0 Introduction

杨雅君

yjyang@tju.edu.cn

智能与计算学部
天津大学

2022

授课教师介绍

杨雅君

- 智能与计算学部 人工智能学院 副教授
- CCF信息系统专委会委员、CCF数据库专委会通讯委员
- 研究方向：图数据管理、社交网络、数据挖掘、机器学习
- 2006年和2013年于哈尔滨工业大学数学与应用数学专业和计算机科学与技术专业分别获得学士和博士学位
- 师从世界著名数据库专家李建中教授
- 2009年赴香港中文大学、2017年赴澳大利亚格里菲斯大学担任访问学者。

授课教师介绍

杨雅君

- 以第一作者身份在国际顶级和重要学术会议SIGMOD、VLDB、ICDE、CIKM、DASFAA、ICME，国际重要学术期刊TOIS、Complexity、World Wide Web，以及国内一级学术期刊《计算机学报》、《软件学报》上发表了三十余篇学术论文
- 获得第五届中国传感器网络学术会议最佳论文奖（排名第一）

授课教师介绍

杨雅君

- 近年来主持国家重点研发计划子课题、自然科学基金多项，作为骨干人员参与国家重点研发计划，国家自然科学基金重点项目多项
- 长期担任DASFAA、ADMA、APWEB-WAIM、ICPCSEE、TURC（SIGMOD China）等国际学术会议的程序委员会委员，并担任TKDE、WWWJ、Pattern Recognition等多个重要国际学术刊物审稿专家

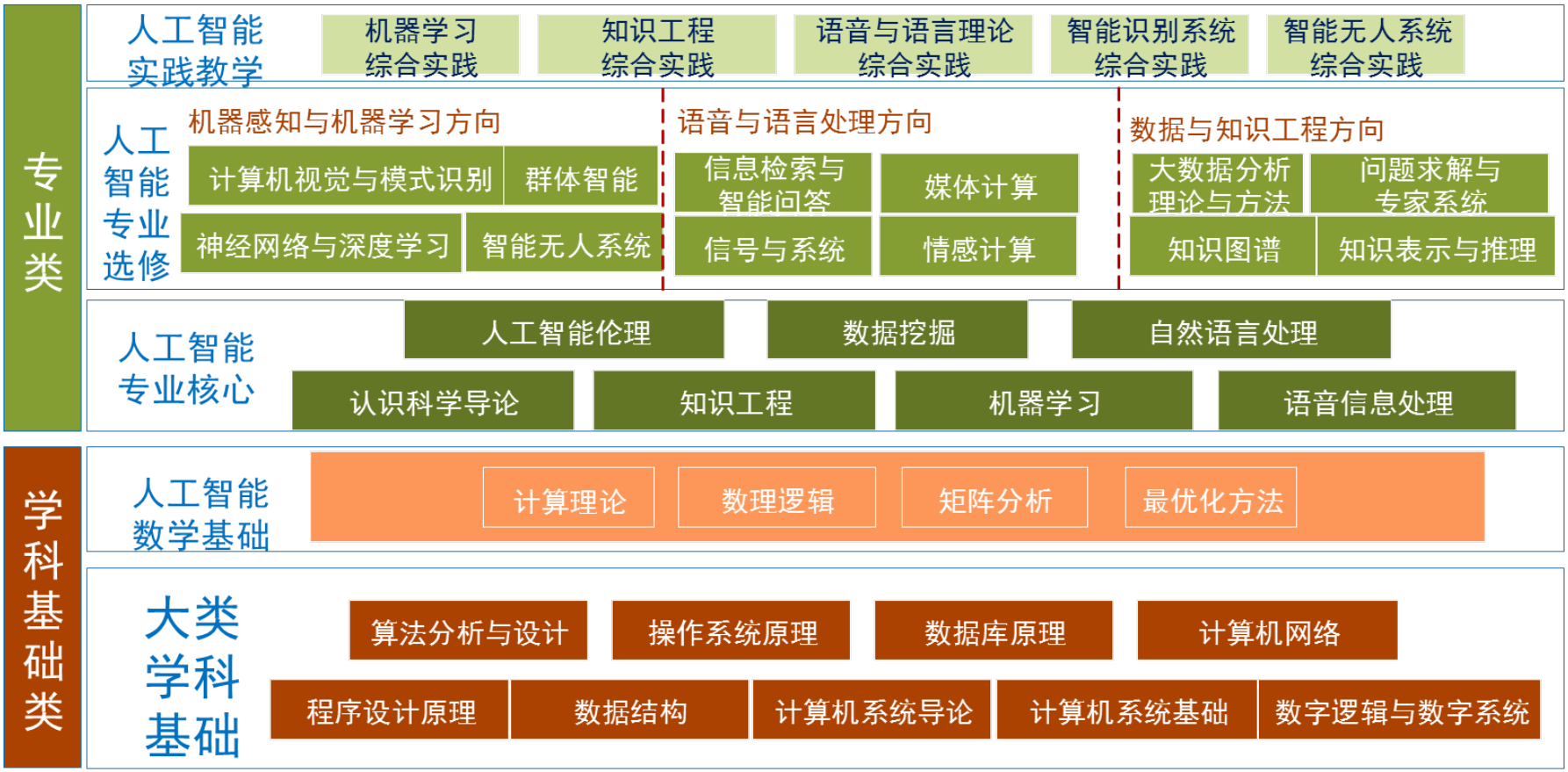
授课教师介绍

杨雅君

曾担任任课教师:

- 本科生课程:《离散数学》、《形式语言与自动机》、《数据库系统原理》
- 研究生课程:《机器学习》、《组合数学》

人工智能数学基础课程简介



人工智能数学基础课程简介

人工智能数学基础课程简介

任课教师：

- 杨雅君：计算理论、数理逻辑
- 刘志磊：矩阵分析、最优化方法

课程目标：

- 掌握的计算理论的基本概念和基本理论
- 理解的数理逻辑的系统语义和语法的本质
- 具备一定将实际问题形式化的数学能力
- 形成对科学问题可计算性进行分析的观念

人工智能数学基础课程简介

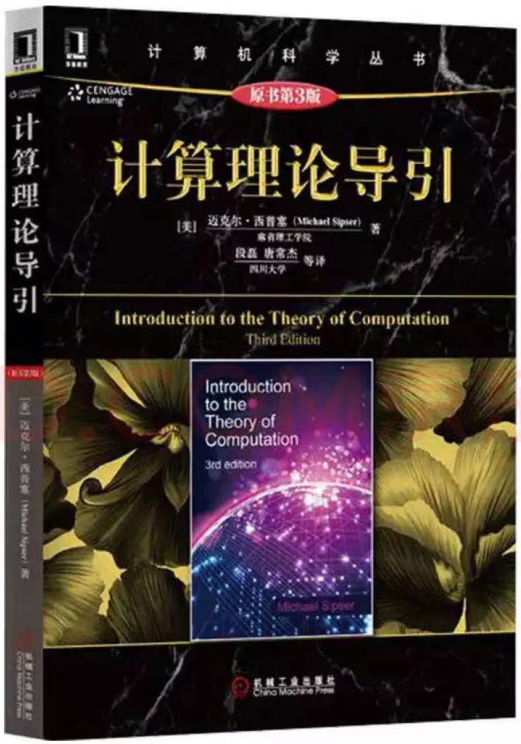
课程成绩组成：

- 卷面考试成绩：80%
- 平时成绩：20%
  - 出勤情况：三次点名未到取消考试资格
  - 日常作业
  - 课堂表现：酌情加分$0.5 \sim 1$分

教材

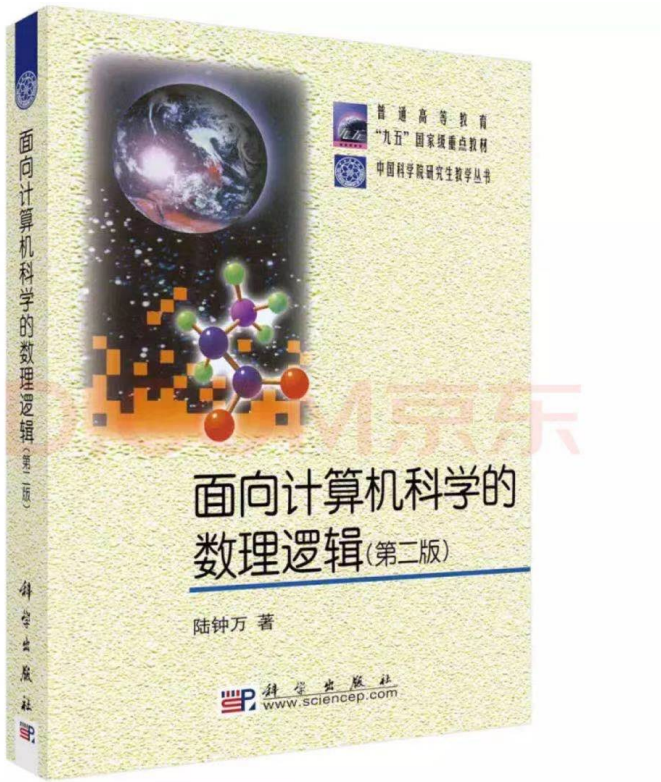
《计算理论导引（第3版）》

- 作者：Michael Sipser
- 出版社：机械工业出版社



《面向计算机科学的数理逻辑》

- 作者：陆钟万
- 科学出版社



参考书

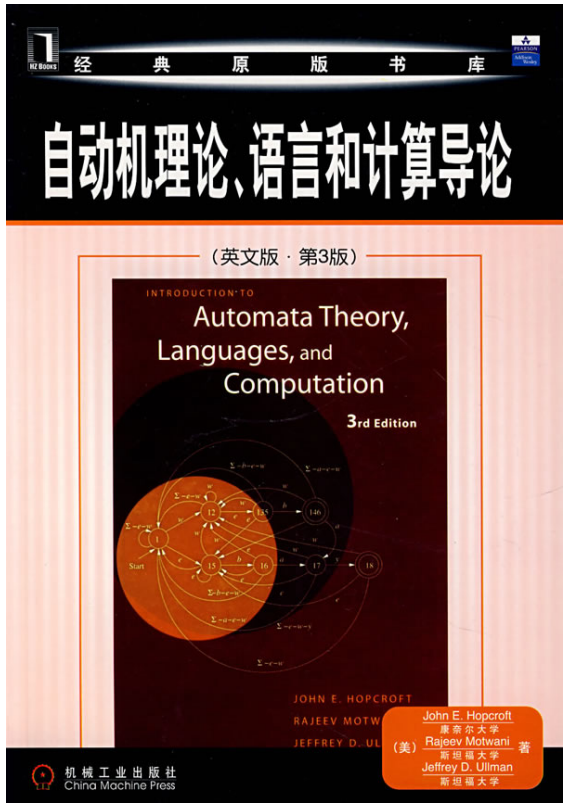
- 《形式语言与自动机》
  - 作者：陈有祺
  - 出版社：机械工业出版社
  - 出版日期：2008
  - 页数：227

参考书

- 《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



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

— Edsger Dijkstra

计算机科学并不只是关于计算机，
就像天文学并不只是关于望远镜一样。

— Edsger Dijkstra

在开始课程之前，我们首先思考几个问题！

在开始课程之前，我们首先思考几个问题！

- 计算机科学到底是不是科学?

在开始课程之前，我们首先思考几个问题！

- 计算机科学到底是不是科学?
- 计算机科学的核心是什么?

唐纳德·克努特（Donald Knuth）

Donald Knuth

唐纳德·克努特（Donald Knuth）

Donald Knuth

- 1974年图灵奖获得者

唐纳德·克努特（Donald Knuth）

Donald Knuth

- 1974年图灵奖获得者
- 《计算机程序设计的艺术》

唐纳德·克努特（Donald Knuth）

Donald Knuth

- 1974年图灵奖获得者
- 《计算机程序设计的艺术》
- Tex排版系统之父

唐纳德·克努特（Donald Knuth）

Donald Knuth

- 1974年图灵奖获得者
- 《计算机程序设计的艺术》
- Tex排版系统之父

算法 + 数据结构 = 程序！

在开始课程之前，我们首先思考几个问题！

- 计算机科学到底是不是科学?
- 计算机科学的核心是什么?

在开始课程之前，我们首先思考几个问题！

- 计算机科学到底是不是科学?
- 计算机科学的核心是什么?
- 计算机的能力是否有局限?

在开始课程之前，我们首先思考几个问题！

- 计算机科学到底是不是科学?
- 计算机科学的核心是什么?
- 计算机的能力是否有局限?
- 算法（计算）的本质和数学定义是什么?

Outline

The Theory of Computation 计算理论

计算机的基本能力和局限性是什么？

The Theory of Computation 计算理论

计算机的基本能力和局限性是什么？

- 要回答这个问题应追溯到20世纪30年代，当时的数理逻辑学家们首先开始探究什么是计算

The Theory of Computation 计算理论

计算机的基本能力和局限性是什么？

- 要回答这个问题应追溯到20世纪30年代，当时的数理逻辑学家们首先开始探究什么是计算

- Automata (自动机)
- Computability (可计算性)
- Complexity (复杂度)

Complexity Theory 计算复杂性理论

计算问题

- 较容易的: e.g., 排序问题
- 较困难的: e.g., 时间表问题

Complexity Theory 计算复杂性理论

计算问题

- 较容易的: e.g., 排序问题
- 较困难的: e.g., 时间表问题

是什么使得某些问题很难计算，而又使另一些问题容易计算？

- 这是计算复杂性理论的核心问题.
- We don't know the answer to it. (researched for over 40 years!)
  - 科研工作者发现了一个根据计算难度给问题分类的完美体系（类似元素周期表）

计算复杂性理论

当面对一个似乎很难计算的问题时

1. 通过弄清楚问题困难的根源.
   - 可以针对问题做某些改动，使问题变得容易解决.
2. 寻求问题的一个并不完美的解.
   - 寻找问题的近似解会相对容易.
3. 有些问题仅在最坏情况下是困难的，而在绝大多数情况下是容易的.
   - 一个偶尔运行很慢而通常运行很快的程序是能够满足需要的.
4. 考虑选择其他计算方式
   - 能够加速某些计算任务的随机计算.

Computability Theory 可计算性理论

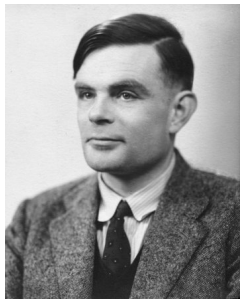
一些看似简单和基本的问题是不能用计算机解决的!

- 确定一个数学命题的真或假.

可计算理论和计算复杂性理论是密切相关的.

- 计算复杂性理论的目标
  - 把问题分成容易计算的和难计算的
- 可计算理论的目标
  - 把问题分成可解的和不可解的

Kurt Gödel (1906–1978)



Alan Turing (1912–1954)

Alonzo Church (1903–1995)

Automata Theory 自动机理论

自动机理论阐述了计算的数学模型的定义和性质!

Models of computation

可计算理论和计算复杂性理论需要对计算机给出一个准确的数学定义.

Automata Theory 自动机理论

自动机理论阐述了计算的数学模型的定义和性质!

Models of computation

可计算理论和计算复杂性理论需要对计算机给出一个准确的数学定义.

- 计算的数学模型
  - 有穷自动机(正则文法)
  - 下推自动机(上下文无关文法)
  - 图灵机（现代计算机的数学模型）

Automata Theory 自动机理论

自动机理论阐述了计算的数学模型的定义和性质!

Models of computation

可计算理论和计算复杂性理论需要对计算机给出一个准确的数学定义.

- 计算的数学模型
  - 有穷自动机(正则文法)
  - 下推自动机(上下文无关文法)
  - 图灵机（现代计算机的数学模型）

自动机理论帮助我们去理解计算的形式化定义.

Outline

Sets 集合

集合是将一组对象看成一个整体.

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

- ***element* 元素** or ***member* 成员**: 集合中的对象. $7 \in \{7, 21, 25\}$, $8 \notin \{7, 21, 25\}$
- ***subset* 子集**: $A \subseteq B$, proper subset 真子集: $A \subsetneq B$
- ***natural numbers* 自然数集**: $\mathcal{N}$, integers 整数集$\mathcal{Z}$
- ***empty set* 空集**: $\emptyset$
- $\{n \mid n = m^2 \text{ for some } m \in \mathcal{N}\}$
- ***union* 并集** $A \cup B$, ***intersection* 交集** $A \cap B$, ***complement* 补集** $\overline{A}$
- ***power set* 幂集**: $A = \{0, 1\}$, $P(A) = \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$.

Sequences and Tuples 序列和元组

序列是某些元素元素或成员按某种顺序排成的表.

$(7, 21, 57)$

- 序列中的元素的顺序和是否重复很重要

有穷序列也被称为**多元组**.

- ***k*-tuple *k*-元组**: k 个元素的序列. $(7, 21, 57)$ is a 3-tuple.
- ***ordered pair* 有序对**: a 2-tuple.

- ***Cartesian product* 笛卡尔积** of A and B:
  $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$.

Functions 函数

函数是一个建立输入-输出关系的对象.

- 对于每一个函数, 同样的输入总会产生同样的输出.
- 函数又称为映射.

$f(a) = b$

- 定义域: 函数所有可能的输入组成的集合.
- 值域: 函数输出结果的集合.

$f : D \to R$

- 映上到这个值域: 如果函数取得值域的所有元素.

Functions 函数

Example

- 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 函数

Example

- 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

函数 g 为模 4 的加法函数.

Functions 函数

- k 元函数: k 个自变量的函数. k: 函数的元数(arity)
- unary function 一元函数: $k = 1$.
- binary function 二元函数: $k = 2$.
- predicate 谓词 or property 性质: 值域为 {TRUE, FALSE} 的函数.

Relations 关系

定义域为 $A \times \cdots \times A$ 的谓词称为关系，或称为 k 元关系.

- binary relation 二元关系: a 2-ary relation.
- 设 R是一个二元关系, 则命题 aRb 表示 $aRb =$TRUE.

equivalence relation 等价关系: 如果二元关系 R 满足以下三个条件，则被称为等价关系:

1. R 是自反的, 即对每一个 x, xRx
2. R 是对称的, 即对每一个 x 和 y, xRy 意味着 yRx
3. R 是传递的, 即对每一个 x, y 和 z, xRy 和 yRz 意味着 xRz

Alphabets 字母表

- 定义字母表(alphabets)为任意一个非空有穷集合.
- 字母表的成员为该字母表的符号(symbol).

Example

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

Strings 字符串

字母表上的字符串为该字母表中符号的有穷序列.

Example

- $\Sigma_1 = \{0, 1\}$, 01001 为 Σ_1 上的一个字符串.
- $\Sigma_2 = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots, \mathtt{z}\}$, abracadabra 为 Σ_2 上的一个字符串.

- 设 w 是Σ 上的一个字符串, w 的长度, 记作 $|w|$, 等于它所包含的符号数.
- empty string 空串: 长度为零的字符串, 记作 ε.
- 若 $|w| = n$, 则 $w = a_1 a_2 \cdots a_n$, 这里 $a_i \in \Sigma$.
- w 的反转(reverse): 记作 $w^{\mathcal{R}}$, 为 $w^{\mathcal{R}} = a_n a_{n-1} \cdots a_1$

Strings 字符串

- substring 子串: 如果字符串 z 连续地出现在 w 中, 则称 z 是 w 的子串.
- cad 是 abracadabra 的子串.
- 字符串 x 和字符串 y 的连接(concatenation), 记作 xy
  - $|x| = m$ and $|y| = n$
  - x 和 y 的连接为 $x_1 \cdots x_m y_1 \cdots y_n$
  - 可采用上标法表示一个字符串自身连接多次, 例如 x^k 表示

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

Languages 语言

- 字符串的字典序(lexicographic order)和大家熟悉的字典序一样.
- 字符串顺序(shortlex order or string order), 它在字典序的基础上将短的字符串排在长的字符串的前面.
  - 字母表 $\{0,1\}$ 上所有的字符串顺序为 $(\varepsilon, 0, 1, 00, 01, 10, 11, 000, \dots)$

Languages 语言

- 字符串的字典序(lexicographic order)和大家熟悉的字典序一样.
- 字符串顺序(shortlex order or string order), 它在字典序的基础上将短的字符串排在长的字符串的前面.
  - 字母表 $\{0,1\}$ 上所有的字符串顺序为 $(\varepsilon, 0, 1, 00, 01, 10, 11, 000, \dots)$
- 字符串 x 是字符串 y 的前缀(prefix), 如果存在字符串 z 满足 $xz = y$.
- 若 $x \neq y$, 则 x 是 y 的真前缀(proper prefix), .

Languages 语言

- 字符串的字典序(lexicographic order)和大家熟悉的字典序一样.
- 字符串顺序(shortlex order or string order), 它在字典序的基础上将短的字符串排在长的字符串的前面.
  - 字母表 $\{0,1\}$ 上所有的字符串顺序为 $(\varepsilon, 0, 1, 00, 01, 10, 11, 000, \dots)$
- 字符串 x 是字符串 y 的前缀(prefix), 如果存在字符串 z 满足 $xz = y$.
- 若 $x \neq y$, 则 x 是 y 的真前缀(proper prefix), .

字符串的集合称为语言(language).

- 如果语言中任何一个成员都不是其他成员的真前缀, 那么该语言则是无前缀的(prefix-free).

语言的运算

Definition (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：

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

Example

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

Definition (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

Example (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）
- $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

Example

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 定义描述了我们使用的对象和概念.
  - 任何数学定义都必须是精准的.
- Mathematical statements 数学命题
  - 命题可能为真, 也可能为假.
- 证明(proof)是一种逻辑论证, 它使人确信一个命题为真.
  - 在谋杀案审判中要求存在“没有任何疑点”的证据.
  - 数学家需要没有任何疑点的证明.
- 定理(theorem)是被证明为真的数学命题.
  - 引理(lemmas)
  - 推论(corollaries)

寻找证明

- 要仔细阅读清楚证明的命题.
  - 是否理解命题的所有记号?
  - 用自己的语言复写命题.
  - 拆开并且分别考虑每一部分.

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 $B \subseteq A$

寻找证明

- 当要证明一个命题或者它的一部分时, 要尽量找到一种直观感觉.
  - 尝试举一些例子对完成证明很有帮助.
  - 尝试寻找反例.
    - 看看在什么地方遇到找反例的困难

- 当找到正确证明的时候, 必须将它严格地书写出来.
  - 一个好的证明是一系列的形式化描述,
  - 每一条语句都由前面的语句经过简单的推理得到.
    - 要有耐心.
    - 回头做.
    - 条例清晰.
    - 表达简洁.

Outline

Proof by Construction 构造性证明

- Proof by Construction 构造性证明
  - 许多定理声明存在一种特定类型的对象.
  - 说明如何构造这样的对象的证明即为构造性证明.

Definition

我们定义：如果图中每一个顶点的度数都为 k, 则称这个图是 k 正则的(k-regular).

Theorem

对于每一个大于 2 的偶数 n, 存在一个有 n 个顶点的 3 正则图.

Proof by Construction 构造性证明

Theorem

对于每一个大于 2 的偶数 n, 存在一个有 n 个顶点的 3 正则图.

Proof.

- 设 n 是大于 2 的整数.
- 构造有 n 个顶点的图 $G = (V, E)$.
- G 的顶点集为 $V = \{0, 1, \ldots, n-1\}$,
- G 的边集为

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

Proof by Contradiction 反证法

- Proof by Contradiction 反证法
  - 假设定理为假.
  - 然后证明这个假设会导致一个明显的错误结论.

Theorem

$\sqrt{2}$ 是无理数.

Proof.

- 假设 $\sqrt{2}$ 是有理数, 则 $\sqrt{2} = \frac{m}{n}$.
- 此处 m 和 n 是整数. 如果 m 和 n 都能被同一个大于 1 的最大整数除尽, 则用那个整数除它们.
- 不会改变分式的值.

Proof by Contradiction 反证法

Theorem

$\sqrt{2}$ 是无理数.

Proof.

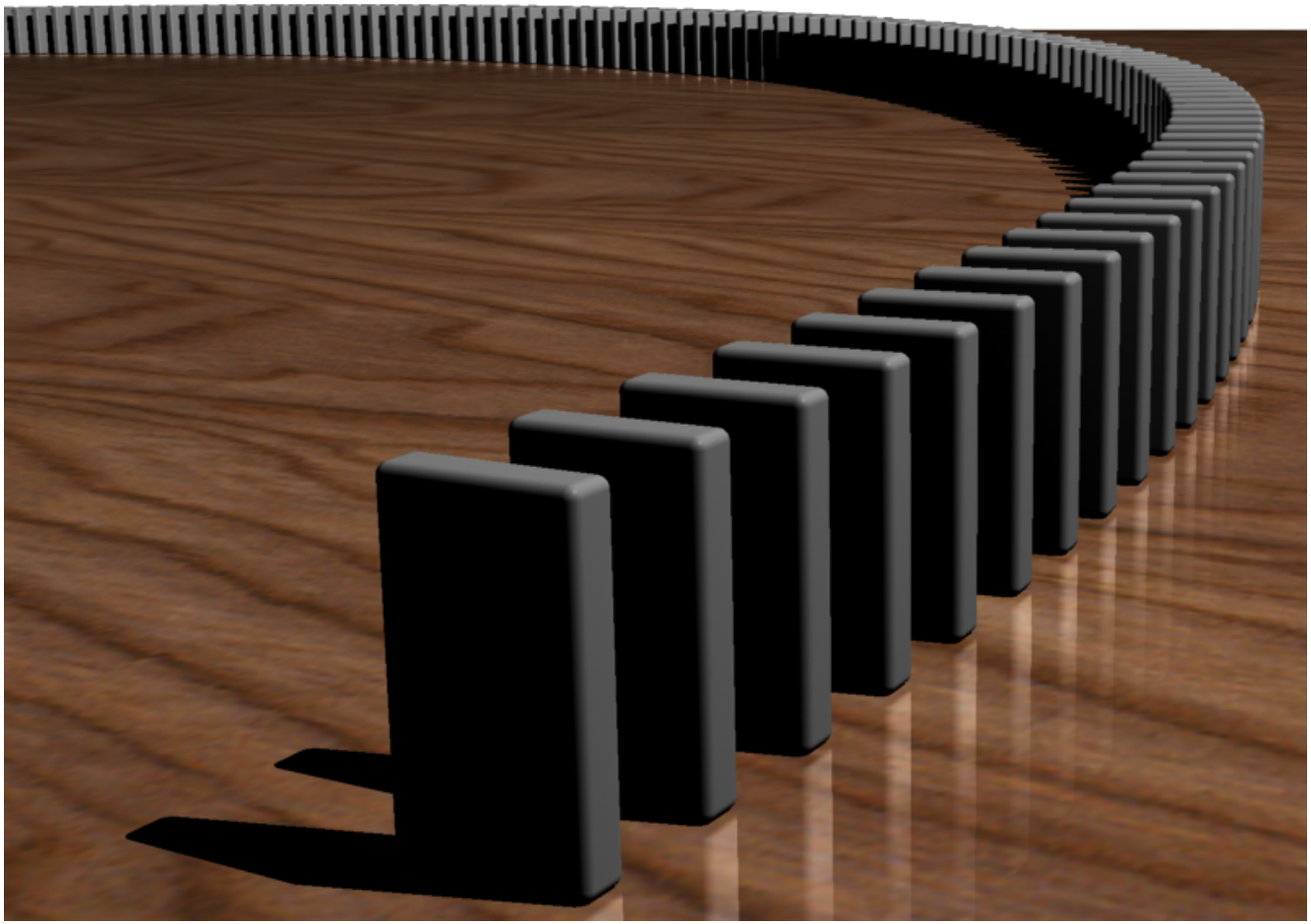
- 假设 $\sqrt{2}$ 是有理数, 则 $\sqrt{2} = \frac{m}{n}$.
- 此处 m 和 n 是整数. 如果 m 和 n 都能被同一个大于 1 的最大整数除尽, 则用那个整数除它们.
- 不会改变分式的值.
- 此时可证, m 和 n 均为偶数.
- $n\sqrt{2} = m \Rightarrow 2n^2 = m^2 \Rightarrow m^2$ 是偶数. $\Rightarrow m$ 是偶数.
- 对某个整数 k, $m = 2k$, $\Rightarrow 2n^2 = 4k^2 \Rightarrow n^2 = 2k^2 \Rightarrow n$ 是偶数.

□

Proof by Induction 归纳法

- Proof by Induction 归纳法
  - 归纳法是证明无穷集合的所有元素具有某种特定性质的高级方法.
  - 归纳基础(Basis): 证明 $\mathcal{P}(1)$ 为真.
  - 归纳步骤(Induction step): 对于每一个 $i \geq 1$, 假设 $\mathcal{P}(i)$ 为真, 并且利用这个假设去进一步证明 $\mathcal{P}(i+1)$ 为真.
    - $\mathcal{P}(i)$ 为真被称为 归纳假设(induction hypothesis).
    - 一种更强的归纳假设: 对于每一个 $j \leq i$, $\mathcal{P}(j)$ 为真.
    - 当要证明 $\mathcal{P}(i+1)$为真时, 我们已经证明对于每一个 $j \leq i$, $\mathcal{P}(j)$ 为真.

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

Definition

(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

Definition

(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

Definition

(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

Theorem

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

- 自动机理论，可计算性理论和计算复杂性理论
- 数学概念与术语
  - 集合
  - 序列和元组
  - 函数与关系
  - 字符串与语言

Conclusion

- 自动机理论，可计算性理论和计算复杂性理论
- 数学概念与术语
  - 集合
  - 序列和元组
  - 函数与关系
  - 字符串与语言
- 定义，定理和证明
- 证明的类型
  - 构造性证明
  - 反证法
  - 归纳法