编译程序设计原理

杜淑敏　王永宁　编著

北京大学出版社

内容简介

本书系统地介绍了编译程序设计的基本原理和方法，较详细地论述了当前各种语法分析技术。书中采用语法制导翻译方法。在文法符号的综合属性和继承属性等概念的基础上，给出了语法制导定义的形式，使得语义分析颇为清晰。在书的编写方式上，着重问题的提出、分析与解决。在内容的选取上，注意深入浅出。本书可作为高等学校计算机系各专业"编译原理"或"编译方法"课程的教材或参考书，也可供其它专业学生及从事计算机工作的有关人员阅读参考。

编译程序设计原理

杜淑敏　王永宁　编著

责任编辑：　邱淑清　彭令华

*

北京大学出版社出版

（北京大学校内）

北京大学印刷厂印刷

新华书店北京发行所发行　各地新华书店经售

*

850×1168 毫米　32 开本　13.5 印张　335 千字

1990 年 11 月第一版　　1990 年 11 月第一次印刷

印数：　0001—5,000 册

ISBN 7-301-01210-1/TP・054

定价：　7.30 元

目　录

第一章　语言的基本知识

在本章中我们将简单介绍形式语言的基本知识。这些知识对学习本书其它章节的内容是必不可少的。

1.1　基本定义

语言是人类思考问题和交流思想的工具。它是随着社会的产生而产生、发展而发展的，是人类传达思想、感情和希望的工具。凡是与人类社会生活有关的活动，都离不开语言。可以说，语言是在人们认识世界和改造世界过程中发展起来的。

计算机的出现促进了语言学的研究，特别是形式语言学的研究。自从著名语言学家 Noam Chomsky 于 1956 年建立了形式语言的描述以来，形式语言学理论发展得很快。这种理论对计算机科学有着深刻的影响，对程序语言的设计和编译程序的构造有着重大的作用。这正是本书所要讨论的课题。程序语言是形式化语言，是人和计算机相互传达信息的工具。以下我们将介绍有关形式语言的几个基本概念。这些基本概念来源于自然语言学的研究。为了使读者易于掌握这些内容，在形式地引进定义之前，我们将首先对文法和语言进行一般的讨论。

1.1.1　文法的讨论

在本节中，我们从一个具体英语例句的分析出发，引进有关文法和语言的基本概念。请考虑英语句子“The grey wolf will eat the goat”。根据英语知识，知道这是一个英语句子，并且可以进行图

解，如图 1.1 所示。这是一棵有序有向树(简称树)。这种图解把句子分解成它的各个组成部分。由图 1.1 可以看出，〈句子〉是由〈主语〉后随〈谓语〉组合而成的；〈主语〉又是由〈冠词〉后随〈形容词〉、再随〈名词〉构成的，等等。

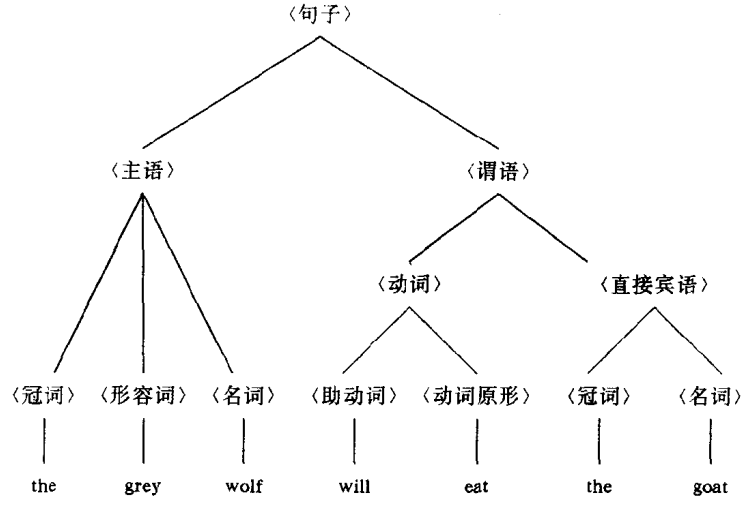

图 1.1 一棵树

为了描述这种结构，在图 1.1 中我们使用了一些新的符号，即所谓语法单位(或称语法实体)，如〈句子〉、〈主语〉、〈冠词〉等等。图 1.1 中凡是用尖括号〈 和 〉括起来的都是语法单位。这样就把所用到的语法单位和语言中的基本字较为明显地区分开了。

如前所述，图 1.1 表明〈句子〉是由〈主语〉后随〈谓语〉组合成的。为了能机械地进行这样的分解，我们必须给出一些形式的精确的规则，用以表明句子的结构。如果我们用符号“→”(或“::=”)表示“定义为”(或“由...组合成的”)，用符号“|”表示“或”，那末，

我们可使用这些符号构成规则。在此例中所需要的全部规则如下：

〈句子〉→〈主语〉〈谓语〉 (1)

〈主语〉→〈冠词〉〈形容词〉〈名词〉 (2)

〈冠词〉→ the (3)

〈形容词〉→ grey (4)

〈谓语〉→〈动词〉〈直接宾语〉 (5)

〈动词〉→〈助动词〉〈动词原形〉 (6)

〈助动词〉→will (7)

〈动词原形〉→eat (8)

〈直接宾语〉→〈冠词〉〈名词〉 (9)

〈名词〉→wolf (10)

〈名词〉→goat (11)

而后面的两个规则可以写在一起：

〈名词〉→wolf|goat

规则(1)读作：〈句子〉是由〈主语〉后随〈谓语〉组合而成的，或读作：〈句子〉定义为〈主语〉后随〈谓语〉。规则(2)则读作：〈主语〉是由〈冠词〉后随〈形容词〉再后随〈名词〉组合而成，等等。

我们把描述语言的语法结构的形式规则称为文法。上述十一个规则，可以说是构成了一个小型文法。让我们仔细分析一下这个文法。此文法包括四个组成部分：

1. 一组终结符号。这里包括 the，grey，wolf，will，eat 和 goat 等六个字。

2. 一组非终结符号。非终结符号用来代表语法单位。这里包括〈句子〉、〈主语〉、〈冠词〉、〈形容词〉、〈谓词〉、〈动词〉、〈助动词〉、〈动词原形〉、〈直接宾语〉和〈名词〉等十个语法单位。

3. 一个开始符号。开始符号是一个特殊的非终结符号。这里，〈句子〉是开始符号。

4. 一组规则(也称产生式或产生规则)。这里共包括十一个规

则。

综上所述，如果我们用 G 表示这一文法，则可写成：

$G=(\{$ the , grey , wolf , will , eat , goat$\}$,
$\{$〈句子〉,〈主语〉,〈冠词〉,〈形容词〉,〈谓词〉,
〈动词〉,〈助动词〉,〈动词原形〉,〈直接宾语〉,
〈名词〉$\}$,〈句子〉,$\mathscr{P}$),

其中 $\mathscr{P}$ 表示由上述十一个规则所组成的集合。

当我们有了一组规则之后，就可以按照下述方式用它们去**推导**或**产生**句子。我们从开始符号〈句子〉开始，在→的左端找带有〈句子〉的规则，并把〈句子〉改写成→的右端的符号串：

〈句子〉⇒〈主语〉〈谓语〉

重复此过程，在符号串〈主语〉〈谓语〉中取一语法单位，比如说〈主语〉，我们再找→的左端有〈主语〉的规则，并用相应的右端的符号串代替原来符号串中的〈主语〉，这就产生了：

〈主语〉〈谓语〉⇒〈冠词〉〈形容词〉〈名词〉〈谓语〉

符号"⇒"意味着：使用文法中的一条规则，代替⇒左端的某个符号，产生⇒的右端的符号串。句子"The grey wolf will eat the goat"的全部推导过程将是：

〈句子〉⇒〈主语〉〈谓语〉
⇒〈冠词〉〈形容词〉〈名词〉〈谓语〉
⇒the 〈形容词〉〈名词〉〈谓语〉
⇒the grey 〈名词〉〈谓语〉
⇒the grey wolf 〈谓语〉
⇒the grey wolf 〈动词〉〈直接宾语〉
⇒the grey wolf 〈助动词〉〈动词原形〉〈直接宾语〉
⇒the grey wolf will 〈动词原形〉〈直接宾语〉
⇒the grey wolf will eat 〈直接宾语〉
⇒the grey wolf will eat 〈冠词〉〈名词〉

$\Rightarrow$the grey wolf will eat the 〈名词〉

$\Rightarrow$the grey wolf will eat the goat

我们用一个新符号“$\overset{+}{\Rightarrow}$”，把该句子的上述推导缩写为：

〈句子〉$\overset{+}{\Rightarrow}$ the grey wolf will eat the goat

通常在每一步推导中，可以去代替任意一个语法单位。只是在上述推导中，总是去代替最左面的那一个语法单位。

文法的目的之一就是用适当条数的规则把语言的全部句子描述出来。对上述文法 G 而言，除上述句子外它还可以描述以下几个句子：

“the grey wolf will eat the wolf”

“the grey goat will eat the wolf”

“the grey goat will eat the goat”

尽管它们之中在语义上(含义上)有的可能是不合适的，但在语法上是正确的。小型文法 G 描述了由四个句子构成的语言。以后我们会看到，常常是用有限个规则把具有无穷多个句子的语言描述出来，因而这样做是有意义的。

通过以上介绍，我们大致上做好了描述文法和语言的形式化概念的准备工作。

1.1.2 字母表

字母表是符号的非空有穷集合。“符号”是一个抽象实体，我们将不去形式地定义它，就如同几何学中的“点”和“线”一样往往不加定义。字母和数字是经常使用的符号。例如，由符号“a”组成的字母表记作$\{a\}$；由符号“a”和“b”组成的字母表记作$\{a,b\}$；由符号“a”，“b”和“c”组成的字母表记作$\{a,b,c\}$；由符号“0”和“1”组成的字母表记作$\{0,1\}$；而 ASCII 字符集是一个常用的计算机字母表的例子。

1.1.3 符号串

由字母表中的符号所组成的任何有穷序列被称之为该字母表上的符号串。符号串也称作“字”。例如，设有字母表 $\Sigma=\{a,b,c\}$，那么，序列 ab 是 Σ 上的一个符号串；同样序列 ba，序列 abc，序列 $bcca$ 等都是 Σ 上的符号串。在语言的理论中，术语“句子”和“字”常常用作术语“符号串”的同义语。符号串 s 的长度记作 $|s|$，是组成该符号串的符号的个数。例如，上述 Σ 上的符号串 ab 的长度是 2，记作 $|ab|=2$。符号串 abc 的长度是 3，记作 $|abc|=3$。空符号串记作 ε，它由零个符号组成，于是 $|\varepsilon|=0$。在本书中常用的与符号串有关的几个术语有：

1. 符号串 s 的前缀： 移走符号串 s 的尾部的零个或多于零个符号所得到的一个符号串。例如 ban 是符号串 banana 的一个前缀。

2. 符号串 s 的后缀： 删去符号串 s 的头部的零个或多于零个符号所得到的一个符号串。例如 nana 是符号串 banana 的一个后缀。

3. 符号串 s 的子串： 从 s 中删去一个前缀和一个后缀而得到的符号串。例如 nan 是 banana 的一个子串。s 的每一个前缀和每一个后缀是 s 的一个子串，但是 s 的每一个子串不一定是 s 的前缀或后缀。对于每一个符号串 s，s 和 ε 两者都是 s 的前缀、后缀和子串。

4. 符号串 s 的真前缀、真后缀、真子串： 任何非空符号串 x，相应地，是 s 的前缀、后缀或子串，并且 $s\neq x$。

5. 符号串 s 的子序列： 从符号串 s 中删去零个或多于零个符号（这些符号不要求是连续的）而得到的符号串。例如 baaa 是 banana 的一个子序列。

术语“**语言**”表示某个确定的字母表上的符号串的任何集合。

这个定义是非常广泛的。在此意义下，不含任何元素的空集合∅，即集合{ }，以及只含空符号串ε的集合{ε}都是语言。大家更为熟悉的例子有如：所有合乎 Pascal 语法的程序所组成的集合以及所有合乎英文文法的句子所组成的集合都是语言。当然这后两者难于给出详细说明。另外要注意的是，这个定义没有描述在一个语言中的符号串的任何含义。有关描述符号串的含义的方法将在第五章进行讨论。

下面让我们介绍符号串之间的运算。

符号串 α,β 的连接 $\alpha\beta$ 是把符号串 β 写在符号串 α 之后得到的符号串。例如，若 $\alpha=ab,\beta=bc$，则有，$\alpha\beta=abbc,\beta\alpha=bcab$。按此约定，因 ε 是不包含任何符号的符号串，所以，对于任意符号串 α 而言，可以得到

$$\varepsilon\alpha=\alpha\varepsilon=\alpha$$

我们还可以定义符号串的方幂。假设 α 是符号串，α^n 定义为：

$$\underbrace{\alpha\alpha\cdots\alpha\alpha}_{n\text{ 个 }\alpha}$$

当 $n=0$ 时，α^0 是空符号串 ε。例如，若 $\alpha=a$，则有

$$\alpha^0=\varepsilon$$

$$\alpha^1=a$$

$$\alpha^2=aa$$

$$\cdots\cdots$$

$$\alpha^n=\underbrace{aa\cdots aa}_{n\text{ 个 }a}$$

又如，若 $\alpha=ab$，则有

$$\alpha^0=\varepsilon$$

$$\alpha^1=ab$$

$$\alpha^2=abab$$

……

$$\alpha^n = \underbrace{abab\cdots abab}_{n\text{个 }ab}$$

我们可以进一步考虑定义在语言之上的几个重要运算。假设 L 和 M 表示两个语言。

1. 语言 L 和 M 的合并(union),记作 $L \cup M$,定义为:

$$L \cup M = \{s \mid s \text{ is in } L \text{ or } s \text{ is in } M\}$$

此集合读作"所有属于 L 或属于 M 的符号串 s 所组成的集合"。并请注意不要把运算符号 $\cup$ 与字母 U 相混。

2. 语言 L 和 M 的连接(concatenation),记作 LM,定义为:

$$LM = \{st \mid s \text{ is in } L \text{ and } t \text{ is in } M\}$$

此集合读作"满足 s 属于 L 和 t 属于 M 的所有符号串 st 所组成的集合"。

因为对于任意符号串 x 总有 $\varepsilon x = x\varepsilon = x$,所以有:

$$\{\varepsilon\}L = L\{\varepsilon\} = L$$

注意,一般而言,$LM \neq ML$,但 $(LM)W = L(MW)$。L 自身的 n 次连接记为:

$$L^n = \underbrace{LL\cdots L}_{n\text{个}}$$

并规定 $L^0 = \{\varepsilon\}$。

3. 语言 L 的 Kleene 闭包,记作 L^*,定义为:

$$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \cdots$$

4. 语言 L 的正闭包,记作 L^+,定义为:

$$L^+ = \bigcup_{i=1}^{\infty} L^i = L^1 \cup L^2 \cup L^3 \cup L^4 \cup \cdots$$

以上运算是在语言上定义的,即它们对于某个字母表上的任何符号串集合都是适用的。

例 1.1 令 L 是集合{A,B,C,…,Z,a,b,c,…,z},并且 D 是集

合{0,1,2,…,9}。我们可以用两种方式来考虑 L 和 D。其一，我们可以把 L 当作包括 26 个大写字母和 26 个小写字母的字母表，把 D 当作包括 10 个数字的字母表；其二，我们可以把 L 和 D 都看作是由长度为 1 的符号串所组成的集合，这样，集合 L 和 D 每一个都是有限的语言。下面就是由语言 L 和 D 应用上述运算后而创建的一些新的语言：

1. $L \cup D$ 是由所有上述字母和数字所构成的集合。

2. LD 是由所有用一个字母后跟一个数字组成的符号串所构成的集合。

3. L^4 是由所有的四个字母的符号串构成的集合。

4. L^* 是由所有的字母组成的符号串，包括空符号串 ε，所构成的集合。

5. $L(L \cup D)^*$ 是由所有的字母打头的字母和数字符号串所构成的集合。

6. D^+ 是由所有的长度大于等于 1 的数字串所构成的集合。

我们知道，一种语言是某个确定的字母表上的某些符号串的集合。此集合是字母表上所有符号串，包括 ε，所构成的集合的一个子集。设字母表为 Σ，则集合

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots = \bigcup_{i=0}^{\infty} \Sigma^i$$

是 Σ 上的所有符号串，包括 ε，所构成的集合。为了详细刻画语言所表示的集合，需要我们对语言（形式语言）给出更为具体而形式化的定义。这是下面将要讨论的内容。

1.1.4 文法和语言的形式定义

本节我们将介绍上下文无关文法和上下文无关语言。所谓上下文无关文法是这样一种文法，它所定义的语法单位（或称语法范畴，或称语法实体）是完全独立于这种语法单位可能出现的上下文