● 文法和语言

1、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}

2、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}，或者：允许 0 开头的非负整数。

3、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

4、已知文法 G[Z]：

Z→aZb|ab

写出 L(G[Z])的全部元素。

答案：L(G[Z])={$a^n b^n$|n>=1}

5、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

6、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分析树。

（1） i+(i+i)

（2） i+i*i

答案:

(1) <表达式>

=><表达式>＋<项>

=><表达式>＋<因子>

=><表达式>＋（<表达式>）

=><表达式>＋（<表达式>＋<项>）

=><表达式>＋（<表达式>＋<因子>）

=><表达式>＋（<表达式>＋i）

=><表达式>＋（<项>＋i）

=><表达式>＋（<因子>＋i）

=><表达式>＋（i＋i）

=><项>＋（i＋i）

=><因子>＋（i＋i）

=>i＋（i＋i）

(2) <表达式>

=><表达式>＋<项>

=><表达式>＋<项>*<因子>

=><表达式>＋<项>*i

=><表达式>＋<因子>*i

=><表达式>＋i*i

=><项>＋i*i

=><因子>＋i*i

=>i＋i*i

7、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。

8、文法 G[S]为:

S →Ac|aB

A→ab

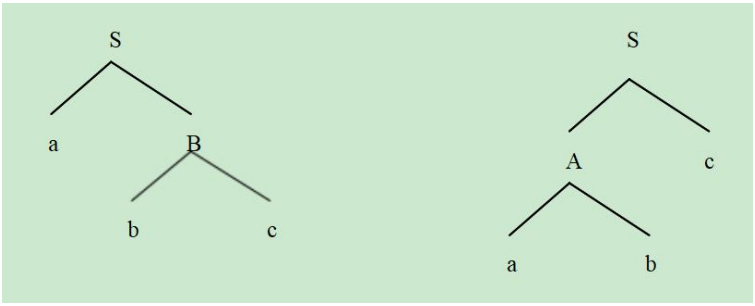
B →bc

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

即存在两个不同的最右推导。所以，该文法是二义的。 或者：对输入字符串 abc，能构造两棵不同的分析树，所以它是二义的。



9、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*，并为该串构造分析树。

(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*=>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。

10、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分析树。

11、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：

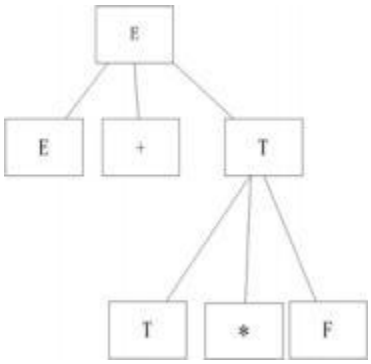
此句型对应分析树如右，故为此文法一个句型。

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

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

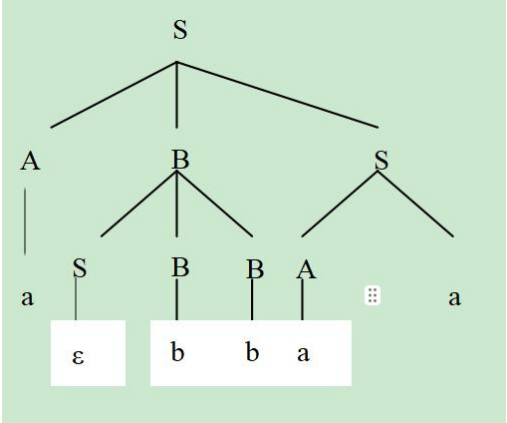


12、一个上下文无关文法生成句子 abbaa 的分析树如下：

(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

(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

13、已知文法 G[A]，写出它定义的语言描述

G[A]: A → 0B|1C

B → 1|1A|0BB

C → 0|0A|1CC

答案:

G[A]定义的语言由 0、1 符号串组成，串中 0 和 1 的个数相同.

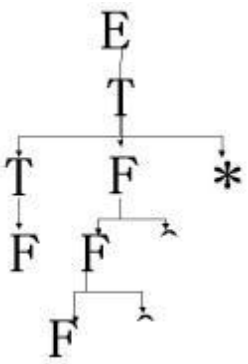
14、已知文法G[E]:

E→ET+|T

T→TF* | F

F→F^ | a

试证：FF^^*是文法的句型，指出该句型的短语、直接短语和句柄.



答案：

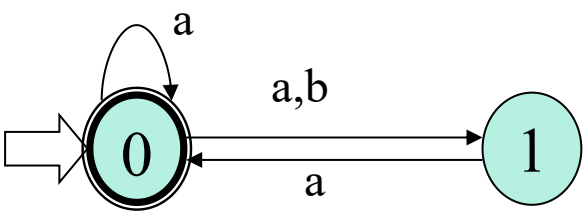
该句型对应的分析树如右：
该句型相对于 E 的短语有 FF^^*
相对于 T 的短语有 FF^^*、F
相对于 F 的短语有 F^、F^^
简单短语有 F、F^
句柄为 F

15、设有文法 G[S]：S∷=S*S|S+S|(S)|a，该文法是二义性文法吗？

答案：根据所给文法推导出句子 a+a*a，画出了两棵不同的分析树，所以该文法是二义性文法。

- 词法分析

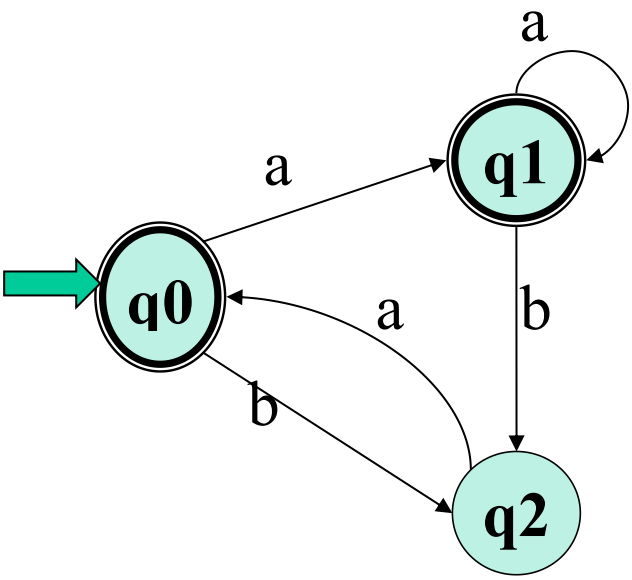
1、将下图的 NFA M 确定化。



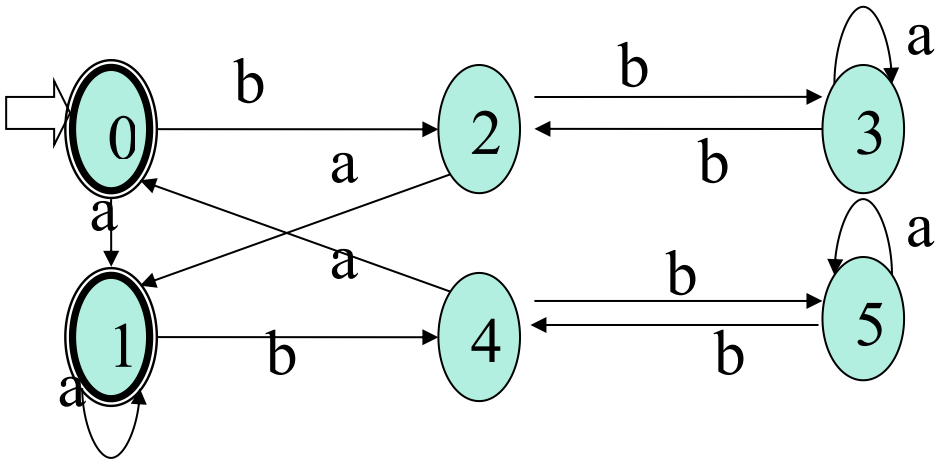
解：

	a	**b**
q0={0}	**{0,1}**	**{1}**
q1={0,1}	**{0,1}**	**{1}**
q2={1}	**{0}**	**Φ**

DFA:



2、将下图的 DFA 化简。



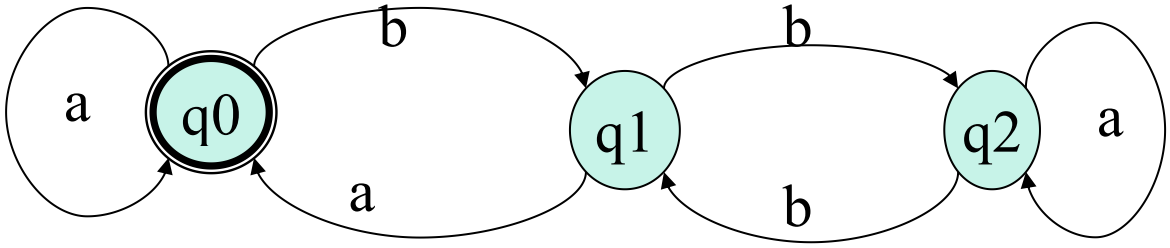
解：

划分	a	b
{0,1}	{1}	{2,4}
{2,3,4,5}	{1,3,0,5}	{3,5,2,4}

划分	a	b
{0,1}	{1}	{2,4}
{2,4}	{0,1}	{3,5}
{3,5}	{3,5}	{2,4}

q0={0,1}　　q1={2,4}　　q2={3,5}

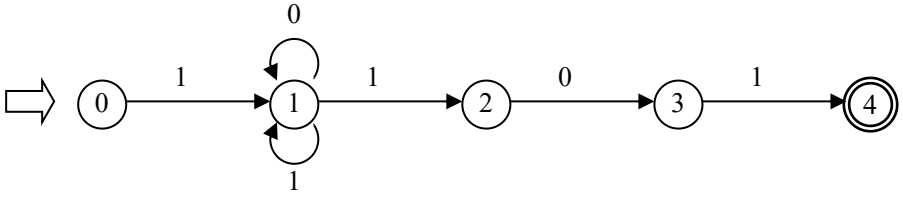
化简后的 DFA：



3、构造以下正规表达式的有限自动机。

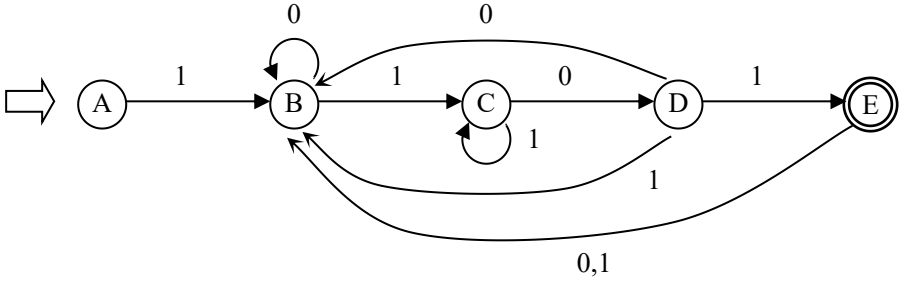
解：

(1) 1(0|1)* 101 对应的 NFA 为：

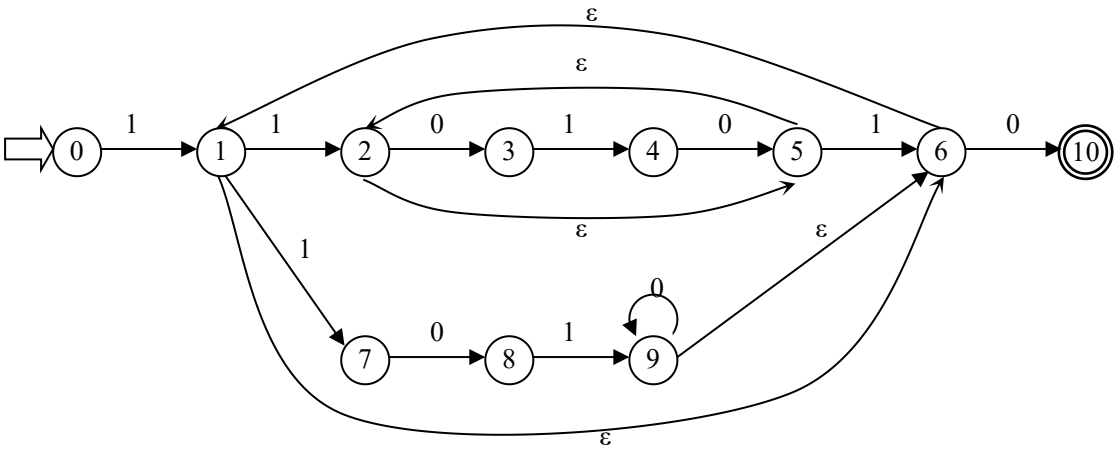


下表由子集法将 NFA 转换为 DFA：

I	I_0 = ε-closure(MoveTo(I,0))	I_1 = ε-closure(MoveTo(I,1))
A[0]		B[1]
B[1]	B[1]	C[1,2]
C[1,2]	D[1,3]	C[1,2]
D[1,3]	B[1]	E[1,4]
E[1,4]	B[1]	B[1]



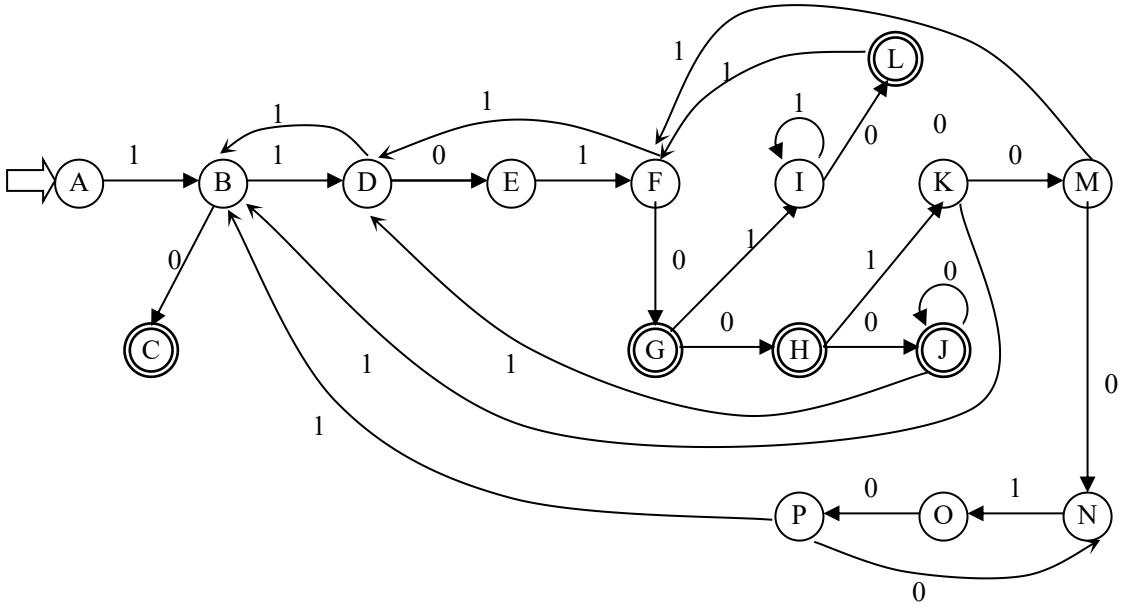
(2)1(1010*｜ 1(010)*1)*0 对应的 NFA 为：



下表由子集法将 NFA 转换为 DFA：

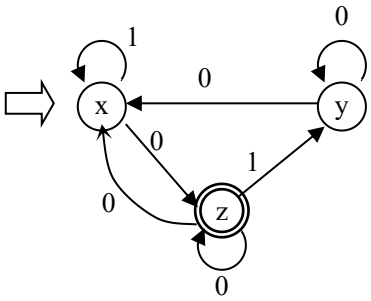
I	I_0 = ε-closure(MoveTo(I,0))	I_1 = ε-closure(MoveTo(I,1))
A[0]		B[1,6]
B[1,6]	C[10]	D[2,5,7]
C[10]		
D[2,5,7]	E[3,8]	B[1,6]
E[3,8]		F[1,4,6,9]
F[1,4,6,9]	G[1,2,5,6,9,10]	D[2,5,7]
G[1,2,5,6,9,10]	H[1,3,6,9,10]	I[1,2,5,6,7]
H[1,3,6,9,10]	J[1,6,9,10]	K[2,4,5,7]
I[1,2,5,6,7]	L[3,8,10]	I[1,2,5,6,7]
J[1,6,9,10]	J[1,6,9,10]	D[2,5,7]
K[2,4,5,7]	M[2,3,5,8]	B[1,6]
L[3,8,10]		F[1,4,6,9]

M[2,3,5,8]	N[3]	F[1,4,6,9]
N[3]		O[4]
O[4]	P[2,5]	
P[2,5]	N[3]	B[1,6]



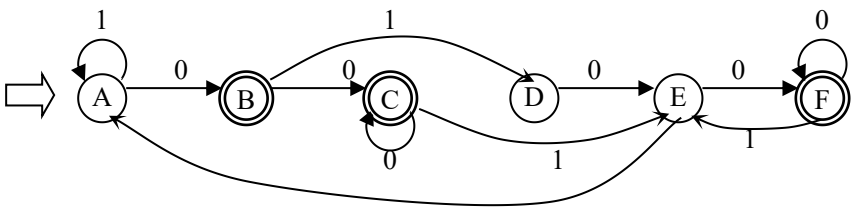
4、已知 NFA=（{x,y,z},{0,1},M,{x},{z}）其中：M(x,0)={z}, M(y,0)={x,y}, M(z,0)={x,z}, M(x,1)={x}, M(y,1)= Φ , M(z,1)={y},构造相应的 DFA。

解：根据题意有 NFA 图如下



下表由子集法将 NFA 转换为 DFA：

I	I_0 = ε-closure(MoveTo(I,0))	I_1 = ε-closure(MoveTo(I,1))
A[x]	B[z]	A[x]
B[z]	C[x,z]	D[y]
C[x,z]	C[x,z]	E[x,y]
D[y]	E[x,y]	
E[x,y]	F[x,y,z]	A[x]
F[x,y,z]	F[x,y,z]	E[x,y]



下面将该 DFA 最小化：

(1) 首先将它的状态集分成两个子集：P_1={A,D,E},P_2={B,C,F}

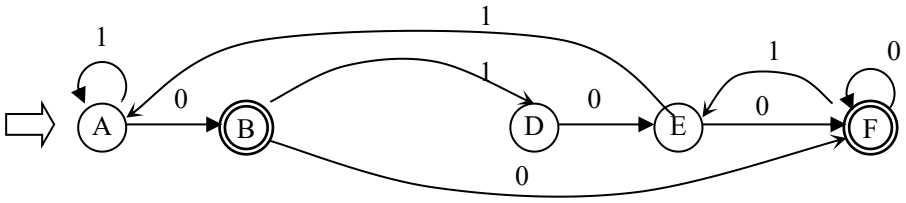
(2) 区分 P_2: 由于 F(F,1)=F(C,1)=E,F(F,0)=F 并且 F(C,0)=C, 所以 F，C 等价。由于 F(B,0)=F(C,0)=C, F(B,1)=D,F(C,1)=E,而 D，E 不等价（见下步），从而 B 与 C，F 可以区分。有 P_{21}={C,F},P_{22}={B}。

(3) 区分 P1:由于 A，E 输入 0 到终态，而 D 输入 0 不到终态，所以 D 与 A，E 可以区分，有 P_{11}={A,E},P_{12}={D}。

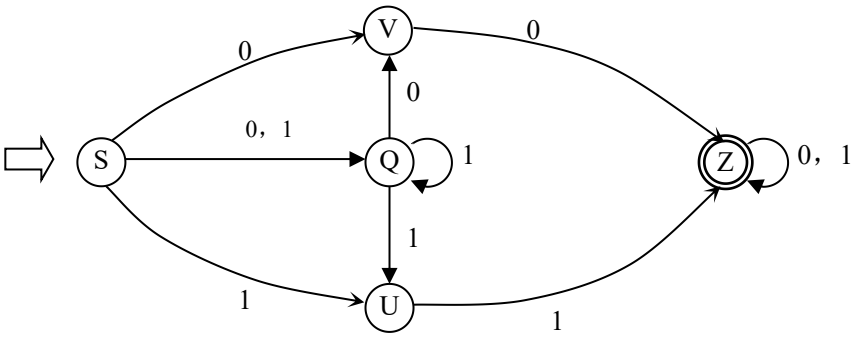
(4) 由于 F(A,0)=B,F(E,0)=F,而 B，F 不等价，所以 A，E 可以区分。

(5) 综上所述，DFA 可以区分为 P={{A}，{B}，{D}，{E}，{C，F}}。所以最小化的 DFA 如下：

(6)

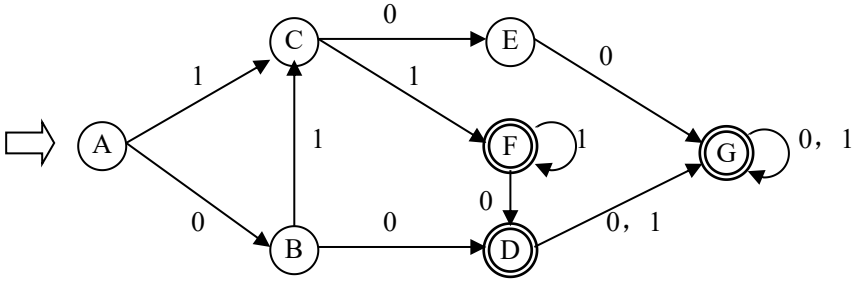


5、确定化：

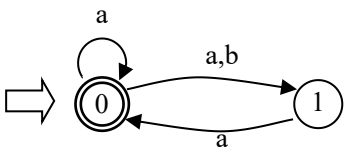


解：下表由子集法将 NFA 转换为 DFA：

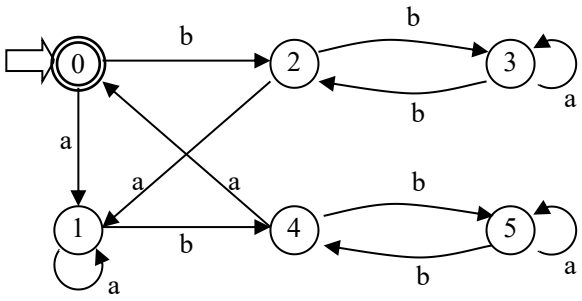
I	I_0 = ε-closure(MoveTo(I,0))	I_1 = ε-closure(MoveTo(I,1))
A[S]	B[Q,V]	C[Q,U]
B[Q,V]	D[V,Z]	C[Q,U]
C[Q,U]	E[V]	F[Q,U,Z]
D[V,Z]	G[Z]	G[Z]
E[V]	G[Z]	
F[Q,U,Z]	D[V,Z]	F[Q,U,Z]
G[Z]	G[Z]	G[Z]



6、把图(a)和(b)分别确定化和最小化：



(a)



(b)

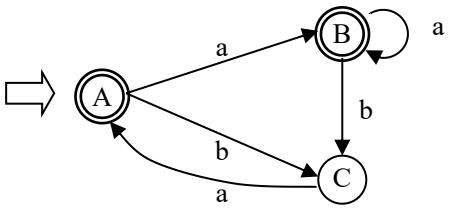
解：

(a)：

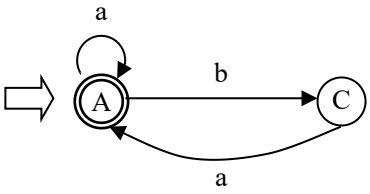
下表由子集法将 NFA 转换为 DFA：

I	I_a = ε-closure(MoveTo(I,a))	I_b = ε-closure(MoveTo(I,b))
A[0]	B[0,1]	C[1]
B[0,1]	B[0,1]	C[1]
C[1]	A[0]	

可得图（a1），由于 F(A,b)=F(B,b)=C,并且 F(A,a)=F(B,a)=B,所以 A,B 等价，可将 DFA 最小化，即：删除 B，将原来引向 B 的引线引向与其等价的状态 A，有图(a2)。（DFA 的最小化，也可看作将上表中的 B 全部替换为 A，然后删除 B 所在的行。）



（a1）确定化的 DFA



（a2）最小化的 DFA

（b）该图已经是 DFA。下面将该 DFA 最小化：

(1) 首先将它的状态集分成两个子集：P_1={0},P_2={1,2,3,4,5}

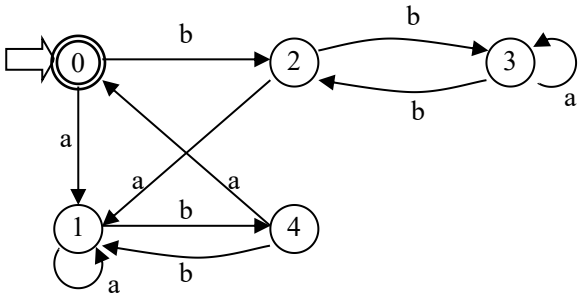
(2) 区分 P_2：由于 F(4,a)=0 属于终态集，而其他状态输入 a 后都是非终态集，所以区分 P_2 如下：P_{21}={4},P_{22}={1,2,3,5}。

(3) 区分 P_{22}:由于 F(1,b)=F(5,b)=4 属于 P_{21},而 F(2,b)与 F(3,b)不等于 4,即不属于 P_{21}，所以区分 P_{22} 如下：P_{221}={1,5},P_{222}={2,3}。

(4) 区分 P_{221}:由于 F(1,b)=F(5,b)=4,即 F(1,a)=1,F(5,a)=5,所以 1,5 等价。

(5) 区分 P_{222}:由于 F(2,a)=1 属于 P_{221}，而 F(3,a)=3 属于 P_{222},所以 2，3 可区分。P_{222} 区分为 P_{2221}{2}，P_{2222}{3}。

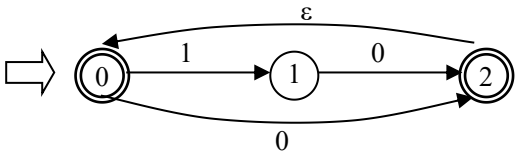
(6) 结论：该 DFA 的状态集可分为：P={ {0},{1,5},{2},{3},{4} },其中 1,5 等价。删去状态 5，将原来引向 5 的引线引向与其等价的状态 1，有图(b1)。



(b1)最小化的 DFA

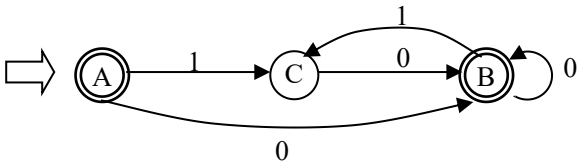
7、构造一个 DFA，它接收 Σ={0，1}上所有满足如下条件的字符串：每个 1 都有 0 直接跟在右边。

解：根据题意，DFA 所对应的正规式应为：(0|(10))*。所以，接收该串的 NFA 如下：

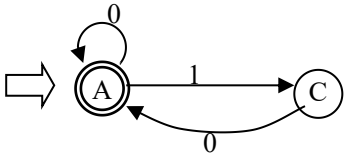


下表由子集法将 NFA 转换为 DFA：

I	I_0 = ε-closure(MoveTo(I,0))	I_1 = ε-closure(MoveTo(I,1))
A[0]	B[0,2]	C[1]
B[0,2]	B[0,2]	C[1]
C[1]	B[0,2]	



显然，A，B 等价，所以将上述 DFA 最小化后有：



- 语法分析

1、有文法 G[A]：A→aABe|ε，B→Bb|b

（1）求每个非终结符号的 FOLLOW 集。

（2）该文法是 LL(1)文法吗？

（3）构造 LL(1)分析表。

解：

（1） FOLLOW(A)=First(B) ∪ {$}={b,$}
FOLLOW(B)={e,b}

（2） 该文法中的规则 B→Bb|b 为左递归，因此该文法不是 LL(1)文法

（3） 先消除文法的左递归（转成右递归），文法变为：A→aABe|ε，B→bB’，B’→bB’|ε，该文法的 LL(1)分析表为：

LL(1)分析表：

	a	e	b	$
A	A→aABe		A→ε	A→ε
B			B→bB’	
B’		B’ →ε	B’→bB’	

2、若有文法 A→(A)A| ε

（1）为非终结符 A 构造 FIRST 集合和 FOLLOW 集合。

（2）说明该文法是 LL(1)的文法。

解：

（1）FIRST(A)＝ {(, ε }
FOLLOW(A)＝ {)，$}

（2）

该文法不含左递归；

FIRST（(A)A）={(}，FIRST（ε）={ ε }，FIRST（(A)A）∩FIRST（ε）=Φ，

且 FOLLOW(A)＝ {)，$}，FIRST（(A)A）∩ FOLLOW(A) =Φ，

因此，该文法满足 LL(1)文法的条件，是 LL(1)文法。

3、考虑下面简化了的 C 声明文法：

<声明语句>→<类型><变量表>;

<类型>→int | float | char

<变量表>→ID,<变量表>|ID

（1）在该文法中提取左因子。

（2）为所得的文法的非终结符构造 FIRST 集合和 FOLLOW 集合。

（3）说明所得的文法是 LL(1)文法。

（4）为所得的文法构造 LL(1)分析表。

（5）假设有输入串为“char x，y，z;”，写出相对应的 LL(1)分析过程。

解：

（1） 规则<变量表>→ID,<变量表>|ID 提取公因子如下：<变量表>→ID（,<变量表>| ε ）

增加新的非终结符<变量表 1>，规则变为：

<变量表>→ID<变量表 1>

<变量表 1>→，<变量表>| ε

C 声明文法改变为：

<声明语句>→<类型><变量表>;

<类型>→int|float|char

<变量表>→ID<变量表 1>

<变量表 1>→，<变量表>| ε

（2） FIRST(<声明语句>)＝FIRST(<类型>)＝｛int，float，char｝

FIRST(<变量表>)＝｛ID｝

FIRST(<变量表 1>)＝｛,， ε｝

FOLLOW(<声明语句>)＝｛$｝

FOLLOW(<类型>)＝FIRST(<变量表>)＝｛ID｝

FOLLOW(<变量表>)＝FOLLOW(<变量表 1>)＝｛;｝

（3） 所得文法无左递归，且

FIRST（int）∩FIRST（float）∩FIRST（char）＝Φ

FIRST（，<变量表>）∩FIRST（ ε ）＝Φ

FIRST（，<变量表>）∩FOLLOW(<变量表 1>)＝Φ

因此，所得文法为 LL(1)文法。

所得的文法构造 LL(1)分析表如下所示：

	;	int	float	char	ID	,	$
<声明语句>		<声明语句>→<类型><变量表>;	<声明语句>→<类型><变量表>;	<声明语句>→<类型><变量表>;			
<类型>		<类型>→int	<类型>→float	<类型>→char			
<变量表>					<变量表>→ID<变量表 1>		
<变量表 1>	<变量表 1>→ε					<变量表 1>→,<变量表>	

（4） 输入串“char x，y，z;”相对应的 LL(1)分析过程如下：

步骤	分析栈	余留输入串	分析表元素	所用产生式
1	$<声明语句>	char x，y，z;$	POP，PUSH(；<变量表><类型>)	<声明语句>→<类型><变量表>;
2	$；<变量表><类型>	char x，y，z;$	POP，PUSH(char)	<类型>→char
3	$；<变量表>char	char x，y，z;$	POP, NEXTSYM	
4	$;<变量表>	x，y，z;$	POP，PUSH(<变量表 1>ID)	<变量表>→ID<变量表 1>
5	$;<变量表 1>x	x，y，z;$	POP, NEXTSYM	
6	$;<变量表 1>	，y，z;$	POP，PUSH(<变量表>，)	<变量表 1>→，<变量表>
7	$;<变量表>,	，y，z;$	POP,	

			NEXTSYM	
8	$；<变量表>	y，z；$	POP， PUSH(<变量表 1>ID)	<变量表>→ID<变量表 1>
9	$；<变量表 1>y	y，z；$	POP, NEXTSYM	
10	$；<变量表 1>	，z；$	POP， PUSH(<变量表>，)	<变量表 1>→，<变量表>
11	$；<变量表>，	，z；$	POP, NEXTSYM	
12	$；<变量表>	z；$	POP， PUSH(<变量表 1>ID)	<变量表>→ID<变量表 1>
13	$；<变量表 1>z	z；$	POP, NEXTSYM	
14	$；<变量表 1>	；$	POP	<变量表 1>→ε
15	$；	；$	POP, NEXTSYM	
16	$	$	ACCEPT	

4、考虑以下的文法：

S→S;T|T

T→a

（1）为这个文法构造规范 LR(0)项目集族。

（2）这个文法是不是 LR(0)文法？如果是，则构造 LR(0)分析表。

（3）对输入串“a;a”进行分析。

解：

（1）拓广文法 G[S’]：

0：S’→S

1：S→S;T

2：S→T

3：T→a

构造规范 LR(0)项目集族：

状态	项目集	转换函数
0	S’→·S S→·S;T S→·T T→·a	GO[0，S]＝1 GO[0，S]＝1 GO[0，T]＝2 GO[0，a]＝3
1	S’→S· S→S·;T	ACCEPT GO[1，;]＝4
2	S→T·	R2
3	T→a·	R3
4	S→S;·T T→·a	GO[4，T]＝5 GO[4，a]＝3
5	S→S;T·	R1

（2）该文法不存在“归约－归约”和“归约－移进”冲突，因此是LR(0)文法。LR(0)分析表如下：

状态	ACTION			GOTO	
	;	a	$	S	T
0		S3		1	2
1	S4		ACCEPT		
2	R2	R2	R2		
3	R3	R3	R3		
4		S3			5
5	R1	R1	R1		

（3）对输入串“a;a”进行分析如下：

步骤	状态栈	符号栈	输入符号栈	ACTION	GOTO
0	0	$	a;a$	S3	
1	03	$a	;a$	R3	2
3	02	$T	;a$	R2	1
4	01	$S	;a$	S4	
5	014	$S;	a$	S3	
6	0143	$S;a	$	R3	5
7	0145	$S;T	$	R1	1
8	01	$S	$	ACCEPT	

5、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	GO[2，B]＝4 GO[2，a]＝5 GO[2，a]＝5

	B→·aAb	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。

6、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？试对下面方法进行改写，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

解：对于一个文法若消除了左递归，提取了左公共因子后不一定为 LL(1)文法。需要进行 LL(1)判断才能决定新方法是否一定是 LL(1)文法。

(1)

由于 SELECT(A→baB)={b},SELECT(A→ ε)=FOLLOW(A)={b,$},两集合有交集,所以该文法不是 LL(1)方法。

该文法已经消除了左递归，与左公共因子，一般来说是不能再改写了。但根据本文法的具体情况有以下改写：

用产生式 A→baB 与 A→ ε 分别替换产生式 B→Abb 有：B→baBbb|bb，提取这两个新产生式的左公共因子有：

B→bB′,B′→aBbb|b,这样改写后文法 G′[A]为：

A→baB| ε

B→bB′|a

B′→aBbb|b

每个产生式的 SELECT 集合为：

SELECT(A→baB)={b}

SELECT(A→ ε)=Φ

SELECT(B→bB′)={b}

SELECT(B→a)={a}

SELECT(B′→aBbb)={a}

SELECT(B′→b)={b}

可见，相同左部产生式的 SELECT 集的交集均为空，所以文法 G′[A]是 LL(1)文法。

(2)

显然文法的第 1，2 个产生式的右部具有左公共因子 a,而产生式 B→Bb 具有左递归，因此文法可改写为：

A→aA′

A′→ABe| ε

B→dB′

B′→bB′| ε

由于 SELECT(A′→ABe)={a},SELECT(A′→ ε)=FOLLOW(A′)=FOLLOW(A)=FIRST(B) ∪ {$}={d,$},交集为空。

而 SELECT(B′→bB′={b},SELECT(B′→ ε)=FOLLOW(B′)=FOLLOW(B)={e},交集也为空。

而非终结符 A 与 B 都只有一个产生式，不存在求 SELECT 的交集问题。

所以改写后的方法为 LL(1)文法。

7、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。

解：增加一个非终结符 S′后，产生原文法的拓广文法有：

S′→A

A→aAd|aAb| ε

下面构造它的 LR(0)项目集规范族为：

状态 \ 当前符号	a	b	d	$	A
I_0: S′→•A A→•aAd A→•aAb A→•	I_2: A→a•Ad A→a•Ab A→•aAd A→•aAb A→•				I_1: S′→A•
I_1: S′→A•				acc	
I_2: A→a•Ad A→a•Ab A→•aAd	I_2				I_3: A→aA•d A→aA•b

A→•aAb A→•					
I_3: A→aA•d A→aA•b		I_4: A→aAb•	I_5: A→aAd•		
I_4: A→aAb•					
I_5: A→aAd•					

从上表可看出，状态 I_0 和 I_2 存在移进-归约冲突，该文法不是 LR(0)文法。

对于 I_0 来说有

FOLLOW(A) ∩ {a}={b, d, $} ∩ {a}=Φ

所以在 I_0 状态下面临输入符号为 a 时移进，为 b, d, $时归约，为其他时报错。

对于 I_2 来说有也有与 I_0 完全相同的结论。

这就是说，以上的移进-归约冲突是可以解决的，因此该文法是 SLR(1)文法。SLR(1)分析表为：

文法的 SLR(1)分析表

状态	ACTION				GOTO
	a	b	d	$	A
0	S_2	r_1	r_2	r_3	1
1				acc	
2	S_2	r_1	r_2	r_3	3
3		S_4	S_5		
4	r_2	r_2	r_2	r_2	
5	r_1	r_1	r_1	r_1	

对输入串 ab$给出分析过程为：

步骤	状态栈	符号栈	输入串	ACTION	GOTO
1	0	$	ab$	S_2	
2	02	$a	b$	r_3	3
3	023	$aA	b$	S_4	
4	0234	$aAb	$	r_2	1
5	01	$A	$	acc	

8、试验证如下文法 G[E] 是 LL(1)文法:

E → [F] E′

E’ → E| ε

F → aF’

F’ → aF’ | ε

其中 E, F, E’,F’ 为非终结符。

答案：

各非终结符的 FIRST 集和 FOLLOW 集如下：

FIRST（E）={[}　FOLLOW（E）={#}

FIRST（E′）={[，ε}　FOLLOW（E′）={#}

FIRST（F）={a}　FOLLOW（F）={]}

FIRST（F′）={a，ε}　FOLLOW（F′）={]}

对于 E’ →E|ε，FIRST(E)∩FIRST(ε)= ϕ

FIRST(E) ∩ FOLLOW（E’）= ϕ

对于 F’ →aF’|ε，FIRST(aF’)∩FIRST(ε)= ϕ

FIRST（aF’）∩ FOLLOW（F’）= ϕ

所以, 文法 G[E]是 LL（1）文法。

9、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：拓广文法为 G′，增加产生式 S′→S

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}

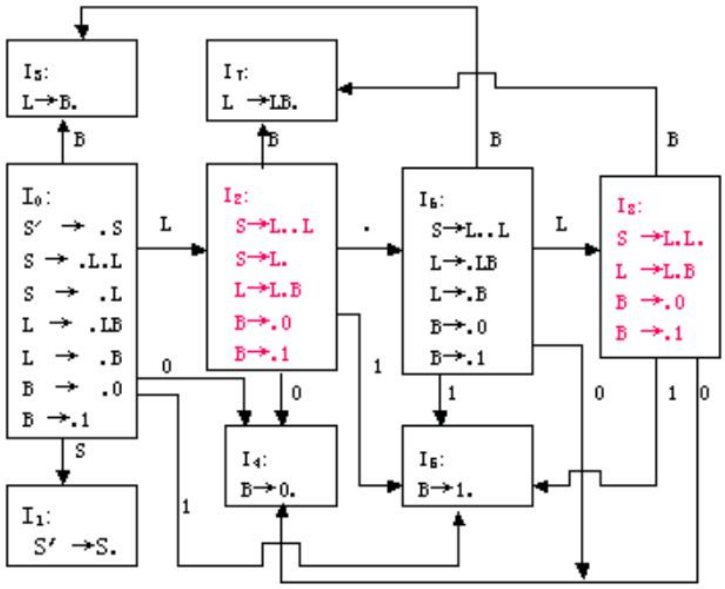
Follow(S') = {$}

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：

状态（State）	Action				Goto		
	•	0	1	#	S	L	B
0		S4	S5		1	2	3
1				acc		.	
2	S6	S4	S5	r2			7
3	r4	r4	r4	r4		.	
4	r5	r5	r5	r5		.	
5	r6	r6	r6	r6		.	
6		S4	S5			8	3
7	r3	r3	r3	r3		.	
8		S4	S5	r1			7

对输入串 101.110$的分析过程：

状态栈（state stack）	文法符号栈	剩余输入串（input left）	动作（action）
0	#	101.110#....	Shift
0 5	#1	01.110#....	Reduce by :B →1
0 3	#B	01.110#....	Reduce by :S →LB
0 2	#L	01.110#....	Shift
0 2 4	#L0	1.110#....	Reduce by :B →0
0 2 7	#LB	1.110#....	Reduce by :S →LB
0 2	#L	1.110#....	Shift
0 2 5	#L1	.110#....	Reduce by :B →1
0 2 7	#LB	.110#....	Reduce by :S →LB
0 2	#L	.110#....	Shift
0 2 6	#L.	110#....	Shift
0 2 6 5	#L.1	10#....	Reduce by :B →1
0 2 6 3	#L.B	10#....	Reduce by :S →B
0 2 6 8	#L.L	10#....	Shift
0 2 6 8 5	#L.L1	0#....	Reduce by :B →1
0 2 6 8 7	#L.LB	0#....	Reduce by :S →LB
0 2 6 8	#L.L	0#....	Shift
0 2 6 8 4	#L.L0	#....	Reduce by :B →0
0 2 6 8 7	#L.LB	#....	Reduce by :S →L.L
0 1	#S	#....	

- 语法制导翻译

1、变量的说明是由下列文法生成的：

D→ i L

L→ ,i L | :T

T→ integer | real

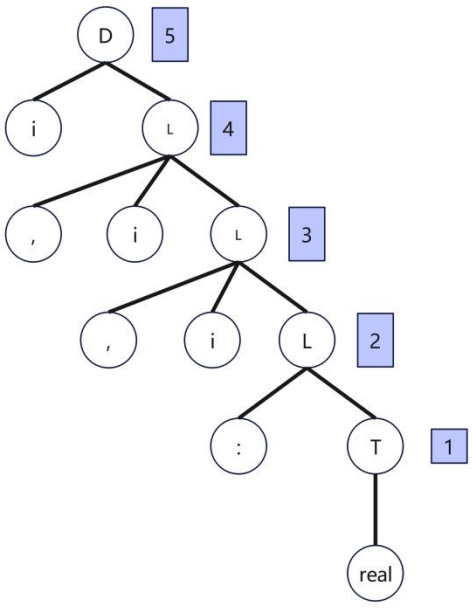
以下语法制导定义，把每一个标识符的类型添加到符号表中。

type 为综合属性，代表类型属性，
函数 addtype 实现向符号表中 i 对应项填类型信息。

语法制导定义

产生式	语义动作
D→i L	D.Type:=L.Type addtype(i.entry, D.type)
L→,i L1	L.Type:=L1.Type addtype(i.entry, L.type)
L→:T	L.type:=T.type
T→integer	T.type:=integer
T→real	T.type:=real

试翻译 1 个句子：i, i, i :real



答：S-属性定义，5 步归约完成翻译。

2、请按语法制导的定义，将后缀表达式翻译成中缀表达式。注意，不允许出现冗余括号，后缀表达式的文法如下：

E→EE+

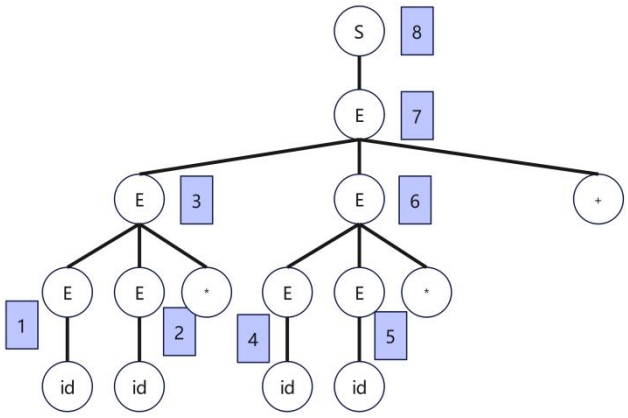
E→EE*

E→id

语法制导定义

产生式	语义规则
S→E	print E.code
$E→E_1E_2+$	$E.code=E_1.code\|\|'+'\|\|E_2.code;$ E.op='+'
$E→E_1E_2*$	IF $E_1.op='+'$ AND $E_2.op='+'$ THEN $E.code="("\|\|E_1.code\|\|')'\|\|'*'\|\|'('\|\|E_2.code\|\|')';$ ELSE IF $E_1.op='+'$THEN $E.code="("\|\|E_1.code\|\|')'\|\|'*'\|\|E_2.code;$ ELSE IF $E_2.op='+'$THEN $E.code=E_1.code\|\|'*'\|\|'('\|\|E_2.code\|\|')';$ ELSE $E.code=E_1.code\|\|'*'\|\|E_2.code\|\|;$
E —>id	E.code:=id.lexeme;

试翻译 1 个句子：id id * id id * +



答：S-属性定义，8 步归约完成翻译。翻译得到中缀表达式：id * id + id * id。

3、下面文法产生的表达式是对整型和实型常数应用算符+形成的。当两个整数相加时，结果为整数，否则为实数。

E→ TR

R→+ TR| ε

T→ num.num | num

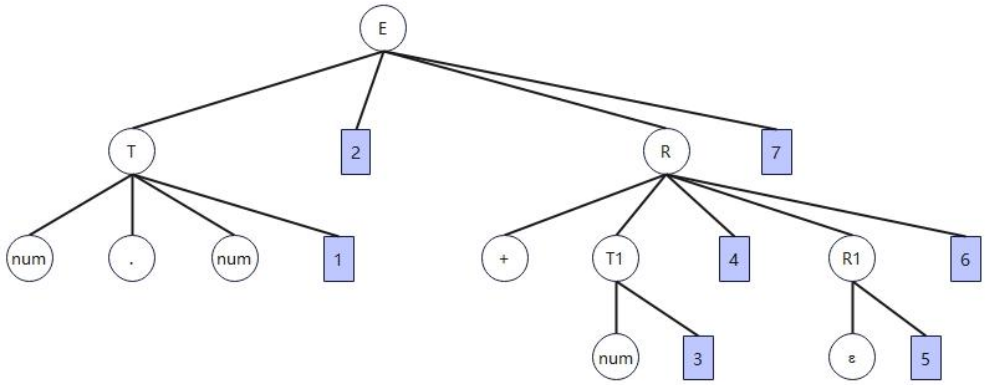
a)给定的语法制导定义确定每个子表达式的类型。

a)设 type 是综合属性，代表各非终结符的“类型”属性
设 in 是继承属性，

翻译方案

产生式	语义规则
E→T R	{R.i:=T.type} {E.Type:=R.s}
R→+ T R1	 {IF (R.i=integer) and (T.type=integer) THEN R1.i:=integer ELSE R1.i :=real} {R.s:=R1.s}
R→ ε	{R.s:=R.i}
T→num.num	T.type:=real
T→num	T.type:=integer

试翻译句子：5.5 + 5（num.num + num）



答：L-属性定义。

1、T.type = real;

2、R.i = T.type = real;

3、T1.type = integer;

4、R1.i = real;

5、R1.s = R1.i = real;

6、R.s = R1.s = real;

7、E.type = R.s = real;

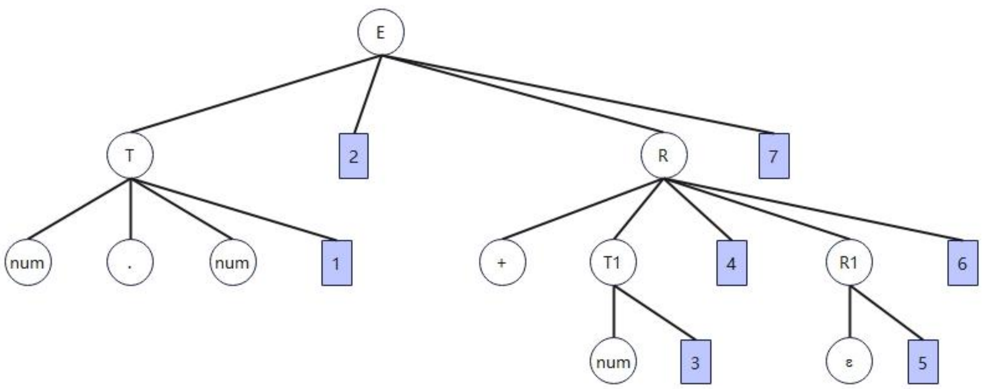
b)以下语法制导定义把表达式翻译成前缀形式，并且决定类型。用一元运算符 inttoreal 把整型值转换为相等的实型值，以使得前缀表达式中两个运算对象是同类型的。 设属性 s 和 i 用于传递属性 type，属性 t 和 j 用于传递属性 val。

产生式	语义规则
E→T R	{R.i:=T.type} {R.j:=T.val} {E.Type:=R.s} {E.val:=R.t}

R→+T
 	
R1
	{IF (R.i=integer) and (T.type=integer) THEN
	BEGIN
	R1.i:=integer
	Print('+',R.j,T.val)
	R1.j:=R.j+T.val
	END
	ELSE BEGIN
	R1.i :=real
	IF R.i=integer THEN
	Begin
	R.i:=real
	R.j:=inttoreal(R.j)
	End
	IF T.type=integer THEN
	Begin
	T.type:=real
	T.val:=inttoreal(T.val)
	End
	Print('+',R..j,T.val)
	R1.j :=R.j+T.val
	END}
	{R.s:=R1.s} {R.t:=R1.t}

R→ ε	{R.s:=R.i} {R.t:=R.j}
T→num.num	{T.type:=real} {T.val:=num.num.lexval}
T→num	{T.type:=integer} {T.val:=num.lexval}

试翻译句子：5.5 + 5（num.num + num）



答：L-属性定义。

1、T.type = real; T.val = 5.5;

2、R.i = T.type = real; R.j = T.val = 5.5;

3、T1.type = integer; T1.val = 5;

4、R1.i = real;

T1.type = real;

T1.val = 5.0;

Print (+, 5.5, 5.0)

R1.j = 5.5 + 5.0 = 10.5;

5、R1.s = R1.i = real; R1.t = R1.j = 10.5;

6、R.s = R1.s = real; R.t = R1.t = 10.5;

7、E.type = R.s = real; E.val = R.t = 10.5

- 语义分析与中间代码生成

1、 翻译算术表达式－(a+b) * (c +d) +(a +b + c)为：

(1) 四元式；

(2) 三元式；

(3) 间接三元式。

答：

先写出三地址代码为：

t1 := a + b

t2 := - t1

t3 := c + d

t4 := t2 * t3

t5 := a + b

t6 := t5 + c

t7: = t4 + t6

a)：对应的四元式为：

	op	arg1	arg2	result
(0)	+	a	b	t1
(1)	uminus	t1		t2
(2)	+	c	d	t3
(3)	*	t2	t3	t4
(4)	+	a	b	t5
(5)	+	t5	c	t6
(6)	+	t4	t6	t7

b)：对应的三元式为：

	op	arg1	arg2
(0)	+	a	b
(1)	Uminus	(0)	
(2)	+	c	d
(3)	*	(1)	(2)
(4)	+	a	b
(5)	+	(4)	c
(6)	+	(3)	(5)

c)：对应的间接三元式为：

	statement
(0)	15
(1)	16
(2)	17
(3)	18
(4)	15
(5)	19
(6)	20

	op	arg1	arg2
15	+	a	b
16	uminus	15	
17	+	c	d
18	*	16	17
19	+	15	c
20	+	18	19

2、给出下面表达式的逆波兰表示(后缀式)：

(1)a*(-b+c)

(2) if (x+y)*z=0 then s：= (a+b)*c else s：= a*b*c

答：

(1) ab-c+*

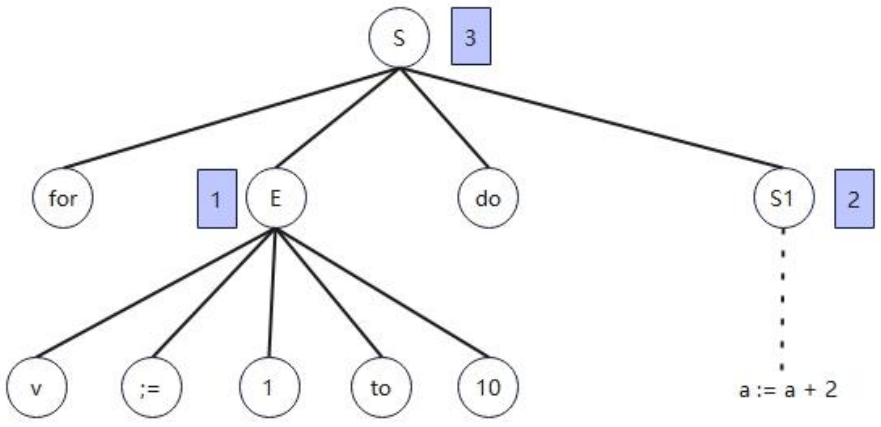
(2)xy+z*0=sab+c*:=sab*c*:=￥（注：￥表示 if-then-else 运算）

3、见 for 语句的翻译方案

产生式	动作
S → for E do S1	S.begin = newlabel S.first = newtemp S.last = newtemp S.curr = newtemp S.code =gen(S.first "=" E.init) ‖gen(S.last "=" E.final) ‖gen("if" S.first ">" S.last "goto" S.next) ‖gen(S.curr "=" S.first) ‖gen(S.begin ":") ‖gen("if " S.curr ">" S.Last "goto" S.next) ‖S1.code ‖gen (S.curr = S.curr + 1) ‖gen("goto" S.begin)
E → v:=initial to final	E.init = initial.place E.final = final.place

试翻译句子：for v := 1 to 10 do a := a + 2

答：S-属性定义，3 步归约完成翻译。（假设 v := v + 2 翻译为：t1 := a + 2; a := t1）



1、E.init = 1 ； E.final =10

2、S1.code = { t1 := a + 2; a := t1 }

3、

```
S.begin = L1;
S.first = t2;
S.last = t3;
S.curr = t4;
S.code = {
        t2 = 1;
        t3 = 10;
        If t2 > t3 goto  S.next;
        t4 = t2;
L1:     if  t4 > t3 goto S.next
        t1 := a + 2;
        a := t1
        t4 = t4  + 1;
        goto  L1
        }
```

- 中间代码优化

1、对下列基本块应用 DAG 进行优化：

B=3；

D= A+C；

E=A*C；

F=D+E；

G=B*F；

H=A+C；

I=A*C；

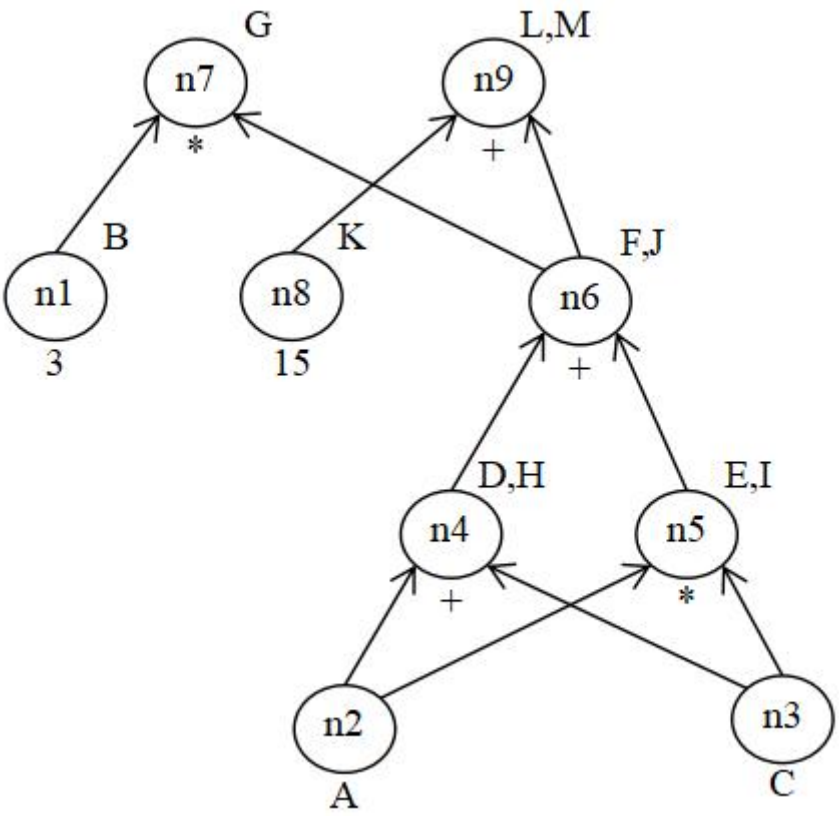
J=H+I；

K=B*5；

L=K+J；

M= L；

解：构造 DAG 如下：



按照上图的 DAG 结点顺序，优化后生成的程序如下：

B=3；

D=A+C；

H=D；

E=A*C；

I=E；

F=D+E；

J=F；

G=3*F；

K=15；

L=15+J；

M=L；

2、分别对以下两个流图：

(1) 求出流图中各结点 n 的必经结点集 D(n)。

(2) 求出流图中的回边。

(3) 求出流图中的循环。

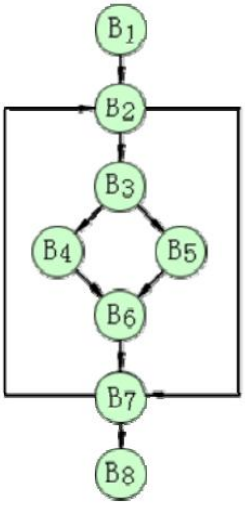


图 1

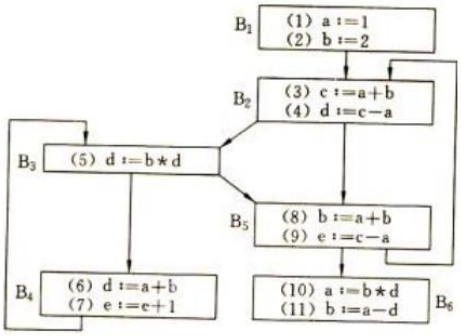


图 2

答案：

图 1、（1）流图中各结点 N 的必经结点集 D(n)：

D(1)={1}

D(2)={1,2}

D(3)={1,2,3}

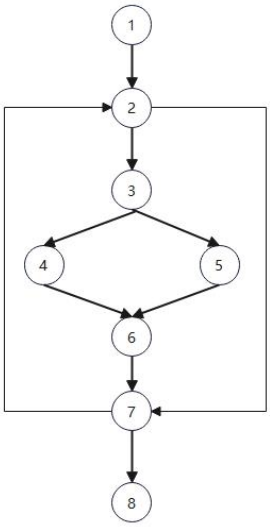
D(4)={1,2,3,4}

D(5)={1,2,3,5}

D(6)={1,2,3,6}

D(7)={1,2,7}

D(8)={1,2,7,8}



（2）回边： 7→2

（3）循环：{2，3，4，5，6，7}

图 2、(1)流图中各结点 N 的必经结点集 D(n)：

D(l)＝{1}

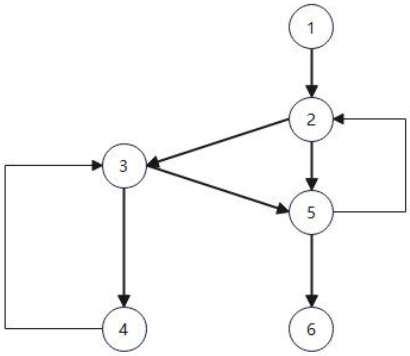
D(2)＝{1,2}

D(3)＝{1,2,3}

D(4)={1,2,3,4}

D(5)＝{1,2,5}

D(6)＝{1,2,5,6}



(2)求出流图中的回边：

5→2，4→3

(3)求出流图中的循环：

回边 5→2 对应的循环：｛2，5，3，4｝

回边 4→3 对应的循环：｛3，4｝

3、C 代码的部分三地址代码序列。

```
void quicksort(m,n)
int m,n;
{ int i,j;
int v,x; if (n<=m) return;
/* fragment begins here */
i = m-1;
j = n;
v = a[n];
while(1) {
do i = i+1;while (a[i]<v);
do j = j-1; while (a[j]>v);
if (i>=j) break;
x = a[i];
a[i] = a[j];
a[j] = x;
}
x = a[i];
a[i] = a[n];
a[n] = x;
/* fragment ends here */
quicksort (m,j);
quicksort(i+1,n);
}
```

三地址代码：

```
(1) i:=m-1
```

(2) j:=n

(3) t1:=4*n

(4) v:=a[t1]

(5) i:=i+1

(6) t2:=4*i

(7) t3:=a[t2]

(8) if t3< v goto (5)

(9) j:=j-1

(10) t5:=4*j

(11) t5:=a[t4]

(12) if t5> v goto (9)

(13) if i >= j goto (23)

(14) t6:=4*i

(15) x:=a[t6]

(16) t7:=4*i

(17) t6:=4*j

(18) t9:=a[t8]

(19) a[t7]:=t9

(20) t10:=4*j

(21) a[t10]:=x

(22) goto (5)

(23) t11:=4*i

(24) x:=a[t11]

(25) t12:=4*i

(26) t13:=4*n

(27) t14:=a[t13]

(28) a[t12]:=t14

(29) t15:=4*n

(30) a[t15]:=x

请将三地址代码序列划分为基本块并做出其流图。

答：

基本块：

B1：（1）-（4）

B2：（5）-（8）

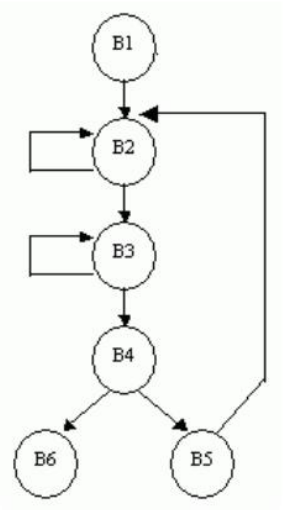
B3：（9）-（12）

B4：（13）

B5：（14）-（22）

B6：（23）-（30）

流图



- 运行时刻环境

1、有如下示意的 Pascal 源程序

```
program main;
        var a,b,c:integer;
        procedure X(i,j:integer);
          var d,e:real;
          procedure Y;
                var f,g:real;
                begin
                ...
                end;{Y}
          procedure Z(k:integer);
                var h,i,j:real;
                begin
                ...
                end;{Z}
          begin
                .....
                10:Y;
                .....
                11:Z;
                .....
          end;{X}
        begin
          .....
          X(a,b);
          .....
        end.{main}
```

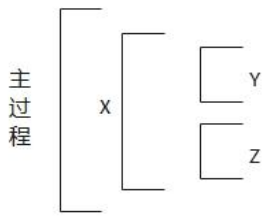
并已知在运行时刻，以过程为单位对程序中的变量进行动态存储分配。当运行主程序而调用过程语句 X (a,b)时，试分别给出以下时刻的运行栈的内容和 DISPLAY 的内容。

（1）已开始而尚未执行完毕标号为 10 的语句。

（2）已开始而尚未执行完毕标号为 11 的语句。

答案：

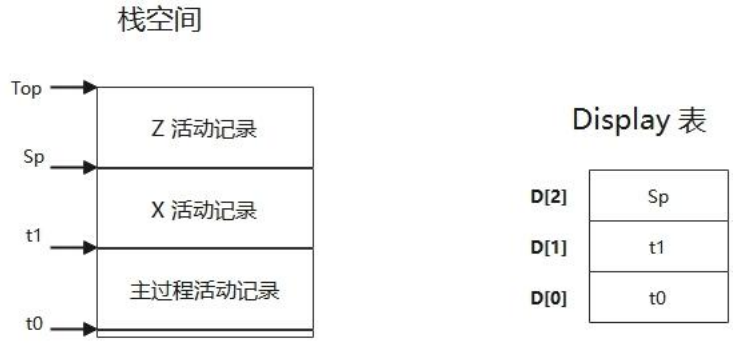
程序结构：



（1）


栈空间
Top
Sp
t1
t0
Y 活动记录
X 活动记录
主过程活动记录
Display 表
D[2]
D[1]
D[0]
Sp
t1
t0

（2）


栈空间
Top
Sp
t1
t0
Z 活动记录
X 活动记录
主过程活动记录
Display 表
D[2]
D[1]
D[0]
Sp
t1
t0