编译程序设计理论

P. M. 刘易斯 II　D. J. 罗森克兰茨　R. E. 斯特恩斯 著

科学出版社

编译程序设计理论

P. M. 刘易斯Ⅱ
D. J. 罗森克兰茨 著
R. E. 斯特恩斯

张文典 徐树荣 黄贵清 译

科 学 出 版 社

1 9 8 4

内 容 简 介

本书较系统地介绍了编译系统设计中的基本理论和方法，是一本专业教材。

全书共分十五章，其中包括：有限状态自动机；下推机；上下文无关文法；翻译和属性翻译；自顶向下处理方法；自底向上处理方法；代码生成程序和代码优化等。

本书理论和实践并重，叙述严谨、简炼，富有启发性，图例练习较多，易于读者学习。

本书可作为高校计算机软件专业学生或研究生的教材，也可供计算机软件工作者参考。

P. M. Lewis Ⅱ, D. J. Rosenkrantz, R. E. Stearns
COMPILER DESIGN THEORY
Addison-Wesley, 1976

编译程序设计理论

P. M. 刘易斯Ⅱ
D. J. 罗森克兰茨 著
R. E. 斯特恩斯

张文典 徐树荣 黄贵清 译
责任编辑 杨家福 那莉莉

科学出版社出版
北京朝阳门内大街 137 号
中国科学院印刷厂印刷
新华书店北京发行所发行 各地新华书店经售

*

1984年5月第一版 开本：850×1168 1/32
1984年5月第一次印刷 印张：21
印数：0001—10,000 字数：551,000

统一书号：15031·560
本社书号：3505·15—8

定价：3.90元

序　言

本书拟作为四年级大学生或一年级研究生学习编译程序设计课程的教材，可学一至两个学期．本书涉及到构成编译程序设计和语言处理程序基础的基本数学理论，并指出在实际设计中如何运用这些理论．

这种可应用的数学概念来自自动机和形式语言理论．我们采用了精确的而非形式化的风格来研究这些概念，以便广大读者，包括那些非数学专业的读者，都能理解它们．我们认为，自动机和形式语言的概念构成了教授编译程序设计和设计实际的编译程序的极好基础．根据这个理论，我们自己已设计了两个商业性的编译程序．

在我们选择和介绍的材料中，强调了“翻译”，而不是“分析”．语法制导的属性翻译形式概念用来说明各种语言处理程序的输入输出性能．

我们强调的另一个概念是“自动机”．本书中使用有限状态自动机和下推机这样的自动机作为编译程序的基本标准块．我们强调为实现特定的翻译而设计的自动机的各个综合过程．

本书的内容基本上构成了编译程序词法和语法部分的完整设计理论．使用属性翻译可使许多被描述为“代码生成”或“语义”的东西放到语法框设计中去．本书还包括有关代码生成的附加材料和代码优化的简要评述．

本书没有讨论运行实现这一课题．虽然在决定编译程序可生成什么样的代码时，这个课题是十分重要的，但是我们认为它不属于“编译程序设计理论”的内容，而是专门讨论程序设计语言结构的教程中单独处理的课题．

因为本书已选编了与编译程序设计有关的自动机理论，所以

某些基本的自动机理论概念就被省略了。因此，本书不能作为自动机理论课的专用课本。然而，采用本书作为教材的学生将发现，学了本书的前部分，接下去的自动机理论课程就显得很简单了，而基本自动机理论方面的某些训练可使学生更快地掌握本书的内容。所以，无论怎样，本书均可用作自动机理论的初级教科书。

编译程序设计理论是完全自成体系的，它只要求读者熟悉程序语言和大学三、四年级通常具有的数学知识。

在引论之后，第二、三、四章涉及到有限状态自动机和其它与词法处理有关的课题。第五章和第六章介绍下推机器和上下文无关文法。

如果学生已经掌握了自动机理论初级教程的内容，那么就可以略去第二章至第六章的大部分材料，并会很快通过剩余的强调自动机理论对编译程序应用的部分，但不论怎样，都可以免学2.7节到2.11节的内容。

第七章介绍翻译和属性翻译的思想；在继续往下学习之前应当掌握这部分材料。

第八、九、十章涉及到自顶向下的处理方法，而第十一、十二、十三章研究自底向上的处理方法。本书这些部分的内容都是独立的，教师可以有选择地介绍这些内容，但在任何情况下都可以省去第8.7，10.5，12.6和13.6节。

为了用实例说明在“实际”设计中的理论概念，本书讲述了为BASIC子集进行的编译程序设计。为了阐述本书所介绍的概念以及有一个普通的运行实现，我们所选择的语言已经很复杂了（正如我们所说的，语言特点的实现是个独立的课题）。这种语言具有各种语法和语义的特点，包括语法的递归控制结构（FOR循环）。在第四章，我们为编译程序设计了词法框；在第十章和第十二章，我们分别设计了自顶向下和自底向上操作的语法框；在第十四章，我们设计了代码生成程序。这种设计或它的某些扩充可在课程的实习阶段完成。

第十五章包含一个代码优化的简要评述。

本书还包含了三个附录：附录A是MINI-BASIC语言手册；附录B讨论了各种检验和设计过程所需要的一些数学关系；附录C介绍几种翻译方法，这些方法把给定的程序设计语言文法翻译成本书所介绍的几种特殊的形式文法之一。

本书作为一年级研究生一学期的课程在纽约特洛伊的伦塞勒工学院和奥尔巴尼的纽约州立大学教了几年，并且还作为大学生一个学期的选修课程，在纽约申内克塔迪的联邦大学使用过。我们感谢这些学院的学生，他们在学习本书时所偶然表现出的畏难情绪促使我们对其内容作了几次修改。

我们要向早期阅读过此译本并发表过有益评论的下述人员表示感谢：John Hutchison, Michael Hammer, Stephen Morse, John Johnston, Donna Phillips, Daniel Berry, Alyce Orne, Gary Fisher, Walter Stone, James Roberts, and Robert Blean.

我们还要感谢通用电器研究和发展中心的管理部门，特别要感谢L. Shuey和James L. Lawson，他们为我们撰写此书提供了方便，创造了良好的环境。

P. M. L.
D. J. R.
R. E. S.
纽约，申内克塔迪
1975年11月

目　录

第一章 引 论

1.1 语言处理程序

人机之间存在着一种天然的通讯空隙。计算机硬件依据二进制和寄存器在很低的水平上进行操作，而人们却倾向于用象英语这样的自然语言或数学符号来表达自己的意图。人机之间的通讯空隙通常用人工语言作为联系的桥梁。这种人工语言使得人们可用意思明确的并能被计算机“理解”的一组词、句子和公式表达他们的意图。为了实现这种通讯，需要给使用者提供说明这种语言所允许的结构和意义的用户手册，并为计算机提供一种软件，以便通过它，计算机可得到人们用这种语言书写的表示命令或程序的比特流，并将这个输入转换为实现人的意图所需要的内部位组合格式。

现有的计算机语言在复杂性方面变化得非常广泛，例如：

特定计算机的指令系统，也就是机器语言，它由机器本身的硬件或微程序来解释；

汇编语言，这种“低级”语言充分反映了特定计算机的指令系统；

控制卡和命令语言，它用于操作系统的通讯；“高级”语言，如 FORTRAN，PL/Ⅰ，LISP 等。它们具有复杂的结构并且不依赖于任何特定机器的指令系统和操作系统。

我们用“语言处理程序”这一术语来描述使计算机能理解人们提供的命令和输入的程序。概括地说，这样的语言处理程序有两种类型：解释程序和翻译程序。

*解释程序*是这样的一种程序，它接受的输入是用称为*源语言*的计算机语言编写的程序，并执行该程序蕴含的计算。

*翻译程序*是这样的一种程序，它接受的输入是用*源语言*写的程序，产生的输出是用一种称为*目标语言*的语言编写的程序。通常，目标语言是某个计算机的机器语言，在这种情况下，程序可在那台计算机上立即执行。把翻译程序分成汇编程序和编译程序多少带有些随意性，它们分别翻译低级语言和高级语言。

所有语言加工的共同数学基础是自动机和形式语言理论。因为本书主要关心编译程序的设计，所以我们只介绍该理论中与编译程序设计关系最密切的那些部分，并指出应用这种数学知识的一些实际方法。尽管这种理论是从编译程序的角度提出来的，但是在任何语言加工程序的设计中都可以使用它。

1.2 朴素的编译程序模型

编译程序的工作是把用某种源语言[1]写的程序的位组合格式翻译成机器指令序列，该序列体现了程序员的意图。这个任务是相当复杂的，其复杂性在于把编译程序作为单一实体来理解或设计是既困难又麻烦的。因此，希望把编译过程看作是一些能较容易地描述的较小过程的相互联接。

对任何特定的编译程序，子过程的选择可能依赖于正被处理的语言细节。无论在何种情况下，当采用有效的设计理论时，就能较好地进行这种选择。因此我们不必考虑一个子过程的特殊组合。另一方面，如果缺乏有关编译程序内部组织的某些概念，而要描述编译程序设计理论是不可能的。因此，为了建立一个参考框架，我们介绍一个略显朴素但很有效的模型。

在这个模型中，编译的工作由顺序联接的三个逻辑框完成。分别称这三个逻辑框为词法框、语法框和代码生成程序。它们可访问一组公共表，表中可以登记程序的长期信息或全局信息。例

1）原文为计算机语言。——译者注

如，其中的一个表是符号表。在这个表中，存放着有关每一个变量或标识符的信息。这些逻辑框和表的联接如图 1.1 所示。下面我们详细叙述这些逻辑框。

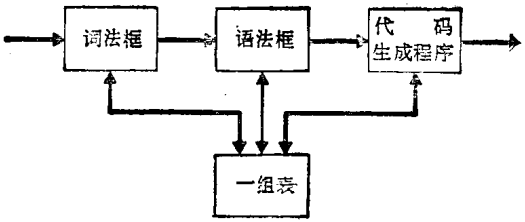

图 1.1

词法框

编译程序的输入表示字符串的位组合格式。词法框是把字符串分解成它们所表示的若干个词。例如，字符串可以是

IF $B1$=13 GOTO 4

词法框可识别出这个字符串表示词 IF，后面跟随变量 $B1$，接着是等号运算符，数 13，词 GOTO 和标号 4。这样，就把 12 个输入字符转换成 6 个新的实体。通常把这些实体称为记号，这就是我们使用的名称。

每一个记号由两部分组成，即类型部分和值部分。类型部分指出记号属于哪种类型，并指明值部分中包含的信息特性。再回到我们的例子。变量 $B1$ 可能是“变量”类型，并且有一个值，这个值是指向符号表项 $B1$ 的指针。实际上，这个符号表指针是变量 $B1$ 的内部名字。记号 13 可能是“常量”类型，其值可以是 13 的位组合格式。等号可能是“运算符”类型，其值可指出是哪一种运算符。记号 IF 可能是“IF”类型，并且不需要值信息。

如果我们把符号表看成是词典的话，那么，词法处理就类似于把字母组成词，并找出它们在词典里的位置。因此，将此逻辑框命名为“词法”是合适的。

语法框

语法框涉及到把词法框所构造的记号序列翻译成另外一种序列，该序列更加精细地反映了程序员想要在程序中实现的操作顺序。例如，如果FORTRAN程序员写了

$$A+B*C$$

他企图把B和C代表的数相乘，再把A代表的数加到那个乘积里。对上面的表达式，一种合适的翻译应该是

MULT（B，C，$R1$）ADD（A，$R1$，$R2$）

这里，MULT（B，C，$R1$）被解释为“B与C相乘，其结果称作$R1$”；ADD（A，$R1$，$R2$）被解释为“A与$R1$相加，其结果称为$R2$”。于是，由词法框提供的五个记号已经转换成表示相同意图的两个新的实体。这些新的实体叫作原子，并且形成了语法框的输出。重要的是原子序列反映了要执行的顺序。程序员把加法运算符写在乘法运算符的前面，但语法框必须把乘法运算放到前面。

我们再次假定每个原子也由类型部分和值部分组成。那么，MULT（B，C，$R1$）可能是“MULT”类型，并且有一个值部分，它由分别指向表项B，C，$R1$的三个指针组成。在编译程序中，这个原子将用一个整数表示“MULT”，用三个指针表示值部分。

为了实现这种转换，语法框必须按某种方法考虑语言的结构。这种方法类似于进行自然语言翻译时对语法的考虑。可以把样品表达式的翻译比作把英语（动词通常出现在句子中间）翻译成德语（动词常常出现在句子末尾）。因此，将此逻辑框命名为“语法框”是恰当的。

代码生成程序

代码生成程序涉及到把语法框构成的原子扩展为能实现相同意图的计算机指令序列。扩展的精确度依赖于代码实际执行时的