数理逻辑

讲义，第 6.3 版，2024 年

北京大学 信息与计算科学系

林作铨

linzuoquan@pku.edu.cn

7 计算机科学基础 *

- 算法
- 可计算性
- 不可判定性
- 计算复杂性
- 定理机器证明
- 计算逻辑
- 智能逻辑

算法

Hilbert 第十问题

是否有一个算法判定一个多项式 Diophantine（丢番图）方程是否有整数解 $\Leftarrow$ 判定问题

注

Diophantine 方程（不定方程）是变量仅许整数的多项式等式，形如

$$a_1 x_1^{b_1} + a_2 x_2^{b_2} + \cdots + a_n x_n^{b_n} = c$$

其中 a_i, b_j, c 均为整数；若能找到一组整数解 $m_1, m_2 \cdots m_n$ 者则称之有整数解

例：勾股定理（$x_1^2 + x_2^2 - x_3^2 = 0$）的整数解

费马大定理（$x_1^m + x_2^m = x_3^m, m > 2$）等

Davis（1973）证明了这是个不可解问题

定义 7.1

一个算法（algorithm）是一个刻画计算过程的机械能行的指令集，它可在有限步执行指令对给定的一类问题中任一问题求解 ◇

例 7.2

Hilbert 第十问题即为是否有下面一类问题的算法

$$\{\text{方程 } E \text{ 是否存在整数解} \mid E \text{ 是一个多项式 } \textit{Diophantine} \text{ 方程}\}$$

注

算法是一个直观的概念，不能直接给出精确定义

问题

什么是计算？

如何给出“计算”的数学定义，亦是计算机科学的理论基础

例 7.3

考虑任何形式的问题类

(a) $\{f(n)$ 的值是多少? $\mid\ n \in D_N\}$ (f 是一个给定的函数)

(b) $\{n$ 是集合 A 的元素吗? $\mid\ n \in D_N\}$ (A 是一个给定的集合)

(c) $\{\mathscr{A}$ 是 $\mathscr{N}$ 中的定理吗? $\mid\ \mathscr{A}$ 是 $\mathscr{N}$ 中的公式$\}$

例 7.4

有算法的问题类

(a) $\{2$ 是 n 的因子吗? $\mid\ n \in D_N\}$

(b) $\{n$ 属于素数集吗? $\mid\ n \in D_N\}$

(c) $\{f(n)$ 的值是多少? $\mid\ n \in D_N\}$，其中 f 是由 $f(n) = 2n\ (n \in D_N)$ 定义的函数

如 (c)，可把问题类限定到“算法可计算的函数类”

问题

直观的算法能否用一定的数学方式（如函数）来描述？

一定的数学方式足以描述所有直观的算法（如递归函数）

- 算法
- 可计算性
- 不可判定性
- 计算复杂性
- 定理机器证明
- 计算逻辑
- 智能逻辑

可计算性

什么是计算？

计算模型

对算法的一般数学定义，使得足以描述所有直观的算法

- Turing（图灵）机（器）：抽象计算机
- Thue（图叶）重写（rewritting）系统：计算过程的形式化
- 递归函数：可计算函数的形式化
- 其它：各种自动机和形式语言（Chomsky 文法）等

算法可计算函数

一个偏函数有算法可计算，若有一个算法当函数有定义时可算出函数值

注

- 计算模型都包含偏函数类

  例：递归函数对最小数规则不加限制（递归偏函数）

  Turing 机永不停机

- 不同的计算模型已证明是等价的，即它们所定义的由算法可计算（偏）函数类已证明属同一类，即递归偏函数

- 算法是一个直观的概念，任一计算模型作为算法的精确定义是否就是算法本身无法证明，亦即此偏函数类是否就是算法可计算的函数类是无法证明的

- 找不到一个算法是计算模型不能刻画的（无反例或反例有争议）

Church 论题（Thesis）

算法可计算的偏函数类等同于递归偏函数类

(I) 每个算法可计算的偏函数是递归偏函数

(II) 每个递归偏函数是算法可计算的

(II) — 基于直观上机械能行的算法可根据递归函数定义归纳地证明

(I) — 只能是论题：无法证明，但有足够理由接受

—— 递归偏函数包含所（已）有的算法可计算函数

—— 不同的计算模型的等价性

注

基于 Church 论题

- 寻找一个函数是否存在算法变成证明该函数是否是递归的
- 发现一个函数的特殊算法说明该函数是递归的

例 7.5

(a) $\mathscr{N}$ 中在 N 为真的公式的 Gödel 数集不是递归的

据 Church 论题，没有算法回答下列问题

$$\{\mathscr{A} \text{ 在 } N \text{ 为真吗?} \mid \mathscr{A} \text{是 } \mathscr{N} \text{ 中一个公式}\}$$

(b) 令 A 是 D_N 的一个子集，它包含所有能表为两个平方数之和的数

A 是递归的（但相当难证），可有一个相当简单的算法回答下列问题

$$\{n \in A? \mid n \in D_N\}$$

定义 7.6 (能枚举)

一个形式系统（如 $\mathscr{N}$）其合适公理（的 Gödel 数）集是递归的，因此可设计一个机械能行的过程把该系统（$\mathscr{N}$）的所有定理枚举出来，即直观上是能枚举的 ◇

注

- 命题 3.25（一阶语言的枚举）和 命题 4.72（一阶逻辑的半可判定性）的证明都基于（直观上）能枚举
- 一个形式系统其合适公理（的 Gödel 数）集是递归的，才有一个算法来判定什么是公理和证明，设计演算的目的是包含尽可能多语义上成真的公式为可证的公式，这是完全性定理所刻画的基本性质

注（续）

- 在不完全性定理证明中，要求形式系统（$\mathscr{N}$ 的扩张 S）其合适公理（的 Gödel 数）集是递归的（参见命题 6.37），就存在一个算法来判定一个公式（序列）为公理（证明），才能考虑该公式是否为（语义上）数学真理
- 即使这样的（算术）形式系统，存在一个算法来判定一个公式（序列）为公理（证明），其定理集并不包含所有（在算术解释下）为真的公式，这是不完全性定理所揭示的本质

定义 7.7 (递归可枚举)

一个 D_N 的子集是递归（可）枚举的（recursive enumerable），若它是一个递归函数的值域，或它是空集 ◇

一个集是递归可枚举的，若

存在一个递归函数 f 使得 $f(0)$, $f(1)$, $f(2)$, $\cdots$ （可重复）是该集所有元素的列表 $\Rightarrow$ 能枚举

注

基于 Church 论题，递归可枚举等同于能枚举

命题 7.8 (递归集与递归可枚举集)

每个递归集都是递归可枚举的 ◇

注

反之不然，只需一个反例（推论 7.10）

定义 7.9 (递归不可判定)

一个形式系统是递归不可判定的（recursive undecidable），若该系统中定理的 Gödel 数集不是递归的 ◇

注

- 据 Church 论题，一个形式系统 S 是递归（不）可判定的若（不）存在一个算法判定下集的问题

$$\{\mathscr{A}\text{ 是否一个定理？} \mid \mathscr{A}\text{ 是 } S\text{ 的一个公式}\}$$

- 前注说明 $\mathscr{N}$ 中所有定理（的 Gödel 数）集是递归可枚举的，可证明 $\mathscr{N}$ 是递归不可判定的，即意味着该集不是递归的

推论 7.10

存在 D_N 的子集，它是递归可枚举的，但不是递归的 ◇

定义 7.11 (递归不可解)

一类问题是递归不可解的（recursively unsolvable），若没有一个算法能提供该类所有问题的解

一个形式系统 S 是递归不可判定的，当且仅当问题类

$$\{\mathscr{A} \text{ 是 } S \text{ 中的一个定理吗?} \mid \mathscr{A} \text{ 是 } S \text{ 的一个公式}\}$$

是递归不可解的

注

递归不可解是比递归不可判定更一般的概念

定义 7.12 (Turing 机)

Turing 机是一种精确描述机械能行的计算过程的抽象机，由一个读写头和一条纸带组成

- 一条分割成方格的纸带（潜在无限长）从读写头中间穿过，通过读写头纸带上可能印有有限个符号，每个方格能印一个符号，也可能没有印
- 依照一定规则操作纸带，并可能在某时刻停机，或永不停机（时间无限）。当停机时，纸带上留下的即为输出信息；否则，计算不终止，没有输出
- 一个纸带符号的字母表，是符号的有限列表，每个纸带方格每次只能输出至多一个符号；以字母 B 表示空白方格，最简单的 Turing 机可只有两个符号：B 和 1

Turing 机的操作

每次读纸带上一个方格，做以下操作之一为一步

(1) 打印一个符号（先擦除之前的符号）；擦除一个符号，即打印 B

(2) 左移一个方格

(3) 右移一个方格

内部状态

每台 Turing 机允许有有限个（内部）状态（即存贮器），在计算过程中内部状态可以改变，计算中每一步需明确

(1) 机器当前的状态

(2) 当前读取的方格的内容

(3) 机器做出的动作

(4) 机器下一步将进入的状态

四元组

使用四元组的有限集表示 Turing 机

(状态, 纸带符号, 动作, 新状态)

- 根据状态和当前读的符号，在四元组集中查找以这对 (状态, 符号) 打头的四元组，依这个四元组执行动作并进入新状态
- 限制四元组集必须是一致的，即对每对 (状态, 符号) 至多有一个四元组以此开头（即 Turing 机是良定义的）
- 停机当且仅当当前的 (状态, 符号) 对不在四元组集中出现
- 用 $q_0, q_1, q_2 \cdots$ 表示状态，q_0 为开始状态，开始时正在读取纸带最左边的非空方格

$\diamondsuit$

注

四元组（集）可看成指令（集），一个四元组集相当于一个指令系统

例 7.13

$$\{(q_0 \quad 1 \quad B \quad q_1),(q_1 \quad B \quad R \quad q_0)\}$$

是一个具有状态 q_0，q_1，符号表为 $\{B, 1\}$ 的 Turing 机的四元组集

第三个符号可以是 L（左移），R（右移），或用来替换当前正在读取的符号的符号

例（续）

若输入带包含一个有穷的 1 的序列，如

...	B	B	1	1	1	1	1	B	B	...

$\uparrow$

q_0

开始状态 q_0，读取最左边的 1，（擦除 1 后）打印 B 并进入状态 q_1

...	B	B	B	1	1	1	1	B	B	...

$\uparrow$

q_1

在状态 q_1 下读到 B，右移一格并进入状态 q_0

...	B	B	B	1	1	1	1	B	B	...

$\uparrow$

q_0

例 (续)

依次右移，将把每个 1 用 B 替换，直至达到如下位置

...	B	B	B	B	B	B	B	B	B	...
								$\uparrow$		
								q_0		

现在没有四元组可进行操作，故计算停止

这台 Turing：删除 1 的序列而后停机

例 7.14

Turing 机可复制输入纸带的内容

如输入纸带上除有限长的 0, 1 序列外都是空白，这样的 Turing 机可由下面的四元组集指定

$$\begin{array}{llll} q_0 & 0 & X & q_1 \\ q_1 & X & R & q_1 \\ q_1 & 0 & R & q_1 \\ q_1 & 1 & R & q_1 \\ q_1 & B & R & q_3 \\ q_3 & 0 & R & q_3 \\ q_3 & 1 & R & q_3 \\ q_3 & B & 0 & q_3 \end{array} \qquad \begin{array}{llll} q_0 & 1 & Y & q_2 \\ q_2 & Y & R & q_2 \\ q_2 & 0 & R & q_2 \\ q_2 & 1 & R & q_2 \\ q_2 & B & R & q_4 \\ q_4 & 0 & R & q_4 \\ q_4 & 1 & R & q_4 \\ q_4 & B & 1 & q_5 \end{array}$$

$$\begin{array}{llll} q_5 & 0 & L & q_5 \\ q_5 & 1 & L & q_5 \\ q_5 & B & L & q_5 \end{array}$$

$$\begin{array}{llll} q_5 & X & R & q_0 \end{array} \qquad \begin{array}{llll} q_5 & Y & R & q_0 \end{array}$$

$$\begin{array}{llll} q_0 & B & L & q_6 \\ q_6 & X & 0 & q_6 \\ q_6 & Y & 1 & q_6 \\ q_6 & 0 & L & q_6 \\ q_6 & 1 & L & q_6 \end{array}$$

例 7.15

计算 m 和 n 的积

q_0	1	X	q_1	q_5	B	L	q_5
q_1	X	R	q_1	q_5	Y	1	q_6
q_1	1	R	q_1	q_6	1	R	q_2
q_1	B	R	q_2	q_2	B	L	q_7
q_2	1	Y	q_3	q_7	1	L	q_7
q_3	Y	R	q_3	q_7	B	L	q_7
q_3	1	R	q_3	q_7	X	B	q_8
q_3	B	R	q_4	q_8	B	R	q_0
q_4	B	1	q_5	q_0	B	R	q_9
q_4	1	R	q_4	q_9	1	B	q_{10}
q_5	q	L	q_5	q_{10}	B	R	q_9

例（续）

例：输入

...	B	1	1	B	1	1	1	B	...

当以状态 q_0 开始读取最左边的 1 时，将最终停止在状态 q_9，纸带如

...	B	1	1	1	1	1	1	B	...

$\uparrow$

q_9

注

由例可见，Turing 的计算过程包含如查找、排序、复制、重复和存储（记忆）等常见程序

注

在 Turing 机计算过程中的任一步，纸带上只有有限部分是非空白的，方便起见，可通过瞬时描述来刻画机器，如

$$1 \quad 1 \quad B \quad 1 \quad B \quad q_2 \quad B \quad 1 \quad B \quad B \quad B \quad 1$$

包含纸带上所有非空白部分以及机器正在读取的方格，把状态符号也包括在序列中，放在正在读的符号的左边

瞬时描述亦可只包含必要的非空白部分（如当前状态所处某段非空白部分）

注 (Turing 机编码)

Turing 机由四元组的列表表示，通过类似 Gödel 数的编码方法，可对 Turing 机编码

- 编码四元组
- 编码四元组的有限列表

  $\Rightarrow$ 任何 Turing 机都可被编码
- 初始状态和输入输出的编码 (描述方法标准化)

  $\Rightarrow$ 与 Turing 机相关的全部信息都可被有效地编码
- 可选择编码使得不同的编码代表不同的 Turing 机

命题 7.16

Turing 机（编码的）集是能枚举的 ◇

命题 7.17 (枚举定理)

所有 Turing 机的集能枚举为 T_0，T_1，T_2，...，使得每个下标（自然数）能完全确定它所对应的机器的指令集 ◇

考虑 Turing 机计算数论函数的值，输入输出都是自然数

输入 $n \in D_N$ 可用一条纸带上的 n 个 1 表示（其余为空白），输出为停机时纸带上非空白符的数目

注

(a) 对每个 Turing 机可计算的（数论）函数 f，有一台字母表为 $\{B, 1\}$ 的 Turing 机计算 f 的值

$\Leftarrow$ 把 B 写成 0，（有限）符号可用二进制编码

(b) 对每个 Turing 机可计算的（数论）函数 f，有一台只有两个（内部）状态的 Turing 机计算 f 的值

$\Leftarrow$ 字母表多于两个的符号通过二进制编码可规约成两个（符号，状态）

定义 7.18 (Turing 可计算)

一个数论（偏）函数是 Turing 可计算的，若有一台 Turing 机按照上面约定的输入输出方式计算它的值 ◇

命题 7.19

对每个 Turing 可计算的（偏）函数 f，有无穷多的 Turing 机可以计算 f 的值 ◇

注

不同的 Turing 机具有不同的指令集，但可计算同一个函数

Turing 论题

Turing 可计算（偏）函数类等同于算法可计算（偏）函数类

任何计算（偏）函数 f 的值的算法都可翻译成一个计算 f 的值的 Turing 机的四元组集

命题 7.20

一个数论（偏）函数是 Turing 可计算的当且仅当它是一个递归（偏）函数 $\diamondsuit$

注

Turing 论题等同于 Church 论题，亦称 Turing-Church 论题

$\Rightarrow$ 不同的计算模型之等价性

考虑一个算法（计算二元偏函数值）

对任意给定 $m, n \in D_N$，枚举一个 Turing 机列表 T_0，T_1，T_2，$\cdots$ 直至找到 T_m，用 T_m 对输入 n 进行计算

据 Turing 论题，存在一个 Turing 机计算二元偏函数值

命题 7.21 (通用 Turing 机)

存在一台通用（Turing）机，即有一台 Turing 机 T，计算二元 m 和 n 的函数的值，亦即对 T_m 输入 n 的计算结果 $\diamondsuit$

通用机包含（计算一元函数的）T_0, T_1，T_2，$\cdots$

考虑如下算法

　　对任意给定 $n \in D_N$，在递归偏函数列表中找到 ϕ_n，再（使用 T_n）计算 $\phi_n(n)$，并对计算结果加 1 赋予该计算结果（赋值语句）

据 Church 论题，该算法定义一个递归偏函数 ϕ_{k_0}

问题

$\phi_{k_0}(k_0)$ 的计算结果？

基于算法，似乎有矛盾

$$\phi_{k_0}(k_0) = \phi_{k_0}(k_0) + 1$$

但其实不矛盾，因 $\phi_{k_0}(k_0)$ 的计算永不停止

命题 7.22

不能枚举所有一元递归（全）函数 $f_0, f_1, f_2 \cdots$ ◇

停机问题（Halting Problem）：判定一个 Turing 机计算是否停机

命题 7.23 (停机问题)

Turing 机的停机问题是不可解的，即没有算法回答下面问题

$$\{\textit{Turing}\text{ 机 } T_m \text{ 对输入 } n \text{ 是否停机？} \mid m, n \in D_N\}$$

◇

令

$$\phi(n) = \begin{cases} 1, & \text{若 } T_n \text{ 对输入 } n \text{ 停机} \\ \text{无定义}, & \text{其他情况} \end{cases}$$

$\phi(n)$ 是偏函数，其定义域记为 K

$$K = \{n \in D_N : T_n \text{ 对输入 } n \text{ 停机}\}$$

由 命题 7.23 可知（不可解），K 不是一个递归集

命题 7.24

K 是递归可枚举的非递归集 ◇

命题 7.25

对任意 Turing 机 T，T 的定义域，即对所有 $n \in D_N$ 使得 T 在输入 n 停机的集合，是递归可枚举的（亦是递归的） ◇

注

与 命题 7.24 不同，是对同一 T 的不同输入

定义 7.26 (归约)

称集合 A 可归约（reducible）到集合 B，若存在一个算法判定 B 的隶属元素就能保证存在一个算法判定 A 的隶属元素

例 7.27

(1) $\{n \in D_N : T_n$ 对每个输入都停机$\}$ 既不是递归的，也不是递归可枚举的

(2) $\{n \in D_N : T_n$ 对每个输入都不停机$\}$ 既不是递归的，也不是递归可枚举的

(3) 对每个固定的 n_0，$\{n \in D_N : T_n$ 对输入 n_0 停机$\}$ 不是递归的，但是递归可枚举的

以上例子都是递归不可解的

(4) $\{n \in K?| \; n \in D_N\}$

(5) $\{T_n$ 对每个输入都停机 $| \; n \in D_N\}$

(6) $\{T_n$ 对有些输入停机 $| \; n \in D_N\}$

(7) $\{T_n$ 对输入 n_0 停机 $| \; n \in D_N\}$，其中 n_0 是任意固定数

这些问题都是递归不可解的

定义 7.28 (字问题)

令 $A = \{a_1, \cdots, a_k\}$ 是一个符号集，作为字母表，字（word）是字母表中有限符号的串

记 S_A 为 A 中所有字的集合，空字 $e \in S_A$（没有任何符号）

对任两个 S_A 中的字，可通过将第二个添加到第一个之后生成一个组合字，此操作称为拼接

拼接满足结合律，S_A 可看成在拼接操作下的一个半群（群的推广，满足结合律），空字是半群的单位元

字问题：对半群 S_A 做的各种操作以满足各种关系

例 7.29

考虑 S_A，a_1a_2 与 a_2a_1 相同

定义一个 S_A 上的等价关系如下

- 若 P 和 Q 都是字，则 $Pa_1a_2Q \sim Pa_2a_1Q$ 和 $Pa_2a_1Q \sim Pa_1a_2Q$
- 对任何字 U 和 V，U 等价于 V，若有一个字序列 $W_1, \cdots, W_n$ 使得 $U = W_1$，$W_1 \sim W_2, \cdots, W_{n\text{-}1} \sim W_n$ 且 $W_n = V$

  ($\sim$ 不满足传递律不是等价关系)

该等价类的集 S_A^* 构成一个半群

$\Leftarrow$ 有生成元 $a_1, \cdots, a_k$ 和 关系 $a_1a_2 = a_2a_1$

S_A^* 上的字问题：对任意给定的字 U 和 V，判定它们是否等价，即它们所表示的是 S_A^* 中的相同元素

基于 S_A，考虑一组关系

$P_1 = Q_1$，$P_2 = Q_2$，$\cdots$，$P_m = Q_m$，P_i，$Q_i \in S_A$，$i = 1$，2，$\cdots$，m

定义 $\sim$ 为 $PP_iQ \sim PQ_iQ$，对任何字 P，Q，$1 \leq i \leq m$

如上述可定义等价关系

该等价类称为具有生成元 $a_1, \cdots, a_k$ 和关系 $P_i = Q_i \ (1 \leq i \leq m)$ 的半群

定义 7.30 (有限可表示)

一个半群是有限可表示的，若它可用有限的生成元集和关系集生成

一个有限可表示半群的字问题是递归可解的，若有一个算法判定任意字对是否等价

半群的字问题：给定任意有限可表示半群及其任意一对字，是否存在一个算法判定它们是否等价 $\diamond$

Turing 机和有限可表示半群

定义 7.31 (瞬时描述与字)

Turing 机的瞬时描述由一串符号构成，例如

$$1\;B\;1\;1\;B\;q_i\;1\;B\;1 \qquad 1\;1\;1\;1\;B\;1\;B\;q_i\;B$$

视之为字，并认为两个字等价当且仅当其中一个可通过一系列 Turing 机操作变换为另一个

令字母表 A 包含 Turing 机所有纸带符号和状态符号，半群关系由 Turing 机的四元组给出

仅有部分 A 中的符号串有 Turing 机瞬时描述的形式（没有影响）

例 7.32

若 $q_1\ 1\ B\ q_2$ 是一个四元组，则

$$P\ q_1\ 1\ Q \sim P\ q_2\ B\ Q,$$

其中，P 和 Q 是 A 中任意符号串

命题 7.33

令 T 是一台 Turing 机，它停机当且仅当输入 α 属于集合 K（K 是递归可枚举但不是递归的） $\diamond$

命题 7.34

存在一个其字问题为递归不可解的有限可表示半群 $\diamond$

引理

T 对输入 α 停机当且仅当 $h\ q_0\ \alpha\ h$ 等价于 $h\ q'\ h$ $\diamond$

命题 7.35

半群的字问题是递归不可解的 $\diamond$

定义 7.36 (有限可表示群)

一个**有限可表示群**类似有限可表示半群定义，其（形式）逆可在字中出现（即由符号 $a_1, \cdots, a_k, a_1^{-1}, \cdots, a_k^{-1}$ 构成），且关系须对任意 $1 \leq i \leq k$ 满足 $a_i a_i^{-1} = e$ 和 $a_i^{-1} a_i = e$，e 为空字 ◇

群的字问题更加复杂

命题 7.37

(1) 存在其字问题递归不可解的有限可表示群

(2) 群的字问题是递归不可解的

(3) Abel 群的字问题是递归可解的

(对一个有限可表示 Abel 群，对每对字母表中的符号 a_i，a_j，关系集须包括 $a_i a_j = a_j a_i$)

$\diamondsuit$

- 算法
- 可计算性
- 不可判定性
- 计算复杂性
- 定理机器证明
- 计算逻辑
- 智能逻辑

不可判定性

回顾

命题演算形式系统 L 是可判定的

命题 7.38

L 中定理的 Gödel 数集是递归集 $\diamondsuit$

注

任何形式系统的可判定性都可通过 Gödel 配数变成是否某个 D_N 的子集是否具有递归性

问题

一阶逻辑（谓词演算形式系统）$K_{\mathscr{L}}$ 是否可判定？

$K_{\mathscr{L}}$ 的可判定性取决于一阶语言 $\mathscr{L}$

令 $\mathscr{L}_1$ 是不含函项符和常元，且只有一个谓词符 A_1^1 的一阶语言

命题 7.39

$K_{\mathscr{L}_1}$ 是递归可判定的 $\diamondsuit$

命题 7.40

令 $\mathscr{L}$ 是不含函项符或常元，且只含一元谓词符（可能有无穷多个）的一阶语言，则 $K_{\mathscr{L}}$ 是递归可判定的 $\diamondsuit$

如上 $K_{\mathscr{L}}$ 即为纯谓词演算，以突出它缺乏函项符和个体常元

命题 7.41

算术系统 $\mathscr{N}$ 是递归不可判定的（假设它是一致的） $\diamondsuit$

一个系统及其扩充的递归可判定性并非是必然相关的

一个递归集可有非递归子集；一个非递归集可有递归子集

命题 7.42

令 S 和 S^+ 是同一语言的一阶系统，S^+ 是 S 的有限扩充，即把包含公式 $\mathscr{A}_1, \cdots, \mathscr{A}_n$ 的有限集加入 S 的公理集所获得 S^+ 的公理集

若 S^+ 是递归不可判定的，则 S 也是递归不可判定的 $\diamondsuit$

命题 7.43 ($K_{\mathscr{L}}$ 不可判定性)

存在一个一阶语言 $\mathscr{L}$ 使得 $K_{\mathscr{L}}$ 是递归不可判定的 $\Diamond$

推论 7.44

完整的一阶谓词演算是递归不可判定的 $\Diamond$

命题 7.45

(1) 以下系统是递归不可判定的

  (a) 一阶群理论

  (b) 一阶环理论

  (c) 一阶域理论

  (d) 一阶半群理论

  (e) ZF 系统

(2) 以下系统是递归可判定的

  (a) Abel 群的一阶理论

  (b) 没有乘法的一阶算术

  (即与 $\mathcal{N}$ 相同但不包含符号 f_2^2，并省掉公理 $(N5)$ 和 $(N6)$ 的系统)

$\diamondsuit$

定义 7.46 (半可判定性)

一个形式系统 S 是半可判定的，若存在一个算法判定下集的问题

$$\{\mathscr{A} \text{ 是一个定理?} \mid \mathscr{A} \text{ 是 } S \text{ 的一个公式}\}$$

但不存在一个算法判定下集的问题

$$\{\mathscr{A} \text{ 不是一个定理?} \mid \mathscr{A} \text{ 是 } S \text{ 的一个公式}\}$$ ◇

可判定 $\Rightarrow$ 半可判定（反之不然）

递归可枚举集 $\iff$ 半可判定

Turing 机的停机问题是半可判定的

命题 7.47

（完整的）一阶逻辑（谓词演算）是半可判定的 ◇

- 算法
- 可计算性
- 不可判定性
- 计算复杂性
- 定理机器证明
- 计算逻辑
- 智能逻辑

计算复杂性

随堂讲解

- 算法
- 可计算性
- 不可判定性
- 计算复杂性
- 定理机器证明
- 计算逻辑
- 智能逻辑

定理机器证明

随堂讲解

- 算法
- 可计算性
- 不可判定性
- 计算复杂性
- 定理机器证明
- 计算逻辑
- 智能逻辑

计算逻辑

随堂讲解

- 算法
- 可计算性
- 不可判定性
- 计算复杂性
- 定理机器证明
- 计算逻辑
- 智能逻辑

随堂讲解