2008

一 、选择题

1.D 2.B 3.D 4.A 5.C 6.C 7.D 8.D 9.C 10.B

二、简答题：

1. 何谓二义性文法？试举一例说明。

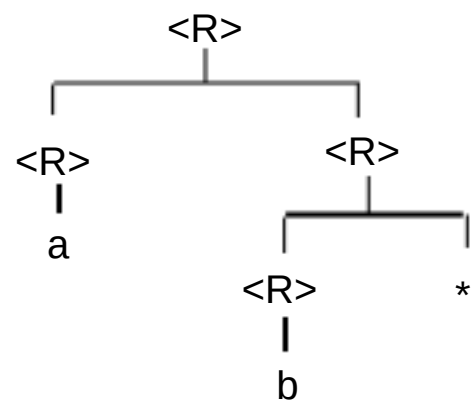
答：若文法 G 的一个句子对应有两棵或两棵以上不同的推导树，则称该句子是二义性的。

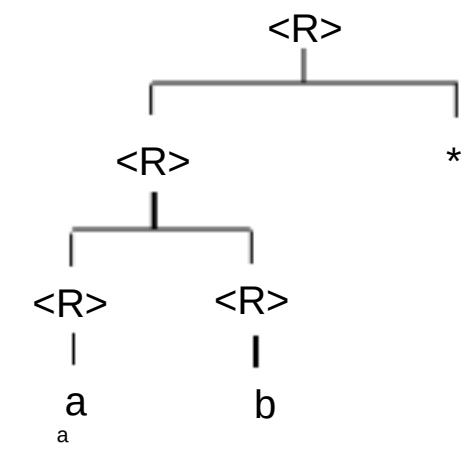
产生二义性句子的文法称为二义性文法，否则该文法是无二义性的。

例子：给定文法 G[<R>] ：

<R> →<R>*|<R><R>|a|b

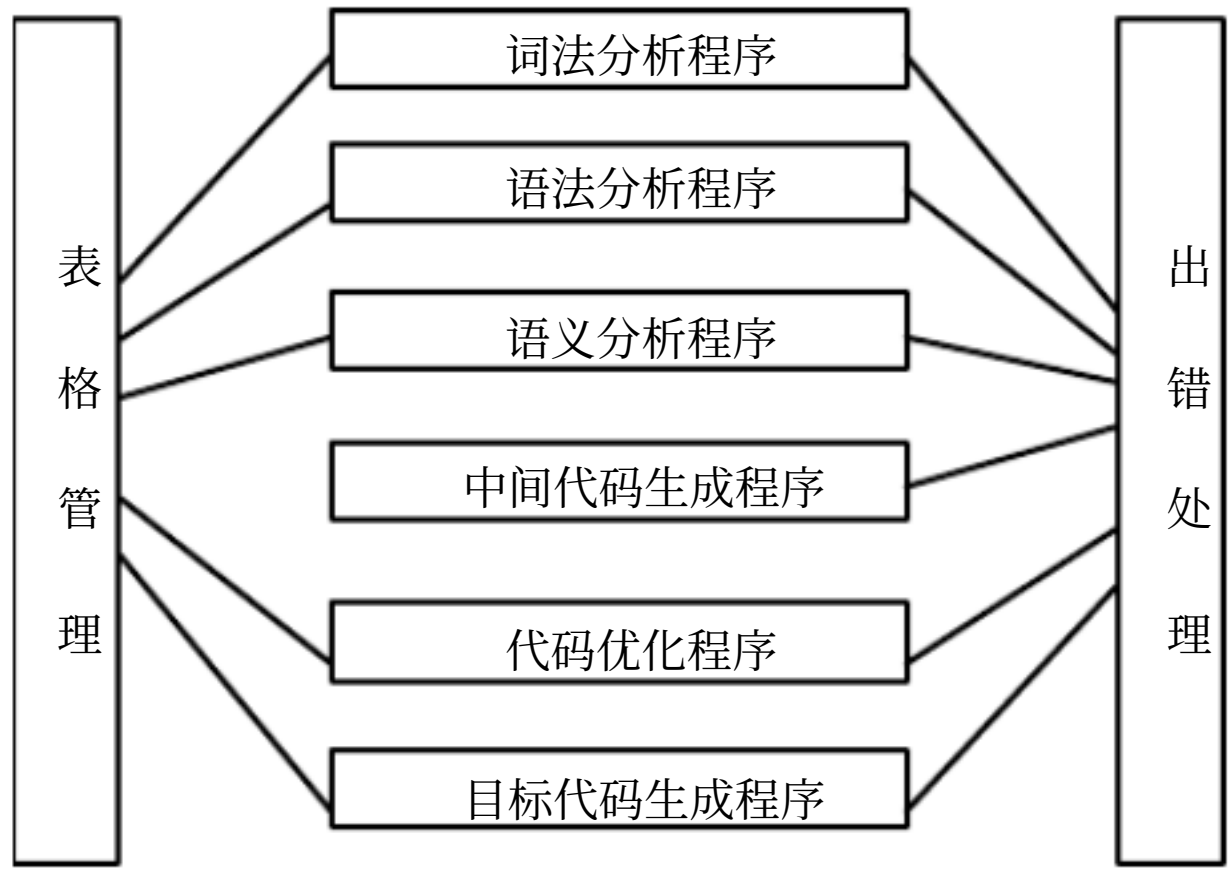
考察句子 ab*，它有两棵不同的推导树，如下所示：





2. 画出编译程序的总框图，并描述各部分的功能。

答：总体框架图如下：



词法分析器，又称扫描器，输入源程序，进行词法分析，输出单词符号。

语法分析器，简称分析器，对单词符号串进行语法分析（根据语法规则进行推导或归约） ，识别出各类语法单位，最终判断输入串是否构成语法上正确的“ 程序 ” 。

语义分析和中间代码产生器， 按照语义规则对于法分析器归约出 （或推导出） 的语法单位进行语义分析并把它们翻译成一定形式的中加代码。

优化器，对中间代码进行优化处理

目标代码生成器，把中间代码翻译成目标程序。

表格管理： 编译程序在工作过程中需要保持一系列的表格， 以登记源程序的各类信息和编译各结点的进展状况。

一个编译程序不仅应能对书写正确的程序进行翻译， 而且应能对出现在源程序中的错误进行

恢复。 如果源程序有错误，编译程序应设法发现错误，把有关错误信息报告给用户，这就是由错误处理程序完成的。

3. 简述为什么自顶向下的语法分析技术不能处理具有左递归的文法。

答：在自顶向下的语法分析技术中，要解决的问题是根据当前输入符号判断将识别符号以及非终结符号替换成哪条规则的右部，若文法具有左递归，则在分析过程中，无法判断替换的规则，造成无穷递归求解过程。

4. 对于同一个文法，LALR(1) 和 SLR(1) 的分析表的状态个数相同，为什么前者的分析能力要比后者强？

答：主要是因为两者对向前看一的集合的取法不同造成的。SLR(1) 是简单向前看的 LR 分析技术，它以 LR(0) 项为基础，仅在遇到冲突时向前看一个符号，以通过这种方法消除冲突。

LALR(1) 是向前看 LR 分析技术，它以 LR(1) 分析表为基础，对 LR(1) 的状态集合进行简化，合并其中的同心项集合，比简单向前看一符号的集合要小，更精确，因此发生冲突的可能性更小。所以 LALR(1) 要比 SLR(1) 分析能力强。

三、推导题

1. 设文法 G(S):

S→Sb　S→Ab　S→b　A→Aa　A→a

(1) 消除左递归和回溯；　(4 分)

(2) 构造相应的 FIRST 和 FOLLOW 集合；(3 分)

(3) 构造预测分析表　(3 分)

答：

（1）消除左递归（ 2 分）和回溯（ 2 分）：

S →AbS' | bS'

S'→ bS' |　ε

A →aA'

A'→aA ’ ｜ ε

（2）构造 FIRST 集合（ 1 分）构造 FOLLOW 集合（ 2 分）

FIRST(S)={a,b} FOLLOW(S)={#}

FIRST(S')={b,　ε } FOLLOW(S')={#}

FIRST(A)={a} FOLLOW(A)={b}

FIRST(A')={a,　ε) FOLLOW(A')={b}

（3）构造预测分析表（ 3 分）

-	a	b	#
S	S→AbS'	S→bS'	-
S'	-	S'→bS'	S'→ ε
A	A→aA ’	-	-
A'	A'→aA ’	A'→ ε	

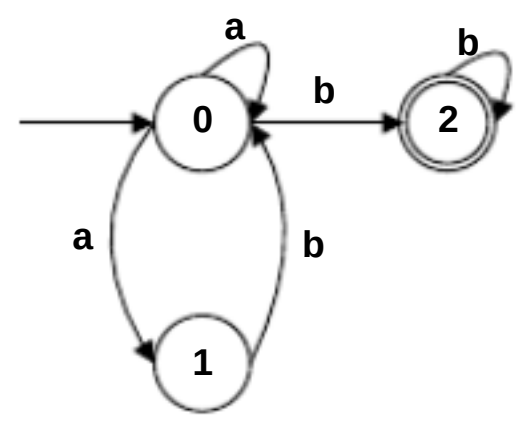
2. 构造一个 NFA，

(1) 接受字母表 {a,b} 上的正规式 (ab|a)*bb* 描述的集合。(4 分)

(2) 将(1)中的 NFA 转换为等价的 DFA　(3 分)

(3)　将(2)中的 DFA 转换为最小状态　DFA （写出步骤）　(3 分)

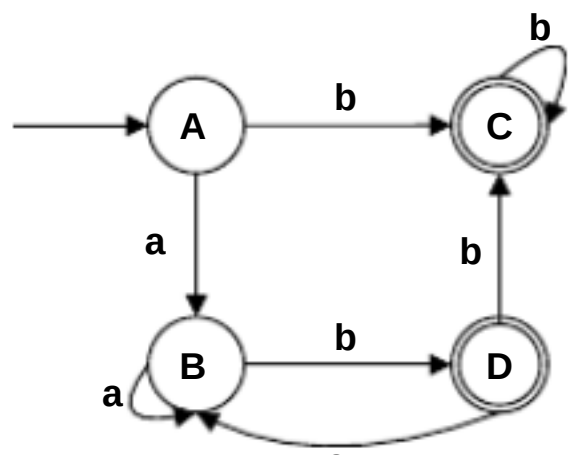
答：构造的　NFA 如下：



确定化过程：

状态集合	a	b
{0}　A	{0,1}　B	{2}　C
{0,1}　B	{0,1}　B	{0,2}　D
{2}　C		{2}　C
{0,2}　D	{0,1}　B	{2}　C

确定的　DFA 如下：



上面的　DFA 已经是最小化的。

3. 为文法　G[S]

S→(L) | a

L→L,S | S

(a)写出一个语法制导定义，计算括号的对数（　　5 分）

(b)写出一个语法制导定义，计算括号嵌套的最大深度（　　5 分）

答：(a)

产生式	语义动作
S→(L)	S.num = L.num + 1
S→a	S.num = 0
$L \rightarrow L_1,S$	L.num = L_1.num + S.num
L→S	L.num = S.num

(b)

产生式	语义动作
S→(L)	S.max = L.max + 1
S→a	S.max = 0
$L \rightarrow L_1,S$	L.max = if L_1.max > S.max then L_1.max else S.max
L→S	L.max = S.max

4. 对于文法　G[S]

S→A

A→ AB

A→ ε

B→ aB

B→ b

(1) 构造 LR(1) 分析表；　(15 分)

(2) 给出用 LR(1) 分析表对输入符号串　abab$的分析过程。　(5 分)

答：

状态	项目集	经过的符号	到达的状态
I0	S’ →, # S S→ · A,# A→ ·AB, a/b/# A→ ·, a/b/#	S A	I1 I2
I1	S’ →, S·		
I2	S→ A ·, # S→ A ·B, a/b/# B→ aB, a/b# B→ b, a/b/#	B a b	I3 I4 I5
I3	S→ AB ·, a/b/#		
I4	B→ a · B, a/b/# B→ · aB, a/b/# B→ · b, a/b/#	B a b	I6 I4 I5
I5	B→ b ·, a/b/#		
I6	B→ aB · a/b/#		

相应的 LR(1) 分析表为：

STATE	ACTION			GOTO		
	a	b	#	S	A	B
0	R3	R3	R3	1	2	
1			acc			
2	S4	S5	R1			3
3	R2	R2	R2			
4	S4	S5				6
5	R5	R5	R5			
6	R4	R4	R4			

用 LR(1) 分析表对输入符号串　abab 的分析过程：

步骤	状态	栈中符号	余留符号	分析动作	下一状态
1	0	#	abab#	R3	2
2	02	#A	abab#	S4	
3	024	#Aa	bab#	S5	
4	0245	#Aab	ab#	R5	6
5	0246	#AaB	ab#	R4	3

6	023	#AB	ab#	R2	2
7	02	#A	ab#	S4	
8	024	#Aa	b#	S5	
9	0245	#Aab	#	R5	6
10	0246	#AaB	#	R4	3
11	023	#AB	#	R2	2
12	02	#A	#	R1	1
13	01	#S	#	acc	

四、计算题

1. 设有以下程序段

```
void Q(int x)
{
        int i=1;
        x = x+2;
        i = 2;
        x = x+2;
}
void main()
{
        int i;
        int B[3];
B[1]=1;
        B[2]=2;
        i = 1;
    Q(B[i]);
}
```

试问：若参数传递方式分别采取 (1)传值调用， (2)引用调用， (3)复制 -恢复调用， (4)传名调用时，程序执行后输出 B[1] 和 B[2] 的值分别是什么？请简要写出计算过程。 (5 分)

答：

(1) 采用传值调用时， 将实在参数的值传递给形式参数， 而后在函数调用过程中， 操作的是形式参数，形式参数的值发生改变，而且这些改变不能重新传递给实在参数，
所以得到的结果是 B[1]=1 ；B[2]=2

(2) 采用应用调用， 将实在参数的地址传递给形式参数， 此时对形式参数的操作就相当于对其指向的地址单元进行操作，其操作影响了实在参数，所以得到的结果是
B[1]=5 ；B[2]=2

(3) 采用复制 -恢复调用，首先将实在参数的值传递给形式参数，此时， x=1， y=2，进行函数调用后，得到， x=5， y=2，调用返回时，将形式参数的值传递到相应的实在参数的地址中， 即 x 的值传递到 B[1] 的地址中，所以得到的结果是 B[1]=5 ;B[2]=2

(4) 采用传名调用，将 B[i] 当成整个的一个整体，替换函数调用中的 x，得到：

```
    i=1;
B[i] = B[i]+2;
i = 2;
B[i] = B[i]+2;
```

计算得到，　B[1]=3 ，　B[2]=4

2.给出如下程序段的三地址代码。　　（5 分）

```
z := 3;
while j< 10 do
        begin
            j := x +1;
            x := x+1 ;
            m: = x+1;
            if x <10        then    y:= A[i] +m
                            else    y:= A[i] -m
            n := z + 10;
        end
```

答：

```
z:=3
Label Ltest
t1:= j<10
fjump t1 Lend1
j := x +1
x := x+1
m: = x+1
t2:= x<10
fjump t2 Lfalse
y:= A[i] +m
jump Lend2
Label Lfalse
y:= A[i]-m
Lable Lend2
n := z + 10
jump Ltest
Label Lend1
```