



数理逻辑

04 - Gödel不完备性定理简介

(Press ? for help, n and p for next and previous slide)

戴望州

南京大学智能科学与技术学院

2024年-春季

<https://daiwz.net>



Gödel 的不完备性定理做了什么？



GÖDEL的论文

On formally undecidable propositions of *Principia mathematica* and related systems I¹ (1931)

1

The development of mathematics toward greater precision has led, as is well known, to the formalization of large tracts of it, so that one can prove any theorem using nothing but a few mechanical rules. The most comprehensive formal systems that have been set up hitherto are the system of *Principia mathematica (PM)*² on the one hand and the Zermelo–Fraenkel axiom system of set theory (further developed by J. von Neumann)³ on the other. These two systems are so comprehensive that in them all methods of proof today used in mathematics are formalized, that is, reduced to a few axioms and rules of inference. One might therefore conjecture that these axioms and rules of inference are sufficient to decide *any mathematical question that can at all be formally expressed in these systems*. It will be shown below that *this is not the case*, that on the contrary there are in the two systems mentioned relatively simple problems in the theory of integers⁴ that cannot be decided on the basis of the axioms. This situa-

GÖDEL 的论文



1. 什么叫*formally undecidable propositions*?
2. 为什么是*Principia Mathematica*?
3. Part II在哪?



可判定与可计算

定义 3.1 (能行可判定, *effectively decidable*) :

一个性质 P (在某个论域 D 上定义) 是能行可判定的当且仅当存在一个算法, 在有穷步数内可判断任意 $o \in D$ 满足或不满足性质 P

- > 命题逻辑中的重言式是能行可判定的
 - » 枚举真值表
- > wff 中的主连词
 - » 对括号个数进行计数



可判定与可计算

定义 3.2 (能行可计算, *effectively computable*) :

一个函数 f (在某个定义域 D 上的全函数) 是能行可判定的当且仅当存在一个算法, 对任意 $o \in D$ 都可以在有穷步数内计算出 $f(o)$

- › 它比可判定性更加一般: 对于一个定义在 D 上的性质, 我们可定义一个特征函数 (*characteristic function*) c_P , 当 o 满足性质 P 时有 $c_P(o) = 1$, 否则 $c_P(o) = 0$
- › 可计算/可判定性的“算法”运行在一个“抽象的计算机”中, 我们不考虑它的物理运算速度或者内存大小, 唯一重要的是有穷步能输出



能行公理化的形式理论

定义 3.3 (能行形式化的语言, *effectively formalized language*) :

一个可解释的语言 L 是能行形式化的当且仅当它拥有:

1. 一个有穷的基本字符 (*basic symbols*) 集合
2. 能行可判定的语法性质 (*syntactic properties*) , 例如什么叫项 (*term*) 、合式公式 (*wff*) 、自由变元 (*free variable*) 、语句 (*sentence*) 等等; 且一个语句的语法结构是能行可决定的 (*effective determinable*) , 即在有穷步数内可解析
3. 上述句法结构与 L 中的语义规则可被用来能行地决定一个语句的唯一解释, 令其拥有一个固定的真值 (*fixed truth-value*)



能行公理化的形式理论

定义 3.4 (能行公理化的形式理论, *effectively axiomized formal theory*) :

一个能行公理化的形式理论 T 必须:

1. 基于一个能行形式化的语言 L
2. 一组能行可判定为公理的 L -wff
3. 一个能行可判定 wff 序列是否为 T 中的一个合法证明的证明系统

- > “理论” (*Theory*) 还有一个更一般的定义, 见 [Enderton, 155]
- > 接下来我们所有提到的“理论” (*theory*) 都是“能行公理化的形式理论”, 除非另外说明
- > 注意: 检验一个证明是否合法和判断是否存在不一样; “证明系统”也并不唯一:
 - » $\Sigma \vdash_S \varphi$ 表示证明系统 S 可以从 Σ 推导出 φ
 - » $T \vdash \varphi$ 表示形式理论 T 可以通过它自身的证明系统推导出 φ , 即 φ 是一个 T -定理 (*T -Theorem*)



形式可判定

定义 3.5 (形式可判定, *formally decides*) :

若 T 是一个理论, 且 φ 是该理论所使用的形式语言描述的一个语句, 那么我们称 T 形式可判定 φ 当且仅当: 要么 $T \vdash \varphi$, 要么 $T \vdash \neg\varphi$ 。显然, 一个语句 φ 在 T 中是形式不可判定的当且仅当 $T \not\vdash \varphi$ 且 $T \not\vdash \neg\varphi$

定义 3.6 (否定完备性, *negation completeness*) :

我们称 T 是一个否定完备 (也叫“语法完备”, *syntactical complete*) 的理论, 当且仅当它形式可判定它的语言描述的任意语句 (闭公式), 即对任意 φ , 要么 $T \vdash \varphi$, 要么 $T \vdash \neg\varphi$



算术、逻辑主义与 PRINCIPIA

1. 算术的基本语言和模型：0、后继、加法、乘法、逻辑连词、全称量词
2. 康德、先天综合判断
3. Gottlob Frege、分析哲学
4. 罗素、《数学原理》
 - » “All mathematics [yes! – all mathematics] deals exclusively with concepts definable in terms of a very small number of logical concepts, and ... all its propositions are deducible from a very small number of fundamental logical principles.”



“人类的绝望”

Gödel's First Theorem says – at a good enough approximation for now – that *nicely formalized theories containing enough arithmetic are always negation incomplete*. Given any nice effectively axiomatized formal theory T , there will be arithmetic truths that can't be proved in that particular theory.



哥德尔的证明思路



回顾

定义 3.6 (可靠性, *Soundness*) :

若 T 是一个理论, 那么它是可靠的当且仅当它的公理均为真 (基于 T 的语言及其上的解释) 且它的证明系统是保真 (*truth-preserving*) 的。也就是说, 它的定理都是真的

定义 3.7 (一致性, *Consistency*) :

若 T 是一个理论, 那么它是 (语法上) 可靠的当且仅当不存在任何 φ 令 $T \vdash \varphi$ 且 $T \vdash \neg\varphi$, 其中 “ \neg ” 是 T 中的否定词



基本算术理论的语言

定义 3.8 (基本算术理论的语言) :

一个可解释的形式化语言 L 包含基本算术语言 (*the language of basic arithmetic*) 当且仅当 L 有：

- › 一个表示零的项
- › 表示后继、加法、乘法的算术函数符号
- › 一般的逻辑连词、等词符号
- › 以及能够对自然数进行量化的量词符号



哥德尔第一不完备性定理概述

定理3.9（哥德尔不完备性定理I）：

若 T 是一个能行公理化的形式理论且它的语言包含基本算术理论。那么若 T 是可靠的，那么就存在一个基本算术理论中为真的语句 G_T ，令 $T \not\vdash G_T$ 且 $T \not\vdash \neg G_T$ 。

也就是说， T 是否定不完备的。

定理3.9'（哥德尔不完备性定理I'，纯语法版本）：

若 T 是一个能行公理化的形式理论且它的语言包含基本算术理论。那么若 T 是一致的且能够证明算术中最基础的某一小部分理论（以及一些常见形式算术理论中的额外性质），那么就存在一个基本算术理论中的语句 G_T ，令 $T \not\vdash G_T$ 且 $T \not\vdash \neg G_T$ 。

因此， T 是否定不完备的。



不完备性与不可完备性

我们考虑语义上的不完备性，既然 $T \not\vdash G_T$ 且 $T \not\vdash \neg G_T$ ，那我们为什么不将 G_T 加进去？即令

$$U = T \cup \{G_T\}$$

是否 U 就完备了呢？

我们考察 U 的性质：

1. 是否可靠？
2. 是否仍然是一个公理化的形式理论？
3. 其语言是否包含算术基本理论？



哥德尔的完备性和不完备性定理

若 T 是由某个 FOL 表达的算术理论，我们有一套一阶演绎系统 S （例如 Hilbert System, Gentzen's Sequent Calculous, etc.），那么根据哥德尔完备性定理，我们有：

$$T \vDash \varphi \Rightarrow T \vdash_S \varphi$$

根据哥德尔不完备性定理，我们有：存在一些对于理论 T 来说为真的 wff φ ，令

$$T \not\vDash \varphi \text{ 且 } T \not\vDash \neg\varphi$$

不仅如此，就算我们把这些 φ 加进 T ，仍然会出现其它的 φ' 导致这个新的形式理论不完备，即 T 永远不可完备



证明所使用的记号

定义3.IO（简写）： \bar{n} 表示自然数 n 在标准数字记法下的简写

- › 标准数字记法只有数字0和后继S函数。 $\bar{5}$ 即为数字 $5 = SSSSS0$ 的简写

定义3.II（含自由变元的公式）：

- › 语言 L 中的开公式（open wff） $\varphi(x)$ 用来表示一个数的性质 P ：对任意 n ， $\varphi(\bar{n})$ 为真，当且仅当 n 拥有性质 P
- › 类似地，开公式 $\psi(x, y)$ 用来表示一个数字间的二元关系 R ：对任意 m, n ， $\psi(\bar{m}, \bar{n})$ 为真，当且仅当 m, n 满足关系 R
- › 开公式 $\chi(x, y)$ 用来表示一个一元函数 f ：对任意 m, n ， $\chi(\bar{m}, \bar{n})$ 为真，当且仅当 $f(m) = n$



哥德尔数

如今，我们对各种数据都可以用数字编码这一事实已经完全熟悉了。

1931年，这个想法当然并不像今天这样普及。

但即便如此，以下这种定义当时也应该显得相当没有问题：

定义3.12（哥德尔数，*Gödel number*）：

一个形式理论 T 的哥德尔编码模式（Gödel number scheme），是一种将 T 的表达式（以及表达式序列）能行地编码为自然数的方法。该模式提供了一个将表达式（或表达式序列）转换为自然数的算法；同时也提供了一个解码算法，即将被编码的自然数转换回将它所编码的唯一表达式（或表达式序列）。

对于一个编码模式，一个表达式（或表达式序列）的哥德尔编码数字即是其唯一的哥德尔数。



哥德尔数

定义 3.13 (公式和证明的哥德尔编码) :

取一个能行公理化的形式理论 T , 并固定一个对 T 的语言中的表达式/表达式序列进行哥德尔编码的模式。那么, 相对于该编码模式, 我们可以定义以下性质/关系:

1. $Wff_T(n)$ 当且仅当 n 是 T -wff 的哥德尔数
2. $Sent_T(n)$ 当且仅当 n 是 T -句子 (*sentence*) 的哥德尔数
3. $Prf_T(m, n)$ 当且仅当 m 是一个 T -证明的哥德尔数, 它证明了哥德尔数 n 的 T -句子



哥德尔数

定理 3.14：

若 T 是一个能行公理化的形式理论，且有一个哥德尔编码模式。那么定义 3.13 中几种关于数字的性质 $Wff_T(n)$ 、 $Sent_T(n)$ 、 $Prf_T(m, n)$ 均为能行可判定的。

证明概要：

1. 对于 Wff_T 和 $Sent_T$ 两种性质，由于 n 解码为字符串是能行的，且字符串的语法判定也是能行的，所以这两种性质是能行可判定的
2. 对于 Prf_T ，由于一个能行公理化的形式理论的证明系统是能行可判定的，且证明序列中的每个哥德尔编码的解码和字符串解析也是能行的，因此该性质仍然是能行可判定的



T 可以表达 Prf_T

有了以上结论，我们就能够得到以下两个更关键的结论：我们能够在 T 中表达“可证”

定理 3.15：

若 T 是一个能行公理化的形式理论，且它能够表达基本算术理论。给定一个哥德尔编码模式。那么 T 就能表达关于数字的性质 Prf_T ，该性质能够被表达为一个算术 wff（即被编码的 T -wff） $\text{Prf}_T(x, y)$

这个结果并不是显而易见的，它的证明大概分两种方法：

1. 自底向上地构造：选取一个特殊的 T ，详细地构建一个基本算术公式来表达 Prf_T 关系。然后对它进行概括，以令它对任意满足定义要求的 T 都成立
2. 自顶向下地证明：通常来说，基本算术的语言有能力表达任何可判定的性质和关系。因为定理 3.14 告诉我们，当 T 是一个能行公理化的形式理论时，数值关系 Prf_T 是可判定的。所以我们能够将这个关于表达能力的性质应用于 Prf_T （哥德尔采用了这个版本）



T 中的“一个命题可证”

基于定理 3.15，我们能够构造如下关于“可证”在 T 中如何表达的定义：

定义 3.16 (可证明性谓词, *provability predicate*) :

定义 $\text{Prov}_T(x) \equiv \exists z \text{Prf}_T(z, x)$ (量词作用在自然数域上)。

那么, $\text{Prov}_T(\bar{n})$, 即 $\exists z \text{Prf}_T(z, \bar{n})$ 为真, 当且仅当某个数 z 恰好是哥德尔编码为 \bar{n} 的句子的一个 T -证明, 也即当且仅当哥德尔编码为 \bar{n} 的句子是 T 的一个定理。因此, Prov_T 被称为可证明性谓词。



哥德尔语句 G_T

接下来，哥德尔构造了一个奇特的语句：

定理 3.17（哥德尔语句，*Gödel sentence*）：

我们可以在基本算术语言中为理论 T 构造一个哥德尔语句 G_T ，它具有以下性质： G_T 为真当且仅当 $\neg \text{Prov}_T(\overline{\Gamma G_T \neg})$ 为真，其中 $\overline{\Gamma G_T \neg}$ 正是 G_T 的哥德尔编码

- › 通过构造可得， G_T 在解释时为真当且仅当 $\neg \text{Prov}_T(\overline{\Gamma G_T \neg})$ 也为真
- › 即当且仅当由 $\overline{\Gamma G_T \neg}$ 表示的哥德尔编码对应的公式不是 T 的定理
- › 即当且仅当 G_T 不是 T 的定理
- › 简而言之，定理 3.17 告诉我们可以找到一个算术语句 G_T ，它在且仅在它不是 T 的定理时才为真



不完备性定理的证明

定理 3.9 (哥德尔不完备性定理 I) :

若 T 是一个能行公理化的形式理论且它的语言包含基本算术理论。那么若 T 是可靠的，那么就存在一个基本算术理论中为真的语句 G_T ，令 $T \not\vdash G_T$ 且 $T \not\vdash \neg G_T$ 。

也就是说， T 是否定不完备的。

证明：

- > 设 G_T 为定理 3.17 中引入的哥德尔语句。假设 T 是可靠的并且 $T \vdash G_T$ ，即 G_T 是一个 T -定理。根据 G_T 的定义，它为假
- > 那么 T 中存在一个错误的定理，即 T 将不可靠，与假设相矛盾
- > 因此， T 不能证明 G_T 。即 $T \not\vdash G_T$
- > 再次根据它的定义，由于 G_T 当且仅当不可证时为真，所以 G_T 为真
- > 那么 $\neg G_T$ 为假。但由于 T 是可靠的，所以它不能证明 $\neg G_T$ 。因此，我们也有 $T \not\vdash \neg G_T$
- > 所以， T 无法形式化地判定 G_T 。根据定义 3.6， T 是否定不完全的

Q.E.D.



说谎者悖论和不完备性定理

看起来都是自指语句，为什么说谎者悖论是悖论，而哥德尔不完备性定理是定理？

- › 这个问题触及了不完备性定理的深层根源
- › 简而言之，假设 T 是一个能行公理化的理论，可以表达足够的算术理论。那么， T 就可以表达作为一个可证明的 T 句子的性质。
- › 然而在事实上， T 不能表达“什么叫做真的 T -语句”这一性质（如果它能够，那么 T 当然会陷入说谎者悖论）
- › 所以，“真的 T -语句”和“可证的 T -语句”是两种完全不同的性质
- › 因此，要么有在 T 为真但不可证的句子，要么有在 T 中为假但可证明句子
- › 假设中的“ T 是可靠的”排除了第二种情况，所以结论是 T 中的“真”超出了 T 中语法可证的定理范围。也就是说， T 不可能是否定完全的



不可判定性与对角化证明



能行可枚举

定义 3.18 (能行可枚举, *effectively enumerable*) :

一个集合 Σ 是能行可枚举的, 当且仅当存在一个算法能够逐步输出该集合元素的一个列表 s_1, s_2, s_3, \dots (允许重复), 使得: 对任意一个 Σ 中的元素, 当算法运行足够步数后它最终会出现在该列表中

定理 3.19:

一个能行公理化的理论 T 中的定理是能行可枚举的



能行可枚举

定理 3.19：

一个能行公理化的理论 T 中的定理是能行可枚举的

证明概要：

- I. 根据定义 3.3 和 3.4, T 的语言 L 只有有穷个基本符号。那么我们可以依字典序对所有的 L 字符串排序并输出
 - » (注意定义 3.18 中的“算法运行足够步数”并未要求“有限步数”)
2. 由于 L 中的 wff 都是可判定的，所以一定可以能行的判定哪些字符串是 wff (类别 a)、哪些是 wff 序列 (类别 b)、哪些这两者均不是 (类别 c)
3. 因为 T 是能行公理化的，那么就可以能行地判定一个 b 类字符串是否是一个 T -证明
4. 这些属于 T -证明的字符串中的最后一个 wff 就是 T -定理。这时我们将它加入输出队列
5. 该过程能够机械化地不断执行下去，因此 T 中的定理是能行可枚举的



否定完备性蕴涵可判定性

一个形式理论的可判定性就是它的所有定理的可判定性：

定义 3.20：

一个理论 T 是可判定的，当且仅当性质（谓词）“是 T 的定理”是能行可判定的，即对任意 L 中的语句 φ 存在一个机械的方式可以决定它是否为 T 的定理

注意区分下面的定义（主要区别在哪？）：

1. 一个理论 T 可形式化判定一个语句 φ ，当且仅当 $T \vdash \varphi$ 或 $T \vdash \neg\varphi$
2. T 是可判定的，当且仅当 L 中的任意语句均能被判定是否是 T 的定理
3. 一个理论可能本身是可判定的，但它却无法判定 L 中的所有语句。比如前面例子中 $\{p, q, r\}$ 语言中的理论 $\{p \wedge \neg r\}$



否定完备性蕴涵可判定性

定理 3.21：

任意一致的、否定完备的、能行公理化的形式理论都是可判定的

证明概要：

1. 根据定理 3.19，一个能行公理化的形式理论 T 中的定理都是能行可枚举的
2. 那么由于 T 是否定完备的，对任意 wff φ ，它或它的否定迟早会出现在以上枚举中
 - » 如果 φ 出现在枚举中，那么它是 T -定理
 - » 若 $\neg\varphi$ 出现在枚举中，根据 T 的一致性， φ 不是 T -定理



将公式性质转化为数值性质

注意到，前面所讨论的“性质”是用来描述 L 语言中的字符串，或wff的。未来我们要用哥德尔编码将wff以及证明转换为数，因此还要定义如何描述可判定的“数字的性质”

定义3.22（数值性质，*numerical propertiy*）：

数值性质 P 由含有基本算术的语言 L 中带有一个自由变量的开公式 $\varphi(x)$ 表达，它当且仅当对于每一个 n ，

1. 如果 n 具有性质 P ，那么 $\varphi(\bar{n})$ 为真
2. 如果 n 不具有性质 P ，那么 $\neg\varphi(\bar{n})$ 为真

下面，我们可以描述在形式理论 T 中可机械化判定的数值性质

定义3.23（刻画，*captures*）：

一个形式理论 T 可通过开公式 $\varphi(x)$ 刻画（captures）数值性质 P ，当且仅当对任意 n ，

1. 如果 n 具有性质 P ，那么 $T \vdash \varphi(\bar{n})$
2. 如果 n 不具有性质 P ，那么 $T \vdash \neg\varphi(\bar{n})$



我们希望的——“充分强的形式理论”

定义 3.24 (充分强的理论, *sufficiently strong theory*) :

我们称一个形式理论 T 是充分强的, 当且仅当它能够刻画任意能行可判定的数值性质

换言之, 它应该能做到以下两件事情:

1. 它的语言能够构建开放公式 $\varphi(x)$ 来表达可判定的数值性质 P
2. 若某个数字 n 有性质 P , 那么 $T \vdash \varphi(\bar{n})$; 否则 $T \vdash \neg\varphi(\bar{n})$



充分强的形式理论是不可判定的

下面，我们可以证明出这个比较强的结论（尽管和哥德尔不完备性定理有一点不一样）：

定理3.25：

任意一致的、能行公理化的、且充分强的形式理论都是不可判定的

证明（对角化）：

- › 下面使用反证法。我们假设 T 是一致的、能行公理化的、充分强的，且它是可判定的
- › 由于 T 是充分强的，因此它能够用开公式表达任意数值性质。由于它是能行公理化的，那么它的语言 L 必然可判定什么叫“包含 1 个自由变元的开公式”，因此可以仿照定理3.19证明这些开公式是可枚举的。令它们的枚举为：

$$\varphi_0(x), \varphi_1(x), \dots$$

- › 定义一个数值性质 D : n 具有性质 D 当且仅当 $T \vdash \neg\varphi_n(\bar{n})$
 - » 根据我们的假设， D 是能行可判定的：对任意 n 我们可以用有穷步枚举至 $\varphi_n(x)$ ，将其中的 x 替换为 \bar{n} （甚至在前面再加一个 \neg ）。由于我们设它是可判定的，那么就存在能行的算法证明 $T \vdash \varphi_n(x)$ 或 $T \vdash \neg\varphi_n(x)$



充分强的形式理论是不可判定的

证明（对角化，Cont'd）：

- > 由于我们假设 T 充分强，那么它能够刻画任意可判定的数值性质，也包括 D
- > 假设刻画性质 D 的开公式为 $\varphi_d(x)$ ，根据刻画的定义，它的意思是：
 1. 若 n 拥有性质 D ，则 $T \vdash \varphi_d(\bar{n})$
 2. 若 n 没有性质 D ，则 $T \vdash \neg\varphi_d(\bar{n})$
- > 令 $n = d$ ，得到
 1. 若 d 拥有性质 D ，则 $T \vdash \varphi_d(\bar{d})$
 2. 若 d 没有性质 D ，则 $T \vdash \neg\varphi_d(\bar{d})$
- > 然而根据上一页的红字定义： d 具有性质 D 当且仅当 $T \vdash \neg\varphi_d(\bar{d})$
 - » 与上面第 2 点结合得到：不管 d 是否拥有性质 D 均有 $T \vdash \neg\varphi_d(\bar{d})$ 。也就是说 $\neg\varphi_d(\bar{d})$ 是 T -定理
- > 因此 d 确实拥有性质 D ，从而根据上面的第 1 点得 $\varphi_d(\bar{d})$ 也是 T -定理
- > 以上结论与 T 是一致的矛盾，因此 T 是不可判定的

Q.E.D.



不可能三角（不完备性！）

定理3.21：

任意一致的、否定完备的、能行公理化的形式理论都是可判定的

定理3.25：

任意一致的、能行公理化的、且充分强的形式理论都是不可判定的

根据这两条定理，很快就能得到另一种形式的不完备性定理：

定理3.26：

任意一致的、能行公理化的、且充分强的形式理论都是否定不完备的



准备工作 I：一阶 Peano 算术



BABY ARITHMETIC

在定义一阶Peano算术前，我们先定义两类比较简单的算术，并考察它们分别能做什么。

第一种算术理论叫Baby Arithmetic，记为BA。它的语言为 L_B 。它的词汇表如下：

符号	说明
0	常元，表示零
$S, +, \times$	函词：后继、加法、乘法
=	谓词：等于
\neg, \rightarrow	逻辑连词
(,)	标点：括号

简写：

- > 项 $\bar{5}$ 表示 $SSSSS0$ ；项 $(\bar{1} + (\bar{2} \times \bar{3}))$ 表示 $S0 + SSS0 \times SSS0$
- > wff的例子： $\neg(\bar{1} + \bar{2} = \bar{2} \times \bar{2})$ ；或简写为 $\bar{1} + \bar{2} \neq \bar{2} \times \bar{2}$



BA 的公理

注意，由于BA没有变元和量词，它和命题逻辑中一样，每条公理都是一个模式。我们继续用希腊字母来作为元语言中的变元来描述它们。它的公理包括以下内容：

- > 命题逻辑公理以及关于等词的公理，例如Leibniz's Law: $\vdash \tau = \sigma \rightarrow \varphi(\tau) = \varphi(\sigma)$
- > 非逻辑公理：对任意数字 ζ, ξ 有
 1. $0 \neq S\zeta$
 2. $S\zeta = S\xi \rightarrow \zeta = \xi$
 3. $\zeta + 0 = \zeta$
 4. $\zeta + S\xi = S(\zeta + \xi)$
 5. $\zeta \times 0 = 0$
 6. $\zeta \times S\xi = (\zeta \times \xi) + \zeta$

尽管它的公理数量是无穷的，但它仍然是一个能行公理化的形式理论：给定一个候选公式，我们显然可以能行地判定它是否是那六个模式中的实例，从而确定它是否是一条公理



理论 BA

定义 3.27 (*Baby Arithmetic, BA*) :

BA是一个语言为 L_B 的形式理论，它包含关于否定词、蕴含词和等词的经典逻辑规则，且它的非逻辑公理包含以上模式I到6的所有数值实例

它能够证明许多定理，例如 $\text{BA} \vdash \bar{2} + \bar{2} = \bar{4}$ ，即 $\text{SS0} + \text{SS0} = \text{SSSS0}$

1. $\text{SS0} + 0 = \text{SS0}$ Ax 3
2. $\text{SS0} + \text{S0} = \text{S}(\text{SS0} + 0)$ Ax 4
3. $\text{SS0} + \text{S0} = \text{SSS0}$ 1, 2 Leibniz's Law (LL)
4. $\text{SS0} + \text{SS0} = \text{S}(\text{SS0} + \text{S0})$ Ax 4
5. $\text{SS0} + \text{SS0} = \text{SSSS0}$ 3, 4 LL



BA 的完备性

定理 3.28:

若 τ 是 L_B 语言中的一个项，且假设它在 BA 的解释中取值为 t ，那么 BA 必然能够在其内部证明这一点，即 $BA \vdash \tau = \bar{t}$

- > 也就是说，不管 τ 多么复杂，BA 总能计算出它的正确取值，比如
$$BA \vdash (\bar{2} + \bar{3}) \times (\bar{2} \times \bar{2}) = \bar{20}$$
- > 因为 BA 只有一个谓词（“等于”），所以如下几条“完备性定理”显然成立

定理 3.29:

1. 若算术等式 $\sigma = \tau$ 为真，那么 $BA \vdash \sigma = \tau$
2. 若 s 和 t 是不同的数字，那么 $BA \vdash \bar{s} \neq \bar{t}$
3. 若算术等式 $\sigma = \tau$ 为假，那么 $BA \vdash \sigma \neq \tau$



BA 的完备性

定理 3.30: BA 是一个可靠的（sound）能行公理化形式理论，且它是否定完备的

证明概要：

1. 可靠性显然成立。因为算术公理是正确的，且逻辑推导系统（命题逻辑）是可靠的（或“保真的”），因此推出的任意 BA-定理都是正确的
2. 下面证明其否定完备性（negation completeness）：
 - » L_B 中的 wff φ 由逻辑连词和原子公式 $\alpha_1, \alpha_2, \dots, \alpha_n$ 构成，其中每个原子公式都是算术等式。因此 BA 中的推理可通过考虑 α_i 本身的真值进行
 - » 令 v 是原子公式上的一个赋值，定义： α_i 为真时 α_i^v 表示 α_i ，否则它表示 $\neg\alpha_i$ 。同理， φ^v 当 φ 为真时表示 φ ，否则表示 $\neg\varphi$
 - » 显然有 $\alpha_1^v, \alpha_2^v, \dots, \alpha_n^v$ 重言蕴涵 φ^v 。根据命题逻辑（PL）的可靠性，我们有
$$\alpha_1^v, \alpha_2^v, \dots, \alpha_n^v \vdash_{PL} \varphi^v$$
 - » 根据定理 3.29，每个 α_i^v 在 BA 中都是可证的
 - » 再因为 BA 包含命题逻辑，所以上面对 φ^v 的证明（不论 φ 还是 $\neg\varphi$ 为真）均在 BA 中可表达。即，它是否定完备的



ROBINSON ARITHMETIC

接下来，我们引入量词定义一种FOL下的算术语言 L_A （Language of Arithmetics）。它的基本成分和 L_B 一致，只多了量词与变元

符号	说明
0	常元，表示零
$S, +, \times$	函词：后继、加法、乘法
=	谓词：等于
\neg, \rightarrow	逻辑连词
(,)	标点：括号
x_0, x_{S0}, \dots	变元
\forall	量词：量化自然数



ROBINSON ARITHMETIC (Q) 的公理

由于Q有变元和量词，所以可以直接在公理中表达变元，即它的非逻辑公理Q便有限了！

> FOL公理

> 非逻辑公理：

1. $\forall x (0 \neq Sx)$
2. $\forall x \forall y (Sx = Sy \rightarrow x = y)$
3. $\forall x (x \neq 0 \rightarrow \exists y (x = Sy))$

» 以上公理只说0不是后继，并未明确其他数字的性质。 L_B 中因为我们直接列出所有常量，所以不用操心该性质。现在有了变元和量词，因此有必要引入该公理

4. $\forall x (x + 0 = x)$
5. $\forall x \forall y (x + Sy = S(x + y))$
6. $\forall x (x \times 0 = 0)$
7. $\forall x \forall y (x \times Sy = (x \times y) + x)$



理论 Q

定义 3.31 (*Robinson Arithmetic, Q*) :

Robinson 算术 Q 是一个语言为 L_A 的形式理论，它包含以上公理 I-7 以及经典的 FOL

定理 3.32 (Q 包含 BA) :

Q 可正确地判定每一个无量词的 L_A 语句。换句话说，如果无量词 wff φ 为真，则 $Q \vdash \varphi$ ；如果无量词 wff φ 为假，则 $Q \vdash \neg\varphi$

定理 3.33 (Q 可以刻画“序”):

Robinson 算术中，“小于等于”关系可以用如下 wff 刻画：

$$\exists v(v + x = y)$$



Q是不完备的

Q看起来很强大，它能证明任意一个这样的公式： $0 + \bar{n} = \bar{n}$ ，比如 $0 + \bar{1} = \bar{1}$

但它无法证明该公式的概化（思考：FOL的常数概括定理失效了吗？），因此有如下定理：

定理3.34： $Q \not\vdash \forall x (0 + x = x)$

证明：

- > 由于Q包含经典FOL，因此对任意 L_A -语句 φ ，我们有 $Q \vdash \varphi \Rightarrow Q \models \varphi$ （FOL的可靠性）
- > 那么证明思路如下：令语句 $U \equiv \forall x (0 + x = x)$ ，只需证明 $Q \not\vdash U$ 即可证明 $Q \not\vdash \forall x (0 + x = x)$
 - » 根据逻辑有效性的定义，我们只需找到一个 L_A 的结构（一个偏离常理的、非预期的“非标准”的解释），令Q中的公理全都为真，但U为假
- > 接下来，我们定义一个新的数域 \mathbb{N}^* ，它除了包含 \mathbb{N} 中所有元素以外，还包含两个新常元 $\{a, b\}$ （你可以任意地解释它们，比如 a 代表哥德尔， b 代表他的好朋友爱因斯坦）
- > 我们仍然将0解释为“零”，将 $S, +, \times$ 解释为“后继、加法、乘法”函数，且所有函数在常规自然数上的意义也是不变的



Q是不完备的

证明 (Cont'd) :

- > a, b 不是自然数，所以我们必须对 $S, +, \times$ 进行延拓（下面只考虑 S 和 $+$, \times 该如何延拓留作练习）
- > 定义 $\{a, b\}$ 上的后继函数如下: $Sa = a, Sb = b$, 即它们的后继就是自己 (*self-successor*)
- > 容易验证，在这个新的解释下，后继函数仍然满足公理 I-3
- > 定义 \mathbb{N}^* 上的加法如下，其中 $m, n \in \mathbb{N}$:

+	n	a	b
m	$m + n$	b	a
a	a	b	a
b	b	b	a

- > 同样可以验证，在该解释下的加法满足公理 4-5
- > 然而，在该新解释下我们有 $0 + a \neq a$
- > 根据全称量词的解释， U 为假，因此 Q 无法逻辑蕴涵 U （因为 Q 的公理全为真时 U 为假）
- > 所以，根据 FOL 可靠性有 $Q \not\models U \Rightarrow Q \not\models U$

Q.E.D.



Q是不完备的

最终，我们可以得到如下定理：

定理3.35：Q是否定不完备的

- › 该定理显然成立：它不仅无法证明 U ，还无法证明 $\neg U$ （只需回到 L_A 的标准解释即可）
- › Q看起来比较“弱”，但它已经足够刻画全部的能行可判定的数值属性了，因此根据我们的定义，它是“充分强”的算术理论
 - » 不仅如此，它也是满足哥德尔定理的“最弱”的算术理论之一
- › 不过，就像我们上面展示的这样：尽管它能够 case-by-case 地证明某个具体的数字是否满足一个可判定的数值属性，但它完全无法对这样的证明进行概括
- › 因此，哥德尔考虑的是更强的 Peano 算术



数学归纳法

解决上面的问题只需要补充一条推理规则——数学归纳法（*principle of mathematical induction, MI*）：

“不管我们选择什么数值性质，只要零具备这个性质，并且任何具备这个性质的数字 n 都会将其传递给它的后继 Sn ，那么我们就可以推断出所有数字都具备这个性质。”

1. $\Gamma \vdash P0$ 且 $\Gamma \vdash Pn \rightarrow PSn$
2. $\Gamma \vdash P0 \rightarrow P1$
3. $\Gamma \vdash P1 \rightarrow P2$
4. \dots



证明(自然演绎):

1. $0 + 0 = 0$ Ax 4
2. $0 + a = a$ Supposition
3. $S(0 + a) = Sa$ 2, LL
4. $0 + Sa = S(0 + a)$ Ax 5
5. $0 + Sa = Sa$ 3,4 LL
6. $0 + a = a \rightarrow 0 + Sa = Sa$ 2-5, $\rightarrow i$
7. $\forall x (0 + x = x \rightarrow 0 + Sx = Sx)$ Gen. Thm.
8. $\{0 + 0 = 0 \wedge \forall x (0 + x = x \rightarrow 0 + Sx = Sx)\} \rightarrow \forall x (0 + x = x)$ MI
9. $\forall x (0 + x = x)$ 1,7,8 MP



数学归纳法作为形式系统公理

由于数学归纳法中的“性质 P ”是任意的，我们无法用 FOL 来形式化表示它。唯一的办法是对一阶谓词进行量化：

$$\forall X (\{X0 \wedge \forall x (Xx \rightarrow XSx)\} \rightarrow \forall x Xx)$$

- > 一方面，二阶逻辑（*Second-Order Logic*, SOL）存在许多问题，因此最好不要把 FOL 系统上升到 SOL；另一方面，我们的 FOL 算术语言 L_A 中亦不存在二阶量词
- > 因此，我们无法直接在系统中表示该 SOL 公理。我们可以采取老办法，把 MI 表示为 FOL 中的一个模式（schema）

定义（一阶归纳模式，*first-order Induction Schema*）：

$$\{\varphi(0) \wedge \forall x (\varphi(x) \rightarrow \varphi(Sx))\} \rightarrow \forall x \varphi(x)$$

其中 φ 是 L_A 中任意包含自由变元 x 的开公式



一阶 Peano 算术

定义 3.36 (一阶 Peano 算术) :

一阶 Peano 算术 PA 是基于经典一阶逻辑，使用语言 L_A ，并以 Q 的公理为基础，然后添加所有可以从 L_A 的开放 wff 中构建出的一阶归纳模式实例（的闭包），从而形成的一个完整的算术理论

- > 显然，PA 和 BA 一样具有无穷多个公理
- > 这看起来可能会带来问题，但和 BA 一样，判定一个 wff 是否是一阶归纳模式的实例是能行的。因此，PA 仍然是一个能行的形式化理论
- > 回顾前面的例子，我们令 $\varphi(x)$ 为 $x \neq Sx$
 1. $PA \vdash \varphi(0)$ 显然成立
 2. 对 $\forall x \varphi(x)$ 取逆否命题，可得 $PA \vdash \forall x \varphi(x) \rightarrow \varphi(Sx)$
 3. 根据数学归纳法可得 $\forall x \varphi(x)$ ，即 $\forall x x \neq Sx$ ，完美地排除了 Q 中我们引入的 a, b



一阶PEANO算术

定义3.36' (一阶Peano算术的非逻辑公理) :

1. $\forall x (0 \neq Sx)$
2. $\forall x \forall y (Sx = Sy \rightarrow x = y)$
3. $\forall x (0 + x = x)$
4. $\forall x \forall y (x + Sy = S(x + y))$
5. $\forall x (0 \times x = 0)$
6. $\forall x \forall y (x \times Sy = (x \times y) + x)$
7. $\{\varphi(0) \wedge \forall x (\varphi(x) \rightarrow \varphi(Sx))\} \rightarrow \forall x \varphi(x)$ 的所有实例形成闭包，其中 φ 为 L_A 的开公式， x 为其中的自由变元



准备工作2：量词复杂性



约束量词

L_A 语言中我们经常想表达：“所有/某些小于等于 τ 的数字满足某个性质 φ ”

- > $\forall x (x \leq \tau \rightarrow \varphi(x))$
- > $\exists x (x \leq \tau \wedge \varphi(x))$
- > 其中 $x \leq \tau$ 等价于 $\exists v (v + x = \tau)$
- > 我们进一步将其简写为有界量词 (*bounded quantifiers*)：
 - » $(\forall x \leq \tau) \varphi(x)$
 - » $(\exists x \leq \tau) \varphi(x)$

显然 Q 和 PA 可以用有穷并/交表达约束量词：对任意 n

1. $Q \vdash \forall x \{x = \bar{0} \vee x = \bar{1} \vee \dots \vee x = \bar{n}\} \leftrightarrow x \leq \bar{n}$
2. 若 $Q \vdash \forall x \{\varphi(\bar{0}) \wedge \varphi(\bar{1}) \wedge \dots \wedge \varphi(\bar{n})\}$ 那么 $Q \vdash \forall (x \leq \bar{n}) \varphi(x)$
3. 若 $Q \vdash \forall x \{\varphi(\bar{0}) \vee \varphi(\bar{1}) \vee \dots \vee \varphi(\bar{n})\}$ 那么 $Q \vdash \exists (x \leq \bar{n}) \varphi(x)$



定义3.37 (Δ_0 公式)：一个 L_A 公式是 Δ_0 的当且仅当它只包含

1. 非逻辑词： L_A 的非逻辑词加上 \leq (定义如前)
2. 逻辑词：命题连词、等号、有界量词

定理3.38 (Δ_0 公式的可判定性)：

1. Δ_0 语句 (*sentence*) 的真假是能行可判定的
2. Q 和 PA 可以正确地判定所有 Δ_0 语句



\sum_1 和 \prod_1 wffs

除了用于定义 \leq 的存在量词之外， Δ_0 公式只涉及有界量词。在量词复杂性的下一个等级，还有所谓的 \sum_1 和 \prod_1 公式，简单而言：

1. \sum_1 wff是通过对 Δ_0 wff进行零次、一次或多次存在量化而得到
2. \prod_1 wff是通过对 Δ_0 wff进行全称量化得到

定义3.39 (\sum_1 和 \prod_1 wff) :

我们按照如下步骤建立严格 Σ_1 wff的类：

1. 任何 Δ_0 wff都是严格的 Σ_1 wff
2. 如果 φ 和 ψ 是严格的 Σ_1 wff，那么 $(\varphi \wedge \psi)$ 和 $(\varphi \vee \psi)$ 也是严格的 Σ_1 wff
3. 如果 φ 是严格的 Σ_1 wff，那么 $(\forall \xi \leq \kappa)\varphi$ 和 $(\exists \xi \leq \kappa)\varphi$ 也是严格的 Σ_1 wff，其中 ξ 是 φ 中任何自由变量， κ 是一个数字或与 ξ 不同的变元
4. 如果 φ 是严格的 Σ_1 wff，那么 $(\exists \xi)\varphi$ 也是严格的 Σ_1 wff，其中 ξ 是 φ 中任何自由变元
5. 其他任何公式都不是严格的 Σ_1 wff



Σ_1 和 Π_1 wffs

定义3.39 (Σ_1 和 Π_1 wff) : (续) 我们按照如下步骤建立严格 Π_1 wff 的类:

1. 任何 Δ_0 wff 也是严格的 Π_1 wff
2. 如果 φ 和 ψ 是严格的 Π_1 wff, 那么 $(\varphi \wedge \psi)$ 和 $(\varphi \vee \psi)$ 也是严格的 Π_1 wff
3. 如果 φ 是严格的 Π_1 wff, 那么 $(\forall \xi \leq \kappa)\varphi$ 和 $(\exists \xi \leq \kappa)\varphi$ 也是严格的 Π_1 wff, 其中 ξ 是 φ 中任何自由变量, κ 是一个数字或与 ξ 不同的变元
4. 如果 φ 是严格的 Π_1 wff, 那么 $(\forall \xi)\varphi$ 也是严格的 Π_1 wff, 其中 ξ 是 φ 中任何自由变元
5. 其他任何公式都不是严格的 Π_1 wff

最后, 一个 wff 是 Σ_1 的当且仅当它在逻辑上等价于一个严格的 Σ_1 wff; 一个 wff 是 Π_1 的当且仅当它在逻辑上等价于一个严格的 Π_1 wff



\sum_1 和 \prod_1 wffs

几个简单的例子：

1. 偶数： $\psi(x) \equiv (\exists v \leq x)(2 \times v = x)$
2. 素数： $\chi(x) \equiv \{x \neq 1 \wedge (\forall u \leq x)(\forall v \leq x)(u \times v = x \rightarrow (u = 1 \vee v = 1))\}$

显然这两个公式都是 Δ_0 wff

> 语句“存在一个素偶数”是（严格的） Σ_1 wff

$$\exists x(\psi(x) \wedge \chi(x))$$

> 哥德巴赫猜想——即每个大于二的偶数都是两个素数之和，是（严格的） Π_1 wff

$$\forall x \{(\psi(x) \wedge 4 \leq x) \rightarrow (\exists y \leq x)(\exists z \leq x)(\chi(y) \wedge \chi(z) \wedge y + z = x)\}$$



\sum_1 和 \prod_1 wffs 的关系

定义3.4o:

1. 一个 Δ_0 wff 的否定仍然是 Δ_0 wff
2. 一个 Σ_1 wff 的否定是一个 Π_1 wff
3. 一个 Π_1 wff 的否定是一个 Σ_1 wff

证明概要:

1. 由定义成立
2. 根据定义, 任意 Σ_1 wff 的否定等价于某个严格的 Σ_1 wff $\neg\varphi$
 - » 处理公式 $\neg\varphi$ 时, 使用 ' $\neg\exists\xi$ ' 等价于 ' $\forall\xi\neg$ ', ' $\neg(\exists\xi \leq \kappa)$ ' 等价于 ' $(\forall\xi \leq \kappa)\neg$ ', 以及 ' $\neg(\forall\xi \leq \kappa)$ ' 等价于 ' $(\exists\xi \leq \kappa)\neg$ ', 结合德摩根定律, 将否定符号“向内推进”, 直到所有的否定符号都附加到 φ 最终构建的 Δ_0 wff 上
 - » 以上方法可得到一个逻辑等价的 wff ψ , 它是使用全称量词、有界量词、合取和析取应用于这些被否定的 Δ_0 wff 构建的
 - » 根据第一条结论, 否定 Δ_0 wff 仍然是 Δ_0 wff。所以 $\neg\varphi$ 的逻辑等价 wff ψ 确实是严格的 Π_1 wff, 而与严格 Π_1 wff 等价的 wff 本身也是 Π_1 wff。第 3 点的证明类似



回顾前面的准备工作



证明思路

- I. 大致的形式介绍了第一个不完备性定理：一个足够“好的”理论 T （包含基本算术语言）总是会是否定不完全的——总会有基本算术的句子它既不能证明也不能否定
2. 可以用两种方式来解释“足够好”的理论。我们可以假设 T 是可靠的；或者，放弃语义的假设，我们可以要求 T 是一个（大致上）一致的理论，它可以证明一些算术中的性质
3. 大致介绍了哥德尔用来建立该定理的基本策略——即我们“算术化语法”（即用数值编码关于可证明性的事实，并且我们可以在形式算术中表达这些事实），然后为理论 T 构造一个哥德尔语句，它当且仅当它在 T 中不可证明时为真
4. 证明了一个一致的、能行公理化的、足够强的形式理论不能是否定完全的。
 - » 这个论证很有启发性，因为它表明我们可以在不调用语法算术化和哥德尔语句构造的情况下得到不完全性结果。然而，这个论证依赖于“足够强”的概念，该概念是根据“可判定性质”的非正式概念定义的（记住，一个理论是足够强的，若它可以刻画自然数的所有可判定的性质）



形式化系统

1. 为了不那么抽象，需要考虑具体的算术理论。作为热身，我们首先定义了BA，这是一个无量词的关于特定数的加法和乘法的算术。这个理论是否定完全的且可判定
2. 若我们允许使用通常的FOL来扩展BA的语言，并用它们显然的全称量化替换BA的模式化公理（并添加一个公理，即每个非零数都是一个后继），我们就得到了更有趣的Robinson算术Q
3. 由于我们显著扩展了我们的算术语言所能表达的内容，但并没有大幅增加公理的力量，所以Q是无法否定完全的。例如，它不能证明 $\forall x (0 + x = x)$ 或它的否定。由此可见，Q是一个非常弱的算术，但已经足够满足哥德尔第一定理中的“适量”
4. 一阶皮亚诺算术PA在Q的基础上增加了一整套归纳公理（归纳模式的每个实例）。探讨发现，它与Q相比非常丰富和强大。在哥德尔之前，我们可能非常合理地认为它是一阶加法和乘法算术的否定完全理论。但该理论仍然是有效公理化的，并且第一定理将适用（假设PA是正确的，或至少是一致的并且满足另一个语法条件）。所以PA也将被证明是无法否定完全的。
5. 存在一些介于Q和PA之间的中间理论，这些理论具有归纳公理，但仅针对某些程度的量化复杂性的wff，即 \sum_1 和 \prod_1 wff



下一步

1. 我们目前所考察的算术形式理论最多只包含后继函数、加法和乘法。但为什么要止步于此呢？即使是高中算术也承认许多更多的数值函数，例如阶乘和指数函数。它们都属于一个非常广泛数值函数类，即所谓的原始递归（p.r.）函数。它们是有效可计算函数的一个主要子类
2. 如何在 L_A ——即现在熟悉的基本算术形式语言——中表达所有的p.r.函数和关系。
3. 重新引入“语法算术化”的关键思想，通过哥德尔编码，在PA中定义各种数值性质/关系，例如 $Wff(n)$, $Sent(n)$, $Prf(m, n)$ 等等。关键的是，这些性质/关系都是原始递归的
4. 构造一个哥德尔句子，该句子在且仅在它在PA中不可证时为真。证明哥德尔第一不完备性定理的语义版本
5. （如果有时间）证明第一不完备性定理的一个语法版本



原始递归函数



比后继、加法、乘法再复杂一点点...

加法：

1. $x + 0 = x$
2. $x + Sy = S(x + y)$

乘法：

1. $x \times 0 = 0$
2. $x \times Sy = (x \times y) + x$

阶乘：

1. $0! = 1$
2. $(Sy)! = y! \times Sy$

指数：

1. $x^0 = S0$
2. $x^{Sy} = (x^y \times x)$

以上“原始递归”的特征：

1. 第二个方程通过调用同一函数对于自变量 y 的值来获得关于 Sy 的函数值
2. 它们都通过一种特殊的函数复合定义，即在另一个函数中调用自身
3. 链式地对旧有函数进行递归复合：例如，加法是递归地通过后继定义；乘法是递归地通过后继和加法定义；阶乘、指数则是递归地通过乘法和后继定义



比后继、加法、乘法再复杂一点点...

考虑到这些简单的启发性例子，我们给出一个粗略的方式来定义原始递归函数：

原始递归函数（*primitive recursive function, p.r. function*）是一类：

- › 从像后继函数这样的简单“初始函数”开始，
- › 通过原始递归和复合函数链式定义的函数。



到底什么是“原始递归”？

1. 什么叫“原始地”递归？
2. 什么叫“复合旧有函数”？
3. 什么叫“简单的初始函数”？它的范围是什么？



到底什么是“原始递归”？

阶乘：

1. $0! = 1$
2. $(Sy)! = y! \times Sy$

阶乘函数的抽象：

1. $f(0) = g$, 这里的 g 只是一个常数
2. $f(Sy) = h(y, f(y))$, 其中 h 是一个已知的、比 f 更早定义的二元函数
 - » 这意味着 h 要么是“初始函数”
 - » 要么是一个更早被“原始递归”定义的函数

进一步抽象二元函数 $f(x, y)$ （例如乘法、加法）：

1. $f(x, 0) = g(x)$, 这里的 x 可看作参数, g 为已知的一元函数
2. $f(x, Sy) = h(x, y, f(x, y))$, 其中 h 是一个三元函数



加法和乘法是原始递归函数

I. $x + 0 = x \Leftrightarrow f(x, 0) = g(x)$

» $g(x) = I(x)$, 其中 $I(x)$ 为恒等函数

2. $x + Sy = S(x + y) \Leftrightarrow f(x, Sy) = h(x, y, f(x, y))$

» 为了将 $h(x, y, z)$ 设定为函数 Sz , 必须允许 h 对它的某些参数不敏感, 而只操作其他某个或某些参数

» 常规方法是引入其他简单恒等函数来“挑选”出特定的参数, 例如 I_3^3 接受 3 个参数, 并仅返回第 3 个参数。因此有 $I_3^3(x, y, z) = z$

» 那么, 加法中可以定义 $h(x, y, z) = S(I_3^3(x, y, z))$



加法和乘法是原始递归函数

I. $x \times 0 = 0 \Leftrightarrow f(x, 0) = g(x)$

» $g(x) = Z(x)$, 其中 $Z(x) = 0$ 为零函数

2. $x \times Sy = (x \times y) + x \Leftrightarrow f(x, Sy) = h(x, y, f(x, y))$

» $h(x, y, z) = f_0(I_1^3(x, y, z), I_3^3(x, y, z))$

» 其中 $f_0(x, y) = x + y$ 是已知的旧 p.r. 函数



原始递归

定义3.4I（原始递归，*primitive recursion*）：

若 f 满足下列条件（其中的 \vec{x} 是一个向量）：

1. $f(\vec{x}, 0) = g(\vec{x})$
2. $f(\vec{x}, Sy) = h(\vec{x}, y, f(\vec{x}, y))$

那么我们称 f 是通过对 g 和 h 的原始递归定义



原始递归中的函数复合

在上面的构造过程中，我们可以看到：

1. 更复杂的函数可通过已知简单函数的复合一步步构造
2. 其中最关键的操作为：将一个函数代入另一函数的自变量

该过程可以一般化为：

定义3.42（代入复合，*composition by substitution*）：

若 $g(\vec{y})$ 和 $h(\vec{x}, u, \vec{z})$ 为函数（其中 \vec{x} 和 \vec{z} 维度大于等于 0），那么我们称 f 是将 g 代入 h 中复合定义的，若 $f(\vec{x}, \vec{y}, \vec{z}) = h(\vec{x}, g(\vec{y}), \vec{z})$

- > 一般的函数复合——将多个函数代入另一函数——可视为迭代的逐次复合
- > 例如， $g(x, y)$ 代入 $h(u, v)$ 来得到 $h(g(x, y), v)$ ；然后再将 $g_0(y, z)$ 代入 $h(g(x, y), v)$ 中，得到复合函数 $h(g(x, y), g_0(y, z))$



原始递归中的初始函数

定义 3.43 (初始函数, *initial functions*) :

初始函数包括:

1. 后继函数 S
2. 零函数 $Z(\vec{x}) = 0$
3. 所有的 k 位恒等函数 $I_i^k(x_1, \dots, x_k) = x_i$, 其中 $k \in \mathbb{N}, 1 \leq i \leq k$



原始递归函数

定义 3.44 (原始递归函数, *p.r. functions*) :

原始递归函数的定义如下:

1. 所有初始函数是原始递归的
2. 若 f 是基于原始递归函数 g 和 h 通过将 g 代入 h 复合而得, 那么 f 是原始递归的
3. 若 f 是基于原始递归函数 h 和 h 通过原始递归而得, 那么 f 是原始递归的
4. 再无其他原始递归函数

(其中, 我们允许条件 2 和 3 中的 g 为常函数)



原始递归函数

就像FOL中的“证明”一样，“计算”现在也能被定义为一个由函数序列组成的链条：

定义3.45（定义链，*definition chain*）：

原始递归函数 f 的定义链是一个函数序列 f_0, f_1, \dots, f_k 。其中每个 f_j 要么是初始函数，要么是由其前序的函数通过复合或原始递归构造而来，且 $f_k = f$

显然，函数的定义链不是唯一的。根据数学归纳法，假设对于某个给定的性质 P ，我们可以依此证明：

1. 初始函数具有性质 P
2. 若函数 g 和 h 具有性质 P ，且 f 是通过 g 和 h 的复合构成，那么 f 也具有性质 P
3. 若函数 g 和 h 具有性质 P ，且 f 是通过 g 和 h 的原始递归构成，那么 f 也具有性质 P

那么任意原始递归函数均具有性质 P



原始递归函数是能行可计算的

我们可以一一检查以上性质：

I. 初始函数是能行可计算的

» 显然成立（只需用*for loop*枚举 + 判定即可）

2. 若函数 g 和 h 具有性质 P , 且 f 是通过 g 和 h 的复合构成, 那么 f 也具有性质 P

» 显然成立（只需先计算 g , 再将结果代入 h ）

3. 若函数 g 和 h 具有性质 P , 且 f 是通过 g 和 h 的原始递归构成, 那么 f 也具有性质 P

» 显然成立（只需用有界的循环计算即可）

```
f(0,1).  
f(n,y) :-  
    n > 0,  
    n1 is n-1  
    f(n1, y1),  
    y is y1*n.
```



在 Q 中表达和刻画原始递归函数

注意“表达”与“刻画”的区别：

I. L_A 语言可表达原始递归函数

- » 考虑最简单的情况，取一个单变量的原始递归函数 f
- » 我们希望有一个相应的逻辑表达式 $\varphi(x, y)$ ，使得对于任意 m, n ，若 $\phi(m, n)$ 为真当且仅当 $f(m) = n$
- » 事实上 L_A 可用低量词复杂度的表达式来表达任何原始递归函数：一个 Σ_1 表达式已足够

2. L_A 语言可刻画原始递归函数

- » 函数与谓词（关系）的“刻画”等价
- » 例如，给定一个单变量原始递归函数 f ，用 Q “刻画”它需要有一个相应的 wff $\varphi(\bar{m}, \bar{n})$
- » 也就是说：使得对自变量 x 的每一个实例 \bar{m}
 - » 若 $f(m) = n$ ，那么就有 $Q \vdash \varphi(\bar{m}, \bar{n})$
 - » 若 $f(m) \neq n$ ，那么就有 $Q \vdash \neg\varphi(\bar{m}, \bar{n})$



L_A 可表达任意 P.R. 函数

定理 3.45: L_A 可表达任意 p.r. 函数

证明概要:

- > L_A 可表达所有初始函数
 - » “后继”是 L_A 中 built-in 的函词，因此 $Sx = y$ 显然可被表达为 $Sx = y$
 - » 零函数 $Z(x) = 0$ 可表达为： $Z(x, y) \equiv (x = x \wedge y = 0)$
 - » k 维恒等函数同样可表达，例如 $I_2^3(x, y, z) = y$ 可表达为： $I_2^3(x, y, z, u) \equiv (y = u)$
- > 若 L_A 可表达 g 和 h ，那么它也能表达二者的复合函数
 - » 考虑一种简单情况， g 和 h 均为 1 元函数，且它们已经被表示为 $G(x, y)$ 和 $H(x, y)$ 。其余更复杂的情况（ n 元函数）可以类推
 - » 那么，复合函数 $y = f(x) = h(g(x))$ 可表达为 wff $\exists z (G(x, z) \wedge H(z, y))$
 - » 也就是说，对于任意 m, n ， $f(m) = n$ 当且仅当 $\exists z (G(\bar{m}, z) \wedge H(z, \bar{n}))$
 - » 必要性显然，只需证充分性： $\exists z (G(\bar{m}, z) \wedge H(z, \bar{n}))$ ，那么必然有一个 k 令 $G(\bar{m}, \bar{k}) \wedge H(\bar{k}, \bar{n})$ 成立。根据 g 和 h 的定义有 $g(m) = k$ 且 $h(k) = n$ ，那么显然有 $f(m) = h(g(m)) = h(k) = n$



L_A 可表达任意 P.R. 函数 (CONT'D)

- > 若 L_A 可表达 g 和 h , 那么它也能表达二者的原始递归
 - » 假设 $f(\vec{x}, 0) = g(\vec{x})$ 且 $f(\vec{x}, Sy) = h(\vec{x}, y, f(\vec{x}, y))$ (事实上, \vec{x} 只是“递归中我们不关心的变量”)。也就是说, 若 $z = f(\vec{x}, y)$, 那么
 1. 存在一个数字序列 k_0, k_1, \dots, k_y 令 $k_0 = g(\vec{x})$, 且对任意 $i < y$ 有 $k_{Si} = h(\vec{x}, i, k_i)$, 且 $k_y = z$
 2. 利用哥德尔的小技巧, 将该序列编码为一个函数:
 - » 将序列 k_0, k_1, \dots, k_y 编码 (coding) 为两个数字 c, d , 解码函数为 $\beta(c, d, i) = k_i$
 - » 只需找到足够大的 c 和一个较小的 d , 就能定义任意有穷序列! (Gödel's β -function Lemma)
 3. 存在 c 和 d 使得 $\beta(c, d, 0) = g(\vec{x})$, 且若 $i < y$ 那么 $\beta(c, d, Si) = h(\vec{x}, i, \beta(c, d, i))$, 且 $\beta(c, d, y) = z$
 4. 定义 $B(\bar{a}, \bar{b}, \bar{j}, \bar{k})$ 为真当且仅当 $\beta(a, b, j) = k$, 且 g 和 h 均能够被表示为 G 和 H , 那么原始递归函数就能被表示为如下 $\varphi(\vec{x}, y, z)$

$$\begin{aligned} \exists c \exists d \{ \exists k [B(c, d, 0, k) \wedge G(\vec{x}, k)] \wedge \\ (\forall i \leq y) [i \neq y \rightarrow \exists v \exists w \{ (B(c, d, Si, w)) \wedge H(\vec{x}, i, v, w) \}] \wedge \\ (B(c, d, y, z)) \} \end{aligned}$$



Q 可表达任意 P.R. 函数 (*CONT'D*)

定理 3.46: Q 可刻画任意 p.r. 函数

证明概要：

1. Q 可刻画所有初始函数（显然）
2. 若 Q 可刻画函数 g 和 h ，那么它就能刻画其复合函数 $f(x) = h(g(x))$
 - » 例如对于 1 元函数，关于 $\exists z (G(x, z) \wedge H(z, y))$ 的证明可递归地进行
3. 若 Q 可刻画函数 g 和 h ，那么它就能刻画它的原始递归
 - » 这一结论不那么显然，但 B 的确可以刻画哥德尔 β 函数。以此为基础，便可递归地刻画

$$\begin{aligned} & \exists c \exists d \{ \exists k [B(c, d, 0, k) \wedge G(\vec{x}, k)] \wedge \\ & (\forall i \leq y) [i \neq y \rightarrow \exists v \exists w \{ (B(c, d, S_i, w)) \wedge H(\vec{x}, i, v, w) \}] \wedge \\ & (B(c, d, y, z)) \} \end{aligned}$$



哥德尔编码：逻辑语法的算术化



L_A 语言的哥德尔编码

逻辑符号	编码
\neg	1
\wedge	3
\vee	5
\rightarrow	7
\leftrightarrow	9
\forall	11
\exists	13
$=$	15
(17
)	19

非逻辑符号	编码
0	21
S	23
+	25
\times	27

变元符号	编码
x	2
y	4
z	6
...	...



表达式的哥德尔编码

定义 3.47 (哥德尔数, *gödel number, g.n.*) :

假设表达式 e 是符号 s_1, s_2, \dots, s_k 和/或变量的序列。那么 e 的哥德尔数 (g.n.) 通过以下步骤计算：

1. 依次取每个 s_i 的基本编码 c_i
2. 使用 c_i 作为第 i 个素数 π_i 的指数
3. 然后将这些结果相乘, 得到 $2^{c_1} \cdot 3^{c_2} \cdot 5^{c_3} \cdots \pi_k^{c_k}$

1. 单个符号 S 的 g.n. 为 2^{23}
2. 数字 SS0 的 g.n. 为 $2^{23} \cdot 3^{23} \cdot 5^{21}$
3. wff $\exists y(S0 + y) = SS0$ 的 g.n. 为:

$$2^{13} \cdot 3^4 \cdot 5^{17} \cdot 7^{23} \cdot 11^{21} \cdot 13^{25} \cdot 17^4 \cdot 19^{19} \cdot 23^{15} \cdot 29^{23} \cdot 31^{23} \cdot 37^{21}$$

4. g.n. 一般非常大, 当我们说通过取素因子指数来解码 g.n. 是“能行的”并非等于“计算效率很高”
5. 我们只是说, 解码所需的计算程序——即反复提取素因子——只涉及基本算术的机械操作



证明的哥德尔编码

我们用“超级哥德尔数”为wff构成的证明序列编码：

定义3.48（超级哥德尔数， *super gödel number*）：

给定一个wff或其他表达式的序列 e_1, e_2, \dots, e_n ，我们首先通过常规的哥德尔数 g_i 编码每个 e_i ，得到一个相应的数字序列 g_1, g_2, \dots, g_n 。然后，我们通过素数幂再次编码，将该常规的g.n.序列编码为一个超级哥德尔数：

$$2^{g_1} \cdot 3^{g_2} \cdot 5^{g_3} \cdot \dots \cdot \pi_n^{g_n}$$

- › 解码一个超级哥德尔数只涉及两层素因子分解，因此仍然是能行可计算的：
 - I. 首先找到超级哥德尔数的素因子的指数序列；
 2. 然后将每个指数依次视为一个常规哥德尔数，再次进行素因子分解，以返回得到表达式的序列



L_A 语法性质的判定可被算术化

回顾：我们希望将关于表达式的语法性质编码为数值性质，哥德尔证明了如下定理：

定理3.49： Wff 和 $Sent$ 都是p.r.可判定的性质（1元谓词）； Prf 是p.r.可判定的关系（2元谓词）

非形式证明概要：

例如，对于语法解析函数，以 $Wff(n)$ 为例：

1. 首先需对 n 进行解码，即素数分解，它可以只用有界的for loop完成，因此是p.r.可判定的
 - » 这需要更多的p.r.函数，更多细节在Peter Smith的*An Introduction to Gödel's Theorems 2nd eds.* (IGT2e) 第14章
2. 询问结果表达式是否为 L_A 的wff。这在算法上是可判定的，所需for loop次数仅由其长度确定（这可以在第一阶段获得），因此也是p.r.可判定的
3. 上一阶段看似在符号串上进行，但是我们可以将“解码”和“解析”复合起来，因此整个计算仍然是在自然数上完成，且也是p.r.可判定的
 - » 其中关于“字符串解析”的部分看起来很不可思议，但确实可以做到，参见IGT2e的第20.2小节



L_A 语法性质的判定可被算术化

非形式证明概要：

判定 $Prf(m, n)$:

1. 首先需要 2 次解码超级哥德尔数 m , 根据上面的结论可知该过程是 p.r. 可判定的
2. 如果它解码成一个 wff 序列, 接着问: 该序列是一个正确构造的 PA 证明吗?
 - » 这也是 p.r. 可判定的 (检查序列中的每个 wff 是否是一个公理, 或者是根据 PA 的 Hilbert Style 推理规则判定它是否由先前的命题公式直接推出)
3. 如果该序列是一个证明, 接着问: 其最终命题公式的哥德尔数是否为 n ? 这依然是 p.r. 可判定的
4. 综上, 存在一个算法可以判断 $Prf(m, n)$ 是否成立。此外, 每个阶段所涉及的计算始终是一个简单的、有界的过程
5. 编码为 m 的表达式序列长度为所有的 for loop 决定了一个上限
6. 因此, Prf 是一个原始递归可判定的关系



对角化（自指）

最后，我们用新符号定义一个简单的构造

定义 3.50 (对角化, *diagnolization*) :

一个带有一个自由变元的命题公式 φ 的对角化是命题公式 $\varphi(\overline{\Gamma\varphi})$

- > 对角化命题公式 $\varphi(x)$ 是通过用整个命题公式 $\varphi(x)$ 的g.n.替换其自由变元 x 的所有出得到
- > 例如, $\exists y F(x, y)$ 的对角化是 $\exists y F(\overline{\exists y F(x, y)}, y)$
- > 更重要的是, 对角化是对表达式进行的一种基本机械操作

定理 3.51 (对角化函数是原始递归函数) :

存在一个原始递归函数 $diag(n)$, 当应用于某个带有一个自由变元的 L_A 命题公式的哥德尔数 n 时, 返回该命题公式对角化后的哥德尔数, 否则返回0

与任何原始递归函数一样, 包含基本算术理论的 T 可通过一个wff来表达和刻画这个函数, 记为 $\text{Diag}_T(x, y)$



对角化（自指）

定理3.51（对角化函数是原始递归函数）：

存在一个原始递归函数 $diag(n)$ ，当应用于某个带有一个自由变元的 L_A 命题公式的哥德尔数 n 时，返回该命题公式对角化后的哥德尔数，否则返回 0

与任何原始递归函数一样，包含基本算术理论的 T 可通过一个 wff 来表达和刻画这个函数，记为 $\text{Diag}_T(x, y)$

证明概要：

- I. 尝试将 n 视为一个哥德尔数，并对其进行解码：
 - » 若未得到一个带有一个自由变元的 wff，则返回 0
 - » 否则可得到一个型为 $\varphi(x)$ wff，跳转到下一步
2. 然后，可以通过一系列机械化操作构成 wff $\varphi(\bar{n})$ ，这便是它的对角化
 - » $Sub(n, n, 17)!$
3. 最后，我们计算该 wff 的哥德尔数，得到函数 $diag(n)$ 的取值
4. 该过程不涉及任何无界的 for loop。因此， $diag$ 是一个原始递归函数



第一不完备性定理



构造哥德尔语句

哥德尔将告诉我们如何构造一个命题公式 G , 它当且仅当在PA中不可证明时为真:

- I. 算术化允许我们构造“表达证明关系的wff”（如 Prf ）
2. 接下来，我们需要让一个wff描述其自身的不可证明性

我们首先调整 Prf 的定义，以得到：

定义3.52（对角化可证）：

关系 $Prfd(m, n)$ 成立，当且仅当 m 是一个PA-证明的超级哥德尔数，它证明了哥德尔数为 n 的wff的对角化

- > 假设 n 是一个wff $\varphi(x)$ 的哥德尔数；
- > 则 $Prfd(m, n)$ 成立当且仅当 m 是wff $\varphi(\bar{n})$ 的证明的超级哥德尔数
 - » $\varphi(\bar{n})$ 就是 $\varphi(x)$ 的对角化（自指语句）
- > 换句话说， $Prfd(m, n)$ 成立当且仅当 $Prf(m, diag(n))$



构造哥德尔语句

定理3.53: Prfd 是原始递归的，它可以被以下wff刻画

$$\text{Prfd}(x, y) \equiv \exists z (\text{Prf}(x, z) \wedge \text{Diag}(y, z))$$

证明概要：

与 $\text{Prf}(m, n)$, 可以机械地检查 $\text{Prfd}(m, n)$ 是否成立：

1. 解码 m , 判定它是否给出一个wff序列
2. 如果是, 判定它是否是一个PA-证明
3. 如果是, 判定证明的结论命题公式是否为哥德尔数 n 的命题公式的对角化
 - » 这仅需包含基本for loop的算法就足够
4. 因此, Prfd 是原始递归的



哥德尔语句

现在开始巧妙的哥德尔构造！

首先我们构造一个开公式，并将其缩写为 U （可以考虑成“不可证明”）。为了明确其中的自由变元，定义：

定义3.54：

$$U(y) \equiv \neg \exists x \text{Prfd}(x, y)$$

接着，我们对 U 进行对角化，得到：

定义3.55：

$$G \equiv U(\overline{\Gamma U}) = \neg \exists x \text{Prfd}(x, \overline{\Gamma U})$$



哥德尔语句不可证

最后，我们得到如下美妙的结果：

定理3.54： G 当且仅当在PA中不可证时为真

证明：

- › 考虑 G 在 L_A 的解释中为真的条件，假设谓词 $Prfd$ 表达数值关系 $Prfd$
- › G ，即 $\neg \exists x Prfd(x, \overline{\Gamma U \Gamma})$ ，当且仅当没有数 m 满足 $Prfd(\overline{m}, \overline{\Gamma U \Gamma})$ 时为真
- › 也就是说，根据 $Prfd$ 的定义， G 当且仅当不存在数 m 使得 m 是哥德尔数为 $\overline{\Gamma U \Gamma}$ 的wff的一个PA-证明
- › 带有哥德尔数 $\overline{\Gamma U \Gamma}$ 的wff当然就是 U ；其对角化是 G
- › 因此， G 成立当且仅当不存在数 m 使得 m 是一个 G 的PA-证明超级哥德尔编码
- › 但若 G 在PA中是可证的，则必然存在某个数 m 是其证明的超级哥德尔编码
- › 因此， G 当且仅当在PA中不可证时为真

Q.E.D.



哥德尔第一不完备性定理

定理3.9 (哥德尔不完备性定理I) :

若 T 是一个能行公理化的形式理论且它的语言包含基本算术理论。那么若 T 是可靠的，那么就存在一个基本算术理论中为真的语句 G_T ，令 $T \not\vdash G_T$ 且 $T \not\vdash \neg G_T$ 。

也就是说， T 是否定不完备的。



The End(less)