计算机系列教材

编译原理与技术（第2版）学习指导与习题解析

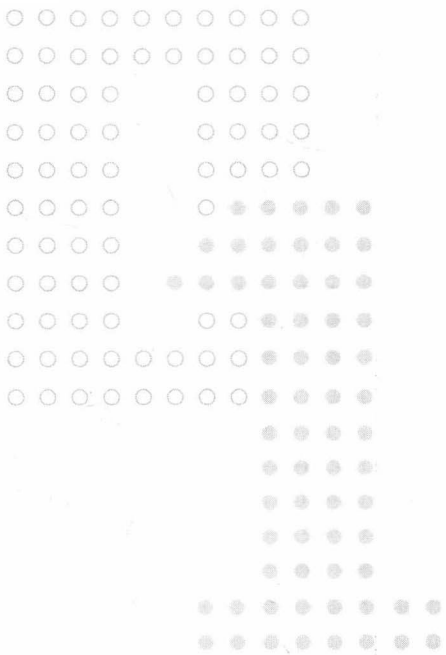
李文生 编著

清华大学出版社

计算机系列教材

李文生 编著

编译原理与技术（第2版）学习指导与习题解析



清华大学出版社
北京

内 容 简 介

本书是《编译原理与技术(第2版)》的配套参考书。作者结合多年讲授“编译原理与技术”课程的经验和体会以及在教学过程中遇到的实际问题,对主教材设计的习题进行了详细的解析。本书前10章名称与主教材保持一致,为使用主教材进行教学提供了支撑,同时对主教材中部分习题的描述进行了修正,使读者可以单独使用本书进行学习。第11章提供了两套模拟试卷和参考答案。

本书保持了主教材理论与实践相结合的特点。针对基础理论方面的习题,给出了所涉及的知识要点和理论,分析了解题思路并给出了参考答案;针对编程实践类习题,给出了程序设计思路和要点,对于难度较大的编程题,还给出了程序伪码说明。

本书的习题覆盖面广,难易结合,灵活性强,对学习“编译原理与技术”课程很有帮助。本书既可作为高等学校计算机类相关专业本科生和其他IT领域工程技术人员学习“编译原理与技术”课程的参考书,也可供报考研究生的读者使用。

图书在版编目(CIP)数据

编译原理与技术(第2版)学习指导与习题解析/李文生编著. —北京:清华大学出版社,2018
(计算机系列教材)
ISBN 978-7-302-49580-2

Ⅰ. ①编… Ⅱ. ①李… Ⅲ. ①编译程序—程序设计—高等学校—教学参考资料 Ⅳ. ①TP314

中国版本图书馆CIP数据核字(2018)第027514号

责任编辑:张瑞庆　战晓雷
封面设计:常雪影
责任校对:梁　毅
责任印制:刘海龙

出版发行:清华大学出版社
网　　址:http://www.tup.com.cn, http://www.wqbook.com
地　　址:北京清华大学学研大厦A座　　**邮　　编**:100084
社 总 机:010-62770175　　**邮　　购**:010-62786544
投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈:010-62772015, zhiliang@tup.tsinghua.edu.cn
课件下载:http://www.tup.com.cn,010-62795954
印 刷 者:北京富博印刷有限公司
装 订 者:北京市密云县京文制本装订厂
经　　销:全国新华书店
开　　本:185mm×260mm　　**印　　张**:15.75　　**字　　数**:374千字
版　　次:2018年7月第1版　　**印　　次**:2018年7月第1次印刷
印　　数:1～1000
定　　价:39.00元

产品编号:076039-01

本书是与作者在清华大学出版社出版的《编译原理与技术(第2版)》(以下简称主教材)一书相配套的学习指导和习题解答教材，也可以与当前各种编译原理教材配套使用，既可辅助和引导学生完成课程学习，也可作为研究生入学考试的参考书。

“编译原理与技术”是计算机学科的一门核心课程，也是计算机系统软件中非常重要的一个分支。作为计算机类专业的学生，了解和掌握编译程序的基本构造原理和实现技术，学习和掌握编译程序的原理和技术，对今后的进一步学习、研究和工作奠定坚实的专业理论基础是十分必要的。由于编译程序的设计与实现涉及程序设计语言、形式语言与自动机理论、数据结构、计算机组成原理、操作系统以及软件工程学等诸多领域，因此需要综合运用这些知识来解决本课程所提出的一系列问题。另外，本课程所阐述的一些概念、原理、技术和方法也可应用于其他系统软件和应用软件的开发中。

对于计算机专业的学生，“编译原理与技术”是比较难于理解和掌握的课程，其包含的算法和思想比较特殊，理论性较强，抽象度较高。学生们都期盼着有一本好的习题解答，能够更好地帮助他们学习和掌握课程内容。作者也多次收到读者的电子邮件，希望能够提供主教材习题的参考答案。

作者讲授“编译原理与技术”这门课程已有二十余年，结合多年的教学经验和体会以及在教学过程中遇到的实际问题，对主教材设计的习题进行了详细的解析，本书前10章的名称与主教材保持一致，为使用主教材进行教学提供了支撑。同时，本书也对主教材中部分习题的描述进行了修正，以便读者可以单独使用本书进行学习。第11章提供了两套模拟试卷及参考答案。本书保持了主教材理论与实践相结合的特点，针对基础理论方面的习题，给出了涉及的知识要点和理论，分析了解题思路并给出了参考答案；针对编程实践类习题，给出了程序设计思路和要点，对于难度较大的编程题，还给出了程序伪码说明。主教材第11章的习题主要是希望读者编译、运行例题中所给的程序，根据程序的运行结果分析其执行过程，读者结合自己掌握的C++程序设计相关知识很容易完成这些习题，因此本书就不再给出这些习题的具体说明了。

限于作者水平，书中难免存在一些不足，敬请广大读者批评指正。

作　者

2018年1月

第1章　编译概述

1.1　知识要点

本章简要介绍编译程序的各个组成部分，对编译各逻辑阶段的内容进行概述。多数概念在以后的各章中有详细介绍。本章要求掌握以下内容：

(1) 源语言、目标语言、实现语言。

(2) 翻译程序、解释程序、编译程序。

(3) 编译程序的各个逻辑阶段及其主要功能。

(4) 编译的前端、后端、遍。

(5) 编译程序的伙伴工具。

1.2　习题解析及参考答案

1.1　高级程序设计语言有哪两种翻译方式？它们的特点分别是什么？

解答：

对高级程序设计语言的翻译有解释和编译两种方式。解释程序直接将源程序翻译成执行结果，编译程序则是把源程序翻译成另外一种形式，如汇编语言程序或机器语言程序。

解释程序对源程序进行解释执行，其特点是：边解释边执行，不生成目标程序，每次执行都需要对源程序重新进行分析解释。

编译程序将源程序转换成目标程序，目标程序在机器上执行。源程序的编译和执行在不同的时间进行，且目标程序可以重复执行而不需要重新编译。

1.2　典型的编译程序可以划分为哪几个主要部分？各部分的主要功能是什么？

解答：

典型的编译程序划分为词法分析、语法分析、语义分析、中间代码生成、代码优化以及目标代码生成6个部分。各部分的主要作用如下。

(1) 词法分析：扫描字符串表示的源程序，根据词法规则识别出具有独立意义的单词符号，输出记号流。

(2) 语法分析：分析程序的逻辑结构，根据源语言的语法规则把记号流按语法结构按层次分组，以形成语法短语。

(3) 语义分析：对语法成分的类型、含义进行检查，以保证程序各部分能够有机地结合在一起，并为以后生成目标代码收集必要的信息，如类型、目标地址等。

(4) 中间代码生成：将源程序转换为一种中间表示形式，以便于进一步转换成目标代码。

（5）目标代码生成：将中间形式代码转换成目标代码。

（6）代码优化：包含中间代码优化和目标代码优化两类。对中间代码或目标代码进行改进以获得高效的代码，即，使优化后的代码占用的空间少、运行速度快。

1.3 编译程序的设计与实现过程涉及哪些方面的知识？它们分别对编译程序有什么影响？

解答：

编译程序的设计与实现涉及以下几方面的知识。

- 程序设计语言：源语言、目标语言以及编译程序的实现语言。
- 形式语言与自动机理论：这是编译程序的理论基础，语言的结构可以用文法进行形式化描述，利用文法与自动机之间的等价关系，可基于自动机理论生成词法和语法分析程序。
- 计算机体系结构：决定了目标语言，影响目标代码生成。
- 数据结构、算法分析与设计：由于程序设计语言中都有数据结构、控制结构等，对源语言和目标语言中结构的理解、它们之间的映射关系、如何编写编译程序实现从源语言到目标语言的转换等，都离不开数据结构、算法分析与设计相关的知识。
- 操作系统：操作系统是编译程序工作的基础，编译程序的实现要借助于底层操作系统提供的功能，如存储空间分配和管理。
- 软件工程：编译程序是一个复杂的系统程序，需要在软件工程的思想指导下进行规范的设计开发。

1.4 编译程序有哪些伙伴工具？它们的作用是什么？

解答：

编译程序的伙伴工具主要有预处理器、汇编程序、连接装配程序。它们的作用如下。

- 预处理器：对源程序进行处理，产生编译程序的输入。预处理器主要完成宏处理、文件包含、语言扩充等功能。
- 汇编程序：有些编译程序产生汇编语言的目标代码，此时就需要由汇编程序对目标代码作进一步处理，生成可重定位的机器代码。
- 连接装配程序：完成连接和装配任务。所谓连接，即把几个可重定位的机器代码文件连接成一个可执行的程序，这些文件可以是分别编译或汇编得到的，也可以是由系统提供的、对任何需要它们的程序而言都可用的子程序组成的库文件。所谓装配，即读入可重定位的机器代码，修改需重定位的地址，把修改后的指令和数据放在内存中适当的地方或形成可执行文件。

1.5 解释下列名词：翻译程序、编译程序、汇编程序、解释程序、编译程序的遍、编译程序的前端、编译程序的后端。

解答：

翻译程序是把用某种语言书写的程序翻译成计算机能够执行的表示形式或直接执行这个程序的程序。

编译程序是翻译程序的一种，它把高级语言源程序转换成目标机器的机器语言或汇编语言程序。

汇编程序是编译程序的一种，它把汇编语言程序转换成目标机器的机器语言程序。

解释程序是翻译程序的一种，它直接对源程序进行解释执行。

编译程序的“遍”是指对源程序或其中间表示形式从头到尾扫描一次，并在扫描过程中作相关的加工处理，生成新的中间表示形式或目标程序。

编译程序的前端主要由与源语言有关而与目标机器无关的部分组成，通常包括词法分析、语法分析、语义分析、中间代码生成、符号表的建立以及与机器无关的代码优化工作，当然，前端也包括相应的错误处理工作和符号表操作。

编译程序的后端由编译程序中与目标机器有关的部分组成，一般来讲，这些部分与源语言无关而仅仅依赖于中间语言。后端包括目标代码的生成、与机器有关的代码优化以及相应的错误处理和符号表操作。

1.6 将编译程序划分成前端和后端的目的是什么？

解答：

将编译程序划分成前端和后端的目的是便于编译程序的移植和构造。例如，重写某个编译程序的后端，可以将该源语言的编译程序移植到另一种机器上，这种方法已经得到了普遍应用。重写某编译程序的前端，使之能把一种新的源语言编译成同一种中间语言，利用原编译程序的后端，从而可构造出新源语言在此机器上的编译程序。

1.7 请指出下面的错误可在编译的哪个阶段发现。

- 关键字拼写错误，如把 while 误写为 whlie。
- 缺少运算对象，如本应为 a=3+b;，却误写为 a=3+;。
- 实参与形参的类型不一致。
- 引用的变量没有定义。
- 数组下标越界。
- 本应为常数，但在数中出现了非数字字符，如误把 2.5 写成 2m5。

解答：

关键字拼写错误可以在语法分析阶段发现。例如，把 while 误写为 whlie，把 for 误写为 fro 等，词法分析程序会把它们识别为用户定义的标识符，只有到了语法分析阶段，在把记号组合成语法成分时才会发现这类问题。

缺少运算对象的错误可以在语法分析阶段发现。例如，源程序中出现了表达式 a+*b，词法分析程序可以识别出每个单词，依次为标识符 a、加号、乘号、标识符 b；而在语法分析阶段，在把它们组合成表达式的过程中，根据表达式的定义，会发现两个运算符号之间缺少运算对象。同样，对于源程序中出现的语句 a=3+;，词法分析程序依次识别出标识符 a、赋值号、常数 3、加号、分号，而在分析表达式结构时会发现缺少运算对象。

实参与形参的类型不一致的错误可以在语义分析阶段进行类型检查时发现。例如，某 C 语言程序中有如下的变量声明语句和函数定义：

```
int a, b;
string s;
int fx(int i, int j) {…}
```

在程序体中调用该函数的语句是

```
a=fx(b, s);
```

则在语义分析阶段，要检查实参和形参的个数是否相同，以及实参与相应形参的类型是否相同或相容，若发现不一致，则报告错误。

引用的变量没有定义的错误可以在语义分析阶段进行作用域检查时发现。例如，对于C语言赋值语句 x=a+4;，其中的名字 x 和 a 要么是在该赋值语句所在的函数中声明的局部变量，要么是全局变量。在分析该赋值语句时，首先在局部变量表（即函数的子符号表）中查找，若没有找到，则进一步在全局变量表（即主符号表）中查找，若还没有找到，则说明引用的名字没有声明。

数组下标越界的错误可以在语义分析阶段或者程序执行阶段发现。例如在某C语言程序中声明数组变量：int a[10]，而在可执行语句中通过下标表达式引用数组元素。若下标表达式是一个整型常数，例如11，则编译程序将发现引用 a[11]是错误的；若下标表达式是一个整型变量，例如 i，若编译程序可以分析出 i 有值，并且 i<0 或者 i>9，则引用a[i]是错误的。即，若编译程序可以确定下标表达式的值，则数组下标越界的错误可以在语义分析时发现；如果编译程序无法确定下标表达式的值，则数组越界的错误只能在程序运行中出现存储溢出才被发现。例如，程序通过整型变量 i 引用 a[i]，该引用出现在一个循环结构中，编译时将无法确定 i 的值，如果越界，也只能在程序运行中出现存储溢出时才能发现。

本应为常数，但在数中出现了非数字字符，这类错误有些可以在词法分析阶段发现，如 2.3.4、2.3@4、3.4E.5 等都属于词法错误；而有些只能在语法分析阶段发现，如程序员误将语句 x=234;写成了 x=2w4;，这样的错误只有在语法分析程序分析赋值语句时才可以发现。

第 2 章　形式语言与自动机基础

2.1　知识要点

本章内容是编译的理论基础，要求读者掌握形式语言与自动机相关的基本概念，重点掌握文法的形式定义、文法的分类、右线性文法、上下文无关文法及文法与自动机之间的等价关系。

本章要求掌握以下基本概念和方法：

(1) 字母表，定义在字母表上的符号串，符号串的前缀、后缀、子串、子序列，语言。

(2) 符号串的连接、符号串的幂，语言的并、连接、幂，语言的闭包。

(3) 文法的形式定义及分类。

(4) 推导、最左推导、最右推导、规范推导、直接推导。

(5) 句型、左句型、右句型，短语、直接短语、句柄。

(6) 句型的分析树、子树，子树与短语的对应关系，分析树与文法二义性的关系。

(7) 左递归文法，文法中左递归的消除方法。

(8) 产生式的左公因子及其提取变换。

(9) 自动机的形式定义，DFA、NFA、具有 ε-转移的 NFA。

(10) NFA 确定化方法，DFA 的化简方法。

(11) 与右线性文法等价的 NFA 的构造方法，与正规表达式等价的 NFA 的构造方法。

2.2　习题解析及参考答案

2.1　写出字符串 $abcd$ 的所有前缀、后缀、子串和子序列，以及真前缀、真后缀和真子串。

解答：

符号串 α 的前缀，指的是从符号串 α 的末尾删除 0 个或多个符号之后得到的符号串。

符号串 α 的后缀，指的是从符号串 α 的开头删除 0 个或多个符号之后得到的符号串。

符号串 α 的子串，指的是删除 α 的前缀和/或后缀之后得到的符号串。

符号串 α 的子序列，指的是从符号串 α 中删除 0 个或多个符号(这些符号可以是不连续的)之后得到的符号串。

如果符号串 β 是 α 的前缀、后缀或子串，并且 $\beta \neq \alpha$，则称 β 是 α 的真前缀、真后缀或真子串。

字符串 $abcd$ 的前缀有 ε、a、ab、abc、$abcd$，真前缀有 ε、a、ab、abc。

字符串 $abcd$ 的后缀有 ε、d、cd、bcd、$abcd$，真后缀有 ε、d、cd、bcd。

字符串 $abcd$ 的子串有 ε、a、b、c、d、ab、bc、cd、abc、bcd、$abcd$，真子串有 ε、a、b、c、d、ab、bc、cd、abc、bcd。

$abcd$ 的所有子串均是其子序列，另外还有 acd、abd、ac、ad 以及 bd，除了 $abcd$ 之外的所有子序列均是 $abcd$ 的真子序列。

2.2 用文字描述下列文法产生的语言。

(1) G_1：$S\rightarrow SaS$　　$S\rightarrow b$

(2) G_2：$S\rightarrow aSb$　　$S\rightarrow c$

(3) G_3：$S\rightarrow a$　　$S\rightarrow aB$　　$B\rightarrow aS$

解答：

(1) $L(G_1)=\{(ba)^*b\}$，或者 $L(G_1)=\{b(ab)^*\}$。

它表示由 a、b 组成，以 b 开头且以 b 结尾，并且若长度大于 1，则 a 和 b 交替出现的符号串的全体。

(2) $L(G_2)=\{a^mcb^m \mid m\in \mathbf{N}$ 并且 $m\geqslant 0\}$。

它表示由 a、b、c 组成，以 c 为中心，c 之前的子串只含有符号 a，c 之后的子串只含有符号 b，并且 a 与 b 的个数一样多的符号串的全体。

(3) $L(G_3)=\{aa^{2n} \mid n\in \mathbf{N}$ 并且 $n\geqslant 0\}$，或者 $L(G_3)=\{a^{2n}a \mid n\in \mathbf{N}$ 并且 $n\geqslant 0\}$。

它表示由奇数个 a 组成的符号串的全体。

2.3 有文法 G：$S\rightarrow aSbS \mid bSaS \mid \varepsilon$。

(1) 通过为该文法的句子 $abab$ 构造两个不同的最左推导说明该文法是二义性的。

(2) 写出句子 $abab$ 的两个不同的最右推导。

(3) 为句子 $abab$ 构造两棵不同的分析树。

(4) 用文字描述该文法所产生的语言。

解答：

(1) 最左推导指的是在每一步推导中都替换当前句型中最左边的非终结符号。题目中句子 $abab$ 的最左推导有

$S\Rightarrow aSbS\Rightarrow abSaSbS\Rightarrow ababS\Rightarrow abab$

$S\Rightarrow aSbS\Rightarrow abS\Rightarrow abaSbS\Rightarrow ababS\Rightarrow abab$

上述推导表明句子 $abab$ 存在两个不同的最左推导，说明句子 $abab$ 是二义性的，含有二义性句子的文法是二义性文法，所以，该文法是二义性的。

(2) 最右推导指的是在每一步推导中都替换当前句型中最右边的非终结符号。题目中句子 $abab$ 有两个不同的最右推导，分别是

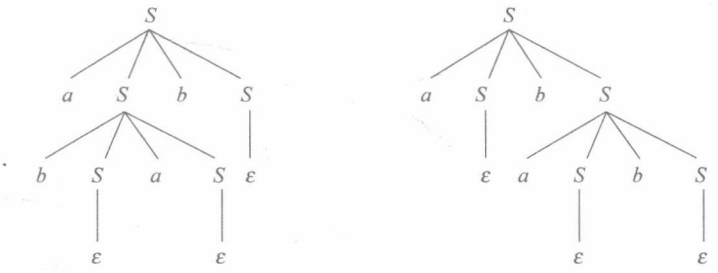
$S\Rightarrow aSbS\Rightarrow aSb\Rightarrow abSaSb\Rightarrow abSab\Rightarrow abab$

$S\Rightarrow aSbS\Rightarrow aSbaSbS\Rightarrow aSbaSb\Rightarrow aSbab\Rightarrow abab$

这同样说明句子 $abab$ 是二义性的，产生该句子的文法是二义性文法。

(3) $abab$ 有如图 2-1 所示的两棵不同的分析树，说明该文法具有二义性。

(4) 该文法所产生的语言是由 a、b 组成的长度为偶数并且 a 和 b 个数相同的所有符号串组成的集合。



(a) 分析树1　　(b) 分析树2

图 2-1　*abab* 的两棵不同的分析树

2.4　考虑文法 G：

$expr \rightarrow expr$ or $term \mid term$

$term \rightarrow term$ and $factor \mid factor$

$factor \rightarrow$ not $factor \mid (expr) \mid$ true $\mid$ false

(1) 写出该文法的终结符号、非终结符号和开始符号。

(2) 构造句子 not(true or false)的分析树，并给出其短语、直接短语和句柄。

(3) 说明该文法产生的语言是全体布尔表达式。

解答：

(1) 通常在书写文法时有一些常用的约定。本书采用以下约定(参见主教材 2.1.3 节)：正体字符串，如 id、begin、if、then 等，表示终结符号；小写的斜体符号串，如 *expr*、*term*、*factor*、*stmt* 等，表示非终结符号；可以直接用产生式的集合代替四元组来描述文法，第一个产生式的左部符号是文法的开始符号。

据此可知该文法的终结符号集合 V_T、非终结符号 V_N 和开始符号分别是

$V_T = \{$or, and, not, (,), true, false$\}$

$V_N = \{expr, term, factor\}$

开始符号：*expr*

(2) 句子 not(true or false)的分析树如图 2-2 所示。

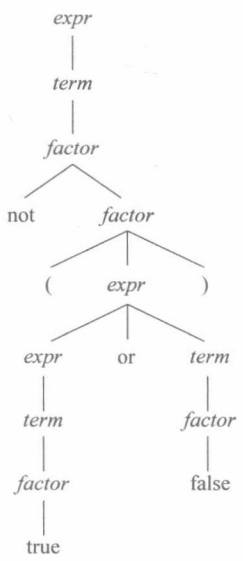


图 2-2　句子 not(true or false)的分析树

一棵分析树的子树是指分析树中的一个特有的内部结点连同它的全部后裔结点、连接这些结点的边以及这些结点的标记。

子树与短语之间存在着十分密切的关系。①句型：一棵分析树的所有叶结点自左至右排列起来形成的文法符号串是文法的一个句型；②短语：任何一棵子树的所有叶结点自左至右排列起来形成的文法符号串都是该句型的一个相对于该子树根结点的短语；③直接短语：分析树中只有父子两代的子树的所有叶结点自左至右排列起来形成的文法符号串是该句型相对于该子树根结点的直接短语；④句柄：分析树中最左边的直接短

语，即最左直接短语，就是该句型的句柄。

图 2-3 给出了句子 not(true or false)的子树及对应的短语。

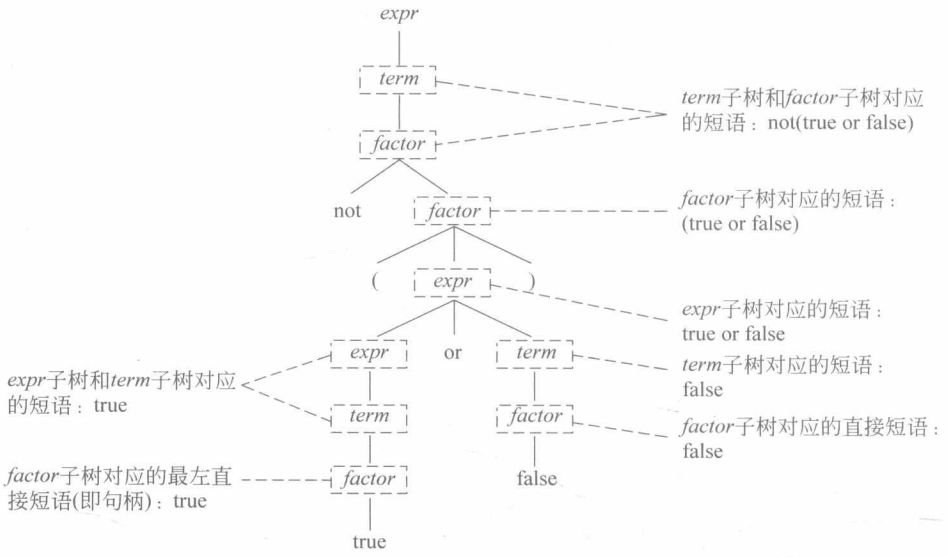


图 2-3 句子 not(true or false)的子树及对应的短语

从图 2-3 可以看出，句子 not (true or false)的短语有 not (true or false)、(true or false)、true or false、true、false。该句子的直接短语有 true、false。该句子的句柄(即最左直接短语)是 true。

(3) 该文法产生的语言是全体布尔表达式，理由如下。

布尔表达式可定义如下：

① 基本布尔量 true、false 是布尔表达式。

② 如果 E 是布尔表达式，则(E)仍是布尔表达式。

③ 如果 E 是布尔表达式，则 not E 仍是布尔表达式。

④ 如果 E_1 和 E_2 是布尔表达式，则 E_1 or E_2、E_1 and E_2 仍是布尔表达式。

⑤ 只有有限次使用上述步骤得到的符号串是布尔表达式，除此之外没有其他的符号串是布尔表达式。

首先，查看该文法，该文法可以推导出的句子或句型如下：

① 从开始符号可以推导出 true、false：

$expr \Rightarrow term \Rightarrow factor \Rightarrow$ true

$expr \Rightarrow term \Rightarrow factor \Rightarrow$ false

② 从开始符号可以推导出带括号的布尔表达式：

$expr \Rightarrow term \Rightarrow factor \Rightarrow (expr)$

③ 从开始符号可以推导出形如 not E 的布尔表达式：

$expr \Rightarrow term \Rightarrow factor \Rightarrow$ not $factor \Rightarrow$ not $(expr)$

④ 从开始符号可以推导出形如 E and E 和 E or E 的布尔表达式：

$expr \Rightarrow term \Rightarrow term$ and $factor \Rightarrow factor$ and $factor \Rightarrow (expr)$ and $(expr)$

$expr \Rightarrow expr$ or $term \Rightarrow expr$ or $factor \Rightarrow expr$ or $(expr)$

$expr \Rightarrow expr$ or $term \Rightarrow term$ or $factor \Rightarrow factor$ or $factor \Rightarrow (expr)$ or $(expr)$

综上所述，该文法推导出的句子均是布尔表达式。

其次，根据布尔表达式的定义可知，所有符合该定义的布尔表达式均可以由该文法推导出。

所以，该文法产生的语言是全体布尔表达式。

2.5 为了矫正 if-then-else 语句中悬而未决的 else，提出了下面的文法，试说明该文法仍然是二义性的。

$stmt \rightarrow$ if $expr$ then $stmt \mid matched\_stmt$

$matched\_stmt \rightarrow$ if $expr$ then $matched\_stmt$ else $stmt \mid$ other

解答：

如果文法的某个句子具有两棵不同的分析树，则该句子是二义性的。具有二义性句子的文法是二义性文法。

考虑该文法的句子：

if $expr$ then if $expr$ then other else if $expr$ then other else other

该句子有如图 2-4 和图 2-5 所示的两棵不同的分析树，说明该句子是二义性的，进而说明该文法仍具有二义性。

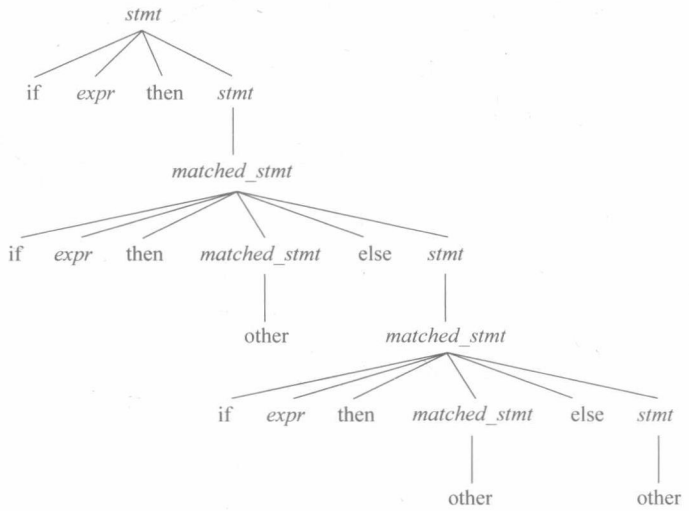


图 2-4 句子 if $expr$ then if $expr$ then other else if $expr$ then other else other 的分析树 1

2.6 请对下面的文法进行消除左递归等价变换。

$S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S$

解答：

该文法中 L 的产生式存在直接左递归。

消除直接左递归的方法如下。

如果文法 G 有产生式 $A \rightarrow A\alpha \mid \beta$，其中候选式 β 不以 A 开头。产生式 $A \rightarrow A\alpha$ 表明非

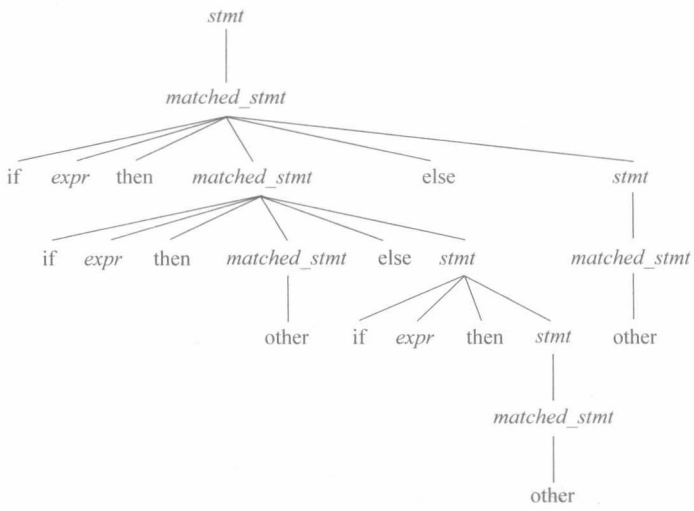


图 2-5　句子 if *expr* then if *expr* then other else if *expr* then other else other 的分析树 2

终结符号 A 是直接左递归的。通过引入一个新的非终结符号 R，把 A 的这两个产生式改写为

$A \rightarrow \beta R$

$R \rightarrow \alpha R \mid \varepsilon$

由于这两组产生式从 A 推导出的符号串是相同的，即都是 $\beta\alpha\cdots\alpha$，故它们是等价的。

针对题目中的文法 $G[S]$，消除左递归后得到的文法如下：

$S \rightarrow (L) \mid a$

$L \rightarrow SL'$

$L' \rightarrow ,SL' \mid \varepsilon$

2.7　请写出下列语言的正规表达式，并为之构造右线性文法。

（1）以字母或“_”开头的，由字母、数字或“_”组成的符号串（如 C 语言中的标识符）的全体。

（2）以 I、J、K、L、M 或 N 开头的、由字母或数字组成的、最大长度为 6 的标识符的全体。

（3）不包含子串 011 的由 0 和 1 组成的符号串的全体。

（4）不包含子序列 011 的由 0 和 1 组成的符号串的全体。

（5）所有具有 3 个 0 的由 0 和 1 组成的符号串的全体。

（6）所有以 00 结尾的由 0 和 1 组成的符号串的全体。

（7）含有奇数个 0 的由 0 和 1 组成的符号串的全体。

（8）含有偶数个 1 的由 0 和 1 组成的符号串的全体。

（9）能被 5 整除的十进制无符号整数，规定整数不能以 0 开头。

解答：

（1）以 letter 表示字母，digit 表示数字，则该语言的正规表达式是

(_|letter)(letter|digit|_)*

相应的右线性文法如下：

id→letter *rid*|_ *rid*

rid→letter *rid*|digit *rid*|_ *rid*|ε

(2) 符号串的首字母已经确定，只能是I、J、K、L、M或N，后面最多可以跟随5个由字母或数字组成的符号串(可以是ε)。若用letter表示任意字母，digit表示任意数字，则该语言的正规表达式为

(I|J|K|L|M|N) (letter|digit|ε)^5

相应的右线性文法如下：

id→I *rid*1|J *rid*1|K *rid*1|L *rid*1|M *rid*1|N *rid*1

*rid*1→letter *rid*2|digit *rid*2|ε

*rid*2→letter *rid*3|digit *rid*3|ε

*rid*3→letter *rid*4|digit *rid*4|ε

*rid*4→letter *rid*5|digit *rid*5|ε

*rid*5→letter|digit|ε

(3) 根据子串的定义可知，符号串中不能有011出现，即在0的后面不允许两个或两个以上的1连续出现。分以下两种情况考虑。

第一种情况，假定该语言中不包含句子ε，则该语言的句子包括以下3种形式：

① 全部由1组成的符号串：1^+(=1^*1)。

② 以0结尾的符号串：$1^*0(0|10)^*$(=$1^*0^+(10^+)^*$)。

③ 以1结尾的符号串：$1^*0^+(10^+)^*1$。

综合上述3种情况，满足要求的符号串的正规表达式是

$$1^*(1|0^+(10^+)^*(1|\varepsilon))$$

识别该语言的DFA如图2-6所示。

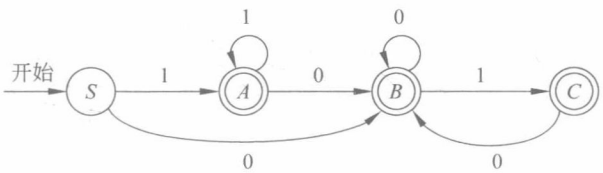


图2-6 识别不含子串011的符号串(不含ε)的DFA

与之相应的右线性文法如下：

S→1*A*|0*B*|1|0

A→1*A*|0*B*|1|0

B→1*C*|0*B*|1|0

C→0*B*|0

第二种情况，假定该语言中包含句子ε，则满足要求的符号串的正规表达式是

$$1^*0^*(10^+)^*(1|\varepsilon)$$

识别该语言的DFA如图2-7所示。