第 1 章　形式语言与自动机概述

王　鑫

wangx AT tju.edu.cn

天津大学　智能与计算学部

Computer Science is no more about computers
than astronomy is about telescopes.

— Edsger Dijkstra

中国学科分类国家标准

- 520 计算机科学技术
  - 520.10 计算机科学技术基础学科
    - 520.1010 自动机理论
    - 520.1020 可计算性理论
    - 520.1030 计算机可靠性理论
    - 520.1040 算法理论
    - 520.1050 数据结构
    - 520.1060 数据安全与计算机安全
    - 520.1099 计算机科学技术基础学科
  - 520.20 人工智能
    - 520.2010 人工智能理论
    - 520.2020 自然语言处理
    - 520.2030 机器翻译
    - 520.2040 模式识别
    - 520.2050 计算机感知
    - 520.2060 计算机神经网络
    - 520.2070 知识工程（包括专家系统
    - 520.2099 人工智能其他学科
  - 520.30 计算机系统结构
    - 520.3010 计算机系统设计
    - 520.3020 并行处理
    - 520.3030 分布式处理系统
    - 520.3040 计算机网络
    - 520.3050 计算机运行测试与性能评价
    - 520.3099 计算机系统结构其他学科
  - 520.40 计算机软件
    - 520.4010 软件理论
    - 520.4020 操作系统与操作环境
    - 520.4030 程序设计及其语言
    - 520.4040 编译系统
    - 520.4050 数据库
    - 520.4060 软件开发环境与开发技术
    - 520.4070 软件工程
    - 520.4099 计算机软件其他学科
  - 520.50 计算机工程
    - 520.5010 计算机元器件
    - 520.5020 计算机处理器技术
    - 520.5030 计算机存储技术
    - 520.5040 计算机外围设备
    - 520.5050 计算机制造与检测
    - 520.5060 计算机高密度组装技术
    - 520.5099 计算机工程其他学科
  - 520.60 计算机应用（具体应用入有关
    - 520.6010 中国语言文字信息处理
    - 520.6020 计算机仿真
    - 520.6030 计算机图形学
    - 520.6040 计算机图象处理
    - 520.6050 计算机辅助设计
    - 520.6060 计算机过程控制
    - 520.6070 计算机信息管理系统
    - 520.6080 计算机决策支持系统

课程地位

理论计算机科学

课程地位

理论计算机科学

- 可计算性和计算复杂度理论
  - 只学过大 O 表示法吗？比如，$O(n^2)$
  - 系统的理论体系：研究生内容

课程地位

理论计算机科学

- 可计算性和计算复杂度理论
  - 只学过大 O 表示法吗？比如，$O(n^2)$
  - 系统的理论体系：研究生内容
- 算法分析与设计
  - 《数据结构》或《算法》课中的简单结论
  - 算法分析与设计理论：研究生内容

课程地位

理论计算机科学

- 可计算性和计算复杂度理论
    - 只学过大 O 表示法吗？比如，$O(n^2)$
    - 系统的理论体系：研究生内容
- 算法分析与设计
    - 《数据结构》或《算法》课中的简单结论
    - 算法分析与设计理论：研究生内容
- 程序设计理论
    - 程序正确性证明、自动程序设计、形式语义学等
    - 较为高深，本科没有介绍

课程地位

理论计算机科学

- 可计算性和计算复杂度理论
  - 只学过大 O 表示法吗？比如，$O(n^2)$
  - 系统的理论体系：研究生内容
- 算法分析与设计
  - 《数据结构》或《算法》课中的简单结论
  - 算法分析与设计理论：研究生内容
- 程序设计理论
  - 程序正确性证明、自动程序设计、形式语义学等
  - 较为高深，本科没有介绍
- 形式语言与自动机理论

课程内容

主线：4 类形式语言 和 4 种自动机

课程内容

主线：4 类形式语言 和 4 种自动机

- 引言（预备知识）
- 形式文法理论

课程内容

主线：4 类形式语言 和 4 种自动机

- 引言（预备知识）
- 形式文法理论
- 正则语言
  - 正则文法
  - 有限自动机
  - 正则表达式
  - 泵引理

课程内容

主线：4 类形式语言 和 4 种自动机

- 引言（预备知识）
- 形式文法理论
- 正则语言
  - 正则文法
  - 有限自动机
  - 正则表达式
  - 泵引理
- 上下文无关语言
  - 上下文无关文法
  - 下推自动机
  - 泵引理

课程内容

主线：4 类形式语言 和 4 种自动机

- 引言（预备知识）
- 形式文法理论
- 正则语言
  - 正则文法
  - 有限自动机
  - 正则表达式
  - 泵引理
- 上下文无关语言
  - 上下文无关文法
  - 下推自动机
  - 泵引理

课程内容

主线：4 类形式语言 和 4 种自动机

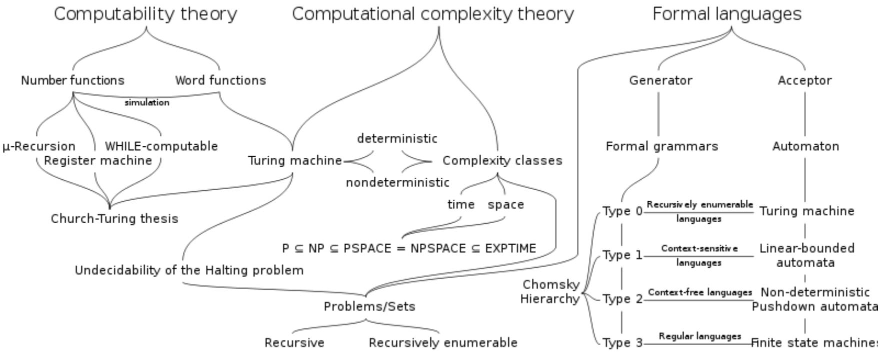
- 引言（预备知识）
- 形式文法理论
- 正则语言
  - 正则文法
  - 有限自动机
  - 正则表达式
  - 泵引理
- 上下文无关语言
  - 上下文无关文法
  - 下推自动机
  - 泵引理
- 上下文有关语言
  - 上下文有关文法
  - 线性有界自动机

课程内容

主线：4 类形式语言 和 4 种自动机

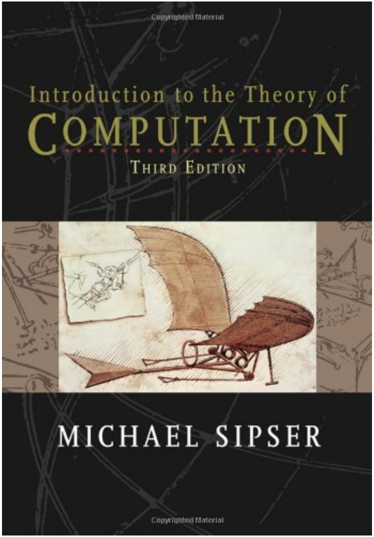
- 引言（预备知识）
- 形式文法理论
- 正则语言
  - 正则文法
  - 有限自动机
  - 正则表达式
  - 泵引理
- 上下文无关语言
  - 上下文无关文法
  - 下推自动机
  - 泵引理
- 上下文有关语言
  - 上下文有关文法
  - 线性有界自动机
- 递归可枚举语言
  - 短语结构文法
  - 图灵机
  - 邱奇-图灵论题

Theoretical Computer Science 理论计算机科学



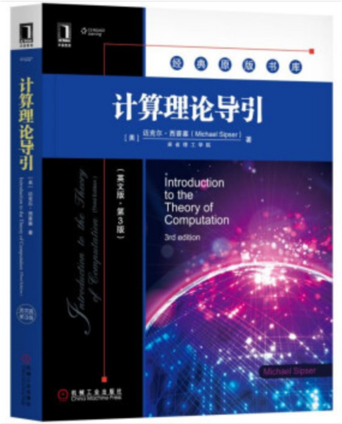
教材

- “Introduction to the Theory of Computation”
  - 作者：Michael Sipser
  - 出版社：Cengage Learning
  - 出版日期：第 2 版 2006；第 3 版 2012
  - 页数：第 2 版 437；第 3 版 458



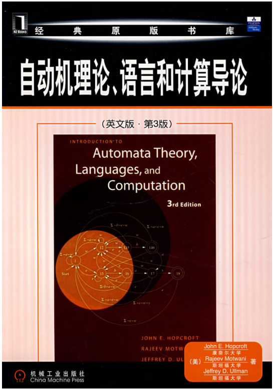
教材

- 《计算理论导引》（英文版 · 第 3 版）
    - 作者： Michael Sipser
    - 出版社： 机械工业出版社 影印
    - 出版日期： 影印 2018
    - 页数： 458



参考书

- 《自动机理论、语言和计算导论》（英文版 · 第 3 版）
  Automata Theory, Languages and Computation
  - 作者： Hopcroft, Motwani, Ullman
  - 出版社： 机械工业出版社 影印
  - 出版日期： 出版 2007， 影印 2008
  - 页数： 535



参考书

- 《形式语言与自动机》
  - 作者：陈有祺
  - 出版社：机械工业出版社
  - 出版日期：2008
  - 页数：227

参考书

- 《形式语言与自动机理论（第 3 版)》
  - 作者：蒋宗礼
  - 出版社：清华大学出版社
  - 出版日期：2013
  - 页数：286

Outline

1. Automata, Computability, and Complexity
2. Mathematical Notions and Terminology
3. Definitions, Theorems, and Proofs
4. Types of Proof

Outline

The Theory of Computation 计算理论

What are the fundamental capabilities and limitations of computers?

The Theory of Computation 计算理论

What are the fundamental capabilities and limitations of computers?

- This question goes back to the 1930s when mathematical logicians first began to explore the meaning of computation.

The Theory of Computation 计算理论

What are the fundamental capabilities and limitations of computers?

- This question goes back to the 1930s when mathematical logicians first began to explore the meaning of computation.
- Automata (自动机)
- Computability (可计算性)
- Complexity (复杂度)

Complexity Theory 复杂度理论

Computer problems

- Easier: e.g., the sorting problem
- Harder: e.g., the scheduling problem

Complexity Theory 复杂度理论

Computer problems

- Easier: e.g., the sorting problem
- Harder: e.g., the scheduling problem

What makes some problems computationally hard and others easy?

- This is the central question of complexity theory.
- We don't know the answer to it. (researched for over 40 years!)
  - An elegant scheme for classifying problems according to their computational difficulty (analogous to the periodic table)

Complexity Theory

Confront a computationally hard problem

1. Find which aspect of the problem is at the root of the difficulty.
    - Alter it so that the problem is more easily solvable.
2. Settle for less than a perfect solution to the problem.
    - Find solutions that only approximate the perfect one is easy.
3. Hard only in the worst case situation, but easy most of the time.
    - A procedure that occasionally is slow but usually runs quickly.
4. Consider alternative types of computation
    - Such as randomized computation.

Computability Theory 可计算性理论

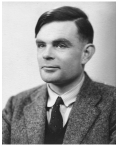
Certain basic problems cannot be solved by computers!

- Determine whether a mathematical statement is true or false.

Computability and Complexity are closely related.

- The objective of complexity theory
  - classify problems as easy ones and hard ones
- The objective of computability theory
  - classify problems as solvable ones and unsolvable ones

Kurt Gödel (1906–1978)



Alan Turing (1912–1954)

Alonzo Church (1903–1995)

Automata Theory 自动机理论

The definitions and properties of mathematical models of computation!

Models of computation

The theories of computability and complexity require a precise definition of a computer.

Automata Theory 自动机理论

The definitions and properties of mathematical models of computation!

Models of computation

The theories of computability and complexity require a precise definition of a computer.

- Models of computation
  - Finite Automaton (有限自动机)
  - Context-Free Grammar (上下文无关文法)
  - Others

Automata Theory 自动机理论

The definitions and properties of mathematical models of computation!

Models of computation

The theories of computability and complexity require a precise definition of a computer.

- Models of computation
  - Finite Automaton (有限自动机)
  - Context-Free Grammar (上下文无关文法)
  - Others

Theoretical models of computers help the construction of actual computers.

几则书评

 etone ★★★★★ 2008-03-08 04:38:47

我们为什么要学习计算理论

在所有我看过的计算理论、可计算性、计算复杂度的教材中，Sipser的这本Introduction to the Theory of Computation是最适合入门的。把计算理论这么个艰深的学问讲解得清晰简洁，直观易懂。而且涵盖了计算理论的各个经典内容。作为一本introduction，真是再好不过了。 计算理论... (加载中...)

 平凡的老鱼 ★★★★★ 2009-03-09 15:19:17 机械工业出版社2006版

很不错的关于计算理论的介绍

事知其然而后知其所以然。 现代计算机体系的构建，图灵机的数学模型的实现，正是指出了这道创世纪的光。 现在书里面的内容已经忘记的差不多了，只是记得不断的证明，一步步的证明，充满了智慧的光芒。 总之，是一本好的数学书。 (加载中...)

 陈炬 ★★★★★ 2009-03-19 11:11:59

CS学生必读书籍

让人了解计算机的本质，它的能力与它的局限性。 计算理论课的教材，上课上的很累，但很有收获。我觉得没读过这本书的不好意思说自己是Computer Science专业毕业的。 (加载中...)

几则书评

然而，尽管我对这本书和这门学问都很推崇，我对于学习计算理论的必要性却并不坚定。我自己喜欢这些，可我该如何向别人解释学习它是必要的?

Sipser在前言中也试图说明这个问题:"After all, isn't theory arcane, boring, and worst of all, irrelevant?"

他很认真的试图从几个方面说服学生，计算理论是“有用”的，但我总觉得这些说服很徒劳：书中的三个部分，对于搞研究的人来说，前两个领域已经或走到头了或不再是主流研究趣味了，只有复杂度尚活跃，但也只是个理论方向之一；而对于那些有志于业界工作的学生，后两个部分几乎永远不会在工作中用到，而只有第一部分的自动机，可能会用到一点点正则表达式。

看来，从“有用”这个方向去为计算理论辩护，难免会遭遇尴尬和勉强。

我能想到的理由就只剩两个：

（一）这些是计算机科学的根本，没有它们计算机科学不能算作是个正经学问，因此，一个自称计算机科学专业的人，应当知道这些。

（二）这些是美好的，值得在短暂的人生中去经历去见识。

我觉得这已是足够的理由了。

Outline

Sets 集合

A ***set*** is a group of objects represented as a unit.

$S = \{7, 21, 25\}$

- ***elements*** or ***members***: the objects in a set. $7 \in \{7, 21, 25\}$, $8 \notin \{7, 21, 25\}$
- ***subset***: $A \subseteq B$, proper subset: $A \subsetneq B$
- ***natural numbers***: $\mathcal{N}$, integers $\mathcal{Z}$
- ***empty set***: $\emptyset$
- $\{n \mid n = m^2 \text{ for some } m \in \mathcal{N}\}$
- ***union*** $A \cup B$, ***intersection*** $A \cap B$, ***complement*** $\overline{A}$
- ***power set*** of $A = \{0, 1\}$: the set of all subsets of A $\{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$.

Sequences and Tuples 序列和元组

A ***sequence*** of objects is a list of these objects in some order.

$(7, 21, 57)$

- The order and repetition do matter in a sequence.

Finite sequences often are called ***tuples***.

- ***k-tuple***: a sequence with k elements. $(7, 21, 57)$ is a 3-tuple.
- ***ordered pair***: a 2-tuple.
- ***Cartesian product*** of A and B: $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$.

Functions 函数

A function is an object that sets up an input–output relationship.

- In every function, the same input always produces the same output.
- A function also is called a mapping.

$f(a) = b$

- domain: the set of possible inputs to the function.
- range: the set of outputs of a function.

$f : D \rightarrow R$

- onto the range: a function that does use all the elements of the range.

Functions

例

- Let $\mathcal{Z}_m = \{0, 1, 2, \ldots, m-1\}$.
- $g : \mathcal{Z}_4 \times \mathcal{Z}_4 \to \mathcal{Z}_4$

g	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Functions

例

- Let $\mathcal{Z}_m = \{0, 1, 2, \ldots, m-1\}$.
- $g : \mathcal{Z}_4 \times \mathcal{Z}_4 \to \mathcal{Z}_4$

g	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

The function g is the addition function modulo 4.

Functions

- k-ary function: a function with k arguments. k: arity
- unary function: $k = 1$.
- binary function: $k = 2$.
- predicate or property: a function whose range is $\{$TRUE, FALSE$\}$.

Relations 关系

A relation, a k-ary relation, or a k-ary relation on A is a property whose domain is a set of k-tuples $A \times \cdots \times A$.

- binary relation: a 2-ary relation.
- If R is a binary relation, the statement aRb means that aRb =TRUE.

equivalence relation: a binary relation R is an equivalence relation if R satisfies three conditions:

1. R is reflexive if for every x, xRx
2. R is symmetric if for every x and y, xRy implies yRx
3. R is transitive if for every x, y, and z, xRy and yRz implies xRz

Alphabets 字母表

- We define an alphabet to be any nonempty finite set.
- The members of the alphabet are the symbols of the alphabet.

例

- $\Sigma_1 = \{0, 1\}$
- $\Sigma_2 = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$
- $\Gamma = \{0, 1, x, y, z\}$

Strings 字符串

A string over an alphabet is a finite sequence of symbols from that alphabet.

例

- $\Sigma_1 = \{0, 1\}$, 01001 is a string over Σ_1
- $\Sigma_2 = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots, \mathtt{z}\}$, abracadabra is a string over Σ_2

- If w is a string over Σ, the length of w, written $|w|$, is the number of symbols that it contains.
- empty string: the string of length zero, written ε.
- If $|w| = n$, then $w = a_1 a_2 \cdots a_n$, where $a_i \in \Sigma$.
- the reverse of w: written $w^{\mathcal{R}}$, is $w^{\mathcal{R}} = a_n a_{n-1} \cdots a_1$

Strings

- substring: String z is a substring of w if z appears consecutively within w.
- `cad` is a substring of `abracadabra`
- concatenation of string x and string y, written xy
  - $|x| = m$ and $|y| = n$
  - the concatenation of x and y is $x_1 \cdots x_m y_1 \cdots y_n$
  - To concatenate a string with itself, use the notation x^k to mean

$$\overbrace{xx \cdots x}^{k}$$

Languages 语言

- The lexicographic order of strings is the same as the familiar dictionary order.
- shortlex order or string order is identical to lexicographic order, except that shorter strings precede longer strings.
  - the string order of all strings over $\{0, 1\}$ is $(\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots)$

Languages 语言

- The lexicographic order of strings is the same as the familiar dictionary order.
- shortlex order or string order is identical to lexicographic order, except that shorter strings precede longer strings.
  - the string order of all strings over $\{0, 1\}$ is $(\varepsilon, 0, 1, 00, 01, 10, 11, 000, \dots)$
- String x is a prefix of string y if a string z exists where $xz = y$.
- x is a proper prefix of y if in addition $x \neq y$.

Languages 语言

- The lexicographic order of strings is the same as the familiar dictionary order.
- shortlex order or string order is identical to lexicographic order, except that shorter strings precede longer strings.
    - the string order of all strings over $\{0, 1\}$ is $(\varepsilon, 0, 1, 00, 01, 10, 11, 000, \dots)$
- String x is a prefix of string y if a string z exists where $xz = y$.
- x is a proper prefix of y if in addition $x \neq y$.

A language is a set of strings.

- A language is prefix-free if no member is a proper prefix of another member.

语言的运算

定义 (语言的连接)

设 L_1 为字母表 Σ_1 上的语言，L_2 为字母表 Σ_2 上的语言，L_1 和 L_2 的连接L_1L_2 由下式定义：

$$L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$$

例

设 $\Sigma_1 = \{a, b\}$，$\Sigma_2 = \{0, 1\}$，$L_1 = \{ab, ba, bb\}$，$L_2 = \{00, 11\}$，则

$$L_1L_2 = \{ab00, ab11, ba00, ba11, bb00, bb11\}$$

语言的运算

定义 (语言的闭包)

语言 L 的**闭包**记作 L^*，定义如下：

1. $L^0 = \{\varepsilon\}$；
2. 对于 $n \geqslant 1$，$L^n = LL^{n-1}$；
3. $L^* = \bigcup_{n \geqslant 0} L^n$。

语言 L 的**正闭包**记作 L^+，定义为 $L^+ = \bigcup_{n \geqslant 1} L^n$。

语言的运算

例 (语言的闭包)

设 $\Sigma = \{0, 1\}$，$L = \{10, 01\}$，则 $L^0 = \{\varepsilon\}$，$L^1 = L = \{10, 01\}$
$L^2 = LL = \{1010, 1001, 0110, 0101\}$，$\cdots$，
$L^* =$
$\{\varepsilon, 10, 01, 1010, 1001, 0110, 0101, 101010, 101001, 100110, 100101, \cdots\}$

语言的运算

字母表 Σ 本身也是 Σ 上的语言。

- Σ^+：由 Σ 中的字符组成的全体字符串的集合（不包括 ε）
- $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

例

设 $\Sigma = \{0, 1\}$，则

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \cdots\}$

由 0 和 1 组成的一切长度、一切次序的串（包括空串）。

Outline

Definitions, Theorems, and Proofs 定义、定理和证明

- Definitions describe the objects and notions that we use.
  - Precision is essential to any mathematical definition.
- Mathematical statements
  - The statement may or may not be true.
- A proof is a convincing logical argument that a statement is true
  - A murder trial demands proof "beyond any reasonable doubt".
  - A mathematician demands proof beyond ***any*** doubt.
- A theorem is a mathematical statement proved true.
  - lemmas: statements that assist in the proof of another, more significant statement.
  - corollaries of the theorem: a theorem or its proof may allow us to conclude easily that other, related statements are true.

Finding Proofs

- Carefully read the statement you want to prove.
  - Do you understand all the notation?
  - Rewrite the statement in your own words.
  - Break it down and consider each part separately.

Multipart statements

- P if and only if Q or P iff Q or $P \iff Q$
  (Both P and Q are mathematical statements.)
  - P only if Q: If P is true, then Q is true, written $P \Rightarrow Q$
  - P if Q: If Q is true, then P is true, written $P \Leftarrow Q$
- Two sets A and B are equal.
  - A is a subset of B or $A \subseteq B$
  - B is a subset of A or $B \subseteq A$

Finding Proofs

- When you want to prove a statement or part thereof, try to get an intuitive, "gut" feeling of why it should be true.
  - Experimenting with examples is especially helpful.
  - Try to find a counterexample.
    - Seeing where you run into difficulty when you attempt to find a counterexample

- When that you have found the proof, you must write it up properly.
  - A well-written proof is a sequence of statements,
  - each one follows by simple reasoning from previous statements.
    - Be patient.
    - Come back to it.
    - Be neat.
    - Be concise.

Outline

Proof by Construction 构造性证明

- Proof by Construction
  - Many theorems state that a particular type of object exists.
  - Demonstrating how to construct the object.

定义

We define a graph to be k-regular if every node in the graph has degree k.

定理

For each even number n greater than 2, there exists a 3-regular graph with n nodes.

Proof by Construction 构造性证明

定理

For each even number n greater than 2, there exists a 3-regular graph with n nodes.

Proof.

- Let n be an even number greater than 2.
- Construct graph $G = (V, E)$ with n nodes as follows.
- The set of nodes of G is $V = \{0, 1, \ldots, n-1\}$,
- and the set of edges of G is the set

$$
\begin{aligned}
E &= \{(i, i+1) \mid \text{for } 0 \le i \le n-2\} \cup \{(n-1, 0)\} \\
&\quad \cup \{(i, i+n/2) \mid \text{for } 0 \le i \le n/2 - 1\}
\end{aligned}
$$

$\square$

Proof by Contradiction 反证法

- Proof by Contradiction
  - Assume that the theorem is false.
  - and then show that this assumption leads to an obviously false consequence, called a contradiction.

定理

$\sqrt{2}$ *is irrational.*

Proof.

- Assume that $\sqrt{2}$ is rational. Thus $\sqrt{2} = \frac{m}{n}$
- where m and n are integers. If both m and n are divisible by the same integer greater than 1, divide both by the largest such integer.
- Doing so doesn't change the value of the fraction.

$\square$

Proof by Contradiction 反证法

定理

$\sqrt{2}$ *is irrational.*

Proof.

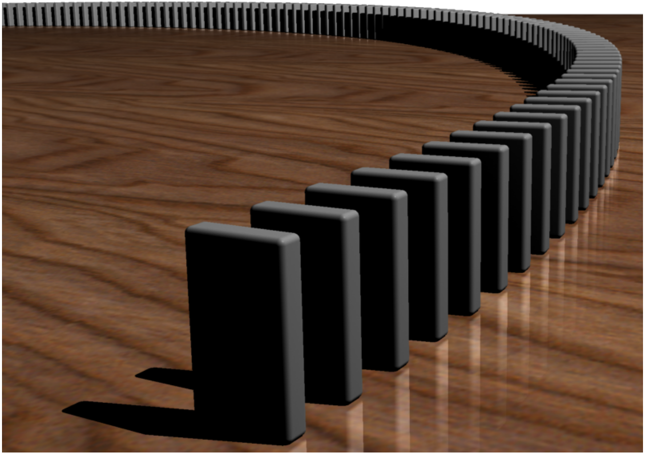
- Assume that $\sqrt{2}$ is rational. Thus $\sqrt{2} = \frac{m}{n}$
- where m and n are integers. If both m and n are divisible by the same integer greater than 1, divide both by the largest such integer.
- Doing so doesn't change the value of the fraction.
- Now, at least one of m and n must be an odd number.
- $n\sqrt{2} = m \Rightarrow 2n^2 = m^2 \Rightarrow m^2$ is even. $\Rightarrow m$ is even.
- $m = 2k$ for some integer k. $\Rightarrow 2n^2 = 4k^2 \Rightarrow n^2 = 2k^2 \Rightarrow n$ is even.

$\square$

Proof by Induction 归纳法

- Proof by Induction
  - An advanced method used to show that all elements of an infinite set have a specified property.
  - Basis: Prove that $\mathcal{P}(1)$ is true.
  - Induction step: For each $i \geq 1$, assume that $\mathcal{P}(i)$ is true and use this assumption to show that $\mathcal{P}(i+1)$ is true.
    - $\mathcal{P}(i)$ is true is called the induction hypothesis.
    - A stronger induction hypothesis: $\mathcal{P}(j)$ is true for every $j \leq i$.
    - When we want to prove that $\mathcal{P}(i+1)$ is true, we have already proved that $\mathcal{P}(j)$ is true for every $j \leq i$.

Proof by Induction 归纳法



数学归纳法：结构归纳法

证明与结构有关的命题，多数是递归定义的

定义

(1) 任意数字或字母都是表达式；

(2) 如果 E 或 F 是表达式，则 $E+F$，$E*F$ 和 (E) 也都是表达式；

(3) 表达式只能通过 (1)、(2) 两条给出。

数学归纳法：结构归纳法

证明与结构有关的命题，多数是递归定义的

定义

(1) 任意数字或字母都是表达式；

(2) 如果 E 或 F 是表达式，则 $E+F$，$E*F$ 和 (E) 也都是表达式；

(3) 表达式只能通过 (1)、(2) 两条给出。

递归定义

- (1) 是递归基础，必须有的
- (2) 是归纳，产生无穷多个表达式
- (3) 是排他，表达式不能再有其他形式

数学归纳法：结构归纳法

证明与结构有关的命题，多数是递归定义的

定义

(1) 任意数字或字母都是表达式；

(2) 如果 E 或 F 是表达式，则 $E+F$，$E*F$ 和 (E) 也都是表达式；

(3) 表达式只能通过 (1)、(2) 两条给出。

递归定义

- (1) 是递归基础，必须有的
- (2) 是归纳，产生无穷多个表达式
- (3) 是排他，表达式不能再有其他形式

根据 (1)：$2, 3, 6, 8, x, y, z$ 等是表达式；

根据 (2)：$x+3, y*6, 8*(2+x)$ 等也都是表达式。

数学归纳法：结构归纳法

定理

由前面定义的每个表达式中，左括号的个数一定等于右括号的个数。

数学归纳法：结构归纳法

Proof.

归纳基础：按照 (1)，表达式只包含单个数字或字母，没有括号，左括号和右括号个数均为 0，当然相等。

数学归纳法：结构归纳法

Proof.

归纳基础：按照 (1)，表达式只包含单个数字或字母，没有括号，左括号和右括号个数均为 0，当然相等。

归纳步骤：设表达式 B 是通过 (2) 构造出来的，有 3 种构造方法：

(1) $B = E + F$；　(2) $B = E * F$；　(3) $B = (E)$。

设表达式 E、F 包含的左、右括号数目相等，

且设 E 中有 m 个左、右括号，F 中有 n 个左、右括号。

数学归纳法：结构归纳法

Proof.

归纳基础：按照 (1)，表达式只包含单个数字或字母，没有括号，左括号和右括号个数均为 0，当然相等。

归纳步骤：设表达式 B 是通过 (2) 构造出来的，有 3 种构造方法：

(1) $B = E + F$； (2) $B = E * F$； (3) $B = (E)$。

设表达式 E、F 包含的左、右括号数目相等，

且设 E 中有 m 个左、右括号，F 中有 n 个左、右括号。

在 $B = E + F$ 和 $B = E * F$ 时，B 中左、右括号数均等于 $m + n$；

在 $B = (E)$ 时，B 中左、右括号数等于 $m + 1$。

在 3 种情况下，构造出的新表达式 B 所包含的左、右括号数目仍然相等。 □

Conclusion

- Automata, Computability, and Complexity
- Mathematical Notions and Terminology
  - Sets
  - Sequences and Tuples
  - Functions and Relations
  - Strings and Languages

Conclusion

- Automata, Computability, and Complexity
- Mathematical Notions and Terminology
  - Sets
  - Sequences and Tuples
  - Functions and Relations
  - Strings and Languages
- Definitions, Theorems, and Proofs
- Types of Proof
  - Proof by Construction
  - Proof by Contradiction
  - Proof by Induction