《编译原理》课后习题

第 **1** 章引论

第 **1** 题解释下列术语：

（1） 编译程序：如果源语言为高级语言，目标语言为某台计算机上的汇编语言或机器语

言，则此翻译程序称为编译程序。

（2） 源程序：源语言编写的程序称为源程序。

（3） 目标程序：目标语言书写的程序称为目标程序。

（4） 编译程序的前端：它由这样一些阶段组成：这些阶段的工作主要依赖于源语言而与

目标机无关。通常前端包括词法分析、语法分析、语义分析和中间代码生成这些阶

段，某些优化工作也可在前端做，也包括与前端每个阶段相关的出错处理工作和符

号表管理等工作。

（5） 后端：指那些依赖于目标机而一般不依赖源语言，只与中间代码有关的那些阶段，

即目标代码生成，以及相关出错处理和符号表操作。

（6） 遍：是对源程序或其等价的中间语言程序从头到尾扫视并完成规定任务的过程。

第 **2** 题

一个典型的编译程序通常由哪些部分组成？各部分的主要功能是什么？并画出编译程

序的总体结构图。

答案：一个典型的编译程序通常包含 8 个组成部分，它们是词法分析程序、语法分析程序、语义分析程序、中间代码生成程序、中间代码优化程序、目标代码生成程序、表格管理程序和错误处理程序。其各部分的主要功能简述如下。

词法分析程序：输人源程序，拼单词、检查单词和分析单词，输出单词的机内表达形式。

语法分析程序：检查源程序中存在的形式语法错误，输出错误处理信息。

语义分析程序：进行语义检查和分析语义信息，并把分析的结果保存到各类语义信息表

中。

中间代码生成程序：按照语义规则，将语法分析程序分析出的语法单位转换成一定形式

的中间语言代码，如三元式或四元式。

中间代码优化程序：为了产生高质量的目标代码，对中间代码进行等价变换处理。

目标代码生成程序：将优化后的中间代码程序转换成目标代码程序。

表格管理程序：负责建立、填写和查找等一系列表格工作。表格的作用是记录源程序的

各类信息和编译各阶段的进展情况，编译的每个阶段所需信息多数都从表格中读取，产生的

中间结果都记录在相应的表格中。可以说整个编译过程就是造表、查表的工作过程。需要指

出的是，这里的“表格管理程序”并不意味着它就是一个独立的表格管理模块，而是指编译

程序具有的表格管理功能。

错误处理程序：处理和校正源程序中存在的词法、语法和语义错误。当编译程序发现源

程序中的错误时，错误处理程序负责报告出错的位置和错误性质等信息，同时对发现的错误

进行适当的校正（修复），目的是使编译程序能够继续向下进行分析和处理。

第 **3** 题何谓翻译程序、编译程序和解释程序？它们三者之间有何种关系？

答案：翻译程序是指将用某种语言编写的程序转换成另一种语言形式的程序的程序，如编译程序和汇编程序等。

编译程序是把用高级语言编写的源程序转换（加工）成与之等价的另一种用低级语言编

写的目标程序的翻译程序。

解释程序是解释、执行高级语言源程序的程序。解释方式一般分为两种：一种方式是，

源程序功能的实现完全由解释程序承担和完成，即每读出源程序的一条语句的第一个单词，

则依据这个单词把控制转移到实现这条语句功能的程序部分，该部分负责完成这条语句的功

能的实现，完成后返回到解释程序的总控部分再读人下一条语句继续进行解释、执行，如此

反复；另一种方式是，一边翻译一边执行，即每读出源程序的一条语句，解释程序就将其翻

译成一段机器指令并执行之，然后再读人下一条语句继续进行解释、执行，如此反复。无论

是哪种方式，其加工结果都是源程序的执行结果。目前很多解释程序采取上述两种方式的综

合实现方案，即先把源程序翻译成较容易解释执行的某种中间代码程序，然后集中解释执行

中间代码程序，最后得到运行结果。

广义上讲，编译程序和解释程序都属于翻译程序，但它们的翻译方式不同，解释程序是

边翻译（解释）边执行，不产生目标代码，输出源程序的运行结果。而编译程序只负责把源

程序翻译成目标程序，输出与源程序等价的目标程序，而目标程序的执行任务由操作系统来

完成，即只翻译不执行。

第 **4** 题对下列错误信息，请指出可能是编译的哪个阶段（词法分析、语法分析、语义分析、

代码生成）报告的。

（**1**） **else** 没有匹配的**if**

（**2**） 数组下标越界

（**3**） 使用的函数没有定义

（**4**） 在数中出现非数字字符

答案：（1） 语法分析

（2） 语义分析

（3） 语法分析

（4） 词法分析

第**3** 章 文法和语言

第**1** 题文法**G**＝**({A,B,S},{a,b,c},P,S)**其中**P** 为：

**S→Ac|aB**

**A→ab**

**B→bc**

写出**L(G[S])**的全部元素。

答案：L(G[S])={abc}

第**2** 题文法**G[N]**为：

**N**→**D|ND**

**D**→**0|1|2|3|4|5|6|7|8|9**

**G[N]**的语言是什么？

答案**:**G[N]的语言是V+。V={0,1,2,3,4,5,6,7,8,9}

N=>ND=>NDD.... =>NDDDD...D=>D......D

第３题为只包含数字、加号和减号的表达式，例如**9-2**＋**5**，**3-1**，７等构造一个文法。

答案：G[S]:

S->S+D|S-D|D

D->0|1|2|3|4|5|6|7|8|9

第**4** 题已知文法**G[Z]**：**Z**→**aZb|ab** 写出**L(G[Z])**的全部元素。

答案：Z=>aZb=>aaZbb=>aaa..Z...bbb=> aaa..ab...bbb

L(G[Z])={anbn|n>=1}

第**5** 题写一文法，使其语言是偶正整数的集合。 要求：**(1)** 允许**0** 打头；**(2)**不允许**0** 打头。

答案：(1)允许0 开头的偶正整数集合的文法

E→NT|D

T→NT|D

N→D|1|3|5|7|9

D→0|2|4|6|8

(2)不允许0 开头的偶正整数集合的文法

E→NT|D

T→FT|G

N→D|1|3|5|7|9

D→2|4|6|8

F→N|0

G→D|0

第**6** 题已知文法**G**：

**<**表达式**>::=<**项**>**｜**<**表达式**>**＋**<**项**>**

**<**项**>::=<**因子**>**｜**<**项**>\*<**因子**>**

**<**因子**>::=**（**<**表达式**>**）｜**i**

试给出下述表达式的推导及语法树。

（1）**i**（2）**（i）**（3）**i\*i** （4）**i\*i +i** （5）**i+(i+i)**（6）**i+i\*i**

第**7** 题证明下述文法**G[**〈表达式〉**]**是二义的。

〈表达式〉∷**=a|(**〈表达式〉**)|**〈表达式〉〈运算符〉〈表达式〉

〈运算符〉∷**=+|-|\*|/**

答案：可为句子a+a\*a 构造两个不同的最右推导:

最右推导1 〈表达式〉 =>〈表达式〉〈运算符〉〈表达式〉

=>〈表达式〉〈运算符〉a

=>〈表达式〉\* a

=>〈表达式〉〈运算符〉〈表达式〉\* a

=>〈表达式〉〈运算符〉a \* a

=>〈表达式〉+ a \* a

=>a + a \* a

最右推导2 〈表达式〉=>〈表达式〉〈运算符〉〈表达式〉

=>〈表达式〉〈运算符〉〈表达式〉〈运算符〉〈表达式〉

=>〈表达式〉〈运算符〉〈表达式〉〈运算符〉 a

=>〈表达式〉〈运算符〉〈表达式〉 \* a

=>〈表达式〉〈运算符〉a \* a

=>〈表达式〉+ a \* a

=>a + a \* a

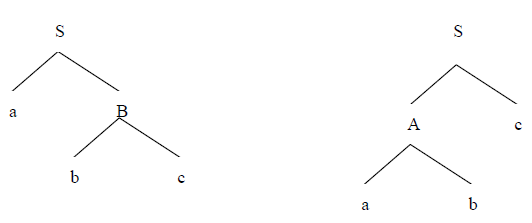
第**8** 题文法**G[S]**为：**S**→**Ac|aB** **A**→**ab** **B**→**bc**

该文法是否为二义的？为什么？

答案：对于串abc(1)S=>Ac=>abc (2)S=>aB=>abc

即存在两不同的最右推导。所以，该文法是二义的。

或者：对输入字符串abc，能构造两棵不同的语法树，所以它是二义的。



第**9** 题

考虑下面上下文无关文法：**S**→**SS\*|SS+|a**

**(1)**表明通过此文法如何生成串**aa+a\***，并为该串构造语法树。 **(2)G[S]**的语言是什么？

答案：(1)此文法生成串aa+a\*的最右推导如下

S=>SS\*=>SS\*=>Sa\*=>SS+a\*=>Sa+a\*=>aa+a\*

(2)该文法生成的语言是：\*和+的后缀表达式，即逆波兰式。

第**10** 题文法**S**→**S(S)S|ε** **(1)** 生成的语言是什么？**(2)** 该文法是二义的吗？说明理由。

答案：（１） 嵌套的括号

（２） 是二义的，因为对于（）（）可以构造两棵不同的语法树。

第**11** 题令文法**G[E]**为：**E→T|E+T|E-T** **T→F|T\*F|T/F** **F→(E)|i**

证明**E+T\*F** 是它的一个句型，指出这个句型的所有短语、直接短语和句柄。

答案：此句型对应语法树如右，故为此文法一个句型。

或者：因为存在推导序列: E=>E+T=>E+T\*F，所以 E+T\*F 句型

此句型相对于E 的短语有:E+T\*F；相对于T 的短语有T\*F

直接短语为：T\*F 句柄为：T\*F

第**13** 题一个上下文无关文法生成句子**abbaa** 的推导树如下：

**(1)**给出串**abbaa** 最左推导、最右推导。**(2)**该文法的产生式集合**P** 可能有哪些元素？

**(3)**找出该句子的所有短语、直接短语、句柄。

答案：(1)串abbaa 最左推导:

S=>ABS=>aBS=>aSBBS=>aBBS=>abBS=>abbS=>abbAa=>abbaa

最右推导：S=>ABS=>ABAa=>ABaa=>ASBBaa=>ASBbaa=>ASbbaa=>Abbaa=>abbaa

(2)产生式有：S→ABS |Aa|ε A→a B→SBB|b

可能元素有：ε aa ab abbaa aaabbaa ……

(3)该句子的短语有：

a 是相对A 的短语

ε 是相对S 的短语

b 是相对B 的短语

εbb 是相对B 的短语

aa 是相对S 的短语

aεbbaa 是相对S 的短语

直接短语有：a ε b

句柄是：a

第**14** 题给出生成下述语言的上下文无关文法：

（**1**）**{ anbnambm| n**，**m>=0}**

（**2**）**{ 1n0m 1m0n| n**，**m>=0}**

（**3**）**{WaWr|W** 属于**{0|a}\***，**Wr** 表示**W**的逆**}**

答案：

（１）S→AA

A→aAb|ε

（２）S→1S0|A

A→0A1|ε

（３）S→0S0|1S1|ε

第**16** 题给出生成下述语言的三型文法：

**(1){an|n >=0 }**

**(2) { anbm|n,m>=1 }**

**(3){anbmck|n,m,k>=0 }**

答案：(1) S→aS|ε

(2)S→aA

A→aA|B

B→bB|b

(3)A→aA|B

B→bB|C

C→cC|ε

第**18** 题解释下列术语和概念：

答案：（１）字母表：是一个非空有穷集合。

（２）串：符号的有穷序列。字：字母表中的元素。

句子：如果 Z-+ x , x ∈V \*T 则称 x 是文法 G 的一个句子。

（３）语言：它是由句子组成的集合，是由一组记号所构成的集合。程序设计的语言就是所

有该语言的程序的全体。语言可以看成在一个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。

语法：表示构成语言句子的各个记号之间的组合规律。程序的结构或形式。

语义：表示按照各种表示方法所表示的各个记号的特定含义。语言所代表的含义。

第**4** 章 词法分析

第**5** 章 自顶向下语法分析方法

第**1** 题对文法**G[S]**

**S→a|**∧**|(T)**

**T→T,S|S**

**(1)** 给出**(a,(a,a))**和**(((a,a),**∧**,(a)),a)**的最左推导。

**(2)** 对文法**G**，进行改写，然后对每个非终结符写出不带回溯的递归子程序。

**(3)** 经改写后的文法是否是**LL(1)**的**?**给出它的预测分析表。

**(4)** 给出输入串**(a,a)#**的分析过程，并说明该串是否为**G** 的句子。

答案：(1) 对(a,(a,a)的最左推导为：

S=>（t）

=> (T,S)

=> (S,S)

=> (a,S)

=> (a,(T))

=> (a,(T,S))

=> (a,(S,S))

=> (a,(a,S))

=> (a,(a,a))

对(((a,a),∧,(a)),a) 的最左推导为：

S => (T)

=> (T,S)

=> (S,S)

=> ((T),S)

=> ((T,S),S)

=> ((T,S,S),S)

=> ((S,S,S),S)

=> (((T),S,S),S)

=> (((T,S),S,S),S)

=> (((S,S),S,S),S)

=> (((a,S),S,S),S)

=> (((a,a),S,S),S)

=> (((a,a),∧,S),S)

=> (((a,a),∧,(T)),S)

=> (((a,a),∧,(S)),S)

=> (((a,a),∧,(a)),S)

=> (((a,a),∧,(a)),a)

(2) 改写文法为：

0) S→a

1) S→∧

2) S→( T )

3) T→S N

4) N→, S N

5) N→ε

非终结符 FIRST 集 FOLLOW 集

S {a,∧,(} {#,,,)}

T {a,∧,(} {)}....

N {,,ε} {)}....

对左部为N 的产生式可知：

FIRST （→, S N）={，}

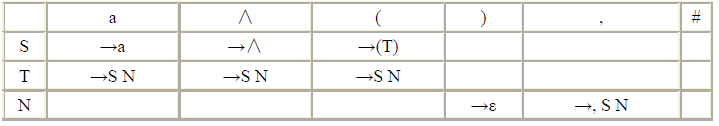
FIRST （→ε）={ε}

FOLLOW （N）={)}

由于SELECT(N →, S N)∩SELECT(N →ε) ={，}∩ { )}=

所以文法是LL(1)的。

预测分析表（Predicting Analysis Table）



也可由预测分析表中无多重入口判定文法是LL(1)的。

(3) 对输入串（a,a）#的分析过程为：



可见输入串（a,a）#是文法的句子。

第**3** 题

已知文法**G[S]**：

**S→MH|a**

**H→LSo|ε**

**K→dML|ε**

**L→eHf**

**M→K|bLM**

判断**G** 是否是**LL(1)**文法，如果是，构造**LL(1)**分析表。

答案：

文法展开为：

0） S→M H

1） S→a

2） H→L S o

3） H→ε

4） K→d M L

5） K→ε

6） L→e H f

7） M→K

8） M→b L M

非终结符 FIRST 集 FOLLOW 集

S {a,d,b,ε,e} {#,o}

M {d,ε,b} {e,#,o}

H {ε,e} {#,f,o}

L {e} {a,d,b,e,o,#}

K {d,ε} {e,#,o}

对相同左部的产生式可知：

SELECT(S→M H)∩SELECT(S→a) ={ d,b ,e，#,o }∩ { a }=

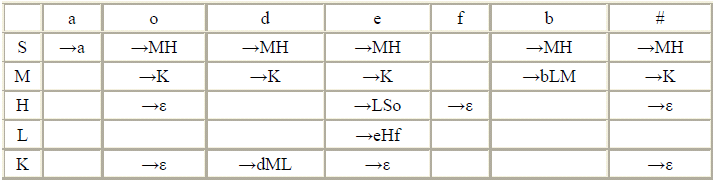
SELECT(H→L S o)∩SELECT(H→ε) ={ e }∩ { #,f,o }=

SELECT(K→d M L)∩SELECT(K→ε) ={ d }∩ { e,#,o }=

SELECT(M→K)∩SELECT(M→b L M) ={ d，e,#,o }∩ { b }=

所以文法是LL(1)的。

预测分析表：



由预测分析表中无多重入口也可判定文法是LL(1)的。

第**7** 题

对于一个文法若消除了左递归，提取了左公共因子后是否一定为**LL(1)**文法**?**试对下面

文法进行改写，并对改写后的文法进行判断。

**(**１**)A**→**baB|**ε

**B**→**Abb|a**

**(2) A→aABe|a**

**B→Bb|d**

**(3) S→Aa|b**

**A→SB**

**B→ab**

答案：(１)先改写文法为：

0) A→baB

1) A→ε

2) B→baBbb

3) B→bb

4) B→a

再改写文法为：

0) A→baB

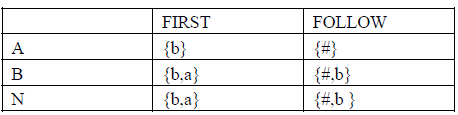
1) A→ε

2) B→bN

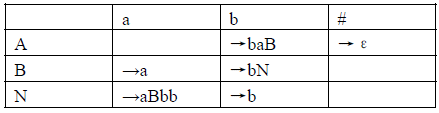
3) B→a

4) N→aBbb

5) N→b



预测分析表：



由预测分析表中无多重入口判定文法是LL(1)的。

（2） 文法：

A→aABe|a

B→Bb|d

提取左公共因子和消除左递归后文法变为：

0) A→a N

1) N→A B e

2) N→ε

3) B→d N1

4) N1→b N1

5) N1→ε

非终结符 FIRST 集 FOLLOW 集

A {a} {#,d}

B {d} {e}

N {a,ε} {#,d}

N1 {b,ε} {e}

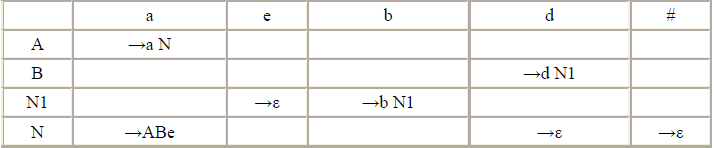
对相同左部的产生式可知：

SELECT(N→A B e)∩SELECT(N→ε) ={ a }∩ {#,d }=

SELECT(N1→b N1)∩SELECT(N1→ε) ={ b }∩ { e }=

所以文法是LL(1)的。

预测分析表（Predicting Analysis Table）



也可由预测分析表中无多重入口判定文法是LL(1)的。

（3）文法：

S→Aa|b

A→SB

B→ab

第**1** 种改写：用A 的产生式右部代替S 的产生式右部的A 得：

S→SBa|b

B→ab

消除左递归后文法变为：

0) S→b N

1) N→B a N

2) N→ε

3) B→a b

非终结符 FIRST 集 FOLLOW 集

S {b} {#}

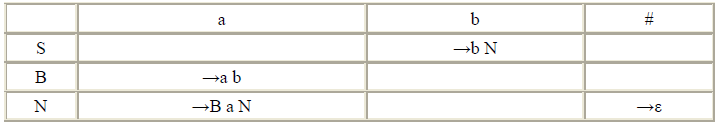
B {a} {a}

N {ε,a} {#}

对相同左部的产生式可知：SELECT(N→B a N)∩SELECT(N→ε) ={ a }∩ {# }=

所以文法是LL(1)的。

预测分析表（Predicting Analysis Table）



也可由预测分析表中无多重入口判定文法是LL(1)的。

第**2** 种改写：

用S 的产生式右部代替A 的产生式右部的S 得：

S→Aa|b

A→AaB|bB

B→ab

消除左递归后文法变为：

0) S→A a

1) S→b

2) A→b B N

3) N→a B N

4) N→ε

5) B→a b

非终结符 FIRST 集 FOLLOW 集

S {b} {#}

A {b} {a}

B {a} {a}

N {a,ε} {a}

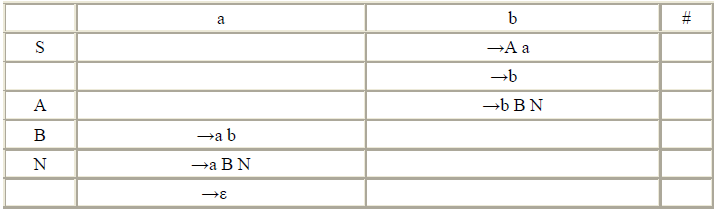
对相同左部的产生式可知：

SELECT(S→A a)∩SELECT(S→b) ={ b }∩ { b }={ b }≠

SELECT(N→a B N)∩SELECT(N→ε) ={ a }∩ { a }={ a }≠

所以文法不是LL(1)的。

预测分析表：



也可由预测分析表中含有多重入口判定文法不是LL(1)的。

第**6** 章 自底向上优先分析

第**1** 题已知文法**G[S]**为：**S→a|**∧**|(T)**

**T→T,S|S**

**(1)** 计算**G[S]**的**FIRSTVT** 和**LASTVT**。

**(2)** 构造**G[S]**的算符优先关系表并说明**G[S]**是否为算符优先文法。

**(3)** 计算**G[S]**的优先函数。

**(4)** 给出输入串**(a,a)#**和**(a,(a,a))#**的算符优先分析过程。

答案：

文法展开为：

S→a

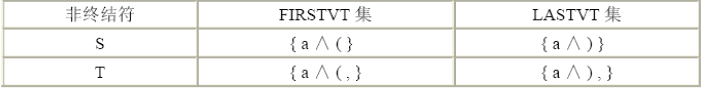
S→∧

S→(T)

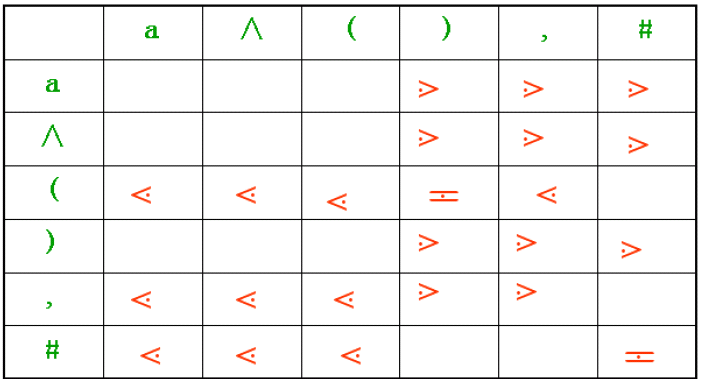
T→T,S

T→S

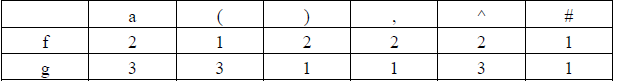
(1) FIRSTVT - LASTVT 表：



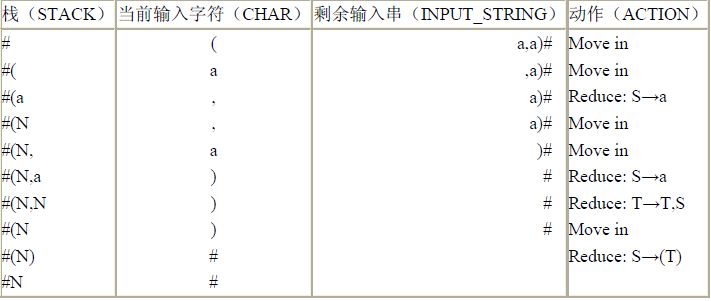
(2) 算符优先关系表：



(3)对应的算符优先函数为：

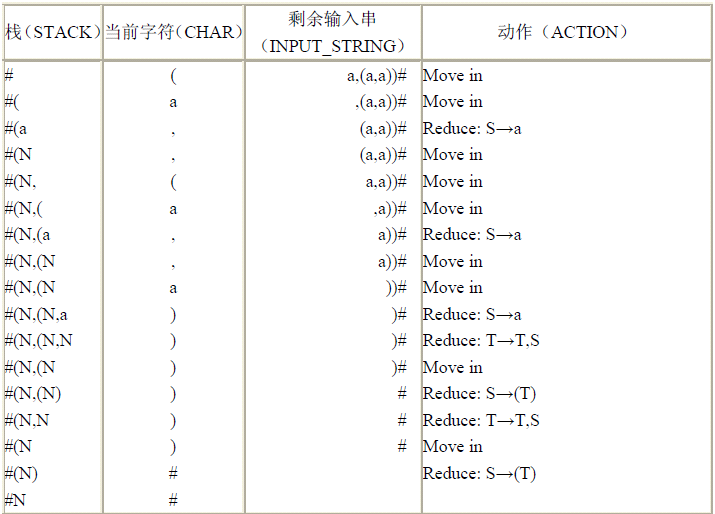


（4）对输入串（a,a）#的算符优先分析过程为



Success!

对输入串（a,(a,a)）# 的算符优先分析过程为：



Success!

第**2** 题已知文法**G[S]**为：**S→a|**∧**|(T)**

**T→T,S|S**

**(1)** 给出**(a,(a,a))**和**(a,a)**的最右推导，和规范归约过程。

**(2)** 将**(1)**和题**1** 中的**(4)**进行比较给出算符优先归约和规范归约的区别。

答案：（１）(a,a)的最右推导过程为：

S => (T)

=> (T,S)

=> (T,a)

=> (S,a)

=> (a,a)

=>（a,(a,a)）的最右推导过程为：

S=> (T)

=> (T,S)

=> (T,(T))

=> (T,(T,S))

=> (T,(T,a))

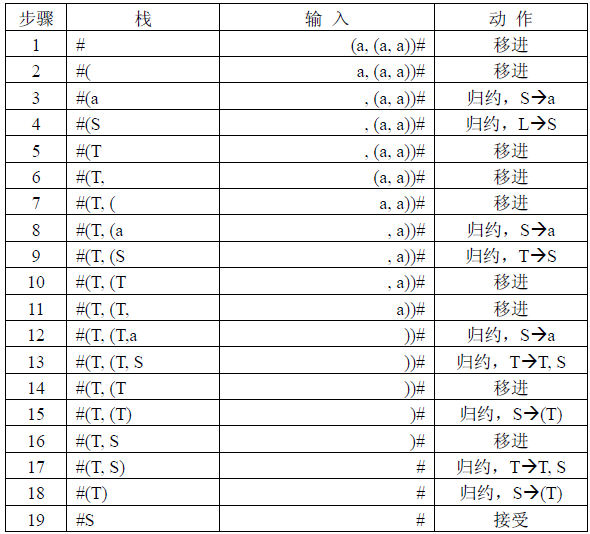
=> (T,(S,a))

=> (T,(a,a))

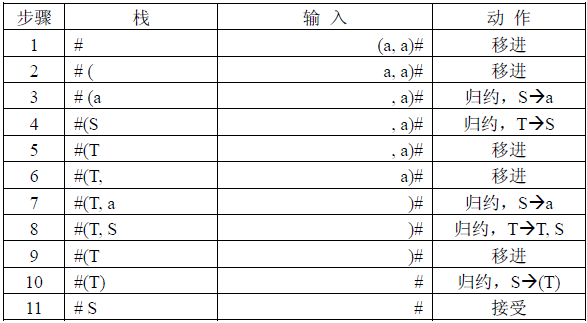
=> (S,(a,a))

=> (a,(a,a))

(a,(a,a))的规范归约过程：



(a,a)的规范归约过程：



(2)算符优先文法在归约过程中只考虑终结符之间的优先关系从而确定可归约串，而与

非终结符无关，只需知道把当前可归约串归约为某一个非终结符，不必知道该非终结符的名

字是什么，因此去掉了单非终结符的归约。

规范归约的可归约串是句柄，并且必须准确写出可归约串归约为哪个非终结符。

第３题：有文法**G[S]**：

**S**->**V**

**V**->**T|ViT**

**T**->**F|T+F**

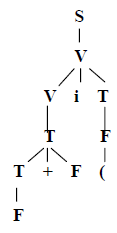
**F**->**)V\*|(**

**(1)** 给出**(+(i(**的规范推导。

**(2)** 指出句型 **F+Fi(**的短语，句柄，素短语。

**(3) G[S]**是否为**OPG**？若是，给出**(1)**中句子的分析过程。

答案：

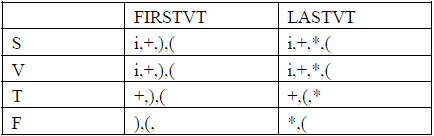


(1)S=>V=>ViT=>ViF=>Vi(=>T i(=>T+F i(=>T+( i(=>F+( i(=>(+( i(

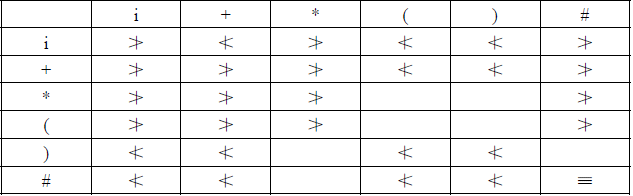
(2)句型F+Fi(的语法树：

短语：F，F+F，(，F+Fi( 句柄：F 素短语：(

(3)FIRSTVT 和LASTVT



算符优先关系



因为该文法是OP，同时任意两个终结符的优先关系唯一，所以该文法为OPG。

(+(i(的分析过程



第４题文法G［S］为：

S→S;G｜G

G→G(T)｜H

H→a｜(S)

T→T+S｜S

（１） 构造G［S］的算符优先关系表，并判断G［S］是否为算符优先文法。

（２） 给出句型a(T+S);H;(S)的短语、句柄、素短语和最左素短语。

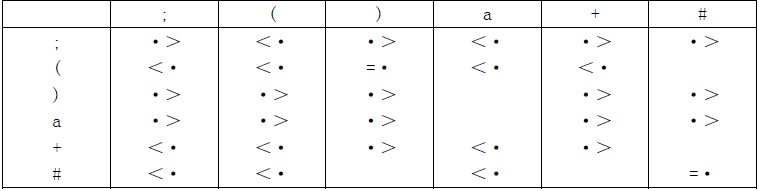
（３） 给出a;(a+a)和(a+a)的分析过程，说明它们是否为G［S］的句子。

（４） 给出（３）中输入串的最右推导，分别说明两输入串是否为G[S]的句子。

（５） 由（３）和（４）说明了算符优先分析的哪些缺点。

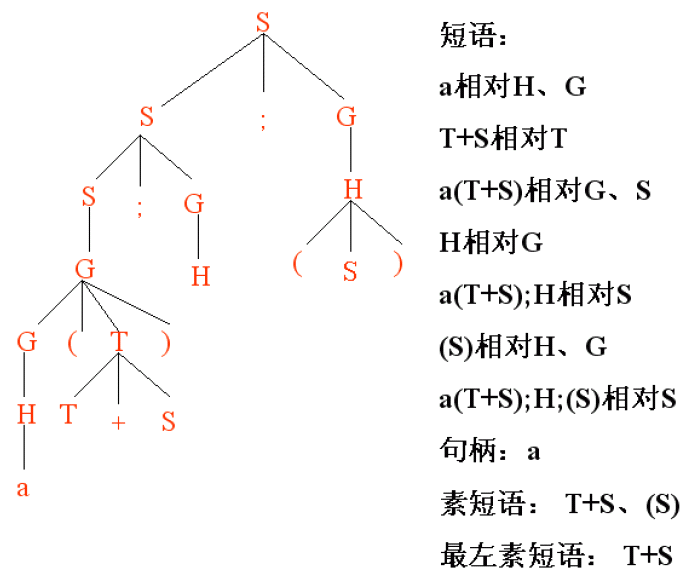
（６） 算符优先分析过程和规范归约过程都是最右推导的逆过程吗？

答案：（1）构造文法G［S］的算符优先关系矩阵：

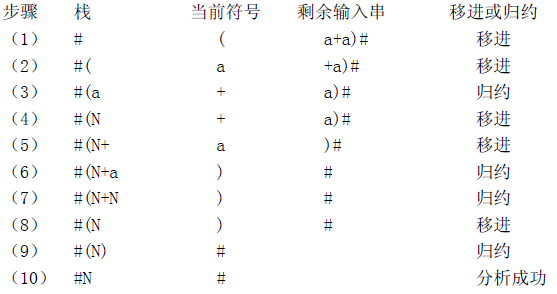


在上表中可看出终结符之间的优先关系是唯一的，或称G［S］的算符优先关系矩阵不含多重入口，因此，G［S］是一个算符优先文法。

（２）



（3）对输入串(a+a)＃的分析过程如下：



说明是它的句子。

（４）试用规范推导：

S⇒ G⇒H⇒（S）由此往下S 不可能推导出a+a，所以 (a+a)不是G［S］的句子。

（5）结果说明：由于算符优先分析法去掉了单非终结符之间的归约，尽管在分析过程中，

当决定是否为句柄时采取一些检查措施，但仍难完全避免把错误的句子得到正确的归约。

（６）算符优先分析过程不是最右推导的逆过程。规范归约过程是最右推导的逆过程。

第7 章 LR 分析

第1 题已知文法A→aAd|aAb|ε

判断该文法是否是SLR(1)文法，若是构造相应分析表，并对输入串ab#给出分析过程。

答案：文法：A→aAd|aAb|ε

拓广文法为G′，增加产生式S′→A

若产生式排序为：

0 S' →A

1 A →aAd

2 A →aAb

3 A →ε

由产生式知：

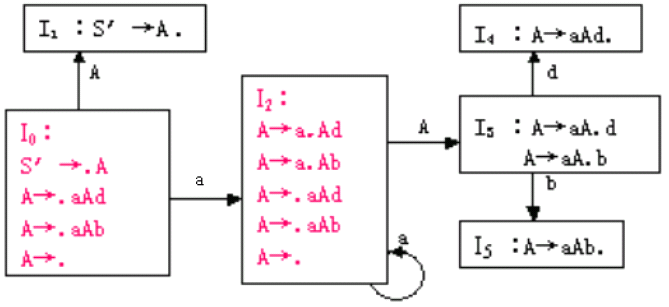
First (S' ) = {ε,a}

First (A ) = {ε,a}

Follow(S' ) = {#}

Follow(A ) = {d,b,#}

G′的LR(0)项目集族及识别活前缀的DFA 如下图所示：

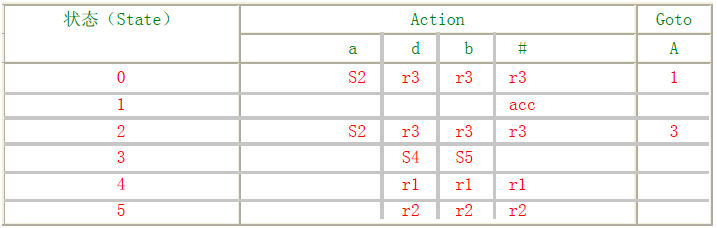


在I0 中：A →.aAd 和A →.aAb 为移进项目，A →.为归约项目，存在移进-归约冲突，因此所给文法不是LR(0)文法。

在I0、I2 中：Follow(A) ∩{a}= {d，b，#} ∩{a}=

所以在I0、I2 中的移进-归约冲突可以由Follow 集解决，所以G 是SLR(1)文法。

构造的SLR(1)分析表如下：题目1 的SLR(1)分析表



题目1 对输入串ab#的分析过程



分析成功，说明输入串ab 是文法的句子。

第2 题若有定义二进制数的文法如下：

S→L·L|L

L→LB|B

B→0|1

(1) 试为该文法构造LR 分析表，并说明属哪类LR 分析表。

(2) 给出输入串101.110 的分析过程。

答案：文法：

S→L.L|L

L→LB|B

B→0|1

拓广文法为G′，增加产生式S′→S

若产生式排序为：

0 S' →S

1 S →L.L

2 S →L

3 L →LB

4 L →B

5 B →0

6 B →1

由产生式知：

First (S' ) = {0,1}

First (S ) = {0,1}

First (L ) = {0,1}

First (B ) = {0,1}

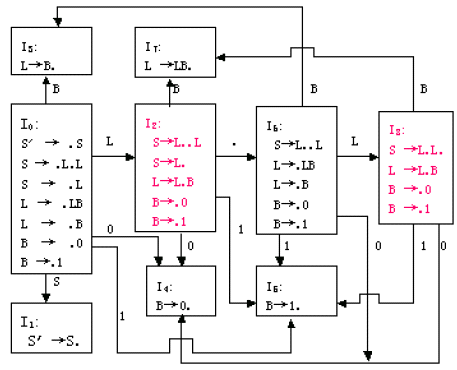
Follow(S' ) = {#}

Follow(S ) = {#}

Follow(L ) = {.,0,1,#}

Follow(B ) = {.,0,1,#}

G′的LR(0)项目集族及识别活前缀的DFA 如下图所示：

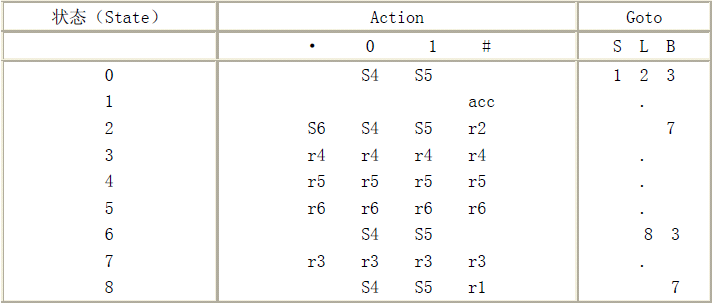


在I2 中：B →.0 和 B →.1 为移进项目，S →L.为归约项目，存在移进-归约冲突，因此所给文法不是LR(0)文法。

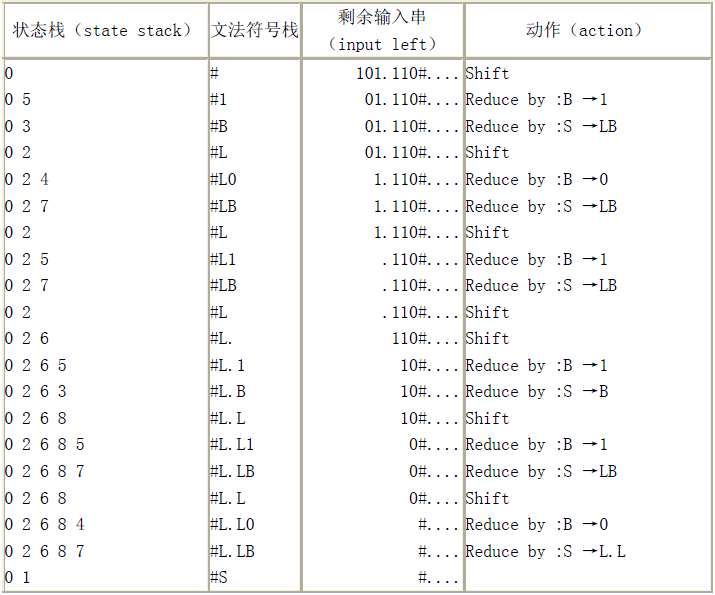
在I2、I8 中：Follow(s) ∩{0，1}= { #} ∩{0，1}=

所以在I2 、I8 中的移进-归约冲突可以由Follow 集解决，所以G 是SLR(1)文法。

构造的SLR(1)分析表如下：题目2 的SLR(1)分析表



题目2 对输入串101.110#的分析过程



分析成功，说明输入串101.110 是题目2 文法的句子。

题目3考虑文法S-> A S | b

A -> S A | a

（1）列出这个文法的所有LR(0)项目

（2）按（1）列出的项目构造识别这个文法活前缀的NFA，把这个NFA 确定化为DFA，说明

这个DFA 的所有状态全体构成这个文法的LR(0)规范族

（3）这个文法是SLR 的吗？若是，构造出它的SLR 分析表

（4）这个文法是LALR 或LR(1)的吗？

答：

（1）令拓广文法G’为

(0) S’ -> S

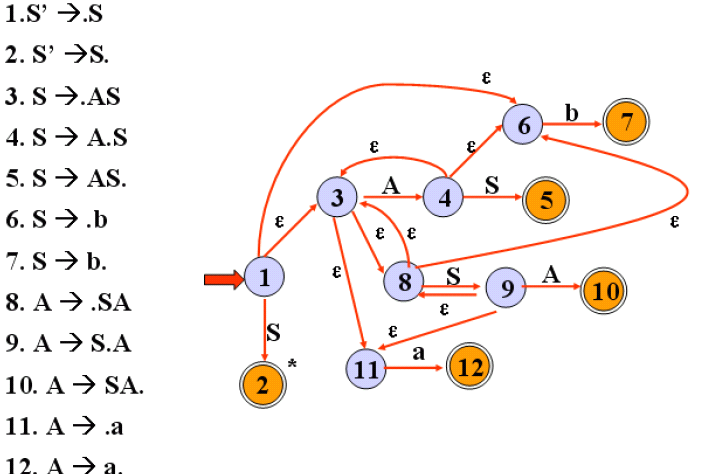
(1)S-> A S

(2)S-> b

(3)A -> S A

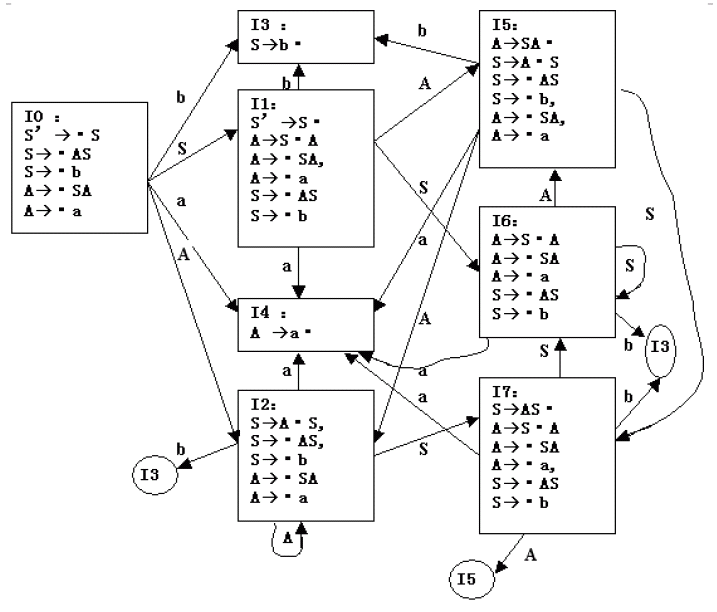
(4)A -> a

其LR(0)项目：



(2) 识别这个文法活前缀的NFA 如上图所示：

确定化为DFA 如下图所示：



（3）因为I5 中：FOLLOW（A）∩{a，b}≠Ф

I7 中：FOLLOW（S）∩{a，b}≠Ф

所以，该文法不是SLR（1）文法。

（4）LR(1)项目集族为：

I0 :

S’->·S, #

S ->·AS, #

S ->·b, #

S ->·SA, a / b

A ->·a, a / b

I1 : S’ -> S·,#

A -> S·A, a / b

A ->·a, a / b

A -> ·SA, a / b

S ->·AS, a / b

S ->·b, a / b

I2 : S ->A·S, #

S ->·b, #

S ->·AS, #

A ->·SA, a / b

A A ->·a, a / b

I3 : S -> b·, #

I4 : A -> a·, a / b

I5 : A -> SA·, a / b

S -> A·S, a / b

S -> ·AS, a / b

S ->·b, a/ b

A ->·SA, a / b

A ->·a, a / b

I6 : A -> S·A,a/b

A ->·SA, a / b

A ->·a, a / b

S->·AS, a / b

S ->·b, a / b

I7: S -> b·, a / b

I8 : S -> AS·, #

A -> S·A, a / b

A ->·SA, a / b

A ->·a, a / b

S ->·AS, a / b

S ->·b, a / b

I9 :

S ->A·S, #

S ->·AS, #

S ->·b, #

S ->·SA, a / b

A ->·a, a / b

I10 :

S ->AS·, a/b

A ->S·A, a/b

A ->·S A, a/b

A ->·a, a / b

S ->·b, a/b

S ->·AS, a / b

I11 :

S ->A·S, a/b

S ->·b, a/b

S ->·AS, a / b

A ->·S A, a/b

A ->·a, a / b

I12 :

S ->SA·, a/b

S ->A·S, a/b

S ->·b, a/b

S ->·AS, a / b

A ->·S A, a/b

A ->·a, a / b

因为I5 状态集中存在“归约――移进”冲突，所以不是LR(1)文法，也不是LALR

文法。

第6 题文法G=({U,T,S},{a,b,c,d,e},P,S)

其中P 为:

S→UTa|Tb

T→S|Sc|d

U→US|e

(1) 判断G 是LR(0)，SLR(1)，LALR(1)还是LR(1)，说明理由。

(2) 构造相应的分析表。

答案：文法：

S→UTa|Tb

T→S|Sc|d

U→US|e

拓广文法为G'，增加产生式S'→S

若产生式排序为：

0 S' →S

1 S →UTa

2 S →Tb

3 T →S

4 T →Sc

5 T →d

6 U →US

7 U →e

由产生式知：

First (S' ) = {d,e}

First (S ) = {d,e}

First (U ) = {e}

First (T ) = {d,e}

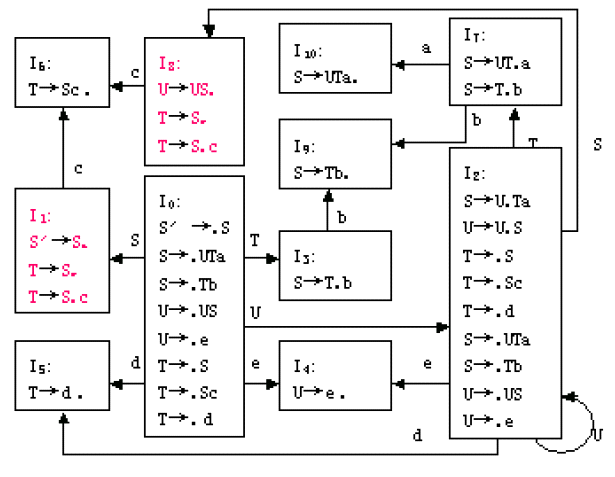
Follow(S' ) = {#}

Follow(S ) = {a,b,c,d,e,#}

Follow(U ) = {d,e}

Follow(T ) = {a,b}

G′的LR(0)项目集族及识别活前缀的DFA 如下图所示：



在I1 中：

S' →S.为接受项目，T →S. 为归约项目，T →S.c 为移进项目，存在接受-归约和移进-归

约冲突，因此所给文法不是LR(0)文法。

在I1 中：

Follow(S') ∩ Follow(T)= { #} ∩{a ，b}=

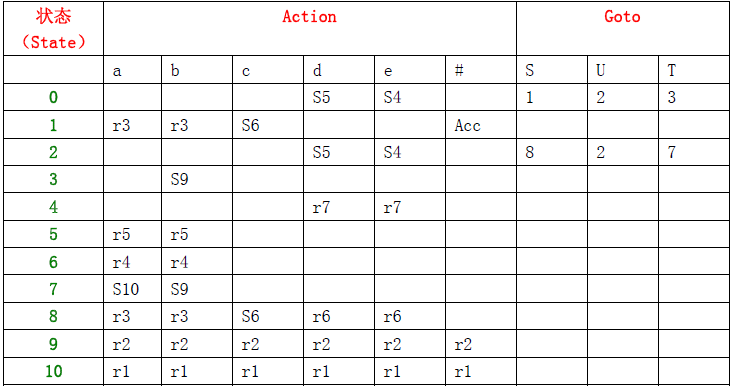
Follow(T) ∩{ c}= { a ，b} ∩{c}=

在I8 中：Follow(U) ∩ Follow(T) ∩{ c}={d,e}∩{ a ，b} ∩{c}=

所以在I1 中的接受-归约和移进-归约冲突与I8 中的移进-归约和归约-归约冲突可以由

Follow 集解决，所以G 是SLR(1)文法。

构造的SLR(1)分析表如下：



第8 题证明文法：

S→A$

A→BaBb|DbDa

B→ε

D→ε是LR(1)但不是SLR(1)。(其中'$'相当于'#')

答案：文法：

A→BaBb|DbDa

B→ε

D→ε

拓广文法为G′，增加产生式S′→A

若产生式排序为：

0 S'→A

1 A →BaBb

2 A →DbDa

3 B →ε

4 D →ε

由产生式知：

First (S' ) = {a,b}

First (A ) = {a,b}

First (B ) = {ε}

First (D ) = {ε}

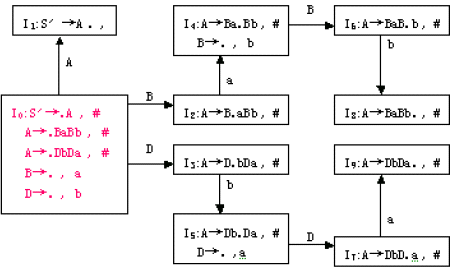
Follow(S' ) = {#}

Follow(A ) = {#}

Follow(B ) = {a,b}

Follow(D ) = {a,b}

G′的LR(1)项目集族及识别活前缀的DFA 如下图所示：



在I0 中：B →.,a 和T →.，b 为归约项目，但它们的搜索符不同，若当前输入符为a 时用产生式B →ε归约，为b 时用产生式D →ε 归约，所以该文法是LR(1)文法。

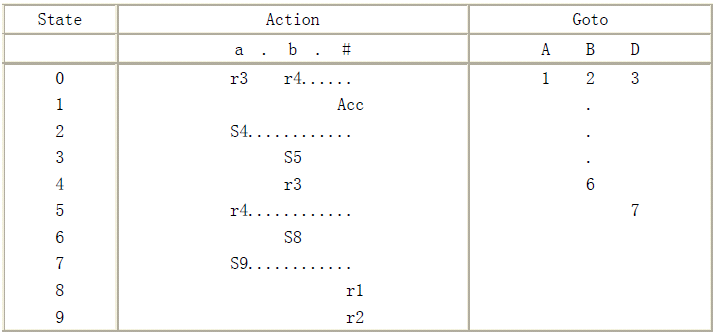
若不看搜索符，在I0 中项目B →.和T →.为归约-归约冲突，而

Follow(B ) ∩Follow(D ) = {a,b} ∩{a,b}≠ ，冲突不能用Follow 集解决，所以该文法

不是SLR(1)文法。

构造的LR(1)分析表如下：

题目4 的LR(1)分析表



第16 题给定文法：S→do S or S|do S|S;S|act

(1) 构造识别该文法活前缀的DFA。

(2) 该文法是LR(0)吗?是SLR(1)吗?说明理由。

(3) 若对一些终结符的优先级以及算符的结合规则规定如下：

a) or 优先性大于 do;

b) ;服从左结合；

c) ;优先性大于 do ；

d) ;优先性大于 or ；

请构造该文法的LR 分析表,并说明LR(0)项目集中是否存在冲突和冲突如何解决的。

答案：首先化简文法,用d 代替do;用o 代替 or;用a 代替 act;文法可写成：

S→dSoS|dS|S;S|a

拓广文法为G'，增加产生式S′→S

若产生式排序为：

0 S' →S

1 S →dSoS

2 S →dS

3 S →S;S

4 S →a

由产生式知：

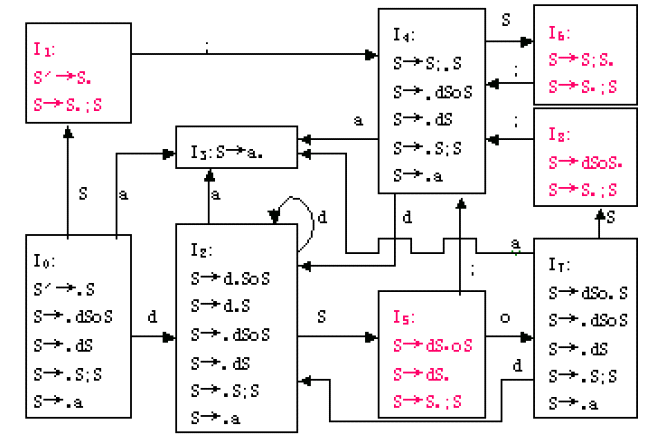
First (S' ) = {d,a}

First (S ) = {d,a}

Follow(S' ) = {#}

Follow(S ) = {o,;,#}

* 识别该文法活前缀的DFA 如下图。



(2) 该文法不是LR(0)也不是SLR(1)因为：

在I5、I6、和I8 中存在移进-归约冲突，因此所给文法不是LR(0)文法。

又由于Follow(S ) = {o, ; ,#}

在I6、和I8 中：Follow(S ) ∩{ ; }={o, ; ,#} ∩{ ; }={;}≠

在I5 中：Follow(S ) ∩{ ; , o }={o , ; ,#} ∩{ ; , o }={; , o }≠

所以该文法也不是SLR(1) 文法。

此外很容易证明所给文法是二义性的，

S`→S→dSoS→ddSoS

S`→S→dS→ddSoS

因此该文法不可能满足LR 文法。

(3) 若对一些终结符的优先级以及算符的结合规则规定如下：

a) or 优先性大于 do;

b) ;服从左结合；

c) ;优先性大于 do ；

d) ;优先性大于 or ；

则：

在I5 中：“or 和;优先性都大于 do”，所以遇输入符o 和; 移进；遇#号归约。

在I6 中：“; 号服从左结合”，所以遇输入符属于Follow(S )的都应归约。

在I8 中：“; 号优先性大于 do ”， 所以遇输入符为;号 移进；遇o 和#号归约。

此外，在I1 中：接受和移进可以不看成冲突，因为接受只有遇#号。

由以上分析，所有存在的移进-归约冲突可用规定的终结符优先级以及算符的结合规则解决，

所构造的LR 分析表如下：

