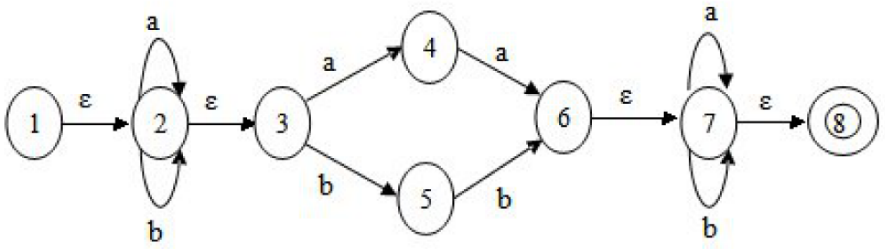
《编译原理》期末复习资料

【题 1】

1.（**a|b**）*（**aa|bb**）（**a|b**）*画出状态转换图。

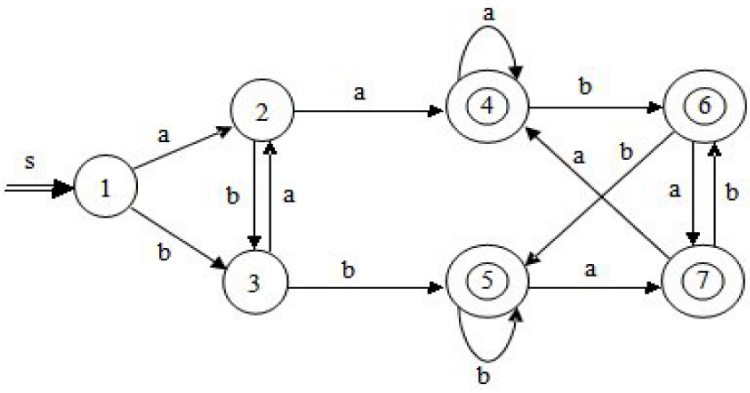


（此题中第一个空输入可以省略，即①可以省略。）

	Ia	Ib
① 1,2,3	2,3,4	2,3,5
② 2,3,4	2,3,4,6,7,8	2,3,5
③ 2,3,5	2,3,4	2,,3,5,6,7,8
④ 2,3,4,6,7,8	2,3,4,6,7,8	2,3,5,7,8
⑤ 2,3,5,6,7,8	2,3,4,7,8	2,3,5,6,7,8
⑥ 2,3,5,7,8	2,3,4,7,8	2,3,5,6,7,8
⑦ 2,3,4,7,8	2,3,4,6,7,8	2,3,5,7,8

	Ia	Ib
1	2	3
2	4	3
3	2	5
4	4	6
5	7	5
6	7	5
7	4	6

新的状态转换图如下：



（判断终结，含有最后一个数字，含有 8 的）

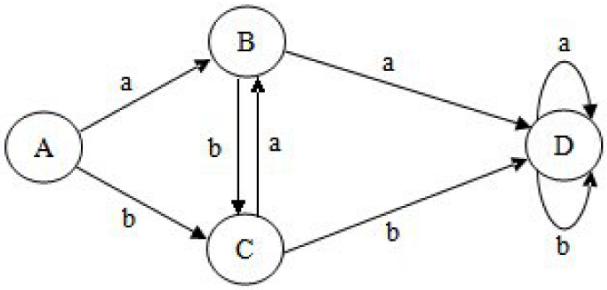
（1）A={1,2,3}，B={4,5,6,7}　Aa={2,4} ×

（2）A={1,3}，B={2}，C={4,5,6,7}　Aa={2}⊂B，Ab={3,5} ×

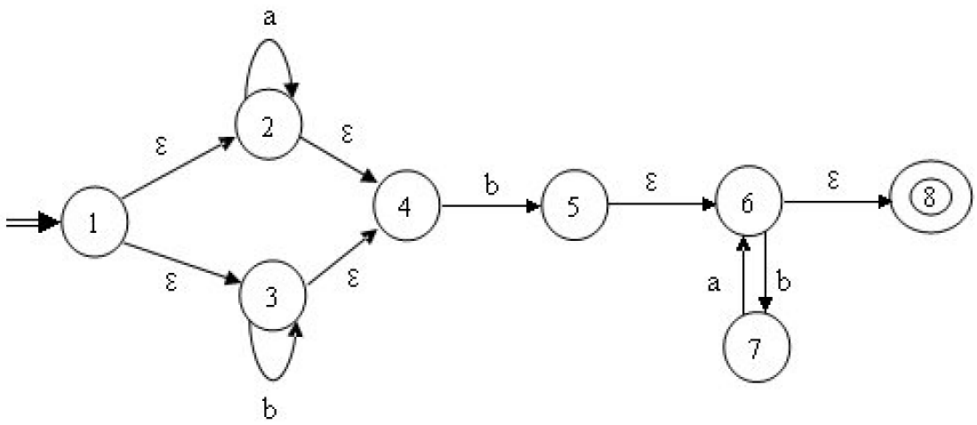
（3）A={1}，B={2}，C={3},D={4,5,6,7}（单元素可以不用看，必有，古先看 D）

Da={4,7}⊂D，Db={5,6}⊂D，Aa={2}⊂B，Ab={3}⊂C，Ba={4}⊂D，Bb={3}⊂C，Ca={2}⊂B，Cb={5}⊂C，则有

	a	b
A	B	C
B	D	C
C	B	D
D	D	D



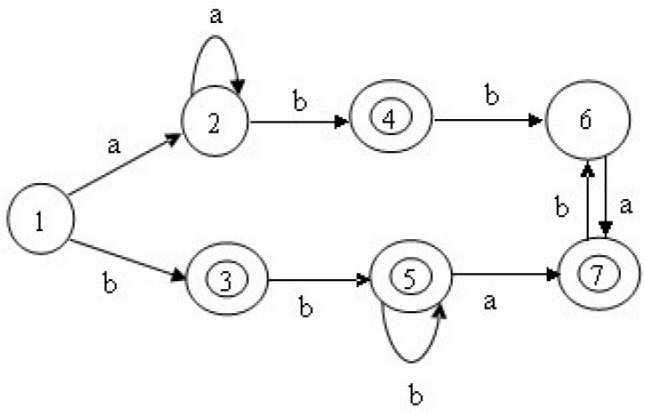
2.（a*|b*）b（ba）*的状态转换图。



	Ia	Ib
① 1,2,3,4	2,4	3,4,5,6,8
② 2,4	2,4	5,6,8
③ 3,4,5,6,8	---	3,4,5,6,7,8
④ 5,6,8	---	7
⑤ 3,4,5,6,7,8	6,8	3,4,5,6,7,8
⑥ 7	6,8	---
⑦ 6,8	---	7

	Ia	Ib
1	2	3
2	2	4
3	---	5
4	---	6
5	7	5
6	7	---
7	---	6

新的状态转换图如下：



化简：（用终结状态与非终结状态，然后输出状态一致分一类）。

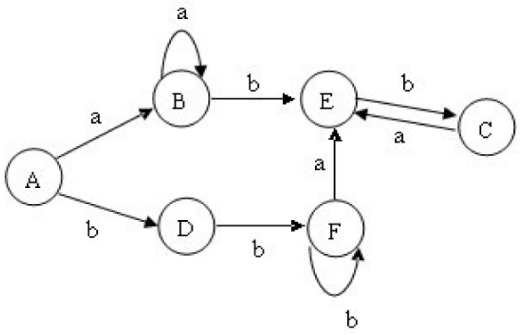
（1）A={1,2,6}，B={3,4,5,7}　Aa={2} ×

（2）A={1,2}，B={6}，C={3,4,7}，D={5}　Cb={5,6} ×（只要有一个不属于任何一个集合，就不行）

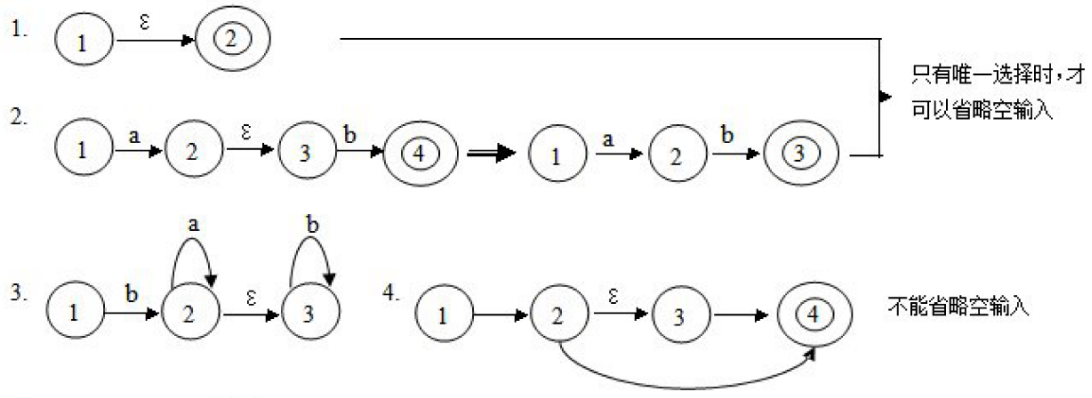
（3）A={1,2}，B={6}，C={3}，D={4,7}，E={5}　Ab={3,4} ×

（4）A={1}，B={2}，C={6}，D={3}，E={4,7}，F={5}　Aa={2}⊂B，Ab={3}⊂D，Ba={2}⊂B，Bb={4}⊂E，Ca={7}⊂E，Db={5}⊂F，Eb={6}⊂C，Fa={7}⊂E，Fb={5}⊂F

	a	b
A	B	D
B	B	E
C	E	---
D	---	F
E	---	C
F	E	F



[注意事项]：



[知识要点]：

◆ 正则表达式：$a^*, a|b, ab, (ab)^*, (a|b)^*, a(a|b)^*, a|a^*b, a|ab^*$

$a^* \Leftrightarrow \{\varepsilon, a, aa, aaa, a\cdots a\}$； $a|b(a或b) \Leftrightarrow \{a, b\}$； $ab(a和b) \Leftrightarrow \{ab\}$；

$(ab)^* \Leftrightarrow \{\varepsilon, ab, abab, ababab, \cdots\cdots\}$ ；

$(a|b)^* \Leftrightarrow (a|b)(a|b)\cdots\cdots(a|b) \Leftrightarrow \{\varepsilon, a, b, aab, aabb, baba, \cdots\cdots\}$

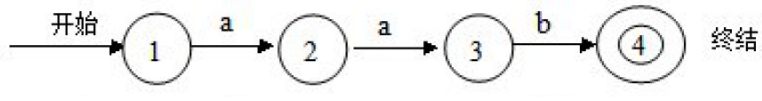
$a(a|b)^*$ 是最左边一个字母一定是 a，其余字母为 a、b 的任意组合，不包括 ε。

$a|a^*b \Leftrightarrow \{a, b, ab, aab, aaab, aaaab, \cdots\}$ ⇔{a 和若干个 a（包括 0 的情形）后跟一个 b 构成的符号串集合}

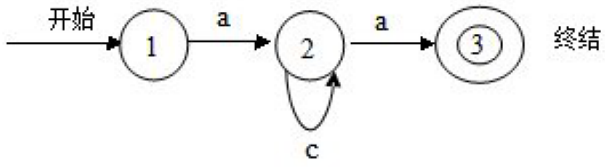
$a|ab^* \Leftrightarrow \{a(\varepsilon|b^*)\} \Leftrightarrow \{a(\varepsilon|b)^*\} \Leftrightarrow ab^* \Leftrightarrow \{a, ab, abb, abbb, abbbb, \cdots\}$ ⇔{a 和 a 后跟若干个（包括 0 的情形）b 构成的符号串集合}

◆ 状态转换图（有穷状态自动机）：

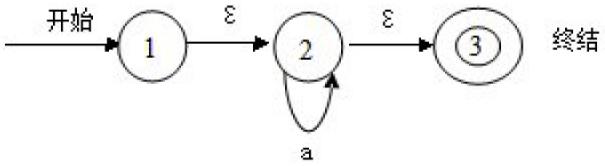
aab



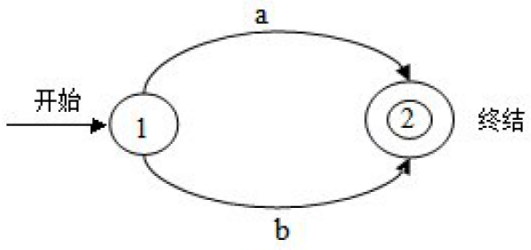
ac^*b



a^*

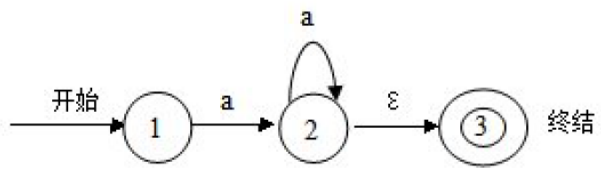


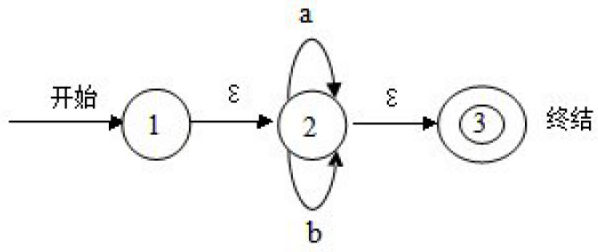
$a|b$



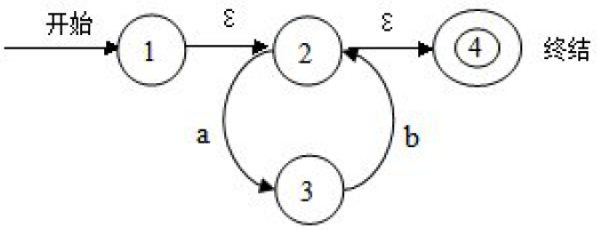
ab

$a^+ = aa^*$



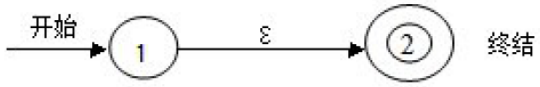


$(a|b)^*$

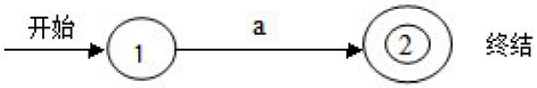


$(ab)^*$

Φ



ε



a

【题 2】

1.求如下简单算术表达式文法G_{enr}中语法变量的 **FOLLOW** 集。

$E \to TE'$

$E \to +TE' \mid \varepsilon$

$T \to FT'$

$T' \to *FT' \mid \varepsilon$

$F \to (E) \mid \mathrm{id}$

[解答]：(1) 求表达式文法的语法符号的 FIRST 集：

FIRST(F)={ (, id}

FIRST(T)=FIRST(F)={ (, id}

FIRST(E)=FIRST(T)={ (, id}

FIRST(E')={+, ε }

FIRST(T')={*,ε }

FIRST(+)={+}, FIRST(*)={*}

FIRST(()={ (}

FIRST( ))={) }

FIRST(id)={id}

（2）求表达式文法的语法变量的 FOLLOW 集：

FOLLOW(E) = { #,) }

FOLLOW(E')= FOLLOW(E) = { #,) }

FOLLOW(T) = {FIRST(E')-{ε }}∪FOLLOW(E)∪FOLLOW(E')= {+,),#}

FOLLOW(T')= FOLLOW(T)= {+,),#}

FOLLOW(F) = FIRST(T')∪FOLLOW(T)∪FOLLOW(T') ={*,+,),#} 6

[知识点]

First 集合的求法：

First 集合最终是对产生式右部的字符串而言的，但其关键是求出非终结符的 First 集合，由于终结符的 First 集合就是它自己，所以求出非终结符的 First 集合后，就可很直观地得到每个字符串的 First 集合

1. 直接收取：对形如 U->a…的产生式（其中 a 是终结符），把 a 收入到 First(U)中
2. 反复传送：对形入 U->P1P2P3…Pn 的产生式（其中 P 是非终结符），应先把 First(P1)中的全部内容传送到 First(U)中,如果 P1中有 ε ，把 First(P2)中的内容传送到 First(U)中,类推直到 Pi 中无 ε 。

Follow 集合的求法：

Follow 集合是针对非终结符而言的，Follow(U)所表达的是句型中非终结符 U 所有可能的后随终结符号的集合，特别地，“$”是识别符号的后随符,先直接加入到 S 中。

1. 直接收取：注意产生式右部的每一个形如“…Ua…”的组合，把 a 直接收入到 Follow(U)中。
2. 直接收取：对形如“…UP…”(P 是非终结符)的组合，把 First(P)中非 ε 收入到 Follow(U)中。
3. 反复传送：对形如 U—>aP 的产生式（其中 P 是非终结符）或 U->aPQ(P,Q 为非终结符且 Q 中含 ε ），应把 Follow(U)中的全部内容传送到 Follow(P)中。

[例] 文法：$S\rightarrow ABc \quad A\rightarrow a/\varepsilon \quad B\rightarrow b/\varepsilon$

First 集合求法:能由非终结符号推出的所有的开头符号或可能的 ε ，但要求这个开头符号是终结符号。如此题 A 可以推导出 a 和 ε ，所以 FIRST（A）= {a，ε }；同理 FIRST（B）={b,ε };S 可以推导出 aBc，还可以推导出 bc，还可以推导出 c，所以 FIRST(S) = {a，b，c}

Follow 集合的求法:紧跟随其后面的终结符号或＃。但文法的识别符号包含＃，在求的时候还要考虑到ε。具体做法是把所有包含你要求的符号的产生式都找出来，再看哪个有用。 Follow（S）= {＃} 如求 A 的，产生式：S→ABc A→a|ε ，但只有 S→ABc 有用。跟随在 A 后面的终结符号是 FIRST（B）= {b，ε}，当 FIRST（B）的元素为ε时，跟随在 A 后的符号就是 c，所以 Follow（A）= {b，c} 同理 Follow（B）= {c}

2.对下面的文法 **G** ：

$E \rightarrow TE'$

$E' \rightarrow +E \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow T \mid \varepsilon$

$F \rightarrow PF'$

$F' \rightarrow *F' \mid \varepsilon$

$P \rightarrow (E) \mid a \mid b \mid \wedge$

(1)计算这个文法的每个非终结符的 FIRST 集和 FOLLOW 集。

(2) 证明这个方法是 LL(1) 的。

(3) 构造它的预测分析表。

解：（1）计算这个文法的每个非终结符的 FIRST 集和 FOLLOW 集。

FIRST 集合有：

FIRST(E)=FIRST(T)=FIRST(F)=FIRST(P)={(,a,b,^};
FIRST(E')={+,ε}
FIRST(T)=FIRST(F)=FIRST(P)={(,a,b,^};
FIRST(T')=FIRST(T)∪{ε}={(,a,b,^,ε};
FIRST(F)=FIRST(P)={(,a,b,^};
FIRST(F')=FIRST(P)={*,ε};
FIRST(P)={(,a,b,^};
FOLLOW 集合有：
FOLLOW(E)={),#};
FOLLOW(E')=FOLLOW(E)={),#};
FOLLOW(T)=FIRST(E')∪FOLLOW(E)={+,),#};//不包含 ε
FOLLOW(T')=FOLLOW(T)=FIRST(E')∪FOLLOW(E)={+,),#};
FOLLOW(F)=FIRST(T')∪FOLLOW(T)={(,a,b,^,+,),#};//不包含 ε
FOLLOW(F')=FOLLOW(F)=FIRST(T')∪FOLLOW(T)={(,a,b,^,+,),#};
FOLLOW(P)=FIRST(F')∪FOLLOW(F)={*,(,a,b,^,+,),#};//不包含 ε
(2)证明这个方法是 LL(1)的。
各产生式的 SELECT 集合有：
SELECT(E->TE')=FIRST(T)={(,a,b,^};
SELECT(E'->+E)={+};
SELECT(E'->ε)=FOLLOW(E/)={),#}
SELECT(T->FT')=FIRST(F)={(,a,b,^};
SELECT(T'->T)=FIRST(T)={(,a,b,^};
SELECT(T'->ε)=FOLLOW(T/)={+,),#};
SELECT(F->PF')=FIRST(P)={(,a,b,^};
SELECT(F'->*F')={*};
SELECT(F'->ε)=FOLLOW(F')={(,a,b,^,+,),#};
SELECT(P->(E))={(}
SELECT(P->a)={a}
SELECT(P->b)={b}
SELECT(P->^)={^}
可见，相同左部产生式的 SELECT 集的交集均为空，所以文法 G[E]是 LL(1)文法。
(3)构造它的预测分析表。
文法 G[E]的预测分析表如下：

	+	*	(	)	a	b	^	#
E			→TE′		→TE′	→TE′	→TE′	
E′	→+E			→ε				→ε
T			→FT′		→FT′	→FT′	→FT′	
T′	→ε		→T	→ε	→T	→T	→T	→ε
F			→PF′		→PF′	→PF′	→PF′	
F′	→ε	→*F′	→ε	→ε	→ε	→ε	→ε	→ε
P			→(E)		→a	→b	→^	

【题 3】

考虑下面的文法 G_e：

$(1)E \to T$
$(2)E \to E+T$
$(3)E \to E-T$
$(4)T \to F$
$(5)T \to T*F$
$(6)T \to T/F$
$(7)F \to (E)$
$(8)F \to id$

（1）求出所有语法变量的 FIRSTOP 集合和 LASTOP 集合。

（2）构造文法G_e的算符优先关系表，并判断G_e是否为算符优先文法。

（3）计算文法G_e的算符优先函数。

（4）给出表达式id_1+id_2和 id*(id+id)的算符优先分析过程。

[解答]：

（1）所有语法变量的 FIRSTOP 集合和 LASTOP 集合如下：

$FIRSTOP(E)=\{+,-,*,/,(,id\}$
$FIRSTOP(T)=\{*,/,(,id\}$
$FIRSTOP(F)=\{(,id\}$
$LASTOP(E)=\{+,-,*,/,),id\}$
$LASTOP(T)=\{*,/,),id\}$
$LASTOP(F)=\{),id\}$

（2）文法G_e的算符优先关系表

算符 关 系 算 符	**+**	**-**	*****	**/**	**(**	**)**	**id**
+	⋗	⋗	⋖	⋖	⋖	⋗	⋖
-	⋗	⋗	⋖	⋖	⋖	⋗	⋖
*****	⋗	⋗	⋗	⋗	⋖	⋗	⋖
/	⋗	⋗	⋗	⋗	⋖	⋗	⋖
（	⋖	⋖	⋖	⋖	⋖	⋍	⋖
）	⋗	⋗	⋗	⋗		⋗	
id	⋗	⋗	⋗	⋗		⋗	

因为文法中任意两个终结符之间只存在一种关系，因此该文法为算符优先文法。

（3）文法G_e的算符优先函数

优先函数 算符	+	-	*	/	（	）	id	#
f（栈内优先函数）	2	2	4	4	0	4	4	0
g（栈外优先函数）	1	1	3	3	5	0	5	0

（4）表达式id_1+id_2的算符优先分析过程

步骤	栈	输入串	优先关系	动作
1	#	id_1+id_2#		
2	#id_1	$+id_2$#	#$\lessdot id_1$	移进id_1
3	#F	$+id_2$#	#$\lessdot id_1 \gtrdot$#	用$F \to id$归约
4	#F+	id_2#	$\lessdot$	移进+
5	#F+id_2	#	$\lessdot$	移进id_2
6	#F+F	#	$+\lessdot id_2 \gtrdot$#	用$F \to id$归约
7	#E	#	#$\lessdot+\gtrdot$#	用$E \to E+T$归约

表达式 id*(id+id)的算符优先分析过程

步骤	栈 S	优先关系	当前输入字符 R	输入字符串
0	#	$\lessdot$	id	*(id+id)#
1	#id	$\gtrdot$	*	(id+id)#
2	#N	$\lessdot$	*	(id+id)#
3	#N*	$\lessdot$	(	id+id)#
4	#N*(	$\lessdot$	id	+id)#
5	#N*(id	$\gtrdot$	+	id)#
6	#N*(N	$\lessdot$	+	id)#
7	#N*(N+	$\lessdot$	id	)#
8	#N*(N+id	$\gtrdot$	)	#
9	#N*(N+N	$\gtrdot$	)	#
10	#N*(N	$\doteq$	)	#
11	#N*(N)	$\gtrdot$	#	
12	#N*N	不确定	#	
13	#N	停止	#	

2.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)# 的算符优先分析过程。

解：（1）各符号的 FIRSTVT 和 LASTVT：

	FIRSTVT	LASTVT
S	a、^、(	a、^、)
T	,、a、^、(	,、a、^、)

（2）算符优先关系表：

	a	(	)	,	^	#
a			⋗	⋗		⋗
(	⋖	⋖	≡		⋖	
) ,	⋖	⋖	⋗ ⋗	⋗ ⋗	 ⋖	⋗
^			⋗	⋗		⋗
#	⋖	⋖			⋖	

因为文法中任意两个终结符之间只存在一种关系，因此该文法为算符优先文法。

（3）对应的算符优先函数为：

	a	(	)	,	^	#
S	2	1	2	2	2	1
T	3	3	1	1	3	1

（4）句子(a,a)#分析过程如下：

步骤	栈	优先关系	当前符号	剩余输入串	移进或归约
1	#	#⋖(	(	a,a)#	移进
2	#(	(⋖a	a	,a)#	移进
3	#(a	a⋗,	,	a)#	归约
4	#(F	(⋖,	,	a)#	移进
5	#(F,	,⋖a	A	)#	移进
6	#(F,a	A⋗)	)	#	归约
7	#(F,F	,⋗)	)	#	归约
8	#(F	(≡)	)	#	移进
9	#(F)	)⋗#	#		归约
10	#F	#=#	#		接受

[知识点]

FIRSTVT 及 LASTVT 求法

构造集合 FIRSTVT （ P ）的两条规则。

（i）若有产生式 P→a… ，或 P→Qa… ，则 a ∈ FIRSTVT （ P ）。

（ii）若 a ∈ FIRSTVT （ P ），且有产生式 P→Q… ，则 a ∈ FIRSTVT （ P ）。

构造集合 FIRSTVT （ P ）的两条规则

（i）有产生式 P→…a，或 P→…aQ，则 a∈LASTVT（P）。

（ii）若 a ∈ LASTVT （ Q ），且有产生式 P→Q… ，则 a ∈ FIRSTVT （ P ）。

【题 4】

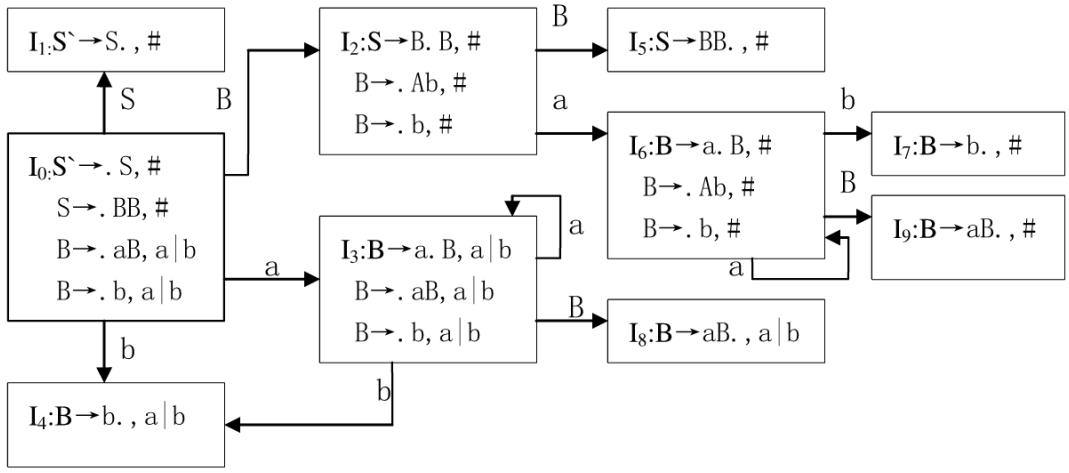
1. 令文法 G：S→ BB B→ aB B→ b 判断该文法是否 LR（1）文法，若是构造 LR（1）分析表。

解 1）将文法 G 拓广为 G`：（0）S`→S （1）S→BB （2）B→Ab （3）B→b

2)求出 G`的非终结符的 FOLLOW 和 FIRST 集

A	FOLLOW(A)	FIRST(A)
S`	#	a, b
S	#	a, b
B	a, b, #	a, b

3）构造个 G`的 LR(1)的项目集族及 GO 函数



3)判断文法是否为 LR（1）文法。

该文法构出的 C＝{I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9}中，每个状态集均无冲突，所以该文法是 LR（1）文法。

4）构造 LR(1)分析表

状态	ACTION			GOTO	
	a	b	#	S	B
0	S3	S4		1	2
1			acc		
2	S6	S7			5
3	S3	S4			8
4	r3	r3			
5			r1		
6	S6	S7			
7			r3		
8	r2	r2			
9			r2		

填空题

1. 消除左递归（P124）

文法左递归问题:一个文法是含有左递归的，如果存在非终结符 P。 $P \overset{+}{\Rightarrow} P\alpha$

◆ 直接消除见诸于产生式中的左递归：

假定关于非终结符 P 的规则为 P→Pα | β,其中 β 不以 P 开头。那么，我们可以把 P 的规则等价地改写为如下的非直接左递归形式：

P→βP′

P′→αP′|ε

一般而言，假定关于 P 的全部产生式是 P→Pα1 | Pα2 | … | Pαm | β1 | β2|…|βn,其中，每个 α 都不等于 ε，而每个 β 都不以 P 开头,那么，消除 P 的直接左递归性就是把这些规则改写成：

P→β1P′ | β2P′ | … | βnP′

P′→α1P′ | α2P′ |… | αmP′ | ε

[例] 文法

E→E＋T | T

T→T*F | F

F→(E) | i

经消去直接左递归后变成：

E→TE′

E′→+TE′ | ε

T→FT′

T′→*FT′ | ε

F→(E) | i

例如文法

S→Qc|c

Q→Rb|b

R→Sa|a

虽没有直接左递归，但 S、Q、R 都是左递归的 S⇒Qc⇒Rbc⇒Sabc

◆ 一个文法消除左递归的条件：

1) 不含以 ε 为右部的产生式（无空产生式）；

2) 不含回路。 $P \overset{+}{\Rightarrow} P$

◆ 消除左递归的算法：

1. 把文法 G 的所有非终结符按任一种顺序排列成 P1，P2，…，Pn；按此顺序执行；

2. FOR i:=1 TO n DO

BEGIN

FOR j:=1 TO i-1 DO

把形如 Pi→Pjγ 的规则改写成

Pi→δ1γ|δ2γ|…|δkγ ;

(其中 Pj→δ1|δ2|…|δk 是关于 Pj 的所有规则)

消除关于 Pi 规则的直接左递归性

END

3. 化简由 2 所得的文法。即去除那些从开始符号出发永远无法到达的非终结符的产生规则。

[例] 考虑文法 G(S)

S→Qc|c

Q→Rb|b

R→Sa|a

令它的非终结符的排序为 R、Q、S。对于 R，不存在直接左递归。

把 R 代入到 Q 的有关候选后，把 Q 的规则变为 Q→Sab | ab | b，现在的 Q 不含直接左递归。
把 Q 代入到 S 的有关候选后，S 变成 S→Sabc | abc | bc | c，由于 S→Sabc | abc | bc | c 存在直接左递归，消除 S 的直接左递归后：

S→abcS′ | bcS′ | cS′
S′→abcS′ | ε
Q→Sab |ab | b
R→Sala

关于 Q 和 R 的规则已是多余的，化简为：

S→abcS′ | bcS′ | cS′
S′→abcS′ | ε

注意：由于对非终结符排序的不同，最后所得的文法在形式上可能不一样。但不难证明，它们都是等价的。
同样：[例] 考虑文法 G(S)
S→Qc|c
Q→Rb|b
R→Sala
非终结符排序选为 S、Q、R，那么，

R →Qcalcala
R →Rbcalbcalcala

最后所得的无左递归文法是：

S→Qc | c
Q→Rb | b
R→bcaR′ | caR′ |a R′
R′→ bca R′ | ε

不同排序所得的文法的等价性是显然的。
2.文法的分类**(Chomsky** 体系**)**（**P41**）
（1）短语结构文法（PSG）
如果 G 满足文法定义的要求，则 G 是 0 型文法（短语结构文法 PSG: Phrase Structure Grammar ）。
L(G)为 PSL。
（1）上下文有关文法(CSG)
如果对于 $\forall \alpha \to \beta \in P$，均有|β |≥|α |成立(S →ε 除外)，则称 G 为 1 型文法。即：上下文有关文法（CSG）
L(G)为 1 型/上下文有关/敏感语言(CSL)。其他定义方法：设文法 G[S]，若 P 中任一产生式α →β 的形式为 $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$,其中 α_1,β ,α_2 ∈(V∪T）*,β ≠ε ,A ∈V。
（2）上下文无关文法(CFG)
如果对于 $\forall \alpha \to \beta \in P$，均有|β |≥|α |，并且α ∈V 成立，则称 G 为 2 型文法。即：上下文无关文法(CFG:)
L(G)为 2 型/上下文无关语言（CFL），CFG 能描述程序设计语言的多数语法成分。
（3）正规（则）文法(RG)
设 A,B ∈ V，w∈T+或为 ε，如果对于 $\forall \alpha \to \beta \in P, \alpha \to \beta$ 均具有如下形式：
右线性(Right Linear)文法： $A \to \omega B$ 或 $A \to \omega$
左线性(Left Linear)文法： $A \to B\omega$ 或 $A \to \omega$
都是 3 型文法(正规文法 RG)，L(G)为 3 型/正规集/正则集/正则语言（RL），能描述程序设计语言的多数单词。左、右线性文法不可混用。
[例]正规文法（RG）：

G1：S 0 | 1 | 00 | 11

G3：S 0 | 1 | 0A | 1B，A 0，B 1

G5：S 0 | 0S

G8：A aS | bS | cS | a | b | c

上下文无关文法（CFG）：

G2：S A | B | AA | BB, A 0，B 1

G4：S A | B | BB, A 0，B 1

G14：S 0 | 1 | 2 | 3 | 0S0 | 1S1 | 2S2 | 3S3

上下文有关文法（CSG）：

G：

S→CD

C→aCA

CA→Ca

CaD→daD

dAc→dec

- G =(V，T， P， S)是一个文法,α →β ∈ P

* G 是 0 型文法，L(G)是 0 型语言；---其能力相当于图灵机

* |α |≤|β |:G 是 1 型文法，L(G)是 1 型语言(除 S →ε);---其识别系统是线性界限自动机

* α ∈ VN : G 是 2 型文法，L(G)是 2 型语言;---其识别系统是不确定的下推自动机

* A→aB 或 A→a: G 是右线性文法，L(G)是 3 型语言

A→Ba 或 A→ a : G 是左线性文法，L(G)是 3 型语言---其识别系统是有穷自动机

3.逆波兰表示法，请注意 **IF---ELSE** 及 **FOR** 循环 的特例

逆波兰表达式又叫做后缀表达式。在通常的表达式中，二元运算符总是置于与之相关的两个运算对象之间，所以，这种表示法也称为中缀表示。波兰逻辑学家 J.Lukasiewicz 于1929年提出了另一种表示表达式的方法。按此方法，每一运算符都置于其运算对象之后，故称为后缀表示。

将一个普通的中序表达式转换为逆波兰表达式的一般算法是：

(1)首先构造一个运算符栈，此运算符在栈内遵循越往栈顶优先级越高的原则。

(2)读入一个用中缀表示的简单算术表达式，为方便起见,设该简单算术表达式的右端多加上了优先级最低的特殊符号“#”。

(3)从左至右扫描该算术表达式，从第一个字符开始判断，如果该字符是数字，则分析到该数字串的结束并将该数字串直接输出。

(4)如果不是数字，该字符则是运算符，此时需比较优先关系。做法如下：将该字符与运算符栈顶的运算符的优先关系相比较。如果，该字符优先关系高于此运算符栈顶的运算符，则将该运算符入栈。倘若不是的话，则将栈顶的运算符从栈中弹出，直到栈顶运算符的优先级低于当前运算符，将该字符入栈。

(5)重复上述操作(3)-(4)直至扫描完整个简单算术表达式，确定所有字符都得到正确处理，我们便可以将中缀式表示的简单算术表达式转化为逆波兰表示的简单算术表达式。

逆波兰表达式，它的语法规定，表达式必须以逆波兰表达式的方式给出。逆波兰表达式又叫做后缀表达式。这个知识点在数据结构和编译原理这两门课程中都有介绍，下面是一些例子：

正常的表达式 逆波兰表达式

a+b ---> a,b,+

a+(b-c) ---> a,b,c,-,+

a+(b-c)*d ---> a,b,c,-,d,*,+

a+d*(b-c)--->a,d,b,c,-,*,+

a=1+3 ---> a=1,3 +

http=(smtp+http+telnet)/1024 --->http=smtp,http,telnet,+,+,1024,/

例 9 将下列语句翻译为逆波兰表示(后缀式)、三元式和四元表示：

a:=(b+c)*e+(b+c)/f

【解】解题思路

把中缀式转换为后缀式的简单方法：按中缀式中各运算符的优先规则，从最先执行的部分开始写，一层层套。如 a≤b+c∧a＞d∨a+b≠e，先把 b+c 写为 bc+；然后把 a≤套上去，成为 abc+≤；再把 a＞d 表示为 ad＞；然后把∧套上去，成为 abc+≤ad＞∧，依此类推。

［练习］给出下列表达式的逆波兰表示

$a*b+(c-d)/e$

（1）$-a+b*(-c+d)$

（2）$(a>b)\wedge(b<c)$

5．句柄

设有 **CFG G=(V,T,P,S),G** 的句型的最左直接短语称为句柄。

附：定义 2.27 设有 CFG G=(V，T，P，S)，$\exists\alpha$，γ，β ∈(V∪T)*，$S\overset{*}{\Rightarrow}\gamma A\beta$，$A\overset{+}{\Rightarrow}\alpha$，则称α 是句型 $\gamma\alpha\beta$ 的相对于变量 A 的短语(phrase)；如果此时有 $A\Rightarrow\alpha$，则称α 是句型 $\gamma\alpha\beta$ 的相对于变量 A 的直接短语。在无意义冲突时，简称为句型 $\gamma\alpha\beta$ 的直接短语,直接短语也叫做简单短语。

[例] 对于文法 $G_{12}:E\rightarrow \mathrm{id}\,|\,\mathrm{c}\,|+E\,|-E\,|E+E\,|E-E\,|E*E\,|E/E\,|E**E\,|(E)$ 的 id_1+c*E 句型来说，它的短语有：id_1，c，$c*E$，id_1+c*E。其中直接短语有：id_1，c。而 id_1 就是句型 id_1+c*E 的句柄。

6.综合属性和继承属性 **2012---2013** 年已经取消这题

如果节点的属性值是通过分析树中该节点或其子节点的属性值计算出来的，则称其为综合属性；如果节点的属性值是由该节点、该节点的兄弟节点或父节点的属性值计算出来的，则称其为继承属性。

[例 1] 文法符号的属性有综合属性和继承属性两种。

[例 2] S—属性文法中的每个文法符号，只含有综合属性。

[例 3] 终结符只有综合属性，它们有词法分析器提供。

7.语法制导定义，出现在第四大题的最后一步

（0）s'->s{printf（“0”）；}

（1）s->BB {printf（“1”）；}

（2）B->aB {printf（“2”）；}

(3)B->b {printf（“3”）；} 输入#bab# 输出结果是 33210