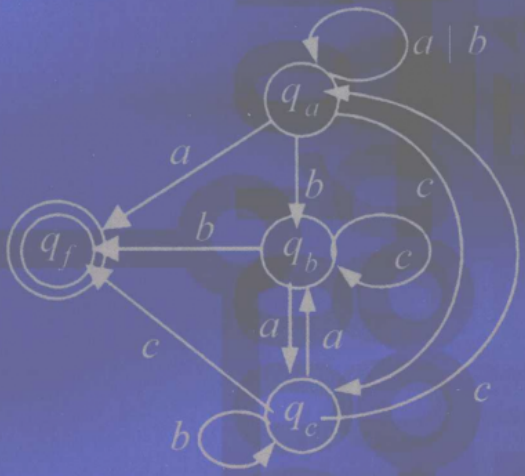
计算机科学的数学基础

周经野 刘任任 编著

- 形式语言及自动机理论
- 可计算理论
- 逻辑学
- 程序设计理论



湘潭大学出版社

计算机科学的数学基础

ISBN 978-7-81128-006-7
9 787811 280067 >
定价:45.00 元

计算机科学的数学基础

周经野　刘任任　编著

JISUANJI KEXUE DE SHUXUE JICHU

湘潭大学出版社

图书在版编目(CIP)数据

计算机科学的数学基础/ 周经野，刘任任编著. -湘潭：湘潭大学出版社，2007. 9

ISBN 978-7-81128-006-7

Ⅰ. 计… Ⅱ. ①周…②刘… Ⅲ. 电子计算机-数学基础-研究生-教材 Ⅳ. TP301.6

中国版本图书馆 CIP 数据核字 (2007) 第 138346 号

计算机科学的数学基础

周经野　刘任任　编著

责任编辑：朱美香
封面设计：罗志义
出版发行：湘潭大学出版社
社　　址：湖南省湘潭市 湘潭大学出版大楼
电话(传真)：0732-8298966　邮编：411105
网　　址：http://web.xtu.edu.cn:8080/pub/
印　　刷：湖南新华印刷集团邵阳有限公司
经　　销：湖南省新华书店
开　　本：787×1092　1/16
印　　张：22
字　　数：509 千字
版　　次：2007 年 9 月第 1 版　2007 年 9 月第 1 次印刷
印　　数：1-2000
书　　号：ISBN 978-7-81128-006-7
定　　价：45.00 元

前　言

形式语言与自动机理论、可计算理论、逻辑学和程序设计理论，都是研究计算模型的。它们之间也是相互关联的，共同构成了现代计算机科学技术的理论基础。这些理论都是属于数学学科的。形式语言与自动机理论、可计算理论和逻辑学的研究都始于20世纪初叶，特别是20世纪30年代的数学家Church（邱奇）、Gödel（哥德尔）、Kleene（克林）、Post（波斯特）以及Turing（图灵）等人的杰出工作催生了现代电子数字计算机的硬件和软件的诞生。程序设计理论的研究相比则要迟一些，是20世纪后半叶现代电子数字计算机以及程序设计语言和软件诞生之后的事情了。它是专门研究程序设计语言和程序设计方法的数学理论。这些工作对于计算机科学的实践和理论的发展有着深远的影响。比如，图灵机模型就被证明是现代电子数字计算机的理论模型。这些先驱者的工作在今天看来似乎是很平常的，它们的思想渊源甚至并不为今天众多的计算机的使用者所知道。但是这些先驱者的工作确实是应该被那些从事计算机科学技术的工作者们所熟悉、所掌握的。因为这些思想和方法将对他们的工作产生很重要的启示和指导作用。正是因为这一点，形式语言与自动机理论、可计算理论、逻辑学和程序设计理论一直以来都是国内外计算机科学技术专业硕士研究生的课程，而且还是作为重要的课程来开设的。

然而，20多年来计算机科学技术已经发生了翻天覆地的变化，它作为一门年轻的学科已经增长到了几乎无法想象的程度，而同时计算机的应用也已经到了无所不在的地步。在这种情形下，人们都感到有必要对原有的课程设置作一番调整。尽管在计算机科学技术领域越来越追求实用化，但是我们仍然执著地认为系统地了解和掌握计算机科学的数学理论对于从事计算机科学技术的工作者是重要的。本着“厚基础、宽口径、重能力”的精神，我们在计算机科学技术硕士研究生的教学计划的调整中将这几门课程合并为一门课程，取名为《计算机科学的数学基础》，并把它作为计算机科学技术一级学科的硕士研究生的必修课程。通过这几年的教学实践，我们感到这个调整是成功的。但是一直苦于在国内外找不到一本与之相适应的统一的教材，这样就萌发了编写这样一本教材的想法。我们的想法立刻得到了湘潭大学出版社的支持。在他们的大力支持下，我们完成了这本

书的编写。

这本书定位为计算机科学技术专业的硕士研究生的教材，大致安排60课时。由于涵盖了原来的多门课程的内容，同时考虑到一般读者的阅读，内容必须保持一定的完整性和系统性，因此本书的内容比较多，其中还有小部分的内容是在计算机科学技术专业本科阶段的某些课程中涉及了的。因而在教学过程中应该有重点地讲述，而更多地指导鼓励学生阅读自学。这样也能够更好地培养学生的学习精神和学习方法。

当然，也可以作为该专业高年级本科生的选修课程，以及其他从事计算机科学技术工作的人们自学提高的教材。

本书的第一部分《形式语言与自动机理论》、第二部分《可计算理论》和第四部分《程序设计理论》由周经野编著，第三部分《逻辑学》由刘任任编著。在本书的编写工程中，陈振宇、高新宇、陈新伟、邹剑章、胡旺、刘娟、汤博、朱玲芳、刘玉珍等同志为文稿的输入和校对做了很多的工作，在此表示感谢。

作 者

2007年9月20日

目　录

第一部分　形式语言与自动机理论

第三部分　逻辑学

第十一章　命题逻辑与一阶逻辑

第十二章　直觉主义逻辑

第十三章　模态逻辑

第十四章　非单调逻辑

第十五章　模糊逻辑

第十六章　多值逻辑

第一部分
形式语言与自动机理论

第一章　语言与正规语言

1.1　符号、符号串及其运算

符号和符号串在形式语言中是非常重要的基本概念。任何一种语言，不论是自然语言，还是计算机程序设计语言，都是由该语言的基本符号所组成的符号串集合。每一个程序设计语言都有它自己的基本符号集，例如：任何一个用程序设计语言 PASCAL 写的程序都是由关键字，即类似 if、while、begin、end 等符号，以及字母、数字和界限符等基本符号所组成。这些基本符号构成的集合就是 PASCAL 语言的字母表。每个 PASCAL 程序都可看成是一个定义在这个字母表上的、按照一定规则构成的符号串。此外，在计算机科学的发展中，符号主义一直占据着非常重要的位置。符号主义的观点就是任何一个演算的过程，都是一个符号推演的过程。换言之，计算机中的运算实质上也是一个语言的问题。

语言的基础是字母表，我们首先就由字母表开始。

字母表　一个非空的有限集合称为字母表，通常用 Σ 或者大写的西文字母表示。字母表中的元素称作为字母或符号，一般用小写字母、数字等表示。

不同的语言可以有不同的字母表，例如，汉语的字母表包括汉字等。英语的字母表包含了 26 个英文字母。C 语言的字母表是由字母、数字、运算符、间隔符及 if、while 之类的

关键字等组成。

注意,这里符号被定义为字母表中的元素。换句话说,任何一个非空的有限集合中的元素都可以看作是一个符号,因此符号并不仅限于大家所熟悉的汉字、英文字母等等,也可以是图像或者任何其他的东西。

符号串 一个符号串是由字母表中的字母组成的一个有限序列。

例如,设字母表 $\Sigma=\{1,2\}$,则 1、2、12、121、1122 都是 Σ 上的符号串。

注意:在符号串中,符号的顺序是非常重要的,12 和 21 是 Σ 上的两个不同的符号串。符号串可用大写字母表示,设 $X=12$,则称 X 是按 1、2 次序组成的一符号串。

符号串的长度 符号串所包含符号的个数称为符号串的长度。符号串 w 的长度记为 $|w|$。

例如,符号串 121 中包含有 3 个字符,长度为 3,记作 $|121|=3$。

空串 长度为 0 的符号串称为空串,用 ε 表示。

注意:由于空串 ε 的长度为 0,即 $|\varepsilon|=0$,所以 ε 是不含有任何符号的符号串。

符号串的联结 联结是符号串的基本运算。两个符号串 X 和 Y 的联结,记为 XY,就是把 Y 跟随在 X 的后面形成的符号串。

例 1.1 设 $\Sigma=\{1,2\}$ 是一个字母表。设 $X=11$、$Y=22$ 分别是 Σ 上的两个符号串。则

$XY=1122$ 是 X、Y 两个符号串的联结,XY 也是 Σ 上的一符号串。

$YX=2211$ 是 Y、X 两个符号串的联结,YX 也是 Σ 上的一符号串。

一般来说,符号串的联结不满足交换律,符号串的联结是满足结合律的,即有,$(XY)Z=X(YZ)$。在例 1. 1 中,显然有 $XY\neq YX$,$(XY)X=X(YX)=112211$。

由于 ε 是不含符号的符号串,所以对任意符号串 X 都有,$\varepsilon X=X\varepsilon=X$。由此可以认为 ε 是符号串联结运算的单位元。

符号串的方幂 设 X 是符号串,把 X 自身联结 n 次后,得到的符号串 Z,即 $Z=XX\cdots XX=X^n$,称为 X 的方幂。我们约定 $X^0=\varepsilon$。这个定义可以递归地表示为:

$$X^n=\begin{cases}\varepsilon & n=0\\ X^{n-1}X & n>0\end{cases}$$

例 1.2 设 $\Sigma=\{a,b\}$,$X=ab$ 是 Σ 上的符号串,则 $X^3=ababab$,$X^0=\varepsilon$。

符号串的子串、前缀和后缀 符号串 V 是符号串 W 的子串,当且仅当存在符号串 X 和 Y,使得 $W=XVY$。这里,X 和 Y 都可能是空串 ε。如果 X 和 Y 都是 ε,显然 $W=V$,所以每个符号串都是它自身的子串。如果 $X=\varepsilon$,则 V 是 W 的前缀,而如果同时有 $Y\neq\varepsilon$,则 V 是 W 的真前缀。类似的,如果 $Y=\varepsilon$,则 V 是 W 的后缀,而如果同时有 $X\neq\varepsilon$,则 V 是 W 的真后缀。如果我们对符号串 W 的某一部分感兴趣,而对其余部分不感兴趣,可以采用省略写法:用 $W=X\cdots$,$W=\cdots Y$ 和 $W=\cdots V\cdots$ 来分别表示只对 W 的前缀,后缀和子串 V 感兴趣。

集合的联结 设 A 和 B 都是符号串的集合,定义集合 A 和 B 的联结为:

$$AB=\{XY\mid X\in A\text{ 且 }Y\in B\},$$

即集合 A 和 B 的联结是集合 A 中的符号串和集合 B 中的符号串的联结所构成的集合。

例 1.3　设集合 $A=\{ab,cd\}$，$B=\{10,01\}$，$C=\{\varepsilon\}$，$D=\varnothing$，则

$$AB=\{ab10,cd10,ab01,cd01\},AC=\{ab,cd\}=A,AD=\varnothing,$$

这里要注意集合 C 和集合 D 的不同，集合 D 是空集，不含有任何的符号串；而集合 C 中含有符号串 ε，虽然 ε 是一个不含符号的空串，但是仍然是一个符号串，所以集合 C 不为空。

集合的方幂　设 A 是符号串的集合，把 A 自身联结 n 次后，得到的新的集合 A^n，即 $A^n=A\cdots A\cdots A$，称为集合 A 的方幂。我们约定 $A^0=\{\varepsilon\}$。这个定义可以递归地表示为：

$$A^n=\begin{cases}\{\varepsilon\} & n=0\\ A^{n-1}A & n>0\end{cases}$$

集合的闭包和正闭包　设 A 是符号串的集合，用 A^* 表示 A 的所有的有限次方幂的并集，则称 A^* 为集合 A 上的闭包，即：

$$A^*=A^0\cup A^1\cup A^2\cup\cdots\cup A^n\cup\cdots=\cup_{i=0}^{\infty}A^i。$$

而称 $A^+=A^1\cup A^2\cup\cdots\cup A^n\cup\cdots$ 为 A 上的正闭包，显然有 $A^*=A^0\cup A^+$，$A^+=A^*A=AA^*$。

例 1.4　设 $A=\{a,b\}$，则

$$A^*=\{\varepsilon,a,b,aa,ab,ba,bb,aaa,aab,\cdots\}$$

$$A^+=\{a,b,aa,ab,ba,bb,aaa,aab,\cdots\}$$

注意　闭包 A^* 与正闭包 A^+ 的差别在于是否包含空串 ε。在闭包 A^* 中去掉空串 ε 后就成为正闭包 A^+。A^* 具有可数无穷多的符号串。

特别是，一个字母表 Σ 的闭包 Σ^* 就是在字母表 Σ 上的所有符号串的集合。

世界上所有的语言都有一个字母表，都可以看作是这个字母表上的一些符号串的集合。于是我们可以抽象地将语言定义如下：

语言　令 Σ 为一个字母表。若 $L\subseteq\Sigma^*$，则 L 是字母表 Σ 上的一个语言。

1.2　文法与语言的形式定义

语言被抽象地定义为在某个字母表 Σ 上的一些符号串的集合后，还要刻画出这个集合是如何构成的。为此，需要对它们进行更进一步形式化的描述。通常语言都是用文法来描述的。在定义了符号、符号串及其运算的基础上，让我们来考虑给出文法的形式化定义。

一个文法实际上是一组有限的规则式。这些规则式给出语言中的各种语法成分以及它们是如何组成句子的。因此，为了定义规则式，还需要引进一类新的符号、语法符号。这类符号不是字母表中的符号，因此他们不会出现在语言的句子中。为了区分字母表上的符号和语法符号，将它们分别称为终结符和非终结符。

终结符　是一个语言的字母表中的符号。将它记为 T。

非终结符　也是一种符号，但不是字母表中的符号。在一个形式语言中，每一个非终结符表示的该语言的一个语法成分，并不出现在该语言的句子中。将它记为 V。

对于一个形式语言 L，设 T 和 V 分别是它的终结符集和非终结符集，显然有 $L\subseteq T^*$，且 $T\cap V=\varnothing$。

在下面的讨论中，把终结符集和非终结符集统称为符号集，记为 V'，即 $V'=T\cup V$。

1.2.1 文法的形式化定义

语言是用文法来定义的。文法的核心是一组规则式，又称规则式为产生式。

定义 1.1 一条产生式（又称生成式）是一个有序对(α, β)，通常可写作如下形式：

$$\alpha ::= \beta \text{ 或 } \alpha \rightarrow \beta,$$

其中$\alpha \in V$, $\beta \in V'^{*}$，α称为产生式的左部，β称为产生式的右部。

一个产生式表示该产生式的左部α可以用右部β来定义，符号::=或→可以读作“可以是”或者“可定义为”，即“α可以是β”或者“α可定义为β”。

注意：$\alpha \in V$说明α是一个非终结符且$\alpha \neq \varepsilon$，即产生式的左部不允许是空串。$\beta \in V^{*}$说明产生式的右部是这样的一个符号串，它可以含有终结符，也可以含有非终结符，同时还可以为空串。

定义 1.2 文法 G 定义为一个四元组

$$G = (V, T, P, S)$$

其中：

1. V是一个非空的有穷集合，称为非终结符集。
2. T是一个非空的有穷集合，称为非终结符集，且$V \cap T = \varnothing$。
3. P是一个非空的有穷的产生式的集合。
4. $S \in V$，称为文法的开始符号，S至少要在P中的一条产生式中作为左部出现。

例 1.5 设文法$G = (V, T, P, S)$，其中$V = \{A\}$，$T = \{a, b, c\}$，$P = \{A \rightarrow aAb, A \rightarrow c\}$，$S = A$。

这样就给出了一个文法 G。上述文法可缩写为$G = (\{A\}, \{a, b, c\}, P, A)$，其中：$P = \{A \rightarrow aAb, A \rightarrow c\}$。

例 1.6 设文法$G = (\{A, E\}, \{a\}, P, A)$，其中$P = \{A \rightarrow a, A \rightarrow aE, E \rightarrow aA\}$。

在许多的文法中，有多条产生式的左部相同，为书写方便起见，可以将左部相同的产生式写成合并的产生式形式。在例 1.5 的文法 G 中，P中的两个产生式的左部相同，都是A，可以合并写成$A \rightarrow aAb \mid c$，即$P = \{A \rightarrow aAb \mid c\}$。在例 1.6 的文法 G 中，$P$中的前两个产生式的左部相同，都是$A$，可以合并为$A \rightarrow a \mid aE$，这样一来，$P = \{A \rightarrow a \mid aE, E \rightarrow aA\}$。

例 1.7 设文法$G = (\{\langle\text{标识符}\rangle, \langle\text{字母}\rangle, \langle\text{数字}\rangle\}, \{a, \cdots, z, 0, \cdots, 9\}, P, \langle\text{标识符}\rangle)$，其中：

$P = \{\langle\text{标识符}\rangle \rightarrow \langle\text{字母}\rangle \mid \langle\text{标识符}\rangle\langle\text{字母}\rangle \mid \langle\text{标识符}\rangle\langle\text{数字}\rangle$

$\langle\text{字母}\rangle \rightarrow a \mid \cdots \mid z$

$\langle\text{数字}\rangle \rightarrow 0 \mid \cdots \mid 9\}$

为了明显地区分终结符和非终结符，一般情况下，在需要区分的时候使用尖括号“<”和“>”将非终结符括起来。

在许多情况下，不需要将文法的四元组显式地书写出来，而只需要将文法的产生式写出就可以表明该文法了。我们可以约定，第一条产生式的左部是文法的开始符；用尖括号“<”和“>”括起来的符号是非终结符；不用尖括号“<”和“>”括起来的符号是终结符，或用大写字母表示非终结符，小写字母表示终结符。此外也可将文法 G 写为$G[S]$，其中S是文法 G 的开始符。比如，例 1.5 中文法可以写为：

$$G: A \rightarrow aAb \mid c \text{ 或者 } G[A]: A \rightarrow aAb \mid c$$

1.2.2　推导的形式化定义

以上我们已经对文法进行了定义。假如已经定义了一个文法 G，如何来确定 G 产生的语言呢？语言中的句子应该都是用它的文法中的规则式推导出来的。因此，为了给出文法所生成的语言，还需要对推导进行形式化的定义，以便通过推导来生成符合文法的句子。

定义 1.3　给定一个文法 $G=(V, T, P, S)$，如果 $\alpha\to\beta$ 是 G 中的一条产生式（即：$\alpha\to\beta\in P$），δ 和 γ 是 V^* 中的任意符号，若存在符号串 x，y 满足：$x=\delta\alpha\gamma$，$y=\delta\beta\gamma$，则称 x 使用了产生式 $\alpha\to\beta$ 直接产生了 y，或者称 y 是 x 的直接推导，或者称 y 可以直接归约到 x，记作 $x\Rightarrow y$。

对上述例 1.5 中的文法 G，可以给出如下的一些直接推导的例子：

令 $x=aAb$，$y=acb$，$\delta=a$，$\gamma=b$，则 y 是 x 的直接推导，即：$aAb\Rightarrow acb$，所使用的产生式为 $A\to c$。

令 $x=A$，$y=aAb$，$\delta=\varepsilon$，$\gamma=\varepsilon$，则 y 是 x 的直接推导，即：$A\Rightarrow aAb$，所使用的产生式为 $A\to aAb$。

令 $x=aAb$，$y=aaAbb$，$\delta=a$，$\gamma=b$，则 y 是 x 的直接推导，即：$aAb\Rightarrow aaAbb$，所使用的产生式为 $A\to aAb$。

对于例 1.7 中的文法 G，直接推导的例子如：

令 $x=$ <标识符>，$y=$ <标识符> <字母>，$\delta=\varepsilon$，$\gamma=\varepsilon$，则 y 是 x 的直接推导，即：<标识符> ⇒ <标识符> <字母>，所使用的产生式为 <标识符> → <标识符> <字母>。

令 $x=$ <标识符> <字母> <数字>，$y=$ <字母> <字母> <数字>，$\delta=\varepsilon$，$\gamma=$ <字母> <数字>，则 y 是 x 的直接推导，即：<标识符> <字母> <数字> ⇒ <字母> <字母> <数字>，所使用的产生式为 <标识符> → <字母>。

令 $x=bc12$ <数字>，$y=bc129$，$\delta=bc12$，$\gamma=\varepsilon$，则 y 是 x 的直接推导，即有：$bc12$ <数字> ⇒ $bc129$，所使用的产生式为 <数字> →9。

定义 1.4　给定一个文法 $G=(V, T, P, S)$，设 $x, y\in V^*$，如果：

1. 存在如下的直接推导序列：

$$x=w_0\Rightarrow w_1\Rightarrow w_2\Rightarrow\cdots\Rightarrow w_n=y(n>0)$$

则称 x 推导出（产生）y，推导长度为 n，或者称为 y 归约到 x，记作 $x\Rightarrow^n y$。

2. 用 $x\Rightarrow^+ y$ 表示存在 $n>0$ 且 $x\Rightarrow^n y$；用 $x\Rightarrow * y$ 表示有 $x\Rightarrow^+ y$ 或者 $x=y$。

对上述例 1.5 中的文法 G，可以给出如下的直接推导序列的例子：

$$x=A\Rightarrow aAb\Rightarrow aaAbb\Rightarrow aaaAbbb\Rightarrow aaacbbb=y,$$

也可以记作：$A\Rightarrow^4 aaacbbb$，或者 $A\Rightarrow^+ aaacbbb$，或者 $A\Rightarrow * aaacbbb$。

对上述例 1.7 中的文法 G，可以给出如下的直接推导序列的例子：

$x=$ <标识符> ⇒ <标识符> <数字>

⇒ <标识符> <字母> <数字>

⇒ <字母> <字母> <数字>

⇒b <字母> <数字> ⇒bd <数字>

$\Rightarrow bd0$

即：<标识符>$\Rightarrow^{+} bd0$，

也可以记作：<标识符>$\Rightarrow^{4} bd0$，或者<标识符>$\Rightarrow^{+} bd0$，或者<标识符>$\Rightarrow^{*} bd0$。

从上述符号串 $bd0$ 的推导序列可以看出，在推导的某一步，可能会出现多个非终结符。例如，在进行第二步推导前，有<标识符>、<字母>和<数字>三个非终结符，此时，需要对在哪一个非终结符上和使用哪一条产生式做出选择。一般地说，符号串可能会有多种不同的推导序列。为了规范符号串的推导过程，在形式语言中定义了如下术语：

最左(右)推导 如果在推导的每一步 $x \Rightarrow y$，都是对 x 中的最左(右)边的非终结符选用产生式进行替换，则这种推导称为最左(右)推导。最右推导也称为**规范推导**。

上述对符号串 $bd0$ 的推导是最左推导，同样，还可以给出如下的最右推导(规范推导)。

<标识符>$\Rightarrow$ <标识符><数字>

$\Rightarrow$ <标识符>0

$\Rightarrow$ <标识符><字母>0

$\Rightarrow$ <标识符>$d0$

$\Rightarrow$ <字母>$d0$

$\Rightarrow bd0$

应当注意的是，只要在符号串中存在非终结符，而且在文法中存在以该非终结符为左部的产生式，那么就能使用该产生式推导出新的符号串出来。但是，如果符号串中不存在非终结符，那么推导过程就必须终止。

定义 1.5 给定一个文法 $G=(V, T, P, S)$，如果符号串 x 是从文法 G 的开始符号 S 推导出来的，即 $S \Rightarrow^{*} x$，则称 x 是文法 G 的句型。如果符号串 x 是仅由终结符组成的句型，即 $S \Rightarrow^{*} x$ 且 $x \in T^{*}$，则称 x 是文法 G 的句子。

在例 1.5 中，符号串 aAb、$aaAbb$、$aaaAbbb$、$aaacbbb$ 是该文法的句型，其中符号串 $aaacbbb$ 是该文法的句子。在例 1.6 中，像符号串：aE、aaA、$aaaE$、$aaaaA$、$aaaaa$。都是该文法的句型，其中符号串 $aaaaa$ 是该文法的一个句子。而在例 1.7 中，像符号串：<标识符><数字>、<标识符><字母><数字>、<字母><字母><数字>、b <字母><数字>、bd <数字>、$bd0$ 是该文法的句型，其中符号串 $bd0$ 是该文法的一个句子。

注意：给定一个文法 G 和一个符号串 x，要判定 x 是否是 G 的一个句型，实际上就是要从 G 的开始符号开始，能否通过一系列直接推导，推出 x。如果能推导出 x，那么 x 就是 G 的一个句型，否则 x 不是 G 的句型。如果文法 G 的一个句型 x 是仅由 G 的终结符组成的符号串，即：x 中没有 G 的非终结符，那么句型 x 就是 G 的句子。

对于给定的一个文法，从文法的开始符，每使用一次产生式所得到的新符号串是该文法的一个句型。由于对一个符号串推导，可能会有多种不同的推导序列，例如：最左推导序列、规范推导序列等，不同的推导序列所产生的句型是不同的，由规范推导所得到的句型就称之为规范句型。比如，在例 1.6 中，符号串<标识符><数字>、<标识符>0、<标识符><字母>0、<标识符>$d0$、<字母>$d0$、$bd0$ 等都是规范句型。下面给出与句型有关的一些概念的定义，它们是：短语、直接短语和句柄。

定义 1.6　设 $G[S]$ 是一文法，$x=\alpha w\beta$ 是一句型，如果：

$$S\Rightarrow\alpha A\beta \text{ 且 } A\Rightarrow^{+} w$$

称 w 是句型 x 的一个相对于非终结符 A 的短语；如果：

$$S\Rightarrow\alpha A\beta \text{ 且 } A\Rightarrow w$$

称 w 是句型 x 的一个相对于非终结符 A 的直接短语（或简单短语）；如果 w 是一个句型 x 的最左直接短语，称 w 为句型 x 的句柄。

下面通过例 1.7 中对句型 $bd0$ 的推导过程，进一步理解短语、直接短语和句柄的概念。

因为 <标识符> $\Rightarrow^{*}$ <标识符>0，并且 <标识符> $\Rightarrow^{+}$ <标识符>d，所以 <标识符>d 是句型 <标识符>$d0$ 的一个相对于非终结符 <标识符> 的短语。

因为 <标识符> $\Rightarrow^{*}$ <标识符>d <数字>，并且 <数字> $\Rightarrow 0$，所以 0 是句型 <标识符>$d0$ 的一个相对于非终结符 <数字> 的直接短语，但 0 不是句柄，因为在直接短语 0 的左边，句型 <标识符>$d0$ 还有直接短语 d。

<标识符> $\Rightarrow$ <标识符> <字母>0 且 <字母> $\Rightarrow d$，则称 d 是一个相对于非终结符 <字母> 的句型 <标识符>$d0$ 的直接短语，而且是最左直接短语，因此 d 是一个句柄。

1.2.3　语言的形式化定义

由上所述，文法是描述语言的装置，它可以说是生成语言的一种工具，也可以说是识别语言的一种工具。下面形式化地给出了文法所生成的语言的定义。

定义 1.7　给定一个文法 $G=(V, T, P, S)$，由 G 所生成（或产生）的语言记作 $L(G)$，它的定义为：$L(G)=\{x \mid S\Rightarrow^{+}x \text{ 且 } x\in T^{*}\}$，其中 x 称为语言 $L(G)$ 的句子。

有了推导和由文法产生语言的概念，可以看出：文法 G 中的 P 是产生语言的一组产生式，语言 $L(G)$ 中的每一个句子 x，是由 G 的开始符 S 经推导得到的，且完全由终结符组成的符号串。文法产生的语言是该文法生成的所有句子的集合。

非终结符是引起推导可以继续的中间符号。在推导进行到某一步时，如果再没有非终结符留在推导的结果中，则称推导结束；不论推导进行到哪一步，若总有非终结符留在推导的结果中，则称推导失败。

如果由文法 G 的开始符 S 经推导得不到任何的句子，也就是说，由文法 G 的开始符 S 所进行的推导都是失败的，那么称该文法 G 所产生的语言是空语言，记为：$L(G)=\varnothing$。因此，一个文法 G 总能产生一个语言 $L(G)$，$L(G)$ 中可能包含一些句子，也可能不包含任何句子。注意，如果文法 G 生成语言 L，即 $L(G)=L$，则任何用文法 G 推导出来的句子必定是 L 中的符号串。而另一方面，L 中的任何符号串也必定是可以用文法 G 推导出来的句子。

例 1.8　给定文法 $G[S]$：$S\rightarrow aSb \mid ab$

不难看出，由该文法生成的任何一个句子都是：a^nb^n

先使用产生式 $S\rightarrow aSb$ 若干次得到：$S\Rightarrow aSb\Rightarrow aaSbb\Rightarrow\cdots\Rightarrow a^{n-1}Sb^{n-1}$，即 $S\Rightarrow^{+}a^{n-1}Sb^{n-1}$；再使用产生式 $S\rightarrow ab$ 一次得到：$S\Rightarrow^{+}a^{n-1}Sb^{n-1}\Rightarrow a^nb^n$。

不难对推导的步数用数学归纳法证明该文法推导的所有符号串都是 a^nb^n 的形式。

另一方面，也不难对符号串的长度用数学归纳法证明，对任何形式为 a^nb^n，$n\geqslant 1$ 的符