别。此外，使用更高级的变量和常量使得能够以与其他类型构造操作相同的模式来表达笛卡尔积的消除和相等规则。在他们的新表述中，这些规则如下

Π-消除

$$\frac{c \in \Pi(A,\, B) \qquad \begin{array}{c}(y(x) \in B(x)\ (x \in A)) \\ d(y) \in C(\lambda(y))\end{array}}{\mathsf{F}(c,\, d) \in C(c)}$$

和

Π-相等

$$\frac{\begin{array}{cc}(x \in A) & (y(x) \in B(x)\ (x \in A)) \\ b(x) \in B(x) & d(y) \in C(\lambda(y))\end{array}}{\mathsf{F}(\lambda(b),\, d) = d(b) \in C(\lambda(b))}$$

分别。 这里 y是一个绑定的函数变量， F是一个新的非规范（消除）运算符，通过它可以定义二元应用操作，即

$$\mathsf{Ap}(c,\, a) \;\equiv\; \mathsf{F}(c, (y)\, y(a)),$$

并且 $y(x) \in B(x)\,(x \in A)$ 是一个假设，本身是假设性的，已经被放在括号中表示它正在被解除。 一个新形式的程序 $\mathsf{F}(c, d)$ 有值 e，只要 c有值 $\lambda(b)$ 并且 $d(b)$ 有值 e。 当进行上述定义时，评估 $\mathsf{F}(c, d)$ 的规则简化为惰性评估规则 $\mathsf{Ap}(c, a)$。 选择 $C(z)$ 为 $B(a)$，因此与 z无关，并且选择 $d(y)$ 为 $y(a)$，新的消除规则简化为旧的规则，新的等式规则简化为旧的两个等式规则中的第一个。 此外，其中第二个规则，即规则

$$\frac{c \in \Pi(A,\, B)}{c = (\lambda x)\, \mathsf{Ap}(c,\, x) \in \Pi(A,\, B)}$$

可以通过相同的方式使用 I规则推导出

$$\frac{c \in \Sigma(A,\, B)}{c = (\mathsf{p}(c), \mathsf{q}(c)) \in \Sigma(A,\, B)}$$

在主要文本的第33页上通过示例推导得出。 相反，新的消除和等式规则可以通过对旧规则进行定义来推导出

$$\mathsf{F}(c,\, d) \;\equiv\; d((x)\, \mathsf{Ap}(c,\, x))。$$

所以，实际上它们是等价的。

我只剩下感谢乔瓦尼・桑宾，因为他自己建议，写作和打字这些笔记的工作量相当大，从而使得讲座对更广泛的受众可见。

斯德哥尔摩，1984年1月，

Per Martin-Löf

导言

数理逻辑和逻辑与数学之间的关系至少有三种不同的解释：

(1) 数理逻辑作为符号逻辑，或者使用数学符号的逻辑；

(2) 数理逻辑作为数学的基础（或哲学）；

(3) 数理逻辑作为使用数学方法研究的逻辑，作为数学的一个分支。

我们在这里主要关注数理逻辑的第二种意义。
我们所做的也是数理逻辑的第一种意义，但绝对不是第三种意义。

《数学原理》完成后，根据作者的说法，剩下的主要问题是证明还原性公理（或者现在我们所说的非预测性包容性公理）。 类型分层理论是预测性的，但它不能推导出甚至是分析的基本部分。 因此，还原性公理是出于实用的考虑而添加的，尽管无法提供令人满意的证明（解释）。 然后，类型分层的整个意义就消失了，所以它可能被废除。 那么剩下的就是简单类型理论。 它的正式证明（维特根斯坦，拉姆齐）基于将命题解释为真值，将命题函数（一个或多个变量的）解释为真值函数。 经典命题逻辑的定律显然是有效的，量词定律也是如此，只要量化限制在有限域上。 然而，在这种对命题和命题函数概念的解释下，似乎无法理解对无限域（如自然数域）的量化。 因此，我们在这里发展的是一种直观的类型论，它也是预测性的（或分层的）。 它摆脱了罗素的类型分层理论的缺陷，因为它具有允许我们形成任意给定集合族的笛卡尔积的操作，特别是从一个集合到另一个集合的所有函数的集合。

至少在两个领域中，我们的语言似乎比传统的基础语言更具优势。 首先，策梅洛-弗兰克尔集合论无法充分解决范畴论的基础问题，其中考虑了所有集合的范畴、所有群的范畴、从一个这样的范畴到另一个范畴的函子的范畴等等。这些问题通过直观类型论中对集合和范畴（在逻辑或哲学意义上，而不是范畴论的意义上）的区分来解决。 其次，现有的逻辑符号体系作为编程语言是不足够的，这解释了为什么计算机科学家们开发了自己的语言（如FORTRAN、ALGOL、LISP、PASCAL等）和证明规则系统（如Hoare、Dijkstra等）。

我们已经在其他地方展示了类型论相对于一阶谓词逻辑的额外丰富性，使其可用作编程语言。 我们已经在其他地方展示了类型论相对于一阶谓词逻辑的额外丰富性，使其可用作编程语言。

命题和判断

在这里，命题（德语：Satz）和断言或判断（德语：Urteil）之间的区别是至关重要的。我们通过逻辑运算组合的内容
$(\bot,\ \supset, \&,\ \vee,\ \forall,\ \exists)$ 并且我们认为是真实的命题。 当我们认为一个命题是真实的时候，我们做出了一个判断：

特别地，逻辑推理的前提和结论都是判断。
从弗雷格到《原理》之间，命题和判断的区别是清晰的。 这些概念后来被形式主义概念（在形式系统中的公式和定理）所取代。 与公式不同，命题不是归纳定义的。 可以说，它们形成了一个开放的概念。

在标准教科书的一阶逻辑中，我们可以区分三个相当独立的步骤：

(1) 项和公式的归纳定义,

(2) 公理和推理规则的规定,

(3) 语义解释.

公式和推导只有通过语义才能赋予意义，通常是在塔斯基的基础上并假设集合论。

我们在这里所做的是更接近普通数学实践。
我们将避免将形式和意义（内容）分开。 相反，我们将同时展示在数学证明中使用的某些判断和推理形式，并对其进行语义解释。 因此，我们明确了通常被默认接受的内容。 当将逻辑视为数学的任何其他分支时，如希尔伯特发起的元数学传统中，这些判断和推理只在所谓的对象语言中部分和形式化地表示，而在所谓的元语言中，它们被隐含地使用，就像在任何其他数学分支中一样。

---

[1] C. A. 霍尔，计算机编程的公理基础，ACM通信，
第12卷，1969年，第576-580和583页。

[2] E. W. 迪科斯特拉，编程学的一门学科，Prentice Hall，恩格尔伍德克利夫斯，新泽西州，1976年。

[3] P. 马丁-洛夫，构造性数学和计算机编程，逻辑，方法-
论和科学哲学VI，由L. J. 科恩，J. 洛斯，H. Pfeiffer和K.-
P. Podewski编辑，北荷兰，阿姆斯特丹，1982年，第153-175页。

我们的主要目标是建立一个能够以最佳方式表示非正式（数学）推理的形式规则系统。 在通常的自然推理风格中，给出的规则并不完全正式。例如，规则

$$\frac{\text{一个}}{\text{一个} \vee B}$$

它默认 A和 B是公式，然后才说当 A为真时我们可以推断 $A \vee B$为真。 如果我们要给出一个形式规则，我们必须明确地写出这一点

$$\frac{A\text{命题。} \qquad B\text{命题。} \qquad A\text{为真。}}{A \vee B\text{为真。}}$$

或者

$$\frac{A,\ B\text{命题。}}{\text{在我们的规则系统中，这总是明确的。}}$$

推理规则通过解释在已知前提条件下得出结论来进行证明。
因此，在证明推理规则之前，必须解释我们必须知道什么才能有权对前提条件和结论的各种形式进行判断。

一个推理规则的合理性是通过解释在已知前提条件下得出结论来证明的。因此，在证明一个推理规则之前，必须解释我们必须知道什么才能有权对前提条件和结论的各种形式进行判断。

我们使用四种判断形式：

(1) A是一个集合（缩写为 $A\ set$），

(2) A 和 B 是相等的集合（$A = B$),

(3) a是集合 A的元素（$a \in A$），

(4) a和 b是集合 A的相等元素（$a = b \in A$）。

（如果我们将 $\in$字面上解释为 $\epsilon\sigma\tau$， $\acute{\iota}$，那么我们可以分别写作 $A \in \mathrm{Set}$， $A = B \in \mathrm{Set}$，$a \in \mathrm{El}(A)$， $a = b \in \mathrm{El}(A)$。当然，可以使用任何语法变量；只是为了方便起见，我们使用小写字母表示元素，大写字母表示集合。 请注意，在普通集合论中， $a \in b$和 $a = b$是命题，而在这里它们是判断。 形式为 $A = B$的判断没有意义，除非我们已经知道 A和 B是集合。 同样，形式为 $a \in A$的判断假设 A是一个集合，形式为 $a = b \in A$的判断首先假设 A是一个集合，其次假设 a和 b是 A的元素。

每种判断形式都有几种不同的解读，如下表所示：

一个集合	$a \in A$	
A是一个集合	a是集合 A的元素	A是非空的
A是一个命题	a是一个证明（构造）命题 A	A是真实的
A是一个意图（期望）	a是一种实现的方法（实现）这个意图（期望） A	A是可实现的（可实现的）
A是一个问题（任务）	a是一种解决方法的方法问题（完成任务） A	A是可解的

第二个逻辑解释与下面的规则一起讨论。 第三个由Heyting[4]提出，第四个由Kolmogorov[5]提出。 最后一个非常接近编程。“a是一种方法. . . ”可以理解为“a是一个程序. . .”。 由于编程语言对程序 a有正式的符号表示，但对 A没有，我们用“. . .”来完成句子。 满足规范的 A”。 在Kolmogorov的解释中，问题一词指的是要做的事情，程序一词指的是如何做。第一和第二解释之间的类比在Brouwer-Heyting解释中是隐含的。 Curry和Feys[6]更加明确地提出了这一点，但仅适用于蕴含片段，并且Howard[7]将其扩展到直觉主义的一阶算术。 这是目前已知的唯一解释直觉主义逻辑的方式，使得选择公理成立。

为了区分判断的证明（通常以树状形式呈现）和命题的证明（在此与元素等同，因此在 $\in$的左边），我们将*construction*这个词保留给后者，并在可能引起混淆时使用它。

判断形式的解释

对于每一种判断形式，我们现在解释该形式的判断意味着什么。 我们可以通过回答以下三个问题之一来解释一个判断的意义，比如第一种形式的判断：

什么是集合?

为了有权利判断某个东西是一个集合，我们必须知道什么?

[4] A. Heyting, Die intuitionistische Grundlegung der Mathematik, *Erkenntnis*, Vol. 2, 1931, pp. 106–115.

[5] A. N. Kolmogorov, Zur Deutung der intuitionistischen Logik, *Mathematische Zeitschrift*, Vol. 35, 1932, pp. 58–65.

[6] H. B. Curry和R. Feys，组合逻辑，第1卷，北荷兰，阿姆斯特丹，1958年，页码312-315。

[7] W. A. Howard，构造的公式即类型概念，对H. *B. C*urry：组合逻辑、λ演算和形式主义的论文，学术出版社，伦敦，1980年，页码479-490。

形如“A是一个集合”的判断是什么意思?

第一个是本体论的（古希腊），第二个是认识论的（笛卡尔，康德，...），第三个是语义学的（现代）提出本质上相同问题的方式。乍一看，我们可以假设一个集合是通过规定其元素如何形成来定义的。当我们说自然数集合 $\mathbb{N}$的定义是通过给出规则来完成的时，我们就是这样做的：

$$0 \in \mathbb{N} \qquad \frac{a \in \mathbb{N}}{a' \in \mathbb{N}}$$

通过这种方式构建其元素。然而，这个定义的弱点是明显的：例如，尽管不能用给定的规则获得，但10^{10}显然是 $\mathbb{N}$的一个元素，因为我们知道我们可以将其变为某个 $\mathbb{N}$中的元素 a'。因此，我们必须区分那些具有可以直接看到它们是规则之一的结果的形式的元素，称之为规范元素，以及所有其他元素，我们将称之为非规范元素。但是，为了能够定义两个非规范元素何时相等，我们还必须规定两个相等的规范元素是如何形成的。所以：

（1）集合 A通过规定如何形成 A的规范元素以及如何形成两个相等的 A的规范元素来定义。

这是对形如 A是一个集合的判断意义的解释。
例如，对于上面的 $\mathbb{N}$的规则，我们必须添加

$$0 = 0 \in \mathbb{N} \quad \text{和} \quad \frac{a = b \in \mathbb{N}}{a' = b' \in \mathbb{N}}$$

再举一个例子， $A \times B$由以下规则定义

$$\frac{a \in A \qquad b \in B}{(a,\, b) \in A \times B}$$

它规定了如何形成规范元素，以及规则

$$\frac{a = c \in A \qquad b = d \in B}{(a,\, b) = (c,\, d) \in A \times B}$$

通过这些规则形成相等的规范元素。对于定义集合的规定没有限制，除了规范元素之间的相等必须始终被定义为是自反的、对称的和传递的。

现在假设我们知道 A和 B是集合，也就是说，我们知道如何构造 A和 B的规范元素和相等的规范元素。然后我们规定：

(2) 如果一个元素$a \in A$，那么两个集合 A和 B

$$\frac{\text{是相等的}}{a \in B} \quad (\text{也就是说，} \frac{a \in A}{a \in B} \text{和一个元素} \frac{a \in B}{\text{一个元素} a \in}$$

并且

$$\frac{\frac{\text{一个元素}a = \text{一个元}}{\text{一个元素}a}}{}$$

$=$一个元素 $b \in B$对于任意的规范元素 a, b

来说，这是一个判断 $A = B$的含义。

当我们解释一个集合 A的元素时，我们必须假设我们知道 A是一个集合，也就是说，特别是它的规范元素是如何构造的。
然后：

(3) 一个集合 A的元素 a是一个方法（或程序），当执行时，产生一个 A的规范元素作为结果。

这是一个形式为 $a \in A$的判断的意义。 请注意，这里我们假设方法的概念是原始的。 当前语言的计算规则（执行）是这样的，即一个集合 A的元素 a的计算在外层形式为 b的时候终止，并返回一个值 b，该值表明它是 A的一个规范元素（正常顺序或惰性求值）。 例如，计算 $2 + 2 \in \mathbb{N}$的结果是值 $(2 + 1)'$，这是 $\mathbb{N}$的一个规范元素，因为 $2 + 1 \in \mathbb{N}$。最后：

(4) 一个集合 A的任意两个元素 a和 b相等，当且仅当它们执行后返回相等的 A的规范元素。

这是一个形式为 $a = b \in A$的判断的意义。 这个定义是有意义的，因为一个集合的定义中包含了两个规范元素相等的含义。

例子。 如果$\mathbf{e}$，$\mathrm{f} \in A \times B$，则 e和 f是产生规范元素$(\mathbf{a}$，$b)$，$(\mathrm{c}$，$\mathrm{d}) \in A \times B$的方法，分别作为结果，并且 $e = f \in A \times B$如果$(\mathbf{a}$，$b) = (\mathrm{c}$，$\mathrm{d}) \in A \times B$，这反过来成立如果 $a = c \in A$和 $b = d \in B$。

命题

经典地，命题只是一个真值，即真和假的元素，其两个元素是真和假。 由于通过对无限域进行量化来证明形成命题的规则的困难，当命题被理解为真值时，直觉主义者拒绝了这种解释，并用以下方式替代：

命题通过规定命题的证明来定义，并且

如果一个命题有证明，即可以给出一个证明，则该命题为真[8].

[8]D. Prawitz，直观逻辑：一个哲学挑战逻辑与哲学，由G. H. von Wright编辑，Martinus Nijhoff，海牙，第1-10页.

因此，直观上，真实与可证明等同，尽管当然不等同于任何特定形式系统内的可导出性（由于哥德尔的不完全性定理）.

逻辑操作的含义解释与直观命题的概念相吻合，由标准表给出:

命题的证明	由...组成
$\bot$	–
A和 B	A的证明和B的证明
一个 $\vee B$	一个证明 A 或一个证明 B
$A \supset B$	一种方法，它接受任何一个证明 A为参数，并产生一个证明 B
$(\forall x)\, B(x)$	一种方法，它接受任意一个个体 a 为参数，并产生一个证明 $B(a)$
$(\exists x)\, B(x)$	一个个体 a 和一个证明 $B(a)$

第一行的解释是说没有任何东西可以被视为 $\bot$的证明。

上述表格可以更明确地表达为：

命题的证明	具有以下形式
$\bot$	–
A和 B	(a, b)，其中 a是一个证明 A， b是一个证明 B
一个 $\vee B$	i(a)，其中 a是一个证明 A，或者j(b)，其中 b是一个证明 B
$A \supset B$	$(\lambda x)\, b(x)$，其中 $b(a)$是一个证明 B，只要 a是一个证明 A
$(\forall x)\, B(x)$	$(\lambda x)\, b(x)$，其中 $b(a)$是 $B(a)$的证明，假设 a是一个个体
$(\exists x)\, B(x)$	(a, b)，其中 a是一个个体， b是 $B(a)$的证明

就目前而言，这个表格并不严格正确，因为它只显示了规范形式的证明。 任意的证明，类似于集合中的任意元素，是生成规范形式证明的方法。

如果我们认真对待一个命题是通过规定其规范证明的生成方式来定义的（如上表中的第二个表格），并接受一个集合是通过规定其规范元素的生成方式来定义的，那么很明显，将命题和集合（以及命题的证明和集合的元素的相关概念）分开只会导致不必要的重复。 相反，我们只是将它们等同起来，也就是将它们视为同一概念。 这就是直观类型论的公式即类型（命题即集合）解释基础。

等式规则

我们现在开始建立一套规则。 首先，我们给出以下相等性规则，这些规则可以很容易地通过它们根据定义成立的事实来解释：

自反性

$$\frac{a \in A}{a = a \in A} \qquad \frac{A\text{集合}}{A = A}$$

对称性

$$\frac{\text{一个元素} a = \text{一个元}}{b = a \in A} \qquad \frac{A = B}{B = A}$$

传递性

$$\frac{\text{一个元素} a = \text{一个元} \qquad b = c \in A}{a = c \in A} \qquad \frac{A = B \qquad B = C}{A = C}$$

例如，传递性的详细解释是： $a = b \in A$意味着a和 b分别产生规范元素 d和 e，并且 $d = e \in A$。类似地，如果 c产生 f，那么 $e = f \in A$。 由于我们假设规范元素具有传递性，我们得到 $d = f \in A$，这意味着 $a = c \in A$。 $A = B$的含义是

$$\frac{\frac{a \in A}{}}{a \in B}$$

和

$$\frac{\frac{\text{一个元素} a = \text{一个元}}{}}{a = b \in B}$$

对于 A和 B的规范元素 a和 b。 从 $B = C$的同样情况中，我们也得到

$$\frac{\frac{a \in A}{}}{a \in C}$$

和

$$\frac{\frac{\text{一个元素} a = \text{一个元}}{}}{\text{对于规范元素} a \text{和} b\text{，这就}}$$

是 $A = C$的含义。同样明显的方式， $A = B$的含义证明了以下规则：

集合的相等

$$\frac{a \in A \qquad A = B}{a \in B} \qquad \frac{\text{一个元素} a = \text{一个元} \qquad A = B}{a = b \in B}$$

假设判断和替换规则

四种基本判断形式被推广以表达假设判断，即在假设下进行的判断。 在本节中，我们处理一个这样的假设。 所以假设 A是一个集合。第一种判断形式被推广为假设形式(1) $B(x)\ set\ (x \in A)$

它表示在假设$x \in A$下， $B(\ x)$是一个集合，或者更好地说，$B(x)$是一个在 A上的集合族。 更传统的表示法是 $\{B_x\}_{x\in A}$或者$\{B_x : x \in A\}$。 判断形式 (1)的含义是，当a是 A的元素时， $B(\ a)$是一个集合，而且当a和c是 A的相等元素时， $B(\ a)$和 $B(\ c)$是相等的集合。 根据这个含义，我们立即可以看出以下替换规则是正确的：

替换

$$\frac{\begin{array}{cc} & (x \in A) \\ a \in A & B(x)\text{集合} \end{array}}{B(a)\text{集合}} \qquad \frac{\begin{array}{cc} & (x \in A) \\ a = c \in A & B(x)\text{集合} \end{array}}{B(a) = B(c)}$$

这个符号

$$\begin{array}{c} (x \in A) \\ B(x)\text{集合} \end{array}$$

只是回忆我们对 $B(x)$ 是一个集合的判断（有一个证明），在假设 $x \in A$下，这并不意味着我们必须在任何特定的形式系统中有一个推导（就像我们正在构建的这个系统一样）。 当一个假设 $x \in A$通过应用规则被解除时，我们将其写在括号内。

形如（2） $B(x) = D(x)\ (x \in A)$的假设判断的意义是，B(a)和D(a)是集合A上的相等集合，因此，B(x)和D(x)必须是A上的集合族。因此，规

意味着 $B(x)$和 $D(x)$是 A上的集合族。因此，规则

替换

$$\frac{\begin{array}{cc} & (x \in A) \\ a \in A & B(x) = D(x) \end{array}}{B(a) = D(a)}$$

是正确的。 我们现在可以推导出规则

$$\frac{\begin{array}{cc} & (x \in A) \\ a = c \in A & B(x) = D(x) \end{array}}{B(a) = D(c)}$$

从上述规则中得出。 事实上，根据 $a = c \in A$和 $B(x)$集合 $(x \in A)$，我们通过第二个替换规则得到$B(a) = B(c)$，并且根据第三个替换规则，由于 $c \in A$（这是隐含的），在 $a = c \in A$中， $B(c) = D(c)$。 因此，通过传递性，有 $B(a) = D(c)$。

形式为(3)的假设判断

$(x \in A)$时， $b(x) \in B(x)$

意味着我们知道 $b(a)$是集合 $B(a)$的一个元素，假设我们知道a是集合 A的一个元素，并且当a和c是 A的相等元素时， $b(a) = b(\ c) \in B(a)$。 换句话说， $b(x)$是一个具有域 A和范围 $B(x)$的外延函数，取决于参数 x。 然后，以下规则是合理的：

替换

$$\frac{a \in A \qquad \begin{array}{c}(x \in A)\\ b(x) \in B(x)\end{array}}{b(a) \in B(a)} \qquad \frac{a = c \in A \qquad \begin{array}{c}(x \in A)\\ b(x) \in B(x)\end{array}}{b(a) = b(c) \in B(a)}$$

最后，形式为的判断

(4) $b(x) = d(x) \in B(x)\ (x \in A)$

意味着对于集合 A的任意元素 a， $b(a)$和 $d(a)$是集合 $B(a)$的相等元素。我们有：

替换

$$\frac{a \in A \qquad \begin{array}{c}(x \in A)\\ b(x) = d(x) \in B(x)\end{array}}{b(a) = d(a) \in B(a)}$$

这是最后的替换规则。

具有多个假设和上下文的判断

我们现在可以进一步推广判断，包括具有任意数量 $n$of假设的假设性判断。 我们通过归纳来解释它们的含义，也就是假设我们理解具有$n - 1$个假设的判断的含义。所以假设我们知道

A_1是一个集合，

$A_2(x_1)$是一个在 A_1上的集合族，

$A_3(x_1, x_2)$是一个具有两个指标 $x_1 \in A_1$和 $x_2 \in A_2(x_1)$的集合族，

. . .

$A_n(x_1, \ldots, x_{n-1})$ 是一个具有 $n-1$ 个指标的集合族 $x_1 \in A_1, x_2 \in A_2(x_1), \ldots, x_{n-1} \in A_{n-1}(x_1, \ldots, x_{n-2})$。

然后形式为的判断

(1) $A(x_1, \ldots, x_n)$ *set* $(x_1 \in A_1, x_2 \in A_2(x_1), \ldots, x_n \in A_n(x_1, \ldots, x_{n-1}))$

意味着当$a_1 \in A_1, a_2 \in A_2(a_1), \ldots, a_n \in A_n(a_1, \ldots, a_{n-1})$时，$A(a_1, \ldots, a_n)$ 是一个集合，并且当 $a_1 = b_1 \in A_1, a_2 = b_2 \in A_2(a_1), \ldots, a_n = b_n \in A_n(a_1, \ldots, a_{n-1})$时，$A(a_1, \ldots, a_n) = A(b_1, \ldots, b_n)$，$\ldots, a_n = b_n \in A_n(a_1, \ldots, a_{n-1})$. 我们说 $A(x_1, \ldots, x_n)$ 是一个带有指标的集合族。 在形式为 (1) 的判断中，假设构成了我们所称的上下文，它在逻辑推理中起到了类似于公式集合 Γ, Δ（额外的公式）在Gentzen序列中出现的作用。还要注意，上下文的任何初始段总是一个上下文。 由于形式为 (1) 的假设判断的含义，我们可以将替换的前两个规则扩展到假设的情况，并且我们理解这些扩展是给定的。

现在我们清楚了如何解释剩余形式的假设判断的含义：

(2) $A(x_1, \ldots, x_n) = B(x_1, \ldots, x_n)$ $(x_1 \in A_1, \ldots, x_n \in A_n(x_1, \ldots, x_{n-1}))$ (具有 n指标的相等集族),

(3) $a(x_1, \ldots, x_n) \in A(x_1, \ldots, x_n)$ $(x_1 \in A_1, \ldots, x_n \in A_n(x_1, \ldots, x_{n-1}))$ (具有 n参数的函数),

(4) $a(x_1, \ldots, x_n) = b(x_1, \ldots, x_n) \in A(x_1, \ldots, x_n)$ $(x_1 \in A_1, \ldots, x_n \in A_n(x_1, \ldots, x_{n-1}))$ (具有 n个参数的相等函数),

并且我们假设相应的替换规则已经给出。

集合和范畴

一个范畴通过解释范畴的对象是什么以及何时两个这样的对象相等来定义。一个范畴不需要是一个集合，因为我们可以理解成为给定范畴的对象而无需详尽的形成规则。 例如，我们现在理解什么是集合以及何时两个集合相等，因此我们已经定义了集合的范畴（以及同样的道理，命题的范畴），但它不是一个集合。到目前为止，我们已经定义了几个范畴：

集合的范畴（或命题的范畴），

给定集合的元素的范畴（或命题的证明的范畴），

给定集合 A上的集合族 B（对于每个x）（其中$x \in A$），

函数的类别 $b(x) \in B(x)\,(x \in A)$，其中 A是一个集合， $B(x)$是一个集合 $(x \in A)$,

集合族的类别 $C(x, y)\,(x \in A,\, y \in B(x))$，其中 A是一个集合， $B(x)$是一个集合 $(x \in A)$，

函数的类别 $c(x, y) \in C(x, y)\,(x \in A,\, y \in B(x))$，其中 A是一个集合， $B(x)\,(x \in A)$ 和 $C(x, y)\,(x \in A,\, y \in B(x))$ 是集合族,

等等。

除了这些，还有更高阶的类别，比如将两个集合映射到另一个集合的二元函数的类别。函数 $\times$，将两个集合 A和 B映射到它们的笛卡尔积 $A \times B$，是该类别中的一个对象的例子。

我们将说一个范畴的对象，但是一个集合的元素，这反映了范畴和集合之间的差异。 定义一个范畴并不需要规定它的对象是如何形成的，只需要理解范畴的（任意的）对象是什么。每个集合确定一个范畴，即集合的元素范畴，但反之不成立：例如，集合的范畴和命题的范畴都不是集合，因为我们无法描述它们所有元素是如何形成的。 现在我们可以说，判断是一个关于某个对象是一个范畴的陈述（$a \in A,\, A\ set, \ldots$）或者两个范畴的对象相等的陈述（$a = b \in A$, $A = B, \ldots$）。

那么在逻辑意义上，与罗素的分层（或简单）类型理论中给出的类型一词有关吗？ 类型与范畴还是与集合同义吗？ 在某些情况下是与一个同义词，而在其他情况下是与另一个同义词。 正是这种将两个不同概念混淆在一起的困惑导致了简单类型理论的非预测性。 当类型被定义为命题函数的意义范围时，类型就是量词的范围，这时类型似乎与集合是相同的东西。另一方面，当谈论简单类型的命题、个体属性、个体之间的关系等时，类型和范畴似乎是相同的。 命题、属性、关系等的分层类型和 all命题、属性、关系等的简单类型之间的重要区别在于，分层类型是（或可以理解为）集合，因此对它们进行量化是有意义的，而简单类型只是范畴。

例如， B^A是一个集合，是从集合 A到集合 B的函数集（B^A将被引入为（$\Pi x \in A$） B（x），当 B（x）恒等于 B时）。 特别地， $\{0, 1\}^A$是一个集合，但它与 $\wp$（A）不是同一个东西，后者只是一个范畴。 之所以可以将 B^A看作是一个集合，是因为我们将函数的概念作为原始概念，而不是将函数定义为有序对的集合或满足通常的存在性和唯一性条件的二元关系（这将使其成为一个范畴，类似于 $\wp$（A））而不是一个集合。

当谈论计算机科学中的数据类型时，可以同样地说数据集。 因此，在这里，类型始终与集合同义，而不是与范畴。

关于规则的一般说明

我们现在开始给出我们使用的不同符号的规则。我们将遵循一个常见的模式来给出这些规则。 对于每个操作，我们有四条规则：集合形成，

引入，

消除，

相等。

形成规则表示我们可以从其他集合（命题）或集合族（命题函数）中形成某个特定的集合（命题）。
引入规则说明集合的规范元素（以及相等的规范元素），从而给出了它的含义。 消除规则展示了我们如何在由引入规则定义的集合上定义函数。 相等规则通过展示由消除规则定义的函数如何作用于由引入规则生成的集合的规范元素来关联引入和消除规则。

在将集合解释为命题时，形成规则用于形成命题，引入和消除规则类似于Gentzen的规则[9]，而相等规则对应于Prawitz的约简规则[10]。我们还在这里提到，对于每个形成规则、引入规则和消除规则，都存在一个相等规则，允许我们用相等的替代物替换相等的。

规则应该是直接推理的规则；我们不能进一步分析它们，只能解释它们。然而，最终，没有解释可以替代每个人的理解。

一族集合的笛卡尔积

给定一个集合 A和一个集合族 $B(x)$ 在集合 A上，我们可以构造积：

Π-形成

$$\frac{\text{一个集合} \qquad \begin{array}{c}(x \in A)\\ B(x)\text{集合}\end{array}}{(\Pi x \in A)\, B(x)\text{集合}} \qquad \frac{A = C \qquad \begin{array}{c}(x \in A)\\ B(x) = D(x)\end{array}}{(\Pi x \in A)\, B(x) = (\Pi x \in C)\, D(x)}$$

[9] G. Gentzen, Untersuchungen ¨ uber das logische Schliessen, *Mathematische Zeitschrift*, Vol. 39, 1934, pp. 176–210 and 405–431.

[10] D. Prawitz, *Natural Deduction, A Proof-Theoretical Study* , Almqvist & Wiksell, Stockholm, 1965.

第二条规则表明，从相等的参数中我们得到相等的值。 对于所有其他形成集合的操作，情况也是一样的，我们将不再详述。 第一条规则的结论是某个东西是一个集合。要理解它是哪个集合，我们必须知道它的规范元素以及它的相等规范元素是如何形成的。 这由引入规则解释：

Π-引入

$$\frac{\begin{matrix}(x \in A)\\ b(x) \in B(x)\end{matrix}}{(\lambda x)\, b(x) \in (\Pi x \in A)\, B(x)}$$

$$\frac{\begin{matrix}(x \in A)\\ b(x) = d(x) \in B(x)\end{matrix}}{(\lambda x)\, b(x) = (\lambda x)\, d(x) \in (\Pi x \in A)\, B(x)}$$

请注意，即使 $b(a)$不是 $B(a)$的规范元素，对于 $a \in A$，这些规则也引入了规范元素和相等的规范元素。 此外，我们假设满足通常的变量限制，即 x不在任何假设中自由出现，除了（形式为） x的假设。
$\in$ 一个. 请注意，必须理解 $b(x) \in B($
$x)$ $(x \in A)$ 是一个函数，才能形成规范元素 $(\lambda x)\, b(x) \in (\Pi x$
$\in A)\, B(x)$；我们可以说后者是前者的名称。由于一般情况下，没有穷尽的规则来生成从一个集合到另一个集合的所有函数，因此我们无法归纳地生成形如 $(\Pi x$
$\in A)\, B(x)$（或者特别地，形如 B^A
，如 $\mathbb{N}^{\mathbb{N}}$）的集合的所有元素。

现在我们可以证明集合形成的第二条规则。 因此，设 $(\lambda x)\, b(x)$ 是 $(\Pi x$" 的一个规范元素。 $\in A)\, B(x)$. 然后 $b(x) \in B(x)$ $(x \in A)$. 因此，假设 x
$\in C$，我们得到 x $\in$ 通过前提中集合的对称性和相等性，我们有$A = C$
，因此 $b(x) \in B(x)$。 现在，根据前提 $B(x) = D(x)$ $(x \in A)$，再次根据集合的相等性（这也适用于集合族），我们得到 $b(x) \in D(x)$，因此 $(\lambda x)\, b(x)$
$\in (\Pi x \in C)\, D(x)$ 通过Π引入。
另一个方向类似。

$$\frac{\dfrac{\dfrac{(x \in C) \quad \dfrac{A = C}{C = A}}{x \in A}}{b(x) \in B(x)} \quad \dfrac{\dfrac{(x \in C) \quad \dfrac{A = C}{C = A}}{x \in A}}{B(x) = D(x)}}{\dfrac{b(x) \in D(x)}{(\lambda x)\, b(x) \in (\Pi x \in C)\, D(x)}}$$

我们注意到上述推导不能被视为类型论中第二个Π形成规则的正式证明，因为没有一个正式规则来证明两个集合之间的相等性，这直接对应于这种相等性的解释。 我们还必须证明

$$\frac{(\lambda x)\, b(x) = (\lambda x)\, d(x) \in (\Pi x \in A)\, B(x)}{(\lambda x)\, b(x) = (\lambda x)\, d(x) \in (\Pi x \in C)\, D(x)}$$

在相同的假设下。 所以让$(\lambda x)\, b(x)$和$(\lambda x)\, d(x)$成为$(\Pi x \in A)\, B(x)$的相等的规范元素。然后 $b(x) = d(x) \in B(x)\ (x \in A)$，因此推导

$$\dfrac{\dfrac{\dfrac{(x \in C) \qquad \dfrac{A = C}{C = A}}{x \in A} \qquad \dfrac{\dfrac{(x \in C) \qquad \dfrac{A = C}{C = A}}{x \in A}}{B(x) = D(x)}}{b(x) = d(x) \in B(x)}}{b(x) = d(x) \in D(x)}}{(\lambda x)\, b(x) = (\lambda x)\, d(x) \in (\Pi x \in C)\, D(x)}$$

表明 $(\lambda x)\, b(x)$ 和 $(\lambda x)\, d(x)$ 是 $(\Pi x \in C)\, D(x)$ 的相等规范元素。

Π-消除

$$\frac{c \in (\Pi x \in A)\, B(x) \qquad a \in A}{\mathsf{Ap}(c,\, a) \in B(a)}$$

$$\frac{c = d \in (\Pi x \in A)\, B(x) \qquad \text{一个元素} a = \text{一个元}}{\mathsf{Ap}(c,\, a) = \mathsf{Ap}(d,\, b) \in B(a)}$$

我们必须解释新常量Ap（Ap代表应用）的含义。Ap（c, a）是获取B（a）的规范元素的方法，现在我们解释如何执行它。我们知道c $\in$ （$\Pi x \in A$） B（x），也就是说，c是一个方法，它产生一个规范元素（λx）b（x）作为结果。
现在取$a \in A$，并将其替换为b（x）中的x。然后b（a）$\in B$（a）。 计算b（a），我们得到B（a）的一个规范元素，正如所需。 当然，在这个解释中，没有进行具体的计算；它具有思想实验（德语Ged$a$nkene$x$periment）的性质。我们使用$A\mathrm{p}$（c, a）而不是更常见的b（a），以区分应用二元函数$A\mathrm{p}$到两个参数c和a的结果与应用b到a的结果。 $A\mathrm{p}$（c, a）对应于组合逻辑中的应用操作（$c\, a$）。. 但请记住，在组合逻辑中没有类型限制，因为可以始终形成（$c\, a$），对于任何c和a。

Π-相等

$$\frac{\dfrac{(x \in A)}{a \in A \qquad b(x) \in B(x)}}{\mathsf{Ap}((\lambda x)\, b(x),\, a) = b(a) \in B(a)}$$

$$\frac{c \in (\Pi x \in A)\, B(x)}{c = (\lambda x)\, \mathsf{Ap}(c,\, x) \in (\Pi x \in A)\, B(x)}$$

第一个等式规则展示了新函数 Ap如何作用于$(\Pi x \in A)\, B(x)$的规范元素。 将$(\lambda x)\, b(x)$视为程序 $b(x)$的名称。
然后第一个规则表明，将程序的名称应用于参数会产生与将该参数作为输入执行程序相同的结果。
类似地，第二个规则用于获得一个记法 $\mathsf{Ap}(c, x)$，用于表示一个只知道名称 c的程序。 第二个规则可以解释如下。
回想一下，如果两个元素产生相等的规范元素作为结果，则它们是相等的。 所以假设 c产生结果$(\lambda x)\, b(x)$，其中 $b(x) \in B(x)\ (x \in A)$。由于$(\lambda x)\, \mathsf{Ap}(c, x)$是规范的，我们要证明的是

$$(\lambda x)\, b(x) = (\lambda x)\, \mathsf{Ap}(c,\, x) \in B(x) \qquad (x \in A)$$

根据相等元素的Π引入规则，我们需要 $b(x) = \mathsf{Ap}(c, x) \in B(x)(x \in A)$。 这意味着 $b(a) = \mathsf{Ap}(c, a) \in B(a)$，只要 $a \in A$。 但这是真的，因为 c产生$(\lambda x)\, b(x)$，因此 $\mathsf{Ap}(c, a)$产生与 $b(a)$相同的值。

乘积的规则包含了 B^A的规则，它是从集合 A到集合 B的函数集合。 实际上，我们将 B^A定义为$(\Pi x \in A)\, B$，其中 B不依赖于 x。 在这里，定义相等性的概念是有用的。

定义等式

定义相等性是内涵相等性，或者是意义上的相等性（同义词）。
我们使用符号 $\equiv$或 $=_{\text{def}}$。 （首次由布拉利-福尔蒂引入）。定义相等性 $\equiv$是语言表达式之间的关系；不应将其与对象之间的相等性（集合、集合元素等）混淆，我们用 $=$ 表示。 定义相等性是由缩写定义、绑定变量的改变和等同替换原则生成的等价关系。 因此，它是可判定的，但并不是以 $a \equiv b \vee \neg(a \equiv b)$ 的方式成立，仅仅因为 $a \equiv b$在当前理论的意义上不是一个命题。 定义相等性在检查证明的形式正确性方面是必不可少的。 实际上，要检查推理的正确性，如下所示

$$\frac{A\text{为真。} \qquad\qquad B\text{真}}{A \,\&\, B \text{ 真}}$$

例如，我们必须特别确保上面线条上的表达式 A和 B的出现与下面对应的出现相同，也就是说它们在定义上是相等的。请注意，对表达式的重写不被视为正式推理。

笛卡尔积的应用

首先，使用定义相等性，我们现在可以通过放置来定义 B^A

$$B^A \ \equiv\ A \to B \ \equiv\ (\Pi x \in A)\, B,$$

只要 B不依赖于 x。 接下来，我们考虑将命题解释为集合的Π规则。 如果在第一个规则中，Π形成，我们将 $B(x)$视为命题而不是集合，则在定义$(\forall x \in A)\, B(x) \equiv (\Pi x \in A)\, B(x)$，它变成了规则

∀-形成

$$\frac{\begin{array}{c}(x \in A)\\ \text{一个集合} \qquad B(x)\text{命题 .}\end{array}}{(\forall x \in A)\, B(x)\text{命题 .}}$$

它表明全称量化形成了命题。 一个集合仅仅表示全称量词的范围是一个集合，这就是为什么我们不将其改为 A命题的原因。注意，∀-形成规则只是Π-形成的一个实例。我们同样有

∀-引入

$$\frac{\begin{array}{c}(x \in A)\\ B(x)\ \text{真}\end{array}}{(\forall x \in A)\, B(x)\ \text{真}}$$

这是通过抑制证明 $b(x)$从Π-引入规则获得的。
通常情况下，我们写 A真而不是 $a \in A$，当 A被视为命题时，我们不关心它的证明（构造）是什么。

更一般地，我们可以按如下方式省略证明。假设

$$\begin{array}{l} a(x_1, \ldots, x_n) \in A(x_1, \ldots, x_m) \\ \quad (x_1 \in A_1, \ldots, x_m \in A_m(x_1, \ldots, x_{m-1}), \\ \quad\ x_{m+1} \in A_{m+1}(x_1, \ldots, x_m), \ldots, x_n \in A_n(x_1, \ldots, x_m)) \end{array}$$

也就是说，假设 A_{m+1}到 A_n和 A仅依赖于 $x_1, \ldots, x_m$。 那么，如果我们只对 $A(x_1, \ldots, x_m)$的真实性感兴趣，写出 $A_{m+1}, \ldots, A_n$的元素的显式符号是不必要的；因此我们用缩写表示

$$\begin{array}{l} A(x_1, \ldots, x_m)\ \mathit{true} \\ \quad (x_1 \in A_1, \ldots, x_m \in A_m(x_1, \ldots, x_{m-1}), \\ \quad\ A_{m+1}(x_1, \ldots, x_m)\ \mathit{true}, \ldots, A_n(x_1, \ldots, x_m)\ \mathit{true}). \end{array}$$

同样地，我们写

$$\begin{array}{l} A(x_1, \ldots, x_m)\text{命题 .} \\ \quad (x_1 \in A_1, \ldots, x_m \in A_m(x_1, \ldots, x_{m-1}), \\ \quad\ A_{m+1}(x_1, \ldots, x_m)\ \mathit{true}, \ldots, A_n(x_1, \ldots, x_m)\ \mathit{true}) \end{array}$$

也就是说，对于A（x_1，...，x_m）是一个命题，只要$x_1 \in A_1$，...，$x_m \in A_m$（x_1，...，$x_{m\text{-}1}$）和A_{m+1}（x_1，...，x_m），...，A_n（x_1，...，x_m）都为真，作为A（x_1，...，x_m）的缩写。

（$x_1 \in A_1$，...，$x_m \in A_m$（x_1，...，$x_{m\text{-}1}$），$x_{m+1} \in A_{m+1}$（x_1，...，x_m），...，$x_n \in A_n$（x_1，...，x_m））。回到$\forall$规则，从Π消去规则，我们特别地有

$\forall$-消除

$$\frac{a \in A \qquad (\forall x \in A)\, B(x) \text{ 真}}{B(a) \text{ 真}}$$

恢复证明，我们可以看到，如果 c是 $(\forall x \in A)\, B(x)$ 的证明，那么 $\mathsf{Ap}(c, a)$是 $B(a)$ 的证明；因此，$(\forall x \in A)\, B(x)$ 的证明是一种将 A的任意元素映射为 $B(a)$ 的证明方法，与直观的全称量词解释一致。

现在我们定义

$$A \supset B \;\equiv\; A \to B \;\equiv\; B\,A \;\equiv\; (\Pi x \in^{A})\; B$$

，其中 B不依赖于 x，我们从Π规则中得到了蕴含的规则。 根据Π形成规则，假设 B不依赖于 x，我们得到

$\supset$-形成

$$\frac{A\text{命题。} \qquad \begin{array}{c}(A \text{ 真})\\ B\text{命题}.\end{array}}{A \supset B\text{命题}.}$$

这是对形成 $A \supset B$的通常规则的推广，因为我们也可以使用假设 A真来证明 B命题。这种推广在科尔莫戈洛夫解释中可能更明显，因为我们可能只有在假设问题 A可以解决的情况下才能判断问题 B是一个问题，这显然足以使问题 $A \supset B$，也就是解决问题 B的问题，有意义。 $\supset$的推理规则为： $\supset$-引入

$$\frac{\begin{array}{c}(A \text{ 真})\\ B\text{真}\end{array}}{A \supset B \text{ 真}}$$

这来自于Π-引入规则通过抑制证明而来，以及

$\supset$-消除

$$\frac{A \supset B \text{ 真} \qquad A\text{为真。}}{B\text{真}}$$

这是通过Π消除规则得到的相同过程。

例子（组合子 I）。 假设 A是一个集合，且 $x \in A$。. 然后，通过Π-引入，我们得到 $(\lambda x)\, x \in A \to A$，因此，对于任何命题 A，$A \supset A$是成立的。 这表达了 $A \supset A$的证明方法：采用相同的证明（构造）。 我们可以通过定义组合子 I来表示，即 $I \equiv (\lambda x)\, x$。 注意，相同的 I属于任何形式为 $A \to A$的集合，因为我们没有为不同类型使用不同的变量。

例子（组合子 K）。 然后，通过对 y进行 λ抽象，我们得到 $(\lambda y)x \in B(x) \to A$，并且，通过对$x$进行 λ抽象，得到 $(\lambda x)\,(\lambda y)x \in (\Pi x \in A)\,(\,B\,(x) \to A\,)$。 我们可以通过定义组合子 K来表示，即 $K \equiv (\lambda x)\,(\lambda y)\, x$。 如果我们将 A和 B看作命题，其中B不依赖于x， K作为 $A \supset (\,B \supset A)$的证明出现；因此 $A \supset (\,B \supset A)$是成立的。 K表示的方法是：给定任何 A的证明 x，取从B到 A的函数，该函数对于任何B的证明 y都是常数 x。

例子（组合子 S） 然后 Ap（ f，x） $\in B$（x）和 Ap（ g，x） $\in$（Π $y \in B$（x）） C（x，y）通过Π消除。 所以，再次通过Π消除，

$$\mathsf{Ap}(\mathsf{Ap}(g,\, x), \mathsf{Ap}(f,\, x)) \in C(x, \mathsf{Ap}(f,\, x)).$$

现在，通过对x进行λ-抽象，我们得到

$$(\lambda x)\, \mathsf{Ap}(\mathsf{Ap}(g,\, x), \mathsf{Ap}(f,\, x)) \in (\Pi x \in A)\, C(x, \mathsf{Ap}(f,\, x)),$$

然后，通过对f进行λ-抽象，

$$\begin{aligned}&(\lambda f)\ (\lambda x)\, \mathsf{Ap}(\mathsf{Ap}(g,\, x), \mathsf{Ap}(f,\, x)) \\ &\quad \in (\Pi f \in (\Pi x \in A)\, B(x))\ (\Pi x \in A)\, C(x, \mathsf{Ap}(f,\, x)).\end{aligned}$$

由于右侧的集合不依赖于 g，对 g进行抽象，我们得到

$$\begin{aligned}&(\lambda g)\ (\lambda f)\ (\lambda x)\, \mathsf{Ap}(\mathsf{Ap}(g,\, x), \mathsf{Ap}(f,\, x)) \\ &\quad \in (\Pi x \in A)\ (\Pi y \in B(x))\, C(x,\, y) \\ &\qquad \to (\Pi f \in (\Pi x \in A)\, B(x))\ (\Pi x \in A)\, C(x, \mathsf{Ap}(f,\, x)).\end{aligned}$$

现在我们可以放置

$$S \ \equiv\ (\lambda g)\ (\lambda f)\ (\lambda x)\, \mathsf{Ap}(\mathsf{Ap}(g,\, x), \mathsf{Ap}(f,\, x))$$

这是常见的组合子 S，用$\lambda g\, f\, x.\, g\, x\ (f\, x)$在组合逻辑中表示。 通过这种方式，我们给组合子 S分配了一个类型（集合）。 现在将 C（x, y）视为一个命题函数。然后我们证明了

$$\begin{aligned}&(\forall x \in A)\ (\forall y \in B(x))\, C(x,\, y) \\ &\quad \supset (\forall f \in (\Pi x \in A)\, B(x))\ (\forall x \in A)\, C(x, \mathsf{Ap}(f,\, x))\ \text{真}\end{aligned}$$

传统上写作

$$(\forall x \in A)\,(\forall y \in B(x))\,(C(x,\,y)) \supset (\forall f \in \textstyle\prod_{x\in A} B_x)\,(\forall x \in A)\,C(x,\,f(x)).$$

如果我们假设 $C(x,\,y)$ 不依赖于 y, 那么 $(\Pi y \in B(x))\,C(x,\,y) \equiv B(x) \to C(x)$ 因此

$$S \in (\Pi x \in A)\,(B(x) \to C(x)) \to ((\Pi x \in A)\,B(x) \to (\Pi x \in A)\,C(x)).$$

因此，如果我们将 $B(x)$ 和 $C(x)$ 视为命题，我们有

$$(\forall x \in A)\,(B(x) \supset C(x)) \supset ((\forall x \in A)\,B(x) \supset (\forall x \in A)\,C(x))\ true.$$

现在假设B（x）不依赖于x，C（x，y）不依赖于x和y。然后我们得到

$$S \in (A \to (B \to C)) \to ((A \to B) \to (A \to C)),$$

也就是说，在逻辑解释中，

$(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$ 成立。

这正是希尔伯特风格命题演算的第二公理。 在最后一种情况下，上面的证明在树形式中写出如下：

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(x \in A) \qquad f \in A \to B}{\mathsf{Ap}(f,\,x) \in B} \qquad \dfrac{(x \in A) \qquad g \in A \to (B \to C)}{\mathsf{Ap}(g,\,x) \in B \to C}}{\mathsf{Ap}(\mathsf{Ap}(g,\,x), \mathsf{Ap}(f,\,x)) \in C}}{(\lambda x)\,\mathsf{Ap}(\mathsf{Ap}(g,\,x), \mathsf{Ap}(f,\,x)) \in A \to C}}{(\lambda f)\,(\lambda x)\,\mathsf{Ap}(\mathsf{Ap}(g,\,x), \mathsf{Ap}(f,\,x)) \in (A \to B) \to (A \to C)}}{(\lambda g)\,(\lambda f)\,(\lambda x)\,\mathsf{Ap}(\mathsf{Ap}(g,\,x), \mathsf{Ap}(f,\,x)) \in (A \to (B \to C)) \to ((A \to B) \to (A \to C))}$$

一族集合的不交并

第二组规则是关于一个集合族的不交并。

Σ-形成

$$\dfrac{\begin{matrix}(x \in A)\\ \text{一个集合} \qquad B(x)\text{集合}\end{matrix}}{(\Sigma x \in A)\ B\ (x)\ \text{集合}}$$

（$\Sigma x \in A$） B（x）的传统记法是 $\sum_{x\in A} B_x$（$\coprod_{x\in A} B_x$ 或 $\bigcup_{x\in A} B_x$）。我们现在解释集合（$\Sigma x \in A$） B（x）是如何形成的。我们使用以下规则：

Σ-引入

$$\frac{a \in A \qquad b \in B(a)}{(a,\, b) \in (\Sigma x \in A)\, B(x)}$$

我们现在可以证明与Σ-形成相关的等式规则：

$$\frac{\begin{array}{cc} & (x \in A) \\ A = C & B(x) = D(x) \end{array}}{(\Sigma x \in A)\, B(x) = (\Sigma x \in C)\, D(x)}$$

事实上，$(\Sigma x \in A)\, B(x)$的任何规范元素都具有形式$(a,\, b)$，其中
$a \in A$且 $b \in B(a)$由Σ-引入。 然后，根据集合的相等性和替换，我们还有 $a \in C$ 和 $b \in D(a)$。 因此，根据
Σ-引入，(a, b)∈(Σx∈C)D(x)。 另一个方向类似。

Σ-消除

$$\frac{\begin{array}{cc} & (x \in A,\, y \in B(x)) \\ c \in (\Sigma x \in A)\, B(x) & d(x,\, y) \in C((x,\, y)) \end{array}}{\mathsf{E}(c, (x,\, y)\, d(x,\, y)) \in C(c)}$$

在这里，我们假设前提 $C(z)$集合 $(z \in (\Sigma x \in A)\, B(x))$，尽管它没有明确写出。 （准确地说，我们还应该写出前提 A集合和 $B(x)$集合 $(x \in A)$。）我们通过展示新常量 E如何操作其参数来解释Σ-消除规则。 所以假设我们知道前提。 然后我们按照以下方式执行 $\mathsf{E}(c, (x, y)\, d(x, y))$。 首先执行 c，得到一个形式为（a, b）的规范元素，其中 $a \in A$且$b \in B(a)$。 现在在右前提中分别用 a 和 b替换x和 y，得到 $d(a, b) \in C((a, b))$。 执行 $d(a, b)$我们得到一个规范元素 e 属于 $C((a, b))$。 我们现在想要证明 e也是 $C(c)$的一个规范元素。 这是一个普遍事实，如果 $a \in A$且 a具有值 b，则 $a = b \in A$（注意，然而，这并不意味着 $a = b \in A$必定可以通过某个特定的形式规则推导出来）。 在我们的情况下， $c = (a, b) \in (\Sigma\, x \in A)\, B(x)$因此，通过替换， $C(c) = C((a, b))$。 记住两个集合相等的含义，我们从 e是 $C((a, b))$的一个规范元素这一事实推断出 e也是 $C(c)$ 的一个规范元素。

对于 $\mathsf{E}(c, (x, y)\, d(x, y))$的另一种表示可以是 $(\mathsf{E}x, y)\ (c, d(x, y)),$但我们更喜欢第一种表示，因为它更清楚地显示了 x和 y只在 $d(x, y)$中绑定。

Σ-相等性

$$\frac{\begin{array}{ccc} & & (x \in A,\, y \in B(x)) \\ a \in A & b \in B(a) & d(x,\, y) \in C((x,\, y)) \end{array}}{\mathsf{E}((a,\, b), (x,\, y)\, d(x,\, y)) = d(a,\, b) \in C((a,\, b))}$$

(在这里，就像在Σ-消除中一样， $C(z)\ set\ (z \in (\Sigma x \in A)\, B(x))$是一个隐含的前提。) 假设我们知道前提，通过想象执行E$((a,b),(x,y)\, d(x,y))$来证明结论是合理的。 实际上，我们首先执行(a,b)，得到(a,b)本身作为结果；然后我们将 a, b 替换为 x, y在 $d(x,y)$中，得到$d(a,b) \in C((a,b))$，并执行 $d(a,b)$直到我们得到一个规范元素 $e \in C((a,b))$。 相同的规范元素由 $d(a,b)$产生，因此结论是正确的。

Σ-等式的第二条规则，类似于Π-等式的第二条规则，现在是可导出的，我们稍后会看到。

不交并的应用

正如我们已经对笛卡尔积所做的那样，我们现在将看到不相交并的逻辑解释。如果我们放置

$$(\exists x \in A)\, B(x) \ \equiv\ (\Sigma x \in A)\, B(x),$$

然后，根据Σ-规则，将 $B(x)$ 解释为 A上的命题函数，我们得到特殊情况：

∃-形成

$$\frac{\begin{array}{cc} & (x \in A) \\ \text{一个集合} & B(x)\text{命题} \end{array}.}{(\exists x \in A)\, B(x)\text{命题。}}$$

∃-引入

$$\frac{a \in A \qquad B(a)\ \text{真}}{(\exists x \in A)\, B(x)\ \text{真}}$$

根据存在量词的直观解释，Σ-引入规则可以解释为一个（规范的）证明，即(a, b)的对，其中 b是一个证明，证明了 a满足 B。省略证明，我们得到∃-引入规则，但通常不明确指定第一个前提 $a \in A$。

∃-消除

$$\frac{(\exists x \in A)\, B(x)\ \text{真} \qquad \begin{array}{c} (\text{对于每个} x \in A \text{和}\ B(\text{对于每个} x)\text{成立}) \\ C\text{成立} \end{array}}{C\text{成立}}$$

在这里，通常情况下，除了明确写出的变量x之外，不允许有任何假设。 Σ-消除规则比 ∃-消除规则更强，后者是通过抑制证明而得到的，因为我们还考虑了证明（构造），这在一阶谓词逻辑的语言中是不可能的。

这种额外的强度在处理下面的左投影和右投影时将会显现出来。 如果我们定义

并集的规则也提供了传统的合取规则和两个集合的笛卡尔积的传统性质，如果我们定义

$$A \,\&\, B \;\equiv\; A \times B \;\equiv\; (\text{对于每个} x \in A$$

其中 B不依赖于 x。 我们这里只推导出合取的规则。

$\&$-形成

$$\frac{\begin{array}{cc} & (A \text{ 真} \quad) \\ A\text{命题。} & B\text{命题 .} \end{array}}{A \,\&\, B \text{ 命题}}$$

这个规则是Σ-形成的一个实例，也是通常形成命题$A \;\&\; B$的一般化规则，因为我们只有在假设A为真的情况下才能知道B是一个命题。

$\&$-引入

$$\frac{A\text{为真。} \qquad B\text{真}}{A \,\&\, B \text{ 真}}$$

恢复证明，我们可以看到$A \;\&\; B$的（规范的）证明是一个对(a, b)的配对，其中a和b分别是A和B的给定证明。

$\&$-消除

$$\frac{\begin{array}{cc} & (A\text{为真},\, B\text{为真}) \\ A \,\&\, B \text{ 真} & C\text{为真} \end{array}}{C\text{成立}}$$

根据这个$\&$-消除规则，我们可以选择C为A和B本身，从而得到标准的$\&$-消除规则：

$$\frac{A \,\&\, B \text{ 真} \qquad (A\text{为真})}{A\text{为真。}} \qquad \frac{A \,\&\, B \text{ 真} \qquad (B\text{为真})}{B\text{真}}$$

例子（左投影）。我们定义

$$\mathsf{p}(c) \;\equiv\; \mathsf{E}(c, (x,\, y)\, x)$$

并将其称为左投影 $\text{of } c\text{since}$它是通过任意元素 $c\text{of } (\Sigma x \in A)\, B(x)$产生的一对的第一个（左）坐标的值的方法。 实际上，如果我们将解释Σ-消除中的项 $d(x, y)$取为 x，则我们可以看到执行 $\mathsf{p}(c)$时，我们首先获得一对(a, b)其中 $a \in A$和 $b \in B(a)$，这是 c的值，然后在x， y中用 a， b替换，得到 a，这将执行以产生 A的规范元素。因此，将 $C(z)$取为 A，将 $d(x, y)$取为 x在Σ-消除和Σ-等式的规则中，我们得到以下派生规则：

左投影

$$\frac{c \in (\Sigma x \in A)\, B(x)}{\mathsf{p}(c) \in A} \qquad \frac{a \in A \qquad b \in B(a)}{\mathsf{p}((a,\, b)) = a \in A}$$

如果我们现在转向逻辑解释，我们可以看到

$$\frac{c \in (\Sigma x \in A)\, B(x)}{\mathsf{p}(c) \in A}$$

成立，这意味着从一个 $proof$of $(\exists x \in A)\, B(x)$我们可以得到一个元素of A，使得属性 Bholds。 因此，我们不需要描述运算符 $(\iota x)\, B(x)$（满足 $B(\ x)$的x）或选择运算符 $(\epsilon x)\, B(x)$（满足 $B(\ x)$的x），因为从直观的角度来看，当我们有一个证明时，$(\exists x \in A)\, B(x)$是真的。一个 epsilon 项 $(\epsilon x)\, B(x)$的困难在于它被构造为 $B(x)$属性本身的函数，而不是$(\exists x)\, B(x)$的证明。 这就是为什么希尔伯特不得不假设一个形式为的规则

$$\frac{(\exists x)\, B(x) \text{ 真}}{(\epsilon x)\, B(x) \text{ 个体}}$$

我们刚刚证明了一个对应的对应物，以及一个形式为的规则

$$\frac{(\exists x)\, B(x) \text{ 真}}{b((\epsilon x)\, B(x)) \text{ 真}}$$

这在我们将在下一个例子中看到的“右投影”的规则中有一个对应物。

例子（右投影）。我们定义

$$\mathsf{q}(c) \quad \equiv \quad \mathsf{E}(c,\, (x,\, y)\, y)。$$

将 $d(x, y)$取为Σ-消除规则中的 y。 从 $x \in A,\ y \in B(x)$我们得到 $\mathsf{p}((x, y)) = x \in A$通过左投影，因此 $B(x) = B(\mathsf{p}((x, y)))$。
现在选择 $C(z)$集合 $(z \in (\Sigma x \in A)\, B(x))$ 作为家族 $B(\mathsf{p}(z))$集合 $(z \in (\Sigma x \in A)\, B(x))$. 然后$\Sigma$-消除规则给出 $\mathsf{q}(c) \in B(\mathsf{p}(c))$. 更正式地说:

$$\frac{c \in (\Sigma x \in A)\, B(x) \qquad \dfrac{(y \in B(x)) \qquad \dfrac{\dfrac{\dfrac{(x \in A) \qquad (y \in B(x))}{\mathsf{p}((x,\, y)) = x \in A}}{x = \mathsf{p}((x,\, y)) \in A}}{B(x) = B(\mathsf{p}((x,\, y)))}}{y \in B(\mathsf{p}((x,\, y)))}}{\mathsf{q}(c) \equiv \mathsf{E}(c, (x,\, y)\, y) \in B(\mathsf{p}(c))}$$

所以我们有：

右投影

$$\frac{c \in (\Sigma x \in A)\, B(x)}{\mathsf{q}(c) \in B(\mathsf{p}(c))} \qquad \frac{a \in A \qquad b \in B(a)}{\mathsf{q}((a,\, b)) = b \in B(a)}$$

这些规则中的第二个是通过Σ-相等性推导出来的，方式与第一个规则通过Σ-消除相同。

当 $B(x)$被看作是一个命题函数时，右投影的第一个规则表明，如果 c是一个构造$(\exists x \in A)\, B(x)$的话，那么 $\mathsf{q}(c)$是一个构造 $B(\mathsf{p}(c))$的，其中，根据左投影， $\mathsf{p}(c) \in A$。 因此，在结论中省略构造， $B(\mathsf{p}(c))$是真的。 然而需要注意的是，在$B(x)$依赖于 x的情况下，无法省略前提中的构造，因为结论依赖于它。

最后，当 $B(x)$ 不依赖于 x 时，我们可以简单地将其写作 B，并且 A和 B 都被视为命题，右投影的第一个规则简化为

$\&$-消除

$$\frac{A \,\&\, B \text{ 真}}{B\text{真}}$$

通过抑制前提和结论中的构造。

例子（合取的公理）。 我们首先推导出 $A \supset (B \supset (A \,\&\, B))$为真，这是对应于合取引入规则的公理。. 然后通过 Σ-引入，$(x, y) \in (\Sigma x \in A)\ B(x)$，并且通过 Π-引入，$(\lambda y)\,(x, y) \in B\ (\ x) \to (\Sigma x \in A)\, B(x)$（注意 $(\Sigma x \in A)\ B(x)$ 不依赖于 y），以及 $(\lambda x)\,(\lambda y)\,(x, y) \in$

逻辑阅读是

$$(\forall x \in A)\,(B(x) \supset (\exists x \in A)\, B(x)) \text{ 真} \quad ,$$

从中可以得出，特别是当 B不依赖于 x时，

$$A \supset (B \supset (A \,\&\, B)) \text{ 真} \quad .$$

我们现在使用左投影和右投影来推导 $A \,\&\, B \supset A$ 真 和 $A \,\&\, B \supset B$ 真. 为了得到第一个结果，假设 $z \in (\Sigma x \in A)\ B(x)$. 然后，通过左投影，可以得到 $\mathsf{p}(z) \in A$，通过 λ-抽象，可以得到 z，

$$(\lambda z)\, \mathsf{p}(z) \in (\Sigma x \in A)\, B(x) \to A.$$

特别是当 $B(x)$ 不依赖于 x时，我们得到

$$A \,\&\, B \supset A \ true.$$

为了得到第二个结论，从 $z \in (\Sigma x \in A)\, B(x)$ ，我们通过右投影得到 $\mathsf{q}(z) \in B(\mathsf{p}(z))$，因此，通过 λ-抽象，

$$(\lambda z)\, \mathsf{q}(z) \in (\Pi z \in (\Sigma x \in a)\, B(x))\, B(\mathsf{p}(z))$$

(注意 $B(\mathsf{p}(z))$ 依赖于 z). 特别是当 $B(x)$ 不依赖于 x时，我们得到

$$A \,\&\, B \supset B \ true.$$

例子（不相交并的另一个应用）。 Σ-消去规则表明，任何带有参数在 A和 $B(x)$中的函数(x, y)也可以作为单个参数的函数（通过Σ-相等性具有相同的值）$(\Sigma x \in A)\, B(x)$。. 我们现在证明的是对应于这个规则的公理。

我们想要找到一个元素 $(\Pi x \in A)\, (\Pi y \in B(x))\ C((x, y)) \rightarrow (\Pi z \in (\Sigma x \in A)\ B\ (x))\, C(z)$。

为了方便起见，我们定义 $\mathsf{Ap}(f, x, y) \equiv \mathsf{Ap}(\mathsf{Ap}(f, x), y)$。 那么 $\mathsf{Ap}(f, x, y)$ 是一个三元函数，并且 $\mathsf{Ap}(f, x, y) \in C((x, y))\ (x \in A,\ y \in B(x))$。 因此，假设$z \in (\Sigma x \in A)\, B(x)$，通过$\Sigma$-消除，我们得到 $\mathsf{E}(z, (x, y)\, \mathsf{Ap}(f, x, y)) \in C(z)$(在假设 $x \in A$和 $y \in B(x)$的情况下)，通过对 z进行 λ-抽象，我们得到了这个函数。

$$(\lambda z)\, \mathsf{E}(z, (x,\, y)\, \mathsf{Ap}(f,\, x,\, y)) \in (\Pi z \in (\Sigma x \in A)\, B(x))\, C(z)$$

带有参数 f。 因此，我们仍然有假设

$$f \in (\Pi x \in A)\ (\Pi y \in B(x))\, C(x,\, y),$$

我们通过 λ抽象来解除这个假设，得到

$$\begin{aligned}&(\lambda f)\ (\lambda z)\, \mathsf{E}(z, (x,\, y)\, \mathsf{Ap}(f,\, x,\, y)) \in \\ &\quad (\Pi x \in A)\ (\Pi y \in B(x))\, C\,(x,\, y) \rightarrow (\Pi z \in (\Sigma x \in A)\, B(x))\, C\,(z)。\end{aligned}$$

在逻辑阅读中，我们有

$$(\forall x \in A)\ (\forall y \in B(x))\, C((x,\, y)) \supset (\forall z \in (\Sigma x \in A)\, B(x))\, C(z)\ true,$$

这可以简化为常见的

$$(\forall x \in A)\ (B(x) \supset C) \supset ((\exists x \in A)\, B(x) \supset C)\ true$$

当 C不依赖于 z时，以及

$$(A \supset (B \supset C)) \supset (A \,\&\, B) \supset C)\ true$$

当此外， B独立于 x时。

选择公理

我们现在展示，到目前为止引入的规则，我们可以给出选择公理的证明，其在我们的符号中表示为：

$$(\forall x \in A)\,(\exists y \in B(x))\,C(x,\,y)$$
$$\supset (\exists f \in (\Pi x \in A)\,B(x))\,(\forall x \in A)\,C(x, \mathsf{Ap}(f,\,x))\;\;true.$$

直觉主义数学中的通常论证，基于对逻辑常量的直观解释，大致如下：为了证明$(\forall x)\,(\exists y)\,C(x,\,y) \supset (\exists f)\,(\forall x)\,C(x,\,f(x))$，假设我们有前提的证明。 这意味着我们有一种方法，应用于任意的 x，产生一个证明$(\exists y)\,C(x,\,y)$，即一个由元素 y和一个证明 $C(x,\,y)$组成的对。 令 f为这种方法，对于任意给定的 x，它分配给这个对的第一个分量。 那么对于任意的x，都成立 $C(x,\,f(\,x))$，因此也成立结论。 相同的思想可以用符号表示，得到直观类型论中的形式证明。 如果x是集合 A的任意元素，即$x \in A$，那么通过Π-消去，我们得到

$$\mathsf{Ap}(z,\,x) \in (\Sigma y \in B(x))\,C(x,\,y).$$

我们现在应用左投影来获得

$$\mathsf{p}(\mathsf{Ap}(z,\,x)) \in B(x)$$

和右投影来获得

$$\mathsf{q}(\mathsf{Ap}(z,\,x)) \in C(x, \mathsf{p}(\mathsf{Ap}(z,\,x))).$$

通过 λ-抽象在 x上（或Π-引入），解除 $x \in A$，我们有

$$(\lambda x)\,\mathsf{p}(\mathsf{Ap}(z,\,x)) \in (\Pi x \in A)\,B(x),$$

并且，通过Π-相等性，

$$\mathsf{Ap}((\lambda x)\,\mathsf{p}(\mathsf{Ap}(z,\,x)),\,x) = \mathsf{p}(\mathsf{Ap}(z,\,x)) \in B(x).$$

通过替换，我们得到

$$C(x, \mathsf{Ap}((\lambda x)\,\mathsf{p}(\mathsf{Ap}(z,\,x)),\,x)) = C(x, \mathsf{p}(\mathsf{Ap}(z,\,x)))$$

因此，根据集合的相等性，

$$\mathsf{q}(\mathsf{Ap}(z,\,x)) \in C(x, \mathsf{Ap}((\lambda x)\,\mathsf{p}(\mathsf{Ap}(z,\,x)),\,x))$$

其中 $(\lambda x)\,\mathsf{q}(\mathsf{Ap}(z,\,x))$ 不依赖于 x。通过对 x进行抽象，

$$(\lambda x)\,\mathsf{q}(\mathsf{Ap}(z,\,x)) \in (\Pi x \in A)\,C(x, \mathsf{Ap}((\lambda x)\,\mathsf{p}(\mathsf{Ap}(z,\,x)),\,x)).$$

现在我们使用配对规则（即Σ-引入）来得到

$$((\lambda x)\,\mathsf{p}(\mathsf{Ap}(z,\,x)),(\lambda x)\,\mathsf{q}(\mathsf{Ap}(z,\,x)))\in$$
$$(\Sigma f\in(\Pi x\in A)\,B(x))\;(\Pi x\in A)\,C(x,\mathsf{Ap}(f,\,x))$$

（注意，在最后一步中，引入了新变量 f并替换了
$(\lambda x)\,\mathsf{p}(\mathsf{Ap}(z,\,x))$ 在右侧）。最后通过对 z进行抽象，我们得到

$$(\lambda z)\;((\lambda x)\,\mathsf{p}(\mathsf{Ap}(z,\,x)),(\lambda x)\,\mathsf{q}(\mathsf{Ap}(z,\,x)))\in(\Pi x\in A)\;(\Sigma y\in B(x))\,C(x,\,y)$$
$$\supset(\Sigma f\in(\Pi x\in A)\,B(x))\;(\Pi x\in A)\,C(x,\mathsf{Ap}(f,\,x)).$$

在策梅洛-弗兰克尔集合论中，没有选择公理的证明，因此必须将其作为公理，然而，它似乎很难声称是自明的。 在上述证明中，提供了选择公理的详细证明。 在许多分类语言中，选择公理是可表达的，但没有机制可以证明它。例如，在有限类型的海廷算术中，必须将其作为公理。 在深入发展直觉主义数学时，需要选择公理是明显的，例如，在找到实数序列的极限或满射函数的部分逆时。

such that的概念

除了不相交的并集、存在量词、笛卡尔积 $A\times B$和合取 $A\;\&\;B$，操作Σ还有第五种解释：所有 A中的元素 a，使得 $B(a)$ 成立。 设 A为一个集合， $B(x)$ 为 $x\in A$的命题。 我们想要定义所有 A中使得 $B(\;a)$ 成立的元素的集合（通常写作 $\{x\in A:B(x)\}$）。 拥有一个元素$a\in A$使得$B(a)$ 成立意味着拥有一个元素 $a\in A$以及一个证明 $B(a)$，即一个元素 $b\in B(\;a)$。 因此，满足 $B(x)$ 的 A的所有元素的集合的元素是对(a,b) 的对，其中 $b\in B(a)$，即$(\Sigma x\in A)\,B(x)$ 的元素。然后，Σ规则起到了包含公理（或ZF中的分离原则）的作用。由 $b\in B(a)$ 给出的信息被费弗曼称为证明信息[11]。 一个典型的应用是以下内容。

例子（实数作为柯西序列）。

$$\mathbb{R}\;\equiv\;(\Sigma x\in\mathbb{N}\to\mathbb{Q})\,\text{柯西}\quad(x)$$

是实数的定义，作为满足柯西条件的有理数序列的集合,

$$\text{柯西}\quad(a)\;\equiv\;(\forall e\in\mathbb{Q})\,(e>0\supset(\exists m\in\mathbb{N})\,(\forall n\in\mathbb{N})\,(|a_{m+n}-a_m|\leq e)),$$

[11]S. Feferman, 函数和类的构造性理论，逻辑学讨论会*78*，由M. boffa，D. van Dalen和K. Mc Aloon编辑，北荷兰，阿姆斯特丹，1979年，第159-224页。

其中 a是序列 $a_0, a_1, \ldots$ 这样，一个实数是一个有理数序列，连同一个证明它满足柯西条件的证明。
因此，假设 $c \in \mathbb{R}$, $e \in \mathbb{Q}$和 $d \in (e > 0)$（换句话说， d是一个证明命题$e > 0$的证明），那么通过投影，我们得到 $\mathsf{p}(c) \in \mathbb{N} \to \mathbb{Q}$和 $\mathsf{q}(c) \in Cauchy(\mathsf{p}(c))$。然后

$$\mathsf{Ap}(\mathsf{q}(c), e) \in (e > 0 \supset (\exists m \in \mathbb{N})\ (\forall n \in \mathbb{N})\ (|a_{m+n} - a_m| \leq e))$$

和

$$\mathsf{Ap}(\mathsf{Ap}(\mathsf{q}(c), e), d) \in (\exists m \in \mathbb{N})\ (\forall n \in \mathbb{N})\ (|a_{m+n} - a_m| \leq e).$$

应用左投影，我们得到所需的 m，即

$$\mathsf{p}(\mathsf{Ap}(\mathsf{Ap}(\mathsf{q}(c), e), d)) \in \mathbb{N},$$

现在，通过将 $\mathsf{p}(c)$ 应用于它，我们得到$_m$

$$\mathsf{Ap}(\mathsf{p}(c), \mathsf{p}(\mathsf{Ap}(\mathsf{Ap}(\mathsf{q}(c), e), d))) \in \mathbb{Q}.$$

只有通过证明 $\mathsf{q}(c)$ 我们才知道为了所需的近似值要走多远。

两个集合的不交并

我们现在给出两个集合的和（不相交的并集或余并集）的规则。

$+$-形成

$$\frac{\text{一个集合} \qquad B\text{集合}}{A + B\text{集合}}$$

$A + B$的规范元素是通过以下方式形成的：

$+$-引入

$$\frac{a \in A}{\mathsf{i}(a) \in A + B} \qquad \frac{b \in B}{\mathsf{j}(b) \in A + B}$$

其中 i和 j是两个新的原始常量；它们的使用是为了提供关于 $A + B$的元素来自 A或 B的信息，以及两者之间的情况。
不言而喻，我们也有相等元素的$+$-引入规则：

$$\frac{a = c \in A}{\mathsf{i}(a) = \mathsf{i}(c) \in A + B} \qquad \frac{b = d \in B}{\mathsf{j}\ (b)\ = \mathsf{j}\ (d)\ \in A + B}\text{由于}$$

$A + B$的任意元素 c产生了形式为i（a）或 j（b）的规范元素，知道 $c \in A + B$意味着我们也可以确定该元素 c来自 A和 B中的哪一个集合。

+-消除

$$\frac{\begin{array}{ccc} & (x \in A) & (y \in B) \\ c \in A + B & d(x) \in C(\mathsf{i}(x)) & e(y) \in C(\mathsf{j}(y)) \end{array}}{\mathsf{D}(c, (x)\, d(x), (y)\, e(y)) \in C(c)}$$

其中前提 A集合, B集合和 $C(z)$集合 $(z \in A + B)$ 被假定，尽管没有明确写出。我们现在必须解释如何执行一个新形式的程序 $\mathsf{D}(c, (x)\, d(x), (y)\, e(y))$。假设我们知道 $c \in A + B$。那么 c将产生一个规范元素 $\mathsf{i}(a)$ 其中 $a \in A$或 $\mathsf{j}(b)$ 其中 $b \in B$。在第一种情况下，在 $d(\ x)$ 中用 a替换x，得到 $d(a)$，然后执行它。根据第二个前提， $d(a) \in C(\mathsf{i}(a))$，所以 $d(a)$ 产生一个 $C(\mathsf{i}(a))$ 的规范元素。类似地，在第二种情况下，必须使用 $e(y)$ 而不是 $d(x)$ 来获得 $e(b)$，这将产生一个 $C(\mathsf{j}(b))$ 的规范元素。无论哪种情况，我们都得到了一个 $C(c)$ 的规范元素，因为如果 c具有值 $\mathsf{i}(a)$，那么 $c = \mathsf{i}(a) \in A + B$和因此 $C(c) = C(\mathsf{i}(a))$，如果 c具有值 $\mathsf{j}(b)$，那么 $c = \mathsf{j}(b) \in A + B$和因此$C(c) = C(\mathsf{j}(b))$。从 D的含义的这个解释中，我们得到了相等规则：

+-相等

$$\frac{\begin{array}{ccc} & (x \in A) & (y \in B) \\ a \in A & d(x) \in C(\mathsf{i}(x)) & \text{不等于}(y) \in C(\mathsf{j}(y)) \end{array}}{\mathsf{D}(\mathsf{i}(a), (x)\, d(x), (y)\, e(y)) = d(a) \in C(\mathsf{i}(a))}$$

$$\frac{\begin{array}{ccc} & (x \in A) & (y \in B) \\ b \in B & d(x) \in C(\mathsf{i}(x)) & \text{不等于}(y) \in C(\mathsf{j}(y)) \end{array}}{\mathsf{D}(\mathsf{j}(b), (x)\, d(x), (y)\, e(y)) = e(b) \in C(\mathsf{j}(b))}$$

变得明显。

现在，两个命题的析取被解释为两个集合的和。因此我们有:

$$A \vee B \ \equiv \ A + B.$$

从+的形成和引入规则，我们得到了 ∨的相应规则:

∨-形成

$$\frac{A\text{命题。} \qquad B\text{命题。}}{A \vee B\text{命题。}}$$

∨-引入

$$\frac{A\text{为真。}}{A \vee B \text{ 真}} \qquad \frac{B\text{真}}{A \vee B\text{为真。}}$$

请注意，如果 a是 A的证明，则 $\mathsf{i}(a)$ 是 $A \vee B$的（规范）证明，对于 B也是类似的。

∨-消除

$$\frac{A \vee B\text{为真。} \qquad \begin{matrix}(A\text{ 真}\quad) \\ C\text{真}\end{matrix} \qquad \begin{matrix}(B\text{ 真}\quad) \\ C\text{真}\end{matrix}}{C\text{成立}}$$

通过选择一个不依赖于z的族 $C \equiv C(z)\,(\,z \in A{+}B)$ 来遵循+-消除规则，并在前提中抑制证明（构造），包括假设和结论。

例子（析取的引入公理）。 假设 A是一个集合， B是一个集合，并且假设 $x \in A$。. 那么通过+-引入，我们有 $\mathsf{i}(x) \in A + B$，因此 $(\lambda x)\,\mathsf{i}(x) \in A \to A + B$通过 λ-抽象在 x上。 如果 A和 B是命题，我们有 $A \supset A \vee B$真。 同样地，通过λ-抽象在y上，我们有$(\lambda y)\,\mathsf{j}(\mathbf{y}) \in B \to A + B$，因此 $B \supset A \vee B$真。

例子（析取的消去公理）。 然后，通过Π-消去，从 $x \in A$，我们有 $A\mathbf{p}\,(f, x) \in C\,(\mathsf{i}\,(x))$，以及从 $y \in B$，我们有 $A\mathbf{p}\,(g, y) \in C\,(\mathsf{j}\,(y))$。 因此，使用 $z \in A + B$，我们可以应用 $+$-消去来获得 $\mathsf{D}\,(z, (x)\,A\mathbf{p}\,(f, x), (y)\,A\mathbf{p}\,(g, y)) \in C\,(z)$，从而解除了 $x \in A$ 和 $y \in B$。 通过λ-抽象，按照顺序 z，g，f，我们得到

$$(\lambda f)\,(\lambda g)\,(\lambda z)\,\mathsf{D}(z, (x)\,\mathsf{Ap}(f, x), (y)\,\mathsf{Ap}(g, y))$$
$$\in (\Pi x \in A)\,C(\mathsf{i}(x)) \to ((\Pi y \in B)\,C(\mathsf{j}(y)) \to (\Pi z \in A + B)\,C(z)).$$

当 $C\,(z)$ 被看作命题时，这给出

$$(\forall x \in A)\,C(\mathsf{i}(x)) \supset ((\forall y \in B)\,C(\mathsf{j}(y)) \supset (\forall z \in A + B)\,C(z))\ true.$$

如果，此外， $C(z)$ 不依赖于 z和 $A,\,B$也是命题，我们有

$$(A \supset C) \supset ((B \supset C) \supset (A \vee B \supset C))\text{为真}.$$

命题等式

现在我们转向等式的公理。 这是一个传统（源自《数学原理》）将谓词逻辑中的等式称为恒等。 然而，恒等这个词更适用于定义等式， $\equiv \text{or} =_{\text{def.}}$，如上所述。 事实上，一个等式陈述，例如，$2^2 = 2 + 2$ 在算术中，并不意味着这两个成员是相同的，而仅仅意味着它们具有相同的值。 然而，在谓词逻辑中的等式也是不同的从我们的等式 $a = b \in A$，因为前者是一个命题，而后者是一个判断。 一种命题等式仍然是必不可少的：我们希望有一个等式 $I(A, a, b)$，它断言 a和 b是集合 A的相等元素，但我们可以对其进行逻辑操作（回想一下，例如否定或量化一个判断是没有意义的）。 在某种意义上， $I(A, a, b)$ 是一个内部形式的 $=$。 然后我们有四种类型的等式：

(1) $\equiv$或 $=$定义

(2) $A = B$,

(3) $a = b \in A$,

(4) $I(A,\, a,\, b)$.

对象之间的相等性通过判断来表达，并且必须针对每个类别（如集合类别（2）或集合元素类别（3））单独定义；（4）是一个命题，而（1）只是一个语言表达式之间的规定关系。 然而，注意 $I(A, a, b)true$是一个判断，它将被证明等价于 $a = b \in A$（这并不意味着它具有相同的意义）。 （1）是内涵的（意义相同），而（2）、（3）和（4）是外延的（对象之间的相等性）。 至于弗雷格，元素 a, b可能具有不同的意义，或者是不同的方法，但具有相同的值。 例如，我们当然有$2^2 = 2 + 2 \in \mathbb{N}$，但不是$2^2 \equiv 2 + 2$。

我-形成

$$\frac{\text{一个集合} \qquad a \in A \qquad b \in A}{I(A,\, a,\, b)\text{集合}}$$

现在我们需要解释如何形成 $I(A, a, b)$的规范元素。 知道 $I(A, a, b)$为真的标准方法是有 $a = b \in A$。 因此，引入规则很简单：如果 $a = b \in A$，则存在一个规范证明 rof $I(A, a, b)$。 在这里， r不依赖于 a, b或 A；当 $a = b \in A$时，不管 $I(A, a, b)$具有什么规范元素，只要它有一个即可。

I-引入

$$\frac{\text{一个元素}a = \text{一个元}}{\mathsf{r} \in I(A,\, a,\, b)}$$

我们现在可以采用与Π、Σ、$+$相同风格的 I的消除和等式规则，即引入一个新的消除操作符。 我们将得出以下规则，我们在这里将其作为原始规则：

I-消除

$$\frac{c \in I(A,\, a,\, b)}{\text{一个元素}a = \text{一个元}}$$

I-相等

$$\frac{c \in I(A,\, a,\, b)}{c = \mathsf{r} \in I(A,\, a,\, b)}$$

最后，注意到 I-形成是迄今为止唯一允许形成集合族的规则。 如果只允许使用Π、Σ、$+$、 $\mathbb{N}_n$、 $\mathbb{N}$、 $\mathcal{W}$这些操作，我们只能得到常数集合。

例子（身份的入门公理）。 假设 A是一个集合，并且让 $x \in A$。然后 $x = x \in A$，通过 I-引入，我们有 $\mathsf{r} \in I(A, x, x)$。. 通过对 x进行抽象，我们有$(\lambda x)\, \mathsf{r} \in (\forall x \in A)\, I\,(A, x, x)$。 因此，$(\lambda x)\, \mathsf{r}$是关于 A的等同性定律的一个规范证明。

$$\dfrac{\dfrac{\dfrac{(x \in A)}{x = x \in A}}{\mathsf{r} \in I(A,\, x,\, x)}}{(\lambda x)\, \mathsf{r} \in (\forall x \in A)\, I(A,\, x,\, x)}$$

例子（等同性的消除公理）。 给定一个集合 A和一个性质$B(x)\mathbf{prop}$。在 A上，我们声称相等的元素满足相同的性质，即满足莱布尼兹的不可辨性原则的等同性法则成立。

$$(\forall x \in A)\ (\forall y \in A)\ (I(A,\, x,\, y) \supset (B(x) \supset B(y)))\ true。$$

为了证明它，假设 $x \in A, y \in A$ 和 $z \in$ 我$(A,\, x,\, y)$. 然后 $x = y \in A$并且因此 $B(x) = B(y)$通过替换。 所以，假设 $w \in B(x)$，根据集合的相等性，我们得到 $w \in B(y)$。 现在通过依次对 $w,\, z,\, y,\, x$ 进行抽象，我们得到一个命题的证明:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(z \in I(A,\, x,\, y))}{x = y \in A} \quad \begin{matrix}(x \in A)\\ B(x)\text{集合}\end{matrix}}{B(x) = B(y)} \quad (w \in B(x))}{w \in B(y)}}{(\lambda w)\, w \in B(x) \supset B(y)}}{(\lambda z)\ (\lambda w)\, w \in I(A,\, x,\, y) \supset (B(x) \supset B(y))}}{(\lambda x)\ (\lambda y)\ (\lambda z)\ (\lambda w)\, w \in (\forall x \in A)\ (\forall y \in A)\, I(A,\, x,\, y) \supset (B(x) \supset B(y))}$$

同样的问题（证明莱布尼兹原理的问题）在《原理》中通过使用非预测性的二阶量化来解决。在那里定义了

$$(a = b)\ \equiv\ (\forall X)\, (X(a) \supset X(b))$$

由此可见，莱布尼兹原理是显而易见的，因为它被用来定义同一性的含义。 在当前语言中，无法对属性进行量化，因此同一性的含义必须以另一种方式定义，而不会使莱布尼兹原理无效。

例子（投影定律的逆证明）。 我们现在可以证明推理规则

$$\dfrac{c \in (\Sigma x \in A)\, B(x)}{c = (\mathsf{p}(c), \mathsf{q}(c)) \in (\Sigma x \in A)\, B(x)}$$

是可导出的。 这是第二个Π-等式规则的类比，只要Π-规则按照相同的模式进行推导，也可以推导出来。 根据投影定律， $\mathsf{p}((x,y)) = x \in A$和 $\mathsf{q}((x,y)) = y \in B(x)$。 然后，通过Σ-引入（相等元素形成相等对），

$$(\mathsf{p}((x,y)), \mathsf{q}((x,y))) = (x,y) \in (\Sigma x \in A)\, B(x)。$$

通过 I-引入，

$$\mathsf{r} \in I((\Sigma x \in A)\, B(x), (\mathsf{p}((x,y)), \mathsf{q}((x,y))), (x,y)).$$

现在将家族 $C(z)$ 在Σ-消除规则中取为 $I((\Sigma x \in A)\, B(x),$
$(\mathsf{p}(z), \mathsf{q}(z)), z)$. 然后我们得到

$$\mathsf{E}(c, (x,y)\, \mathsf{r}) \in I((\Sigma x \in A)\, B(x), (\mathsf{p}(c), \mathsf{q}(c)), c)$$

因此，根据 I-消除规则，$(\mathsf{p}(c), \mathsf{q}(c)) = c \in (\Sigma x \in A)\, B(x).$

$$\dfrac{c \in (\Sigma x \in A)\, B(x) \qquad \dfrac{\dfrac{(x \in A) \qquad (y \in B(x))}{\mathsf{p}((x,y)) = x \in A} \qquad \dfrac{(x \in A) \qquad (y \in B(x))}{\mathsf{q}((x,y)) = y \in B(x)}}{(\mathsf{p}((x,y)), \mathsf{q}((x,y))) = (x,y) \in (\Sigma x \in A)\, B(x)}}{\mathsf{r} \in I((\Sigma x \in A)\, B(x), (\mathsf{p}((x,y)), \mathsf{q}((x,y))), (x,y))}$$
$$\dfrac{\mathsf{E}(c, (x,y)\, \mathsf{r}) \in I((\Sigma x \in A)\, B(x), (\mathsf{p}(c), \mathsf{q}(c)), c)}{(\mathsf{p}(c), \mathsf{q}(c)) = c \in (\Sigma x \in A)\, B(x)}$$

这个例子是典型的。I规则被系统地用来展示一个函数的唯一性，该函数的存在由消除规则给出，其属性由相关的等式规则表示。

例子（属性和索引元素族）。 有两种看待集合 B的子集的方式：

(1) B的子集是一个命题函数（属性）C（$y \in B$）；

(2) B的子集是一个元素的索引族b（$x \in B$（$x \in A$））。使用恒等规则，

我们可以证明这两个概念的等价性。
给定一个如（2）所示的索引族，相应的属性是

$$(\exists x \in A)\, I(B, b(x), y) \quad (y \in B),$$

反之，给定一个如（1）所示的属性，相应的索引族是

$$\mathsf{p}(x) \in B \ (x \in (\Sigma y \in B) C(y))。$$

有限集合

注意，到目前为止，我们没有从无中建立集合的操作，而只有从给定的集合（和集合族）中获得新集合的操作。
我们现在介绍有限集合，它们是直接给出的；因此它们的集合形成规则没有前提条件。 实际上，我们有无限多个规则，每个规则组对应一个$n= 0, 1, \ldots$

$\mathbb{N}_n$-形成

$$\mathbb{N}_n\text{集合}$$

$\mathbb{N}_n$-引入

$$m_n \in \mathbb{N}_n \qquad (m = 0,\, 1,\, \ldots,\, n-1)$$

所以我们有集合 $\mathbb{N}_0$没有元素，集合 $\mathbb{N}_1$有单一的规范元素0_1，
集合$\mathbb{N}_2$有规范元素0_2，1_2等等。

$\mathbb{N}_n$-消除

$$\frac{c \in \mathbb{N}_n \qquad c_m \in C(m_n)\ (m = 0,\, 1,\, \ldots,\, n-1)}{\mathsf{R}_n(c,\, c_0,\, \ldots,\, c_{n-1}) \in C(c)}$$

在这里，像往常一样，集合族C（z）（其中z∈$\mathbb{N}$）可以被解释为$\mathbb{N}$上的一个属性。 假设我们知道前提，R_n的解释如下：首先执行c，其结果是m，其中m在0和$_n$-1之间。 选择相应的元素cm来执行C（m，$_n$），然后继续执行它。结果是一个规范元素d∈C（c），因为c已被证明等于m，并且cm∈C（m，$_n$）是一个前提。R_n是对有限集合$\mathbb{N}$的递归；它是一种按情况定义的方式。 根据上述解释给出的R_n的含义，我们有以下规则（注意m∈$\mathbb{N}_n$由$\mathbb{N}_n$引入）：

$\mathbb{N}_n$-等式

$$\frac{c_m \in C(m_n)\ (m = 0,\, 1,\, \ldots,\, n-1)}{\mathsf{R}_n(m_n,\, c_0,\, \ldots,\, c_{n-1}) = c_m \in C(m_n)}$$

（对于结论中每个选择的 $m= 0, 1, \ldots,\, n-1$，都有一个这样的规则）。 另一种方法是假设仅对于 n等于0和1的规则，定义 $\mathbb{N}_2 \equiv \mathbb{N}_1 + \mathbb{N}_1$， $\mathbb{N}_3 \equiv \mathbb{N}_1 + \mathbb{N}_2$ 等等，然后推导出所有其他规则。

例子（关于 $\mathbb{N}_0$）。 $\mathbb{N}_0$没有引入规则，因此没有元素；因此自然地将其定义为

$$\bot \;\equiv\; \emptyset \;\equiv\; \mathbb{N}_0\text{。}$$

消除规则变得简单

$\mathbb{N}_0$-消除

$$\frac{c \in \mathbb{N}_0}{\mathsf{R}_0(c) \in C(c)}$$

规则的解释是，我们明白我们永远不会得到一个元素 $c \in \mathbb{N}_0$，所以我们永远不需要执行 $\mathsf{R}_0(c)$。 因此，执行形式为 $\mathsf{R}_0(c)$的程序的指令集是空的。 这类似于Dijkstra引入的编程语句 $abort$[12]。当 $C(z)$不依赖于 z时，可以在结论和前提中都省略证明（构造）。 然后我们得到了逻辑推理规则

$\bot$-消除

$$\frac{\bot\text{真}}{C\text{成立}}$$

传统上称为$ex\ falso\ quodlibet$。 这个规则经常在普通数学中使用，但是以以下形式

$$\frac{A \vee B\text{为真。} \qquad \begin{array}{c}(B\ \text{真}\quad)\\ \bot\text{真}\end{array}}{A\text{为真。}}$$

很容易看出这与上述形式等价。

例子（关于 $\mathbb{N}_1$）。我们定义

$$\top \ \equiv \ \mathbb{N}_1.$$

那么 0_1是一个（规范的）证明 $\top$，因为 $0 \quad _1 \in \mathbb{N}_1$通过 $\mathbb{N}_1$-引入。所以 $\top$是真的。 我们现在想要证明 0_1实际上是 $\mathbb{N}_1$的唯一元素，也就是说，这个规则

$$\frac{c \in \mathbb{N}_1}{c = 0_1 \in \mathbb{N}_1}$$

是可导出的。 事实上，从 $0_1 \ \in \ \mathbb{N}_1$，我们得到 $0_1 = 0_1 \ \in \ \mathbb{N}_1$，因此 $\mathsf{r} \ \in I(\mathbb{N}_1, 0_1, 0_1)$。 现在应用 $\mathbb{N}_1$-消除，对于集合族 $C(z)\ (z \ \in \mathbb{N}_1)$，使用 $I(\ \mathbb{N}_1, z, 0_1)\ (z \ \in \ \mathbb{N}_1)$。 利用假设 $c \in \mathbb{N}_1$，我们得到 $\mathsf{R}_1(c, \ \mathsf{r}) \ \in I(\mathbb{N}_1, c, 0_1)$，因此 $c = 0_1 \ \in \mathbb{N}_1$。反过来，通过进行定义 $\mathsf{R}_{\ 1}(c, c_0) \equiv c_0$，从规则 $\mathbb{N}_1$-消除可以推导出

$$\frac{c \in \mathbb{N}_1}{c = 0_1 \in \mathbb{N}_1}$$

而规则 $\mathbb{N}_1$-相等性变得平凡。 因此，操作 R_1可以被省略。

[12] 见注2。

例子（关于 $\mathbb{N}_2$）。我们给出定义

$$\text{布尔} \equiv \mathbb{N}_2。$$

布尔是编程中使用的类型，由两个真值 true和 false组成。因此，我们可以将 true $\equiv 0_2$和 false $\equiv 1_2$。然后我们可以定义如果 c则 c_0否则 $c_1 \equiv \mathsf{R}_2(c, c_0, c_1)$，因为如果 c是 true，即 c产生0_2，那么 $\mathsf{R}_2(c, c_0, c_1)$ 的值与 c_0相同；否则 c产生1_2， $\mathsf{R}_2(c, c_0, c_1)$ 的值与 c_1相同。

至于 $\mathbb{N}_1$上面，我们可以证明 $\mathbb{N}_2$的任何元素要么是0_2要么是1_2，但显然只能以命题形式存在

$$\frac{c \in \mathbb{N}_2}{\text{我}(\mathbb{N}_2，\ c，\ 0_2) \vee \text{我}(\mathbb{N}_2，\ c，\ 1_2)\text{真}}$$

例子（否定）。如果我们放置

$$\sim A \ \equiv \ \neg A \ \equiv \ -A \ \equiv \ A \to \mathbb{N}_0$$

我们可以轻松推导出所有常规否定规则。

一致性

关于我们的规则系统的一致性，我们能说些什么？ 我们可以从两个不同的角度理解一致性：

（1）元数学一致性。 然后，为了在数学上证明一个理论 T的一致性，我们考虑另一个理论 T'，它包含原始理论 T的命题 $codes$和一个谓词 Der，使得 $Der(\text{`}A\text{'})$表示命题 $A$的代码`$A$'在 T中是可导出的。 然后，我们定义 $Cons \equiv \neg Der(\text{`}\bot\text{'}) \equiv Der(\text{`}\bot\text{'}) \supset \bot$，并（尝试）证明 $Cons$在 T'中为真。 当像希尔伯特一样，我们放弃了对公理和推理规则的语义证明的希望时，这种方法是唯一适用的；它也可以成功地应用于直观类型论，但是由于我们对其语义和语法一样细致入微，我们不需要它。相反，我们直接通过以下简单的方式使自己相信它的一致性。

(2) 简单的一致性。 这意味着简单地说，$\bot$不能被证明，或者我们将永远没有权利判断$\bot$为真（与上面的命题Cons不同，它不是一个数学命题）。 为了使自己相信这一点，我们可以这样论证：如果$c \in \bot$对于某个元素（构造）c成立，那么c将产生一个规范元素$d \in \bot$；但是根据定义，这是不可能的（回想一下我们定义了$\bot \equiv \mathbb{N}_0$）。 因此，$\bot$为真不能通过正确规则的系统来证明。 因此，如果我们找到了$\bot$为真的证明，我们将知道证明中一定存在错误；而且，如果找到了$\bot$为真的形式化证明，那么其中至少有一个形式化规则是不正确的。 反思每个的含义

根据直觉型理论的规则，我们最终会相信它们是正确的；因此，我们将永远无法使用它们找到一个证明⊥为真。

最后，需要注意的是，无论如何，我们都必须依赖于至少在其中证明了Cons的简单一致性的理论T'，以获得从原始理论T的元数学一致性得到的简单一致性（这是我们真正关心的一致性形式）。 实际上，一旦证明了某个$c \in C\text{ons}$，我们必须按照以下方式进行论证：如果T不一致，那么我们将在T中有一个⊥为真的证明，或者对于某个$a \in \mathbb{N}_0$，有一个$a \in \mathbb{N}_0$的证明。 通过编码，这将给出'a'$\in D$er('⊥')；然后我们将得到Ap(c,'a')$\in \perp$，即⊥为真在T'中是可导出的。 在这一点上，为了得出⊥为真在T中不可证明的结论，我们必须确信⊥为真在T'中也不可证明。

自然数

到目前为止，我们还没有构造无限集合的方法。现在我们介绍最简单的一个，即自然数集合，通过以下规则：

$\mathbb{N}$-形成

$$\mathbb{N}\text{集合}$$

$\mathbb{N}$-引入

$$0 \in \mathbb{N} \qquad \frac{a \in \mathbb{N}}{a' \in \mathbb{N}}$$

需要注意的是，与任何其他引入规则一样， $a' \in \mathbb{N}$始终是规范的，无论元素 a是什么。因此， $a \in \mathbb{N}$表示 a的值可以是0或 a'_1，其中 a_1的值可以是0或 a'_2，依此类推，直到最终达到一个值为0的元素 a_n。

$\mathbb{N}$-消除

$$\frac{c \in \mathbb{N} \qquad d \in C(0) \qquad \begin{array}{c}\\ e(x,\, y) \in C(x')\end{array}}{\mathsf{R}(c,\, d,\, (x,\, y)\, e(x,\, y)) \in C(c)} \quad (x \in \mathbb{N},\, y \in C(x))$$

$\mathsf{R}(c, d,\ (x, y)\, e(x, y))$的解释如下：首先执行 c，得到 $\mathbb{N}$的一个规范元素，它可以是0或者某个 $\mathbb{N}$中的$'$。. 在第一种情况下，继续执行 d，得到 $C(0)$中的一个规范元素；但是，由于在这种情况下 $c = 0 \in \mathbb{N}$，所以 f也是 $C(c) = C(0)$的一个规范元素。 在第二种情况下，将$\mathsf{R}(a, d, (x, y)\, e(x, y))$（即前面的值）替换 tx和 y中的值，得到$(a, \mathsf{R}(a, d, (x, y)\ e(x, y)))$。 执行它，我们得到一个规范 f，根据右前提，它在 $C(\ a')$（因此也在 $C(c)$中，因为 $c = a' \in \mathbb{N}$）的假设下。 如果 ta的值为0，则 $\mathsf{R}(a, d, (x, y) e(x, y))$在 $C(\ a)$中，根据第一种情况。 否则，继续进行第二种情况，直到最终达到值0。 这个对消规则的解释也使得等式规则成立。

$\mathbb{N}$-等式

$$\frac{\begin{array}{cc} & (x \in A, y \in C\,(x)) \\ d \in C(0) & \neq (x, y) \in C\,(x') \end{array}}{\mathsf{R}(0, d, (x, y) \in (x, y)) = d \in C(0)}$$

$$\frac{\begin{array}{ccc} & & (x \in \mathbb{N},\, y \in C(x)) \\ a \in \mathbb{N} & d \in C(0) & e(x,\, y) \in C(x') \end{array}}{\mathsf{R}(a', d, (x, y) \in (x, y)) = (a, \mathsf{R}(a, d, (x, y) \in (x, y))) \in C(a')}$$

显然。 将$C(z)$ $(z \in \mathbb{N})$ 视为一个命题函数（性质），并在$\mathbb{N}$-消除规则的第二个和第三个前提以及结论中省略证明（构造），我们得到

数学归纳

$$\frac{\begin{array}{ccc} & & (x \in \mathbb{N},\, C(x) \text{ 真} \quad) \\ c \in \mathbb{N} & C(0) \text{ 真} & C(x') \text{ 真} \end{array}}{C(c) \text{ 真}}$$

如果我们明确地写出 $C(c)$ 的证明（构造），我们可以看到它是通过递归获得的。 因此，当命题被解释为集合时，递归和归纳实际上是相同的概念。

例子（前驱函数）。我们定义

$$\mathsf{pd}(a) \quad \equiv \quad \mathsf{R}(a, 0, (x,\, y)\, x)。$$

这个定义是通过计算 $\mathsf{R}(a, 0, (x, y)\, x)$ 来证明的：如果 $a\text{yields}\, 0$，那么$\mathsf{pd}(a)$ 也会得到0，如果 a yields b'，那么 $\mathsf{pd}(a)$ 得到的值与$\mathsf{R}(b', 0, (x, y)\, x)$ 相同，而 b 的值也相同。 所以我们有 $\mathsf{pd}(0) = 0$和 $\mathsf{pd}(a') = a$，这是通常的定义，但这里这些等式不是定义上的。更准确地说，我们有

$$\frac{a \in \mathbb{N}}{\mathsf{pd}(a) \in \mathbb{N}}$$

是 $\mathbb{N}$-消除的一个实例，以及

$$\begin{cases} \mathsf{pd}(0) = 0 \in \mathbb{N}, \\ \mathsf{pd}(a') = a \in \mathbb{N}, \end{cases}$$

通过 $\mathbb{N}$-等式我们得到

使用 pd，我们可以推导出第三个Peano公理

$$\frac{a' = b' \in \mathbb{N}}{a = b \in \mathbb{N}}$$

事实上，从 $a' = b' \in \mathbb{N}$，我们得到 $\mathsf{pd}(a') = \mathsf{pd}(b') \in \mathbb{N}$，再加上 $\mathsf{pd}(a') = a \in \mathbb{N}$ 和 $\mathsf{pd}(b') = b \in \mathbb{N}$，得到 $a = b \in \mathbb{N}$ 通过对称性和传递性。 我们也可以用通常的形式得到它（$\forall x,\, y$）（$x' = y' \supset x = y$），也就是说，在当前的符号体系中，

$$(\forall x \in \mathbb{N})\ (\forall y \in \mathbb{N})\ (I(\mathbb{N},\, x',\, y') \supset I(\mathbb{N},\, x,\, y))\text{为真}.$$

事实上，假设 $x \in \mathbb{N},\, y \in \mathbb{N}$和 $z \in I(\mathbb{N}, x', y')$。 通过 I-消去，我们得到 $x' = y' \in \mathbb{N}$；因此 $x = y \in \mathbb{N}$，由此可得$\mathsf{r} \in I(\mathbb{N}, x, y)$通过 I-引入。 然后，通过λ-抽象，我们得到 $(\lambda x)\,(\lambda y)\,(\lambda z)$ r是一个证明（构造）该命题的方法。

例子（加法）。我们定义

$$a + b \ \equiv\ \mathsf{R}(b,\, a,\, (x,\, y)\, y').$$

$a + b$的意义是对 a执行 b次后继操作。 然后，
很容易推导出以下规则：

$$\frac{a \in \mathbb{N} \qquad b \in \mathbb{N}}{a + b \in \mathbb{N}}$$

$$\frac{a \in \mathbb{N}}{a + 0 = a \in \mathbb{N}} \qquad \frac{a \in \mathbb{N} \qquad b \in \mathbb{N}}{a + b' = (a + b)' \in \mathbb{N}}$$

从中我们还可以推导出相应的一阶算术公理，就像前面的例子一样。 再次注意，这里的等号不是定义性的。

例子（乘法）。我们定义

$$a \cdot b \ \equiv\ \mathsf{R}(b, 0, (x,\, y)\ (y + a)).$$

乘积的常规属性 $a \cdot b$可以很容易地推导出来。

例子（有界 μ运算符）。 我们想要解决的问题是：给定一个布尔函数 f在自然数上，即 $f \in \mathbb{N} \to \mathbb{N}_2$，找到最小的参数 a（在界限$\mathbf{a} \in \mathbb{N}$下），使得 f的值为真。.解将是一个函数 μ（$\mathbf{x}$，f）$\in \mathbb{N}$（其中$x \in \mathbb{N}$， $f \in \mathbb{N} \to \mathbb{N}_2$），满足以下条件：

$$\mu(a,\, f) = \begin{cases} \text{最小的} b < a\text{，使得 } \mathsf{Ap}\,(f,\, b) = 0_2 \in \mathbb{N}\text{，如果存在这样的 } b \\ \text{，则为} a\text{，否则为。} \end{cases}$$

通过解递归方程获得这样的函数：

$$\begin{cases} \mu(0,\, f) = 0 \in \mathbb{N}, \\ \mu(a',\, f) = \mathsf{R}_2(\mathsf{Ap}(f, 0), 0, \mu(a, \overleftarrow{f}\,)') \in \mathbb{N}, \end{cases}$$

其中 $\overleftarrow{f} \equiv (\lambda x)\,\mathsf{Ap}(f, x')$ 是 f 向左移动一步，即 $\mathsf{Ap}(\overleftarrow{f}, x) = \mathsf{Ap}(f, x') \in \mathbb{N}_2\ (x \in \mathbb{N})$。事实上，当限制为零时，$\mu(0, f) = 0 \in \mathbb{N}$，无论函数 f是什么。当限制具有后继形式时，$\mu(a', f) = \mu(a, \overleftarrow{f})' \in \mathbb{N}$，前提是$(0) = \mathsf{false} \equiv 1_2 \in \mathbb{N}_2$；否则，$\mu(a', f) = 0 \in \mathbb{N}$。因此，要计算 $\mu(a, f)$，我们可以将 f向左移动直到限制为0，但每次都要检查值是否为 $\mathsf{true} \equiv 0_2$或 $\mathsf{false} \equiv 1_2$。即使它可以通过原始递归解决，但问题最容易通过高阶类型来解决，我们现在将详细看到。我们想要找到一个函数 $\mu(x) \in (\mathbb{N} \to \mathbb{N}_2) \to \mathbb{N}(x \in \mathbb{N})$，使得

$$\begin{cases} \mu(0) = (\lambda f)\ 0 \in (\mathbb{N} \to \mathbb{N}_2) \to \mathbb{N}, \\ \mu(a') = (\lambda f)\,\mathsf{R}_2(\mathsf{Ap}(f, 0), 0, \mathsf{Ap}(\mu(a), \overleftarrow{f})') \in (\mathbb{N} \to \mathbb{N}_2) \to \mathbb{N} \end{cases}$$

,所以我们可以通过将 $\mu(a, f) \equiv \mathsf{Ap}(\mu(a), f)$来定义我们要找的函数 $u(a, f)$。对于 $\mu(a)$的要求可以通过普通的原始递归来满足，但是在更高的类型上；这个任务通过$\mathbb{N}$-消除规则来完成。我们得到

$$\mu(a) \equiv \mathsf{R}(a, (\lambda f)\ 0, (x, y)\ (\lambda f)\,\mathsf{R}_2(\mathsf{Ap}(f, 0), 0, \mathsf{Ap}(y, \overleftarrow{f})')) \in (\mathbb{N} \to \mathbb{N}_2) \to \mathbb{N}$$

在前提条件下 $a \in \mathbb{N}$ 和 $f \in \mathbb{N} \to \mathbb{N}_2$, 因此

$$\mu(x, f) \in \mathbb{N} \qquad (x \in \mathbb{N},\ f \in \mathbb{N} \to \mathbb{N}_2).$$

将上述对 $\mu(a,\ f) \in \mathbb{N}$的证明以树形式写出如下:

$$\dfrac{\dfrac{a \in \mathbb{N} \quad \dfrac{0 \in \mathbb{N}}{(\lambda f)\ 0 \in (\mathbb{N} \to \mathbb{N}_2) \to \mathbb{N}} \quad \dfrac{\dfrac{(f \in \mathbb{N} \to \mathbb{N}_2) \quad 0 \in \mathbb{N}}{\mathsf{Ap}(f, 0) \in \mathbb{N}_2} \quad \dfrac{\dfrac{(y \in (\mathbb{N} \to \mathbb{N}_2) \to \mathbb{N}) \quad \overleftarrow{f} \in \mathbb{N} \to \mathbb{N}_2\ (f \in \mathbb{N} \to \mathbb{N}_2)}{\mathsf{Ap}(y, \overleftarrow{f}) \in \mathbb{N}}}{0 \in \mathbb{N} \qquad \mathsf{Ap}(y, \overleftarrow{f})' \in \mathbb{N}}}{\mathsf{R}_2(\mathsf{Ap}(f, 0), 0, \mathsf{Ap}(y, \overleftarrow{f})') \in \mathbb{N}}}{(\lambda f)\,\mathsf{R}_2(\mathsf{Ap}(f, 0), 0, \mathsf{Ap}(y, \overleftarrow{f})') \in (\mathbb{N} \to \mathbb{N}_2) \to \mathbb{N}}}{\mu(a) \equiv \mathsf{R}(a, (\lambda f)\ 0, (x, y)\ (\lambda f)\,\mathsf{R}_2(\mathsf{Ap}(f, 0), 0, \mathsf{Ap}(y, \overleftarrow{f})')) \in (\mathbb{N} \to \mathbb{N}_2) \to \mathbb{N} \qquad f \in \mathbb{N} \to \mathbb{N}_2}}{\mu(a,\ f) \equiv \mathsf{Ap}(\mu(a),\ f) \in \mathbb{N}}$$

观察 $\mu(a, f) \equiv \mathsf{Ap}(\mu(a), f) \equiv \mathsf{Ap}(\mathsf{R}(a, (\lambda f)\ 0, (x, y)\ (\lambda f)\,\mathsf{R}_2(\mathsf{Ap}(f, 0), 0, \mathsf{Ap}(y, \overleftarrow{f})')), f)$ 的求值过程。首先，评估ta。如果ta的值为0，则 $\mu(a, f)$ 的值等于 $\mathsf{Ap}((\lambda f)\ 0, f)$ 的值，即0。
另一方面，如果a的值是b'，则$\mu(a, \mathrm{f})$的值等于

$$\mathsf{Ap}((\lambda f)\mathsf{R}_2(\mathsf{Ap}(f,0),0, \mu(b, \overleftarrow{f})'),$$

$f)$，这又等于

$$\mathsf{R}_2(\mathsf{Ap}(f,0),0, \mu(b, \overleftarrow{f})')。$$

接下来，将评估$\mathsf{Ap}(f,0)$的值。如果$\mathsf{Ap}(f,0)$的值为$\mathrm{true} \equiv 0_2$，则$\mu(\mathrm{a}, f)$的值为0。另一方面，如果$\mathsf{Ap}(f,0)$的值为$f\mathrm{alse} \equiv 1_2$，则$\mu(\mathrm{a}, f)$的值等于$\mu(b, \overleftarrow{f})$' 的值。

列表

我们可以按照定义自然数的相同模式来引入其他归纳定义的集合。 我们在这里看到了列表的例子。

列表形成

$$\frac{\text{一个集合}}{\text{列表（}A\text{）集合}}$$

直观解释是：列表（A）是由集合A的元素组成的列表的集合（A的有限序列）。

列表引入

$$\text{nil} \in \text{列表（}A\text{）} \qquad \frac{a \in A \qquad b \in \text{列表（}A\text{）}}{(a.b) \in \text{列表（}A\text{）}}$$

我们也可以使用符号 $() \equiv \text{nil}$。

列表消除

$$\frac{c \in \text{列表（}A\text{）} \qquad d \in C(\text{nil}) \qquad \overset{(x \in A,\, y \in \text{列表（}A\text{）}\,,\, z \in C\,(y)\,)}{\neq (x, y, z) \in C((x.y))}}{\text{listrec}(c,\, d, (x,\, y,\, z)\, e(x,\, y,\, z)) \in C(c)}$$

其中C（z）（z∈$List$（A））是一族集合。 执行l$istrec$的指令是：首先执行c，如果得到n$\mathbb{i}$l，则继续执行d并获得f∈C（n$\mathbb{i}$l）=C（c），或者得到（a.b）其中a∈A且b∈$List$（A）；在这种情况下，执行e（a，b，l$istrec$（b，d，（x，y，z）e（x，y，z））））将得到一个规范元素f∈C（（a.b））=C（c）。 如果我们定义g（c）≡l$istrec$（c，d，（x，y，z）e（x，y，z）），那么f是e（a，b，g（b））的值。

列表-相等性

$$\frac{d \in C(\text{nil}) \qquad \overset{(x \in A,\, y \in \text{列表（}A\text{）}\,,\, z \in C\,(y)\,)}{\neq (x, y, z) \in C((x.y))}}{\text{列表递归}(\text{空列表}, d, (x, y, z)\text{属于 } C(\text{空列表})) = d\text{属于C}(\text{空列表})}$$

$$\frac{a \in A \qquad b \in \text{列表（}A\text{）} \qquad d \in C(\text{nil}) \qquad \overset{(x \in A,\, y \in \text{列表（}A\text{）}\,,\, z \in C\,(y)\,)}{\neq (x, y, z) \in C((x.y))}}{\begin{array}{l}\text{列表递归}((a.b), d, (x, y, z)\text{属于C}(\text{空列表})) \\ \quad = e(a,\, b, \text{列表递归}(b,\, d, (x,\, y,\, z)\text{属于}(x,\, y,\, z)))\text{属于 } C\,((a.b))\end{array}}$$

类似的规则可以用于有限树和其他归纳定义的概念。

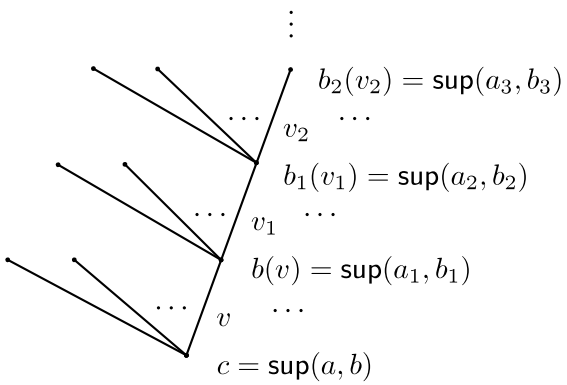
良序

良序的概念和超限归纳原理最初由康托尔引入。然而，一旦它们在ZF中被形式化，它们失去了原始的计算内容。 我们可以直观地将序数构造为良序树，这意味着它们不再是完全有序的。

$\mathcal{W}$-形成

$$\frac{\text{一个集合} \qquad (x \in A)\ B(x)\text{集合}}{(\mathcal{W}x \in A)\, B(x)\text{集合}}$$

对于 c是$(\mathcal{W}x \in A)\, B(x)$的元素意味着什么？ 这意味着，当计算 c时，会得到形式为 $\mathsf{sup}(a, b)$的值，其中 a和 b是某些值，其中$a \in A$， b是一个函数，对于任何选择的元素 $v \in B(a)$，b应用于 v会得到一个值 $\mathsf{sup}(a_1, b_1)$，其中 $a_1 \in A$ ， b_1是一个函数，对于任何选择的元素 v_1在 $B(a_1)$中， b_1应用于 v_1会得到一个值 $\mathsf{sup}(a_2, b_2)$，等等，直到在任何情况下（即无论如何进行连续选择），最终达到形式为 $\mathsf{sup}(a_n, b_n)$的底部元素，其中 $B(a_n)$为空，因此无法选择 $B(a_n)$中的任何元素。 下面的图片，我们松散地写作 $b(v)$表示 $\mathsf{Ap}(b, v)$，可以帮助理解（从下到上看）：



通过前面的解释，引入规范元素的以下规则是合理的：

$\mathcal{W}$-引入

$$\frac{a \in A \qquad b \in B(a) \to (\mathcal{W}x \in A)\, B(x)}{\mathsf{sup}(a,\, b) \in (\mathcal{W}x \in A)\, B(x)}$$

将$\mathsf{sup}(a, b)$视为序数$b(v)$的最小上界（大于所有序数$b(\ v)$的序数），其中v在$B(\ a)$上变化。

我们也可以有一个底部子句，例如$0 \in (\mathcal{W}x \in A)\, B(x)$，但是我们通过将$B(\ x)$中的一个集合（满足t$x \in A$）设置为空集来获得$0$：如果

$a_0 \in A$且 $B(a_0) = \mathbb{N}_0$，则 $\mathsf{R}_0(y) \in (\mathcal{W}x \in A)\, B(x)$（其中$y \in B(a_0)$），因此$\mathsf{sup}(a_0, (\lambda y)\, \mathsf{R}_0(y)) \in (\mathcal{W}x \in A)\, B(x)$是一个底部元素。

从对 $(\mathcal{W}x \in A)\, B(x)$ 元素的解释中，我们可以看到消除规则的正确性，这同时也是超穷归纳和超穷递归。超穷归纳的适当原则是：如果性质 $C(w)$ $(w \in (\mathcal{W}x \in A)\, B(x))$ 是归纳的（即，如果它对所有前驱 $\mathsf{Ap}(b, v)$ 都成立）

$\in (\mathcal{W}x \in A)\, B(x)\, (v \in B(a))$ 的一个元素 $\mathsf{sup}(a, b)$ 的性质 $C(c)$ 对于任意元素 $c \in (\mathcal{W}x \in A)\, B(x)$ 都成立。稍微正式一点说，

$$\frac{c \in (\mathcal{W}x \in A)\, B(x) \qquad (\forall x \in A)\, (\forall y \in B(x) \to (\mathcal{W}x \in A)\, B(x)) \qquad ((\forall v \in B(x))\, C(\mathsf{Ap}(y, v)) \supset C(\mathsf{sup}(x, y)))\ true}{C(c)\ 真}$$

现在我们解决这个问题，得到 $\mathcal{W}$-消除规则。其中一个前提是
如果 x，则 $C(\mathsf{sup}(x, y))$ 为真 $\in$ 令 $d(x, y, z)$ 为一个函数，它根据x给出$C(\mathsf{sup}(x, y))$ 的证明

$\in A,\ y \in B(x) \to (\mathcal{W}x \in A)\, B(x)$ 和证明 $z \to (\forall v \in B(x))\, C(\mathsf{Ap}(y, v))$，我们得到规则

$\mathcal{W}$-消除

$$\frac{c \in (\mathcal{W}x \in A)\, B(x) \qquad (x \in A,\ y \in B(x) \to (\mathcal{W}x \in A)\, B(x),\ z \in (\Pi v \in B(x))\, C(\mathsf{Ap}(y, v))) \qquad d(x, y, z) \in C(\mathsf{sup}(x, y))}{\mathsf{T}(c, (x, y, z)\, d(x, y, z)) \in C(c)}$$

其中 $\mathsf{T}(c, (x, y, z)\, d(x, y, z))$ 执行如下。首先执行 c，得到 $\mathsf{sup}(a, b)$，其中 $a \in A$和 $b \in$ 选择组件 a和 b并将它们替换为 x和 y在 d中，得到 $d(a,\ b,\ z)$。
现在我们必须替换 z为先前函数值的整个序列。
这个序列是
$(\lambda v)\, \mathsf{T}(\mathsf{Ap}(b, v), (x, y, z)\, d(x, y, z))$，因为 $\mathsf{Ap}(b, v)$ $\in (\mathcal{W}x \in A)\, B(x)\, (v \in B(a))$是枚举 $\mathsf{sup}(a, b)$的子树（前驱）的函数。然后

$$d(a,\ b,\ (\lambda v)\, \mathsf{T}(\mathsf{Ap}(b,\ v), (x,\ y,\ z)\, d(x,\ y,\ z)))$$

在假设下，产生一个规范元素 $e \in C(c)$ 作为值

$$\mathsf{T}(\mathsf{Ap}(b,\ v), (x,\ y,\ z)\, d(x,\ y,\ z)) \in C(\mathsf{Ap}(b,\ v))\ (v \in B(a)).$$

如果我们写 $f(c) \equiv \mathsf{T}(c, (x,\ y,\ z)\, d(x,\ y,\ z))$，那么当 c yields $\mathsf{sup}(a,\ b)$时，$f(c)$ yields 与 $d(a,\ b,\ (\lambda v)\, f(\mathsf{Ap}(b,\ v)))$ 相同的值。这个解释还显示
规则

$\mathcal{W}$-等式

$$\frac{\begin{array}{l}(x \in A,\, y \in B(x) \to (\mathcal{W}x \in A)\, B(x),\, z \in (\Pi v \in B(x))\, C(\mathsf{Ap}(y,\, v))) \\ a \in A \quad b \in B(a) \to (\mathcal{W}x \in A)\, B(x) \quad d(x,\, y,\, z) \in C(\mathsf{sup}(x,\, y))\end{array}}{\begin{array}{l}\mathsf{T}(\mathsf{sup}(a,\, b), (x,\, y,\, z)\, d(x,\, y,\, z)) \\ \quad = d(a,\, b,\, (\lambda v)\, \mathsf{T}(\mathsf{Ap}(b,\, v), (x,\, y,\, z)\, d(x,\, y,\, z))) \in C(\mathsf{sup}(a,\, b))\end{array}}$$

is correct.

例子（第一个数类）。 拥有 $\mathcal{W}$操作和
一族集合 $B(x)$ $(x \in \mathbb{N}_2)$ 使得 $B(0_2) = \mathbb{N}_0$和 $B(1_2) = \mathbb{N}_1$，我们
可以将第一个数类定义为 $(\mathcal{W}x \in \mathbb{N}_2)\, B(x)$ 而不是将其作为
原始的。

例子（第二数类）。 我们在这里给出了一个简单序数集的规则，即第二数类的所有序数集合 $\mathcal{O}$，并展示了它们如何作为良序规则的实例获得。

$\mathcal{O}$-形成

$\mathcal{O}$集合

康托尔通过应用以下两个原则从初始序数0生成第二数类：

（1）给定 $\alpha \in \mathcal{O}$，形成后继 $\alpha' \in \mathcal{O}$；

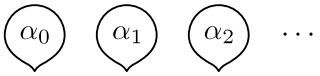
(2) 给定一个序列的序数 $\alpha_0,\, \alpha_1,\, \alpha_2, \ldots$ 在 $\mathcal{O}$中，形成比序列中每个元素都大的最小序数。

我们可以给出图片：

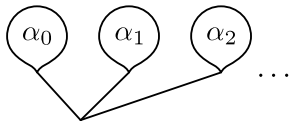
(1) 如果

在 $\mathcal{O}$中，则我们可以构建后继 α'：

(2) 如果



是一个在 $\mathcal{O}$中的序数序列，则我们可以构建上确界 $\mathsf{sup}_n(\alpha_n)$：



因此 $\mathcal{O}$将通过三个规则归纳定义：

$\mathcal{O}$-引入

$$0 \in \mathcal{O} \quad \frac{a \in \mathcal{O}}{a' \in \mathcal{O}} \quad \frac{b \in \mathbb{N} \to \mathcal{O}}{\mathsf{sup}(b) \in \mathcal{O}}$$

对 $\mathcal{O}$的超穷归纳显然成立，并且它由以下给出：

$$\frac{c \in \mathcal{O} \quad C(0)\text{为真} \quad (x \in \mathcal{O},\, C(x)\text{为真}) \quad C(x')\text{为真} \quad (z \in \mathbb{N} \to \mathcal{O},\, (\forall n \in \mathbb{N})\, C(\mathsf{Ap}(z,\, n))\text{为真}) \quad C(\mathsf{sup}(z))\text{为真}}{C(c)\text{ 真}}$$

其中 $C(z)$ $(z \in \mathcal{O})$ 是对 $\mathcal{O}$的性质。 用证明写出来，我们得到

$\mathcal{O}$-消除

$$\frac{c \in \mathcal{O} \quad d \in C(0) \quad (x \in \mathcal{O},\, y \in C(x)) \quad e(x,\, y) \in C(x') \quad (z \in \mathbb{N} \to \mathcal{O},\, w \in (\Pi n \in \mathbb{N})\, C(\mathsf{Ap}(z,\, n))) \quad f(z,\, w) \in C(\mathsf{sup}(z))}{\mathsf{T}(c,\, d, (x,\, y)\, e(x,\, y), (z,\, w)\, f(z,\, w)) \in C(c)}$$

其中，无穷递归运算符 T的执行如下。首先，执行 c。 我们区分三种可能情况：

如果我们得到$0 \in \mathcal{O}$，那么 $\mathsf{T}(c, d, (x, y)\, e(x, y), (z, w)\, f(z, w))$的值是 $d \in C(0)$的值；

如果我们得到 a', 那么该值就是

$$e(a,\, \mathsf{T}(a,\, d, (x,\, y)\, e(x,\, y), (z,\, w)\, f(z,\, w)));\text{的值。}$$

如果我们得到 $\mathsf{sup}(b)$, 我们继续执行

$$f(b,\, (\lambda x)\, \mathsf{T}(\mathsf{Ap}(b,\, x),\, d, (x,\, y)\, e(x,\, y), (z,\, w)\, f(z,\, w)))。$$

无论如何，我们得到了 $C(c)$的规范元素作为结果。

现在很容易检查，我们可以通过将
O ≡ ($\mathcal{W}$x∈N3)B(x)来获得所有的O-规则（包括未明确说明的O-

equality）。

其中B（x）（$x \in \mathbb{N}_3$）是一组集合，使得B（$_{03}$）$=\mathbb{N}_0$，B（$_{13}$）$=\mathbb{N}_1$，B（2_3）$=\mathbb{N}$。 这样的一组集合可以通过宇宙规则构建。

例子（良序的初始元素）。 我们想要证明，如果至少一个索引集为空，则良序（$\mathcal{W}x{\in}A$）B（x）非空。
回想一下，我们想要以直观的方式做到这一点，并且回想一下，$A\,\mathrm{true}$等价于$A\,\mathrm{none}mpty$，因此$\neg A\,\mathrm{true}$等价于$A\,\mathrm{e}mpty$。所以我们的断言是：

$$(\exists x{\in}A)\ \neg B\ (x)\ \rightarrow\ (\mathcal{W}x{\in}A)\ B\ (x)\ true。$$

为了看到这一点，假设 $x \in A$，$y \in \neg B(x)$ 和 $v \in B(x)$。 然后，$Ap(y, v) \in \mathbb{N}_0 \equiv \bot$，因此 $\mathsf{R}_0(Ap(y, v)) \in (\mathcal{W}x \in A)B(x)$，应用 $\mathbb{N}_0$-消除规则。现在我们对 v 进行抽象，得到 $(\lambda v)\mathsf{R}_0(Ap(y, v))$。 $\in B(x) \rightarrow (\mathcal{W}x \in A)B(x)$，并且通过 $\mathcal{W}$-引入，有 $\mathsf{sup}(x, (\lambda v)\mathsf{R}_0(Ap(\mathrm{y}, v)))$。 $\in$ 假设 $z \in (\Sigma x \in A)\neg B(x)$，通过 Σ-消除，我们有

$$\mathsf{E}(z, (x, y)\, \mathsf{sup}(x, (\lambda v)\, \mathsf{R}_0(\mathsf{Ap}(y, v)))) \in (\mathcal{W}x \in A)\, B(x),$$

通过 λ抽象在 z上

$$(\lambda z)\, \mathsf{E}(z, (x, y)\, \mathsf{sup}(x, (\lambda v)\, \mathsf{R}_0(\mathsf{Ap}(y, v)))) \in (\Sigma x \in A)\, \neg B(x) \rightarrow (\mathcal{W}x \in A)\, B(x).$$

现在我们想要证明一个逆命题。 然而，注意我们不能有$(\mathcal{W}x\ \in\ A)\, B(x)\ \rightarrow\ (\exists x\ \in\ A)\, \neg B(x)\ \ true$，因为存在量词的直观含义。但我们有：

$$(\mathcal{W}x \in A)\, B(x) \rightarrow \neg(\forall x \in A)\, B(x)\ \ true.$$

注意 $B(\ x) \rightarrow \mathbb{N}_0 \equiv (\Pi v \in B\ (x))\, C(\mathsf{Ap}(y, v))$ 对于 $C(w) \equiv \mathbb{N}_0$，因此我们可以应用$\mathcal{W}$-消除规则。 假设 $f \in (\Pi x \in A)\ B(x)$，我们有 $\mathsf{Ap}(f, x)\ {\in}B(x)$，因此也有 $\mathsf{Ap}(z, \mathsf{Ap}(f, x)) \in \mathbb{N}_0$。 $Ap(z,\ \mathsf{Ap}(f, x))$ 扮演了 $\mathcal{W}$-消除规则中的 $d(x, y, z)$ 的角色。因此，如果我们假设 w $\in (\mathcal{W}x \in A)\, B(x)$，我们得到 $\mathsf{T}(w, (x,\ y,\ z)\, \mathsf{Ap}(z, \mathsf{Ap}(f,\ x))) \in \mathbb{N}_0$。 对于 f的抽象，我们有

$$(\lambda f)\, \mathsf{T}(w, (x,\ y,\ z)\, \mathsf{Ap}(z, \mathsf{Ap}(f,\ x))) \in \neg(\forall x \in A)\, B(x),$$

并且，对于 w的抽象，我们有

$$(\lambda w)\ (\lambda f)\, \mathsf{T}(w, (x,\ y,\ z)\, \mathsf{Ap}(z, \mathsf{Ap}(f,\ x))) \in (\mathcal{W}x \in A)\, B(x) \rightarrow \neg(\forall x \in A)\, B(x)。$$

宇宙

到目前为止，我们只有有限类型的结构，因为我们只能从 $I(A, a, b), \mathbb{N}_0, \mathbb{N}_1, \ldots$ 开始迭代给定集合的形成操作 并且 $\mathbb{N}$有限次数。 为了加强语言，我们可以添加超限类型，在我们的语言中通过引入宇宙来获得。 回想一下，没有所有集合的集合，因为我们无法一次性展示所有可能的集合形成操作。 （所有集合的集合必须通过规定如何形成其规范元素，即集合来定义。但这是不可能的，因为我们总是可以完美地描述新的集合，例如，所有集合的集合本身。）然而，在范畴论中，我们需要集合的集合。

思路是将宇宙定义为在某些指定的集合形成操作下封闭的最小集合。到目前为止，我们使用的操作有：

$$\frac{\text{一个集合} \qquad (x \in A)\ B(x)\text{集合}}{(\Pi x \in A)\, B(x)\text{集合}} \qquad \frac{\text{一个集合} \qquad (x \in A)\ B(x)\text{集合}}{(\Sigma x \in A)\, B(x)\text{集合}} \qquad \frac{\text{一个集合} \qquad B\text{集合}}{A + B\text{集合}}$$

$$\frac{\text{一个集合} \qquad b,\, c \in A}{I(A,\, b,\, c)\text{集合}} \qquad \mathbb{N}_0\text{集合} \qquad \mathbb{N}_1\text{集合} \ldots \qquad \mathbb{N}\text{集合} \qquad \frac{\text{一个集合} \qquad (x \in A)\ B(x)\text{集合}}{(\mathcal{W} x \in A)\, B(x)\text{集合}}$$

有两种可能的构建宇宙的方式，即通过可能的超限迭代来获得闭包。

罗素式表述。考虑Π，Σ，. . . 既作为形成集合的操作，也作为形成集合 U 的规范元素的操作。 这就像分层类型理论中的情况一样。

塔斯基式表述。之所以这样称呼，是因为下面的家族 $T(x)\ (x \in U)$ 与塔斯基的真值定义之间的相似性。 我们使用新的符号，反映了Π，Σ，. . . ，来构建 U的规范元素。 然后 U由集合的索引组成（类似于递归理论）。 因此，我们将有以下规则：

U-形成

$$U\text{集合} \qquad \frac{a \in U}{T(a)\text{集合}}$$

U和 $T(x)\ (x \in U)$ 是通过同时超限归纳来定义的，通常可以从以下引入规则中读取：

U-引入

$$\frac{\begin{array}{c}(x \in T(a))\\ a \in U \qquad b(x) \in U\end{array}}{\pi(a, (x)\, b(x)) \in U} \qquad \frac{\begin{array}{c}(x \in T(a))\\ a \in U \qquad b(x) \in U\end{array}}{T(\pi(a, (x)\, b(x))) = (\Pi x \in T(a))\, T(b(x))}$$

$$\frac{\begin{array}{c}(x \in T(a))\\ a \in U \qquad b(x) \in U\end{array}}{\sigma(a, (x)\, b(x)) \in U} \qquad \frac{\begin{array}{c}(x \in T(a))\\ a \in U \qquad b(x) \in U\end{array}}{T(\sigma(a, (x)\, b(x))) = (\Sigma x \in T(a))\, T(b(x))}$$

$$\frac{a \in U \qquad b \in U}{a + b \in U} \qquad \frac{a \in U \qquad b \in U}{T(a + b) = T(a) + T(b)}$$

$$\frac{a \in U \qquad b \in T(a) \qquad c \in T(a)}{i(a,\, b,\, c) \in U} \qquad \frac{a \in U \qquad b \in T(a) \qquad c \in T(a)}{T(i(a,\, b,\, c)) = I(T(a),\, b,\, c)}$$

$$n_0 \in U \quad n_1 \in U \quad \ldots \qquad T(n_0) = \mathbb{N}_0 \quad T(n_1) = \mathbb{N}_1 \quad \ldots$$

$$n \in U \qquad T(n) = \mathbb{N}$$

$$\frac{\begin{array}{c}(x \in T(a))\\ a \in U \qquad b(x) \in U\end{array}}{w(a, (x)\, b(x)) \in U} \qquad \frac{\begin{array}{c}(x \in T(a))\\ a \in U \qquad b(x) \in U\end{array}}{T(w(a, (x)\, b(x))) = (\mathcal{W} x \in T(a))\, T(b(x))}$$

在这一点上，我们可以迭代这个过程，得到第二个宇宙 U'，带有两个新的引入规则：

$$u \in U' \qquad T'(u) = U$$

$$\frac{a \in U}{t(a) \in U'} \qquad \frac{a \in U}{T'(t(a)) = T(a)}$$

然后是第三个宇宙 U''，以此类推。

在罗素的表述中， T消失了，我们只使用大写字母。因此，上述规则变为：

U-形成

$$U\text{集合} \qquad \frac{A \in U}{\text{一个集合}}$$

U-引入

$$\frac{A \in U \qquad B(x) \in U \quad (x \in A)}{(\Pi x \in A)\, B(x) \in U} \qquad \frac{A \in U \qquad B(x) \in U \quad (x \in A)}{(\Sigma x \in A)\, B(x) \in U}$$

$$\frac{A \in U \qquad B \in U}{A + B \in U} \qquad \frac{A \in U \qquad b,\, c \in \multimap}{I(A,\, b,\, c) \in U}$$

$$\mathbb{N}_0 \in U \quad \mathbb{N}_1 \in U \quad \ldots \quad \mathbb{N} \in U$$

$$\frac{A \in U \qquad B(x) \in U \quad (x \in A)}{(\mathcal{W} x \in A)\, B(x) \in U}$$

然而， U本身不是 U的一个元素。 事实上，公理 $U \in U$导致了一个矛盾（吉拉尔悖论[13]）。 我们说一个集合 A是小的，或者是一个 U-集合，如果它有一个代码 $a \in U$，也就是说，存在一个元素 $a \in U$使得 $T(a) = A$。更一般地，一个族 $A(x_1, \ldots, x_n)\, (x_1 \in A_1, \ldots,\ x_n \in$ 一个$A(x_1, \ldots, x_{n-1}))$被称为小的，如果 $A(x_1, \ldots, x_n) = T(a(x_1, \ldots, x_n))\, (x_1 \in A_1, \ldots, x_n \in A_n(x_1, \ldots, x_{n-1}))$。因此，小集合的范畴在$\Sigma$、$\Pi$等运算下是封闭的。 U是一个完全合适的集合，但它不是小的。使用 U，我们可以构造超限类型（例如，使用值在 U中的递归）。

集合 V定义为 $(\forall\, x \in U)\, T(x)$（或者按照 Russell 的表述方式，简单地为 $(\forall\, X \in U)\, X$），Aczel 使用它来通过直观类型论给出构造性的 Zermelo-Fraenkel 集合论的含义。

例子（第四个 **Peano** 公理）。 我们现在想要证明第四个 **Peano** 公理，这是唯一一个不能从我们的规则中直接推导出来的。因此，要证明的命题是：

对于任意 $x \in \mathbb{N}$，$\neg\, I\,(\mathbb{N}, 0, x')$ 成立。

我们在证明中使用 U 规则；可能无法以其他方式证明它。
现在假设 $y \in I\,(\mathbb{N}, 0, x')$。 然后，根据 I 消去规则，得到 $0 = x' \in \mathbb{N}$。 根据 U 引入规则，有 $n\,0 \in U$ 和 $n_{\,1} \in U$。 然后我们定义 $f\,(a) \equiv \mathsf{R}\,(a, n\,0, (x, y)\, n_{\,1})$，这样 $f\,(0) = n\,0 \in U$，并且对于 $a \in \mathbb{N}$，$f\,(a') = n_{\,1} \in U$。 根据 $0 = x' \in \mathbb{N}$，通过 $\mathbb{N}$ 消去规则中的等式部分，得到 $\mathsf{R}\,(0, n\,0, (x, y)\, n_{\,1}) = \mathsf{R}\,(x', n\,0, (x, y)\, n_{\,1}) \in U$。 但是 $\mathsf{R}\,(0, n\,0, (x, y)\, n_{\,1}) = n\,0 \in U$，并且 $\mathsf{R}\,(0, n\,0, (x, y)\, n_{\,1}) = n_{\,1}$ $\in$ 根据$\mathbb{N}$-等式规则，U由U的规则决定。 因此，根据对称性和传递性，$n_0 = n_1 \in U$。 根据U的形成规则中的等式部分，$T(n_0) = T(n_1)$。因此，从

13 J. Y. Girard，函数解释和高阶算术的消除，Thèse，Université Paris VII，1972年。

14 P. Aczel，构造集合论的类型论解释，*Logic Colloquium 77*，由A. Macintyre，*L.* Pacholski和J. Paris编辑，North-Holland，Amsterdam，1978年，第55-66页。

由于$_{01} \in \mathbb{N}_1$，我们也有$_{01} \in \mathbb{N}_0$。因此，$(\lambda y)\,_{01} \in I(\mathbb{N},\,_0, x') \to \mathbb{N}_0$，并且$(\lambda x)\,(\lambda y\,)\,_{01} \in (\forall x \in \mathbb{N})\,\neg I(\mathbb{N},\,_0, x')$。

我们注意到，虽然很明显（通过反思其含义），
$0 = a' \in \mathbb{N}$是不可证明的，但证明 $\neg I$（$\mathbb{N}$， 0， a'）为真似乎涉及将
集合视为元素，以便定义一个在0上为 $\bot$，在
$\top$上为 a'的命题函数。