南京航空航天大学

2020-2021年春季学期

# 形式语言与自动机理论

# Formal Languages and Automata Theory

康达周

dzkang@nuaa.edu.cn

计算机科学与技术学院

# 课程政策

- 评分依据
  - 70% 期末考试
  - 30% 平时作业和表现
- 期末考试
  - 课程结束两周后
  - 开卷
- 作业要求
  - 在课堂上会不定时发布作业
  - 作业一般要求在下一周内交上来
  - 无特殊原因迟交作业将不予批改和记分

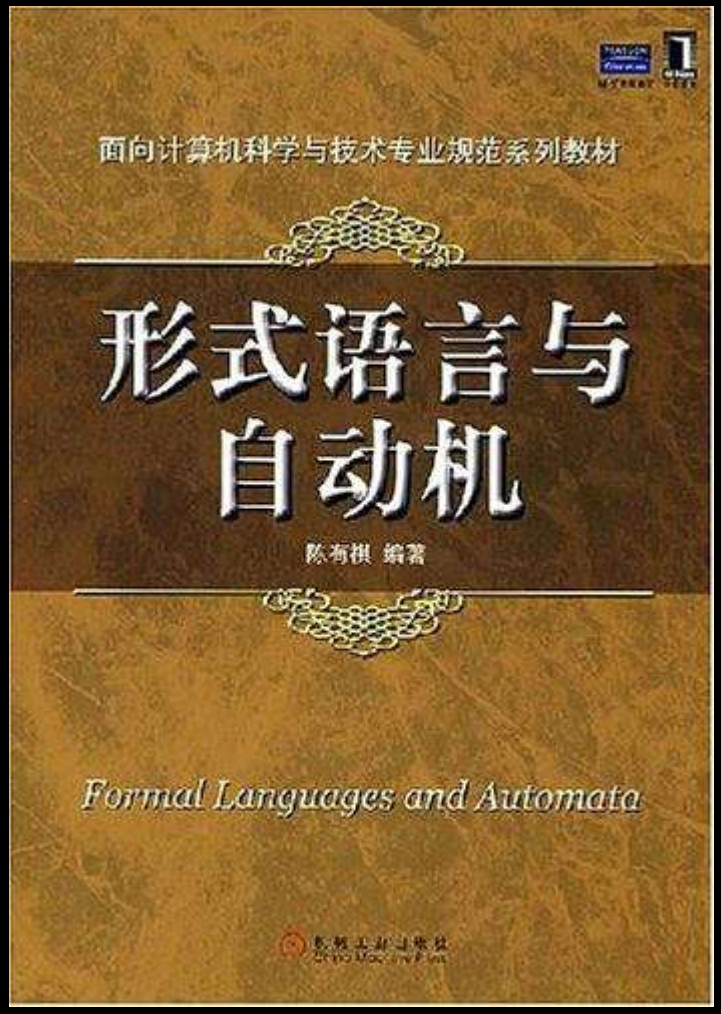
# FYI

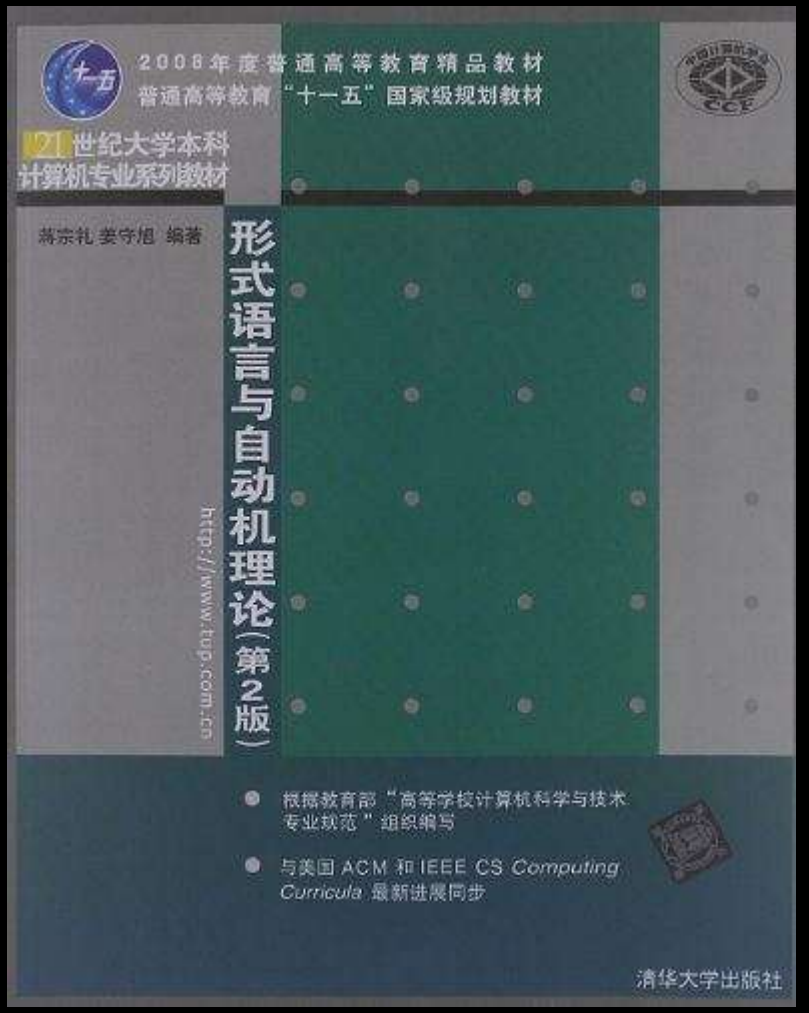
- 授课教师：康达周
- Email: dzkang at nuaa.edu.cn
- 办公地点：计算机院楼404

# 请注意以下几点

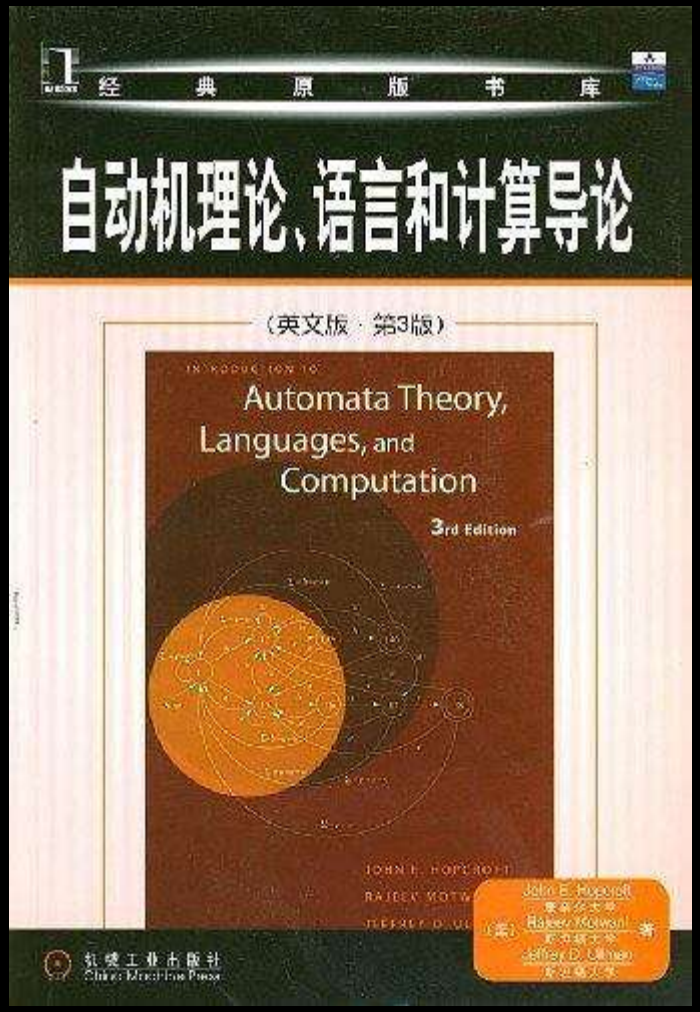
- 课本
  - 本课程向大家提供了课本的中英文电子版
  - 推荐大家尽量阅读英文课本辅助学习，当然也可以使用中文课本，或者两本书一起对照阅读
- PPT
  - 课程PPT中的重要内容都为中文或中英文对照，但有部分证明内容未翻译，请谅解
  - 极少数术语和描述的翻译与中文课本有差别，各种翻译版本在作业和考试中都可以用
- 在线平台
  - 课程使用了超星学习通作为在线平台发布信息、分享资料、进行答疑，请大家学习和使用

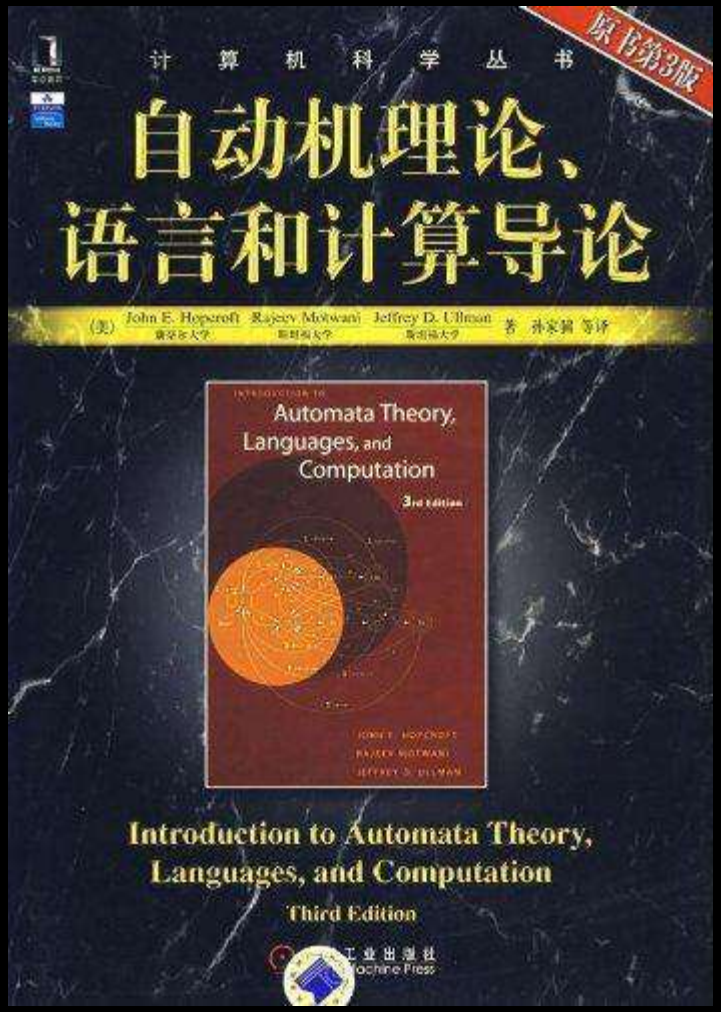
# 过去几年使用过的教材


面向计算机科学与技术专业规范系列教材
形式语言与自动机
陈有祺 编著
Formal Languages and Automata


2006年度普通高等教育精品教材
普通高等教育"十一五"国家级规划教材
21世纪大学本科计算机专业系列教材
蒋宗礼 姜守旭 编著
形式语言与自动机理论(第2版)
http://www.tup.com.cn
根据教育部"高等学校计算机科学与技术专业规范"组织编写
与美国ACM和IEEE CS Computing Curricula最新进展同步
清华大学出版社

# 目前推荐大家使用的教材



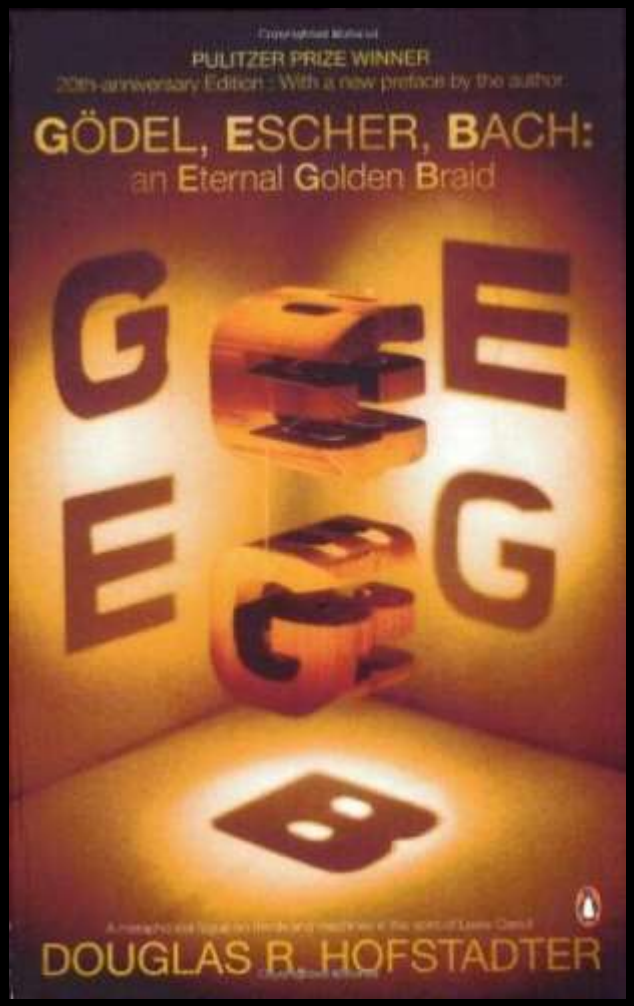
# 课本

- Hopcroft, Motwani, Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd Edition.
- 我们主要学习其中的前9个章节

# 参考书


- John C. Martin (2011). *Introduction to Languages and The Theory of Computation*. New York, NY 10020: McGraw Hill. ISBN 978-0-07-319146-1.
- Anderson, James A. (2006). *Automata theory with modern applications*. With contributions by Tom Head. Cambridge: Cambridge University Press. ISBN 978-0-521-61324-8. Zbl 1127.6804.
- Peter Linz (2011). *An Introduction to Formal Languages and Automata*. J&b India. ISBN: 978-9-38-085328-4

# 另外，强烈推荐阅读


PULITZER PRIZE WINNER
GÖDEL, ESCHER, BACH:
an Eternal Golden Braid
DOUGLAS R. HOFSTADTER


GÖDEL
ESCHER
BACH
哥德尔
艾舍尔
巴赫
——集异璧之大成
〔美〕侯世达 著
商务印书馆

# 01 导论

什么是自动机（理论）？

什么是形式语言？

学它们有什么用？

2B from NieR: Automata. Square Enix Co., 2017

如果我们在Google 上搜索automata（自动机） …

这里面也包含一个 “formal languages（形式语言）” 的例子 …

2 -> ll | t[w]o

9 -> ni[ne]

A2 2B 9S = All To Be Nice

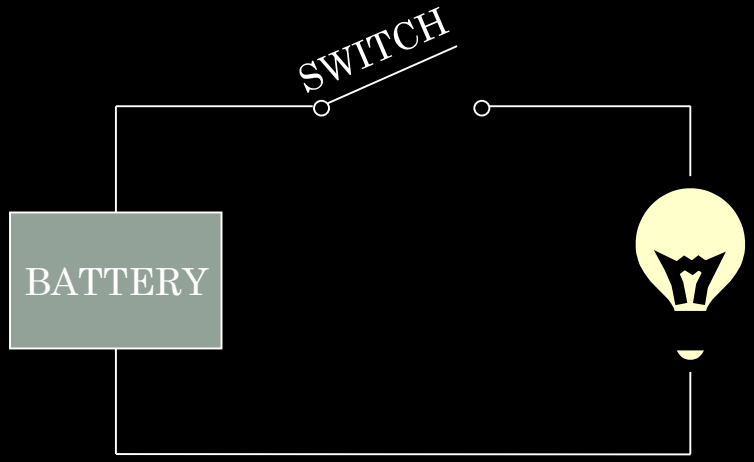
B -> Be

S -> [i]ce

# 何为自动机理论 automata theory?

- 自动机理论研究抽象计算设备 abstract computational devices
- 抽象设备是真实计算机的（简化）模型
- 计算发生在很多地方：服务器、笔记本电脑、手机、自然界 …
- 为什么我们需要抽象模型？

# 一个简单的“计算机”



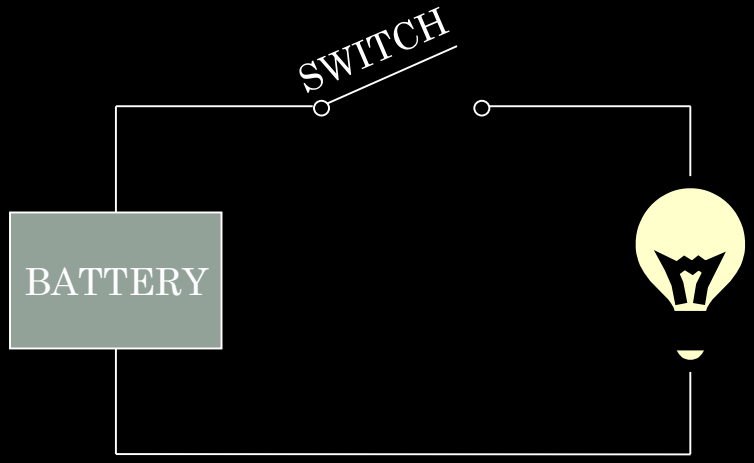

**input:** switch

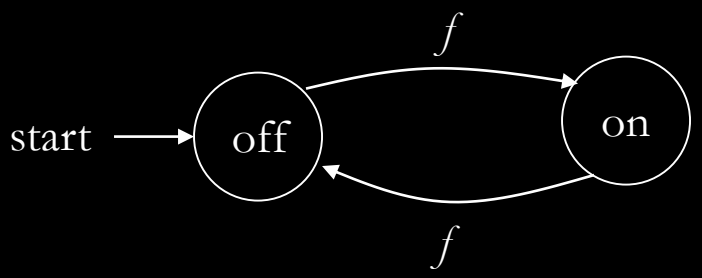
**output:** light bulb

**actions:** flip switch

**states:** on, off

# 一个简单的“计算机”




**input:** switch

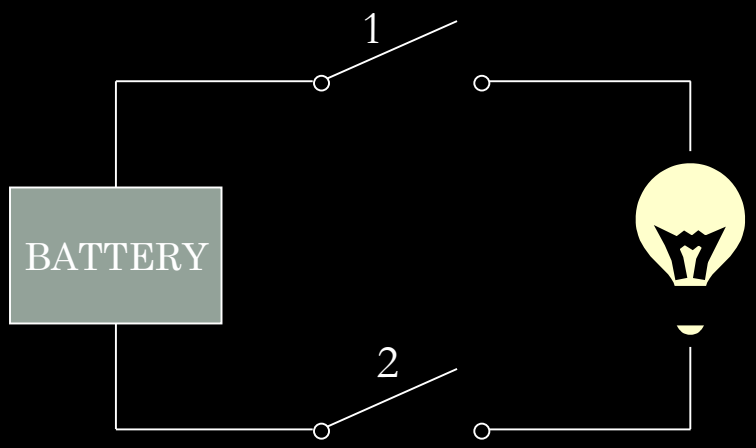
**output:** light bulb

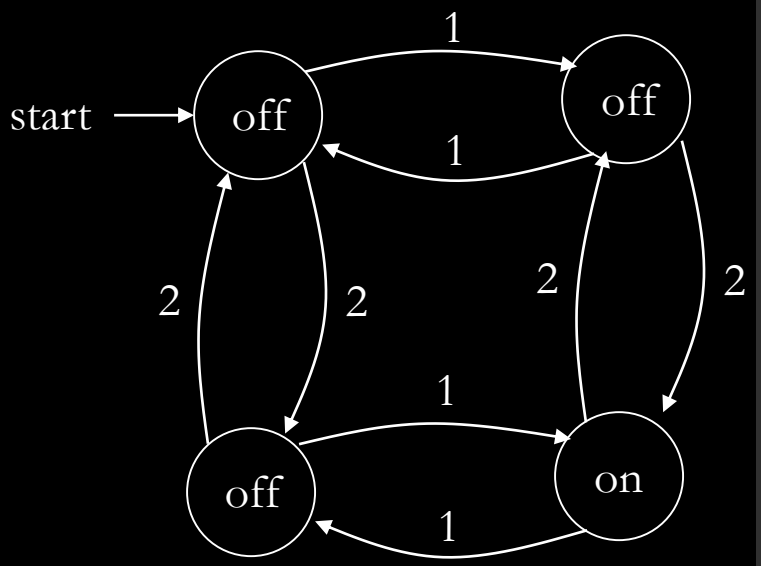
**actions:** *f* for “flip switch”

**states:** on, off

当且仅当出现奇数次翻转时，灯泡才亮

# 另一个“计算机”



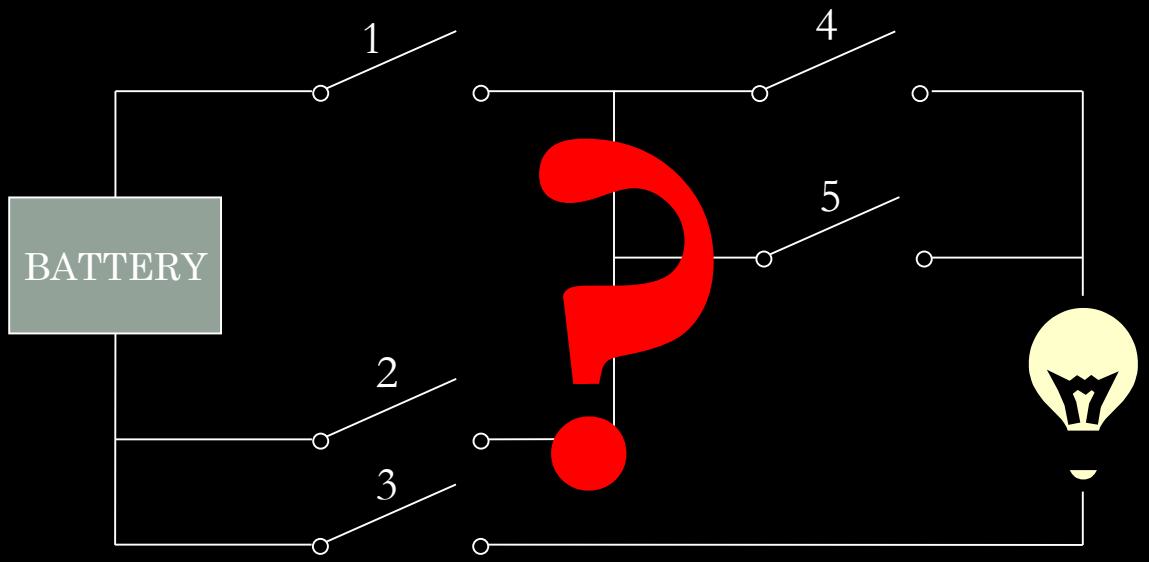

**inputs:** switches 1 and 2

**actions:** 1 for “flip switch 1”
2 for “flip switch 2”

**states:** on, off

当且仅当两个开关都被翻转了奇数次时，灯泡才亮

# 一个设计问题



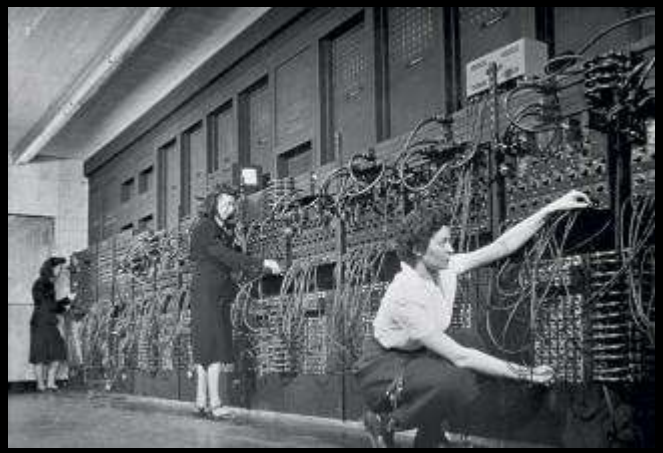

要求设计一个当且仅当所有开关均翻转相同次数时才点亮灯的电路

它能设计出来吗？它会有多复杂？会有什么有趣的特性？能用来做其它事吗？……

# 解决设计问题

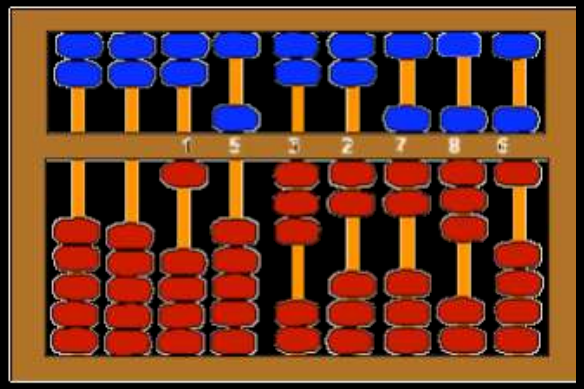
- 这些问题很难回答，因为计算设备可以以多种不同方式设计
- 通过将它们表示为比较容易理解和分析的简化模型，即抽象计算设备或自动机，我们将学习如何回答此类问题

A difference engine created by Charles Babbage, 1822



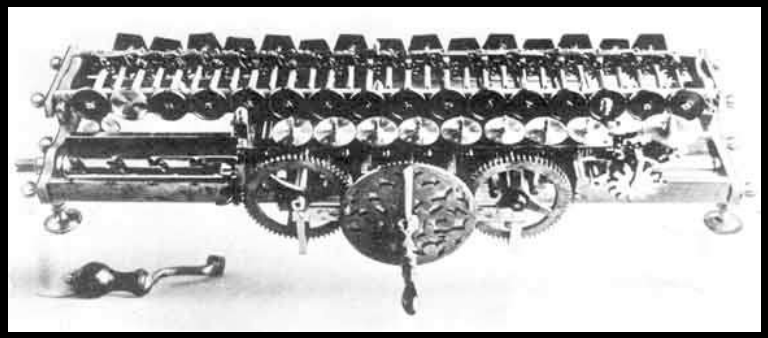
ENIAC, 1945

# 许许多多的计算设备

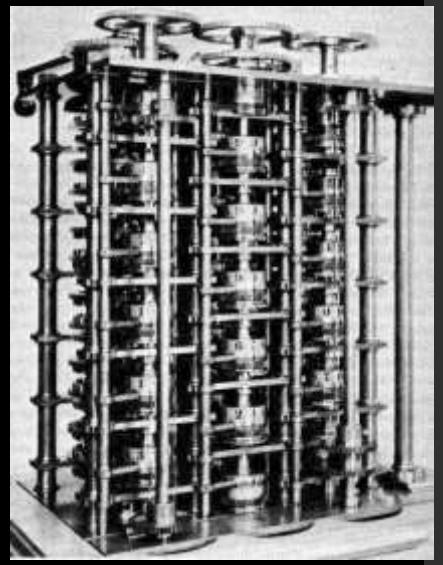


abacus
(Babylon 500BC, China 1300)

slide rule (1600s)



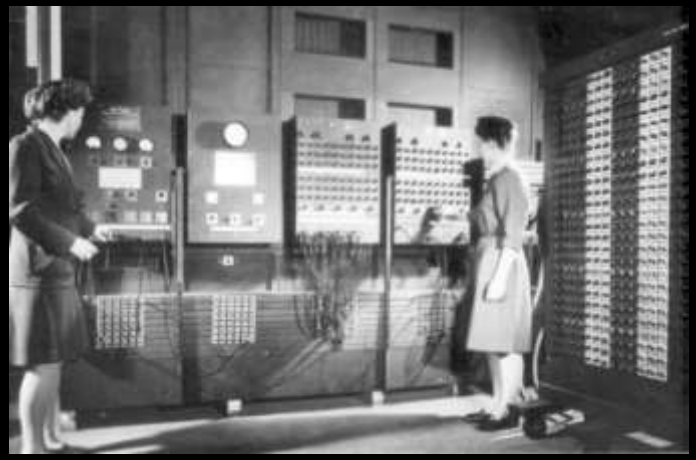
Leibniz calculator (1670s)



Babbage's
mechanical
engine
(1820s)

# 许许多多的计算设备

Z3 (Germany, 1941)



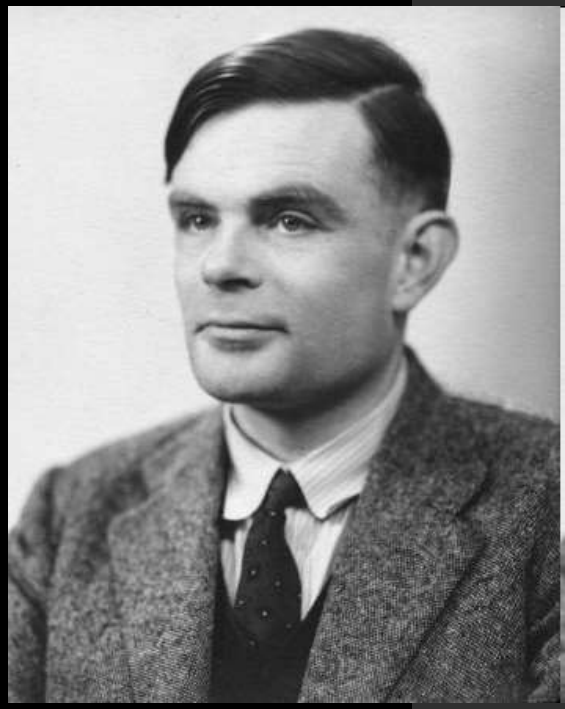
ENIAC (Pennsylvania, 1945)

PC (1980)

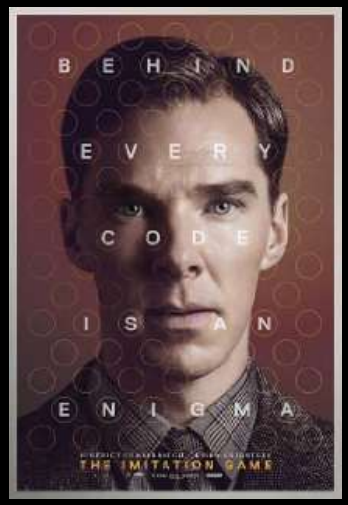
MacBook Air (2008)

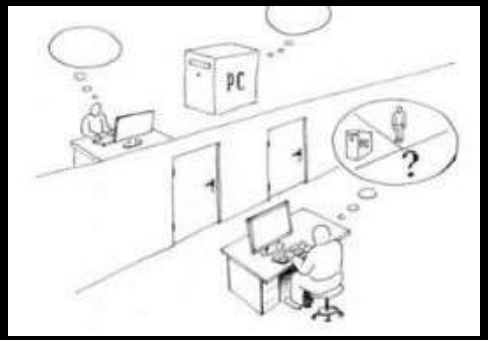
自动机理论的先驱

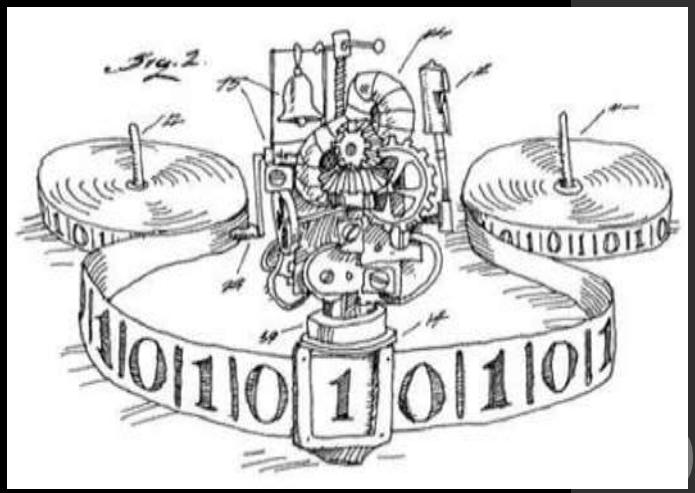
# Alan Turing (1912-1954)



- 现代计算机科学之父
- 英国数学家
- 提出抽象计算模型 ***Turing machines***（图灵机），当时甚至还发明电子计算机
- 听说过图灵测试吗?



# 自动机可对很多不同的计算建模

- 例如，它们可以描述任何“小型计算机”的操作，例如闹钟或微波炉的控制组件
- 它们还用于词法分析器中，以识别编程语言中格式正确的表达式：

  `ab1` is a legal name of a variable in C

  `5u=` is not

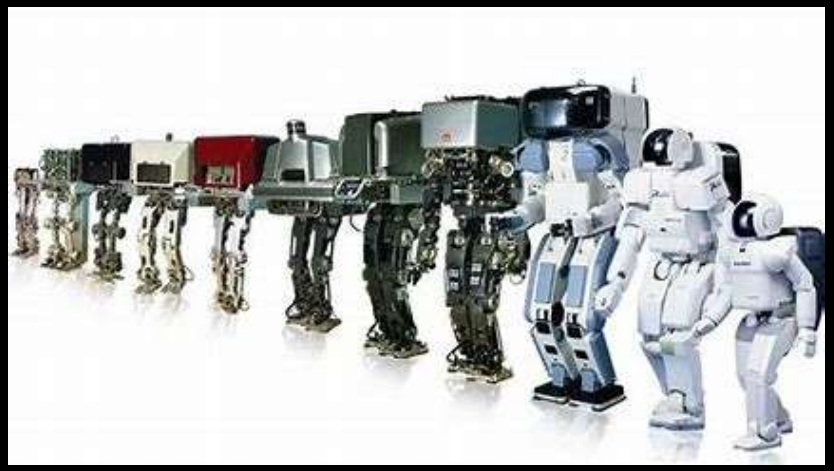
- 它们能用来证明一个计算机或程序是否正确
- 它们还能告诉我们有些数学问题其实是永远无法计算出答案的（真的？）

# 自动机理论的发展历史

| | |
|---|---|
| 1930s | • Alan Turing提出Turing Machine图灵机<br>• Decidability 可判定性<br>• Halting problem 停机问题 |
| 1940-1950s | • Finite automata 有穷自动机<br>• Formal languages 形式语言<br>• Noam Chomsky 提出形式语言的“Chomsky Hierarchy” 乔姆斯基分类 |
| 1969 | Cook发现“不易处理”的问题<br>即“NP-Hard”问题 |
| 1970- | 当代计算机科学: compilers 编译器,<br>computational & complexity theory<br>计算和复杂性理论 |

# 不同类型的自动机

- 我们发现，可以构造出不同种类的自动机
- 下面将学习不同的自动机，并研究以下问题：
  - 给定类型的自动机可以计算什么，其局限性是什么？
  - 一种自动机比另一种自动机功能强大吗？



# 将要学习的机器有：

| | |
|---|---|
| finite automata<br>有穷自动机 | 具有有限内存的设备。<br>用于为“小型”计算机建模。 |
| push-down automata<br>下推自动机 | 可以以受限方式访问无限内存的设备。<br>用于建模解析器等。 |
| Turing Machines<br>图灵机 | 具有无限内存的设备。<br>用于为任何计算机建模。 |

# 不同类型自动机的特点

- 有穷自动机
  - 我们将了解具有有限内存的设备可以做什么以及不能做什么
  - 引入仿真：一台设备“模仿”另一台设备的能力
  - 引入不确定性：设备做出任意选择的能力
- 下推自动机
  - 这些设备与语法有关，它们描述了编程（和自然）语言的结构

# 不同类型自动机的特点

- 图灵机
  - 这是计算机的通用模型，包括我们可能希望计算的任何内容
  - 令人惊讶的是，我们无法计算出许多东西，例如：

编写一个程序，给定另一个用C编写的代码，该程序将说明给定的代码是否输出单词“hello”

  - 看起来只需要读一下代码就能知道结论，但实际上这是不可能的！
    - Why？答案见最后一章……

# 自动机要解决的“问题”

- 如何将形式化表示下面问题：

设备 A 是否能求解 问题 B?

- 首先，我们需要一种形式化的方式来描述我们有兴趣解决的问题

# 自动机要解决的“问题”

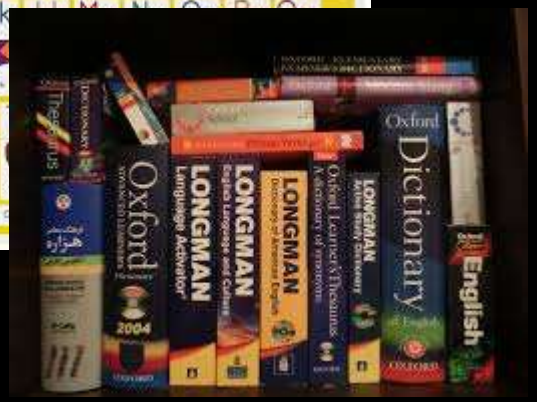
- 问题示例
  - 给定符号串s，它是否包含子符号串t?
  - 给定数字n，它是否可以被7整除?
  - 给定两个符号串s和t，它们是否相同?
  - 给定带括号的表达式，例如（（）（）），每个左括号是否与随后的右括号匹配?
- 所有这些都以“是/否”为答案，称为判定问题。
  - 还有其他类型的问题，询问“查找问题”或“计数问题”，但我们将不予考虑。
- 问题用formal languages 形式语言编写。

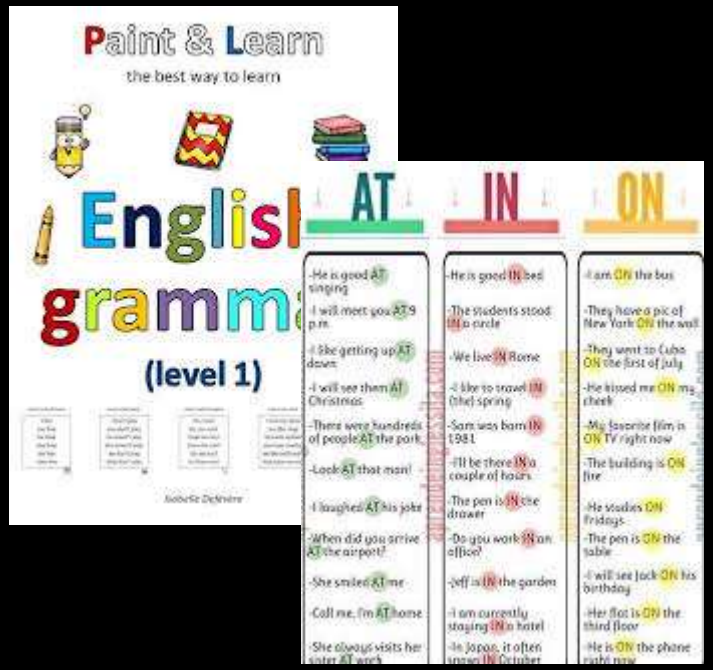
# 什么是语言？


- 自然语言：汉语、英语、日语……
  - 强大，但有可能不完整、不明确、有歧义
- 人工语言：数学语言、人造语言、编程语言……
- 形式语言
  - 完整、明确、无歧义


# 我们如何学习英语？

- 学字母、背单词
- 学习语法、句型
- 会话、阅读、写作


# 什么是形式语言?

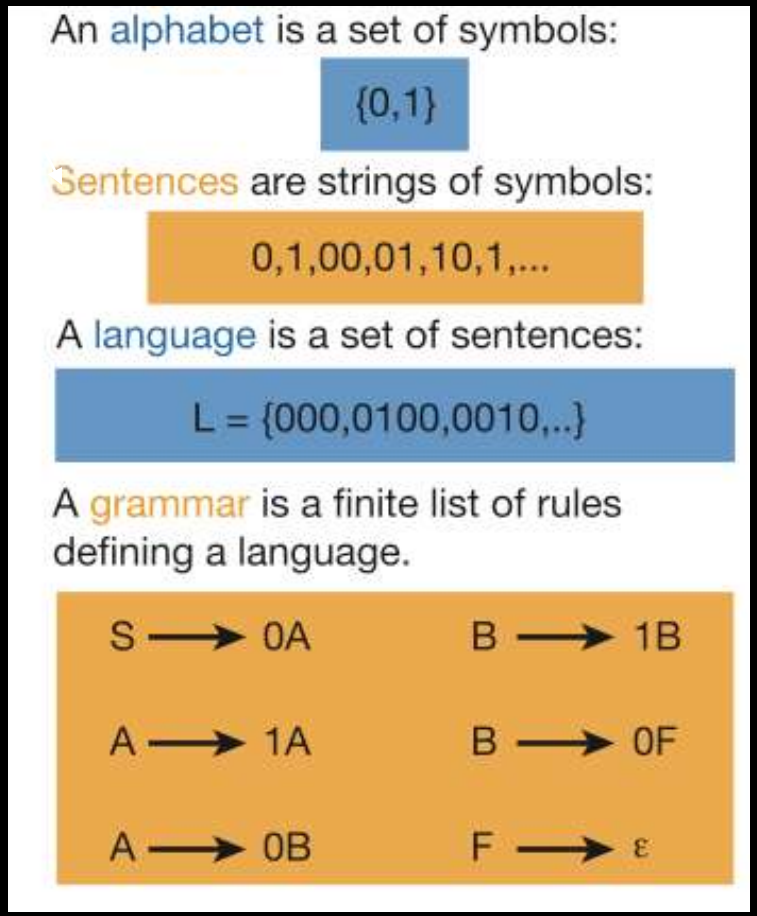



Image source: Nowak et al. Nature, vol 417, 2002

- 语言 Languages: "*A language is a collection of sentences of finite length all constructed from a finite alphabet of symbols*"
- 文法 Grammars: "*A grammar can be regarded as a device that enumerates the sentences of a language*" - nothing more, nothing less
- *- N. Chomsky, Information and Control, Vol 2, 1959*

# 形式语言

- 语言是有限长度的句子的集合
- 所有句子均由有限的符号构成的符号串
- 所有符号都来自于一个有限的字母表
- 语法是枚举语言中所有句子的装置
  - 如果一个句子属于该语言，则一定可以枚举出来
  - 如果枚举出一个句子，则一定属于该语言

# 字母表 Alphabet

- 处理单词、数字、词组、句子等的常见方法是将它们表示为符号串
- 为了定义符号串，我们从字母开始

字母表是符号的有限集合。

- 示例

$\Sigma_1 = \{a, b, c, d, \ldots, z\}$: the set of letters in English

$\Sigma_2 = \{0, 1, \ldots, 9\}$: the set of (base 10) digits

$\Sigma_3 = \{a, b, \ldots, z, \#\}$: the set of letters plus the special symbol #

$\Sigma_4 = \{ (, ) \}$: the set of open and closed brackets

# 更多的字母表例子

- ASCII, Unicode, {0,1} (*binary alphabet* ), {a,b,c}.

GB2312


Manchu Alphabet

# 句子/符号串 Sentences/Strings

字母表Σ上的符号串是Σ中符号的有限序列.

- 空串，即长度为0的符号串，写作ε
- 示例

abfbz is a string over $\Sigma_1 = \{a, b, c, d, \ldots, z\}$

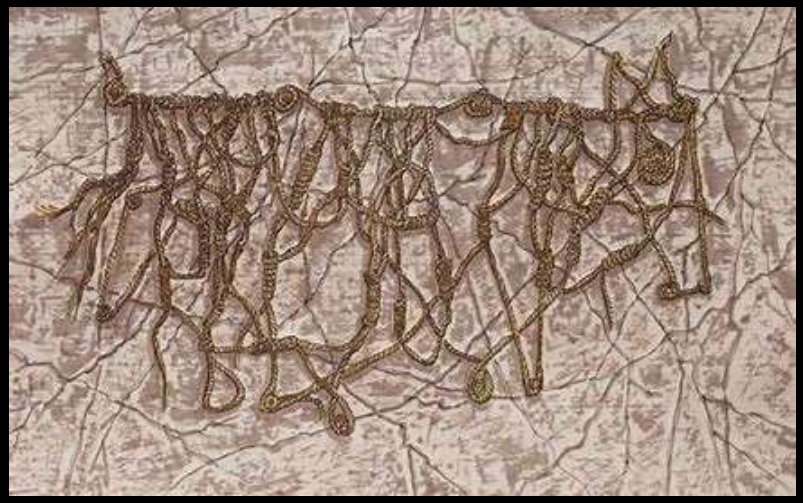
9021 is a string over $\Sigma_2 = \{0, 1, \ldots, 9\}$

ab#bc is a string over $\Sigma_3 = \{a, b, \ldots, z, \#\}$

))()(() is a string over $\Sigma_4 = \{ (, ) \}$

# 符号串集合

- Σ*表示Σ上所有可能符号串的集合
- {0,1}* = {ϵ, 0, 1, 00, 01, 10, 11, 000, 001, . . . }
- 一点说明：单个的符号有时作为符号串，有时作为符号，通过上下文确定类型。



# 语言 Languages

语言是给定字母表上符号串的一个集合.

- 语言可用来描述答案为“是/否” 的判定问题，例如：

$L_1$ = The set of all strings over $\Sigma_1$ that contain the substring “fool”

$L_2$ = The set of all strings over $\Sigma_2$ that are divisible by 7
= {7, 14, 21, …}

$L_3$ = The set of all strings of the form s#s where s is any string over {a, b, …, z}

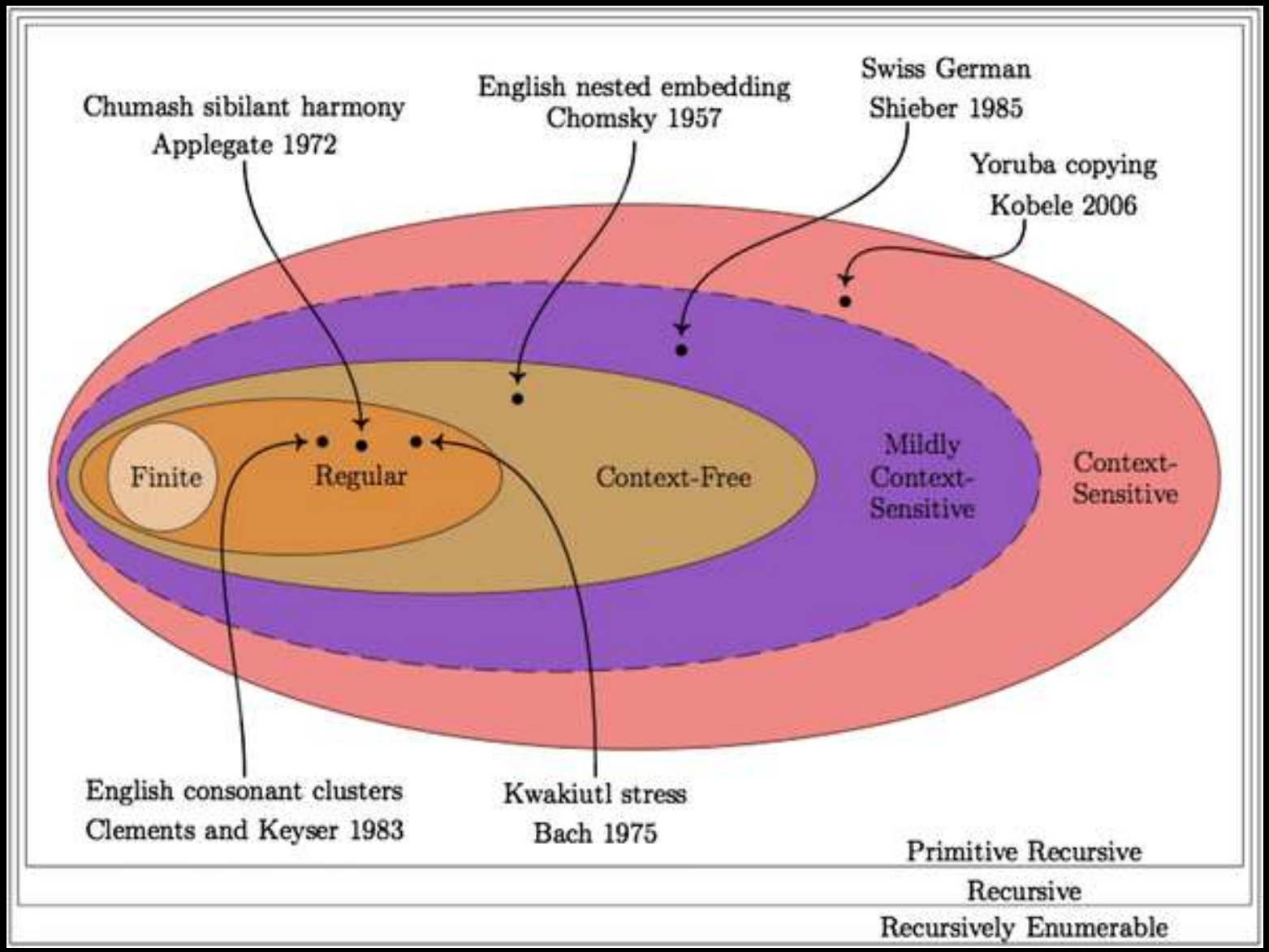
$L_4$ = The set of all strings over $\Sigma_4$ where every（ can be matched with a subsequent ）

# 语言

- 给定字母表Σ， Σ上的一个语言是Σ*的一个子集
- 示例：由0和1组成但没有连续两个1的符号串集，。
- L = {ϵ, 0, 1, 00, 01, 10, 000, 001, 010, 100, 101, 0000, 0001, 0010, 0100, 0101, 1000, 1001, 1010, . . . }

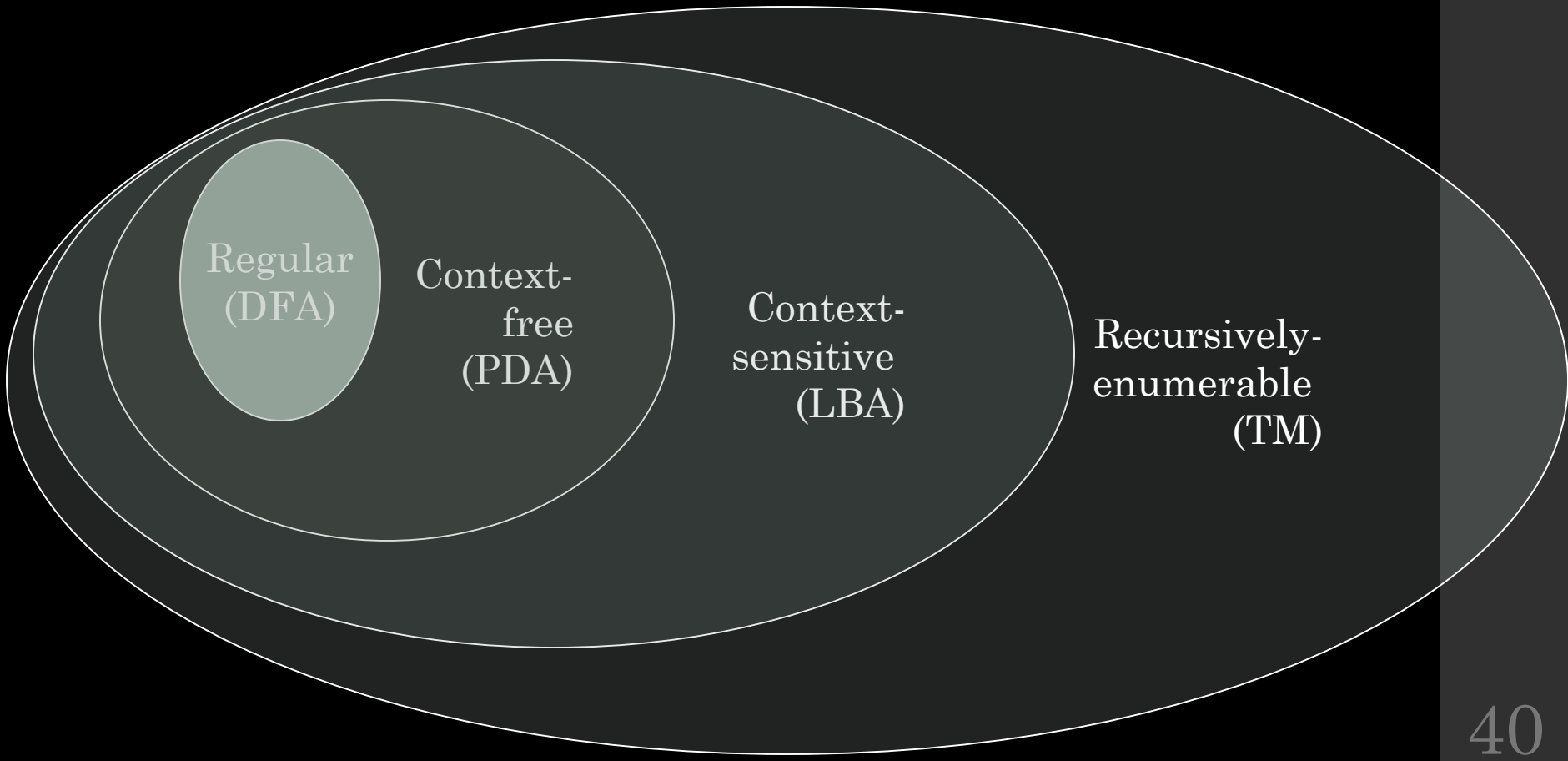
1个长度为0， 2个长度为1，3个长度为2，5个长度为3，8个长度为4。
多少个长度为5、为6?

# 形式语言研究简史


Chumash sibilant harmony
Applegate 1972
English nested embedding
Chomsky 1957
Swiss German
Shieber 1985
Yoruba copying
Kobele 2006
Finite
Regular
Context-Free
Mildly
Context-
Sensitive
Context-
Sensitive
English consonant clusters
Clements and Keyser 1983
Kwakiutl stress
Bach 1975
Primitive Recursive
Recursive
Recursively Enumerable

# 乔姆斯基分类

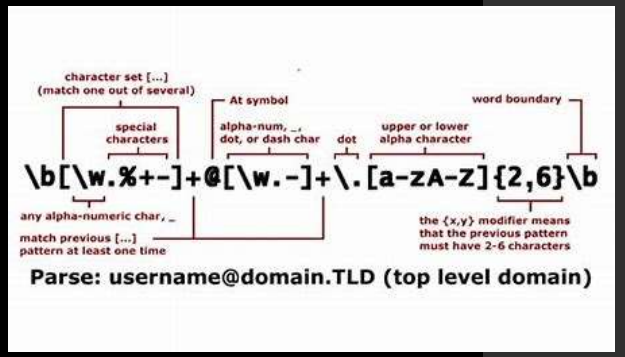
Chomsky Hierarchy



# 为什么要学习形式语言和自动机？

- 在CS的许多科目中很有用，如编译原理
- 作为清晰的计算模型
- 分析或解决现实和理论问题的方便工具

# 例如？

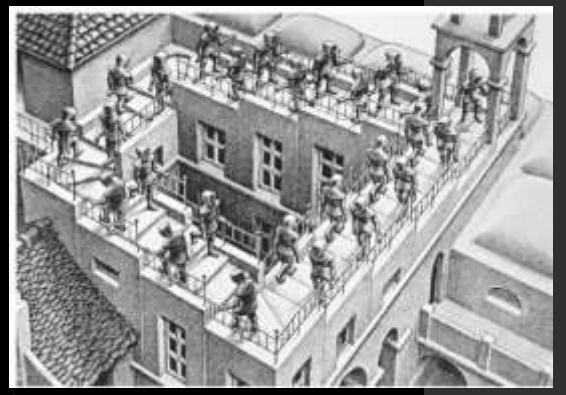



- 正则表达式(RE)用于许多系统中
  - UNIX a.*b.
  - DTD使用RE格式描述XML标签，例如person (name, addr, child*).
- 有穷自动机对协议、数字电路建模进行分析
  - 模型检查 *model-checking*.

# 例如？ – (2)

- 上下文无关语言(CFG)用于描述基本上所有编程语言的语法。
- 也不要忘记它们在描述自然语言中的重要作用。
- 从整体上看，DTD也是一种CFG。

# 例如? – (3)



- 在开发解决实际问题的解决方案时，经常会遇到软件能做到什么的限制。
  - *Undecidable*不可判定的问题 – 任何程序都无法做到。
  - *Intractable*不易处理的问题 – 有程序可以做到，但没有快速的方法。
- 本课程将提供相关的分析工具。

# 课程大纲

- 有穷自动机和正则语言
  - 有穷自动机 Deterministic finite automata (DFA)
  - 非确定有穷自动机 Nondeterministic finite automata (NFA)
  - 正则语言 Regular languages
  - 正则表达式 Regular expressions (RE)
  - 正则语言的判定性质 Decision properties
  - 正则语言的闭包性质 Closure properties
  - 有穷自动机的构造、转换、最小化等算法
  - 等价性证明
  - 正则语言各种性质的证明

# 课程大纲– (2)

- 下推自动机和上下文无关语言
  - 上下文无关语言 Context-free languages (CFL)
  - 上下文无关文法 Context-free grammars (CFG)
  - 下推自动机 Pushdown automata (PDA)
  - 判定和闭包性质 Decision and closure properties
  - 相关算法和证明
  - 在编程语言中的应用

# 课程大纲– (3)

- 图灵机和递归可枚举语言
  - 递归语言和递归可枚举语言 Recursive and recursively enumerable languages
  - 图灵机 Turing machines (TM)
  - 问题的可判定性 Decidability of problems
  - 可计算的边界和限制
- 不易处理的问题 Intractable problems
  - 不能在多项式时间内解决的问题
  - NP完全和NP难（选讲）

# JFLAP

- JFLAP是用于形式语言实验的软件，支持
  - 非确定有穷自动机
  - 非确定下推自动机
  - 多带图灵机
  - 数种类型的文法，
  - 解析和L系统。
- 它还允许尝试从一种形式到另一种形式的构造证明，例如
  - 将NFA转换为DFA，将最小状态DFA转换为正则表达式或正则文法

# 作业1

- 阅读第1章
- 下载和安装JFLAP
  http://www.jflap.org/
  - 需要Java环境
- 运行JFLAP并尝试跟着说明文档学习一遍基本操作
- 上传一个用JFLAP画的自动机截图