计算概论

《什么是“计算”》

裘宗燕

北京大学数学学院信息科学系

2015, 春季

“什么是”的问题

- 什么是“计算”？

  这个问题好回答吗？

- 什么是“数学”？
  - 什么是“物理”？
  - 什么是“……”？
- 它研究什么

  研究对象，研究范围，相对比较确定。但，能说清楚吗？

- 它的理论体系、研究的方法和技术

  一个学科有知识积累，不断发展和变化，包括逐渐扬弃了许多过时的东西，或后来认识到不正确/无意义的东西

什么是“数学”

- 研究范围？

  数学家或者组织是否给出了明确的界定？

- 恩格斯《自然辩证法》：数学是研究现实世界中的数量关系和空间形式的一门科学（**1873-1895**）

- **Wiki: Mathematics is the abstract study of topics such as quantity (numbers), structure, space, and change.**

- 百度百科：“数学是研究数量、结构、变化以及空间模型等概念的一门学科”。（是上面 **wiki** 的翻译，不准确，过度确定）

- 分析这些说法，清晰性/局限性/区分能力？

  关心数学的同学可以找找其他定义，或给出自己的看法。关心科学的同学也可以看看其他学科的情况

什么是“数学”

Aristotle defined mathematics as "the science of quantity", and this definition prevailed until the 18th century. Starting in the 19th century, when the study of mathematics increased in rigor and began to address abstract topics such as group theory and projective geometry, which have no clear-cut relation to quantity and measurement, mathematicians and philosophers began to propose a variety of new definitions. Some of these definitions emphasize the deductive character of much of mathematics, some emphasize its abstractness, some emphasize certain topics within mathematics. Today, no consensus on the definition of mathematics prevails, even among professionals. There is not even consensus on whether mathematics is an art or a science. A great many professional mathematicians take no interest in a definition of mathematics, or consider it undefinable. Some just say, "Mathematics is what mathematicians do."

-- wikipedia

数学与计算

- 数学研究的对象是自然事物的抽象
  - 数量、形状、结构，空间和变化等等
  - 或者是这些的再抽象
- 计算是一种过程
  - 按一种特殊的方式，操作一些抽象的对象（例如数字，或其他符号），得到所需的抽象结果
  - 是一类人为过程的汇集和抽象
  - 没发现它在自然界的背景（或许没有？至少是很难找到）
  - 很清晰，其中一些东西属于数学，另一些东西更实际

计算的需要

- 计算是从人类的社会生活凝结出的一种需要

  分配，计划，安排，记录，等等

  - 人们需要考虑计量，度量，方位等
  - 进一步需要：季节和生产，交换，等等
- 最基本的计算工具是人的头脑，
  - 逐渐辅佐以自然的工具，例如：手指、石块、划痕等
  - 进一步发展出专用的辅助计算工具
- 例如中国早期的算筹，

计算工具的发展

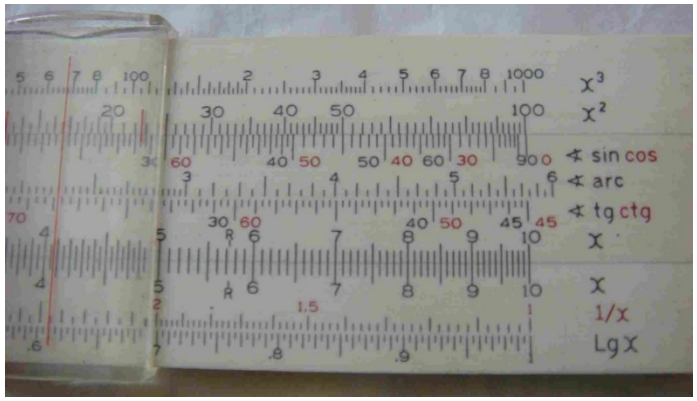
- 中国古代计算工具发展的顶峰算盘

  - 数字化，数据表示是离散的
  - 最基本的操作是上下拨动算珠
  - 一套基本操作（由珠算口诀表示）实现基本算术计算
    - 六上六，六去四进一，六上一去五进一，等
    - 根据遇到的情况（条件）选择使用的口诀

- 古代算法实例：2300年前战国《算表》包括21支竹简。描述的方法可将乘法变为加法，完成除法和开方，快速计算100以内整数乘积，还能计算包含分数“半”的两位数乘法。其计算范围超过以前发现的“里耶秦简九九表”和“张家界汉简九九表”等古代乘法表

计算工具的发展

- 国外计算工具发展的重要里程碑是计算尺（17世纪初），在工程领域沿用到 20 世纪中后期



- 模拟表示（用长度和位置表示数值）
  - 对数的思想，用加减表示乘除，是重大发明
- 在科技发展中起过重要作用。现已完全淘汰，变成“收藏”

  虽然连续数学继续蓬勃发展，但采用连续表示的计算却已经彻底败给了离散表示（数字化）的计算

计算工具的发展

- 1642年物理学家帕斯卡发明机械齿轮式加减法器，1673年数学家莱布尼兹发明乘除器。1820年出现商品机械计算器

- 机械计算器被长期用于各种工程和科学计算
  - 科技工作者或专职计算员拿着细致的计算清单，按明确说明的步骤一项项输入和操作，记录得到的结果
  - 美国陆军弹道实验室在二战中雇佣大批妇女计算火炮弹道，发明电子计算机就是为代替她们（computer）

计算工具的发展

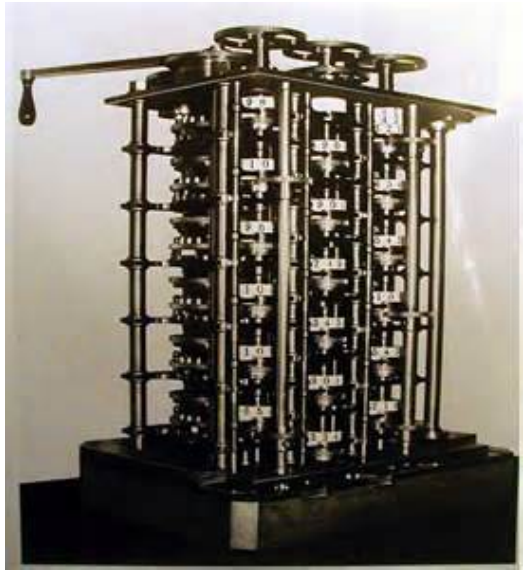
- 早期辅助计算工具的功能只是记录数据
- 后来的工具（如机械计算器）还能帮助产生计算的结果
  - 操作员只需要做一系列很简单的动作
  - 可以在需要时从工具中读出结果
  - 辅助完成一些特定的计算，即使使用者不知道怎么做这些计算，也完全不需要知道计算什么
- 到20世纪前期，所有计算工具都需要人一步步推动
  - 计算过程不是自动的
- 另一方面，钟表是自动化信息处理工具的重要里程碑
  - 可以认为钟表一直在计数，并计算整除和取模
  - 完全是自动进行的

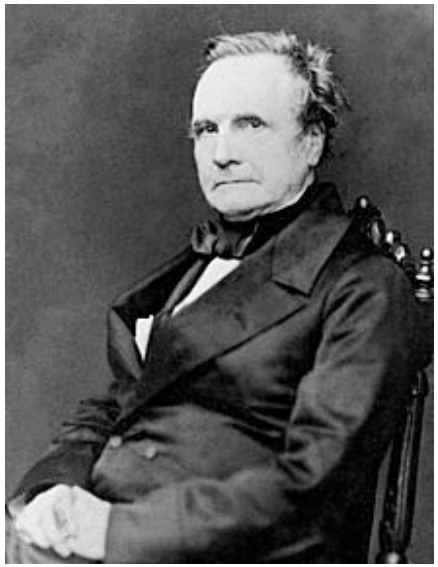
计算工具的发展

- 工业化提供了动力机械
  - 出现了大量动力驱动的设备（自动设备）
  - 人们自然会考虑，如何将动力与基本机械式计算设备结合，做出自动化的计算机器

- 这方面发展的顶峰是英国发明家 **Charles Babbage** 的差分机

  能计算多项式函数





计算的模型（计算的数学理论）

- 数学家一般认为计算不是一般的数学，似乎问题更简单

  希望为计算建立抽象（而且精确）的模型

- 朴素的认识：计算是一种过程，
  - 从一些初始符号开始，通过操作得到一些符号结果
  - 只有有穷种不同的基本操作，规则清晰明确
    - 如十进制加法，不同数字相加的规则 **55** 条
    - 算盘，上/下拨算珠（两个基本动作），或珠算口诀
  - 一系列明确的步骤形成任意步数的操作过程
    - 操作进行的过程事先有严格规定，可以机械地进行
- 允许有穷操作的组合（基本操作的组合能行），例如：

  基于加法定义乘法，完成辗转相除的过程，都是计算

计算模型（计算的数学理论）

- 人们把这种（基于朴素直观认识的）过程称为能行过程（**feasible procedure**，或有效过程，**effective procedure**），特点（有别于其他智力活动）
  - 操作者在过程进行中不需要思考（是机械的）
  - 按部就班，严格根据指令一步步操作，就能完成
- 这些回答了前面的一个问题：
  - 计算是否需要思维（“智能”）？
  - 回答：计算的实施不需要思维。计算的每步都是照章办事，不用想，没有任何灵活性（傻子也会做！）
- 这也使前面提出的另一个问题更难回答：

  从严格描述并严格执行的活动中能产生出“智能”吗？

计算模型（基本考虑）

- 直观计算应该有输入/输出（计算对象/结果），一个机器做计算

  确定输入和输出符号集（可以是同一个集合）

机器读输入写输出，计算过程：

- 在输入带写好输入（一串符号）
- 启动机器运行
- 在输出带查看计算结果

基本执行循环：

- 读一个符号
- 输出字符（可选）
- 移动读写头（可选）

计算模型（1）

- 注意：

  一个计算模型表示一类计算机器，它们具有相同的结构和行为特征，一部具体机器完成一种特定“计算”

- 最简单的模型是有穷对照表机器：机器内部有一个对照表
  - 反复读入，一次一个符号
  - 查对照表输出
  - 读入和输出后读写头都向右移动一格
- 机器实例：**0 → 9, 1 → 8, …, 8 → 1, 9 → 0**
  - 完成十个数字的对调
  - 在这部机器的输入带上放一个十进制数字序列，计算完成时输出带有另一个十进制数字序列

计算模型（1）

- 特点：有穷描述，一个对照表实现一个机器
  - 操作是机械的，确定的，
  - 可以处理任意长的符号序列
- 对照表机器的简单变形（输出不必与输入等长）：
  - 允许读入某符号时连续输出几个符号
  - 允许读入某符号时不输出
  - 等等
- 对照表机器能力有限，很多事不能做
  - 一个符号对应另一符号，与前面的其他输入无关。这一特征大大限制了对照表机器的计算能力
  - 如果想扩充机器的能力，看来需要能记录一些历史

计算模型（1’）

例：希望设计一个十进制正整数减一机器

十进制数从低位到高位按从左到右的顺序排列

- 计算规则有两条：

  1. 遇 0 输出 9，继续考虑十进制数其余部分减一
  2. 遇 d 输出 d-1，而后原样拷贝其余数字直至结束

  显然这个过程完全是机械的

  但是，不存在实现这个计算的对照表机器：任何对照表机器遇到同一符号总是做同一个动作，不会做两种不同动作

- 注：进制数从高位到低位写是习惯写法（约定俗成）

  如果输入带上的数据采用常规描述方式，就应该从右到左逐位计算（允许读头/写头左移）

计算模型（1’）

- 分析计算减一的问题，其中存在两种情况
  - 第一种情况下读 0 输出 9，读到其余数字 d 输出d-1
  - 第二种情况下读到任何 d 都输出 d（原样拷贝）
  - 机器启动后处于第一种情况
  - 在第一种情况下遇到非 0 并输出后进入第二种情况
- 这里出现了不同“情况”（历史），需要反映在计算模型中
- 抽象：为表示不同“情况”，需要为机器引入状态机制
  - 一部机器有若干个状态，初始时处于某个状态
  - 运行中机器状态可能由于遇到某些输入而改变（状态转移）
  - 状态和输入共同决定输出（或其他动作）

计算模型（2，FSA）

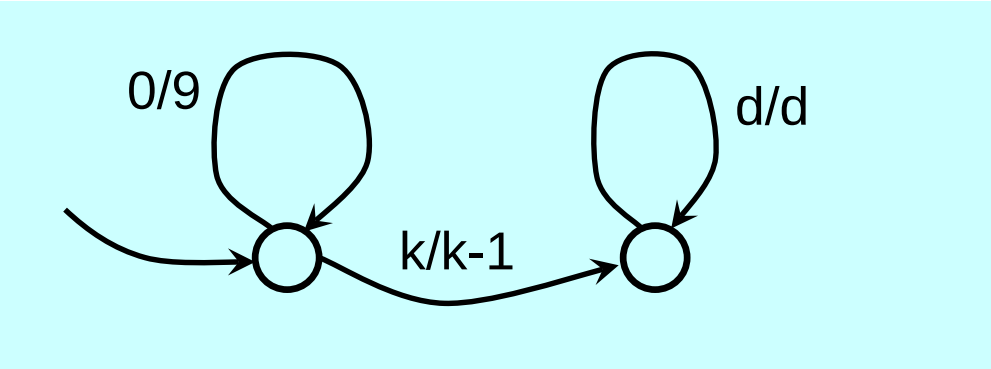
- 如果一部机器的状态有穷，就是有穷状态自动机，**FSA**
  - 对照表机器是 **FSA** 的特殊情况，其中只有一个状态
  - **FSA** 是一种重要的计算模型，在计算中有许多应用
- **FSA** 的一种直观图示
  - 小圆圈表示状态；
  - 状态间的带箭头弧线表示状态转移，弧线上用 **a/b** 的形式标明输入符号 **a** 就输出符号 **b**，状态不变的转换用从一个状态到自身的弧线表示
  - 用一个无源箭头指明初始状态
  - 可以有一类称为终止状态的状态，计算结束时机器位于这种状态，就表示计算成功完成

计算模型（2，FSA）

- 完成十进制正整数减一的有穷状态自动机

  弧线旁 **n/m** 表示读到符号 **n** 时做这个转移并输出 **m**

  下图有简化，**k/k-1** 表示 **9** 条弧线，分别标 **1/0, 2/1, ..., 9/8**，**d/d** 表示 **10** 条弧线，分别标 **0/0, 1/1, 2/2, ..., 9/9**



- 有穷状态自动机（有穷自动机）是一种简单计算模型

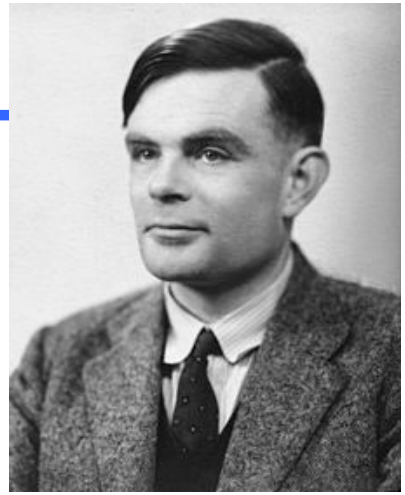
  在计算机科学技术领域有广泛的应用

计算模型（2，FSA）

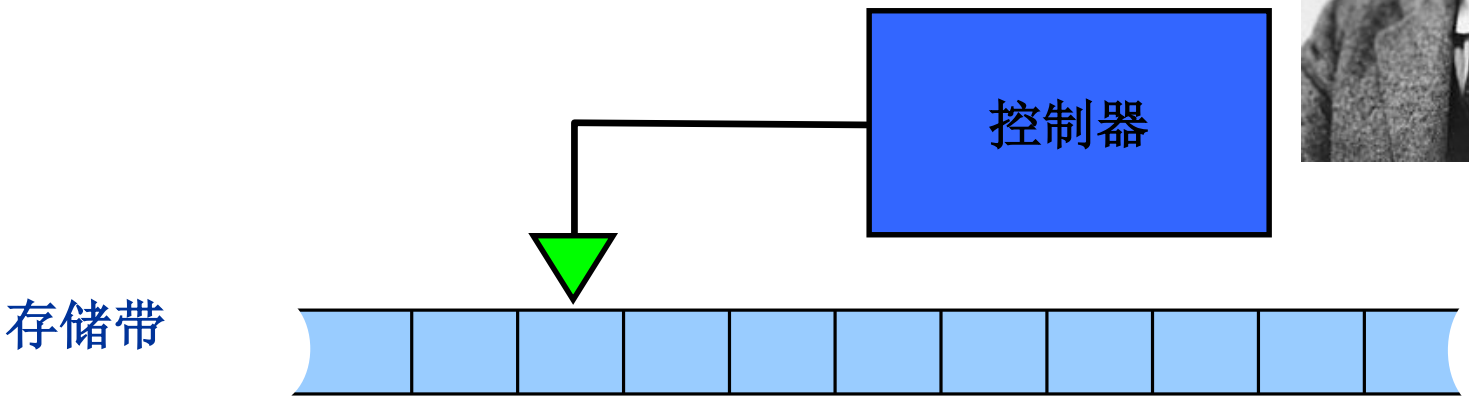
- FSA 状态有穷，每个 FSA
  - 具有固定个数的状态，例如 n
  - 能表示的不同状态变化情况受到 n 的限制
  - 许多人们都认为是计算的过程无法用 FSA 实现
  - 说明 FSA 的描述能力不够强
- 可以考虑设法扩充已有的计算模型或者建立新的计算模型，使得到的模型具有更强的计算能力

  这样做时，还要求不越出直观“计算”（能行过程）的范围
- 随之产生的问题：
  - 这种扩充有限度吗？
  - 是否存在一种“终极的”计算模型？

计算模型（3，图灵机）



- 英国数学家 Turing 1936 年发表文章，提出了后来被称为“图灵机”的计算模型，如下图所示



- 两端无穷的存储带包含无穷多格子，每格可存储一个符号
- 控制器是一个有穷状态机器，它根据从存储带读到的符号和当前状态确定自己的一步动作
- 每步动作包括：可以转换控制器状态，可以在读头处写一字符，可以指挥读头向左或向右移动一格

计算模型（3，图灵机）

- 构造一部具体的图灵机 T，需要
  - 选定一个有穷字符集 A 和一个状态集合 S
  - 定义一个状态转换函数

$$\theta : S \times A \rightarrow S \times (A \cup \{\_\}) \times \{left, right, \_\}$$

left/right 表示读写头移动方向，_ 表示不输出或不移动

- 图灵机 T 表示的计算：
  - 将 A 中的一个符号序列 I 放在存储带，读写头置于其左端
  - 令 T 运行，直到再到无转换规则可用时 T 的运行终止，当时存储带上的序列 O 就是计算结果
  - 对 A 上每个 I，可能得到一个 O，T 完成的计算可看作是函数 T(I) = O。这是部分函数，因为 T 对 I 的计算可能不终止

计算模型（3，图灵机）

- 可以用具体实例说明，对于很多复杂的计算问题，我们都能构造出完成该计算的图灵机
- 由于有无穷长的存储带
  - 初始串可任意长（但有穷）
  - 可以存储任意多的中间结果
  - 可能产生任意长的计算结果
- 各种直接扩展，如增加存储带，增加读写头等，都不能得到更强的计算模型。扩展后的模型都可以用图灵机模拟
- 认识到这些情况后，图灵进一步考虑：
  - 这种计算模型是否已经足够强？或者已经是“终极模型”？
  - 但，怎么“展示”这种模型非常强（或者“最强”）呢？

编码

- 这里就用到了计算的另一支柱：编码（前面讨论过，简单复习）
- 理论上说，任一类可以用有限个符号表达的结构，都可以设计一种编码方式，其中
  - 选用一个包含至少两个符号的集合 **R**
  - 定义一个表示函数 **[.]**，可能还有一个解释函数 $[.]^{-1}$
  - 使对属于该类的任一 **S**，**[S]** 是 **R** 中符号的序列。（如果有解释函数，则 $[[S]]^{-1}$ 与 **S** 有某种“等价关系”）
- **Gödel** 定义了一种从逻辑公式到整数的编码，编码产生的整数称为 **Gödel** 数，这种技术称为 **Gödel** 编码
  - 推而广之，通过有穷符号和一组规则定义的结构，都可以用 **Gödel** 的方法编码为整数
  - 整数可以作为计算的对象，然后 ……

图灵机的能力有多强？

- 图灵的想法是：给出（一个）例子，说明能在图灵机模型的范围里，可以定义出功能极为强大的机器
- 图灵实际考虑的问题是：

  设法用一部图灵机完成所有图灵机有可能完成的工作

  原因：每部图灵机完成的工作都是计算，如果没有一部图灵机能模拟所有这些计算，这个模型可能还不够强
- 基于这种想法，图灵提出了通用图灵机的概念。他设计了一种图灵机编码方式 **P(T)**，并证明存在（构造了）一部图灵机 **U** 使得
  - 对于任意一部图灵机 **T**，把 **P(T) + I**（即，**T** 的编码和送给 **T** 的输入 **I**）放在 **U** 的存储带上
  - **U** 对 **P(T) + I** 计算出 **O** 当且仅当 **T** 对 **I** 计算出 **O**
  - 注意：一台机器 **U**，一种统一编码方式 **P**，模拟所有图灵机

“一”就是“一切”

- 如前所述，编码图灵机的符号集只需包含至少两个符号
  - 为图灵机确定一种编码方式
  - 通用图灵机的设计与编码方式有关
  - 实际上，对任何良好的编码，都能做出对应的通用图灵机
- 通用图灵机的存在性，说明了：
  - 用一部图灵机就能模拟所有的图灵机，解释所有的图灵机的行为。（存在一部通用图灵机，就存在无穷多部）
  - 这意味着，图灵机模型具有一种完备性和自封闭性
  - 例如，完全可以用 **U** 解释 **U** 自身，为此只需把 **U** 的编码 **P(U)** 和它处理的 **P(T) + I** 一起放在 **U** 的带上。即，**U** 也能模拟自己。这种模拟还可以继续做下去

    因为 **U** 也是图灵机，编码 **P** 对它有效

计算与图灵机

- 回到初始话题：能行过程/有效过程与图灵机有什么关系？
- 图灵定义了图灵机可计算（函数）的概念：存在一部图灵机实现该（函数的）计算
- 图灵论题：任何能行可计算（函数）都是图灵机可计算的

  又称 **Church-Turing Thesis**（丘奇-图灵论题）

  只是一个“猜想”性的论断（图灵猜想），无法证明

  但可以证否，也可以添加更多证据
- 图灵论题断言：
  - 不可能做出一个比图灵机更强的计算模型
  - 图灵机不能做的事情（有吗？这是一个问题，下面讨论），就不可能“机械地完成”

计算与图灵机

- 许多研究工作“旁证”了图灵论题难以被证否
- 人们提出了许多计算模型，并证明了其中最强大的模型都与图灵机的计算能力等价。例如：
  - λ-演算：由 A. Church（图灵的博士导师）比图灵机稍早提出，基于函数抽象和函数应用，在计算理论中仍然非常重要

    其中也有“通用函数”的概念，但没引起足够重视
  - 递归函数： Gödel，S. Kleene 的工作
  - Post 系统：E. Post 的工作
  - 一阶逻辑：数理逻辑，描述
  - 寄存器机器：一种更接近计算机的抽象机器模型
  - Markov 算法：一种字符串重写系统

图灵机理论告诉我们……

- 图灵论题告诉我们
  - 如果考虑计算，只需考虑与图灵机（能力）等价的计算系统
  - 实现自动计算工具，只需考虑与图灵机等价的模型
- 通用图灵机的结论告诉我们：
  - 无须为每种计算实现一部自动机器（潜在的机器无穷多）
  - 只需要开发一种与通用图灵机等价的机器
  - 和一种机器编码技术，用它编码具体的机器
- 这种机器已经造出来了，通用/电子/数字/计算机

  已开发了大量相当于图灵机编码的东西：程序
- Python 解释器就是一部通用机器，我们的程序是专用机器

  每种高级语言的语言系统，也是一部通用机器

通用性

- 我们有一些“专用的通用设备”
  - 磁带录放机
  - 通用机床，等
- 计算机（一个编程语言就是一部抽象的通用计算机）比它们更进了一步：它能模拟自己
- 请想一想：
  - 在其他领域中有没有具有类似通用功能的东西？
  - 例如：电子线路？汽车？机床？催化剂？书？药？
  - 有可能吗？
- 但计算机什么都能做吗？

不可计算

- 图灵机模型的表达能力很强，那么其能力有边界吗？
  - 例如：是不是每个整数函数都能用图灵机计算？
- 很容易证明：存在图灵机不可计算的整数函数
  - 图灵机可以用整数编号（能编码就是证据）
  - 用对角线法，可以做出不可计算函数的存在性证明
  - 根据集合的基数（可数、不可数）也可得到这一结论
- 图灵给出了第一个具体的不可计算问题（停机问题）：
  - 问题：判断任一部图灵机对任一个输入是否停机
  - 图灵证明了停机问题是不可计算的（也用反证法）
  - 人们已找到许多不可计算问题，常用证明方法就是把已知不可计算问题（如停机问题）归结到要证的问题

停机问题证明

- 用反证法，设存在图灵机（等价的，Python函数）S(T, I)，在 T 对 I 终止（停机）时 S 返回 1，不终止时返回 0
- 定义函数：

```
def NS(T):
    if S(T, T) == 1: # 把 T 自身作为 T 的输入
        while True: # 如果 T 对 T 终止就无穷循环
            pass
    # else: pass   # T 对 T 不终止时 NS 立刻结束
```

- 考虑 NS(NS)，将 NS 应用于 NS 自身
  - 若 NS(NS) 终止：那么 S(NS, NS) == 1，在函数 NS 定义里 if 的条件成立，执行进入无穷循环，NS 不终止，矛盾
  - 若 NS(NS) 不终止，NS 里 if 的条件不成立，终止，也矛盾
  - 两种情况都导致矛盾，所以 S 不可能存在

不可计算

- 不可计算问题说明了计算机的能力有限
  - 存在无穷多不能用计算机解决的问题
  - 如果确定了一个问题不可计算，就不要想去用计算机解决之，因为不可能写出解决它们的程序
- 不可计算问题的一个特点是过于一般（general，问题太大）
  - 因此不可能有一个程序统一地解决它
  - 但有可能找到该问题中的某些可计算的特殊子问题
  - 例如，所有程序的一般停机问题不可判定，但有些种类的程序，其停机问题可以判定
  - 对不可计算的问题，找出其有意义的可计算的子问题，经常也是很有意义的工作

“计算”及其研究领域

- 计算有了一个被广泛承认的“严格定义”
  - 计算就是一类特殊过程，图灵的理论划清了它的边界
  - 不仅说明了“什么是计算”，而且说明了什么不是
- “计算”的定义确定了本领域的研究对象，作为研究领域，这里研究与下面各项相关的理论和实际问题：
  - 计算的性质：计算模型，可计算性，计算复杂性等
  - 计算的描述：算法及其性质，编程语言及其理论等
  - 计算的实现：计算机结构和组织，其他实现可能性
  - 计算的应用：计算和其他学科的关系
  - 显然，这些不同方面联系紧密，相互交错
- 计算学科：研究计算的性质、描述、实现和应用的学科