计算机科学与技术学科前沿丛书

计算机科学与技术学科研究生系列教材（中文版）

形式语言与自动机理论引论

蒋宗礼 姜守旭 编著

清华大学出版社

计算机科学与技术学科前沿丛书
计算机科学与技术学科研究生系列教材（中文版）

形式语言与自动机理论引论

蒋宗礼 姜守旭 编著

清华大学出版社
北京

内 容 简 介

形式语言与自动机理论因其以体现计算学科中模型描述、模型研究和模型计算为问题求解的主要特征而成为计算机科学与技术、软件工程、网络空间安全等计算机类学科教育的最重要的内容之一。本书按照我国当前计算机类及相关学科研究生教育实际需求，结合作者30余年的教学实践编著而成，以正则语言与上下文无关语言的文法、识别模型及其性质，以及图灵机基本知识为载体，分9章讨论相关内容，力图强化学生基于模型的建立、研究、处理，实现问题求解的意识，让学生掌握相应的基本方法，提升解决问题的能力与水平。

本书适合计算机类及相关学科研究生使用，也可以供相关专业高年级本科生、教师和科研人员参考。

图书在版编目(CIP)数据

形式语言与自动机理论引论/蒋宗礼，姜守旭编著. —北京：清华大学出版社，2017
(计算机科学与技术学科前沿丛书)
(计算机科学与技术学科研究生系列教材(中文版))
ISBN 978-7-302-45602-5

Ⅰ. ①形…　Ⅱ. ①蒋… ②姜…　Ⅲ. ①形式语言－研究生－教材 ②自动机理论－研究生－教材　Ⅳ. ①TP301

中国版本图书馆CIP数据核字(2016)第283907号

责任编辑：张瑞庆
封面设计：傅瑞学
责任校对：李建庄
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954

印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：16.25　　**字　　数**：395千字
版　　次：2017年3月第1版　　**印　　次**：2017年3月第1次印刷
印　　数：1～1000
定　　价：39.00元

产品编号：071877-01

前　言

FOREWORD

计算机科学与技术、软件工程、网络空间安全等计算机类学科，统称为计算学科。学科通过在计算机上建立模型和系统，模拟实际过程进行科学调查和研究；通过数据搜集、存储、传输与处理等进行问题求解，包括科学、工程、技术和应用4部分。其科学部分的核心在于通过抽象建立模型实现对计算规律的研究；其工程部分的核心在于根据规律低成本地构建从基本计算系统到大规模复杂计算应用系统的各类系统；其技术部分的核心在于研究和发明用计算进行科学调查和研究中使用的基本手段和方法；其应用部分在于构建、维护和使用计算系统实现特定问题的求解。其根本问题是“什么能且如何被有效地自动计算”，而计算的自动化，均以问题描述和处理的符号化为基础。从事计算机科学研究、工程设计、开发、运行、应用、维护的计算机人才都不例外。也正因为如此，才有人用“问题抽象”、“系统抽象”、“数据抽象”来表述对计算机类专业人才的基本要求。

2016年6月2日，在吉隆坡召开的国际工程联盟大会上，中国获全票通过，正式加入《华盛顿协议》，成为第18个正式成员，这被认为是中国高等教育取得的具有里程碑意义的历史性突破。该协议虽然是针对本科教育的，但对我们按照工程教育的要求，不断提升研究生教育的水平与质量也是很重要的。根据《华盛顿协议》，工程被定义为包括数学、自然科学和工程知识、技术和技能整体的、有目的性的应用。可见，“工程”对基础的强调。另外，《华盛顿协议》还规定，两年制的专科教育旨在培养学生解决狭义工程问题的能力，三年制专科旨在培养学生解决广义工程问题的能力，本科教育则聚焦培养学生解决复杂工程问题的能力。所谓的复杂工程问题是满足如下条件的工程问题：

（1）必须运用深入的工程原理经过分析才可能解决；

（2）需求涉及多方面的技术、工程和其他因素，并可能相互有一定冲突；

（3）需要通过建立合适的抽象模型才能解决，在建模过程中需要体现出创造性；

（4）不是仅靠常用方法就可以完全解决的；

（5）问题中涉及的因素可能没有完全包含在专业标准和规范中；

（6）问题相关各方利益不完全一致；

（7）具有较高的综合性，包含多个相互关联的子问题。

作为本科教育后继的研究生教育，显然应该在本科教育的基础上开展。但是目前存在的问题是，很多年来，由于各方面的限制，部分专业点的教育还很难说达到了这一基本定位的要求，表现出来的现象是，进入研究生阶段学习的学生在对“深入原理的掌握”、“分析方法的掌握”，从而具有“分析能力”以及“建立合适的抽象模型”并且“在建模过程中需要体现出

创造性”等方面都存在一定的差距。建模能力是如此之重要，以至于《华盛顿协议》还明确要求包括机械、电子、化工、建筑等在内的所有工科专业的教育都要包括数学、数值分析、统计，以及计算机和信息科学关于形式化方面的基本内容，以支持学科问题的分析和建模（WK2：(mathematics and computer) conceptually-based mathematics, numerical analysis, statistics and formal aspects of computer and information science to support analysis and modeling applicable to the discipline）。作为培养工程应用型人才为主的计算机类学科专业，更应如此。而形式语言与自动机理论则是培养学生这些方面能力的非常恰当的重要载体。希望通过这类课程的教育，支持这一要求的达成，同时促进学生在基础理论上更好地达到我国学位条例中“在本门学科上掌握坚实的基础理论和系统的专门知识”的要求。

另外，从复杂工程问题的定义不难看出，解决这类问题，需要有扎实的基础理论和深入的“原理性”知识，同时要强调对基本理论的应用。所以，包括研究生教育在内，需要追求理论指导下的、非简单的、高水平实践能力的培养，以追求高水平动脑指挥下的问题求解（实践、动手）。落实到本教材对应的课程教育，绝不仅仅是要让学生掌握其基本知识，而是要以这些知识为载体，使学生掌握其中的思想和典型方法，培养他们模型描述和模型计算的意识和能力。为此，要开展研究型的教与学：教师在对问题的研究中教，学生在对未知的研究中学，努力体现基于产出（outcome-based education，OBE）的基本教育观点，纠正基于课程的教育观点（curricular-based education，CBE）。

为了达到这一目的，在本书的写作中，我们除了以定义、定理、算法的形式叙述基本概念和结论外，还尽力进行相关的问题分析。希望读者不要限于简单地记忆定义、定理和相应的算法，要通过深入的分析，掌握相关的知识、思想和方法。多提问题，多想问题，这样才能有真正的收获。

本书是基于国家精品教材《形式语言与自动机》编著的，重点考虑了研究生教育的需求，全书共分 9 章。第 1 章介绍语言和文法，包括相关的基本概念、文法的构造、Chomsky 体系、左线性文法和右线性文法。第 2 章讨论有穷状态自动机，包括有穷状态自动机的定义、描述、构造方法，确定的、非确定的、带空移动的有穷状态自动机以及与正则文法的等价性。第 3 章研究正则表达式及其与有穷状态自动机的等价性。第 4 章讨论正则语言的性质，包括正则语言的泵引理、封闭性、Myhill-Nerode 定理、极小化、判定算法。第 5 章为上下文无关语言，包括文法二义性、派生树、化简、Chomsky 范式和 Greibach 范式。第 6 章讨论下推自动机，包括基本定义和构造方法以及下推自动机与上下文无关文法的等价性。第 7 章是上下文无关语言的性质，包括上下文无关语言的泵引理、Ogden 引理、封闭性、判定算法。第 8 章介绍图灵机，包括基本定义、接受的语言、构造技术、通用图灵机、Church-Turing 论题、图灵机的变形、可计算语言、不可判定性、P-NP 问题。第 9 章介绍上下文有关语言，包括图灵机与短语结构文法的等价性，以及线性界限自动机的定义及与上下文有关语言的关系。

书中难免存在不足，请读者不吝赐教。联系地址：jiangzl@bjut.edu.cn。

作者

2017 年 1 月

目　录

CONTENTS

第 1 章 语言与文法

计算学科通过在计算机上建立模型并模拟物理过程来进行科学调查和研究，系统地研究信息描述和变换算法，主要包括信息描述和变换算法的理论、分析、效率、实现和应用，根本问题是什么能被(有效地)自动计算。

问题的计算机求解建立在高度的抽象上，问题的符号表示及其处理过程的机械化的固有特性，决定了数学是计算学科的重要基础之一，数学及其形式化描述、严密的表达和计算，是计算机科学与技术学科所用的重要工具。建立物理符号系统并对其实施变换是计算机学科进行问题的描述和求解的重要手段。学科所要求问题求解的“可行性”，使得我们必须遵循从问题抽象开始，并依据理论的指导进行设计(实现)的科学实践过程，“可行性”所需要“形式化”后呈现出的“离散”特征实质上确定了计算学科进行问题求解的重要特征。经过形式化，一类问题的描述和处理将变成“语言”的描述与处理。

语言学家乔姆斯基（Avram Noam Chomsky）最初从产生语言的角度研究语言。通过抽象，他将语言形式地定义为一个字母表中的字母组成的串的集合：对任何语言 L，有一个字母表 Σ，使得 $L\subseteq\Sigma^*$。而在实际应用中，很少会一次将一个语言的所有句子都产生出来，往往只需要考查一个给定的字符串是否为一个给定语言的句子。如果是，它的具体组成结构是什么样的？如果不是，它在结构的什么地方出了错？进一步地，这个错是什么样的错？如何更正？……这些问题对有穷语言来说是比较容易解决的。但是，对一个无穷语言，即使它具有适当的有穷描述，处理起来也是比较困难的。其原因在于，对有穷语言来说，当判定一个字符串是否是该语言的一个句子时，至少可以顺序地扫描这个语言，用这个字符串去与语言中的句子分别进行匹配，一旦匹配成功，表明这是一个合法的句子。如果扫描完所有的句子都未能找到匹配的，表明它是非法的。由于该语言是有穷的，所以，无论被考查的字符串是否为该语言的合法句子，这个过程都必定会结束，因此它是一个可行的算法。可以通过设置恰当的数据结构来存放语言中的所有句子，并配以适当的算法，以此提高判定的效率。由于是逐个进行匹配的，所以，当这个字符串不是该语言的合法句子时，要想回答诸如“这个错是什么样的错？”、“如何更正？”之类的问题是比较容易的。但是，当语言是无穷的时候，这种方法就难以生效了。

然而，在实际工作中，遇到的大多数都是关于无穷语言的问题，或者语言所含的句子是如此的多，以至于把它当成有穷语言处理时难以获得有效的结果。例如，解决问题所需要的时间是如此之长，使得系统无法忍受；或者所需要的存储空间是如此之大，使得系统难以提

供。这些都要求我们能从语言句子的一般特征去考虑问题，这就是语言的有穷描述。我们希望语言的有穷描述能够表达出语言的结构特征。对于一个具体的字符串，当判定它是否为一个给定语言的句子时，可以通过分析句子的“结构”，看它是否符合相应语言的有穷描述所表达出来的结构特征。如果符合，则该字符串是给定语言的合法句子；否则它是非法的。当它是非法的时候，仍然可以根据语言的有穷描述来判定出相应错误的类型，并且知道应该如何更正。

这种用语言的有穷描述来表达语言的方法对一般的语言都是有效的。尤其在用计算机系统去判断一个句子是否是某语言的合法句子时，从句子和语言的结构特征上着手更是非常重要的。一般可以通过看这个句子是否能由给定语言对应的文法产生来做出判断，如果能，它就是合法的；否则，它就是非法的。对一类语言，可以在字母表上按照一定的规则，根据语言的结构特点，定义一个文法。用文法作为相应语言的有穷描述不仅可以描述出语言的结构特征，而且还可以产生这个语言的所有句子。

新的问题是，文法是什么样的？对于给定的语言，如何根据该语言的特征构造出它的文法？文法如何产生句子，如何产生语言？语言的文法有什么特征？本章将讨论这一类问题。讨论将从问题的抽象及一般化的抽象处理开始，希望读者逐渐习惯这种新的方法。

1.1 语　　言

1.1.1 什么是语言

什么是语言呢？它仅仅是由一些字组成的么？下面先看一些自然情况。

在现实世界中，存在多种多样的语言，它们被一定的群体用作信息交流的工具，可以通过音、形（书面、口头、动作）表达信息，而且必须有一系列的生成规则、理解（语义）规则，只有当使用者都按照这些规则来构造“句子”和理解“句子”时，才能达到交流信息的目的。

例如，人们都知道句子“我是一名研究生”的含义。但是，又有谁能够说出“名是生研我一究”所表达的含义呢？通过分析发现，虽然它与句子“我是一名研究生”用的是同一组汉字，但“名是生研我一究”并没有按照人们共同的约定组合。所以，它不能表达人们能够懂得的含义。这就是说，作为一种语言，除了组成句子的基本字外，还需要配套的组合规则，且这些规则应该是使用该语言的群体所共同遵守的。否则，就难以实现意思的表达、传递和理解。

斯大林曾经从强调语言的作用出发，把语言定义为“为广大的人群所理解的字和组合这些字的方法”。语言学家韦波斯特（Webster）将语言定义成“为相当大的团体的人所懂得并使用的字和组合这些字的方法的统一体”。

这些定义用来建立语言的数学模型，以对语言的性质进行研究是不够精确的。为此，需要将语言抽象成一个数学系统，其形式性足以支持给出语言的严格描述，并能将由此发展出的知识（理论）用到适当的模型中，使之能够在人们的科学实践中起到良好的指导作用。这就是形式语言。它通过抽象，对语言结构、描述（生成与识别）、性质、理论进行研究。

1.1.2 形式语言与自动机理论的产生

毕业于美国宾夕法尼亚大学的语言学家乔姆斯基最初从产生语言的角度研究语言。

1956 年，通过抽象，他将语言形式地定义为是由一个字母表中的字母组成的一些串的集合：对任何语言 L，有一个字母表 Σ，使得 $L \subseteq \Sigma^*$。可以在字母表上按照一定的规则定义一个文法(grammar)，该文法产生的所有句子组成的集合就是该文法描述的语言。判断一个句子是否是某语言的句子，需要判断该句子是否能由该语言对应的文法产生出来。如果能，它就是；否则，它就不是。1959 年，乔姆斯基根据产生语言的文法的特性，又将语言划分成三大类。注意，这里所说的文法就是通常人们所说的语法。根据习惯，本书主要称之为“文法”，只是在个别地方用“语法”一词。

1951—1956 年间，克林(Kleene)在研究神经细胞的过程中建立了自动机，从识别的角度研究语言，从而给出了语言的另一种描述模型：对于按照一定的规则构造的任一个自动机，该自动机就定义了一个语言，这个语言由该自动机所能识别的所有句子组成。

语言作为同一个对象，它的两种不同表示模型应该是等价的。但是，它们真的是等价的么？如果它们确实是等价的，是否存在一种方法，实现二者之间的等价变换？如果这种等价变换还可以自动化地进行，将给人们带来更多的方便和结果。

1959 年，乔姆斯基将他本人的研究成果与克林的研究成果结合了起来，证明了文法与自动机的等价性。此时形式语言才真正诞生，并被置于了数学的光芒之下。

形式语言出现之后，很快就在计算机科学与技术领域中找到了应用。20 世纪 50 年代，人们用巴科斯范式(Backus-Naur form 或 Backus normal form，BNF)成功地实现了对高级语言 ALGOL-60 的描述。实际上，巴科斯范式就是上下文无关文法(context free grammar，CFG)的一种表示形式。这一成功，使得形式语言在 20 世纪 60 年代得到了大力的发展。尤其是上下文无关文法，被作为计算机程序设计语言文法的最佳的近似描述而得到了较为深入的研究。后来，人们又将该文法用到了模式匹配、模型化处理等方面，并成为算法描述和分析、计算复杂性理论、可计算性理论等研究的基础。

实际上，形式语言与自动机理论除了在计算机科学领域中的直接应用外，更在计算机科学与技术学科的人才培养中占有极其重要的地位，人们甚至将一个人是否学习并掌握有关方面的知识、是否有相应的修养作为衡量他(她)是否受到过良好的计算机科学学科训练的一个标准。事实上，支持问题分析与建模的计算机与信息科学的形式化思想、方法、技术也受到了国际工程联盟的极大重视，该组织在《华盛顿协议》中明确地将其作为工程类本科生教育的必须掌握知识的第二条的重要内容(WK2：conceptually-based mathematics，numerical analysis，statistics and formal aspects of computer and information science to support analysis and modeling applicable to the discipline)，所以，作为一名计算机类学科和信息类学科的研究生，更应该在这些方面有较深的造诣。

1.1.3　基本概念

对语言的研究，包括以下 3 个方面。

首先是如何给出语言的**表示**(representation)。如果语言只包含有穷多个句子，这个问题就比较简单了，因为只要简单地列出所有的句子就可以了(在后面会看到，有穷集是结构最简单的正则语言)。当语言含有无穷多个语句的时候，它的表示就成了问题，对人们来说，关心的是它的有穷描述。

第二个问题是一个给定的语言是否存在**有穷描述**(finite description)。是否所有的语

言都有有穷描述呢？答案是否定的。按照乔姆斯基的定义，语言是由字母表上的字符串组成的，对任意的一个字母表，该字母表的所有字符串组成的集合是可数无穷的。从而一个语言要么是有穷的，要么是可数无穷的——含可数无穷多个句子。根据集合论中关于可数无穷集合的子集的结论，这个可数无穷集的所有子集组成的集合（即它的幂集）是一个不可数的集合。这就是说，一个字母表上有不可数无穷多个语言。而有穷表示，无论它是什么样的，只能有可数无穷多个。众所周知，可数无穷集和不可数无穷集之间是不存在一一对应的。而且，按照不严格的说法，一个不可数无穷集中所含的元素“远远地多于”一个可数无穷集中所含的元素。因此，存在有这样的语言，它不存在有穷表示。

第三个问题是具有有穷表示的语言的**结构**(structure)是什么样的？它们有什么样的特性？对某一类语言来说，这是非常重要的。

本书将主要讨论正则语言和上下文无关语言的各种表示、表示的等价性，以及它们的性质。

为了便于相关的讨论，先定义以下基本概念。

定义 1-1 **字母表**(alphabet)是一个非空有穷集合，字母表中的元素称为该字母表的一个**字母**(letter)，也可叫作**符号**(symbol)或者**字符**(character)。

值得注意的是，字母表具有非空性和有穷性。本书通常用 Σ 表示字母表。

例 1-1 以下是不同的字母表：

(1) $\{a,b,c,d\}$。

(2) $\{a,b,c,\cdots,z\}$。

(3) $\{0,1\}$。

(4) $\{aa,ab,bb\}$。

(5) $\{a,a',b,b'\}$。

(6) $\{\infty,\wedge,\vee,\geqslant,\leqslant\}$。

字母表中的字母是组成字母表上的语言中的任何句子的最基本元素，所以，字母表中的字符必须具有如下两个特点。

(1) **整体性**(monolith)，也叫不可分性。例如，$\{aa,ab,bb\}$ 中的 aa、ab、bb 均是单个字符，aa 不能被拆分成两个 a，ab 不能被拆分成一个 a 和一个 b，bb 也不能被拆分成两个 b。

$\{a,a',b,b'\}$ 中的 a 和 a' 是两个不同的字符，b 和 b' 也是两个不同的字符。对字符 a' 来说，a 和′是不可分的，同样 b 和′也是不可分的。

为了简明起见，一般都不用类似 a' 和 aa 的字符作为字母表中的字母，尤其是在本书的讨论中，除非特殊需要，将不用这样的字符作为字母表中的字母，以免在讨论中引起不必要的麻烦。

(2) **可辨认性**(distinguishable)，也叫可区分性。这一点要求字母表中的字符是两两不相同的，必须明确区分。如 $\{a,b,a\}$ 就不是字母表，因为其中的 a 和另一个 a 是无法区分的。换句话说，字母表不可以是多重集。

定义 1-2 设 Σ_1,Σ_2 是两个字母表，Σ_1 与 Σ_2 的**乘积**(product)：

$$\Sigma_1\Sigma_2=\{ab\mid a\in\Sigma_1,b\in\Sigma_2\}$$

例 1-2 字母表的乘积。

(1) $\{0,1\}\{0,1\}=\{00,01,10,11\}$。

(2) $\{0,1\}\{a,b,c,d\}=\{0a,0b,0c,0d,1a,1b,1c,1d\}$。

(3) $\{a,b,c,d\}\{0,1\}=\{a0,a1,b0,b1,c0,c1,d0,d1\}$。

(4) $\{aa,ab,bb\}\{0,1\}=\{aa0,aa1,ab0,ab1,bb0,bb1\}$。

显然,字母表的乘积不具有交换律。

定义 1-3　设 Σ 是一个字母表,Σ 的 n 次**幂**(power)递归地定义为:

(1) $\Sigma^0=\{\varepsilon\}$。

(2) $\Sigma^n=\Sigma^{n-1}\Sigma, n\geqslant 1$。

其中,ε 是由 Σ 中的 0 个字符组成的。

定义 1-4　设 Σ 是一个字母表,Σ 的**正闭包**:

$$\Sigma^+=\Sigma\cup\Sigma^2\cup\Sigma^3\cup\Sigma^4\cup\cdots$$

Σ 的**克林闭包**:

$$\Sigma^*=\Sigma^0\cup\Sigma^+=\Sigma^0\cup\Sigma\cup\Sigma^2\cup\Sigma^3\cup\cdots$$

例 1-3　字母表的闭包。

(1) $\{0,1\}^+=\{0,1,00,01,10,11,000,001,010,011,100,\cdots\}$。

(2) $\{0,1\}^*=\{\varepsilon,0,1,00,01,10,11,000,001,010,011,100,\cdots\}$。

(3) $\{a,b,c,d\}^+=\{a,b,c,d,aa,ab,ac,ad,ba,bb,bc,bd,\cdots,aaa,aab,aac,aad,aba,abb,abc,\cdots\}$。

(4) $\{a,b,c,d\}^*=\{\varepsilon,a,b,c,d,aa,ab,ac,ad,ba,bb,bc,bd,\cdots,aaa,aab,aac,aad,aba,abb,abc,\cdots\}$。

通常,有

$\Sigma^*=\{x\mid x$ 是 Σ 中的若干个(包括 0 个)字符连接而成的字符串$\}$

$\Sigma^+=\{x\mid x$ 是 Σ 中的至少一个字符连接而成的字符串$\}$

定义 1-5　设 Σ 是一个字母表,$\forall x\in\Sigma^*$,x 叫作 Σ 上的一个**句子**(sentence)。两个句子称为相等的,如果它们对应位置上的字符都对应相等。

句子还可以称为**字**(word)、(**字符**、**符号**)**行**(line)、(**字符**、**符号**)**串**(string)。

定义 1-6　设 Σ 是一个字母表,$x,y\in\Sigma^*$,$a\in\Sigma$,句子 xay 中的 a 叫作 a 在该句子中的一个**出现**(appearance)。

设 $xay\in\Sigma^*$,则

(1) 当 $x=\varepsilon$ 时,a 的这个出现为字符串 xay 的首字符,也就是该字符串的第 1 个字符。

(2) 如果 a 的这个出现是字符串 xay 的第 n 个字符,则 y 的首字符是字符串 xay 的第 $n+1$ 个字符。

(3) 当 $y=\varepsilon$ 时,a 的这个出现是字符串 xay 的尾字符。

例 1-4　字母的出现。

设字母表 $\Sigma=\{a,b\}$,则 $abaabb$ 是 Σ 上的一个字符串。在这个字符串中,字母 a 有 3 个不同的出现:第 1 个出现是该字符串的首部,第 2 个出现是该字符串的第 3 个字符,第 3 个出现是该字符串的第 4 个字符。字母 b 也有 3 个不同的出现。

定义 1-7　设 Σ 是一个字母表,$\forall x\in\Sigma^*$,句子 x 中字符出现的总个数叫作该句子的**长度**(length),记作 $|x|$。

长度为 0 的字符串叫**空句子**,将空句子记作 ε。

例 1-5 句子的长度。

字母表 $\Sigma=\{a,b\}$ 上的字符串 $abaabb$ 的长度为 6，$bbaa$ 的长度为 4，ε 的长度为 0，$bbabaabbbaa$ 的长度为 11。即

$|abaabb|=6$

$|bbaa|=4$

$|\varepsilon|=0$

$|bbabaabbbaa|=11$

注意：

(1) ε 是一个句子。

(2) $\{\varepsilon\}\neq\varnothing$。这是因为$\{\varepsilon\}$不是一个空集，它是含有一个空句子 ε 的集合。$|\{\varepsilon\}|=1$，而$|\varnothing|=0$。

定义 1-8 设 Σ 是一个字母表，$x,y\in\Sigma^*$，x,y 的**并置**(concatenation)是这样一个串，该串是由串 x 直接接串 y 所组成的，记作 xy。并置又叫作**连接**。

对于 $n\geqslant 0$，串 x 的 n 次**幂**递归地定义为：

(1) $x^0=\varepsilon$。

(2) $x^n=x^{n-1}x$。

例如，$\{0,1\}$上的串 $x=001$，$y=1101$，则 $xy=0011101$；如果 $x=0101$，$y=110110$，则 $xy=0101110110$。

对 $x=001$，$y=1101$，有 $x^0=y^0=\varepsilon$，$x^4=001001001001$，$y^4=1101110111011101$。

对 $x=0101$，$y=110110$，有 $x^2=01010101$，$y^2=110110110110$，$x^4=0101010101010101$，$y^4=1101101101101101101101101101$。

不难证明，对 Σ^* 上的任意串 x,y,z，并置具有如下性质。

(1) 结合律：$(xy)z=x(yz)$。

(2) 左消去律：如果 $xy=xz$，则 $y=z$。

(3) 右消去律：如果 $yx=zx$，则 $y=z$。

(4) 唯一分解性：存在唯一确定的 $a_1,a_2,\cdots,a_n\in\Sigma$，使得 $x=a_1a_2\cdots a_n$。

(5) 单位元素：$\varepsilon x=x\varepsilon=x$。

定义 1-9 设 Σ 是一个字母表，$x,y,z\in\Sigma^*$，且 $x=yz$，则称 y 是 x 的**前缀**(prefix)；如果 $z\neq\varepsilon$，则称 y 是 x 的**真前缀**(proper prefix)。z 是 x 的**后缀**(suffix)，如果 $y\neq\varepsilon$，则称 z 是 x 的**真后缀**(proper suffix)。

例 1-6 句子的前缀、后缀、真前缀和真后缀。

字母表 $\Sigma=\{a,b\}$ 上的句子 $abaabb$ 的前缀、后缀、真前缀和真后缀如下。

前缀：$\varepsilon,a,ab,aba,abaa,abaab,abaabb$。

真前缀：$\varepsilon,a,ab,aba,abaa,abaab$。

后缀：$\varepsilon,b,bb,abb,aabb,baabb,abaabb$。

真后缀：$\varepsilon,b,bb,abb,aabb,baabb$。

显然，对于字母表上任意给定的句子 x：

(1) x 的任意前缀 y 有唯一的一个后缀 z 与之对应，使得 $x=yz$；反之亦然。

(2) x 的任意真前缀 y 有唯一的一个后缀 z 与之对应，使得 $x=yz$；反之亦然。

(3) $|\{w|w$ 是 x 的后缀$\}|=|\{w|w$ 是 x 的前缀$\}|$。

(4) $|\{w|w$ 是 x 的真后缀$\}|=|\{w|w$ 是 x 的真前缀$\}|$。

(5) $\{w|w$ 是 x 的前缀$\}=\{w|w$ 是 x 的真前缀$\}\cup\{x\}$，

$|\{w|w$ 是 x 的前缀$\}|=|\{w|w$ 是 x 的真前缀$\}|+1$。

(6) $\{w|w$ 是 x 的后缀$\}=\{w|w$ 是 x 的真后缀$\}\cup\{x\}$，

$|\{w|w$ 是 x 的后缀$\}|=|\{w|w$ 是 x 的真后缀$\}|+1$。

(7) 对于任意字符串 w，w 是自身的前缀，但不是自身的真前缀；w 是自身的后缀，但不是自身的真后缀。

(8) 对于任意字符串 w，$w\neq\varepsilon$，ε 是 w 的前缀，且是 w 的真前缀；ε 是 w 的后缀，且是 w 的真后缀。

另外，为了以后叙述方便，约定：

(1) 用小写字母表中较为靠前的字母 $a,b,c,\cdots$ 表示字母表中的字母。

(2) 用小写字母表中较为靠后的字母 $x,y,z,\cdots$ 表示字母表上的句子。

(3) 用 x^{T} 表示 x 的倒序。例如，如果 $x=abc$，则 $x^{\mathrm{T}}=cba$。

定义 1-10　设 Σ 是一个字母表，$x,y,z,w,v\in\Sigma^*$，且 $x=yz$，$w=yv$，则称 y 是 x 和 w 的**公共前缀**(common prefix)；如果 x 和 w 的任何公共前缀都是 y 的前缀，则称 y 是 x 和 w 的**最大公共前缀**。如果 $x=zy$，$w=vy$，则称 y 是 x 和 w 的**公共后缀**(common suffix)；如果 x 和 w 的任何公共后缀都是 y 的后缀，则称 y 是 x 和 w 的**最大公共后缀**。

定义 1-11　设 Σ 是一个字母表，$w,x,y,z\in\Sigma^*$，且 $w=xyz$，则称 y 是 w 的**子串**(substring)。

例如，$\{0,1\}$ 上的串 $x=0101$ 的不同子串有 $\varepsilon,0,1,01,10,010,101,0101$；字母表 $\Sigma=\{a,b,c\}$ 上的句子 $abacb$ 的不同子串有 $\varepsilon,a,b,c,ab,ba,ac,cb,aba,bac,acb,abac,bacb,abacb$。

根据这两个例子，任给一个字母表上的句子 x，读者都能够找出它的所有不同子串来。

定义 1-12　设 Σ 是一个字母表，$t,u,v,w,x,y,z\in\Sigma^*$，且 $t=uyv$，$w=xyz$，则称 y 是 t 和 w 的**公共子串**(common substring)。如果 $y_1,y_2,\cdots,y_n$ 是 t 和 w 的公共子串，且 $\max\{|y_1|,|y_2|,\cdots,|y_n|\}=|y_j|$，则称 y_j 是 t 和 w 的**最大公共子串**。

显然，两个串的最大公共子串并不一定是唯一的。

定义 1-13　设 Σ 是一个字母表，$\forall L\subseteq\Sigma^*$，$L$ 称为字母表 Σ 上的一个**语言**(language)；$\forall x\in L$，x 叫作 L 的一个句子。

例如，$\{00,11\},\{0,1,00,11\},\{0,1,00,11,01,10\},\{0,1\},\{00,11\}^*,\{01,10\}^*,\{00,01,10,11\}^*,\{0,1\}^*$ 都是字母表$\{0,1\}$上的不同语言。

由于 Σ 的非空有穷性和 Σ 中字符的可区分性，从集合的基数角度分，Σ 上的语言 L 可分为两类：它可以是含有有穷个句子的，也可以是含有可数无穷个句子的。当含有有穷个句子时，称 L 为有穷语言；当含有可数无穷个句子时，称 L 为无穷语言。

由此定义可以知道，一个字母表上的语言就是这个字母表上的一些句子的集合，这些句子都满足一个给定的条件。如果能用便于计算机处理的适当形式给出这些条件的有穷描述，则对语言的处理是非常有用的。就目前的计算机技术来讲，这个“适当形式”应该是形式化的。

定义 1-14　设 Σ_1,Σ_2 是字母表，$L_1\subseteq\Sigma_1^*$，$L_2\subseteq\Sigma_2^*$，语言 L_1 与 L_2 的**乘积**(product)是一

个语言：

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

该语言是字母表 $\Sigma_1\cup\Sigma_2$ 上的语言。

例 1-7 令 $\Sigma=\{0,1\}$，下面是 Σ 上的语言的例子：

(1) $L_1=\{0,1\}$。

(2) $L_2=\{00,01,10,11\}$。

(3) $L_3=\{0,1,00,01,10,11,000,\cdots\}=\Sigma^+$。

(4) $L_4=\{\varepsilon,0,1,00,01,10,11,000,\cdots\}=\Sigma^*$。

(5) $L_5=\{0^n \mid n\geqslant 1\}$。

(6) $L_6=\{0^n1^n \mid n\geqslant 1\}$。

(7) $L_7=\{1^n \mid n\geqslant 1\}$。

(8) $L_8=\{0^n1^m \mid n,m\geqslant 1\}$。

(9) $L_9=\{0^n1^n0^n \mid n\geqslant 1\}$。

(10) $L_{10}=\{0^n1^m0^k \mid n,m,k\geqslant 1\}$。

(11) $L_{11}=\{x \mid x\in\Sigma^+$，且 x 中 0 和 1 的个数相同$\}$。

(12) $L_{12}=\{0^n1^m0^k \mid n,m,k\geqslant 0\}$。

为了了解一个语言的结构，下面来看上述几个语言的部分特点及相互关系。

上述所有语言都是 L_4 的子集(子语言)。

L_1 和 L_2 是有穷语言，其他为无穷语言。L_1 是 Σ 上的所有长度为 1 的句子组成的语言，L_2 是 Σ 上的所有长度为 2 的句子组成的语言。

L_3 和 L_4 分别是 Σ 的正闭包和克林闭包。

$L_5L_7\neq L_6$，$L_5L_7=L_8$；同样 $L_9\neq L_{10}$；但是有 $L_6\subset L_5L_7$，$L_9\subset L_{10}$。

L_6 的句子中 0 和 1 的个数是相同的，并且所有的 0 在所有的 1 的前面。L_{11} 句子中虽然保持着 0 的个数和 1 的个数相等，但它并没要求所有的 0 在所有的 1 的前面。例如，0101，$1100\in L_{11}$，但是 $0101,1100\notin L_6$。而对 $\forall x\in L_6$，有 $x\in L_{11}$，所以，$L_6\subset L_{11}$。

$L_1\subset L_{12}$，$L_2\subset L_{12}$，$L_5\subset L_{12}$，$L_6\subset L_{12}$，$L_7\subset L_{12}$，$L_8\subset L_{12}$，$L_9\subset L_{12}$，$L_{10}\subset L_{12}$

但是

$L_1\not\subset L_{10}$，$L_2\not\subset L_{10}$，$L_5\not\subset L_{10}$，$L_6\not\subset L_{10}$，$L_7\not\subset L_{10}$，$L_8\not\subset L_{10}$，$L_{12}\not\subset L_{10}$

可见，条件的微小差别，会导致语言的很大差别。

实际上，后面会看到，按照乔姆斯基的分类方法，该例中的 L_1，L_2，L_3，L_4，L_5，L_7，L_8，L_{10}，L_{12} 为正则语言，L_6 和 L_{11} 为上下文无关语言，而 L_9 为上下文有关语言。

例 1-8 对于任一字母表 Σ，下列语言具有不同的结构：

(1) $\{x \mid x=x^{\mathrm{T}},x\in\Sigma\}$。

(2) $\{xx^{\mathrm{T}} \mid x\in\Sigma^+\}$。

(3) $\{xx^{\mathrm{T}} \mid x\in\Sigma^*\}$。

(4) $\{xwx^{\mathrm{T}} \mid x,w\in\Sigma^+\}$。

(5) $\{xx^{\mathrm{T}}w \mid x,w\in\Sigma^+\}$。

定义 1-15 设 Σ 是一个字母表，$\forall L\subseteq\Sigma^*$，$L$ 的 n 次**幂**是一个语言：

(1) 当 $n=0$ 时，$L^n=\{\varepsilon\}$。

(2) 当 $n \geqslant 1$ 时，$L^n = L^{n-1}L$。

L 的**正闭包** L^+ 是一个语言：

$$L^+ = L \cup L^2 \cup L^3 \cup L^4 \cup \cdots$$

L 的**克林闭包** L^* 是一个语言：

$$L^* = L^0 \cup L \cup L^2 \cup L^3 \cup L^4 \cup \cdots$$

1.2 文　　法

文法的概念最早是由语言学家在研究自然语言的理解时完成形式化的。那时，他们不但关心如何准确地确定哪些是、哪些不是给定语言的句子，而且还关心如何提供句子结构特征的描述。因为，对一种自然语言，如果能够找到一个形式文法，则对这个语言的自动理解将是非常有用的。例如，可以用来帮助实现语言的机器翻译、文章摘要的提取、文稿的校对与更正等。

下面考虑如何表示高级程序设计语言中的只含加(+)、乘(*)、幂(↑)3 种运算的简单算术表达式的描述。这些表达式构成一个无穷集合，根据上一章的定义，这个集合就是一个语言，可以使用简单的术语给出其如下递归定义：

＜算术表达式＞是＜项＞，＜算术表达式＞加上＜项＞还是＜算术表达式＞；

＜项＞是＜因子＞，＜项＞乘上＜因子＞还是＜项＞；

＜因子＞是＜初等量＞，＜因子＞的＜初等量＞次幂还是＜因子＞；

＜初等量＞是标识符，＜初等量＞是加括号的＜算术表达式＞。

简单起见，可以认为标识符就是变量名、常数。为了使描述更清晰，更符合计算机处理，引入适当的符号表示这些对象：用 E(expression)表示＜算术表达式＞，用 T(term)表示＜项＞，用 F(factor)表示＜因子＞，用 P(primary)表示＜初等量＞，用 id 表示标识符。这样，就可以得到如下“式子”：

$E \rightarrow T$

$E \rightarrow E+T$

$T \rightarrow F$

$T \rightarrow T*F$

$F \rightarrow P$

$F \rightarrow F \uparrow P$

$P \rightarrow (E)$

$P \rightarrow \mathrm{id}$

读者很容易将减(−)和除(/)加进去。根据这些表达＜算术表达式＞的组成规则的“式子”，不难得到一系列算术表达式。例如，得到 $a+b \uparrow c$ 的过程如下：

根据 $E \rightarrow E+T$，得到 $E+T$；根据 $E \rightarrow T$，可以将 $E+T$ 中的 E 变成 T(请读者考虑为什么)得到 $T+T$；根据 $T \rightarrow F$，将 $T+T$ 中的第一个 T 变成 F 得到 $F+T$；根据 $F \rightarrow P$，将 $F+T$ 中的 F 变成 P，得到 $P+T$；根据 $P \rightarrow \mathrm{id}$，将 $P+T$ 中的 P 变成第一个具体的标识符 a，得到 $a+T$；如此下去，根据这些“式子”，最终可以得到 $a+b \uparrow c$ 这个句子。这个过程可以用图 1-1 表示。