配套国防工业版 陈火旺 等编著的《编译原理》

# 编译原理
# 学习与应用指导

主　编　张永梅　靳雁霞
副主编　李　玲　宋礼鹏

国防工业出版社
National Defense Industry Press

# 编译原理学习与应用指导

主　编　张永梅　靳雁霞
副主编　李　玲　宋礼鹏

国防工业出版社
·北京·

# 内 容 简 介

编译原理是计算机专业的核心课程之一，是每位优秀的计算机专业人员必修的一门课程。本书以研究程序设计语言编译程序构造的基本原理和基本实现方法为主要目标，系统地介绍了编译技术的基本原理、典型题解、上机实习方法以及编译原理在相关领域的典型应用。全书由十三章组成，前三章分别为：引论、高级语言及其语法描述、词法分析的主要内容及相应的题解。第四章至第十一章依次介绍语法分析，属性文法和语法制导翻译，语义分析和中间代码产生，符号表，运行时存储空间组织，优化，目标代码生成，并行编译基础的主要内容以及典型题解。第十二章介绍编译原理实习方法及其实例。第十三章给出了编译原理的一些典型应用。

本书编写时注重难点的分散安排，尽量由易到难，便于读者掌握。

本书既可作为高等学校计算机类专业本科和专科生的参考书，同时适合作为报考计算机专业研究生的复习指导用书，也可供相关科技人员参考。

**图书在版编目(CIP)数据**

编译原理学习与应用指导 / 张永梅，靳雁霞主编 .
北京：国防工业出版社，2006.3
ISBN 7-118-04396-6

Ⅰ. 编 … Ⅱ. ①张 …②靳 … Ⅲ. 编译程序－程序设计 Ⅳ. TP314

中国版本图书馆 CIP 数据核字(2006)第 014148 号

※

国防工业出版社出版发行
(北京市海淀区紫竹院南路 23 号 邮政编码 100044)
北京奥鑫印刷厂印刷
新华书店经售
*
**开本** 787×1092 1/16 **印张** 20¾ **字数** 518 千字
2006 年 3 月第 1 版第 1 次印刷 **印数** 1—4000 册 **定价** 35.00 元

**(本书如有印装错误，我社负责调换)**

国防书店：(010)68428422 发行邮购：(010)68414474
发行传真：(010)68411535 发行业务：(010)68472764

# 前　言

计算机语言之所以能由单一的机器语言发展到现在的数千种语言，就是因为有了编译技术。编译技术是计算机科学中发展得最迅速、最成熟的一个分支，它集中体现了计算机发展的成果与精华。编译原理是计算机专业设置的一门重要的专业课程，虽然只有少数人从事编译方面的工作，但是这门课在理论、技术、方法上都对学生提供了系统而有效的训练，有利于提高软件人员的素质和能力，有利于将本课程讨论的概念和技术应用于其他软件设计中，可以比较迅速地掌握新的语言工具。通过本课程的学习可以培养学生的抽象思维、逻辑推导和概括能力。

几乎每本编译原理的教材都是讲述编译程序的各个组成部分，如词法分析、语法分析、属性文法和语法制导翻译、语义分析和中间代码产生、符号表、运行时存储空间组织、优化、目标代码生成等内容。要求学生对编译原理的内容有较好的掌握，为后续课程的学习以及以后的工作打下良好的基础。

编译原理课程的一个明显特点是它的理论性很强，在学习的时候需要很强的逻辑思维能力。而独立完成作业是理解、消化理论知识最好的辅助手段，同时也有助于培养学生理论与实践结合、思考与创新的能力。

计算机科学本身就是一门实践性很强的课程，如果能够学以致用，那才叫真正的学会。自己编程实现编译器是最好的实践过程，这里的实践过程不是指课程中习题作业的完成，也不是指某些编译原理或方法的孤立实验，而是指学生要在编译原理的学习过程中完成一个规模适当的模型语言的编译程序，并且该编译程序必须概括地应用了课程中所介绍的主要理论和方法。换句话说，学生要将编译的主要理论和方法实现在一个各部分能够相互协调工作的统一体中，而不是将它们分别地进行孤立实验。

本书主要包括编译原理各章的学习指导、典型题解以及参考解决方案。引导读者实践从“学习理论”到“结合实际理解理论”再到“自己亲自动手解决问题”的学习过程，意在帮助读者深刻理解本课程涉及的原理和概念，掌握基本的学习方法，从而透彻地领悟本课程的精髓。

本书典型习题难易恰当，既可以作为计算机系本科生学习编译原理课程的参考书，又适合作为报考计算机专业研究生的复习指导书。

本书是作者结合多年来的教学实践经验编写而成的，由张永梅、靳雁霞主编，李玲、宋礼鹏副主编。在本书的编写过程中，得到了韩燮教授、刘仁富副教授的指导，并提出了许多宝贵意见，对本书的成稿起了很大的作用。

在本书的出版过程中，得到了中北大学教务处杨晓敏的大力协助。

在此谨向对本书的编写和出版做出指导和帮助的各位同志表示深深的谢意！

由于作者水平有限，书中难免存在一些缺点和错误，殷切希望广大读者批评指正。

作　者

2005 年 8 月

# 目　录

# 第1章 概 论

本章重点是对编译程序的功能和结构做一综述，难点是："遍(Pass)"的概念，编译程序各个组成部分在编译阶段的逻辑关系，以及它们如何作为一个整体来完成编译任务。本章需要读者掌握的是：清楚编译程序的总框架，了解编译程序工作的大致过程，能分辨清楚编译前端与后端的区别及其相互配合的方法，清楚编译程序是如何生成的。理解以下概念：编译程序，解释程序，翻译程序，编译前端与后端，"遍"的概念等。

## 1.1 学习指导

### 1.1.1 什么是编译程序

翻译程序指的是这样一个程序，它能够把某一种语言程序(源语言程序)改造成另一种语言程序(目标语言程序)，而两者在逻辑上是等价的。

程序设计语言的转换：

汇编程序：源程序是汇编语言，目标程序是机器语言的翻译程序。

编译程序：源程序是高级语言，目标语言是某种汇编语言或机器语言的翻译程序。

解释程序：将某种高级语言程序作为输入接受并解释执行，不产生能执行的结果目标代码。注意编译程序与解释程序的区别，一个语言的解释程序是这样的程序：它以该语言写的源程序作为输入，但不产生目标程序，而是边解释边执行源程序本身。

术语"编译"的内涵是实现从源语言所表示算法向目标语言所表示算法的等价变换。

### 1.1.2 编译程序各阶段任务

编译程序各阶段任务：

词法分析器：必要。

语法分析器：必要。

语义分析与中间代码产生器：可选。

优化：可选。

目标代码生成器：必要。

除了上述5个阶段外，一个完整的编译程序还应包括"表格管理"和"出错处理"两个阶段。

词法分析器又称扫描器，输入源程序，对构成源程序的字符串进行扫描和分解，根据词法规则(或构词规则)识别出一个个的单词(单词符号或符号)，转换成计算机容易识别的内码形式。

内码用二元式(单词种别，单词符号的属性值)表示。语法分析器，简称分析器，根据语言的语法规则(文法规则)，将单词符号串组成各类语法单位(语法范畴)，如：短语、子句、句子、程

序段、程序等。

语法规则写成 Backus-Naur-Form(BNF)式。语法分析有两种方法:推导(Derive)和规约(Reduce)。对说明语句语法分析填写符号表,对一般语句构造语法树。

语法分析器对单词符号串进行语法分析,即根据语法规则进行推导或规范,最终判断输入串是否构成语法上正确的"程序"。

语义分析与中间代码产生器:根据语义规则产生一种介于源语言与目标代码之间的一种中间代码。

中间代码是不依赖于机器但是又便于生成依赖于机器的目标代码的一种结构简单、含义明确的记号系统。中间代码经常用四元式来表示。

优化:对中间代码进行优化处理。

目标代码生成:把中间代码变换成指定机器上的绝对指令代码或可重新定位的指令代码或汇编指令代码。

目标代码生成与硬件系统功能部件的运用、机器指令的选择、各种数据类型变量的存储空间分配、寄存器的调度等有很大的关系。表格管理:编译程序在工作过程中需要建立一系列的表格,用以登记源程序的各类信息和编译各个阶段的进展状况。合理地设计和使用表格是编译程序构造的一个重要问题。

与编译有关的表格有:符号表、常数表、标号表、分程序入口表、中间代码表、过程引用表、循环特征表、等价名表、公用链表、格式表等。在编译程序使用的表格中,最重要的是符号表。它用来登记源程序中出现的每个名字以及名字的各种属性。

出错处理:错误可能发生在编译的各个阶段,出错处理也贯穿编译全过程。

词法分析阶段可查出的错误,如标识符的组成不符合词法规则;语句结构错误是在语法分析中可查出的错误;还有语义分析阶段可查出的错误,即结构正确,但所涉及的操作无意义或错误。

在编译阶段查出的错误,称为编译时错误(Compile-time error),在运行阶段才表现出来的错误,称为运行时错误(Run-time error)。

### 1.1.3 编译前端与后端

我们有时将编译程序划分为编译前端和编译后端。前端主要由与源语言有关但与目标机无关的那些部分组成。通常包括词法分析、语法分析、语义分析与中间代码产生,有的代码优化工作,也可以包括在前端。

后端包括编译程序中与目标代码有关的部分,例如与目标机有关的代码优化,以及目标代码的生成等。

### 1.1.4 遍(Pass)的概念

对源程序(或其中间形式)从头至尾扫描一次,并进行有关加工处理,生成新的中间形式或最终目标程序,称为一遍。每遍的工作均由从外存上获得前一遍的工作结果开始(对于第一遍来说,是从外存上获得源程序),到完成它所含的有关阶段程序的工作,并把结果记录于外存上为止。

影响分"遍"的因素有:内存,编译功能,源语言繁简,以及目标代码优化程序等。

一遍编译程序:同时驻留内存,各部分之间"调用连接"。

多遍编译程序：每遍的输出结果是下遍的输入对象。

### 1.1.5 编译程序生成方式

编译程序生成方式有如下六种：直接用机器语言编写编译程序、用汇编语言编写编译程序、用高级语言编写编译程序、利用编译工具、自编译、移植产生编译程序。

### 1.1.6 编译程序构造

要在某一台机器上为某种语言构造一个编译程序，必须掌握三方面的内容：源语言、目标语言、编译方法。

### 1.1.7 编译器设计最近的发展

编译器设计最近的发展与复杂的程序设计语言的发展结合在一起。如用于函数语言编译的 Hindley-Milner 类型检查的统一算法。

编译器已成为基于窗口的交互开发环境(IDE)的一部分，IDE 的标准并没有多少，但已经对标准的窗口环境进行了开发。近年来对此进行了大量研究，但是基本的编译器设计近 20 年来没有多大改变，现在正逐渐成为计算机科学课程中的中心一环。

随着多处理机的发展以及对并行处理的要求，最近的研究方向是并行编译。并行化编译技术的目的是提高并行计算机体系结构的性能，满足超大规模计算日益增长的需求——高性能计算机应运而生。

交叉编译技术，由于目标机指令系统与宿主机的指令系统不同，编译时将应用程序的源程序在宿主机上经过编译产生目标机的汇编语言或机器语言，称为交叉编译。

## 1.2 典型题解

1. 编译方式与解释方式的根本区别在于_____。

解：是否生成目标代码。

2. 请画出编译程序的总体结构图，并简述其各个组成部分的主要功能。

解：编译程序的总框图见图 1.1。

其中词法分析器，又称扫描器，它接受输入的源程序，对源程序进行词法分析，识别出一个个的单词符号，其输出结果为单词符号。

语法分析器对单词符号串进行语法分析(根据语法规则进行推导或归约)，识别出程序中的各类语法单位，最终判断输入串是否构成语法上正确的“程序”。

语义分析与中间代码产生器，按照语义规则对语法分析器归约出(或推导出)的语法单位进行语义分析，并把它们翻译成一定形式的中间代码。编译程序可以根据不同的需要选择不同的中间代码形式，有的编译程序可以没有中间代码形式，而直接生成目标代码。

优化器对中间代码进行优化处理。一般最初生成的中间代码执行效率都比较低，因此要进行中间代码的优化，其过程实际上是对中间代码进行等价替换(变换)，使得变换后的代码更为高效(运行结果与变换前代码运行结果相同，而运行速度加快或占用存储空间少，或两者都有)。

目标代码生成器将中间代码翻译成目标程序。中间代码一般是一种与机器无关的表示形

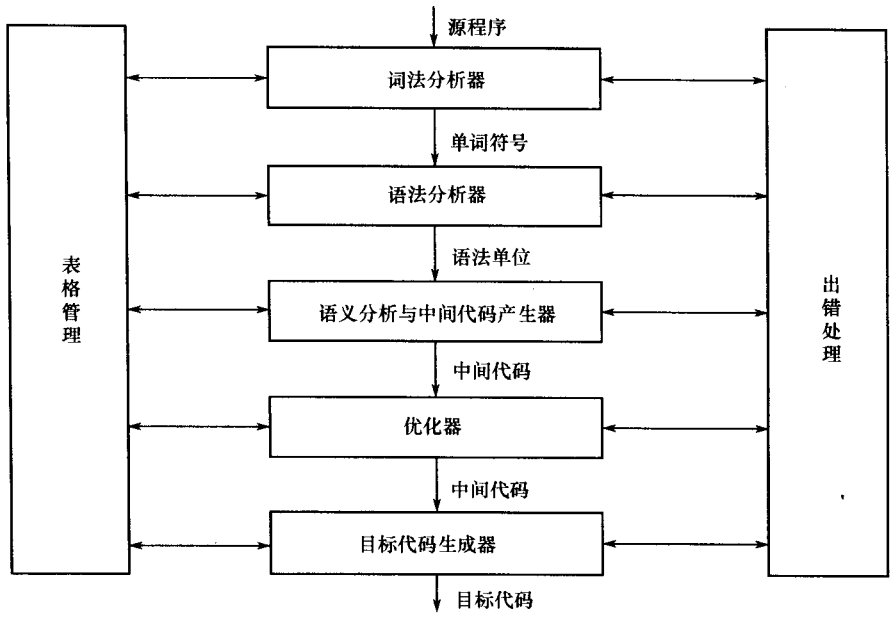


图 1.1　编译程序的总体结构图

式,只有把它再翻译成与机器硬件相关的机器能识别的语言,即目标程序,才能在计算机上运行。

表格管理模块保持一系列的表格,登记源程序的各类信息和编译各阶段的进展状况。编译程序各个阶段所产生的中间结果都记录在表格中,所需要的信息也大多从表格中获取,整个编译过程都在不断地和表格打交道。

出错处理程序对出现在源程序中的错误进行处理。如果源程序有错误,编译程序应设法发现错误,将有关错误信息报告给用户。编译程序的各个阶段都有可能发现错误,出错处理程序要对发现的错误进行处理、记录,并反映给用户。

3.列举出你所使用过的所有计算机语言和所有的"翻译"程序(编译、解释、汇编等)。

解:(1)汇编语言总是与具体的机器相关,早期的有 PDP11 和 VAX 的汇编语言及其汇编器,还有 Intel 系列微机上的 Z80 等汇编语言及其汇编器。

(2)采用解释方式翻译语言的典型代表是早期的 Basic 和 Basic 的解释器。另外操作系统的命令也是解释执行,如 UNIX 的 shell 文件(也有人称其为 shell 程序)。

(3)程序设计语言 Pascal;Ada83、Ada95;C/C++,编译器分别有 Turbo Pascal、VAXAda、ActiveAda、ObjectAda;Turbo C、C++Builder、VC++等。

(4)数据库查询语言 SQL,它一般有两种使用方式:交互式和嵌入式。交互式的翻译一般采用解释,而嵌入式的翻译一般采用编译(实质上是预编译,将 SQL 翻译为宿主语言,如 C、Ada 等)。SQL 的翻译一般由数据库管理系统提供,并且同时提供解释器和编译器,它们包括 Informix 的 isql 和 esql、Oracle 的 sqlplus 和 proc 等。

(5)语言识别器描述语言 Lex 和 Yacc,对应的编译器有 xdcfle 和 xdyacc。

4.如果在 Pascal 源程序中出现这样一些情况:12x　2 * /3　3.5+"end"(运行时 y=0),请指出它们分别是什么类型的错误。

解:12x 是一个词法错误;2 * /3 是一个语法错误,因为两个运算符 * 和/之间没有操作数;

3.5 + "end"是一个静态语义错误,因为子表达式的类型不匹配;x/y 是一个动态语义错误(运行错误或逻辑错误),运行时可能会因为溢出而造成停机。

5. 从你使用过的编译器中选择一个最熟悉的,写出从编写到运行一个应用程序的全过程。

解:当前普遍使用的是 VC 6.0。它是一个集成环境,包括了源程序的编写、编译和运行测试等全过程。使用 VC 6.0 进行程序设计的基本步骤包括:首先建立一个工程(project workspace)并将程序所涉及的文件加入到工程中;然后反复进行下述操作,直到结束。打开工程中的文件并且编写(或修改)源程序,编译并连接(单步或合并进行均可)源程序,若有错误,则修改;否则运行测试。

6. 计算机执行用高级语言编写的程序有哪些途径? 它们之间的主要区别是什么?

解:计算机执行用高级语言编写的程序主要途径有两种,即解释与编译。

像 BASIC 之类的语言,属于解释型的高级语言。它们的特点是计算机并不事先对高级语言程序进行全盘翻译,将其全部变为机器代码,而是每读入一条语句,就用解释器将其翻译为一条机器代码,并予以执行,然后再读入下一条语句,翻译为机器代码,再执行,如此反复。总而言之,是边翻译边执行。

像 C、Pascal 之类的语言,属于编译型的高级语言。它们的特点是计算机事先对高级语言程序进行全盘翻译,将其全部变为机器代码,再统一执行,即先翻译,后执行。

从执行速度上看,编译型的高级语言程序比解释型的高级语言程序运行要快,但解释方式下的人机界面比编译型好,便于程序调试。

两种途径的主要区别在于:解释方式下不生成目标代码程序,而编译方式下生成目标代码程序。

7. 三维数组 a[2:5, -2:2, 5:7]首地址为 100,每个数组元素占 4 个存储单元,请给出数组元素 a[3,1,6]的地址。

解:对 n 维数组 $A[l_1:u_1, l_2:u_2, \cdots, l_n:u_n]$,令 $d_i = u_i - l_i + 1, i = 1, 2, \cdots, n$,a 为数组首地址,k 为每个数组元素的长度(所占单元数),在按行排列的方式下,数组元素 $A[i_1, i_2, \cdots, i_n]$的地址计算公式为:

$$D = a + [(i_1 - l_1)d_2d_3\cdots d_n + (i_2 - l_2)d_3d_4\cdots d_n + \cdots + (i_{n-1} - l_{n-1})d_n + (i_n - l_n)] \times k$$

对本题有:$a = 100, d_2 = 5, d_3 = 3, k = 4$,数组元素 a[3,1,6]的地址为:

$$100 + [(3-2) \times 5 \times 3 + (1+2) \times 3 + (6-5)] \times 4 = 200$$

8. 对于下面的 ALGOL 说明语句所定义的数组 A

$$\text{array } A[-4:5, -3:3]$$

假定数组是按行为序存放的,存储器按字节编址,每 6 个字节为一机器字,令 A 的首地址为 1000,请分别给出数组元素 A[I,J],A[I+1,J+1]的地址。

解:一般而言,设 A 是下面说明语句所定义的一个 ALGOL n 维数组

$$\text{array} \quad A[l_1:u_1, l_2:u_2, \cdots, l_n:u_n]$$

令 $d_i = u_i - l_i + 1, i = 1, 2, \cdots, n$,a 为数组的首地址,即 $A[l_1, l_2, \cdots, l_n]$的地址。那么,在按行排列的前提下,数组元素 $A[i_1, i_2, \cdots, i_n]$的地址 D 为:

$$D=a+(i_1-l_1)d_2d_3\cdots d_n+(i_2-l_2)d_3d_4\cdots d_n+\cdots+(i_{n-1}-l_{n-1})d_n+(i_n-l_n)$$

本题目元素 A[I,J]的地址 $D_1$为

$$D_1=a+((I-(-4))*(3-(-3)+1)+(J-(-3)))*6=$$

$$1000+((I+4)*7+(J+3))*6=1186+42I+6J$$

元素 A[I+1,J+1]的地址 $D_2$为

$$D_2=a+(((I+1)+4)*7+((J+1)+3))*6=$$

$$1000+234+42I+6J=D_1+48$$

# 第 2 章　高级语言及其语法描述

本章的重点与难点是:上下文无关文法、判断一个文法为哪一类文法、推导、句型、句子及语言的定义。

本章需要读者掌握的概念有:推导与归约、句型、句子、最左推导、最右推导、上下文无关文法、语言、短语、直接短语、语法树、文法的二义性。

对于本章,读者还应该掌握:正确理解程序设计语言中语法和语义的含义,了解高级语言的一般特性。掌握上下文无关文法的判断和转化。乔姆斯基的文法分类,以及判断一个文法为哪一类文法的规则。

## 2.1　学 习 指 导

### 2.1.1　字母表

设 L 和 M 是两个符号串集合,则

① 合并:$L \cup M = \{s \mid s \in L$ 或 $s \in M\}$

② 连接:$LM = \{st \mid s \in L$ 且 $t \in M\}$

③ 方幂:$L^0 = \{\varepsilon\}$, $L^1 = L$,

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

④ 语言 L 的 Kleene 闭包,记作 $L^*$

$$L^* = \bigcup L^i (i >= 0) = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \cdots$$

⑤ 语言 L 的正闭包,记作 $L^+$ ($L^+ = LL^*$)

$$L^+ = \bigcup L^i (i >= 1) = L^1 \cup L^2 \cup L^3 \cup L^4 \cup \cdots$$

符号串是由字母表中的符号所组成的有限序列。字母表 A 的闭包是字母表 A 的各次方幂之并,记作 $A^* = A^0 \cup A^1 \cup A^2 \cdots$,其含义是由 A 上符号组成的任意符号串的集合(包括空符号串 $\varepsilon$)。

如果字母表 A 不包含空串 $\varepsilon$,则得到 A 的正闭包,用 $A^+$ 表示,可以看到,$A^+ = A^* - \{\varepsilon\}$,其含义是由 A 上的符号组成的所有符号串(不包括空串 $\varepsilon$)的集合。

如:$L = \{A \sim Z, a \sim z\}$, $D = \{0 \sim 9\}$,则

① $L \cup D = \{A \sim Z, a \sim z, 0 \sim 9\}$

② LD 是由所有由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 4 个字母构成的符号串集合。

④ $L^*$ 是所有长度任意的字母构成的符号串(包括 $\varepsilon$)的集合。

⑤ $L(L \cup D)^*$ 是由所有以字母打头的字母和数字组成的符号串集合。

⑥ $D^+$ 是由所有长度大于等于 1 的数字串所构成的集合。

语法分析的一个重要任务是:判断一个符号串的组成是否满足某个文法的规定。

### 2.1.2 程序语言的定义

任何语言实现基础是语言的定义。程序设计语言主要有语法和语义两方面定义。

任何程序语言都可以看成是一定字符集(称为字母表)上的字符串(有限序列)。但是什么样的字符串才算是一个合适的程序呢？一个语言的语法是指这样的一组规则,用它可以形成和产生一个合适的程序。这些规则一部分称为词法规则,另一部分称为语法规则(或产生规则)。注意这里提到3个概念:① 一个程序只是将一个有限字符集作为字母表;② 词法规则是指单词符号的形成规则。单词符号一般包括:基本字、标识符、各种类型的常数、运算符和界符等;③ 语言的语法规则规定了如何从单词符号形成更大的结构(即语法单位),换言之,语法规则是语法单位的形成规则。一般程序语言的语法单位有:表达式、语句、分程序、函数、过程和程序等。

对于一个语言来说,不仅要给出它的词法、语法规则,而且要定义它的单词符号和语法单位的意义。这就是语义问题。

语义是指这样的一组规则,使用它可以定义一个程序的意义。一般采用语法制导翻译方法。

一个程序语言的基本功能是描述数据和对数据进行运算。所谓程序,从本质上来说就是描述一定数据的处理过程。在学习编译原理的过程中应特别注意:程序语言的每个组成成分都有(抽象的)逻辑和计算机实现两方面的意义。当从数学上考虑每一个组成成分时,我们注重它的逻辑意义,当从计算机这个角度来看时,我们注重它在机内的表示和实现的可能性及效率。

### 2.1.3 上下文无关文法

程序语言语法的形式描述很好用,语义的形式描述不太好用,但它推动了语言理论的发展。

文法是描述语言的语法结构的形式规则。文法可以用来精确而无歧义地描述语言句子的构成方式,文法描述语言的时候不考虑语言的含义。一个文法G包含4个组成部分:一组终结符号,一组非终结符号,一个开始符号,以及一组产生式。

终结符号集合一般用$V_T$表示。终结符号是组成语言的基本符号。从词法分析的角度来看,终结符是组成程序语言中单词的不可再分的基本符号。如:a～z,0～9等。从语法分析的角度来看,终结符号是一个程序设计语言中不可再分的基本符号,即单词符号。如:基本字、标识符、常数、运算符、界符等。

非终结符号集合一般用$V_N$表示,用来代表语法范畴。从词法分析的角度来看,$V_N$表示基本字、标识符、常数、运算符、界符等这些单词。从语法分析的角度来看,$V_N$表示“算术表达式”、“布尔表达式”、“赋值语句”、“子程序”、“函数”等这些语法范畴。一个非终结符号代表一个一定的语法概念,是一个类(集合)记号,而非个体记号。

文法产生式规则的定义:一个产生式是一个有序对$(\alpha,\beta)$,通常写作$\alpha\rightarrow\beta$或$\alpha::=\beta$。其中$\alpha$为左部;$\beta$为右部。

对$\alpha$,$\beta$的限制:$\alpha\in(V_N\cup V_T)^+$且至少包含一个$V_N$;$\beta\in(V_N\cup V_T)^*$。

产生式意味着能将一个符号串用另外一个符号串替换。因而又被称为重写规则或规则。产生式可以简写,例如:$\alpha\rightarrow\beta$,$\alpha\rightarrow\gamma$可以简化为:$\alpha\rightarrow\beta|\gamma$。

乔姆斯基(Chomsky)文法的分类,根据对产生式的不同限制,可分为4类:0型,1型,2型,3型。我们主要用到2型和3型文法。

0型文法又称为短语结构文法(Phrase Structure Grammar,PSG),其产生式为:$\alpha\rightarrow\beta$,$\alpha\in(V_N\cup V_T)^+$且至少含有一个非终结符号,而$\beta\in(V_N\cup V_T)^*$。

0型文法能力相当于图灵机。相应语言称为0型语言,又称为递归可枚举集合。0型语言是不可判定的。

1型文法又称为上下文有关文法(Context Sensitive Grammar,CSG),其产生式为:

$\gamma_1A\gamma_2\rightarrow\gamma_1\delta\gamma_2$,其中$A\in V_N$,$\gamma_1$、$\gamma_2\in(V_N\cup V_T)^*$,$\delta\in(V_N\cup V_T)^+$。$\gamma_1$,$\gamma_2$称为上下文。

1型文法也可以定义为:所有规则的右部都比左部长。1型文法是可判定的,但是现在还没有找到有效的方法。

2型文法又称为上下文无关文法(Context Free Grammar,CFG),其产生式为:$A\rightarrow\delta$,其中$A\in V_N$,$\delta\in(V_N\cup V_T)^*$。

一般的程序设计语言的语法都使用2型文法描述。2型文法是可判定的,而且有有效的判定方法。

3型文法又称为正规文法(Regular Grammar,RG),分为右线性文法和左线性文法。

右线性文法的产生式为:$A\rightarrow\alpha$ 或者 $A\rightarrow\alpha B$,其中$A,B\in V_N$,$\alpha\in V_T^*$。

左线性文法的产生式为:$A\rightarrow\alpha$ 或者 $A\rightarrow B\alpha$,其中$A,B\in V_N$,$\alpha\in V_T^*$。

3型文法,其语言也称为正规语言。

不同文法能力的比较:0型文法强于1型,1型强于2型,2型强于3型。0型文法能力最强,相当于图灵机;1型文法能力相当于线性有界自动机;2型文法能力相当于非确定的下推自动机;3型文法能力相当于有限自动机。

在程序语言中,与词法有关的规则属于正规文法;与局部语法有关的规则属于上下文无关文法;而与全局语法和语义有关的部分往往要用上下文有关文法来描述。

给定一个文法,如何判断其属于何种文法?通常我们判断是按照3型文法、2型文法、1型文法、0型文法从高到低的顺序来判断。例如:一旦判断给定文法属于正规文法之后,就没必要再判断其是否属于上下文无关文法了,因为它必定属于上下文无关文法。我们应该以最高规则来判定一个文法属于的文法类型,其它情况依此类推。只有当我们判断不属于3型文法时,我们才向下判断,它是否属于2型文法,若不属于2型文法,则依此类推再向下判断。最终的结果如果不属于3型文法,2型文法以及1型文法三种类型文法,那就只有属于0型文法了。

所谓上下文无关文法是这样的一种文法,它所定义的语法范畴(或语法单位)完全独立于这种范畴可能出现的环境。上下文无关文法是文法的一种,一个上下文无关文法包括4个组成部分:一组终结符号,一组非终结符号,一个开始符号,以及一组产生式。开始符号是一个特殊的非终结符号,这个语法范畴通常称为"句子"。但在程序语言中我们最终感兴趣的是"程序"这个语法范畴,而其它的语法范畴都只不过是构造"程序"的一块块砖石。产生式(也称为产生规则或简称规则)是定义语法范畴的一种书写规则。一个产生式的形式是:$A\rightarrow\alpha$,其中箭头左边的A是一个非终结符号,称为产生式的左部符号;箭头右边的$\alpha$是终结符号或与非终结符号组成的符号串,称为产生式的右部。产生式是用来定义语法范畴的。

形式上定义一个上下文无关文法G是一个四元组$(V_T,V_N,S,\mathcal{P})$,其中:

$V_T$是一个非空有限集,它的每一个元素称为终结符号;