数理逻辑

03 - 证明、计算及其边界

(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

2025年 - 春季

https://daiwz.net

前情提要

逻辑是什么？

逻辑的作用是**系统性地检验论证的有效性**

1. 什么叫“论证”（arguments）？
2. 什么叫“检验”（evaluation）？
3. 什么叫“有效性”（validity）？
4. 什么叫“系统性地”（systematically）？

“系统性地检验”

在这门课上，我们**只关注第二个维度**：

1. “前提”来自各个领域，其正确性和“逻辑”本身无关
2. 逻辑学家关心的是：“从某些（假定为真的）论据出发，究竟能不能得到最终的结论”

“系统性地检验”

在这门课上，我们**只关注第二个维度**：

1. ~~逻辑学家关心的是：“从某些（假定为真的）论据出发，究竟能不能得到最终的结论”~~
   - 语法：一阶语言 $\mathcal{L}$ 与逻辑公理 Λ
2. ~~“前提”来自各个领域，其正确性和“逻辑”本身无关~~
   - 语义：理论 Γ 与结构 $\mathfrak{A}$

语法（Syntax）

1. 词汇表、符号
   - 逻辑词、非逻辑词
     - 连词、布尔函数、连词的完全组
   - 等词
2. 表达式
   - **合式公式（wff）**
   - 归纳原理（结构归纳）、括号引理、唯一可读性

语义（Semantics）

什么叫做“真”？

- 命题词、素公式、真值指派
- Tarski：“结构”与“解释”

有效性

论证的过程是否**滴水不漏**?

有效性

$$\Gamma \models_{\mathfrak{A}} \varphi[s]$$

有效性

$$\Gamma \models \varphi$$

希望经~~过本学期的学习~~现在可以回答的问题

$\{\sigma \mid \forall \mathfrak{U}\ (\mathfrak{U} \models \sigma)\}$是个怎样的集合?

论证：$\vdash$

定义1.19（*derivation*）：

若Γ是一个wff集合，那么Γ中的一个推导是一个wff有穷序列$\langle \alpha_1, \ldots, \alpha_n \rangle$，对任意$i \leq n$有下面三者之一成立：

1. $\alpha_i \in \Gamma$
2. α_i是一条公理
3. α_i根据推理规则从某些α_j（及α_k）得到，其中$j < i$（且$k < i$）
   - » 唯一的规则：若$\beta \to \alpha$和β同时出现在推导的**前段**，那么β也是正确的
   - » 推理规则（*rule of inference*）给出了推理步骤正确性的充分条件。它其实是一种“元规则”，即一种模式，并不只是一条规则

公理系统中的逻辑演算

$$\vdash \delta \to \delta$$

证明:

1.	$\delta \to ((\delta \to \delta) \to \delta)$	Ax 1.22.7
2.	$(\delta \to ((\delta \to \delta) \to \delta)) \to$ $((\delta \to (\delta \to \delta)) \to (\delta \to \delta))$	Ax 1.22.8
3.	$(\delta \to (\delta \to \delta)) \to (\delta \to \delta)$	1, 2 MP
4.	$\delta \to (\delta \to \delta)$	Ax 1.22.7
5.	$\delta \to \delta$	3, 4 MP

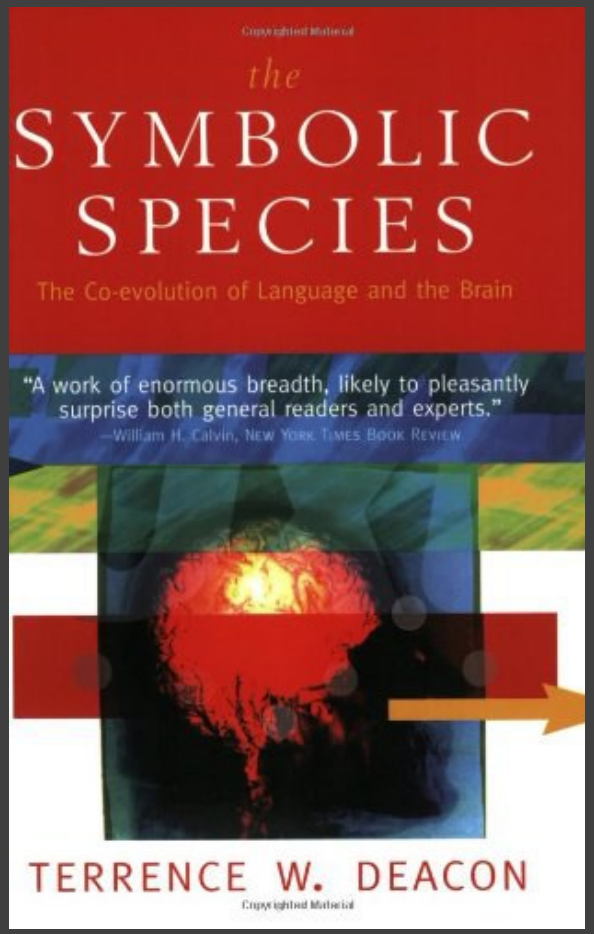
Q.E.D.

元定理

- 演绎定理
- 概括定理
- **可靠性定理**：$\Gamma \vdash \sigma \Rightarrow \Gamma \models \sigma$
  - “一切可以证明的结论，都是真的”
    - 一阶逻辑公理的有效性
      - 替换引理
- **完全性定理**：$\Gamma \models \sigma \Rightarrow \Gamma \vdash \sigma$
  - “一切为真的结论，都可以证明”
    - 可满足性、一致性与完全性
    - 构造性证明、集合的外延式定义
    - Lindenbaum 引理、紧致性定理

语言和逻辑

“语言和大脑共同进化”


Copyrighted Material
the
SYMBOLIC
SPECIES
The Co-evolution of Language and the Brain
“A work of enormous breadth, likely to pleasantly surprise both general readers and experts.”
—William H. Calvin, New York Times Book Review
TERRENCE W. DEACON
Copyrighted Material

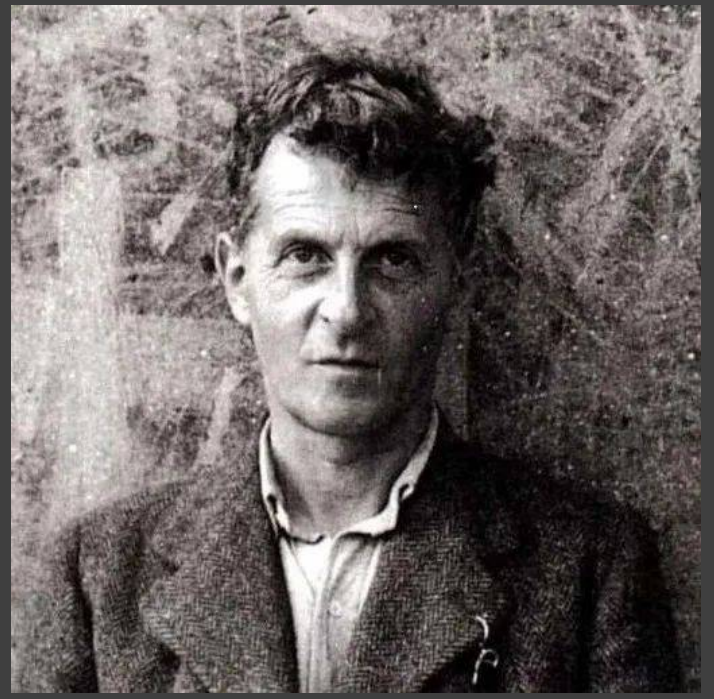
自然语言和经典逻辑

实质蕴涵（material implication, $\to$）的问题：

- 事实条件句按经典逻辑全部为真，但直观上显然有真有假
  - 例如反事实条件句没有逆否规则：从 $A \to B$ 推不出 $\neg B \to \neg A$：
  - “如果希特勒当年没死，他也活不到现在”。这句话直观为真，但其逆否“如果希特勒活到现在，那他当年就死了”显然为假
- 非确定性推理也不符合经典的推理规则
  - 即使你有很高的概率接受 $A \vee B$，你也不一定有很高的概率接受 $\neg A \to B$
  - 你非常相信现在是12点，因此你也非常相信现在是12点或6点，但你不会相信“如果现在不是12点那现在就是6点”
- 大多数自然语言中的条件句都不能通过实质蕴含来形式化，否则
  - 当我们否定一个条件句，我们就能得到该条件句的前件：$\neg(A \to B)$ 可以推出 A
  - 但：“并非（如果我是一个好的逻辑学家，我就是一个好的文学家）”是真的，因此“我是一个好的逻辑学家”

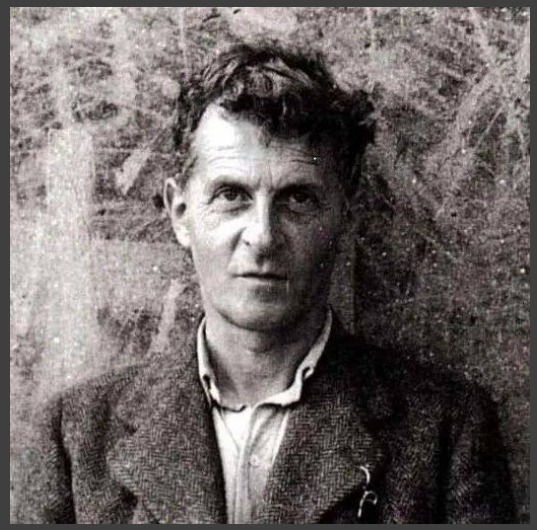
符号的边界

“西边的欧钢有老板
生儿维特根斯坦
他言说马户驴又鸟鸡
到底那马户是驴还是驴是又鸟鸡
那驴是鸡那个鸡是驴那鸡是驴那个驴是鸡
那马户又鸟
是我们人类根本的问题”
——刀郎《罗刹海市》



符号的边界

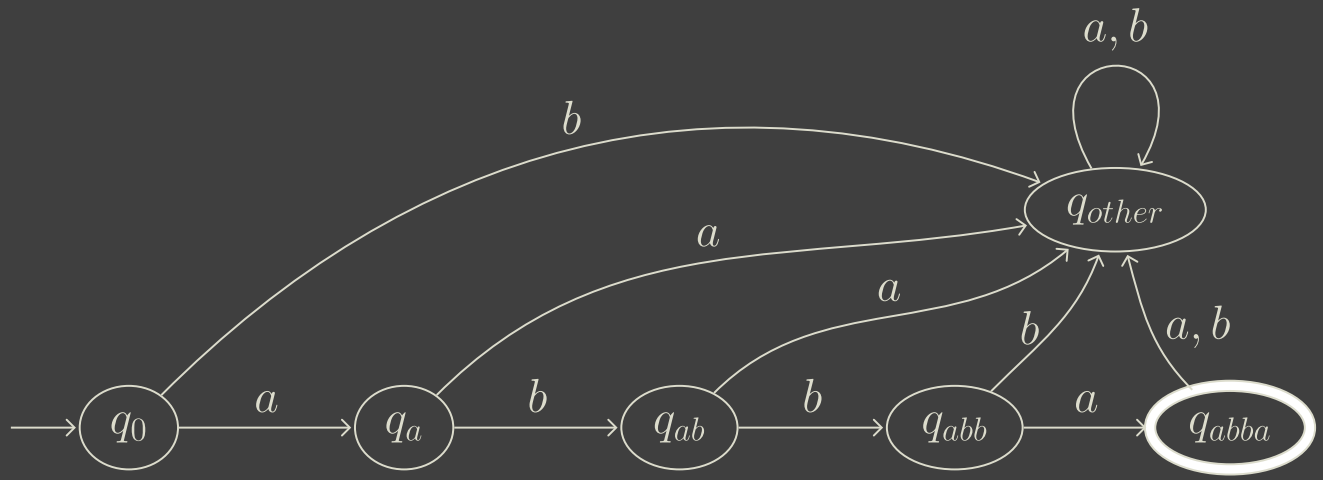
1. **语言游戏（Language Game）**：语言的多样性和情境性
   - 语言不是抽象规则的系统，而是一种在特定情境下的活动。每种语言的使用，都像是在玩一个游戏，而不同游戏有不同的规则和目的
     - 问题和背景不同，就会产生不同的“语言游戏”：法庭上 - 讲证据、辩论；数学课堂上 - 定义、证明；……
   - 语言的意义来自于使用（use），而不是严格的定义
2. **家族相似性（Family Resemblance）**：概念之间关系的方式
   - 概念之间并不是通过一组严格定义的共同特征来划分，而是像家族成员一样，有一些重叠的、交织的相似性
   - 没有必要为所有事物找到一个严格的“本质定义”
   - 就像家族成员：有的有相似的鼻子，有的有相似的眼睛，但并没有一个所有成员都有的特征



计算与逻辑

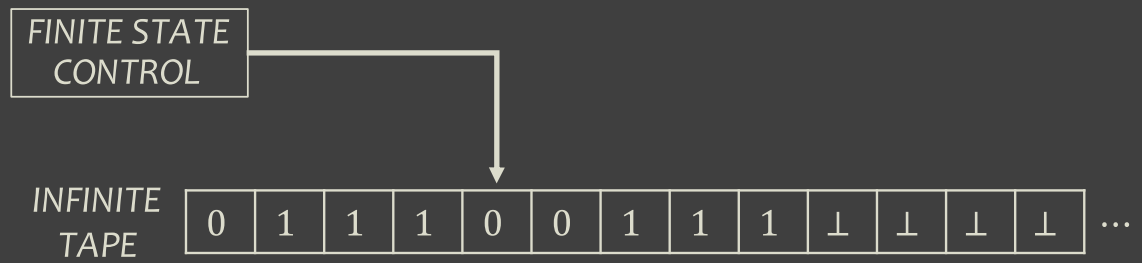
离散状态机

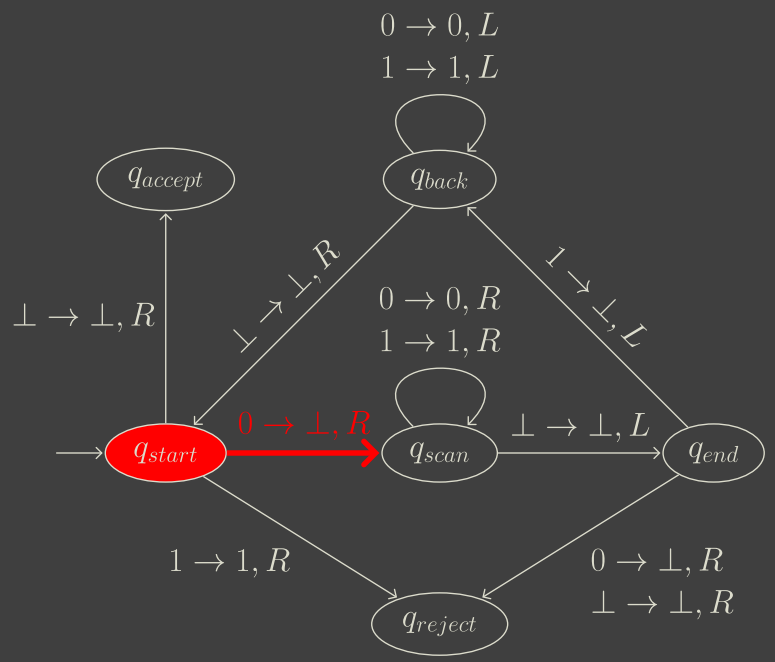
断言：“计算”可以用**离散状态机（discrete state machines）**表示



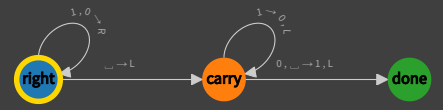
想像你手下有一个足够“愚蠢”的计算员（computer），TA能够可靠地执行任务，但记忆力很差，还很喜欢上厕所。一旦任务进行到一半，上厕所回来以后就会忘记该做什么。因此，我们要给TA提供无限多的草稿纸，并且帮TA把执行任务的规则记下来；同时，TA也能用草稿纸把任务的执行状态记下来，甚至方便另一些“愚蠢”的计算员接替TA干活。

Turing Machine





TM 的例子 (TURINGMACHINE.IO)




```
# Adds 1 to a binary number.
input: '1011'
blank: ' '
start state: right
table:
  # scan to the rightmost digit
  right:
    [1,0]: R
    ' '  : {L: carry}
  # then carry the 1
  carry:
    1      : {write: 0, L}
    [0,' ']: {write: 1, L: done}
  done:


# Exercises:

# • Modify the machine to always halt on the leftmost digit
#   (regardless of the number's length).
#   Hint: add a state between carry and done.

# • Make a machine that adds 2 instead of 1.
#   Hint: 2 is '10' in binary, so the last digit is unaffected.
#   Alternative hint: chain together two copies of the machine from
#   the first exercise (renaming the states of the second copy).

# • Make a machine to subtract 1.
#   To simplify things, assume the input is always greater than 0.

```

Try it out

Run the machine to see it in action. At any time, you can **step** or **pause** to get a closer look, or **reset** to start over.

There are over a dozen different example machines

Make your own

Make spinoffs from the examples or your own creations by using *Edit > Duplicate document*. You can also start from scratch with a new blank document.

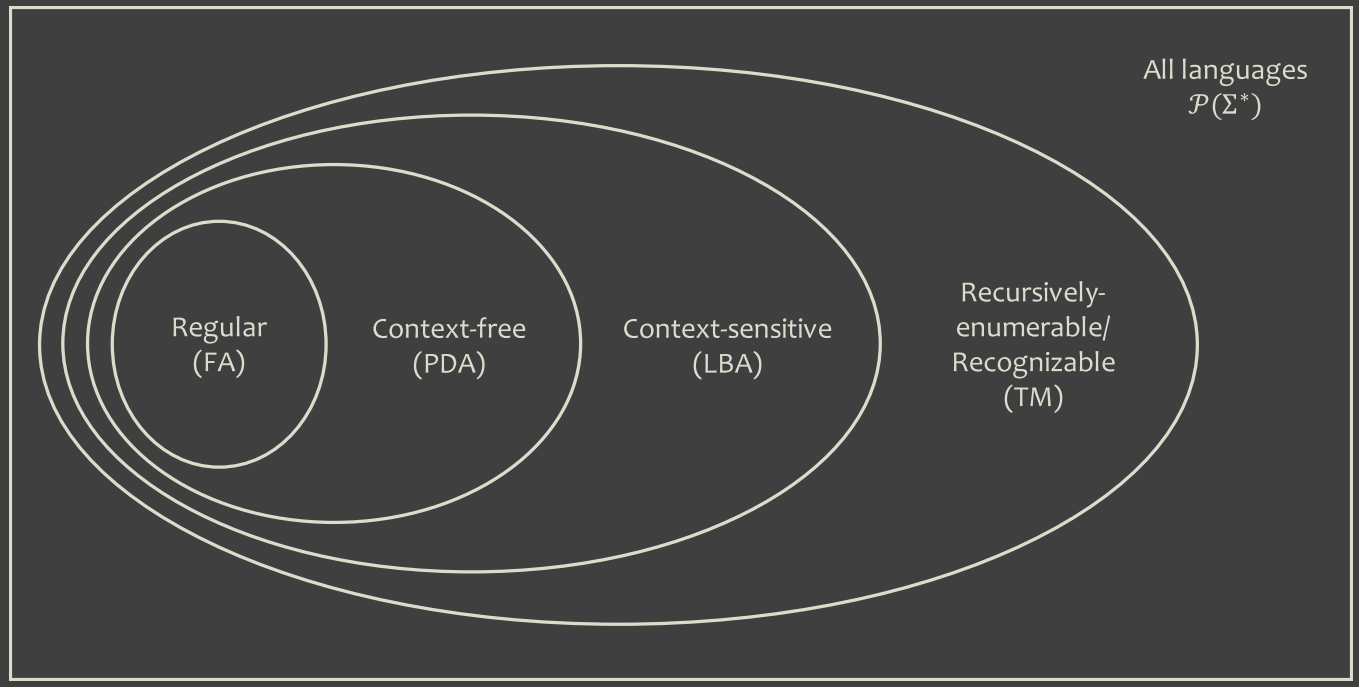
All it takes to describe a Turing machine is a start

Tips

Editor keyboard shortcuts

Ctrl - S	Load machine Save changes and load the machine.

可计算性



等价模型: General Recursive Functions (Gödel), Lambda Calculus (Church), 算盘机 (胡海星；宋方敏), *etc.*

计算与逻辑

```
% UTM
start_binary_add :-
    string_chars("1011+11001", TapeChars),
    run(right, 0, TapeChars).

start_binary_add(S) :-
    string_chars(S, TapeChars),
    run(right, 0, TapeChars).

run(done, _, Tape) :-
    format("*** Machine has halted. Final tape: ~w~n", [Tape]),
    !.
run(State, Pos, Tape) :-
    read_sym(Tape, Pos, Sym),
    transition(State, Sym, NewState, WriteSym, MoveDir),
    write_sym(Tape, Pos, WriteSym, TempTape, PosAfterWrite),
    move_pos(PosAfterWrite, MoveDir, NextPos),
    format("State=~w, Read=~w, Write=~w, Move=~w, NewState=~w, Tape=~w~n",
           [State, Sym, WriteSym, MoveDir, NewState, TempTape]),
    run(NewState, NextPos, TempTape).
```

可判定与可计算

定义3.1（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$f(o)$

- 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$c_P(o) = 0$

可判定与可计算

定义3.2（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中的主连词的位置是能行可判定的
  - 对括号个数进行计数
- 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

逻辑/计算的边界

Gödel的论文

On formally undecidable propositions of *Principia mathematica* and related systems I[1] (*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 integers[4] 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?

Gödel 的终极问题

一个描述能力足够强的数学系统是完全的吗？

回顾

定义（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（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 3.3（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

能行公理化的形式理论

定义 3.4（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 determiniable*），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3.5（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3.6（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$T \nvdash \varphi$且$T \nvdash \neg\varphi$

定义3.7（否定完全性，*negation completeness*）：

我们称T是一个否定完全（也叫“语法完全”，*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$T \vdash \neg\varphi$

哥德尔第一不完全性定理概述

定理3.8（哥德尔不完全性定理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$T \nvdash G_T$且$T \nvdash \neg G_T$。

也就是说，T是否定不完全的。

定理3.8'（哥德尔不完全性定理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$T \nvdash G_T$且$T \nvdash \neg G_T$。

因此，T是否定不完全的。

不完全性与不可完全性

我们考虑语义上的不完全性，既然$T \nvdash \mathsf{G}_T$且$T \nvdash \neg\mathsf{G}_T$，那我们为什么不将$\mathsf{G}_T$加进去？即令

$$U = T \cup \{\mathsf{G}_T\}$$

是否U就完全了呢？

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？

哥德尔的完全性和不完全性定理

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 Calculus, etc.），那么根据哥德尔完全性定理，我们有：

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

完全性指的是"逻辑有效意义下的完全性"。即在任意结构（模型）下都有蕴含关系的话，那么这个结论可证。

根据哥德尔不完全性定理，我们有：存在一些对于理论T来说为真的wff φ 不可证，即

$$\exists \varphi \, (T \vDash_{\mathfrak{A}} \varphi) \Rightarrow T \nvdash_S \varphi \quad \text{且} \quad T \nvdash_S \neg\varphi$$

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全，即T永远不可完全

注意，这里的T有了更多限制，比如"足够描述PA"，它本身也只在部分模型下是有效的。因此这里的T和φ的蕴涵关系不再是"逻辑意义下有效"。故而不完全性指的是"对于T来说为真的模型里存在不可证的结论"。

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3.10（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3.11（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/谓词：

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

- 注意，这里的n都是一个字符串的“编码”

哥德尔数

定理 3.10：

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 3.11 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 3.12： 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）$\mathrm{Prf}_T(x, y)$

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 $PrfT$ 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3.12，我们能够构造如下关于“可证”在T中如何表达的定义：

定义3.13（可证明性谓词，*provability predicate*）：

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

那么，$\mathsf{Prov}_T(\bar{\mathsf{n}})$，即$\exists \mathsf{z} \mathsf{Prf}_T(\mathsf{z}, \bar{\mathsf{n}})$为真，当且仅当某个数$\mathsf{z}$恰好是哥德尔编码为$\bar{\mathsf{n}}$的句子的一个$T$-证明，也即当且仅当哥德尔编码为$\bar{\mathsf{n}}$的句子是$T$的一个定理。因此，$\mathsf{Prov}_T$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 3.14（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 $\neg \mathsf{Prov}_T(\overline{\ulcorner \mathsf{G}_T \urcorner})$ 为真，其中 $\overline{\ulcorner \mathsf{G}_T \urcorner}$ 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 $\neg \mathsf{Prov}_T(\overline{\ulcorner \mathsf{G}_T \urcorner})$ 也为真
- 即当且仅当由 $\overline{\ulcorner \mathsf{G}_T \urcorner}$ 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

不完全性定理的证明概要

定理3.8（哥德尔不完全性定理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$T \nvdash G_T$且$T \nvdash \neg G_T$。

也就是说，T是否定不完全的。

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$T \nvdash G_T$
- 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$T \nvdash \neg G_T$
- 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.

说谎者悖论和不完全性定理

看起来都是自指语句，为什么说谎者悖论是悖论，而哥德尔不完全性定理是定理？

- 这个问题触及了不完全性定理的深层根源
- 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语句的性质。
- 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

The End(less)