第三章: 词法分析

3.1 词法分析的功能

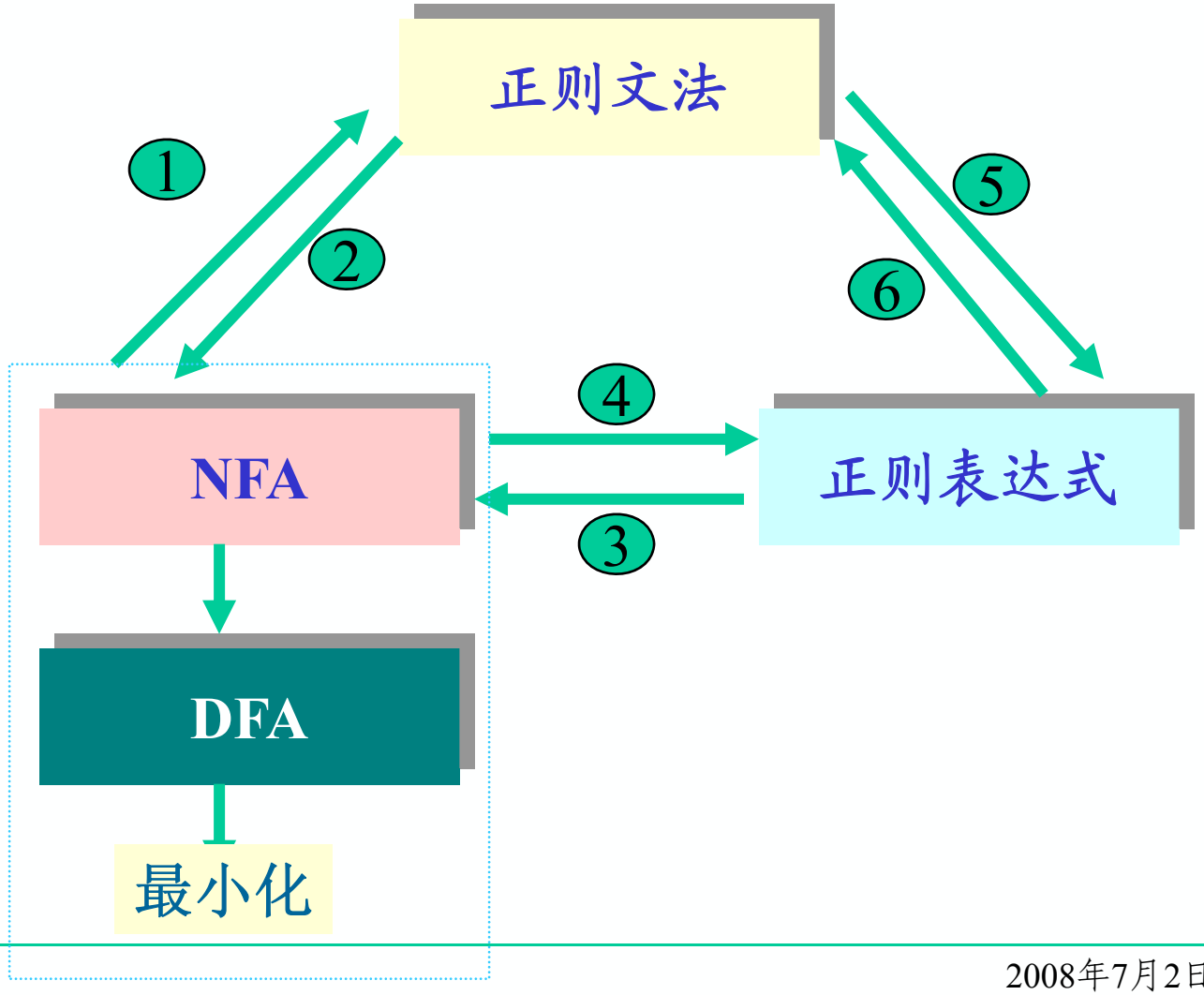
3.2 词法分析程序的设计与实现

- 状态图

3.3 词法分析程序的自动生成

- 有穷自动机、LEX

补充



P67:1. 画出下述文法的状态图

〈Z〉::=〈B〉e

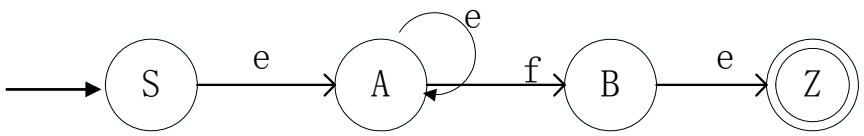
〈B〉::=〈A〉f

〈A〉::= e | 〈A〉e

使用该状态图检查下列句子是否是该文法的合法句子

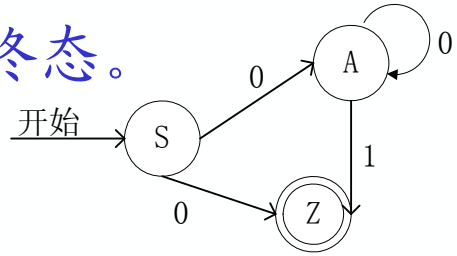
f，eeff，eefe

解:



f，eeff不是该文法的合法句子，eefe是该文法的合法句子

P67：2．有下列状态图，其中S为初态，Z为终态。



（1） 写出相应的正则文法：

（2） 写出该文法的V，V_n和V_t；

（3） 该文法确定的语言是什么？

➢出错点：第(3)小题，文法确定的语言，很多同学回答出了，但是写的格式很不规范。

解：（1） $Z \rightarrow A1 \mid 0$　　$A \rightarrow A0 \mid 0$

（2） $V=\{A, Z, 0, 1\}$　　$Vn=\{A, Z\}$　　$Vt=\{0, 1\}$

（3） $L(G[S])=\{0$或$0^n1, n \geqslant 1\}$

$0 \mid 00^*1$

P67: 5．令A，B，C是任意正则表达式，证明以下关系成立：

A|A=A

（A*）*= A*

A*=ε| AA*

（AB）*A = A（BA）*

（A|B）* =（A*B*）*=（A*|B*）

➢该题做得不错！但少部分同学最后一个等式没有证明。另外，有些同学虽然证对了，但是过程比较烦琐或者是不规范。

- 证明：正则表达式表示的语言相等，则正则表达式相等

(1) 若用L_A表示正则表达式A表示的语言，则A｜A表示的语言是：$L_A \cup L_A$

因为$L_A \cup L_A = L_A$ 所以A｜A＝A

(2)（A*）*表示的语言为$(L_A^*)^*$

A*表示的语言为L_A^*

$(L_A^*)^* = (L_A^*)^0 \cup (L_A^*)^1 \cup (L_A^*)^2 \cdots$

$= \{\varepsilon\} \cup (L_A^*)^1 \cup (L_A^*)^2 \cdots$

下面用数学归纳法证明$(L_A^*)^n = L_A^*$，$n \geq 1$

n＝1时，显然成立

假定n＝k时，$(L_A^*)^k = L_A^*$，则n＝k＋1时

$(L_A^*)^{k+1} = (L_A^*)^k L_A^* = L_A^* L_A^* = (L_A^0 \cup L_A^1 \cup L_A^2 \cdots) L_A^*$

$= L_A^0 L_A^* \cup L_A^1 L_A^* \cup L_A^2 L_A^* \cdots$

$= (L_A^0 \cup L_A^1 \cup L_A^2 \cdots) \cup L_A^1 (L_A^0 \cup L_A^1 \cup L_A^2 \cdots) \cup L_A^2 (L_A^0 \cup L_A^1 \cup L_A^2 \cdots) \cdots$

$= (L_A^0 \cup L_A^1 \cup L_A^2 \cdots) \cup (L_A^1 \cup L_A^2 \cup L_A^3 \cdots) \cup (L_A^2 \cup L_A^3 \cup L_A^4 \cdots) \cdots$

$= L_A^0 \cup L_A^1 \cup L_A^2 \cdots$

$= L_A^*$

∴ $(L_A^*)^{k+1} = L_A^*$，故$(L_A^*)^n = L_A^*$，$n \geq 1$

∴ $(L_A^*)^* = \{\varepsilon\} \cup (L_A^*)^1 \cup (L_A^*)^2 \cdots = \{\varepsilon\} \cup L_A^* = L_A^*$

(3)ε | AA^*所表示的语言是:

$\{\varepsilon\} \cup L_A \cdot L_A^*$

$= L_A^0 \cup L_A\ (L_A^0 \cup L_A^1 \cup L_A^2 \cup \dots)$

$= L_A^0 \cup L_A^1 \cup L_A^2 \cup \dots = L_A^*$

故$\varepsilon \mid AA^* = A^*$

(4)$(L_AL_B)^*L_A=(\{\varepsilon\} \cup L_AL_B \cup L_AL_BL_AL_B \cup L_AL_BL_AL_BL_AL_B \cup \dots)\ L_A$

$=L_A \cup L_AL_BL_A \cup L_AL_BL_AL_BL_A \cup L_AL_BL_AL_BL_AL_B \cup L_A\dots$

$= L_A \cup (\{\varepsilon\} \cup L_BL_A \cup L_BL_AL_BL_A \cup \dots)$

$= L_A(L_BL_A)^*$

$\therefore (AB)^*A=A(AB)^*$

(5)三个表达式所描述的语言都是L_A和L_B中符号串的任意组合

∴（A|B）*=(A*B*)*=(A*|B*)*

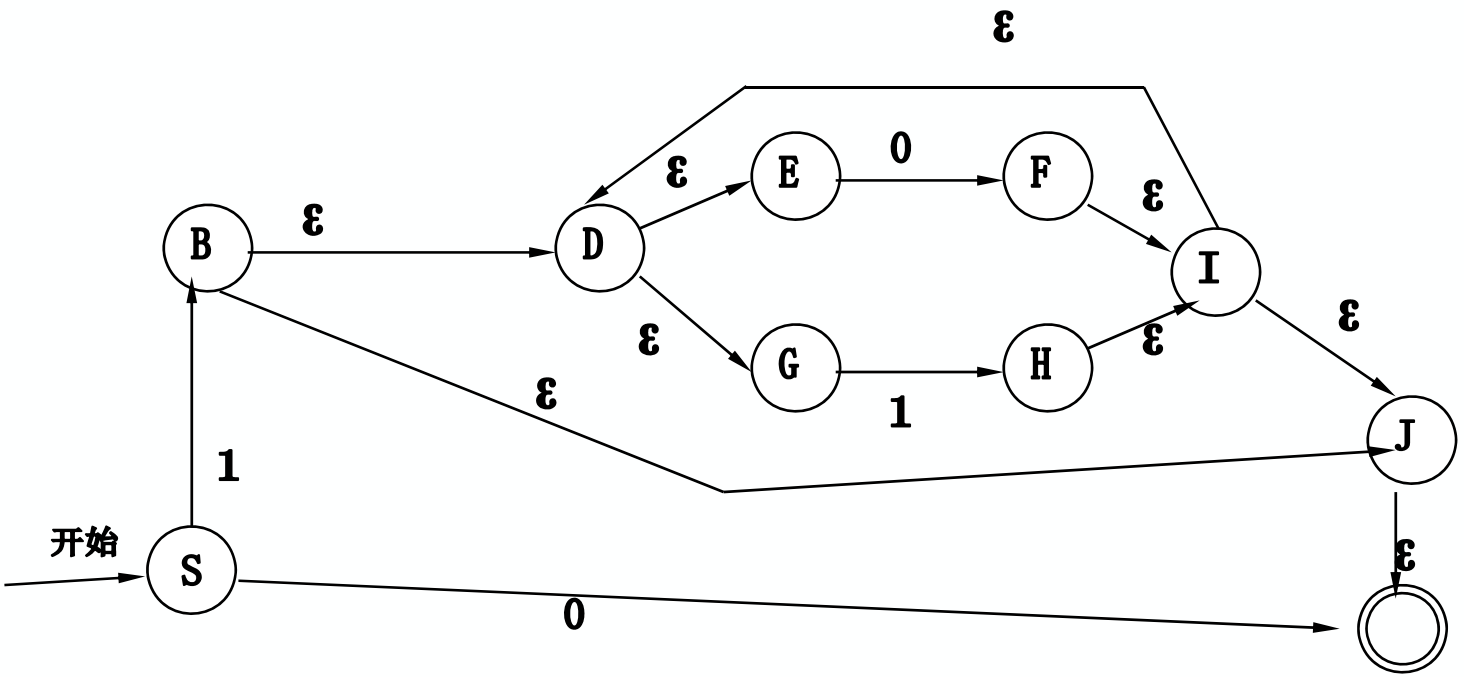
P67: 6.构造下列正则表达式相应的DFA

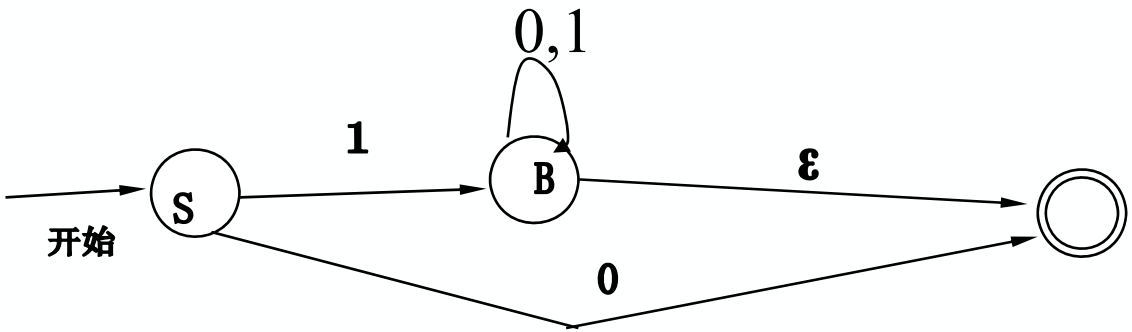
（1）1（0|1）*|0

（2）1（1010*|1（010）*1）*0

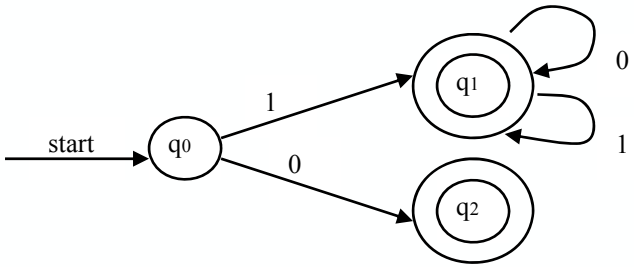
- 在状态图中终止状态没有用双线圆标出

（1）与1（0|1）*|0对应的NFA为：

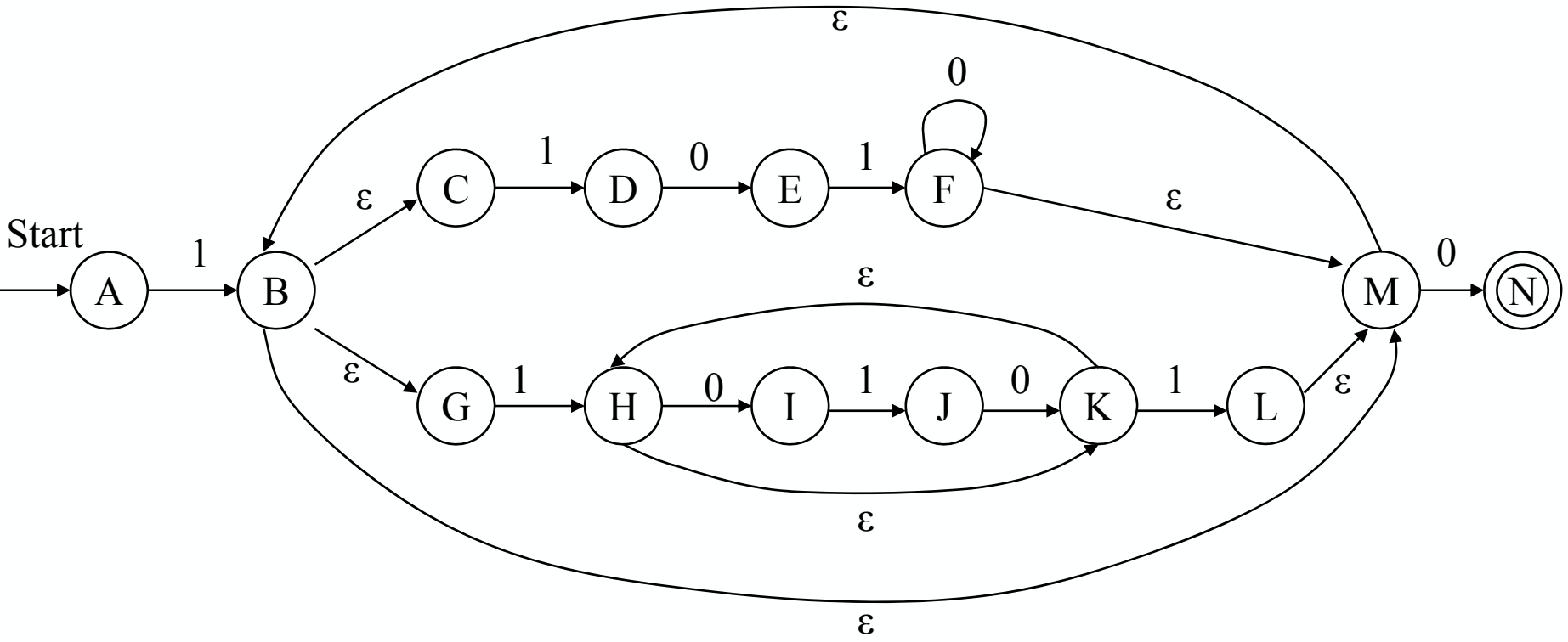




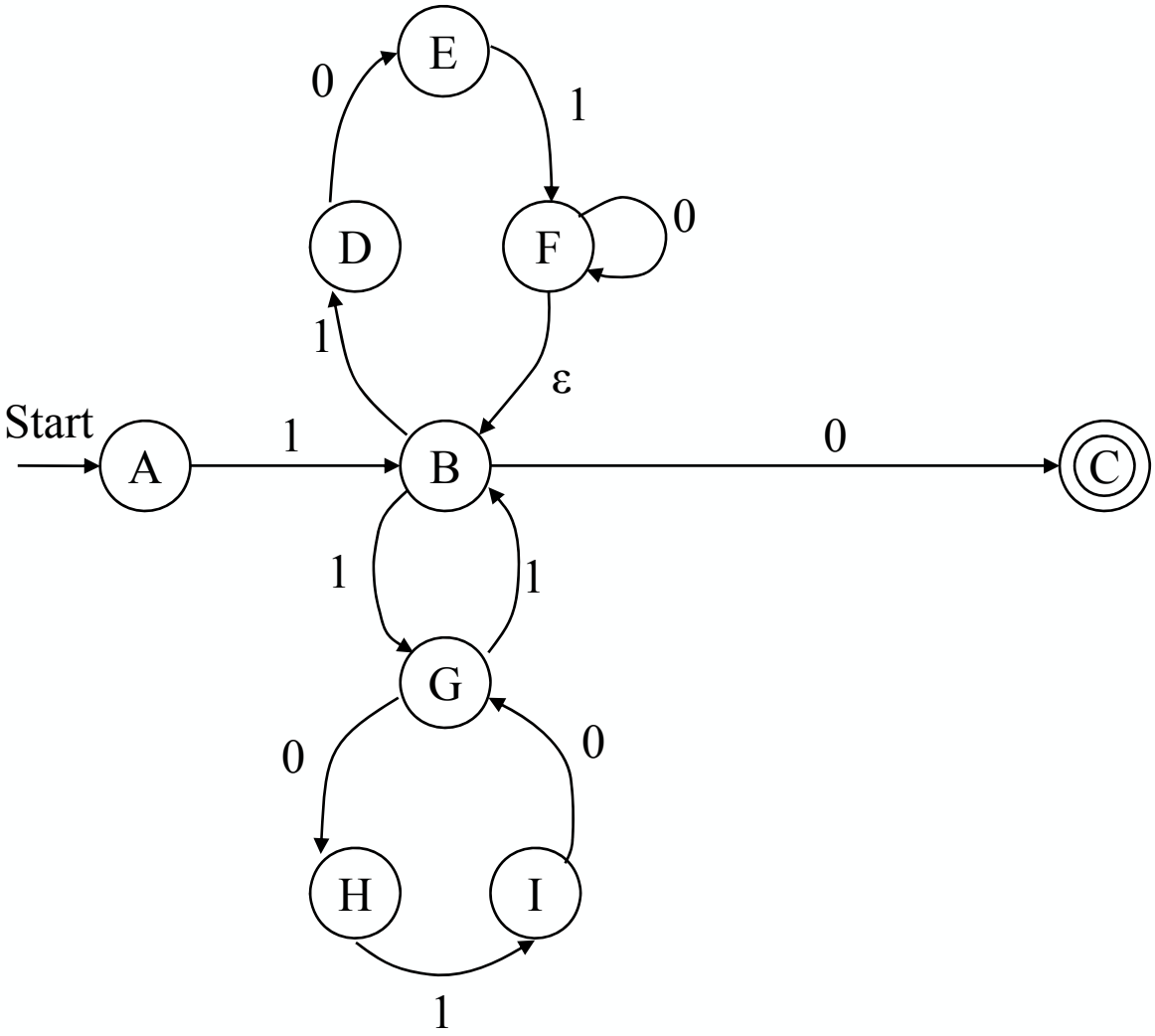
确定化:



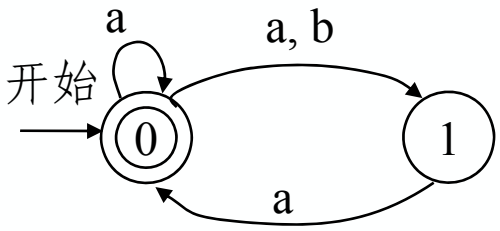
（2）1（1010*|1（010）*1）*0



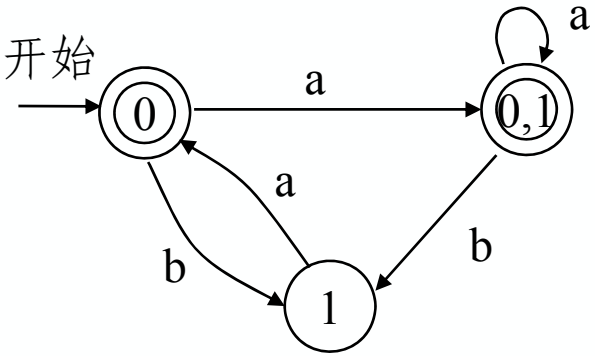
1（1010*|1（010）*1）*0



P68：8.把图3.24的（a）和（b）分别确定化

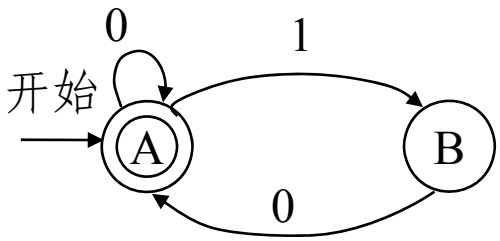


	a	b
{0}	{0,1}	{1}
{0,1}	{0,1}	{1}
{1}	{0}	--

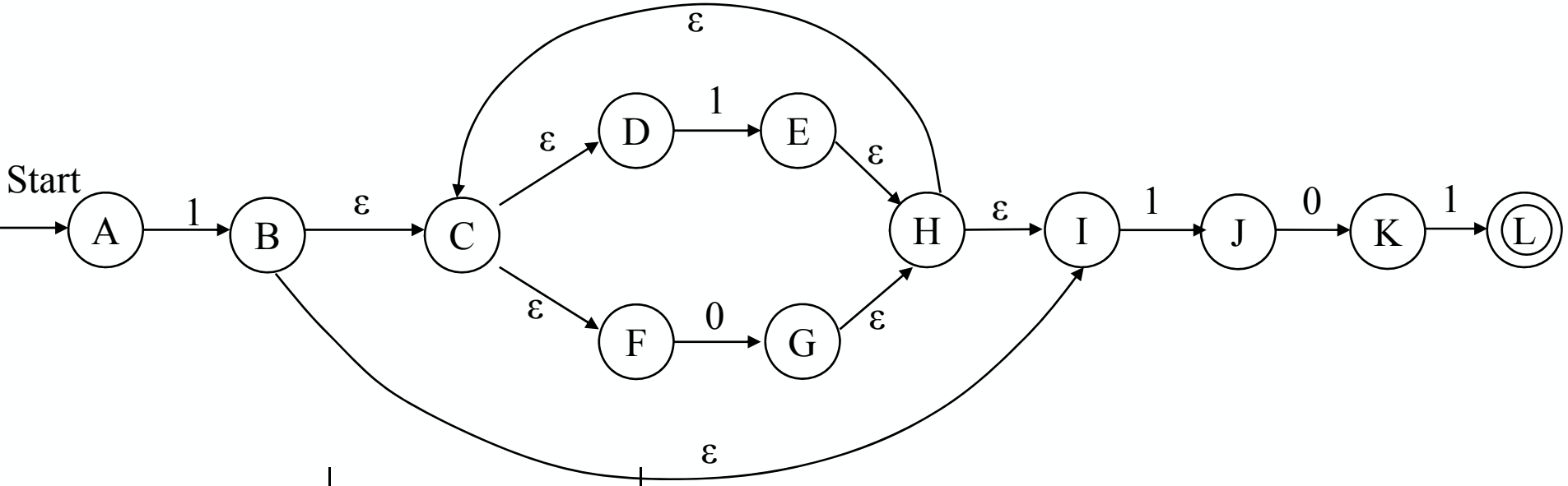


P68: 10 构造一DFA，它接受Σ={0,1}上所有满足如下条件的字符串：每个1都有0直接跟在右边。

(0|10)*



补充题 1.有正则表达式1(0|1)*101,构造DFA,并最小化



	0	1
{A}	--	{BCDFI}
{BCDFI}	{GHICDF}	{EHICDFJ}
{GHICDF}	{GHICDF}	{EHICDFJ}
{EHICDFJ}	{GHICDFK}	{EHICDFJ}
{GHICDFK}	{GHICDF}	{EHICDFJL}
{EHICDFJL}	{GHICDFK}	{EHICDFJ}

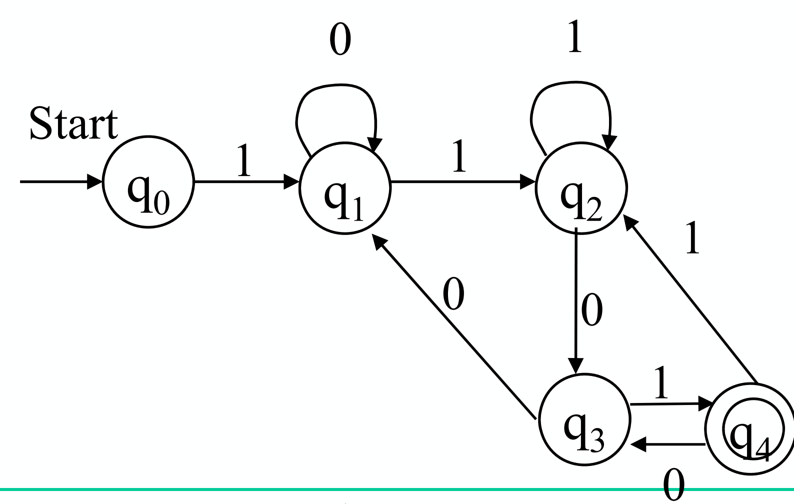
	0	1
q_0	--	q_1
q_1	q_2	q_3
q_2	q_2	q_3
q_3	q_4	q_3
q_4	q_2	q_5
q_5	q_4	q_3

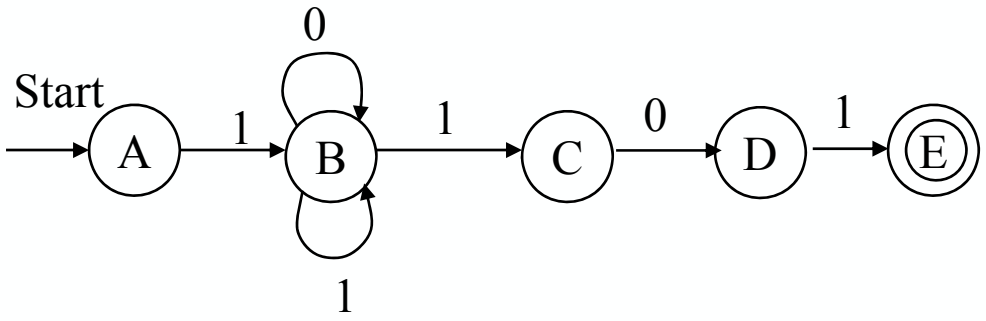
	0	1
q_0	--	q_1
q_1	q_2	q_3
q_2	q_2	q_3
q_3	q_4	q_3
q_4	q_2	q_5
q_5	q_4	q_3

	0	1
q_0	--	q_1
q_1	q_2	q_3
q_2	q_2	q_3
q_3	q_4	q_3
q_4	q_2	q_5
q_5	q_4	q_3

	0	1	
q_0	--	q_1	q_0
q_1	q_2	q_3	q_1
q_2	q_2	q_3	
q_3	q_4	q_3	q_2
q_4	q_2	q_5	q_3
q_5	q_4	q_3	q_4

	0	1
q_0	-	q_1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_1	q_4
q_4	q_3	q_2

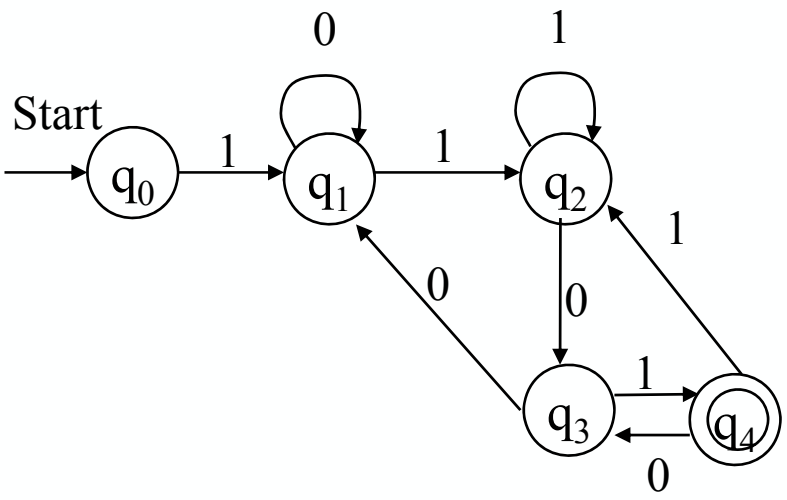




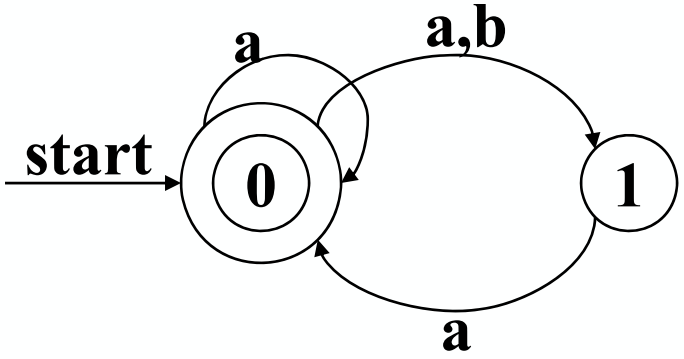
	0	1
{A}	-	{B}
{B}	{B}	{BC}
{BC}	{BD}	{BC}
{BD}	{B}	{BCE}
{BCE}	{BD}	{BC}

	0	1
q_0	-	q_1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_1	q_4
q_4	q_3	q_2

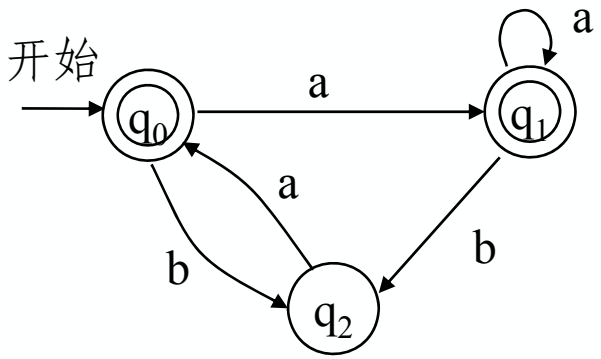
	0	1
q_0	-	q_1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_1	q_4
q_4	q_3	q_2



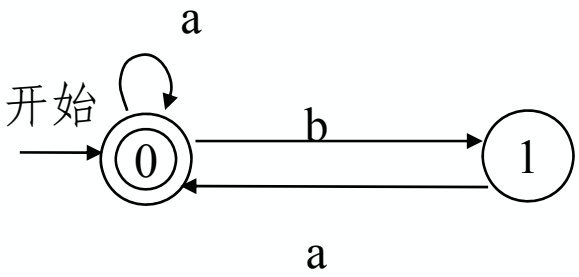
2.试将下列自动机确定化并最小化:



	a	b
{0}	{0,1}	{1}
{0,1}	{0,1}	{1}
{1}	{0}	--



	a	b	
q_0	q_1	q_2	0
q_1	q_1	q_2	
q_2	q_0	--	1



P80习题

a) 部分同学对不能正确的设计语法分析程序（不会设计递归程序）。

b) 有些人在求 first 集 和 follow 集时出错较多。尤其是求Follow集。

c) 在学习递归向下分析法时，不少人习惯于把文法最后归结为一个很长的右部，例如：S ::= (c{c})|dc{c}e, 实际上写成 如 S::= A|dAe ; A::= cA|c 的形式更易于写出分析程序，也更合理。

第四章　语法分析

- 语法分析的功能、基本任务
- 自顶向下分析法
  - 递归子程序法
  - LL(1)分析法
- 自底向上分析法
  - 算符优先分析法
  - LR分析法

采用不带回溯的自顶向下的分析，要求文法：

- 非左递归的

- 对文法的任一非终结符，若其规则右部有多个选择，各选择所推出的终结符号串的头符号集合要两两不相交。

注意：

- 在用递归子程序法和LL(1)分析法之前，对不符合要求的文法要进行改写！

- 判断和证明文法是否为LL(1)，不能改写文法！

判断文法是否为:

- LL(1)文法:
  - 构造分析表M，是否含多重入口
  - 或根据充要条件判断

FIRST,FOLLOW集合

- 算符优先文法:
  - 首先判断是否算符文法
  - 求任意两个终结符之间的优先关系，是否至多只有一种关系成立

FIRSTVT,LASTVT集合

- **LR文法:**
  - **构造LR(0)项目集**
  - **考察每个项目集:**
    - **不包含不充分状态：LR(0)文法**
    - **包含不充分状态:**
      - **能用FOLLOW集解决冲突：SLR文法**
      - **不能用FOLLOW集解决冲突**
        - **构造LR(1)项目集:**
          - **能用超前集解决冲突：LR(1)文法**
          - **合并具有相同核心的LR(1)项目集，无多重入口：LALR(1)文法**

FIRST,FOLLOW集合

P80: 2. 有文法G[A]:　　A::= (B) | dBe
B::= c | Bc

试设计自顶向下的语法分析程序。

解: 消除左递归:
A::= (B) | dBe
B::= c{c}

```
procedure B;
    if CLASS = 'c' then
       begin
          nextsym;
          while CLASS = 'c' do nextsym;
       end;
    else
      error;
```

```
program    G;
  begin
     nextsym;
     A;
  end;
```

```
procedure A;
    if CLASS = '(' then
       begin
          nextsym;
          B;
          if CLASS = ')' then
           nextsym;
          else
           error
       end;
    else
      if CLASS = 'd' then
        begin
           nextsym;
           B;
        if CLASS = 'e' then
           nextsym;
        else
           error;
        end;
      else
        error;
```

P80: 3. 有文法G[Z]:　　Z::= AcB| Bd

A::=AaB|c

B::= aA|a

(1) 试求各选择（候选式）的FIRST集合;

(2) 该文法的自顶向下的语法分析程序是否要编成递归子程序？为什么？

(3) 试用递归下降分析法设计其语法分析程序。

解: (1) FIRST(B)={a} FIRST(A)={c} FIRST(Z)={a,c}

FIRST(AcB)={c}　　FIRST(Bd) ={a}

FIRST(AaB)={c}　　FIRST(c) ={c}

FIRST(aA) ={a}　　FIRST(a) ={a}

(2) 要编成递归子程序，因为文法具有递归性

(3) 改写文法:

Z::= AcB| Bd

A::= c{aB}

B::= a[A]

```
void Z()
{
    if( sym ==‘c’ )
    {
        A();
        if( sym == ‘c’)
        {
            getsym();
            B();
        }
        else
            error();
    }
    else if( sym == ‘a’)
    {
        B();
        if( sym != ‘d’)
            error();
        else
            getsym();
    }
    else
        error();
}
```

Z::= AcB| Bd
A::= c{aB}
B::= a[A]

```
void A()
{
    if( sym == ‘c’)
    {
        getsym();
        while( sym == ‘a’)
        {
            getsym();
            B();
        }
    }
    else
        error();
}
```

```
void B()
{
    if( sym!=‘a’ )
        error();
    else
    {
        getsym();
        if( sym == ‘c’)
            A();
    }
}
```

```
void main()
{
    getsym();
    Z();
}
```

P 87: 1．对下面的文法G[E]:

$E \rightarrow TE'$
$E' \rightarrow +E \mid \varepsilon$
$T \rightarrow FT'$
$T' \rightarrow T \mid \varepsilon$
$F \rightarrow PF'$
$F' \rightarrow * F' \mid \varepsilon$
$P \rightarrow (E) \mid a \mid b \mid \wedge$

(1) 计算这个文法的每个非终结符号的FIRST和FOLLOW集合
(2) 证明这个文法是LL(1)的
(3) 构造它的预测分析表

- 典型的错误是：有相当的同学在F'上表项填得不完整。

解：(1)

FIRST(E) = {(, a, b, ∧ }
FIRST(E') = { +,ε}
FIRST(T) = { (, a, b, ∧}
FIRST(T') = { (, a, b, ∧,ε}
FIRST(F) = { (, a, b, ∧}
FIRST(F') = { *,ε}
FIRST(P) = {(, a, b, ∧ }

FOLLOW(E) = {#,) }
FOLLOW(E') = {#,)}
FOLLOW(T) = { #,),+}
FOLLOW(T') = {#,),+ }
FOLLOW(F) = { (, a, b, ∧,#,),+}
FOLLOW(F') = { (, a, b, ∧,#,),+}
FOLLOW(P) = {*,(, a, b, ∧,#,),+ }

E → TE‘
E' → + E |ε
T → FT'
T' → T |ε
F → PF'
F' → * F' |ε
P →(E)| a | b | ∧

(2) 证明:

FIRST(+E) ∩FIRST(ε) = {+}∩{ε} = ∅

FIRST(+E) ∩FOLLOW(E') = {+}∩{#,)} = ∅

FIRST(T) ∩FIRST(ε) = { (, a, b, ∧}∩{ε}= ∅

FIRST(T) ∩FOLLOW(T') ={ (, a, b, ∧}∩{#,),+ }= ∅

FIRST(*F') ∩FIRST(ε) = {*} ∩{ε} = ∅

FIRST(*F') ∩FOLLOW(F') = {*}∩{ (, a, b, ∧,#,),+} = ∅

FIRST((E)) ∩FIRST(a) ∩ FIRST(b) ∩FIRST(∧)= ∅

所以此文法是LL(1)文法

（3）分析表

	+	*	(	)	a	b	∧	#
E			E → TE'		E → TE'	E → TE'	E → TE'	
E'	E' → + E			E' →ε				E' → ε
T			T → FT'		T → FT'	T → FT'	T → FT'	
T'	T' →ε		T' → T	T' → ε	T' → T	T' → T	T' → T	T'→ ε
F			F → PF'		F → PF'	F → PF'	F → PF'	
F'	F' →ε	F' →* F'	F' →ε	F' →ε	F' →ε	F' →ε	F' →ε	F' →ε
P			P →(E)		P →a	P →b	P →∧	

2. 对于文法G[S]: S → aABbcd | ε
A → ASd | ε
B → SAh | eC | ε
C → Sf | Cg | ε
D → aBD | ε

（1) 对每一个非终结符号，构造FOLLOW集;
（2) 对每一产生式的各侯选式，构造FIRST集;
（3) 指出此文法是否为LL（1）文法。

(1)做题情况:
该题做得不理想，出错情况教多。
(2)错误情况
①FIRST(SAh)={a,d} //缺少了h 应为FIRST(SAh)={a,d,h}
原因分析: 在计算SAh的FIRST集时，忽略了S->ε， A->ε的情况，因此漏掉了h。
②FOLLOW(A),FOLLOW(S)的集合写错的较多。
原因分析: 一方面是由于FIRST集写得不对，另一方面主要的原因是忽略了产生式中直接推出ε的情况，没有进一步的分析下去。
如: S->aABbcd |ε
由上式可得出FOLLOW(A)=FIRST(B)-{ε}
而当B->ε时，FOLLOW(A)={b}
所以，FOLLOW(A)= FIRST(B)-{ε}+{b}={a,b,d,e,h}
基本上类似的出错都是由于这个原因

2.构造集合FOLLOW的算法

设$S, A, B \in V_n$，

算法：连续使用以下规则，直至FOLLOW集合不再扩大

(1) 若S为识别符号,则把“#”加入FOLLOW(S)中

(2) 若$A::=\alpha B\beta$ $(\beta \neq \varepsilon)$,则把FIRST(β)-{ε}加入FOLLOW(B)

(3) 若$A::=\alpha B$ 或$A::=\alpha B\beta$，且$\beta \Rightarrow^{*} \varepsilon$则把FOLLOW(A)加入FOLLOW(B)

注：FOLLOW集合中不能有ε

2.构造集合FOLLOW的算法

设S, A, B$\in V_n$,
算法：连续使用以下规则，直至FOLLOW集合不再扩大

(1) 若S为识别符号,则把“#”加入FOLLOW(S)中

(2) 若A::=αBβ (β≠ε),则把FIRST(β)-{ε}加入FOLLOW(B)

(3) 若A::=αB 或A::=αBβ，且β$\Rightarrow^{*}$ε则把FOLLOW(A)加入FOLLOW(B)

注：FOLLOW集合中不能有ε

解：（1）FIRST(S) = {a, ε}　　S → aABbcd | ε

FIRST(A) = {a,d, ε}　　A → ASd | ε

FIRST(B) = {a,d,h,e, ε}　　B → SAh | eC | ε

FIRST(C) = {a,f,g, ε}　　C → Sf | Cg | ε

FIRST(D) = {a, ε}　　D → aBD | ε

FOLLOW(S) = {d,a,f,h#}

FOLLOW(A) = {a,h,e,b,d}

FOLLOW(B) = {b,a}

FOLLOW(C) = {g,b,a}

(2) FIRST(aABbcd) = {a}　FIRST(ε) = {ε}

FIRST(ASd) = {a,d}　FIRST(ε) = {ε}

FIRST(SAh) = {a,d,h}　FIRST(eC) = {e}　FIRST(ε) = {ε}

FIRST(Sf) = {a,f}　FIRST(Cg) = {a,f,g}　FIRST(ε) = {ε}

(3) 不是LL(1)文法，因

FIRST(Sf) ∩ FIRST(Cg) = {a,f}∩ {a,f,g}≠ ∮

或FOLLOW(S) ∩FIRST(aABbcd) = {d,a,f,h#} ∩{a}≠ ∮

或FOLLOW(A) ∩FIRST(ASd) = { a,h,e,b,d } ∩{ a,d }≠ ∮

或FOLLOW(B) ∩FIRST(SAh) = { a,b } ∩{ a,d,h }≠ ∮

或FOLLOW(C) ∩FIRST(Sf) = { g,a,b } ∩{ a,f }≠ ∮

或FOLLOW(C) ∩FIRST(Cg) = { g,a,b } ∩{ a,f ,g}≠ ∮

P88: 6. 一个文法G是LL(1)的必要与充分条件是什么？试证明之。

充要条件是：对于G的每一个非终结符A的任何两条不同规则A::= α|β, 有：

(1) FIRST(α)∩FIRST(β)= ∅

(2) 假若β=*=>ε, 则FIRST(α)∩FOLLOW(A)= ∅

证明：

充分性：条件（1）（2）成立

反证：若分析表中存在多重入口，即

M[B,a] = { B::= α_1, B::= α_2} ,表明FIRST(α_1)∩FIRST(α_2) ≠ ∅

或M[B,a] = { B::= α_1, B::= α_2}，其中α_2= ε 或α_2=+=> ε

表明FIRST(α_1)∩FOLLOW(B) ≠ ∅

与条件（1）（2）矛盾。

必要性：文法是LL(1)文法，即分析表中不含多重入口

若条件(1)不成立，即存在某非终结符B的两条规则B::= α_1|α_2，

FIRST(α_1)∩FIRST(α_2) ≠ ∅

则对任意的a∈FIRST(α_1)∩FIRST(α_2)，有

M[B,a] = { B::= α_1, B::= α_2},矛盾

若条件（2）不成立，即存在某非终结符B的两条规则B::= α_1|α_2，α_2=*=> ε

有FIRST(α_1)∩FOLLOW(B) ≠ ∅

则对任意的a∈FIRST(α_1)∩FOLLOW (B)，有

M[B,a] = { B::= α_1, B::= α_2},矛盾

补充题：有文法G[S]：　S::= iCtSS'|a
S'::=eS|ε
C ::= b

(1) 证明该文法是二义性文法
(2) 求FIRST和FOLLOW
(3) 构造LL分析表，证明该文法是非LL(1)文法

解：

(1) 证明：对于句子ibtibtaea 有两棵语法树，所以该文法是二义文法。

(2) FIRST(iCtSS') = {i}　　FIRST(a) = {a}
FIRST(eS) = {e}　　FIRST(ε) = {ε}
FIRST(b) = {b}
FOLLOW(S) = {#,e}　　FOLLOW(S') = {#,e}　　FOLLOW(C) = {t}

(3) FIRST(eS) ∩FOLLOW(S') = {e},所以文法非LL（1）。

P100. 2.试用直观算符优先分析法分析下述表达式:

（2）a+b*(c+d)-e

判明是否是下述文法的合法句子，并列出分析过程。

文法 E::= E+E |
E-E |
E*E |
E↑E|
i |
(E)

步骤	符号栈	输入串	优先关系	动作
1	#	i+i*(i+i)-i#	<	移进
2	# i	+i*(i+i)-i#	>	归约
3	# E	+i*(i+i)-i#	<	移进
4	# E+	i*(i+i)-i#	<	移进
5	# E+i	*(i+i)-i#	>	归约
6	# E+E	*(i+i)-i#	<	移进
7	# E+E*	(i+i)-i#	<	移进
8	# E+E*(	i+i)-i#	<	移进
9	# E+E*(i	+i)-i#	>	归约
10	# E+E*(E	+i)-i#	<	移进
11	# E+E*(E+	i)-i#	<	移进
12	# E+E*(E+i	)-i#	>	归约
13	# E+E*(E+E	)-i#	>	归约
14	# E+E*(E	)-i#	=	移进
15	# E+E*(E)	-i#	>	归约
16	# E+E*E	-i#	>	归约
17	# E+E	-i#	>	归约
18	# E	-i#	<	移进
19	# E-	i#	<	移进
20	# E-i	#	>	归约
21	# E-E	#	>	归约
22	# E	#		接受

P100 4.有文法G[E]: E::= E+T | T
T::= T*F | F
F::= (E) | i

列出下述句型的短语和素短语：E、T、i、T*F、F*F、i*F 、F* i、F+F+F

解：

句型	短语	素短语
E		
T	T	
i	i	i
T*F	T*F	T*F
F*F	F，F*F	F*F
i*F	i，i*F	i
F* i	F，i，F* i	i
F+F+F	F,F,F,F+F, F+F+F	F+F

P100 5. 利用图4.10中的优先矩阵及图4.12的识别算法，
分析文法（4-14）的下列句子： i+i, i*(i*i)

E::= E+T | T
T::= T*F | F
F::= (E) | i

i+i

步骤	S(1)…	关系	R	T(K)…
0	#	<	i	+i#
1	#i	>	+	i#
2	#E	<	+	i#
3	#E+	<	i	#
4	#E+i	>	#	
5	#E+T	>	#	
6	#E			

i*(i*i)

步骤	S(1)…	关系	R	T(K)…
0	#	<	i	*(i*i)#
1	#i	>	*	(i*i)#
2	#T	<	*	(i*i)#
3	#T*	<	(	i*i)#
4	#T*(	<	i	*i)#
5	#T*(i	>	*	i)#
6	#T*(T	<	*	i)#
7	#T*(T*	<	i	)#
8	#T*(T*i	>	)	#
9	#T*(T*F	>	)	#
10	#T*(E	=	)	#
11	#T*(E)	>	#	
12	#T*F	>	#	
13	#E			

补充题：有如下文法G[E]：

E→E+T|T

T→E|(E)|i

1. 求每个非终结符的FIRSTVT和LASTVT集合

2. 构造算法优先关系矩阵

3. 判断该文法是否为算符优先文法。

1. 添加产生式：S→#E#

FIRSTVT(E)={+,(,i}　　LASTVT(E)={+,),i}

FIRSTVT(T)={+,(,i}　　LASTVT(T)={+,),i}

2.

	+	(	)	i	#
+	><	<	>	<	>
(	<	<	=	<	
)	>		>		>
i	>		>		>
#	<	<		<	

3. 非算符优先文法

P104: 1. 考虑具有下列规则的文法

S→E# E→T|E+T T→P| P↑T P→F|P*F F→i|(E)

(a) 下列句型的最右推导步骤中，其活前缀的集合是什么？

(1) E+i*i#

(2) E+P↑(i + i) #

(b) 为下列输入串构造最右推导的逆:

(1) i+i*i#

(2) i+i↑(i+i)#

解: (a)（1）句柄为i，所以活前缀集合为：E，E，E + i

（2）句柄为i，所以活前缀集合为：E，E+, E+P, E+P↑, E+P↑(, E+P↑(I

(b)（1）

i+i*i# <=

F+i*i# <=

P + i*i# <=

T+i*i# <=

E+i*i# <=

E+F*i# <=

E+P*i# <=

E+P*F# <=

E+P# <=

E+T# <=

E# <=

S

P104: 2. 用示例文法（4-16)的分析器，对下列输入串进行LR分析（列出分析过程）。

(a) i+i-i#

步骤	栈内符号	输入串	活前缀	句柄
0	#0	i+i-i#		
1	#0i3	+i-i#	i	i
2	#0T2	+i-i#	T	T
3	0E1	+i-i#	E	
4	0E1+6	i-i#	E+	
5	0E1+6i3	-i#	E+i	i
6	#0E1+6T8	-i#	E+T	E+T
7	#0E1	-i#	E	
8	#0E1-7	i#	E-	
9	#0E1-7i3	#	E-i	i
10	#-E1-7T9	#	E-T	E-T
11	#0E1	#	E	
12	#0E1#5		E#	
13	#E			

P108: 1. 给定具有如下产生式的文法

S→E#　E→E-T|T　T→F|F↑T　F→i|(E)

试求下列活前缀的有效项目集:

(a) F↑ (b) E-(　　(c) E-T

解: (a)E↑

{ T→.F　T→F↑.T　T→.F↑T　F→.i　F→.(E) }

(b)E-(

{F→ (.E), E →.E-T, E →.T, T →.F, T →.F↑T, T →.i, T →.(E)}

(c)E - T

{E →E-T. }

P115: 1. 给定下列产生式的文法:

（注：此做法为讲义上的方法，拓广文法时不加＃）

S→E　E→T|E+T　T→P|T*P　P→F|F↑P　F→i|(E)

(a) 为该文法构造LR（0）机器。

(b) 证明该文法是SLR（1）。

(c) 为该文法构造SLR（1）分析表。

解:(a)构造LR(0)机器

C_0:{S→.E, E→.T, E→.E+T, T→.P, T→.T*P, P→.F, P→.F↑P, F→.i, F→.(E)}

C_1:{S→E., E→E.+T}

C_2:{E→T., T→T.*P}

C_3:{T→P.}

C_4:{P→F., P→F.↑P}

C_5:{F→i.}

C_6:{F→(.E),E→.T,E→.E+T,T→.P,T→.T*P,P→.F,P→.F↑P,F→.i,F→.(E)}

C_7:{E→E+.T,T→.P,T→.T*P,P→.F,P→.F↑P,F→.i,F→.(E)}

C_8:{T→T*.P,P→.F,P→.F↑P,F→.i,F→.(E)}

C_9:{P→F↑.P, P→.F, P→.F↑P, F→.i,F →.(E)}

C_{10}:{F→(E.),E→E.+T}

C_{11}:{E→E+T.,T→T.*P}

C_{12}:{T→T*P.}

C_{13}:{P→F↑P.}

C_{14}:{F→(E).}

(b)对于状态C_2:{E→T.,T→T.*P}

Follow(E)={#,+,)}, *不属于Follow(E)

C_4:{P→F.,P→F.↑P}

Follow(P)={#,+,),*}, ↑不属于Follow(P)

C_{11}:{E→E+T.,T→T.*P}

Follow(E)={#,+,)} , *不属于Follow(E)

∴该文法构成的分析表不会出现多重定义出口。它是SLR(1)文法。

状态	S	E	T	P	F	+	*	↑	i	(	)	#
C0		C1	C2	C3	C4				S5	S6		
C1						S7						A
C2						r1	S8				r1	r1
C3						r3	r3				r3	r3
C4						r5	r5	S9			r5	r5
C5						r7	r7	r7			r7	r7
C6		C10	C2	C3	C4				S5	S6		
C7			C11	C3	C4				S5	S6		
C8				C12	C4				S5	S6		
C9				C13	C4				S5	S6		
C10						S7						
C11						r2	S8				r2	r2
C12						r4	r4				r4	r4
C13						r6	r6				r6	r6
C14						r8	r8	r8			r8	r8