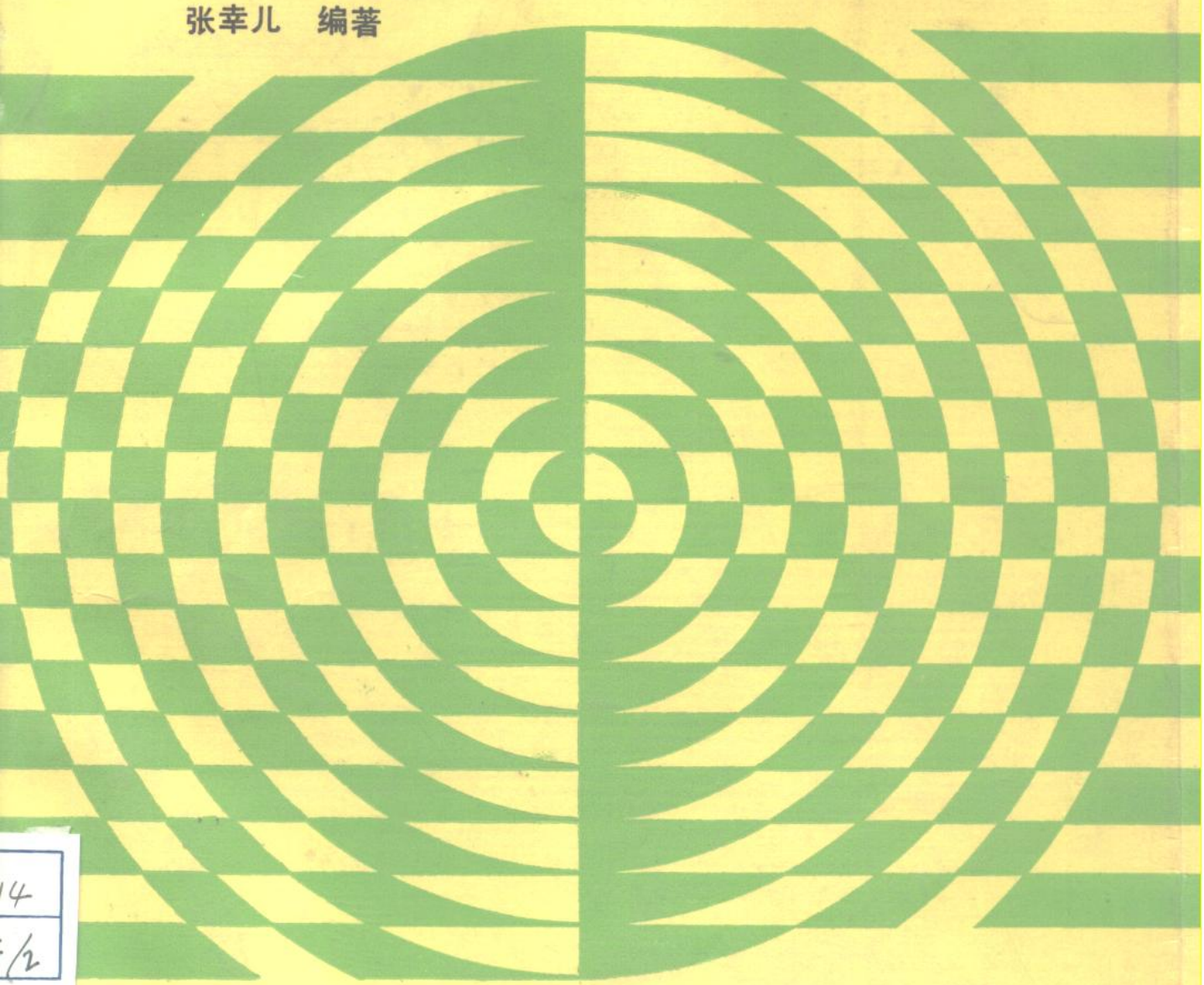
计算机编译理论

JISUANJI BIANYI LILUN

张幸儿 编著



南京大学出版社

内 容 简 介

形式语言在计算机科学的发展中起着十分重要的作用，形式语言理论已成为计算机科学的一个重要组成部分。本书系统地介绍了形式语言理论基础，并从形式语言理论角度讨论了程序设计语言及分析技术，使读者对程序设计语言及编译技术有较深刻的认识，也为进一步开展计算机软件工作打下良好的基础。

全书共分八章，内容包括引言、形式语言理论、扫描程序、自顶向下与自底向上分析技术，以及识别程序的自动构造，最后介绍识别程序书写语言。各章附有习题，书末有总复习思考题。

本书系统性强、条理清晰、文字简炼、深入浅出、通俗易懂、便于自学，概念阐述清晰而形象，特别方便读者对形式语言理论的理解与接受。书中的例题与习题之选择，旨在培养读者证明、计算与演绎等方面的能力。

本书可作为大专院校计算机专业编译理论与编译技术课程的教材或参考书，也可供广大计算机软件工作者、研究生及有关专业人员学习参考。

计算机编译理论
张幸儿 编著
责任编辑 顾其兵

南京大学出版社出版
（南京大学校内）
江苏省丹阳印刷厂印刷
江苏省新华书店发行 各地新华书店经销
开本：787×1092 1/16 印张：14.875 字数：371000
1987年12月第1版 1989年3月第3次印刷
印数：3001—9000
ISBN 7-305-00107-4
TP·8 定价：4.15元

序　　言

自1956年语言学家Chomsky首先描述形式语言以来，形式语言理论得到了迅速发展。形式语言在计算机科学的发展中起着十分重要的作用，形式语言理论已成为计算机科学的一个重要组成部分。形式语言理论不仅有着理论意义，而且有着现实的实践指导意义。

本书是在《编译理论》讲义基础上修改补充而成的。它系统地介绍了形式语言理论基础，并从形式语言理论角度讨论了程序设计语言及分析技术。它强调了各种分析技术各自的特征及相互区别，特别地强调了各种分析技术的应用范围、条件与局限性，给学生留下了深刻印象。它也因为下列特点而深得历届学生的欢迎：

1.系统性强，条理清楚，文字简炼，深入浅出，通俗易懂，便于自学。

2.结合作者的理解与体会，概念阐述清晰而形象，因而较易理解与接受，特别方便形式语言理论的学习。

3.结合一种程序设计语言进行讨论，对程序设计语言相关概念有更深入的理解与认识。

4.通过例题与习题，加强了在证明、计算与演绎等方面能力的培养。

它尤其得到准备参加硕士研究生入学考试的学生的欢迎。

《计算机编译理论》一书的产生背景是在编译技术基础上学习形式语言理论，从形式语言理论角度讨论与认识程序设计语言及分析技术，使读者产生从感性认识到理性认识的飞跃。这样做，使编译技术与形式语言理论学习中的难点分开，从而降低了学习难度，有利于读者较系统地学习掌握编译技术与形式语言。事实上，本书也适用于不熟悉编译技术的读者，第一章中的有关章节将向他们展示粗略的轮廓。期望通过本书的阅读，读者能较为深刻地认识与理解程序设计语言及分析技术的有关问题，能有助于指导程序设计语言的设计与编译程序的编写，也为进一步开展计算机软件工作打下良好的基础。

郑国梁同志仔细审阅了全书，徐永森同志仔细阅读了本书前面部分。他们提出了许多宝贵意见与建议。在此向他们表示衷心的感谢。限于作者水平，错误与不妥之处仍在所难免，欢迎读者批评指正。

作　者　1987年6月
于南京大学计算机科学系

目　录

第一章 引 言

§1 前 言

程序设计语言的引进，特别是诸如ALGOL60、FORTRAN和PASCAL等高级语言的引进，为电子计算机的推广应用创造了极为有利的条件，使广大科技工作者能较迅速而容易地掌握计算机的使用。为了能在计算机上执行用程序设计语言所写的程序，对编译程序的构造及编译方法早已进行了较为深入的研究，各种行之有效的编译技术广泛地被采用在各个编译系统中。

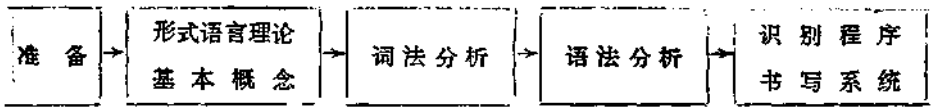
很可能，相当一部分读者学习过计算机编译方法，对程序设计语言及编译原理早已有一定的了解，自然地会提出这样一些问题：为什么要讨论编译理论？编译理论到底包括哪些内容？它与编译方法有什么联系？这是大家所关心的。这里作一简单说明。

如果我们回顾程序设计语言和编译方法课程的通常安排，不难发现，其侧重面都在于应用与实现。譬如，从程序设计语言课程，可以了解到ALGOL60、FORTRAN或PASCAL等语言成分的描述，学会源程序的书写与阅读，从而应用它们去解决某些应用领域中的实际问题；又从编译方法课程可以了解到基本编译原理，掌握某几种编译技术，从而在某型号计算机上为某种语言配备编译程序。但是如果要问“程序设计语言ALGOL60是什么”、“一个语言与用它写的程序的关系是什么”以及“如何产生一个语法上正确的程序”等问题时难免有些模糊的认识。感觉到了的东西，我们不能立刻理解它，只有理解了的东西才能更深刻地感觉它。感觉只解决现象问题。感性的认识有待于发展到理性认识。“编译理论”，顾名思义，是从理论的角度来讨论编译实现这一课题。更确切地说，从形式语言理论的角度来讨论和认识程序设计语言及其编译实现中的问题。重点是讨论形式语言理论与程序的自动识别问题。

什么是形式语言理论？粗略地说，形式语言是一种不考虑含义的符号语言，形式语言理论研究的是组成这种符号语言的符号串的集合、它们的表示法、结构及特性。

当应用形式语言理论来讨论编译理论时，将采用数学那样的符号表示形式，数学那样的严格推理方法来研究和探讨形式语言理论及其在编译实现中的应用。自1956年语言学家Chomsky首先描述形式语言以来，形式语言理论已得到迅速的发展，它已成为计算机科学的一个重要组成部分。它不仅有着理论意义，而且还有现实意义，对程序设计语言的设计与编译实现有极为深远的影响。

编译理论的主要内容可以归结为应用形式语言理论于编译实现。由于讨论是建立在不考虑含义的形式语言基础上的，这里的编译理论并不涉及编译实现中代码生成等与语义有关的方面。因此，形式语言理论这条主线贯串在词法分析与语法分析两个方面。全书的主要线索可图示如下：



准备阶段将简要回顾程序设计语言与编译原理的有关概念。识别程序书写系统主要讨论产生式语言的概念。中间三框构成全书的中心。

我们期望，本书能在下列几方面产生积极的影响：

1）能从形式语言理论角度去理解和认识程序设计语言和编译程序构造。了解应用各种编译技术时将会遇到的问题及其解决办法，指导编译程序的研制；

2）能有利于对编译程序自动生成的讨论；

3）能为进一步学习形式语言理论及其它有关课题打好基础，从而能应用形式语言理论各项技术于其它方面。例如，用于建立词法分析程序的有穷状态技术已被使用于正文编辑程序、文献目录检索系统和模式识别程序。上下文无关文法和面向语法的翻译方案则被使用于建立很多种类的正文处理程序（如数字排字系统）等。

概括之，期望读者在阅读全书之后，能了解和掌握形式语言理论的基本概念，从形式语言理论的角度去认识和考察编译中的问题，以便将来能应用形式语言理论指导程序设计语言的设计与编译实现。

§2 对程序设计语言及若干编译技术的回顾

2.1 程序设计语言的引进

电子计算机的特点是容量大、速度快、精度高，而且具有自动判别的功能，因此被广泛应用于各个应用领域。鉴于机器语言程序与具体计算机密切相关，不仅难学难写难理解，也难于查错改错，用户往往得花相当多精力去学习特定机器的特性（包括指令系统）等，而不能更多地去考虑解题算法。高级程序设计语言的引进，使得人们能用接近于日常数学用语的表示法去表达算法，从而为计算机的推广应用打开了局面。一般地，针对不同的应用领域可以选取最为合适的高级程序设计语言。

且看下列ALGOL60程序：

```
BEGIN
    INTEGER  x,y,t;
    READ  x,y;
    t:=x; x:=y; y:=t;
    WRITE  x,y
END
```

ALGOL60程序的结构成分按层次结构由小到大可表示如下：

基本符号——符号（单词）——量——表达式——语句与说明——分程序——程序

为了表明构造各个组成成分，特别是程序的规则，采用BNF表示法，例如，

〈程序〉::=〈分程序〉|〈复合语句〉

〈分程序〉::=〈分程序首部〉；〈复合尾部〉

〈分程序首部〉::=BEGIN〈说明〉|〈分程序首部〉；〈说明〉

〈复合语句〉::=BEGIN〈复合尾部〉

〈复合尾部〉::=〈语句〉END|〈语句〉；〈复合尾部〉

⋮

〈标识符〉::=〈字母〉|〈标识符〉〈字母〉|〈标识符〉〈数字〉

这些规则描述了ALGOL60程序是如何构造的。其中〈程序〉、〈分程序〉等是语法实体或称非终结符号，::=与|是元语言连接符，它们连同BEGIN等终结符号组成元语言公式。

ALGOL60有四大特点，即局部性、递归性、动态性与严谨性，由于它采用BNF表示法，对程序设计语言的研究与发展产生了重大影响，因而引起广泛的重视。然而它也有一大缺陷。例如，对于规则

〈左部〉::=〈变量〉:=|〈过程标识符〉:=

必须确定何时应用后一选择，又当选取后一选择时还必须确定是函数过程标识符而不是一般过程标识符。又如对于规则

〈左部表〉::=〈左部〉|〈左部表〉〈左部〉

〈赋值语句〉::=〈左部表〉〈算术表达式〉

出现在同一赋值语句中的各个左部变量的类型应该一致，这一要求不能由这规则本身反映出来，必须另行说明。

这一缺陷的根源在于语义与全程语法的非形式化。这是一般程序设计语言的通病。语义形式化问题远比语法形式化问题更为复杂，由于超出本书范围，在此不予讨论。

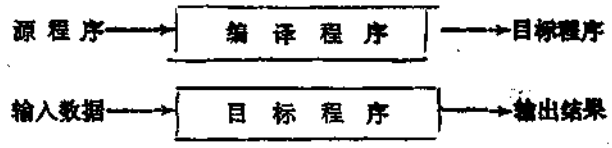
2.2 编译程序的构造

2.2.1 源程序的执行

计算机只能理解和执行机器语言程序，不能直接执行高级语言程序。后者的执行通常采用两种方式，即解释执行方式与翻译执行方式。

解释执行借助于一个解释程序来完成。这时解释程序不是把程序变换成等价的机器语言程序，而是对程序逐个语句进行分析，根据每个语句的含义模拟执行。这种执行方式效率较低，适用于配备有分时操作系统的计算机系统。除某些特殊情况外，一般采用翻译执行方式。

翻译执行借助于一个翻译程序来完成。翻译程序对整个程序进行分析，把它变换成（翻译成）等价的机器语言（或汇编语言）程序，然后执行之。所谓等价，即两者的执行效果完全相同。用某种程序设计语言所写的程序称源程序，写源程序的语言称源语言，而等价的低级语言程序称目标程序或结果程序，相应的低级语言称目标语言。当源语言是汇编语言时，翻译程序称汇编程序，它把汇编语言程序翻译成等价的机器语言程序；当源语言是ALGOL60、FORTRAN或PASCAL等其它高级语言时，翻译程序又称编译程序。源程序编译执行的示意图如下所示。



2.2.2 编译程序的结构

如上所述，编译程序的功能是把高级语言源程序翻译成等价的目标程序。源程序是由基本符号串组成的。各基本符号本身并不一定具有独立的含义（如字母、数字），因此编译程序首先得识别开各个有意义的最小语法单位，即单词或符号——标识符与界限符等；然后根据语法规则分析各语法结构，识别出是什么成份，检查其正确性；同时按各语法结构的含义生成相应的目标指令。这三项工作分别称作词法分析、语法分析与语义分析。在词法分析与语法分析时通常产生一些信息表，如常量表、符号表（标识符表）与循环语句表等。语义分析程序则包括存储分配与代码生成等。编译程序工作的示意图如图1-1所示。

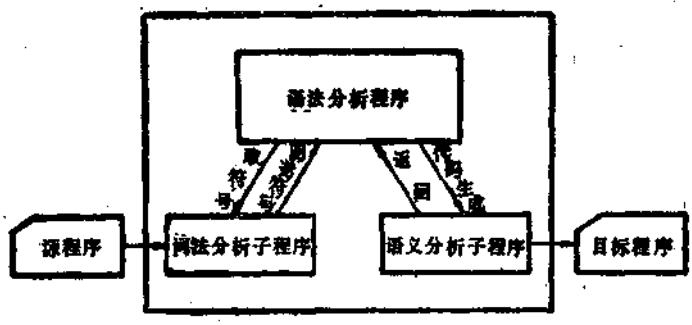


图1-1

词法分析程序又称扫描程序，它从左到右地扫描源程序中各个字符，识别开各个符号，并把它们转换成通常是等长的内部形式以供语法分析程序使用，往往还包括了在语法分析之前需做的各项简单工作，如删除注解、把标识符放入符号表中及加工宏功能等。

语法分析工作与语义分析工作通常在一个程序内实现，即同一个程序在识别语法结构、进行语法检查的同时进行语义检查、构造相应的内部形式并生成目标指令。

编译仅进行一次，但目标程序往往可能执行多次，一般说，编译时间与运行时间相比总是只占总时间的很小一部分，因此宁愿让编译程序多做些工作以得到一个质量较高的目标程序，也就是说，先让源程序经过词法分析与语法分析得到内部表示形式的等价程序，对此进行优化，然后生成目标代码。因此编译程序可有如下的结构（见图1-2）。

一个编译程序一般地是结构复杂、需要相当内存量的较大程序系统，因此有时称编译系统。考虑到下面几个因素：

1）可用内存容量的大小；

2）设计目标，如编译速度、目标程序执行速度、查错改错能力及调试功能等；

3）参加研制人数、人员能力及完成期限等。

往往把编译程序的工作分成若干阶段来完成，每一阶段的输出结果作为下一阶段的输入，这样的每一阶段称一遍（或一趟）。实际上，图1-1与图1-2可分别看作一遍与四遍的编译程序。

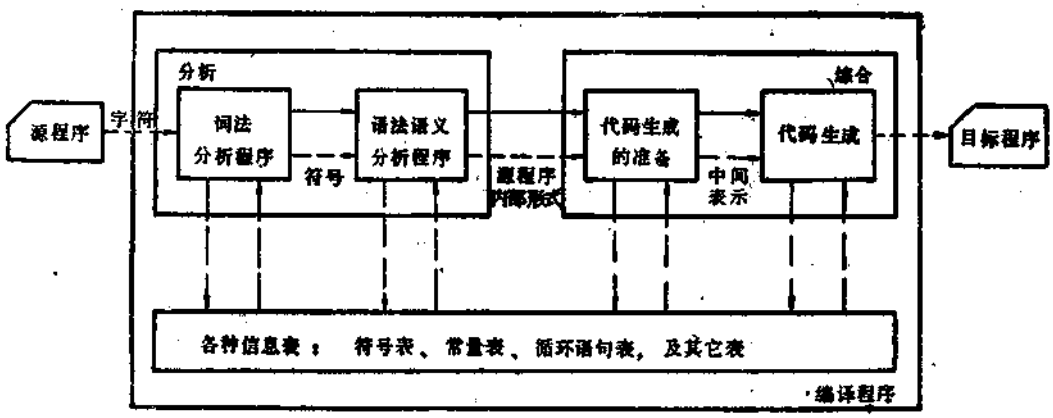


图1-2

2.3 基本编译技术

这里简单回顾在编译方法课程中所能了解的几种基本编译技术。

2.3.1 递归子程序方法

这是最普遍地应用于编译实现中的方法之一。递归子程序方法的思想概而言之是：对照语法规则，为语言的各个语法成分编制一个处理该语法成分的子程序，特别地为递归定义的语法成分设计递归子程序。当语法分析时，从处理识别符号〈程序〉的相应子程序开始，对源程序进行语法语义分析，直到整个源程序处理完。

何谓递归定义？如果一个概念或函数是通过它自身来定义的，也即，在一个概念或函数的定义中又出现该概念或函数本身，则称该概念或函数是递归定义的。例如：

〈算术表达式〉::=〈简单算术表达式〉|

〈如果子句〉〈简单算术表达式〉ELSE〈算术表达式〉

这里〈算术表达式〉是递归定义的。递归子程序则是这样的子程序，即进入它之后，在返回到调用它的程序之前，又可能直接或间接地进入它。例如，对于如上定义的算术表达式可以构造一个递归子程序，其流程图如图1-3所示。

该子程序的结构流程图几乎就是关于算术表达式的语法规则的图解表示。因此这种方法简单直观，易于掌握。

2.3.2 自底向上的直接语法分析方法

按这种方法，将自最左的符号开始，把源程序的符号(往往还包括替换所得的非终结符号)与语法规则的右部直接作比较，当找出与语法规则右部匹配的部分时，用该规则左部的符号代替之，然后重复这一比较过程，直到整个源程序被换为识别符号〈程序〉。具体实现时，按规则排序，给每一符号以一个层号，识别符号的层号最高，终结符号的层号最低（为0）；通过层号的比较来控制翻译流程。

这里给出一个例子。

假定一语言的语法规则及其排序如下：

1.〈I〉::=A|B|C|D|E

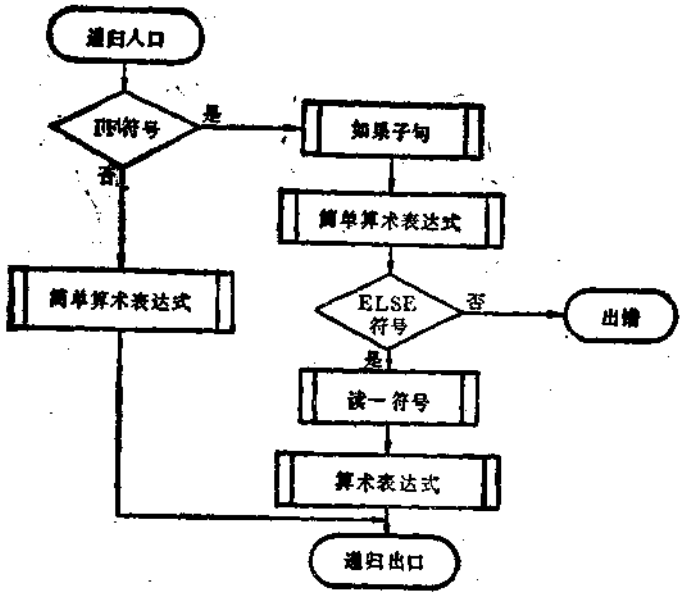


图 1-8

2.〈M〉::= * | /

3.〈R〉::=)

4.〈A〉::= + | -

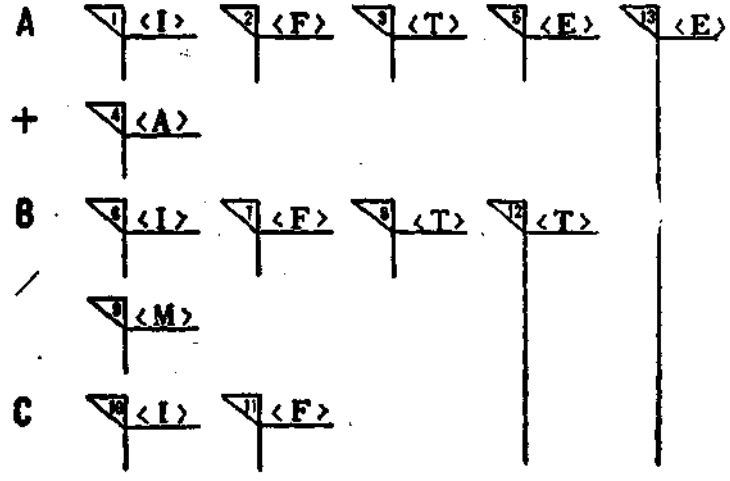
5.〈L〉::= (

6.〈F〉::=〈I〉 | 〈L〉〈E〉〈R〉

7.〈T〉::=〈F〉 | 〈T〉〈M〉〈F〉

8.〈E〉::=〈T〉 | 〈E〉〈A〉〈T〉

各个序号是相应规则左部非终结符号的层号。当输入是 A+B/C 时，将有如下所示的翻译流程。由于我们对语义不感兴趣，省略了所生成的代码部分。



这种方法的优点在于：编译程序可适用于多种程序设计语言，不因语言的语法规则或语

义描述的更改而变动。但这种方法与规则的排序有很大关系，而语法规则的排序与语义描述又都比较麻烦，特别是编译程序所需的内存容量较大，编译速度较慢，而目标质量又较差，因此未能得到普遍推广应用。

2.3.3 运算符优先数法

“先乘除后加减”这一简单的算术运算法则启发我们无论参加运算的是什么，运算符便可以决定运算的次序。例如，A+B*C 相当于 A+(B*C) 而不是(A+B)*C。运算符优先数法可以看成是这种思想的扩充。

按这种方法将对各个运算符确定一个唯一的优先数。当语法分析时，首先藉助于优先数把源程序变换成逆波兰表示(即后缀表示法)的符号行，称逆波兰表达式；然后再处理逆波兰表达式，将它变换成相应的符号指令序列。逆波兰表示的形成过程中利用了一个运算符栈，可形象地用铁路转轨来刻划，如图 1-4 所示。我们看简单算术表达式的情况。分析表达式时，每当遇到运算分量就输出，遇到运算符则把当前运算符的优先数与运算符栈顶运算符的优先数相比较。当前者小时表明可以对栈顶运算符进行运算，因而将栈顶运算符送出到输出区并上退栈；否则把当前运算符下推进栈。不难得到 a+b*c 的逆波兰表示是 abc*+。

为了处理逆波兰表达式，只需设置一个运算分量栈来存放暂时不能处理的运算分量的名字以及中间运算结果的名字。更确切地说，扫视逆波兰表达式时，每当遇到运算分量便把它下推入栈，每当遇到运算符时就根据它是多少目的运算符，对运算分量栈中那样多个运算分量与此运算符生成目标指令，并把它们从栈中退去，把存放运算结果的累加器名字 AC 下推入栈。必要时生成把累加器内容送入某中间变量的指令。如此重复到处理完整个逆波兰表达式。

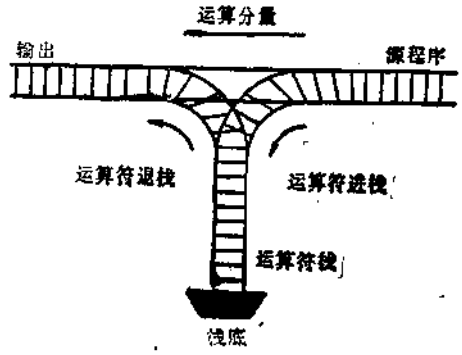


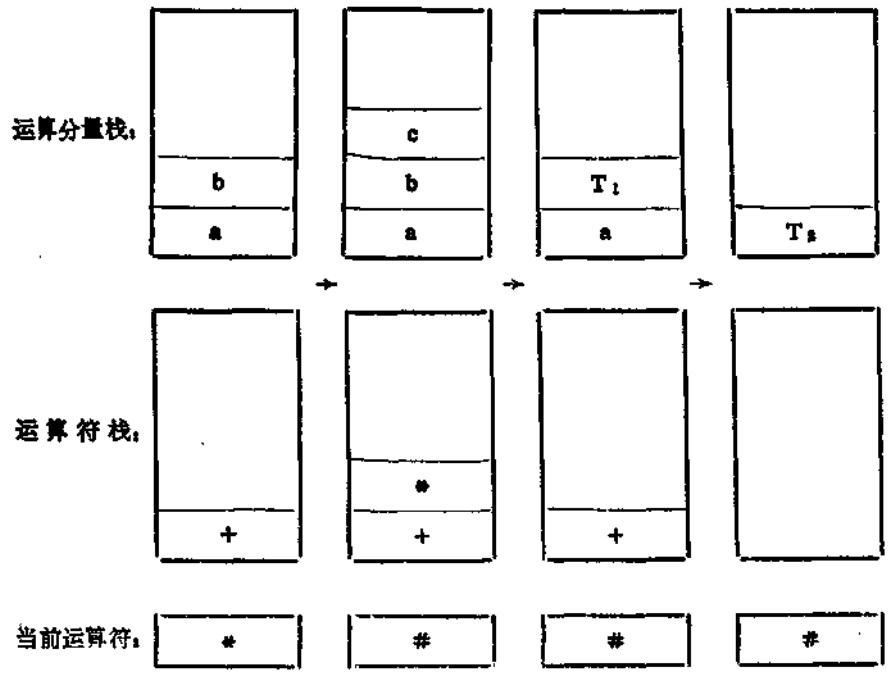
图 1-4

运算符优先数法的两步往往利用运算分量栈而合并成一步。这时除了有运算符栈来保存还不能生成目标的运算符外，还有运算分量栈来保存还不能生成目标指令因而还不能输出的运算分量。仍以 a+b*c 为例，分析过程中栈的变化如图 1-5 所示。

运算符优先数法由于它的简单直观，所需存储容量小，且速度快而被广泛应用。当我们把 BEGIN、END 与 IF 之类界限符也看作运算符(称作广义运算符)，给它们优先数，则运算符优先数法可扩充到整个语言的处理。

2.3.4 状态矩阵法

状态矩阵法是藉助于状态矩阵对源程序进行语法语义分析的方法。这种状态矩阵是关于某种源语言而预先构造好的一个二维数组，一维对应于状态，所谓状态是一种刻划语言成分或语法成分之间关系的记号；另一维对应于各个源语言符号。状态矩阵的元素指明了状态与符号的匹配关系是否正确；当正确时，指明应做的语法语义工作。当用状态矩阵法分析源程序时，设置一个状态栈，每当进入一个语法成分，便把相应的状态下推入栈，用以记录语法分析过程中当前正处于分析什么语法成分；每当退出某一语法成分时，则把相应的状态从栈



其中　$T_1=b*c$，$T_2=a+T_1$，#为输入结束标志

图1-5

中退去。因此状态栈的动态内容刻划了源程序语法结构分析过程中的动态变化。由于状态栈栈顶状态与当前读入的符号决定了新进入的语法结构的状态及要做的语法语义工作，状态矩阵对控制分析过程起着关键的作用。

这方法便于查出源程序中的错误，但一个重大缺点是对语法作一点修改就将对状态矩阵产生很大影响。尽管如此，这方法的应用仍然十分广泛。

2.3.5 LR(k)与SLR(k)方法

当采用某种方法来分析一个源程序时，并不是总能严格地从左到右地扫描的，换句话说，存在有一些情况，在从左到右地扫描源程序时并不能根据当前正读入的符号确定是继续向前读还是进行某项工作，而是必须回顾先前已扫描部分的信息，因而忽左忽右地查看源程序。

LR(k)方法是一种严格地从左到右扫描源程序进行分析的方法。采用LR(k)方法对源程序分析时每一步的动作由已扫描过的全部符号及向前看的k个符号所唯一地确定。这时，编译程序将利用一个后进先出栈和一个分析表。栈中存放状态，每个状态刻划了当前正处的语法地位及预期的向前看符号的信息；因此栈的内容记录了整个分析过程的历史以及当前的展望信息，无需回顾已扫描部分便可由实际的向前看符号确定所应执行的动作。这个动作由当前栈顶状态与实际向前看的k个符号所确定的分析表元素指明。

SLR(k)方法是简单的LR(k)方法之简称，它给出了LR(k)分析表的一种特殊构造法，使得分析过程中无需每次都固定地向前看k个符号，而是在一般情况下不向前看任何符号，

仅根据当前被扫描符号便可确定应执行的动作，在不能确定时也只是最多向前看 k 个符号。例如，对于 k = 1 的情况，分析过程中每步最多向前看 1 个符号便能确定应执行的动作。

一般的程序设计语言，例如 ALGOL60 等都可满足 SLR(1) 方法的要求。SLR(k) 方法的提出简化了 LR(k) 分析表的构造，使 LR(k) 方法能付诸实际应用。一般地 LR(k) 有较好的功效，且适合于自动生成相应的分析程序，因此引起人们的兴趣。关于 LR(k) 方法，本书将在第七章“识别程序的自动生成”中作进一步讨论。

至此，我们简单回顾了五种基本编译技术，它们除了自底向上的直接语法分析方法外都是应用较为普遍的典型技术。今后将从形式语言理论的角度去讨论编译技术，从而使读者能够对它们有更进一步的认识。

鉴于 ALGOL60 语言语法定义的形式化与严谨性，及其在程序设计语言理论的研究与发展中的作用，本书较多地结合 ALGOL60 语言进行讨论，这并不失一般性。另一方面，为了便于读者阅读，算法程序等将尽可能使用 PASCAL 型语言书写。

第二章 文法与语言

§1 语言概况

在讨论形式语言理论及其在编译实现中的应用之前，先一般地了解语言的一些有关基本概念是必要的。事实上，形式语言理论正是在对自然语言的研究中发展起来的，当然有着内在的联系，而程序设计语言同样有着一般语言的共同特征。

1.1 语言是什么

语言是人们在社会活动中交际的工具，也就是说，语言是人们在社会活动中相互之间表达与交流思想的工具，或者说是通讯的工具。如果我们把讲话和文字看作是语言的声与形两种不同表达形式，就无需考虑这两者间的区别。这对计算机的各种程序设计语言同样是正确的，所不同的是，它不仅作为人之间的通讯工具，而且作为人与计算机之间的通讯工具。由于有计算机语言的存在，人们之间不仅可以相互交流算法，让大家共享个人的劳动成果，而且可以让计算机理解人们要做什么，并按照人们的规定而进行工作。

显然仅了解“语言是一种工具”还远远不够，需要更确切地了解语言是什么，它是由什么组成的。

人们在交谈时讲的是一句句的话，当在书写时写的是一篇篇的文章。对于程序设计语言则是一个个的程序，程序是最终要达到的目标。

引进程序设计语言的目的就是用来写各种各样的程序以描述各种不同的计算过程。可以说 ALGOL60 语言是一切 ALGOL60 程序所组成的集合。明显的是，程序及其它语法概念本身，例如分程序与过程说明等等并不出现在程序中，只是在讨论与分析程序的结构时才引进。从字面上看，每个程序都是一个基本符号串。因此概括地说，ALGOL60 语言(或其它各种语言)是基本符号集上定义的、按一定规则构成的一切基本符号串组成的集合。一般地给出如下的定义。

定义2.1 一个语言是在某个特定字母表上定义的、按一定的规则构成的符号串(或字符串)的集合。

请注意，首先，只有特定字母表上的符号(或字符)才被允许。对于汉语，字母表中包括汉字、数字及标点符号等；对于 ALGOL 60，字母表是基本符号集，其中包括字母、数字、一些保留字及某些专用符号。其次，按一定规则构成的符号串才属于相应语言，不符合规则地构成的符号串不被认为属于相应语言。

1.2 语言的要点

从上面对语言定义的讨论可以看到，为构成一个语言必须具备如下三个要点：