编译原理习题精选分析与解答

杨宗源　编著

清华大学出版社

编译原理习题精选分析与解答

杨宗源　编著

清 华 大 学 出 版 社

北　京

内容简介

编译原理是计算机专业的一门重要的专业基础课，理论性、实践性较强。本书在对这门课程的主要内容作出要点提示的基础上，精选了一些典型习题，并根据编者多年的教学实践，有针对性地分析了习题的求解思路，给出了详尽的习题解答过程。

本书可作为高等院校计算机专业本科生的学习指导书、教学参考书，亦可作为报考相关专业研究生的学习辅导书。

图书在版编目(CIP)数据

编译原理习题精选分析与解答/杨宗源编著. —北京：清华大学出版社，2003

ISBN 7-302-06445-8

Ⅰ.编… Ⅱ.杨… Ⅲ.编译程序-高等学校-解答 Ⅳ.TP314-44

中国版本图书馆 CIP 数据核字(2003)第 018879 号

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.tsinghua.edu.cn
策划编辑：朱英彪
责任编辑：杜春杰
印 刷 者：北京昌平环球印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：15.25　**字数**：348 千字
版　　次：2003 年 7 月第 1 版　2003 年 7 月第 1 次印刷
书　　号：ISBN 7-302-06445-8/TP・4853
印　　数：0001～5000
定　　价：19.00 元

目 录

第1章　形式语言基础

1.1　文法与语言的形式定义

1.1.1　要点提示

1．文法 G 是一个四元组 G=（$V_T,V_N,P,<S>$）。其中：V_T 是非空有限的终结符集合，V_N 是非空有限的非终结符集合，P 是非空有限产生式集合，$<S>\in V_N$ 是文法 G 的开始符号。

集合 P 中产生式的一般形式是$\alpha\rightarrow\beta$（或$\alpha::=\beta$）。其中：“$\rightarrow$”是元符号，α称为产生式的左部，β称为产生式的右部。

Chomsky 根据产生式的不同形式，将文法分为四类，分别是：0 型文法（又称短语结构文法）、1 型文法（又称上下文有关文法）、2 型文法（又称上下文无关文法）、3 型文法（又称正则文法）。

编译程序所涉及的主要是 2 型和 3 型文法，分别用于句法分析与词法分析。

2 型文法的一般形式为$<V>\rightarrow\alpha$，$<V>\in V_N$，$\alpha\in(V_N\cup V_T)^*$。

3 型文法的一般形式为$<A>\rightarrow a<B>$或$<A>\rightarrow a$，$<A>$、$<B>\in V_N$，$a\in V_T$（或$<A>\rightarrow<B>a$，$<A>\rightarrow a$）。

2．文法 $G[<S>]$中，$<S>$是文法的开始符号。对$<S>\Rightarrow^*\alpha$，若$\alpha\in(V_N\cup V_T)^*$，则称α是 G 的一个句型；若$\alpha\in V_T^*$，则称α是 G 的一个句子。

文法 $G[<S>]$的句子的全体所组成的集合称为 $G[<S>]$的语言，记作 $L(G[<S>])=\{w|<S>\Rightarrow^*w,w\in V_T^*\}$。

3．从实用的角度出发，在编译原理中所讨论的文法是被简化了的文法，即：

1）文法中不含形如$<U>\rightarrow<U>$的产生式。

2）任意的$<U>\in V_N$ 存在推导$<S>\Rightarrow^*\alpha<U>\beta$（$<S>$是文法的开始符号），以及存在$r\in V_T^*$使$<U>\Rightarrow^*r$，即$<U>$是可及的和有用的。

4．对于两个不同的文法 G 与 G1，若它们的语言集合相同，即 L(G)=L(G1)，则称 G 与 G1 等价。

1.1.2 习题解析

1．没有中心元素的镜像结构语言 $L=\{\alpha\alpha^R|\alpha\in V_T^+$，$\alpha^R$ 为α的逆$\}$和具有中心元素的镜像结构语言 $L1=\{\alpha Z\alpha^R|\alpha\in V_T^+$，$\alpha^R$ 为α的逆$\}$，若 $V_T=\{0，1\}$，试分别给出生成上述语言的上下文无关文法。

分析：

L 接收的句子的形式是 00，11，0110，1001，…，而 L1 接收的句子的形式是 0Z0，1Z1，01Z10，10Z01，…，即它们分别是对称的，所不同的是：L 没有中心元素，而 L1 有中心元素 Z。

解答：

令 G[<S>]的产生式集合 P 中的产生式为：

<S>→0<S>0|1<S>1|ε

则推导得：<S>⇒0<S>0⇒01<S>10⇒0110 知：0110∈L(G)。

而 G1[<S>]为：

<S>→0<S>0|1<S>1|Z

则由推导<S>⇒0<S>0⇒01<S>10⇒01Z10 知：01Z10∈L(G1)。

2．试给出能生成非零开头的正偶数集合的文法 G[<S>]。

分析：

G[<S>]的句子中，由单个符号组成的符号串是 2、4、6、8 中的任 1 个符号；2 个符号组成的符号串中，左边起第一个符号可以是 1~9 中的某一个，第二个符号是 0、2、4、6、8 中的某一个；3 个及 3 个以上符号组成的串，最左边的符号与 2 个符号组成的符号串的第一个符号相同，最右边的符号与 2 个符号组成的符号串的第二个符号相同，而中间的符号可以是 0~9 中的任何一个。掌握了这一规律，就容易给出满足条件的文法。

解答：

G[<S>]的产生式集合为：

<S>→<X><Y><Z>|2|4|6|8

<X>→1|2|3|4|5|6|7|8|9

<Y>→<Y><X>|<Y>0|ε

<Z>→0|2|4|6|8

其中，左部为非终结符<Y>的产生式最为关键。

3．试将如下文法 G[<S>]转换为等价的正则文法（3 型文法）：

①<S>→a<A>

②<S>→<A>

③<A>→abb<S>

④<A>→c<A>

⑤<A>→a

分析：

1）通常称文法 G 的产生式是形如：

<A>→x<B>或<A>→x

其中，<A>、<B>$\in V_N$，$x\in V_T^*$的文法为右线性文法，而形如：

<A>→<B>x 或<A>→x

其中，<A>、<B>$\in V_N$，$x\in V_T^*$的文法为左线性文法，本题的文法是右线性文法。

2）适当地引入一些新的非终结符，容易把一个右线性文法（左线性文法）转换为等价的正则文法。

解答：

按正则文法的定义，只需对产生式②、③进行等价变换。

对产生式③用　<A>→a<bbS>

<bbS>→b<bS>

<bS>→b<S>

来代替。

对产生式②则用　<S>→a<bbS>

<S>→c<A>

<S>→a

来代替。

因此与 G[<S>]等价的正则文法 G1[<S>]为：

<S>→a<A>|a<bbS>|c<A>|a

<A>→a<bbS>|c<A>|a

<bbS>→b<bS>

<bS>→b<S>

其中：<bbS>、<bS>是新引入的非终结符。

4．试将如下的含有ε产生式的上下文无关文法（2 型文法）G[<S>]转换成等价的不含ε产生式的上下文无关文法。

①<S>→a<A><S>

②<S>→b

③<A>→c<S>

④<A>→ε

分析：

一般而言，含有ε产生式的上下文无关文法可以转化为不含ε产生式的等价的上下文无关文法。但当文法的开始符号可以推导出ε，即空串ε属于语言 L(G)时，转化前后的两个文法所接受的语言会相差ε。本题中，<S>不能推导出ε，所以转换后的文法与原文法等价。

解答：

引入非终结符<AN>与<AY>，<AN>表示不可空，而<AY>表示可空。随后消除可空的产生式以及在产生式的右部去除非终结符<AY>即可，因此 G[<S>]可等价表示为：

①<S>→a<AN><S>

②<S>→a<AY><S>

③<S>→b

④<AN>→c<S>

⑤<AY>→ε

最后得到等价的无ε产生式的文法 G1[<S>]为：

①<S>→a<AN><S>

②<S>→a<S>

③<S>→b

④<AN>→c<S>

5．试为如下的语言构造两个（或两个以上）等价文法，它们接受相同的语言。

① $L1=\{a^{2n}b|n\geq 1\}$

② $L2=\{a^ib^jc^k|i，j，k\geq 1\}$

分析：

对给定的一个文法，可惟一地确定其所接受的语言，而对给定的语言却可能为其构造出多个不同的文法，通常称这些文法是等价的。

语言 L1 要求其每个句子具有 2n 个 a 与 1 个 b（n≥1）；

语言 L2 要求其每个句子具有 i 个 a、j 个 b、k 个 c（i、j、k≥1）。

解答：

对语言 L1 构造文法 G1[<S>]为：

<S>→<A>b

<A>→a<A>a

<A>→aa

文法 G2[<S>]为：

<A>→aa<A>

<A>→aa

对语言 L2 构造文法 G3[<S>]为：

<S>→<A><B><C>

<A>→a<A>|a

<B>→b<B>|b

<C>→c<C>|c

文法 G4[<S>]为：

<S>→<S>c|<B>c

<B>→<B>b|<A>b

<A>→<A>a|a

显然，G1 与 G2 等价，G3 与 G4 等价。

6．设文法 G[<S>]的产生式为：

<S>→(<S>)|ε

1）问该文法所产生的语言 L(G[<S>])是什么？

2）试证明 1）的结果。

分析：

欲证明文法所产生的语言，对推导的次数作数学归纳是常用的方法。本题文法 G[<S>]仅有两个产生式，且一旦使用产生式<S>→ε，则推导结束，因此较为简单。

解答：

1）L(G[<S>])={(n)n| n≥0}

2）证明过程为：

（1）首先证明 L(G[<S>])⊆$\{(^n)^n | n\geq 0\}$，对推导次数进行归纳：

① 当推导次数为 1 时，使用产生式<S>→ε，此时句子 ε∈L(G[<S>])。

② 假定推导次数为 n 时，结论成立，即：

$$\langle S\rangle \Rightarrow^{+} \underbrace{(((\cdots(}_{n-1}\langle S\rangle\underbrace{)\cdots)))}_{n-1} \Rightarrow \underbrace{(((\cdots(}_{n-1}\underbrace{)\cdots)))}_{n-1}$$

则当推导次数为 n+1 时，再使用一次产生式<S>→(<S>)即为：

$$\langle S\rangle \Rightarrow^{+} \underbrace{(((\cdots(}_{n-1}\langle S\rangle\underbrace{)\cdots)))}_{n-1} \Rightarrow \underbrace{(((\cdots(}_{n}\langle S\rangle\underbrace{)\cdots)))}_{n} \Rightarrow \underbrace{(((\cdots(}_{n}\underbrace{)\cdots)))}_{n}$$

根据①、②有 $L(G[\langle S\rangle])\subseteq\{(^n)^n \mid n\geq 0\}$。

（2）再证明$\{(^n)^n \mid n\geq 0\}\subseteq L(G[\langle S\rangle])$，对 n 进行归纳：

① 当 n=0 时，使用产生式$\langle S\rangle\rightarrow\varepsilon$ 即可证明。

② 假定当 n=k 时结论成立，即$(^k)^k\in L(G[\langle S\rangle])$，再证明当 n=k+1 时结论成立。

因为$(^k)^k\in L(G[\langle S\rangle])$的推导过程为：

$$\langle S\rangle\Rightarrow^{+}\underbrace{(((\cdots(}_{k}\langle S\rangle\underbrace{)\cdots)))}_{k}\Rightarrow\underbrace{(((\cdots(}_{k}\underbrace{)\cdots)))}_{k}$$

所以当 n=k+1 时推导过程为：

$$\langle S\rangle\Rightarrow^{+}\underbrace{(((\cdots(}_{k}\langle S\rangle\underbrace{)\cdots)))}_{k}\Rightarrow\underbrace{(((\cdots(}_{k+1}\langle S\rangle\underbrace{)\cdots)))}_{k+1}\Rightarrow\underbrace{(((\cdots(}_{k+1}\underbrace{)\cdots)))}_{k+1}$$

因此，$(^{k+1})^{k+1}\in L(G[\langle S\rangle])$。

由①、②可得$\{(^n)^n \mid n\geq 0\}\subseteq L(G[\langle S\rangle])$；

根据（1）、（2）可知：$L(G[\langle S\rangle])=\{(^n)^n \mid n\geq 0\}$。

1.2 推导树与二义性文法

1.2.1 要点提示

1．推导树又称语法树、句法树，它是推导的一种图式表示，因此一个推导对应着一棵推导树。它对于理解和表示许多重要的概念很有用。

2．短语、简单短语、句柄是自底向上句法分析中的几个重要概念。对给定的句型 w，画出它所对应的推导树则容易求得 w 对应的短语、简单短语和句柄。其规则为：推导树每一个子树的叶节点的全部符号组成的符号串是该句型 w（对于子树根）的短语，若该子树树高两层（即只有父子两代），则该短语又是简单短语，最左简单短语称为句柄。

3．若文法 G 的一个句子对应有两棵或两棵以上不同的推导树，则称该句子是二义性的。产生二义性句子的文法称为二义性文法，否则该文法是无二义性的。

4．对二义性文法所产生的语言，有的可找到等价的无二义性文法，因此只说文法的二义性，而不说语言的二义性。当然存在这样的语言，即不存在它们的无二义性文法。例如：$L=\{a^ib^jc^k \mid i=j$ 或 $j=k,i,j,k\geq 1\}$。

1.2.2 习题解析

1．文法 G[<S>]的产生式为：

<S>→a|b|(<R>)

<T>→<S>c<T>|<S>

<R>→<T>

1）构造句型(bc<S>c<T>)的推导树，并指出该句型的所有短语、简单短语、句柄。

2）若句柄存在，求下述符号串的句柄。

（1）((<S>c(b)))　（2）(<S>)　（3）(a)　（4）<S>c<T>　（5）<S>　（6）(b)

分析：

符号串的某一部分符号要成为句柄，必须满足：

1）该符号串必须是该文法的句型（句子）。

2）是最左简单短语，所以文法的开始符号不是句柄。

因此，只要给出句型（句子）对应的推导树就容易求出短语、简单短语和句柄。

解答：

1）句型(bc<S>c<T>)的推导树如图 1.1 所示。

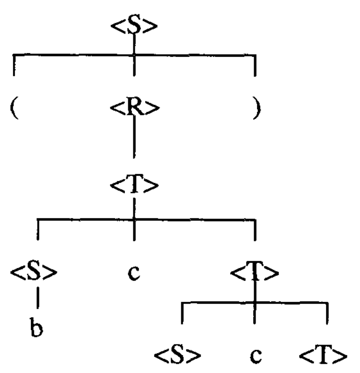

图 1.1　句型(bc<S>c<T>)的推导树

短语：b; bc<S>c<T>; <S>c<T>; (bc<S>c<T>)

简单短语：b; <S>c<T>

句柄：b

2）（1）句型((<S>c(b)))的推导树如图 1.2 所示。

所以句柄为 b。

（2）句型(<S>)的推导树如图 1.3 所示。

所以句柄为<S>。

（3）句子(a)的推导树如图 1.4 所示。

所以句柄为 a。

（4）<S>c<T>不是文法 G[<S>]的句型，所以句柄不存在。

（5）<S>是开始符号，所以句柄不存在。

（6）句子(b)的推导树如图 1.5 所示。

所以句柄为 b。

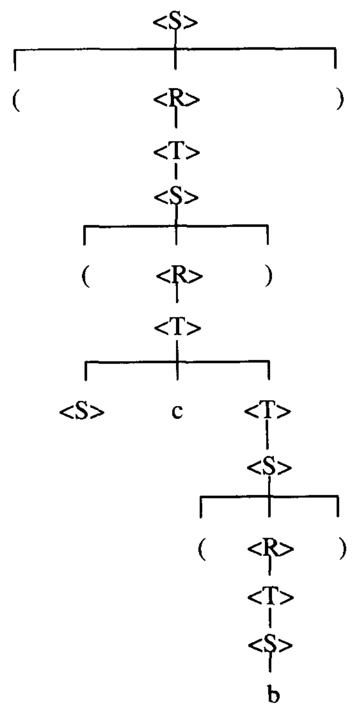

图 1.2　句型((<S>c(b)))的推导树

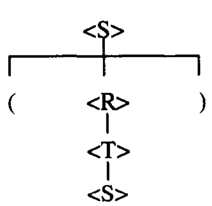

图 1.3　句型(<S>)的推导树

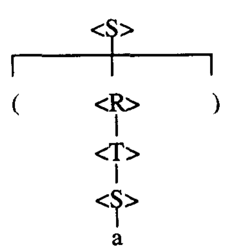

图 1.4　句子(a)的推导树

图 1.5　句子(b)的推导树

2．考察文法 G[<S>]，其产生式为：

<S>→<IS>|a

<IS>→if(<E>)<S>|if(<E>)<S>else<S>

<E>→0|1

1）试证文法 G[<S>]是二义性文法。

2）试给出消除二义性规则，根据此规则改造文法 G[<S>]为 G1[<S>]，使

L(G[<S>])=L(G1[<S>])，但 G1[<S>]是非二义性文法。

分析：

1）本题比较简单，只需找出文法 G[<S>]的一个二义性句子即可。

2）本题实际上是要求解决所谓“悬挂 else 问题”。按通常程序设计语言的规定，消除二义性的规则是：一个 else 部分总是与没有 else 部分的最近的 if 语句相配对，即所谓的“最近原则”，这可以通过增加非终结符、修改和增加产生式来解决。

解答：

1）考虑 G[<S>]的句子 if(0) if(1) a else a，该句子有如下 2 棵不同的推导树，如图 1.6 和图 1.7 所示，因此它是二义性句子，文法 G[<S>]是二义性文法。

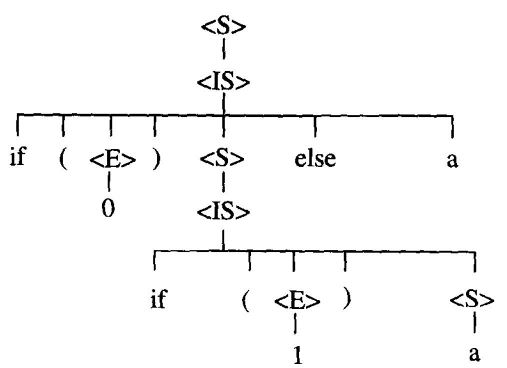

图 1.6　句子 if(0) if(1) a else a 的推导树（1）

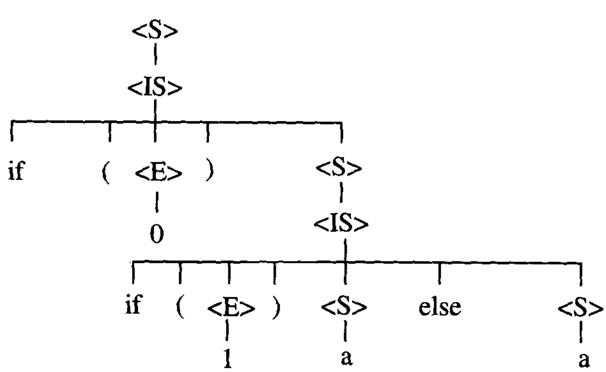

图 1.7　句子 if(0) if(1) a else a 的推导树（2）

2）要想实行“最近原则”，关键是将有 else 子句和无 else 子句的 if 语句区分开，出现在 else 前面的非终结符与无 else 的 if 后面的非终结符不同（非终结符<E>后面的非终结符），因此可将 G[<S>]改写为 G1[<S>]：

<S>→<M>|<U>

<M>→if(<E>)<M>else<M>|a

<U>→if(<E>)<S>|if(<E>)<M>else<U>

<E>→0|1

这样对句子 if(0) if(1) a else a 就只能有惟一的推导树，如图 1.8 所示。

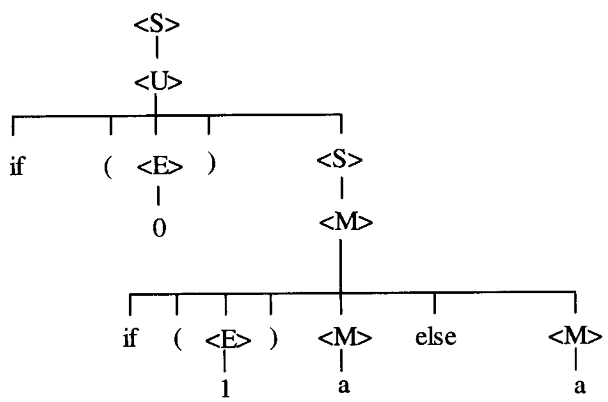

图 1.8 句子 if(0) if(1) a else a 的推导树

该问题还有一种解，其思想方法是为 if 语句带上一个结尾关键字，如 fi(或 end if)。这样 G[<S>]可改写为：

<S>→if<E><S>fi|if<E><S>else<S>fi|a

<E>→0|1

当然改造后的文法与 G[<S>]不等价了，因为必须增加终结符 fi，但从实用的角度来看，它解决了悬挂 else 子句的二义性问题。

3. 考察文法 G[<R>]

<R>→<R> '|' <R>

<R>→<R><R>

<R>→<R>*

<R>→(<R>)

<R>→a

<R>→b

注意：

'|' 是终结符，而不是元符号。

1）证明文法 G[<R>]生成字母表 V={a,b}上的正则表达式的全体；

2）证明文法 G[<R>]是二义性文法；

3）若消除二义性规则为：*的优先级最高，并置其次，‘|’最低且都为左结合的。试根据此规则给出等价的无二义性文法 G1[<R>]；

4）试分别使用文法 G 与 G1 构造句子 a|b*a 的推导树。

分析：

1）欲证明 G[<R>]生成 V 上所有正则表达式，数学归纳法是有力的“武器”。我们可以对终结符*、‘|’、(,)（注意(,)成对出现）的个数 n 应用归纳法。

2）消除文法 G[<R>]的二义性规则的使用，可参考一般算术表达式文法在消除二义性时所采用的技术。

解答：

1）在文法 G[<R>]中，V_T={‘|’,*,(,),a,b}.因此由正则表达式的定义，该文法所产生的任一句子显然为 V={a,b}上的正则表达式。

下面证明 V={a,b}上的任一正则表达式 REX 均为文法 G 的一个句子。不妨设 REX 中的运算符共有 n 个（它们可能是‘|’,*或一对(,)），对 n 作归纳：

（1）若 n=0，即 REX 中无运算符，则 REX=a 或 REX=b，这可由<R>→a 或<R>→b 生成。

（2）不妨设 n≤k，REX 可由文法 G 生成。

对 n=k+1，若 REX=R|S，则 R、S 中运算符的个数均小于等于 k，由归纳假设，R、S 均可由文法 G 生成，因此 REX 可由产生式<R>→<R>‘|’<R>生成；同理，REX=(R)可由产生式<R>→(<R>)生成，REX=R*可由产生式<R>→<R>*生成，所以 n=k+1 时成立。

综上，文法 G[<R>]生成 V={a,b}上的所有正则表达式。

2）考察句子 ab*，它有 2 棵不同的推导树，如图 1.9 所示，所以文法 G[<R>]是二义性文法。

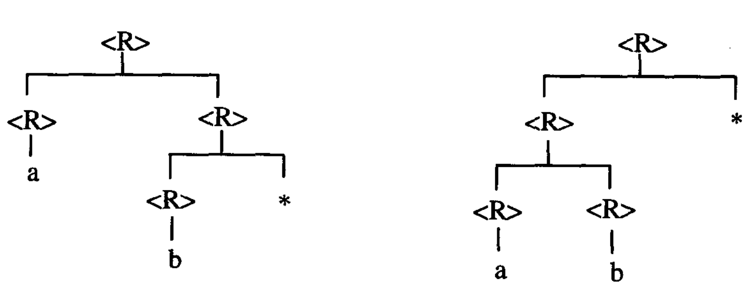

图 1.9 句子 ab*的 2 棵不同的推导树