chapter1

1、何谓源程序、目标程序、翻译程序、编译程序和解释程序？它们之间可能有何种关系？

源程序：用源语言编写的程序。

目标程序：源程序经翻译程序过加工处理后生成的程序。

翻译程序：将源程序转换为与其逻辑上等价的目标程序。

编译程序：源语言为高级语言，目标语言为汇编语言或机器语言的翻译程序。

① 先翻译后执行 ②执行速度快 ③多次运算

解释程序：源语言程序作为输入，但不产生目标程序，而是边解释边执行源程序本身。

① 边解释边执行 ② 有利于程序的调试 ③ 1次运算

chapter1

2、一个典型的编译系统通常由有哪些部分组成？各部分的主要功能是什么？

- 编译系统
  - 编译程序
    - 词法分析
    - 语法分析
    - 语义分析与中间代码生成
    - 优化
    - 目标代码生成
  - 运行系统

chapter1

① 词法分析(Lexical Analysis)：

从左到右一个字符一个字符的读入源程序，对构成源程序的字符串进行扫描和分解，从而识别出一个个单词（也称单词符号或简称符号）。

② 语法分析(Syntax Analysis)：

在词法分析的基础上将单词序列分解成各类语法短语，如“程序”，“语句”，“表达式”等等。

③ 语义分析(Syntactic Analysis)：

语义分析是在语法分析程序确定出语法短语后，审查有无语义错误，并为代码生成阶段收集类型信息。

chapter1

④ 中间代码生成(Generation of intermediate code)：

完成语法分析和语义处理工作后，编译程序将源程序变成一种内部表示形式，这种内部表示形式叫做中间语言或称中间代码，它是一种结构简单、含义明确的记号系统。

⑤ 代码优化(Optimization of code)：

为了使生成的目标代码更为高效，可以对产生的中间代码进行变换或进行改造，这就是代码的优化。

⑥ 代码生成(Generation of code)：

目标代码生成是编译器的最后一个阶段。在生成目标代码时要考虑以下几个问题：计算机的系统结构、指令系统、寄存器的分配以及内存的组织等。

chapter2

1.写出C语言和Java语言的输入字母表。

C语言：0~9数字，大小写英文字母，键盘上可见的字符

Java语言：Unicode可以包括的所有字符。

6.文法G_6为: N →D|ND
D →0|1|2|3|4|5|6|7|8|9

（1） G_6的语言是什么?

G_6的语言是： 0~9的数字组成的任意*非空*串

$L(G_6)=\{x|x\in\{0,1,2,3,4,5,6,7,8,9\}^+\}$

（2）给出句子0127、34和568的最左和最右推导。

7、 写一文法，使其语言是奇数集。 要求：不以0打头。

(一位) :D → 1|3|5|7|9

复杂的情况:分三部分

末尾：以1|3|5|7|9结尾 D→1|3|5|7|9

开头：除了0的任意数字 B→2|4|6|8|D

中间部分：空或者任意数字串 C→CA|ε

A→0|B

所以题目要求的文法G[N]可以写成:

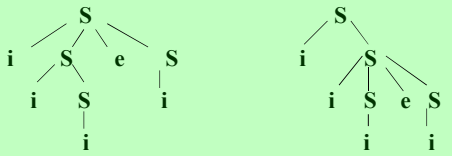
N → BCD|D
D → 1|3|5|7|9
B → 2|4|6|8|D
C → CA |ε
A → 0 | B

9、证明文法: S → iSeS | iS | i 是二义的。

二义性的含义: 如果文法存在某个句子对应两棵以上不同的语法树，或者两种以上不同的最左/右推导，则称这个文法是二义的。

首先：找到此文法对应的一个句子 iiiei

其次：构造与之对应的两棵语法树



结论：因为该文法存在句子iiiei对应两棵不同的语法树，因而该文法是二义的。

11、给出下面语言的相应文法

$L_1=\{a^nb^nc^i|\ n\geq1,i\geq0\}$

从n，i的不同取值来把L_1分成两部分：

前半部分是 $a^n\,b^n$： A →aAb | ab

后半部分是 c^i： B →Bc | ε

所以整个文法G_1[S]可以写为：

$G_1(S)$: S→AB
A→aAb|ab
B→cB|ε

$L_2=\{a^ib^nc^n \mid n\geq 1, i\geq 0\}$

$G_2(S)$: $S\rightarrow AB$

$A\rightarrow aA|\varepsilon$

$B\rightarrow bBc|bc$

$L_3=\{a^nb^na^mb^m \mid m,n\geq 0\}$

$G_3(S)$: $S\rightarrow AB$

$A\rightarrow aAb|\varepsilon$

$B\rightarrow aBb|\varepsilon$

$L_4=\{1^n\ 0^m\ 1^m\ 0^n \mid n,m\geq 0\}$

可以看成是两部分：

中间部分是 $0^m\ 1^m$ ： $A\rightarrow 0A1 \mid \varepsilon$

剩下两边的部分就是： $S\rightarrow 1S0 \mid A$

所以$G_4[S]$可以写为：

$S\rightarrow 1S0 \mid A$

$A\rightarrow 0A1 \mid \varepsilon$

chapter3

7.构造下列正规式相应的DFA。

问题：将状态转换图与DFA混淆。

步骤：

①. 根据正规式画出对应的状态转换图;

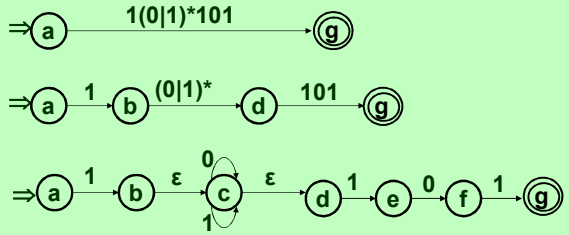
②. 根据状态转换图画出对应的状态转换矩阵;

③. 根据状态转换矩阵得到重命名后的状态转换矩阵;

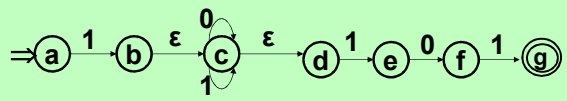
④. 根据重命名后的状态转换矩阵得出最后的DFA.

1(0|1)*101

①.状态转换图

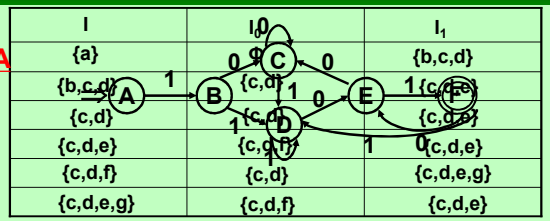


②.状态转换矩阵



I	I_0	I_1
{a}	Φ	{b,c,d}
{b,c,d}	{c,d}	{c,d,e}
{c,d}	{c,d}	{c,d,e}
{c,d,e}	{c,d,f}	{c,d,e}
{c,d,f}	{c,d}	{c,d,e,g}
{c,d,e,g}	{c,d,f}	{c,d,e}

④.DFA



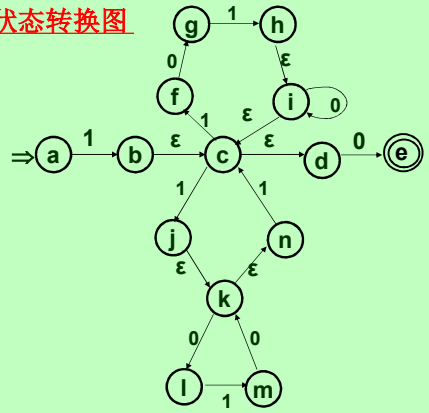
I	I_0	I_1
{a}	Φ	{b,c,d}
{b,c,d}	{c,d}	{c,d,e}
{c,d}	{c,d}	{c,d,e}
{c,d,e}	{c,d,f}	{c,d,e}
{c,d,f}	{c,d}	{c,d,e,g}
{c,d,e,g}	{c,d,f}	{c,d,e}

③. 重命名后的状态转换矩阵

S	0	1
A(始态)	Φ	B
B	C	D
C	C	D
D	E	D
E	C	F(终态)
F(终态)	E	D

1(1010*|1(010)*1)*0

①.状态转换图



②.状态转换矩阵

I	I_0	I_1
{0}	Φ	{1,2,3}
{1,2,3}	{4}	{5,9,10,11}
{5,9,10,11}	{6,12}	{2,3}
{6,12}	Φ	{2,3,7,8,13}
{2,3}	{4}	{5,9,10,11}
{2,3,7,8,13}	{2,3,4,8,10,11}	{5,9,10,11}
{2,3,4,8,10,11}	{2,3,4,8,12}	{2,3,5,9,10,11}
{2,3,4,8,12}	{2,3,4,8}	{5,9,10,11,13}
{2,3,5,9,10,11}	{4,6,12}	{2,3,5,9,10,11}
{2,3,4,8}	{2,3,4,8}	{5,9,10,11}
{5,9,10,11,13}	{6,10,11,12}	{2,3}
{4,6,12}	Φ	{2,3,7,8,13}
{6,10,11,12}	{12}	{2,3,7,8,13}
{12}	Φ	{13}
{13}	{10,11}	Φ
{10,11}	{12}	{2,3}

③. 重命名后的状态转换矩阵　　④.DFA

I	I_0	I_1
1	Φ	2
2	3	4
4	5	6
5	Φ	7
6	3	4
7	8	4
8	9	10
9	11	12
10	13	10
11	11	4
12	14	6
13	Φ	7
14	15	7
15	Φ	16
16	17	Φ
17	15	6

8、给出下面正规表达式

（1）以01结尾的二进制数串。

(0 | 1)*01

（2）能被5整除的十进制数。

0 | 5 | (1|2|3|4|5|6|7|8|9) (0|1|2|3|4|5|6|7|8|9)*(0 |5)

（3）包含奇数個1或奇数個0的二進制串

0*1(0|10*1)*|1*0(0|10*1)*

（4）英文字母组成的所有符号串，要求符号串中的字母按字典序排列。

(A | a)* (B | b)* (C | c)*… (Z | z)*

8、给出下面正规表达式

（5）沒有重複出現的數字的數字符號串的全體

令r_i=i|ε ,i=0,1,2...9

R0|R1|R2|...|R9記為∑Ri i ∈(0,1,2...,9)

P（0,1,2...,9)表示0,1,2...,9的全排列

$\sum r_{i0}r_{i1}...r_{i9}$

$r_{i0}r_{i1}...r_{i9}$ ∈P(0,1,2...,9)

（6）最多有一個重複出現的數字的數字符號串的全體

∑i $\sum r_{i0}r_{i1}...r_{i9}$

i ∈(0,1,2...,9) ri0ri1...ri9 ∈P(0,1,2...,9)

（7）不包含字串abb的由a和b組成的符號串的全體

b*(a*|(ba)*)*

9、对下面情况给出DFA及正规表达式：

（1）{0,1}上的含有子串010的所有串。

正规式：(0 | 1)* 010 (0 | 1)*

（2）{0,1}上不含子串010的所有串。

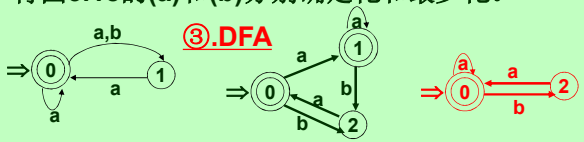
正规式：1*(0|11*1)*

1*(0 | 11)*1*　　1*0*1*　　(0 | 11)*(0 | 1)

DFA做法同第7题。

12、将图3.18的(a)和(b)分别确定化和最少化。

(a)



③.DFA

①.状态转换矩阵

I	I_a	I_b
{0}	{0,1}	{1}
{0,1}	{0,1}	{1}
{1}	{0}	φ

②. 重命名后的状态转换矩阵

I	I_a	I_b
0	1	2
1	1	2
2	0	φ

④.最小化

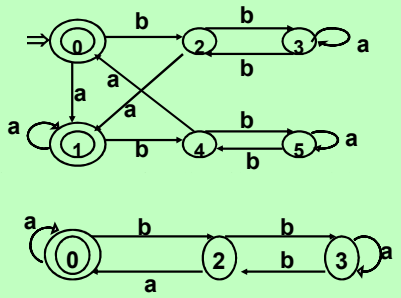
$\Pi_0=(\{0,1\},\{2\})$

$\{0,1\}_a=\{1\}$

$\{0,1\}_b=\{2\}$

因此，不能再分

(b)



{2，4}a={0，1} {2，4}b={3，5} 所以2，4等价

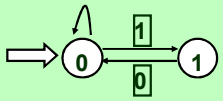
{3，5}a={3，5} {3，5}b={2，4} 所以3，5等价

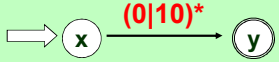
所以分为 {0，1} {2，4} {3，5}

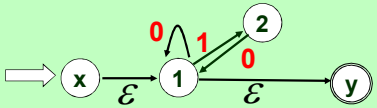
14、构造一个DFA，它接受Σ={0,1}上所有满足如下条件的字符串：每个1都有0直接跟在右边。

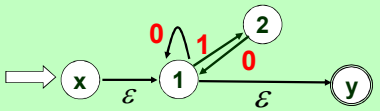
思路：先写出满足条件的正规式，由正规式构造NFA，再把NFA确定化和最小化。

满足条件的正规式：(0|10)*







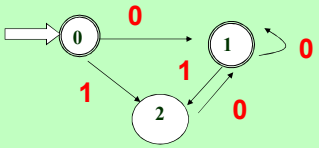


确定化：

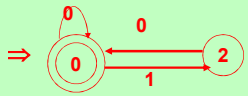
	0	1
{X, 1, Y}	{1, Y}	{2}
{1, Y}	{1, Y}	{2}
{2}	{1, Y}	Φ

给状态编号：

	0	1
0	1	2
1	1	2
2	1	Φ



最小化：
{0,1}，{2}
$\{0,1\}_0$={1} $\{0,1\}_1$={2}
$\{2\}_0$={0}，$\{2\}_1$= ∅ ⊆ {2}或{0,1}
所以{0,1}不可分，用狀態0代表它們

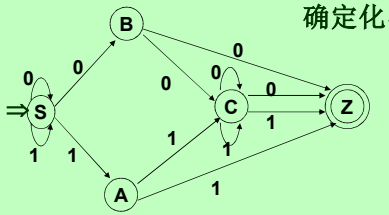


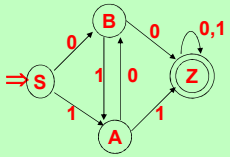
15、给定右线性文法G：求一个与G等价的左线性文法。

S → 0S | 1S | 1A | 0B
A → 1C | 1
B → 0C | 0
C→ 0C | 1C | 0 | 1

G[Z]:
Z → Z0|Z1|B0|A1
B → A0 | 0
A → B1 | 1

确定化、最小化后的DFA为：

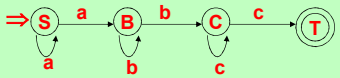




补充：构造一右线性文法，使它与如下文法等价：
S→AB A→UT U→a|aU T→b|bT B→c|cB
并根据所得右线性文法，构造出相应的状态转换图。

思路：
先写出原文法所描述的语言 $L(G)=\{a^m b^n c^k | m,n,k\geq 1\}$

G[S]: S →aS|aB
B →bB|bC
C →cC|c



chapter4

4.1、考虑下面文法G1：S → a | ∧ | （T）
T → T，S | S

（1）消去G1的左递归；

S → a | ∧ | （T）
T → ST’
T’ → ，S T’ |ε

（2）经改写后的文法是否是LL(1)文法，给出预测分析表。

经改写后的文法满足3个条件，所以是LL(1)的

预测分析表构造算法:

1.对文法中的每个产生式A → α执行第二步和第三步;

FIRST(S)={a,∧,(} FIRST(T)={a,∧,(} FIRST(T')= {, , ε }

FOLLOW(S) ={),, ,#} FOLLOW(T) ={)} FOLLOW(T) ={)}

	a	∧	(	,	)	#
S	S →a	S →∧	S →(T)			
T	T →ST'	T →ST'	T →ST'			
T'				T' →,ST'	T' → ε	

预测分析表构造算法:

1.对文法中的每个产生式A → α执行第二步和第三步;

S →a; S →∧; S →(T); T →ST'; T' →,ST' T' → ε

2.对每个终结符a ∈FIRST(α),把A →a加到M[A,a]中;

FTRST（a）={a} FIRST（∧）={∧}

FIRST（（T））={（} FIRST（ST'）={a，∧，（}

FIRST（，ST'）={，} FIRST（ε）={ε}

3.若ε ∈FIRST(α),则对于任何b ∈FOLLOW(A)把A→ α加至M[A,b]中

FOLLOW(T')=FOLLOW(T)={)}

	a	∧	(	,	)	#
S	S →a	S →∧	S →(T)			
T	T →ST'	T →ST'	T →ST'			
T'				T' →,ST'	T' → ε	

递归子程序:

```
procedure S;
begin
      if sym='a' or sym='^'
            then abvance
      else if sym='('
            then begin
                 advance;T;
                 if sym=')' then advance;
                 else error;
                end
      else  error
end;
```



```
procedure T;
begin
      S;T'
End
procedure T';
begin
      if sym=','
          then bengin
               advance;
               S;T'
      end
End
```

sym:是输入串指针IP所指的符号

advance:是把IP调至下一个输入符号

error:是出错诊察程序

补充题：有文法：

E →TE’
E’ →ATE’ |ε
T →FT’
T’ →MFT’ |ε
F →（E）| i
A →+ | -
M →* | /

（1）求First、Follow集，判断是否是LL（1）文法？
（2）若是构造LL（1）分析表？
（3）简述LL（1）分析器的工作原理。

4.2：有文法：

E →TE’
E’ →+E |ε
T →FT’
T’ →T |ε
F →PF’
F’ →*F’ | ε
P →(E) |a|b|^

（1）求First、Follow集，判断是否是LL（1）文法？
（2）若是构造LL（1）分析表？
（3）简述LL（1）分析器的工作原理。

	i	+	-	*	/	(	)	#
E	E →TE’					E → TE’		
E’		E → ATE’	E →ATE’				E’ → ε	E’ → ε
T	T → FT’					T → FT’		
T’		T’->ε	T’ → ε	T’ → MFT’	T’ → MFT’		T’ → ε	T’ → ε
F	F → i					F → (E)		
A		A → +	A → -					
M				M → *	M → /			

FIRST(E)={(， i}
FIRST(E’)={+，-， ε}
FIRST(T)={(， i}
FIRST(T’)={*，/， ε}
FIRST(F)={(， i}
FIRST(A)={+，-}
FIRST(M)={*， /}

FOLLOW(E)={#，)}
FOLLOW(E’)={#，)}
FOLLOW(T)= {+，-，#，)}
FOLLOW(T’)= {+，-，#，)}
FOLLOW(F)={*，/，+，-，#，)}
FOLLOW(A)= {(， i}
FOLLOW(M)= {(， i}

	(	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’→T	T’→T	T’→T	T’→T	T’→ ε		T’→ ε	T’→ ε
F	F→PF’	F→PF’	F→PF’	F→PF’				
F’	F’→ ε	F’→ ε	F’→ ε	F’→ ε	F’→ ε	F’→F’	F’→ ε	F’→ ε
P	P→(E)	P→a	P→b	P→^				

1)FIRST(E)=FIRST(T) =FIRST(F) =FIRST(P) ={(,a,b, ∧}
FIRST(E’) ={+, ε} FIRST(F’) ={*, ε}
FIRST(T’) =FIRST(T) ∪ {ε} = {(,a,b, ∧, ε}
FOLLOW(E) ={#,)} FOLLOW(E’) =FOLLOW(E)= {#,)}
FOLLOW(T) =FIRST(E’)\ ε ∪FOLLOW(E) = {+,#,)}
FOLLOW(T’) =FOLLOW(T)= {+,#,)}
FOLLOW(F) =FIRST(T’)\ ε ∪FOLLOW(T) ={(,a,b, ∧, +,#,)}
FOLLOW{F’) =FOLLOW(F) ={(,a,b, ∧, +,#,)}
FOLLOW(P) =FIRST(F’)\ ε ∪FOLLOW(F) = {*, (,a,b, ∧, +,#,)}

- 2)考虑下列产生式:
- FIRST(+E)∩FIRST(ε)={+}∩{ε}=φ
- FIRST(+E)∩FOLLOW(E')={+}∩{#,)}=φ
- FIRST(T)∩FIRST(ε)={(,a,b,^}∩{ε}=φ
- FIRST(T)∩FOLLOW(T')={(,a,b,^}∩{+,),#}=φ
- FIRST(*F')∩FIRST(ε)={*}∩{ε}=φ
- FIRST(*F')∩FOLLOW(F')={*}∩{(,a,b,^,+,),#}=φ
- FIRST((E))∩FIRST(a) ∩FIRST(b) ∩FIRST(^)=φ
- 所以,该文法式LL(1)文法.

3)預测分析表：

	(	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'→T	T'→T	T'→T	T'→T	T'→ ε		T'→ ε	T'→ ε
F	F→PF'	F→PF'	F→PF'	F→PF'				
F'	F'→ ε	F'→ ε	F'→ ε	F'→ ε	F'→ ε	F'→F'	F'→ ε	F'→ ε
P	P→(E)	P→a	P→b	P→^				

4)程序

```
procedure E;
begin
        if sym='(' or sym='a' or sym='b' or sym='^'
           then begin T; E' end
           else error
end
procedure E';
begin
        if sym='+'
           then begin advance; E end
           else if sym<>')' and sym<>'#' then error
end
procedure T;
begin
        if sym='(' or sym='a' or sym='b' or sym='^'
           then begin F; T' end
           else error
end
```



```
procedure T';
begin
        if sym='(' or sym='a' or sym='b' or sym='^'
           then T
           else if sym='*' then error
end
procedure F;
begin
        if sym='(' or sym='a' or sym='b' or sym='^'
           then begin P; F' end
           else error
end
procedure F';
begin
        if sym='*'
           then begin advance; F' end
end
```

```
procedure P;
begin
        if sym='a' or sym='b' or sym='^'
           then advance
         else if sym='(' then
                 begin
                         advance; E;
                         if sym=')' then advance
                                 else error
                 end
         else error
end;
```


■4.3下面文法中，那些是LL(1)文法的，說明理由

■构造不带回溯的自上而下分析的文法条件

1. 文法不含左递归，

2. 对于文法中每一个非终结符A的各个产生式的候选首符集两两不相交。即，若$A\rightarrow\alpha_1|\alpha_2|\ldots|\alpha_n$

则 $FIRST(\alpha_i)\cap FIRST(\alpha_j)=\phi \quad (i\neq j)$

3. 对文法中的每个非终结符A，若它存在某个候选首符集包含ε，则$FIRST(A)\cap FOLLOW(A)=\phi$

如果一个文法G满足以上条件，则称该文法G为LL(1)文法。

4.3.1

S→Abc

A→a| ε

B→b| ε

是，满足三个条件

4.3.2

S→Ab

A→a|B| ε

B→b| ε

对于A不满足条件3

4.3.3

S→ABBA

A→a| ε

B→b| ε

A、B都不满足条件3

4.3.4

S→aSe|B

B→bBe| **ε**

C→cCe| **d**

满足条件3

- 4.4對下面的文法：
- Expr→-Expr
- Expr→(Expr)|Var ExprTail
- ExprTail→-Expr| **ε**
- Var→id VarTail
- VarTail→(Expr)| **ε**

(1)構造LL(1)分析表

(2)給出對句子id- - id((id))分析過程

解題思路：構造文法的預測分析表，通常應當按下列步驟進行：

1.消除文法的左遞歸（包括所有直接左遞歸和間接左遞歸）

2.對消除左遞歸后的文法，提取公因子

3.對經過上述改造后的文法，計算它的每個非終結符的FIRST集合和FOLLOW集合；

4.根據FIRST集合和FOLLOW集合構造預測分析表：

第1步對文法G的每個產生式A→α執行第1步和第3步；

第2步對每個終結符a∈FIRST(α),把A→α加至M[A,a]中；

第3步若ε∈ FIRST(α),則對任何b ∈FIRST(A),把A→α加至M[A,b]中；

第4步把所有無定義的M[A,a]標上“出錯標誌”

- 4.4對下面的文法：
- Expr→-Expr
- Expr→(Expr)|Var ExprTail
- ExprTail→-Expr| ε
- Var→id VarTail
- VarTail→(Expr)| ε

(1)構造LL(1)分析表
(2)給出對句子id- - id((id))分析過程

解答：	
FIRST(Expr)={-,(,id }	FOLLOW(Expr)={),# }
FIRST(ExprTail)={-, ε}	FOLLOW(ExprTail)={),# }
FIRST(Var)={id}	FOLLOW(Var)={-,),# }
FIRST(VarTail)={(,ε}	FOLLOW(VarTail)={-,),# }

- 4.4對下面的文法：
- Expr→-Expr
- Expr→(Expr)|Var ExprTail
- ExprTail→-Expr| ε
- Var→id VarTail
- VarTail→(Expr)| ε

(1)構造LL(1)分析表
(2)給出對句子id- - id((id))分析過程

	-	id	(	)	#
Expr	Expr→-Expr	Expr→Var ExprTail	Expr→(Expr)		
ExprTai	ExprTail→-Expr			ExprTail→ε	ExprTail→ε
Var		Var→id VarTail			
VarTail	VarTail→ε		VarTail→(Expr)	VarTail→ε	VarTail→ε

	-	id	(	)	#
Expr	Expr→-Expr	Expr→Var ExprTail	Expr→(Expr)		
ExprTai	ExprTail→-Expr			ExprTail→ε	ExprTail→ε
Var		Var→id VarTail			
VarTail	VarTail→ε		VarTail→(Expr)	VarTail→ε	VarTail→ε

(2)給出對句子id- - id((id))分析過程

步驟	符號棧	輸入串	所用產生式
0	#Expr	id- -id((id))#	開始
1	#ExprTail var	id- -id((id))#	Expr→Var ExprTail
2	#ExprTail varTail id	id- -id((id))#	Var→id VarTail
3	#ExprTail varTail	- -id((id))#	出棧，輸入串後移
4	#ExprTail	- -id((id))#	VarTail→ε
5	#Expr -	- -id((id))#	ExprTail→-Expr

	-	id	(	)	#
Expr	Expr→-Expr	Expr→Var ExprTail	Expr→(Expr)		
ExprTai	ExprTail→-Expr			ExprTail→ε	ExprTail→ε
Var		Var→id VarTail			
VarTail	VarTail→ε		VarTail→(Expr)	VarTail→ε	VarTail→ε

(2)給出對句子id- - id((id))分析過程

步驟	符號棧	輸入串	所用產生式
5	#Expr -	- -id((id))#	ExprTail→-Expr
6	#Expr	-id((id))#	出棧，輸入串後移
7	#Expr -	-id((id))#	Expr→-Expr
8	#Expr	id((id))#	出棧，輸入串後移
9	#ExprTail Var	id((id))#	Expr→Var ExprTail
10	#ExprTail VarTail id	id((id))#	Var→id VarTail
11	#ExprTail VarTail	((id))#	出棧，輸入串後移

	-	id	(	)	#
Expr	Expr→-Expr	Expr→Var ExprTail	Expr→(Expr)		
ExprTai	ExprTail→-Expr			ExprTail→ε	ExprTail→ε
Var		Var→id VarTail			
VarTail	VarTail→ε		VarTail→(Expr)	VarTail→ε	VarTail→ε

(2)給出對句子id- - id((id))分析過程

步驟	符號棧	輸入串	所用產生式
11	#ExprTail VarTail	((id))#	出棧，輸入串後移
12	#ExprTail)Expr(	((id))#	VarTail→(Expr)
13	#ExprTail)Expr	(id))#	出棧，輸入串後移
14	#ExprTail))Expr(	(id))#	Expr→(Expr)
15	#ExprTail))Expr	id))#	出棧，輸入串後移
16	#ExprTail)) ExprTail Var	id))#	Expr→Var ExprTail
17	#ExprTail)) ExprTail VarTail id	id))#	Var→id VarTail

	-	id	(	)	#
Expr	Expr→-Expr	Expr→Var ExprTail	Expr→(Expr)		
ExprTai	ExprTail→-Expr			ExprTail→ε	ExprTail→ε
Var		Var→id VarTail			
VarTail	VarTail→ε		VarTail→(Expr)	VarTail→ε	VarTail→ε

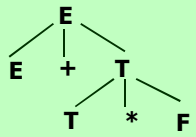
(2)給出對句子id- - id((id))分析過程

步驟	符號棧	輸入串	所用產生式
17	#ExprTail)) ExprTail VarTail id	id))#	Var→id VarTail
18	#ExprTail)) ExprTail VarTail	))#	出棧，輸入串後移
19	#ExprTail)) ExprTail	))#	VarTail→ε
20	#ExprTail))	))#	ExprTail →ε
21	#ExprTail)	)#	出棧，輸入串後移
22	# ExprTail	#	出棧，輸入串後移
23	#	#	ExprTail→ε
成功			

chapter5

1、令文法G1为：

E→E+T | T
T→T*F | 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。

T*F是句型E+T*F相对于T的短语

E+T*F句型E+T*F相对于E的短语

T*F是句型E+T*F相对于T的直接短语

T*F是句柄

2、考虑下面的表格结构文法G_2：

S→a | ^ | (T)
T →T,S | S

（1）给出(a,(a,a))和(((a,a),^,(a)),a)的最左和最右推导。

(a,(a,a))的最左推导：

S ⇒ (T) ⇒(T,S) ⇒(S,S) ⇒(a,S) ⇒(a,(T))
⇒(a,(T,S)) ⇒(a,(S,S)) ⇒(a,(a,S)) ⇒(a,(a,a))

(((a,a),^,(a)),a)的最左推导：

S ⇒ (T) ⇒(T,S) ⇒(S,S) ⇒((T),S) ⇒((T,S),S)
⇒((T,S,S),S) ⇒((S,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),^,a),S) ⇒ (((a,a),^,a),a)

(a,(a,a))的最右推导：

S ⇒ (T) ⇒(T,S) ⇒(T,(T)) ⇒(T,(T,S)) ⇒(T,(T,a))
⇒(T,(S,a)) ⇒(T,(a,a)) ⇒(S,(a,a)) ⇒(a,(a,a))

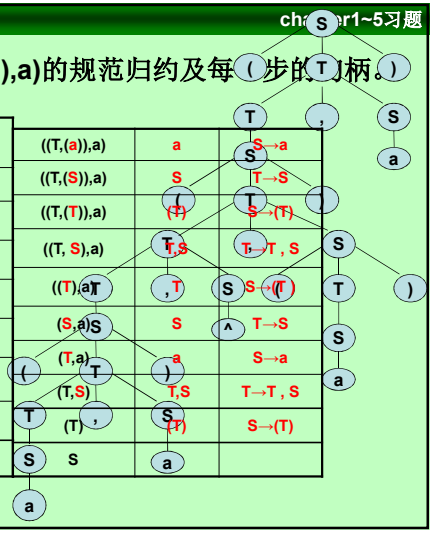
(((a,a),^,(a)),a)的最右推导：

S ⇒ (T) ⇒(T,S) ⇒(S,S) ⇒(S,a) ⇒((T),a)
⇒((T,S,S),S) ⇒((S,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),^,a),S) ⇒ (((a,a),^,a),a)

2）指出(((a,a),^,(a)),a)的规范归约及每一步的句柄。

句型	句柄	归约规则
(((a,a),^,(a)),a)	a	S→a
(((S,a),^,(a)),a)	S	T→S
(((T,a),^,(a)),a)	a	S→a
(((T,S),^,(a)),a)	T,S	T→T , S
(((T),^,(a)),a)	(T)	S→(T)
((S,^,(a)),a)	S	T→S
((T,^,(a)),a)	^	S→^
((T,S,(a)),a)	T,S	T→T , S

((T,(a)),a)	a	S→a
((T,(S)),a)	S	T→S
((T,(T)),a)	(T)	S→(T)
((T, S),a)	T,S	T→T , S
((T),a)	(T)	S→(T)
(S,a)	S	T→S
(T,a)	a	S→a
(T,S)	T,S	T→T , S
(T)	(T)	S→(T)
S	a	

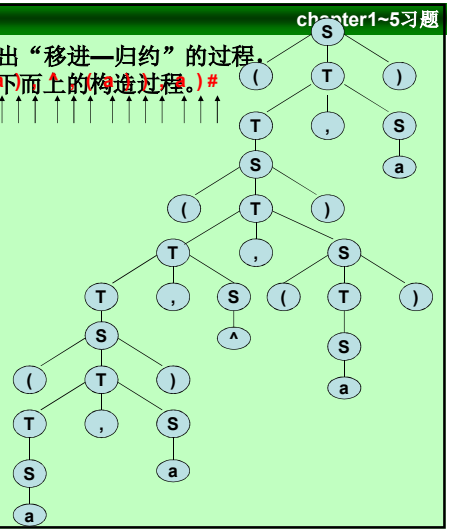


根据这个规范规约，给出“移进—归约”的过程，
并给出它的语法树的自下而上的构造过程。

输入串：a , (a , ^ , (a)) #

句柄	归约规则
a	S→a
S	T→S
a	S→a
T,S	T→T , S
(T)	S→(T)
S	T→S
^	S→^
T,S	T→T , S

符号栈
)
T
S
S
)
T
S
#



3、(1)计算练习2文法G2的FIRSTVT和LASTVT。

G2： S→a | ^ | (T)
T →T,S | S

FIRSTVT(S)={ a， ^，(}　　FIRSTVT(T)={， ， a， ^，(}
LASTVT(S)={a， ^，)}　　LASTVT(T)={， ， a， ^，)}

	a	^	(	)	,	#
a				>	>	>
^				>	>	>
(	<	<	<	=	<	
)				>	>	>
,	<	<	<	>	>	
#	<	<	<			=

1、文法是算术文法，且不含 ε 产生式。

2、由优先关系矩阵可知，任何两个终结符之间的优先关系不多于一种。

综上，该文法是算术优先文法。

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

输入串(a,(a,a))的算符优先过程。 问题：没有依据最左素短语进行规约

步骤	句型	优先关系	最左素短语	规约符号
1	#(a,(a,a))#	# ⋖ (⋖ a ⋗ , ⋖ (⋖ a ⋗ , ⋖ a ⋗) ⋗) ⋗ #	a	S
2	#(S,(a,a))#	# ⋖ (⋖ , ⋖ (⋖ a ⋗ , ⋖ a ⋗) ⋗) ⋗ #	a	S
3	#(S,(S,a))#	# ⋖ (⋖ , ⋖ (⋖ , ⋖ a ⋗) ⋗) ⋗ #	a	S
4	#(S,(S,S))#	# ⋖ (⋖ , ⋖ (⋖ , ⋗) ⋗) ⋗ #	S,S	T
5	#(S,(T))#	# ⋖ (⋖ , ⋖ (≐) ⋗) ⋗ #	(T)	S
6	#(S,S)#	# ⋖ (⋖ , ⋗) ⋗ #	S,S	T
7	#(T)#	# ⋖ (≐) ⋗ #	(T)	S
8	#S#			确认！

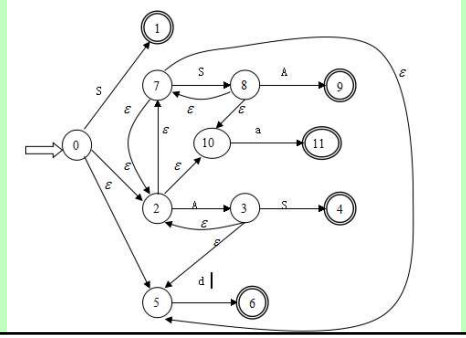
P134-5考虑文法S→AS|b A→SA|a
1、列出这个文法的所有LR(0)项目
2、构造这个文法的LR(0)项目集规范族及识别或前缀的DFA
3、这个文法是SLR的吗？若是，构造出它的SLR分析表
4、这个文法是LALR或LR(1)的吗

解答：1、

0.S'→·S	1.S'→S·	
2.S→·AS	3.S→A·S	4.S→AS·
5.S→·b	6.S→·b·	
7.A→·SA	8.A→S·A	9.A→SA·
10.S→·a	11.S→a·	

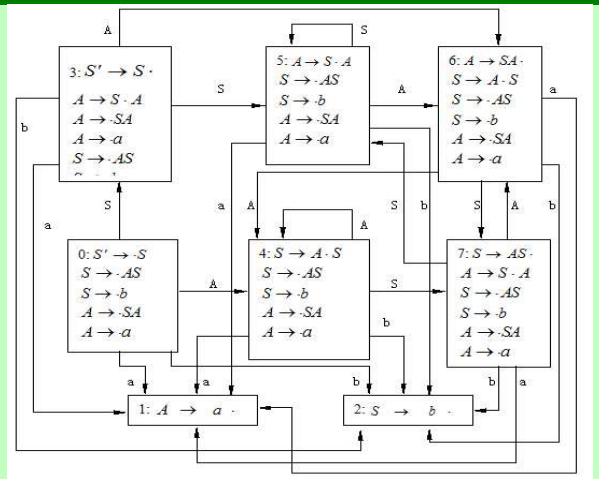
P134-5考虑文法S→AS|b A→SA|a
1、列出这个文法的所有LR(0)项目
2、构造这个文法的LR(0)项目集规范族及识别或前缀的DFA
3、这个文法是SLR的吗？若是，构造出它的SLR分析表
4、这个文法是LALR或LR(1)的吗

解答：1、



确定化：

	S	A	a	b
{0, 2, 5, 7, 10}	{1, 2, 5, 7, 8, 10}	{2, 3, 5, 7, 10}	{11}	{6}
{1, 2, 5, 7, 8, 10}	{2, 5, 7, 8, 10}	{2, 3, 5, 7, 9, 10}	{11}	{6}
{2, 3, 5, 7, 10}	{2, 4, 5, 7, 8, 10}	{2, 3, 5, 7, 10}	{11}	{6}
{2, 5, 7, 8, 10}	{2, 5, 7, 8, 10}	{2, 3, 5, 7, 9, 10}	{11}	{6}
{2, 3, 5, 7, 9, 10}	{2, 4, 5, 7, 8, 10}	{2, 3, 5, 7, 10}	{11}	{6}
{2, 4, 5, 7, 8, 10}	{2, 5, 7, 8, 10}	{2, 3, 5, 7, 9, 10}	{11}	{6}



构造LR(0)项目集规范族也可以用 GO 函数来计算得到。所得到的项目集规范族与上图中的项目集一样：

$I_0=\{S'\to\cdot S,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}$

$GO(I_0, a)=\{A\to a\cdot\}=I_1$

$GO(I_0, b)=\{S\to b\cdot\}=I_2$

$GO(I_0, S)=\{S'\to S\cdot,\ A\to S\cdot A,\ A\to\cdot SA,\ A\to\cdot a,\ S\to\cdot AS,\ S\to\cdot b\}=I_3$

$GO(I_0, A)=\{S\to A\cdot S,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}=I_4$

$GO(I_3, a)=\{A\to a\cdot\}=I_1$

$GO(I_3, b)=\{S\to b\cdot\}=I_2$

$GO(I_3, S)=\{A\to S\cdot A,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}=I_5$

$GO(I_3, A)=\{A\to SA\cdot,\ S\to A\cdot S,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}=I_6$

$GO(I_4, a)=\{A\to a\cdot\}=I_1$

$GO(I_4, b)=\{S\to b\cdot\}=I_2$

$GO(I_4, S)=\{S\to AS\cdot,\ A\to S\cdot A,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}=I_7$

$GO(I_4, A)=\{S\to A\cdot S,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}=I_4$

$GO(I_5, a)=\{A\to a\cdot\}=I_1$

$GO(I_5, b)=\{S\to b\cdot\}=I_2$

$GO(I_5, S)=\{A\to S\cdot A,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}=I_5$

$GO(I_5, A)=\{A\to SA\cdot,\ S\to A\cdot S,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}=I_6$

$GO(I_6, a)=\{A\to a\cdot\}=I_1$

$GO(I_6, b)=\{S\to b\cdot\}=I_2$

$GO(I_6, S)=\{S\to AS\cdot,\ A\to S\cdot A,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}=I_7$

$GO(I_6, A)=\{S\to A\cdot S,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}=I_4$

$GO(I_7, a)=\{A\to a\cdot\}=I_1$

$GO(I_7, b)=\{S\to b\cdot\}=I_2$

$GO(I_7, S)=\{A\to S\cdot A,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}=I_5$

$GO(I_7, A)=\{A\to SA\cdot,\ S\to A\cdot S,\ S\to\cdot AS,\ S\to\cdot b,\ A\to\cdot SA,\ A\to\cdot a\}=I_6$

项目集规范族为 $C=\{I_1, I_2, I_3, I_4, I_5, I_6, I_7\}$

(3)不是 SLR 文法

状态 3，6，7 有移进归约冲突

状态 3：FOLLOW(S′)={#}不包含 a,b

状态 6：FOLLOW(S)={#,a,b}包含 a,b,；移进归约冲突无法消解

状态 7：FOLLOW(A)={a,b}包含 a,b；移进归约冲突消解

所以不是 SLR 文法。

(4) 构造例如 LR(1)项目集规范族

见下图：

对于状态5,因为包含项目[$A\to AS\cdot$ a/b],所以遇到搜索符号 a或 b时,应该用 $A\to AS$ 归约。又因为状态 5 包含项目[$A\to\cdot a$ a/b]，所以遇到搜索符号 a 时，应该移进。因此存在“移进-归约”矛盾，所以这个文法不是 LR(1)文法。

P135-6

1. 构造下列文法的 LALR 分析表。

E->E+T|T

T->TF|F

F->F*|a|b

解： 此文法的拓展文法为：

(0) E`->E

(1) E->E+T

(2) E->T

(3) T->TF

(4) T->F

(5) F->F*

(6) F->a

(7) F->b

该文法的 LR(1)项目集规范族及其转换如下：

I_0:
E'-> • E, $
E-> • E+T, +/$
E-> • T, +/$
T-> • TF, a/b/+/$
T-> • F, a/b/+/$
F-> • F*, a/b/+/*/$
F-> • a, a/b/+/*/$
F-> • b, a/b/+/*/$

I_1:
E'->E •, $
E->E • +T, +/$ —+→ I6

I_2:
E->T •, +/$
T->T • F, a/b/+/$
F-> • F*, a/b/+/*/$
F-> • a, a/b/+/*/$
F-> • b, a/b/+/*/$

I_3:
T->F •, a/b/+/$
F->F • *, a/b/+/*/$ —*→ I8

I_4:
F->a •, a/b/+/*/$

I_5:
F->b •, a/b/+/*/$

I_6:
E->E+ • T, +/$
T-> • TF, a/b/+/$
T-> • F, a/b/+/$
F-> •F*, a/b/+/*/$
F-> • a, a/b/+/*/$
F-> • b, a/b/+/*/$

I_7:
T->TF •, a/b/+/$
F->F •*, a/b/+/*/$ —*→ I8

I_8:
F->F* •, a/b/+/*/$

I_9:
E->E+T •, +/$
T->T • F, a/b/+/$
F-> •F*, a/b/+/*/$
F-> • a, a/b/+/*/$
F-> • b, a/b/+/*/$

根据以上可得出文法的 LR(1)预测分析表如下：

表达式文法的 LR(1)分析表

状态	动作					转移		
	+	*	a	b	$	E	T	F
0			S4	S5		1	2	3
1	S6				acc			
2	r2		S4	S5	r2			7
3	r4	S8	r4	r4	r4			
4	r6	r6	r6	r6	r6			
5	r7	r7	r7	r7	r7			
6			S4	S5			9	3
7	r3	S8	r3	r3	r3			
8	r5		r5	r5	r5			
9	r1		S4	S5	r1			7

没有需要合并的状态，故 LALR 分析表同 LR（1）分析表

P135-7证明下面文法是SLR(1)文法，但不是LR(0)文法

S→A　　A→Ab|bBa　　B→aAc|a|aAb

解：文法G[S]：

0：S→A　　1：A→Ab　　2：A→bBa

3：B→aAc　　4：B→a　　5：B→aAb

构造 LR(0)项目集规范族：

状态	项目集	转换函数
0	S→ • A A→ • Ab A→ • bBa	GO[0, A]=1 GO[0, A]=1 GO[0, b]=2
1	S→A • A→A • b	ACCEPT GO[1, b]=3
2	A→b • Ba B→ • aAc B→ • a B→ • aAb	GO[2, B]=4 GO[2, a]=5 GO[2, a]=5 GO[2, a]=5
3	A→Ab •	R1
4	A→bB • a	GO[4, a]=6

5	B→a • Ac B→a • B→a • Ab A→ • Ab A→ • bBa	GO[5, A]=7 R4 GO[5, A]=7 GO[5, A]=7 GO[5, b]=2
6	A→bBa •	R2
7	B→aA • c B→aA • b A→A • b	GO[7, c]=8 GO[7, b]=9 GO[7, b]=9
8	B→aAc •	R3
9	B→aAb • A→Ab •	R5 R1

状态 5 存在“归约－移进”冲突，状态 9 存在“归约－归约”冲突，因此该文法不是 LR(0)文法。

状态 5:

FOLLOW(B)＝ {a}，因此，FOLLOW(B)∩ {b} ＝Φ

状态 9:

FOLLOW(B)＝ {a}，FOLLOW(A)＝ {#，b，c}，因此 FOLLOW(B)∩FOLLOW(A)＝Φ

状态 5 和状态 9 的冲突均可用 SLR(1)方法解决，构造 SLR(1)分析表如下：

状态	ACTION				GOTO	
	a	b	c	#	A	B
0		S2			1	
1		S3		ACCEPT		
2	S5					4
3		R1	R1	R1		
4	S6					
5	R4	S2			7	
6		R2	R2	R2		
7		S9	S8			
8	R3					
9	R5	R1	R1	R1		

该 SLR(1)分析表无重定义，因此该文法是 SLR(1)文法，不是 LR(0)文法。

p135-8.证明下面的文法是LL(1)的,但不是SLR(1)的。

S→AaAb|BbBa A→ε B→ε

解答:

(1)首先该文法无左递归存在,没有公共左因子。

其次:对于S→AaAb|BbBa FIRST(AaAb)={a} FIRST(BbBa)={b}

FIRST(AaAb)∩FIRST(BbBa)=Φ

所以该文法是LL(1)文法。

(2)证明该文法不是SLR的。

文法的LR(0)项目集规范族为:

I0={S'→.S S→.AaAb S→.BbBa A→. B→.}

I1={ S'→S. } I2={ S→A.aAb } I3={ S→B.bBa }

I4={ S→Aa.Ab A→. } I5={ S→Bb.Ba B→. } I6={ S→AaA.b }

I7={ S→BbB.a } I8={ S→AaAb. } I9={ S→BbBa. }

考察I0:FOLLOW(A)={a,b} FOLLOW(B)={a,b}

FOLLOW(A)∩FOLLOW(B)= {a,b}

产生规约-规约冲突。

所以该文法不是SLR(1)文法。

P135-9

证明下列文法是LR(1)文法，但不是LALR(1)文法。

S→Aa | bAc | Bc | bBa

A→d

B→d

- 思路：该文法产生的所有句子：*da*，*bdc*，*dc*，*bda*。

LR(1)分析器的移进-归约分析：

- 面临*d* → 移进，然后面临*a*或*c*时，分别将*d*归约成*A*或者*B*，再将*Aa*或*Bc*归约成*S*；
- 面临*b* → 移进，接着面临*d*移进，然后面临*a*或*c*时，分别将*d*归约成*A*或者*B*，再将*bAc*或*bBa*归约成*S*

因此该文法是LR(1)的

1、 构造该文法的 SLR(1)语法分析表

①构造增广文法

S' ->S

S->Aa

S->bAc

S->dc

S->bda

A->d

②构造 LR(0)项集族

I0 S'->.S S->.Aa S->.bAc S->.dc S->.bda A->.d	I1 S'->S.	I2 S->A.a	I5 S->Aa.	I8 S->dc.
	I3 S->b.Ac S->b.da A->.d	I4 S->d.c A->d.	I6 S->bA.c	I9 S->bAc.
			I7 S->bd.a A->d.	I10 S->bda.

③GOTO 函数

GOTO(I0, S)=I1 GOTO(I0, A)=I2 GOTO(I0, b)=I3 GOTO(I0, d)=I4

GOTO(I1, $)=acc GOTO(I2, a)=I5 GOTO(I3, A)=I6 GOTO(I3, d)=I7

GOTO(I4, c)=I8 GOTO(I6, c)=I9 GOTO(I7, a)=I10

④构建 SLR 语法分析表如下(FOLLOW(A)={a,c})

状态	ACTION					GOTO	
	a	b	c	d	$	S	A
0		S3		S4		1	2
1					acc		
2	S5						
3				S7			6
4	R5		**S8|R5**				
5					R1		
6			S9				
7	**S10|R5**		R5				
8					R3		
9					R2		
10					R4		

可以看到在图中存在二义性的条目，故该文法不是 SLR(1)文法

2、构造该文法的 LALR(1)语法分析表

①构造该增广文法的 LR(1)项集族如下

I0	I1	I3	I5	I7	I9
S'->.S,$ S->.Aa,$ S->.bAc,$ S->.dc,$ S->.bda,$ A->.d,a	S'->S.,$	S->b.Ac,$ S->b.da,$ A->.d,c	S->Aa.,$	S->bd.a.,$ A->d.,c	S->bAc.,$
I2	**I6**	**I4**	**I8**	**I10**	
S->A.a,$	S->bA.c.,$	S->d.c,$ A->d.,$	S->dc.,$	S->bda.,$	

②项集合并：没有可以合并的项集

③GOTO 函数

GOTO(I0, S)=I1 GOTO(I0, A)=I2 GOTO(I0, b)=I3 GOTO(I0, d)=I4

GOTO(I1,$)=acc GOTO(I2, a)=I5 GOTO(I3, A)=I6 GOTO(I3, d)=I7

GOTO(I4, c)=I8 GOTO(I6, c)=I9 GOTO(I7, a)=I10

④构造 LALR(1)分析表如下

状态	ACTION					GOTO	
	a	b	c	d	$	S	A
0		S3		S4		1	2
1					acc		
2	S5						
3				S7			6
4	R5		**S8**		**R5**		
5					R1		
6			S9				
7	**S10**		R5				
8					R3		
9					R2		
10					R4		

可见该分析表中不存在二义性的条目，故该文法是 LALR(1)文法