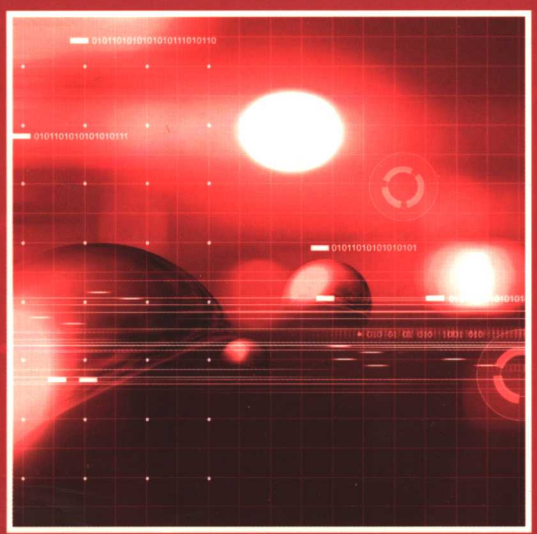
普通高等教育“十一五”计算机类规划教材

形式语言与自动机理论

吴哲辉　吴振寰　编著



机械工业出版社
CHINA MACHINE PRESS

普通高等教育“十一五”计算机类规划教材

形式语言与自动机理论

吴哲辉　吴振寰　编著

机 械 工 业 出 版 社

形式语言与自动机理论是计算机科学理论的重要基础。本书主要介绍乔姆斯基文法体系的四类文法以及它们与有限自动机、下推自动机、线性界限自动机和图灵机之间的关系。此外，对语言的各种运算和封闭性质、判定问题及不可判定性以及确定的上下文无关语言与 LR-文法也进行了讨论。书中还介绍了一些文法和自动机在文本编辑、编译程序、标注语言以及逻辑电路和时序电路设计中的应用。

全书共分 8 章：第 1 章介绍语言及其表示；第 2 章介绍正规表达式、正规文法与有限自动机；第 3 章介绍上下文无关文法与下推自动机；第 4 章介绍图灵机；第 5 章介绍乔姆斯基文法体系；第 6 章介绍语言的运算与封闭性质；第 7 章介绍判定问题与不可判定性；第 8 章介绍确定的上下文无关语言和 LR-文法。

本书可作为高等学校计算机及相关专业研究生及高年级本科生课程教材，也可供从事计算机研究和开发的技术人员参考。为方便教师教学，本书配有教学课件，欢迎选用本书作为教材的老师索取，索取邮箱：llm7785@sina.com。

图书在版编目(CIP)数据

形式语言与自动机理论/吴哲辉，吴振寰编著. —北京：机械工业出版社，2007.4

普通高等教育“十一五”计算机类规划教材

ISBN 978-7-111-20998-0

Ⅰ.形… Ⅱ.①吴…②吴… Ⅲ.①形式语言—高等学校—教材②自动机理论—高等学校—教材 Ⅳ.TP301

中国版本图书馆 CIP 数据核字(2007)第 025780 号

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)

责任编辑：刘丽敏 版式设计：冉晓华 责任校对：陈立辉

封面设计：张 静 责任印制：洪汉军

北京汇林印务有限公司印刷

2007 年 4 月第 1 版第 1 次印刷

184mm×260mm · 11.75 印张 · 287 千字

标准书号：ISBN 978-7-111-20998-0

定价：20.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

销售服务热线电话：(010)68326294

购书热线电话：(010)88379639、88379641、88379643

编辑热线电话：(010)88379726

封面无防伪标均为盗版

前　言

形式语言与自动机理论是研究字符串集合及其性质的一门学科，而字符串是电子计算机最基本的处理对象。

自然语言是人与人之间进行交流的基本手段。计算机(程序设计)语言则是专门为人与计算机之间进行交流而设计的一种人工语言。形式语言着眼于自然语言和人工语言都必须遵循的一般规律的研究。自动机作为语言识别器，同形式语言有着密切的关系。图灵机是最高形式的一种自动机，它不仅是现代电子计算机的理论模型，也是可计算性判定和计算机复杂性度量的基准。因此，美国 ACM 和 IEEE/CS 制定的“Computing Curricula 2001”(简称 CC2001)以及中国计算机学会教育专业委员会和全国高等学校计算机教育委员会制定的“中国计算机科学与技术学科教程 2002”(简称 CCC2002)都把形式语言与自动机理论列入了基本内容和课程体系中。

本书的主要内容包括 Chomsky 文法体系中的四大类文法(正规文法、上下文无关文法、上下文有关文法和无限制文法)，它们所产生的语言，以及识别这四类语言的自动机(有限自动机、下推自动机、线性界限自动机和图灵机)，讨论了它们之间的关系。同时，本书也对语言的运算和封闭性质、语言的判定问题与不可判定性，进行了专门讨论。对同编译原理密切相关的 LR-文法及其对应的确定的上下文无关语言作了简介。书中对正规表达式、有限自动机和上下文无关文法等在编译程序、文本编辑、文本搜索、逻辑电路设计、超文本标记语言(HTML)、可扩展标记语言(XML)等方面的应用进行了介绍。我们希望通过这些内容向读者展示，形式语言与自动机理论不仅是一门较为抽象的理论课程，同时也是有着广泛应用的一门学科。

本书的大部分内容(包括许多例子)都源自参考文献[1](那是世界公认的关于形式语言、自动机和计算理论的经典著作)，也有部分内容引自书后所列的其他参考文献。此外，书中也写入了一些作者在多年从事本门课程教学(包括本科生和研究生的教学)中所积累的一些心得体会。

本书可以作为计算机专业本科生和硕士研究生的教材。作为本科生教材，主要讲授第 1～3 章以及第 4 章的前 6 节，用 40～48 学时讲授。研究生教学则可以覆盖全书的内容，大致需要 48～60 学时。各章后面列出的习题都是比较基本的，难度较大的一些习题可参阅参考文献[1]。

书中第 2、3 章由吴振寰撰写，吴哲辉写了其余各章并负责对全书进行校审。王丽丽和李莹莹两位同学为本书的打印和排版付出了艰辛的劳动，在此表示衷心的感谢。

由于水平所限，错误和不当之处在所难免，敬请读者批评指正。

编　者

吴哲辉

1941年生于广东省连州市，1965年毕业于中山大学数学力学系。1981年到1983年在美国芝加哥伊利诺大学作访问学者，学习和研究计算机科学理论。现任山东科技大学教授，博士生导师。

主要的研究方向有Petri网理论，形式语言与自动机理论，算法设计与分析等。主持承担过（包括在研）7项国家自然科学基金项目的研究，发表学术论文100多篇，获得过省部级科技奖4项。从事形式语言与自动机理论课程（包括本科和研究生课程）教学近20年。

吴振寰

1977年生于山东省济南市，1999年毕业于山东科技大学计算机软件专业，获学士学位，2004年获硕士学位。现任教于山东科技大学，讲师，从事Petri网理论与应用，形式语言与自动机，信息安全等方面的教学和科研工作。

目　录

第1章　语言及其表示

这里所说的语言是一个抽象概念。概括地说，一个语言就是某个字母表上满足某些特定条件的字符串的集合。这个抽象概念既适合于自然语言，也适合于人工语言。譬如，英语的字母表就是26个英文字母，再加上若干个标点符号。英语中的一个句子就是这个字母表上一个满足某些特定条件的字符串，这些特定条件是指单词的组成规则和句子的生成规则(语法)。这是自然语言的一个例子。最常见的人工语言是计算机程序设计语言，特别是高级语言。每种程序设计语言都会规定一批合法的字符，它们组成该语言的字母表。此外，每种程序设计语言都规定好各种语句(如赋值语句、分支语句、循环语句等)的格式。一个语句就是字母表上的一个字符串，各种语句格式的规定就是字符串要满足的特定条件。

语言是用于信息交流的，或者是人与人之间的交流，或者是人与计算机之间的交流。为了能很好地运用语言进行交流，使用者必须了解所使用的语言中什么样的句子是合法句子。这样，就需要有一个产生合法句子的规则或是识别一个字符串是否是合法句子的装置，前者称为文法，后者称为语言识别器。文法和语言识别器是基本的两种语言表示方式。

1.1　字母表、串和语言

1.1.1　字母表

由字符组成的一个非空的有限集称为一个字母表(alphabet)，通常用 Σ 表示。

字母表是讨论字符串和语言的出发点。当我们讨论一个字符串的时候，总是假设它是某个字母表上的字符组成的串。当我们讨论一个语言的时候，总是假设它是某个字母表上的某些字符串的集合。

例如，$\Sigma=\{0,1\}$ 是一个字母表。一个二进制数就是这个字母表上的一个字符串。$\Sigma=\{a,b,c,\cdots,z\}$ 是一个字母表，它由26个英文小写字母组成。所有的ASCII字符的集合也是一个字母表。

字母表中至少要有一个字符，而且字母表中的字符个数必须是有限的。空集不能作为字母表，无限集也不能作为字母表。

1.1.2　串

串(string)是指某个字母表上的字符组成的有限序列。例如，0100110是字母表 $\Sigma=\{0,1\}$ 上的一个串，0000也是这个字母表上的一个串。

一个串中字符的位数称为这个串的长度。例如串0100110的长度等于7，串0000的长度等于4。若用x表示一个串(如 $x=0100110$)，那么 $|x|$ 表示串x的长度(如 $|x|=|0100110|=7$)。

长度等于0的串称为空串，用 ε 表示。换句话说，空串是不含任何字符的串。显然，它可以看作是任一个字母表上的串。

定义 1.1 设 x 为一个串，若将 x 中的字符按逆向顺序重新排列，所得到的串称为 x 的逆(reversal)，记为 x^R。

例如，若 x = abacd，则 x^R = dcaba。

显然，$|x^R| = |x|$。空串的逆还是空串，即 $\varepsilon^R = \varepsilon$。

定义 1.2 设 x 和 y 是两个串，xy 称为 x 和 y 的连接(concatenation)。用符号“∘”表示两个串的连接运算，即 x∘y = xy。

例如，若 x = aba，y = cb，则 x∘y = xy = abacb。

定义 1.3 设 x 和 y 都是串，z = xy。称 x 为 z 的前缀(prefix)；当 $x \neq z$(即 $y \neq \varepsilon$)时，称 x 为 z 的真前缀。称 y 为 z 的后缀(suffix)；当 $y \neq z$(即 $x \neq \varepsilon$)时，称 y 为 z 的真后缀。

例如，若 z = abacb，那么 z 的前缀包括ε，a，ab，aba，abac 和 abacb 6 个串，其中前 5 个串是 z 的真前缀；z 的后缀包括ε，b，cb，acb，bacb 和 abacb 6 个串，其中前 5 个串是 z 的真后缀。

定义 1.4 设 x，y，z 都是串，w = xyz。则称 y 为 w 的子串(substring)，当 $y \neq w$ 时，称 y 为 w 的真子串。

显然，一个串 w 的前缀和后缀都是它的子串。此外，w 可能还有一些子串既不是 w 的前缀，也不是 w 的后缀。例如，设 w = abacb，那么除了前面列出的 w 的前缀和后缀以外，w 的子串还有 ba，bac，ac 等。

设 Σ 为一个字母表，通常用 Σ^* 表示 Σ 上的全体串(包括空串)组成的集合，而 Σ^+ 表示 Σ 上除空串外的一切串的集合，即 $\Sigma^+ = \Sigma^* - \{\varepsilon\}$。

下面我们研究一下 Σ^* 上的串的连接运算的一些性质。

首先，连接运算在 Σ^* 上是封闭的，即 $\forall x, y \in \Sigma^*$，都有 $x \circ y \in \Sigma^*$。可见(Σ^*，∘)构成一个代数系统。

其次，连接运算满足结合律，即 $\forall x, y, z \in \Sigma^*$：$(x \circ y) \circ z = x \circ (y \circ z)$。这样($\Sigma^*$，∘)就构成一个半群。

此外，由于对任意 $x \in \Sigma^*$，都有 $\varepsilon \circ x = x \circ \varepsilon = x$，这表明空串ε是 Σ^* 中关于连接运算的单位元。从而我们有如下定理。

定理 1.1 设 Σ 为一个字母表，“∘”表示 Σ^* 上的串的连接运算。那么(Σ^*，∘)构成一个单元半群(独异点)。

(Σ^*，∘)不能构成一个群。这里因为对于一个串 $x \in \Sigma^*$，(当 $x \neq \varepsilon$时)不存在 x 关于连接运算“∘”的逆元 y，使得 $x \circ y = y \circ x = \varepsilon$。注意定义 1.1 关于一个串 x 的逆串 x^R 同这里所说的 x 关于“∘”的逆元是完全不同的概念。

此外，Σ^* 上的连接运算也不满足交换律。一般地 $x \circ y \neq y \circ x$。

定理 1.2 设 Σ 为一个字母表，$x, y, z \in \Sigma^*$，那么

1) $x \circ y = x \circ z$ 当且仅当 $y = z$；

2) $x \circ z = y \circ z$ 当且仅当 $x = y$。

(结论是显然的)

定义 1.5 设 Σ 为一个字母表，$x \in \Sigma^*$。x 的幂运算定义为

1) $x^0 = \varepsilon$；

2) $x^n = x^{n-1} \circ x$，$n = 1, 2, \cdots$。

当$|x|=1$时，就得到单个字符的幂：$a^0=\varepsilon, a^1=a, a^2=aa, a^3=aaa, \cdots$。

定理 1.3 设Σ为一个字母表，m，n为任意两个非负整数。那么，对任意$x\in\Sigma^*$都有

1) $x^m\circ x^n=x^n\circ x^m=x^{m+n}$； (1.1)

2) $(x^m)^n=x^{mn}$。 (1.2)

(从定义1.5出发，用数学归纳法容易证明式(1.1)和式(1.2)，具体证明留作习题。)

定理 1.4 设Σ_1和Σ_2是两个字母表。那么，代数系统$(\Sigma_1^*,\circ)$和$(\Sigma_2^*,\circ)$同构的充分必要条件是$|\Sigma_1|=|\Sigma_2|$。

证明 充分性：若$|\Sigma_1|=|\Sigma_2|$，那么可以在Σ_1和Σ_2之间定义一个一一对应映射$f:\Sigma_1\to\Sigma_2$。把f拓展为$f':\Sigma_1^*\to\Sigma_2^*$，使得

1) $f'(\varepsilon)=\varepsilon$；

2) $\forall x=a_1a_2\cdots a_n\in\Sigma_1^*$，$f'(x)=f(a_1)f(a_2)\cdots f(a_n)$

那么f'是Σ_1^*到Σ_2^*的一一对应映射，而且易知对$\forall x_1, x_2\in\Sigma_1^*$：

$$f'(x_1\circ x_2)=f'(x_1)\circ f'(x_2)$$

这表明f'是Σ_1^*到Σ_2^*的一个同构映射，从而$(\Sigma_1^*,\circ)$和$(\Sigma_2^*,\circ)$同构。

必要性：若$|\Sigma_1|\neq|\Sigma_2|$，我们证明不存在一一对应映射$f':\Sigma_1^*\to\Sigma_2^*$使得$f'(x_1\circ x_2)=f'(x_1)\circ f'(x_2)(x_1,x_2\in\Sigma^*)$。下面用反证法。

不妨假设$|\Sigma_1|>|\Sigma_2|$，那么至少存在一个字符$a\in\Sigma_1$，使得$|f'(a)|>1$。记$f'(a)=b_1b_2\cdots b_r$，其中$b_i\in\Sigma_2$，$r>1$。设$f'(x_i)=b_i(x_i\in\Sigma_1^*, |x_i|\geqslant 1)$，那么就有

$$f'(a)=f'(x_1)f'(x_2)\cdots f'(x_r)=f'(x_1x_2\cdots x_r) \quad (1.3)$$

由于f'是一一对应映射，从式(1.3)就得到$a=x_1x_2\cdots x_r$。然而这是不可能的，因为$|a|=1$，而$|x_1x_2\cdots x_r|\geqslant r>1$。从而导致矛盾。

1.1.3 语言

定义 1.6 字母表Σ上满足一定条件的串的集合L称为Σ上的一个语言(language)。

若L是Σ上的一个语言，则显然有$L\subseteq\Sigma^*$。对任意一个字母表Σ，Σ上最简单的两个语言分别为ϕ和$\{\varepsilon\}$，其中ϕ(空集)中不含有任何元素，$\{\varepsilon\}$则是只含空串的集合，它不是空集，因为它含有一个元素ε。请注意两者的区别。此外，Σ^*本身也是Σ上的一个语言。当我们以Σ上的字符串作为论域时，Σ^*是一个全集。

例 1.1 全体二进制非负整数的集合L_1是字母表$\Sigma_1=\{0,1\}$上的一个语言。如果从严格意义上理解二进制非负整数，即要求每个二进制正整数的首位必须是1(通常所说的有效数)，那么这个语言可以表示为

$$L_1=\{x\in\{0,1\}^*\mid x=0\text{ 或 }x\text{ 的首位为 }1\}$$

另一个方面，我们也可以从广义上来定义二进制数，即把每个0-1串都看作一个非负二进制整数(它的值等于删去左边第1个“1”之前的全部“0”以后得到的有效数之值)。那么全体非负二进制整数构成的语言就是$L_2=\{0,1\}^*$。在本书后面的讨论中，我们就把L_2理解为全体二进制非负整数的集合。

例 1.2 $L_3=\{x\in\{0,1\}^*\mid |x|\leqslant 3\}$也是字母表$\Sigma_1=\{0,1\}$上的一个语言。这个语言共有15个元素，即

$$L_3 = \{\varepsilon,0,1,00,01,10,11,000,001,010,011,100,101,110,111\}$$

在这个例子中，我们对 L_3 给出了两种表示法。前一种是解析表示法，通过表达式来给出 L_3 中的元素应满足的条件；后一种表示法则列出了 L_3 的全体元素。这两种都是常用的集合表示，因为语言归根结底也是一个集合(只不过这种集合中的元素是字符串)。当然，只有当一个语言是一个有限集(而且元素的个数不太多)时，后一种表示法才能有效地使用。否则就要加入省略号才能实现。

例 1.3 $L_4 = \{a^n b^n \mid n \geqslant 1\}$ 是字母表 $\Sigma = \{a,b\}$ 上的一个语言。它显然是一个无限集，因为对 n 每取一个正整数，L_4 都有一个元素与之对应，而且 n 的取值不同，L_4 中对应的元素也不相同。如果要用罗列语言中的元素的方式来给出 L_4，那么只能加入省略号，写成

$$L_4 = \{ab,aabb,aaabbb,\cdots\}$$

例 1.4 $L_5 = \{x \in \{a,b\}^* \mid x = x^R\}$ 也是字母表 $\Sigma_2 = \{a,b\}$ 上的一个语言。这个语言中的串都具有这样的性质：$x = x^R$。也就是说，若把 x 中的字符按逆向顺序重新排列，所得到的串仍等于 x。我们称 L_5 为字母表 $\{a,b\}$ 上的回文(palindrome)集。易知

$$L_5 = \{\varepsilon,a,b,aa,bb,aaa,aba,bab,bbb,aaaa,abba,\cdots\}$$

定义 1.7 设 L_1 为字母表 Σ_1 上的一个语言，L_2 是字母表 Σ_2 上的一个语言，那么

$$L_1 \circ L_2 = \{x \circ y \mid x \in L_1 \wedge y \in L_2\} \tag{1.4}$$

称为语言 L_1 和 L_2 的连接。显然它是字母表 $\Sigma_1 \cup \Sigma_2$ 上的一个语言。语言的连接运算也称为乘法运算。

定理 1.5 设 $L_i(i=1,2,3)$ 是字母表 Σ_i 上的语言，那么

$$L_1 \circ (L_2 \cup L_3) = (L_1 \circ L_2) \cup (L_1 \circ L_3) \tag{1.5}$$

$$(L_2 \cup L_3) \circ L_1 = (L_2 \circ L_1) \cup (L_3 \circ L_1) \tag{1.6}$$

其中 式(1.5)和式(1.6)中的“∪”是指集合的并运算。

(证明留作习题)

定义 1.8 设 L 为字母表 Σ 上的一个语言，记

$$L^0 = \{\varepsilon\}$$

$$L^n = L^{n-1} \circ L,\ n = 1,2,\cdots$$

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

称 L^* 为语言 L 的 Kleene 闭包(或简称为闭包)，L^+ 为 L 的正闭包。

定理 1.6 设 L 是字母表 Σ 上的一个语言，则

$$L^+ = L^* \circ L = L \circ L^* \tag{1.7}$$

$$L^* = L^+ \cup \{\varepsilon\} \tag{1.8}$$

证明 式(1.8)可以直接由定义 1.7 得到，下面证明式(1.7)。

$$L^* \circ L = \left(\bigcup_{i=0}^{\infty} L^i\right) \circ L = \bigcup_{i=0}^{\infty} L^i \circ L = \bigcup_{i=0}^{\infty} L^{i+1}$$

$$= \bigcup_{i=1}^{\infty} L^i = L^+$$

同理可以证明 $L \circ L^* = L^+$。

1.2 文法

文法(grammar)是语言的产生器。语言学家 N. Chomsky 最初从语言产生的角度来研究语言。1956 年，他将语言形式地定义为一个字母表上满足特定条件的字符串组成的集合。即对于任意一个语言 L，都有一个字母表 Σ，使得 $L \subseteq \Sigma^*$。他提出可以在一个字母表上给出一组语言产生规则，根据这些规则推导出来的全体句子组成的集合就构成一个语言。这样的一组规则就称为一个文法。后来，他又根据对规则的不同限制，对文法及其产生的语言进行分类，形成了 Chomsky 文法体系。

定义 1.9 文法是一个四元组 $G=(V,T;P,S)$，其中 V 是变量的一个有限集(也称为非终极符集)，T 是终极符的一个有限集(也称为字母表)，$V \cap T = \phi$；S 是初始符($S \in V$)；P 是产生式集合，P 中的元素为(α,β)(通常表示为 $\alpha \to \beta$)，$\alpha, \beta \in (V \cup T)^*$，而且 α 中至少有一个字符是 V 的元素。

产生式集合 P 是一个文法的核心，它是形式化规则的一个有限集。由于每个产生式 $\alpha \to \beta(\alpha,\beta \in (V \cup T)^*)$的左边的串 α 中至少要含有变量集 V 中的一个字符，所以也可以把产生式集表示为

$$P \subseteq (V \cup T)^* \circ V \circ (V \cup T)^* \times (V \cup T)^*$$

通常地，我们用 A，B，C，D，S 等大写英文字母表示变量(即变量集 V 中的元素)；用 a，b，c，d 等小写字母表示终极符(终极符集 T 中的元素)；而由终极符组成的串(终极符串)则用 w，x，y，z 等表示；由变量和终极符组成的串则用 α，β，γ 等希腊字母表示。

从文法 G 产生出语言 L(G)的过程是这样的：从初始符开始，根据 P 中的一个产生式可以进行一步推导，如果通过若干步推导得到一个只由终极符组成的串，这个串就是 L(G)中的一个句子。所有能通过推导得到的句子的集合就是语言 L(G)。推导过程中出现的每一个串都称为 G 的一个句型。下面给出相关的形式定义。

定义 1.10 设 $G=(V,T;P,S)$为一个文法，那么

1) S 是 G 的一个句型；

2) 若 $\delta\alpha\gamma$ 是 G 的一个句型，$\alpha \to \beta$ 是 G 的一个产生式，那么 $\delta\beta\gamma$ 也是 G 的一个句型。从句型 $\delta\alpha\gamma$ 得到句型 $\delta\beta\gamma$ 的过程称为文法 G 的一步推导，记为

$$\delta\alpha\gamma \underset{G}{\Longrightarrow} \delta\beta\gamma \quad 或 \quad \delta\alpha\gamma \Longrightarrow \delta\beta\gamma$$

(在不会产生混淆的情况下，推导符$\underset{G}{\Longrightarrow}$下方的字符 G 可以省略。)

3) 不含非终极符的句型是 L(G)的一个句子。

若 $\gamma_i \in (V \cup T)^* (i=1,2,\cdots,n)$，$\gamma_i \underset{G}{\Longrightarrow} \gamma_{i+1} (i=1,2,\cdots,n-1)$是 G 的一步推导，即 $\gamma_1 \underset{G}{\Longrightarrow} \gamma_2$，$\gamma_2 \underset{G}{\Longrightarrow} \gamma_3$，$\cdots$，$\gamma_{n-1} \underset{G}{\Longrightarrow} \gamma_n$，那么可以把 n-1 步的推导过程记为 $\gamma_1 \overset{*}{\underset{G}{\Longrightarrow}} \gamma_n$。特别地，当 n=1 时，$\gamma_1 \overset{*}{\underset{G}{\Longrightarrow}} \gamma_n$ 是零步推导。

定义 1.11 设 $G=(V,T;P,S)$为一个文法，那么 G 所产生的语言为

$$L(G)=\{w\in T^{*}\mid S\overset{+}{\underset{G}{\Rightarrow}}w\} \tag{1.9}$$

下面我们通过例子来说明从文法产生语言的全过程。

例 1.5 设有一个文法 $G_1=(V_1,T_1;P_1,S)$，其中 $V_1=\{S,A\}$，$T_1=\{0,1\}$，

P_1：

① $S\rightarrow 0S$

② $S\rightarrow 0A$

③ $A\rightarrow 1A$

④ $A\rightarrow \varepsilon$

这个文法共有两个变量(S 和 A)，两个终极符(0 和 1)和 4 个产生式。为叙述方便，我们给各个产生式编了序号。对文法 G_1，容易得到下面的一些推导过程：

$$S\overset{①}{\Rightarrow}0S\overset{①}{\Rightarrow}00S\overset{②}{\Rightarrow}000A\overset{③}{\Rightarrow}0001A\overset{④}{\Rightarrow}0001$$

$$S\overset{②}{\Rightarrow}0A\overset{③}{\Rightarrow}01A\overset{④}{\Rightarrow}01$$

$$S\overset{②}{\Rightarrow}0A\overset{④}{\Rightarrow}0$$

$$S\overset{①}{\Rightarrow}0S\overset{②}{\Rightarrow}00A\overset{③}{\Rightarrow}001A\overset{③}{\Rightarrow}0011A\overset{③}{\Rightarrow}00111A\overset{④}{\Rightarrow}00111$$

其中 $\beta\overset{ⓘ}{\Rightarrow}\gamma$ 表示从句型 β 推导出句型 γ 的这一步推导中，所使用的是第ⓘ个产生式。从这 4 个推导过程，我们求出了 $L(G_1)$ 中的 4 个元素：0001，01，0，00111。然而，我们不能指望通过推导求出 $L(G_1)$ 中的全部元素，因为 $L(G_1)$ 中的元素有无限多个。因此，在进行了一定数量的推导后，要善于进行归纳，找出 $L(G_1)$ 中的元素具有的共同性质，而且这些性质是 $L(G_1)$ 中的元素所特有的(即 $L(G_1)$ 以外的元素都不具有这种性质)。由此得到 $L(G_1)$ 的一般表达式。

譬如，通过上面的推导，我们归纳出 G_1 所产生的语言为

$$L(G_1)=\{a^{i}b^{j}\mid i\geqslant 1,j\geqslant 0\} \tag{1.10}$$

剩下的工作是证明我们的归纳结论的正确性。为此，需要从两个方面加以证明。

1) 凡是根据文法 G_1 推导出的终极符串都是式(1.10)中的串。

为便于叙述我们用 $\beta\overset{k\times ⓘ}{\Rightarrow}\gamma$ 表示从句型 β 连续 k 次使用第ⓘ个产生式可推导出句型 γ，那么 $L(G_1)$ 中每个元素的推导过程都具有下面的形式：

$$S\overset{k\times ①}{\Rightarrow}0^{k}S\overset{②}{\Rightarrow}0^{k+1}A\overset{j\times ③}{\Rightarrow}0^{k+1}1^{j}A\overset{④}{\Rightarrow}0^{k+1}1^{j} \tag{1.11}$$

其中 k，j 可以取 0,1,2,…等任意一个非负整数。因此 $L(G_1)$ 中的每个元素都具有

$$0^{k+1}1^{j},\ j,\ k\geqslant 0$$

的形式，或写成

$$0^{i}1^{j},\ i\geqslant 1,\ j\geqslant 0 \tag{1.12}$$

的形式。

2) 每个具有式(1.12)的形式的串都可以根据文法 G_1 推导出来。

这是显然的，因为只要取 $k=i-1$，那么用式(1.11)的推导，就可以推导出式(1.12)。

根据 1)、2)，就证明了式(1.10)是正确的。

例 1.6 设有一个文法 $G_2=(V_2,T_2;P_2,S)$，其中 $V_2=\{S,A,B\}$，$T_2=\{a,b\}$，产生式集为

P_2：　　$S\rightarrow aB$　　$A\rightarrow bAA$

$S\to bA$　　$B\to b$

$A\to a$　　$B\to bS$

$A\to aS$　　$B\to aBB$

这个文法共有3个变量S，A，B，其中S是初始符，有两个终极符a和b，有8个产生式。为了推导过程中便于查找产生式，通常把左端相同的产生式合并在一起，用竖杠“|”把不同的产生式右端分隔开。譬如，文法 G_2 的产生式集又可表示成

P_2：

$$S\to aB\mid bA$$
$$A\to a\mid aS\mid bAA$$
$$B\to b\mid bS\mid aBB$$

根据文法 G_2，我们可以得到下面一些推导：

$$S\Longrightarrow aB\Longrightarrow ab$$
$$S\Longrightarrow bA\Longrightarrow ba$$
$$S\Longrightarrow aB\Longrightarrow aaBB\Longrightarrow aabB\Longrightarrow aabb$$
$$S\Longrightarrow aB\Longrightarrow abS\Longrightarrow abbA\Longrightarrow abba$$
$$S\Longrightarrow bA\Longrightarrow baS\Longrightarrow baaB\Longrightarrow baabS\Longrightarrow baabaB\Longrightarrow baabab$$

从上面的5个推导过程，我们知道串ab，ba，aabb，abba，baabab都是L(G)中的元素(句子)。

通过上述推导，我们归纳出文法 G_2 产生的语言为

$$L(G_2)=\{w\in\{a,b\}^+\mid w\text{中a的个数等于b的个数}\}\tag{1.13}$$

作为一个完整的过程，还需要证明归纳所得到的结论是正确的。要证明的是：“句子w可以从文法 G_2 通过推导产生当且仅当w中a的个数等于b的个数”。

下面我们对句子w的长度用数学归纳法来证明上述结论。这里采用的是互归纳法。所谓互归纳法，就是在一个归纳法证明过程中同时证明多个命题。虽然其中有些命题并不是我们所需要的结论，但借助它们可以为所需要的结论的归纳证明提供便利。就本问题来说，我们用数学归纳法同时证明下面3个命题。

命题1：　$S\overset{*}{\Longrightarrow}w$　当且仅当w中：a的个数=b的个数

命题2：　$A\overset{*}{\Longrightarrow}w$　当且仅当w中：a的个数=b的个数+1

命题3：　$B\overset{*}{\Longrightarrow}w$　当且仅当w中：a的个数+1=b的个数

就问题本身的要求来说，我们所需要的只是命题1的结论，但证明命题1需要借助命题2和命题3，反之命题2和命题3的证明又需要命题1。因此，我们只好把3个命题捆绑在一起，通过一个归纳过程加以证明。下面就给出同时证明这3个命题的归纳证明过程。

基础：证明当$|w|\leqslant 2$时，命题1、命题2和命题3都是为真。

在$\{a,b\}^+$中，满足$|w|\leqslant 2$的串只有6个：a，b，ab，ba，aa，bb。从G的产生式集易知，只有前4个串存在推导

$$A\Longrightarrow a$$
$$B\Longrightarrow b$$
$$S\Longrightarrow aB\Longrightarrow ab$$
$$S\Longrightarrow bA\Longrightarrow ba$$

而且对于上述每一个串，都只有一个推导能产生它。另一个方面，从 G 的任一个变量出发都不存在推导能产生串 aa 或 bb。这些情况表明，当$|w|\leqslant 2$时，3 个命题都为真。

归纳步骤：假设当$|w|\leqslant k(k\geqslant 2)$时，3 个命题都为真，我们证明当$|w|=k+1$时，3 个命题也都为真。

先证必要性，分 3 种情况讨论。

情况 1：$S\overset{*}{\Longrightarrow}w$。则推导过程的第一步只有两种可能：$S\Longrightarrow aB$ 或 $S\Longrightarrow bA$

若第一步推导为 $S\Longrightarrow aB$，记 $w=aw_1$，则有 $|w_1|=k$ 和 $B\overset{*}{\Longrightarrow}w_1$。由归纳假设，$w_1$ 中：a 的个数 +1 = b 的个数。所以 w 中：a 的个数 = b 的个数。

若第一步推导为 $S\Longrightarrow bA$，记 $w=bw_2$，则有 $|w_2|=k$ 和 $A\overset{*}{\Longrightarrow}w_2$。由归纳假设，$w_2$ 中：a 的个数 = b 的个数 +1。所以 w 中：a 的个数 = b 的个数。

情况 2：$A\overset{*}{\Longrightarrow}w$。由于$|w|=k+1\geqslant 3$，推导过程的第一步只有两种可能：$A\Longrightarrow aS$ 或 $A\Longrightarrow bAA$。

若第一步推导为 $A\Longrightarrow aS$，记 $w=aw_1$，则有 $|w_1|=k$ 和 $S\overset{*}{\Longrightarrow}w_1$。由归纳假设，$w_1$ 中：a 的个数 = b 的个数。所以 w 中：a 的个数 = b 的个数 +1。

若第一步推导为 $A\Longrightarrow bAA$，那么存在 w_1 和 w_2（$|w_1|\geqslant 1$，$|w_2|\geqslant 1$，$|w_1|+|w_2|=k$）使得 $w=bw_1w_2$，$A\overset{*}{\Longrightarrow}w_1$ 和 $A\overset{*}{\Longrightarrow}w_2$。由于$|w_1|\leqslant k$，$|w_1|\leqslant k$，从归纳假设知 w_1 和 w_2 中都有：a 的个数 = b 的个数 +1。从而 $w=bw_1w_2$ 中：a 的个数 = b 的个数 +1。

情况 3：$B\overset{*}{\Longrightarrow}w$。仿照情况 2 的讨论可以证明 w 中：a 的个数 +1 = b 的个数。

下面证明充分性，也分 3 种情况讨论。

情况 1：w 中：a 的个数 = b 的个数。我们考察 w 的首字符。

若 w 的首字符为 a，记 $w=aw_1$，那么 $|w_1|=k$ 且 w_1 中 a 的个数 +1 = b 的个数。由归纳假设知存在一个推导 $B\overset{*}{\Longrightarrow}w_1$。从而就存在一个推导

$$S\Longrightarrow aB\overset{*}{\Longrightarrow}aw_1(=w)$$

若 w 的首字符为 b，记 $w=bw_2$，那么 $|w_2|=k$ 且 w_2 中 a 的个数 = b 的个数 +1。由归纳假设知存在一个推导 $A\overset{*}{\Longrightarrow}w_2$。从而就存在一个推导

$$S\Longrightarrow bA\overset{*}{\Longrightarrow}bw_2(=w)$$

情况 2：w 中：a 的个数 = b 的个数 +1。同样考察 w 的首字符。

若 w 的首字符为 a，记 $w=aw_1$，那么 $|w_1|=k$ 且 w_1 中 a 的个数 = b 的个数。由归纳假设知存在一个推导 $S\overset{*}{\Longrightarrow}w_1$，这样就存在一个推导

$$A\Longrightarrow aS\Longrightarrow aw_1(=w)$$

若 w 的首字符为 b，记 $w=bw_2$，那么 $|w_2|=k$ 且 w_2 中 a 的个数 = b 的个数 +2。把 w_2 分成两段：$w_2=w_{21}w_{22}$使得 $|w_{2i}|\geqslant 1$ 而且 w_{2i}中 a 的个数 = b 的个数 +1。注意 $|w_{2i}|<k(i=1,2)$，从而由归纳假设知存在推导

$$A\overset{*}{\Longrightarrow}w_{21}\text{和}A\overset{*}{\Longrightarrow}w_{22}$$

这样，G_2 中就存在推导

$$A\overset{*}{\Longrightarrow}bAA\overset{*}{\Longrightarrow}bw_{21}A\overset{*}{\Longrightarrow}bw_{21}w_{22}(=w)$$

情况 3：w 中：a 的个数 +1 = b 的个数。仿照情况 2 的证明可证存在一个推导 $B\overset{*}{\Longrightarrow}w$。

通过上述讨论，我们证明了若$|w|\leqslant k$时 3 个命题都成立，那么当$|w|=k+1$时，3 个

命题也都成立。

1.3 语言识别器

对语言提供有限表示的另一种常用方法是语言识别器。20 世纪 50 年代初，语言学家 S. C. Kleene 从识别语言的角度来研究语言，并把这种语言识别器称之为自动机(automaton)。

自动机的结构包括三大部分：有限状态控制器、输入带和辅助存储器(有的自动机也可以没有辅助存储器)。有限状态控制器通过读写头或只读输入头同其他两部分相连接(如图 1.1 所示)。通过这些连接，可以在有限状态器的控制下进行信息交流。

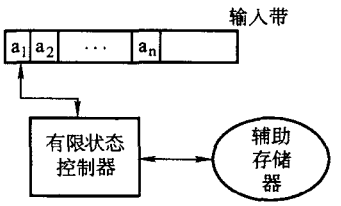


图 1.1 语言识别器的大致结构

输入带上排列着一个个带单元(格)，每个带单元可以存放一个字符。

用自动机识别语言的过程是这样的：对于某个字母表 Σ 上的一个字符串 $x = a_1a_2\cdots a_n$，先把这个串存放在输入带上(从左到右，每格存放一个字符)，通过读写头(或只读输入头)，有限状态控制器从左到右对输入带上的字符串进行扫描。每次扫描一格，然后根据在该格所读到的字符、有限状态控制器的当前状态，以及辅助存储器中的相关信息(如果该自动机带有辅助存储器的话)决定自动机的一个动作。动作的内容通常包含几个方面：有限状态控制器中的状态变化，是否在输入带的当前格写上新的字符(如果连接有限状态控制器和输入带的是只读输入头，则不能写)，读写头(或只读输入头)在带上右移(或左移)一格，以及辅助存储器的信息怎样变更等。当输入带上的整个字符串被扫描完后(对有的自动机来说，不要求恰好扫描完整个字符串)，有限状态控制器进入终止状态，则该字符串被自动机接受。一个自动机所接受的所有字符串的集合(它是 Σ^* 的一个子集)就是该自动机所确定的一个语言。

自动机的状态集是一个有限集，它的输入带的长度可以是有限或者是无限，有限状态控制器可以用读写头同输入带连接，也可以用只读输入头同输入带连接，有的自动机有辅助存储器，有的则没有辅助存储器。不同的配置情况，自动机接受语言的能力(或说范围)也不一样。因此，自动机也可以分为若干种类型。这些将在以后各章展开讨论。

习 题 1

1.1 写出字符串 x = 101101 的全部前缀、后缀和子串。

1.2 设 $L = \{x \in \{a,b\}^* \mid (x = x^R) \wedge (|x| \leqslant 4)\}$，写出 L 的全部元素。

1.3 用数学归纳法证明定理 1.3 中的式(1.1)和式(1.2)。

1.4 以合理的顺序展开下列语言，把它们写成带省略号的枚举表示。

$\{a,b\}^*$　　$\{a\}^*\{b\}^*$　　$\{ab\}^*$

$\{a^nb^{2n} \mid n \geqslant 0\}$

$\{w \in \{0,1\}^* \mid w$ 中 0 的个数和 1 的个数都是偶数$\}$

$\{w \in \{0,1\}^* \mid w$ 中 0 的个数等于 1 的个数$\}$

1.5 设 L 是一个语言，指出下面的式子中，哪个是恒等式(即对任意语言 L 都成立)？哪个是条件等式(当 L 满足某些条件时才成立)？哪个是矛盾式(即该等式对任意语言 L 都不成立)？并说明理由。