第二章

P-36-6

（1）L（G）是 0~9 组成的数字串；

（2）最左推导：

N⇒ND⇒NDD⇒NDDD⇒DDDD⇒0DDD⇒01DD⇒012D⇒0127

N⇒ND⇒DD⇒3D⇒34

N⇒ND⇒NDD⇒DDD⇒5DD⇒56D⇒568

最右推导：

N⇒ND⇒N7⇒ND7⇒N27⇒ND27⇒N127⇒D127⇒0127

N⇒ND⇒N4⇒D4⇒34

N⇒ND⇒N8⇒ND8⇒N68⇒D68⇒568

P-36-7

G（S)：(没有考虑正负符号问题）

S→P|AP

P→1|3|5|7|9

A→AD|N

N→2|4|6|8|P

D→0|N

或者：（1）S→ A B C ｜C

A→ 1 ｜ 2 ｜ 3 ｜ 4 ｜ 5 ｜ 6 ｜ 7 ｜ 8 ｜ 9

B→ B A ｜ B 0 ｜ε

C→1｜3｜5｜7｜9

P-36-8

G（E)：E→T|E+T|E-T

T→F|T*F|T/F

F→（E）|i

最左推导：

E⇒E+T⇒T+T⇒F+T⇒i+T⇒i+T*F⇒i+F*F⇒i+i*F⇒i+i*i

E⇒T⇒T*F⇒F*F⇒i*F⇒i*（E）⇒i*（E+T）⇒i*（T+T）⇒i*（F+T）⇒i*（i+T）⇒i*（i+F）⇒i*（i+i）

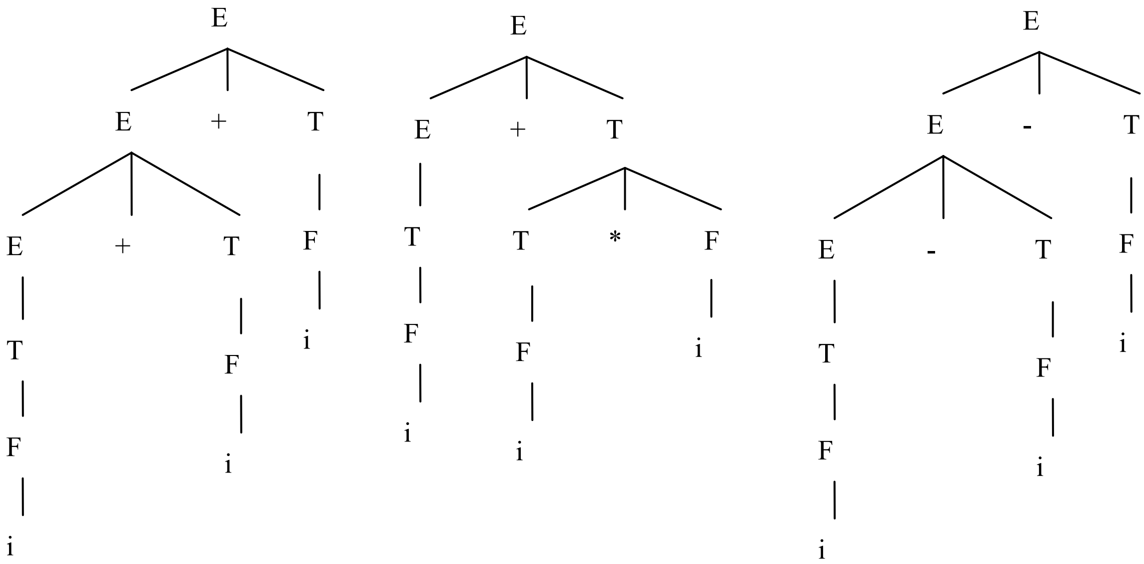
最右推导：

E⇒E+T⇒E+T*F⇒E+T*i⇒E+F*i⇒E+i*i⇒T+i*i⇒F+i*i⇒i+i*i

E⇒T⇒T*F⇒T*（E）⇒T*（E+T）⇒T*（E+F）⇒T*（E+i）⇒T*（T+i）⇒T*（F+i）⇒T*（i+i）

⇒F*（i+i）⇒i*（i+i）

语法树：



i+i+i　　　　i+i*i　　　　i-i-i

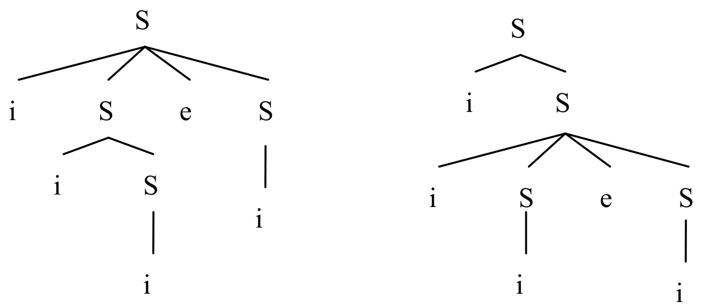
P-36-9

句子：iiiei 有两个语法树：

S⇒iSeS⇒iSei⇒iiSei⇒iiiei

S⇒iS⇒iiSeS⇒iiSei⇒iiiei

因此 iiiei 是二义性句子，因此该文法是二义性的。



P-36-10

S→TS|T

T→(S)|()

P-36-11

L1：G(S)：　S→AC

　　　　　　A→aAb|ab

　　　　　　C→cC|ε

L2：G(S)：　S→AB

　　　　　　A→aA|ε

　　　　　　B→bBc|bc

L3：G(S)：　S→AB

　　　　　　A→aAb|ε

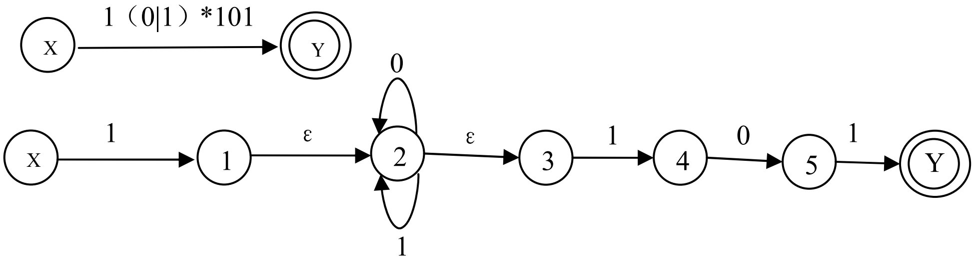
　　　　　　B→aAb|ε

L4：G(S)：　S→1S0|A

　　　　　　A→0A1|ε

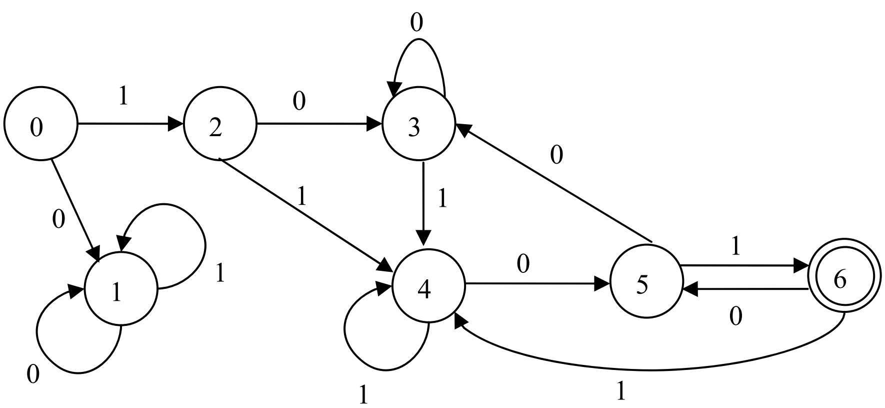
或者：S→A|B　A→0A1|ε　B→1B0|A

（1）



确定化：

	0	1
{X}	Φ	{1，2，3}
Φ	Φ	Φ
{1，2，3}	{2，3}	{2，3，4}
{2，3}	{2，3}	{2，3，4}
{2，3，4}	{2，3，5}	{2，3，4}
{2，3，5}	{2，3}	{2，3，4，Y}
{2，3，4，Y}	{2，3，5}	{2，3，4}



最小化：{0，1，2，3，4，5}，{6}

{0，1，2，3，4，5}$_0$={1，3，5}　　{0，1，2，3，4，5}$_1$={1，2，4，6}

{0，1，2，3，4}，{5}，{6}

{0，1，2，3，4}$_0$={1，3，5}

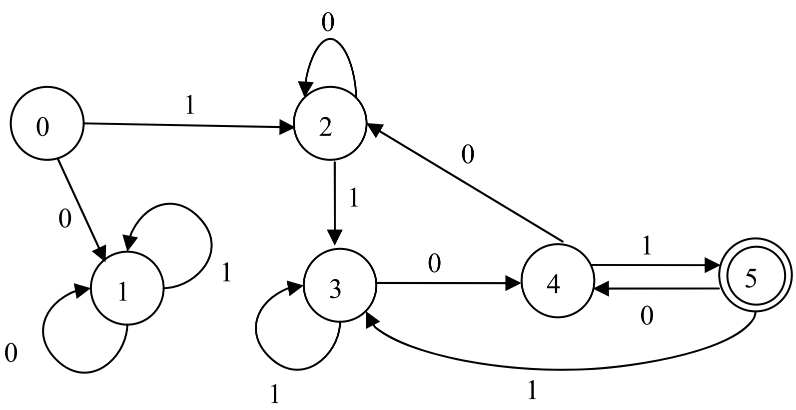
{0，1，2，3}，{4}，{5}，{6}

{0，1，2，3}$_0$={1，3}　　{0，1，2，3}$_1$={1，2，4}

{0，1}，{2，3}，{4}，{5}，{6}

{0，1}$_0$={1}　{0，1}$_1$={1，2}　　{2，3}$_0$={3}　{2，3}$_1$={4}

{0}，{1}，{2，3}，{4}，{5}，{6}



P64-8

(1)

$(0|1)^*01$

(2)

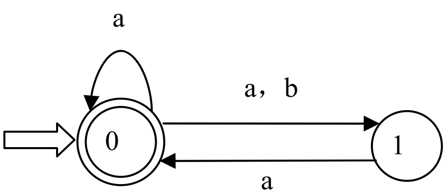
$(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)^*(0|5)\ |\ (0|5)$

(3)

$0^*1(0|10^*1)^*\ |\ 1^*0(1|01^*0)^*$

P84-12

（a）

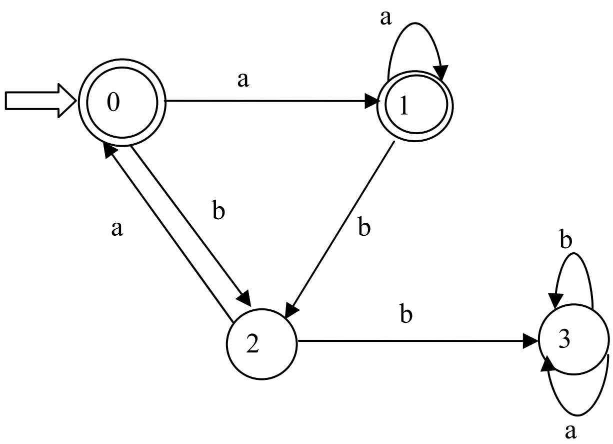


确定化：

	a	b
{0}	{0,1}	{1}
{0,1}	{0,1}	{1}
{1}	{0}	Φ
Φ	Φ	Φ

给状态编号：

	a	B
0	1	2
1	1	2
2	0	3
3	3	3



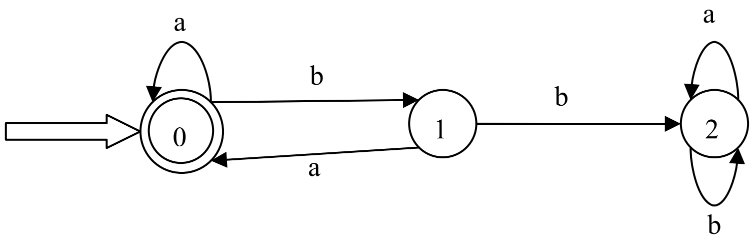
最小化：

{0，1}　　{2，3}

{0，1}a={1}，{0，1}b={2}

{2，3}a={0，3}，{2，3}={3}

{0，1}，{2}，{3}



（b）

已经确定化，只需最小化：

{0，1}，{2，3，4，5}

$\{0，1\}_a = \{1\}$　　　$\{0，1\}_b = \{2，4\}$

$\{2，3，4，5\}_a = \{1，3，0，5\}$　　$\{2，3，4，5\}_b = \{2，3，4，5\}$

又：$\{2，4\}_a = \{1，0\}$　　$\{2，4\}b = \{3，5\}$　　$\{3，5\}_a=\{3，5\}$　　$\{3，5\}_b = \{2，4\}$

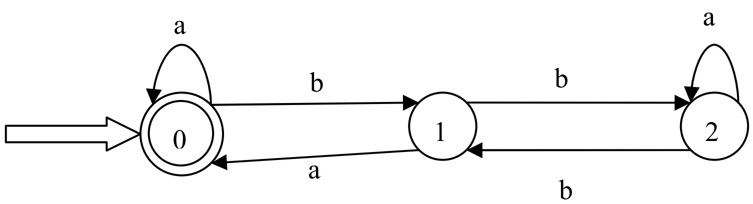
分划为：{0，1}，{2，4}，{3，5}

$\{0，1\}_a = \{1\}$　　　　$\{0，1\}_b = \{2，4\}$

$\{2，4\}_a = \{1，0\}$　　　$\{2，4\}_b = \{3，5\}$

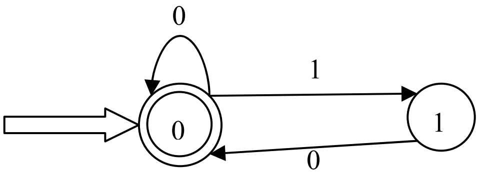
$\{3，5\}_a = \{3，5\}$　　　$\{3，5\}_b = \{2，4\}$

所以不能再分

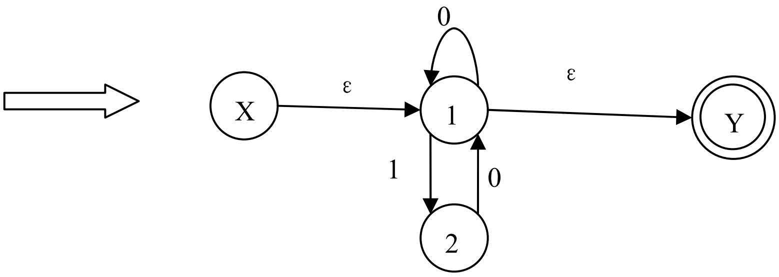


P64-14

正规式：（0|10）*



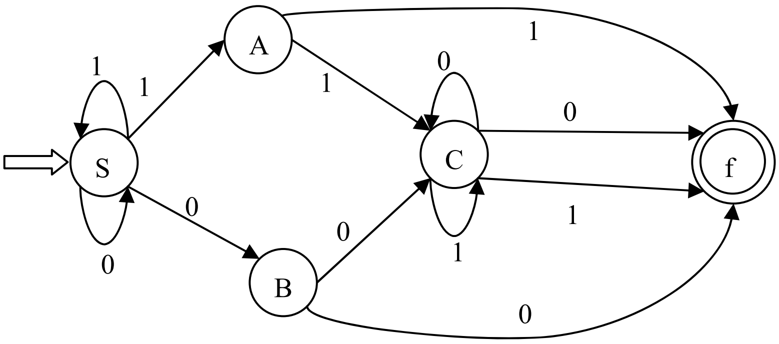
还可以：



然后再确定化，最小化，结果应该一样。

P65-15

首先构造 NFA：



则有：G（f） f→A1|B0|C1|C0

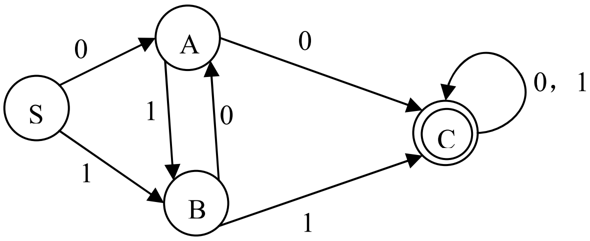
C→C0|C1|A1|B0

A→S1|1

B→S0|0

S→S0|S1|0|1

或者是确定化，然后最小化：

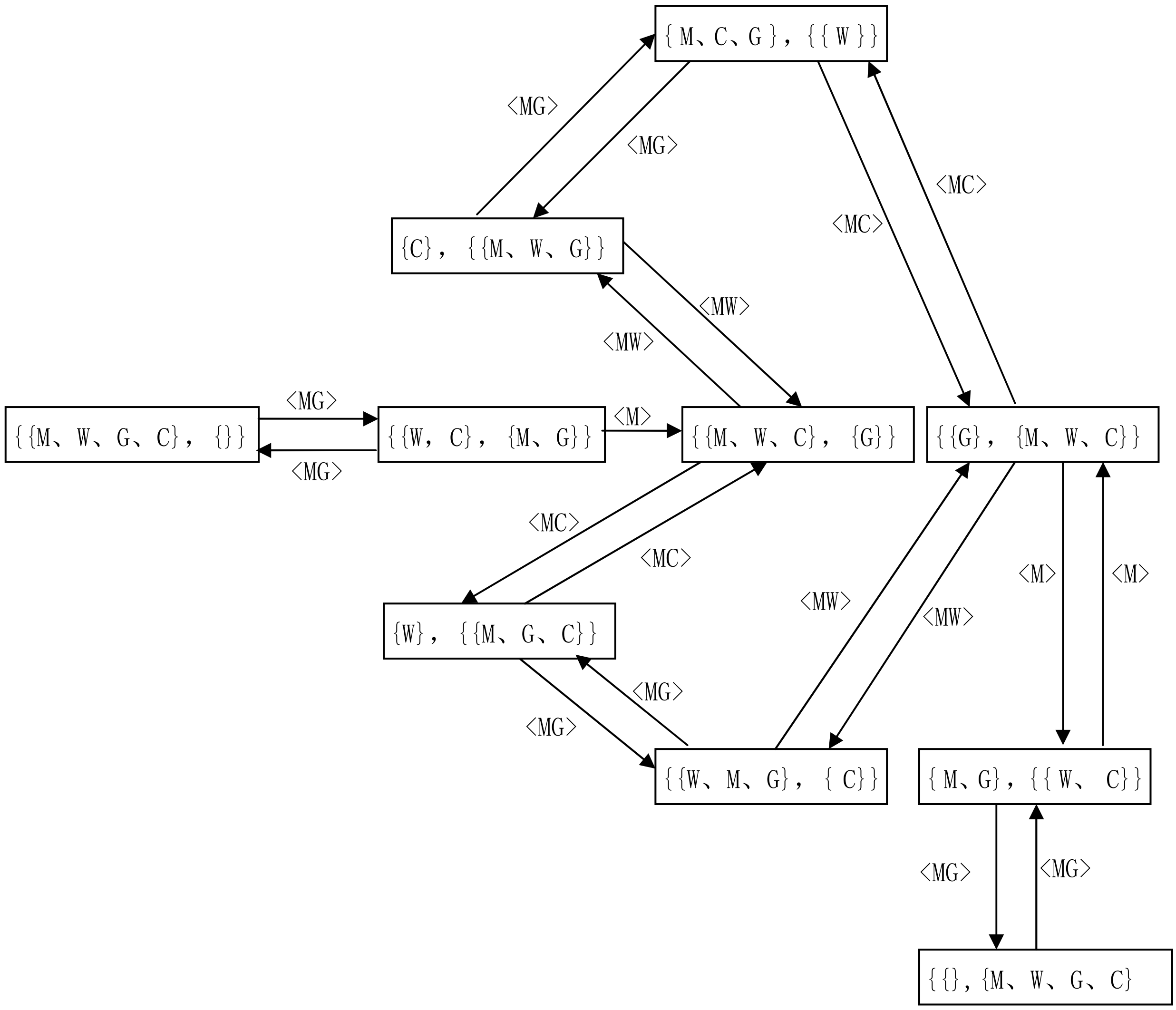


G（C） C→C0|C1|A0|B1

A→0|B0

B→1|A1

人、狼、羊、白菜：
{{M、W、G、C}，{}}表示在左岸，{{}，{M、W、G、C}}在右岸，将可能存在的状态中去掉不安全状态，剩下：
{{M、W、G、C}，{}}，{{}，{M、W、G、C}}，{{M、W、G}，{C}}，{{M、W、C}，{G}}，
{{M、G、C}，{W}}　，{{C}，{M、W、G}}　，{{G}，{M、W、C}}，{{W}，{M、G、C}}，
{{M、G}，{W、C}}　，{{W，C}，{M、G}}
箭弧上的标记符：<M>：表示人单独过河、<MG>：表示人和羊过河、<MW>：表示人和狼过河、<MC>：表示人和白菜过河



第四章

P81-1

（1） 按照 T，S 的顺序消除左递归

G’（S）：S→a|∧|（T）

T→ST’

T’→，ST’|ε

递归下降子程序：

```
procedure S:
begin
    if sym = ‘a’ or sym= ‘∧’
       then advance
    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 begin
       Advance;
       S;T’
   End
End
```

其中：sysm 为输入串指针所指的符号；advance 是把输入指针调至下一输入符号。

（2） 求 First 和 Follow 集合：

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’→ ε	T’→,ST’	

P81-2

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

（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,∧,),# }

（2）文法无左递归，考察 E'→+E|ε　T'→T|ε　F'→*F'|ε　P→（E）|a|b|∧

E'→+E|ε： First(E') = {+, ε}∩Follow(E')={#,)} = Φ

T'→T|ε：　First(T') = {(,a,b, ∧, ε} ∩Follow(T')={+,),#} = Φ

F'→*F'|ε：First(F') = {*, ε}∩Follow(F')={ (,a,b,∧,),# } = Φ

P→（E）|a|b|∧：候选式终结首符集两两不相交

所以该文法为 LL(1)文法。

（3）　LL(1)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→∧	

（4）　构造递归下降程序

```
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 begin  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
```

P81-3

解答:（1）该文法不含左递归，计算 First 集合和 Follow 集合

Fisrt(S) = {a，b，c}　First(A) = {a，ε}　First（B）= {b, ε}

Follow(S)={#}　Follow（A）={b，c}　Follow(B) = {c}

满足 LL(1)文法的 3 个条件，所以是 LL(1)文法；

（2）该文法不含左递归，计算 First 集合和 Follow 集合

Fisrt(S) = {a，b}　First(A) = {a，b，ε}　First（B）= {b, ε}

Follow(S)={#}　Follow（A）={b}　Follow(B) = {b}

考虑 A→a|B|ε，Fisrt(A)中含有ε，而 Fisrt(A)∩Follow（A）={b}，所以不是 LL(1)文法；

（3）该文法不含左递归，计算 First 集合和 Follow 集合

Fisrt(S) = {a，b，ε}　First(A) = {a，ε}　First（B）= {b, ε}

Follow(S)={#}　Follow（A）={a，b，#}　Follow(B) = {a，b，#}

考虑 A→a|ε，Fisrt(A)中含有ε，而 Fisrt(A)∩Follow（A）={a}，所以不是 LL(1)文法；

(4)是 LL（1）文法

P82-4

文法：Expr→-Expr

Expr→(Expr)|Var ExprTail

ExprTail→-Expr|ε

Var→id VarTail

VarTail→(Expr) |ε

解答：First(Expr)={-, (, id}　　First (Var) ={id}

First(ExprTail)={-, ε}　　First(VarTail) = {(, ε}

Follow(Expr)={#,)}　　Follow(Var) = {-, #,)}

Follow (ExprTail) ={#,)}　　Follow(VarTail)={ -, #,)}

所以 LL(1)分析表：

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

分析 id—id((id))

分析栈	输入	所用产生式
#Expr	id--id((id)) #	
#ExprTail Var	id--id((id)) #	Expr→Var ExprTail
#ExprTail VarTail id	id--id((id)) #	Var→id VarTail
#ExprTail VarTail	--id((id)) #	
#ExprTail	--id((id)) #	VarTail→ε
#Expr-	--id((id)) #	ExprTail→-Expr
#Expr	-id((id)) #	
#Expr-	-id((id)) #	Expr→-Expr
#Expr	id((id)) #	
#ExprTail Var	id((id)) #	Expr→Var ExprTail
#ExprTail VarTail id	id((id)) #	Var→id VarTail
#ExprTail VarTail	((id)) #	
#ExprTail)Expr(	((id)) #	VarTail→(Expr)
#ExprTail)Expr	(id)) #	
#ExprTail))Expr(	(id)) #	Expr→(Expr)
#ExprTail))Expr	id)) #	
#ExprTail))ExprTail Var	id)) #	Expr→Var ExprTail
#ExprTail))ExprTail VarTail id	id)) #	Var→id VarTail
#ExprTail)) ExprTail VarTail	)) #	
#ExprTail)) ExprTail	)) #	VarTail→ε
#ExprTail))	)) #	ExprTail→ε
#ExprTail)	) #	
#ExprTail	#	
#	#	ExprTail→ε

第五章

P133-1

E⇒E+T⇒E+T*F

短语：E+T*F，T*F
直接短语：T*F
句柄：T*F
P133-2

文法：S→a|∧|（T）

　　T→T，S|S
（1）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))

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

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))

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

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

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

（2）
(((a，a)，∧，(a))，a)
(((S，a)，∧，(a))，a)
(((T，a)，∧，(a))，a)
(((T，S)，∧，(a))，a)
((（T），∧，(a))，a)
((S，∧，(a))，a)
((T，∧，(a))，a)
((T，S，(a))，a)
((T，(a))，a)
((T，(S))，a)
((T，（T）)，a)
((T，S)，a)
(（T），a)

（S，a）
（T，a）
（T，S）
（T）
S

移进归约过程：

步骤	栈	输入串	动作
0	#	（((a，a)，∧，(a))，a）#	初始
1	#（	((a，a)，∧，(a))，a）#	移进
2	#((	(a，a)，∧，(a))，a）#	移进
3	#(((	A，a)，∧，(a))，a）#	移进
4	#((a	，a)，∧，(a))，a）#	移进
5	#(((S	，a)，∧，(a))，a）#	归约
6	#(((T	，a)，∧，(a))，a）#	归约
7	#(((T，	A)，∧，(a))，a）#	移进
8	#(((T，a	)，∧，(a))，a）#	移进
9	#(((T，S	)，∧，(a))，a）#	归约
10	#(((T	)，∧，(a))，a）#	归约
11	#(((T)	，∧，(a))，a）#	移进
12	#((S	，∧，(a))，a）#	归约
13	#((T	，∧，(a))，a）#	归约
14	#((T，	∧，(a))，a）#	移进
15	#((T，∧	，(a))，a）#	移进
16	#((T，S	，(a))，a）#	归约
17	#((T	，(a))，a）#	归约
18	#((T，	(a))，a）#	移进
19	#((T，(	a))，a）#	移进
20	#((T，(a	))，a）#	移进
21	#((T，(S	))，a）#	归约
22	#((T，(T	))，a）#	归约
23	#((T，(T)	)，a）#	移进
24	#((T，S	)，a）#	归约
25	#((T	)，a）#	归约
26	#((T)	，a）#	移进
27	#（S	，a）#	归约
28	#（T	，a）#	归约
29	#（T，	a）#	移进
30	#（T，a	）#	移进
31	#（T，S	）#	归约
32	#（T	）#	归约
33	#（T）	#	移进
34	#S	#	归约

P133-3：文法：G（S）：S→a|∧|(T)　　T→T,S|S

（1）FIRSTVT（S）={ a、∧、(}　　FIRSTVT（T）={ ,、a、∧、(}

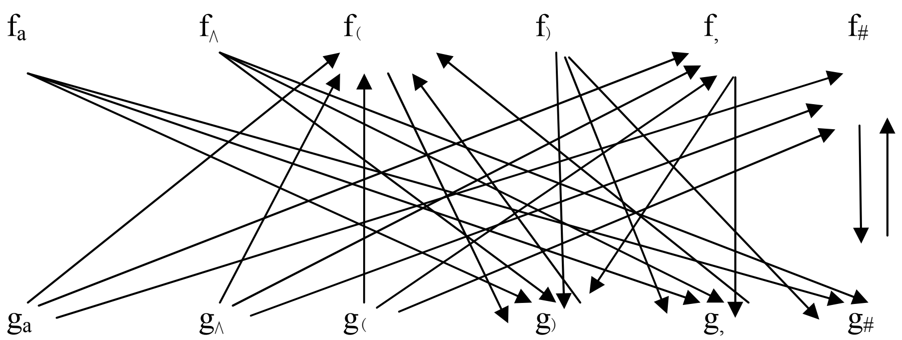
　　LASTVT（S）={ a、∧、）}　　LASTVT（T）={,、a、∧、）}

（2）算符优先分析表

	a	∧	（	）	，	#
a				⋗	⋗	⋗
∧				⋗	⋗	⋗
（	⋖	⋖	⋖	=	⋖	
）				⋗	⋗	⋗
，	⋖	⋖	⋖	⋗	⋗	
#	⋖	⋖	⋖			=

（3）优先函数：

	a	∧	（	）	，	#
f	6	6	2	6	4	2
g	7	7	7	2	3	2



如果不考虑#，则：优先函数：

	a	∧	（	）	，
f	4	4	2	4	4
g	5	5	5	2	3

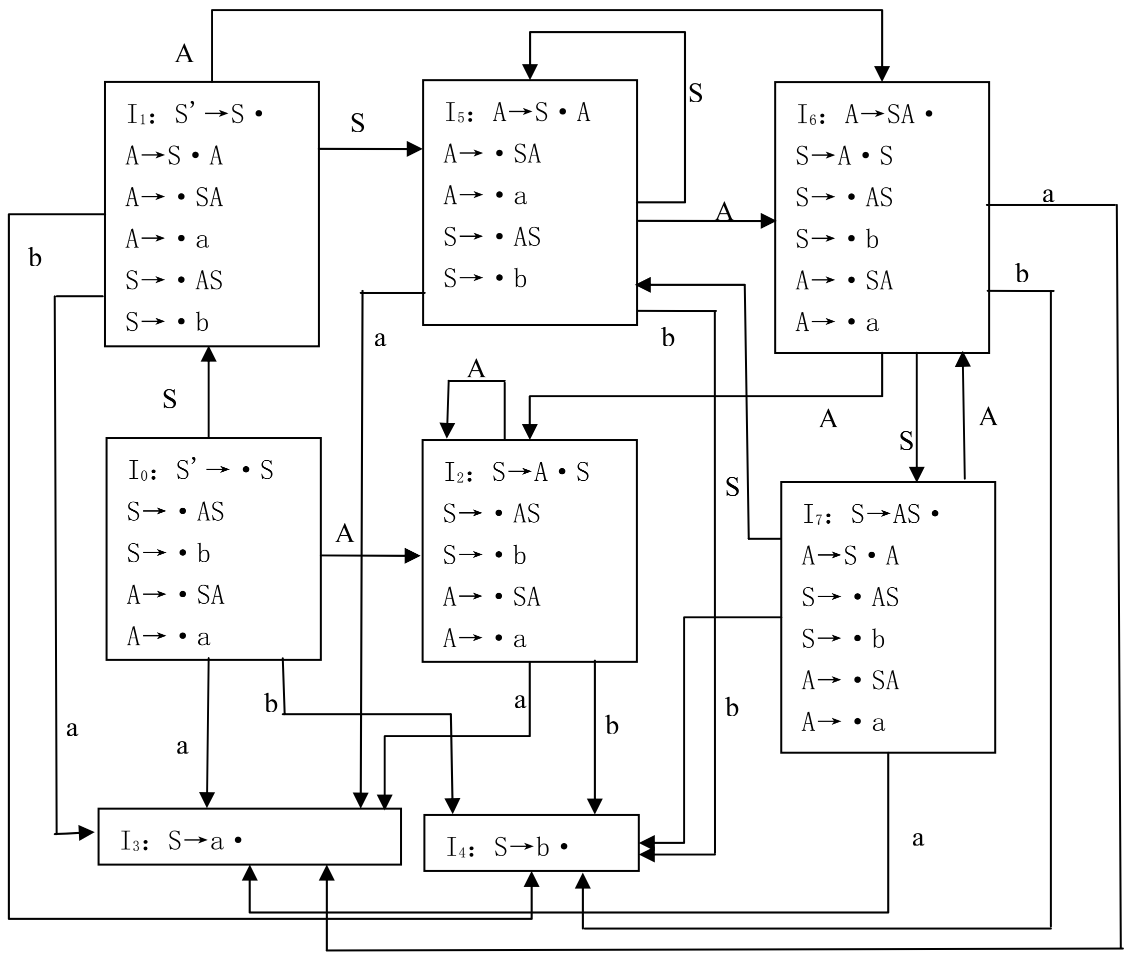
分析过程：

栈	输入	
#	（a,(a,a)）#	初始
#(	a,(a,a)）#	移进
#(a	,(a,a)）#	移进
#(S	,(a,a)）#	归约
#(S,	(a,a)）#	移进
#(S,(	a,a)）#	移进
#(S,(a	,a)）#	移进
#(S,(S	,a)）#	归约
#(S,(S,	a)）#	移进

#(S,(S,a	))#	移进
#(S,(S,S	))#	归约
#(S,(T	))#	归约
#(S,(T)	)#	移进
#(S,S	)#	归约
#(T	)#	归约
#(T)	#	移进
#S	#	归约

P134-5

（1）



考察 I1、I6、I7:

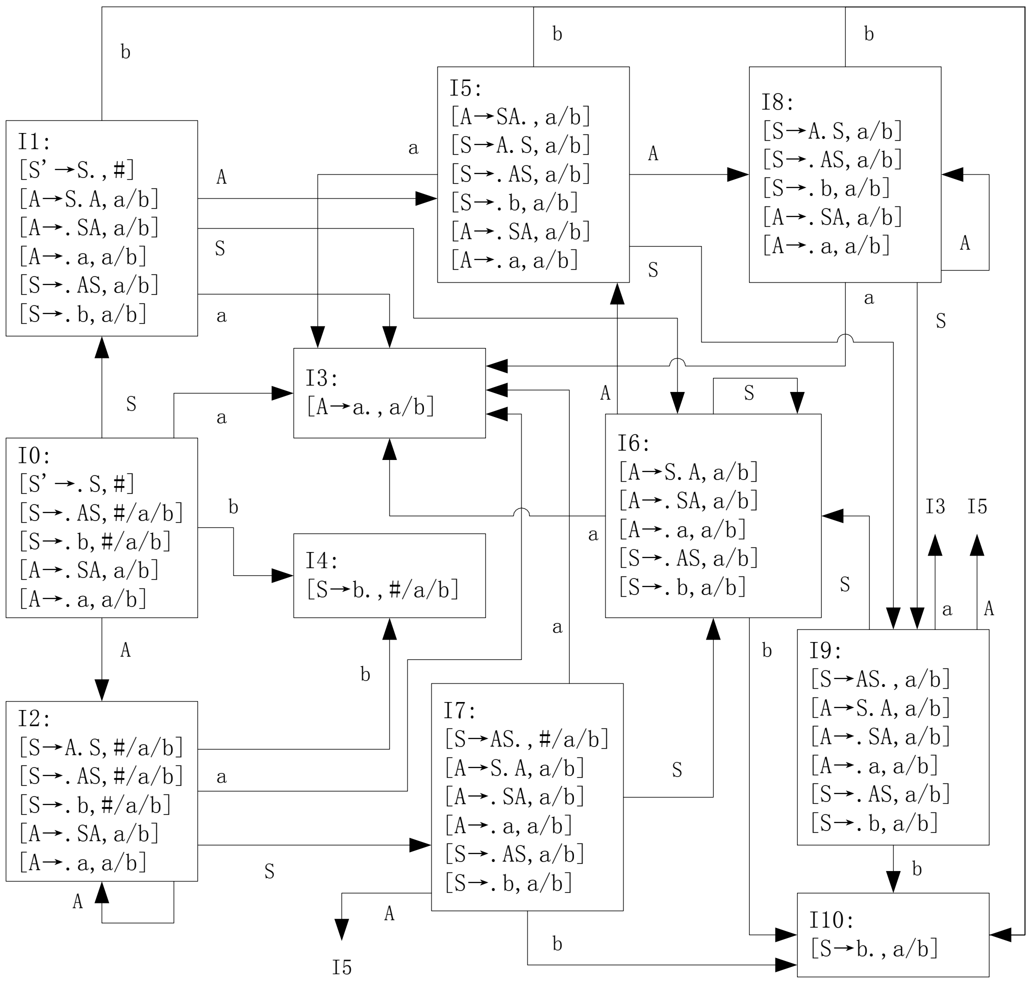
I1:存在移进-归约冲突，因为 Follow（S'）={#},不包含 a 或 b，因此冲突可以使用 SLR 解决方法解决。

I6：存在移进-归约冲突，因为 Follow（A）={a，b}，因此无法使用 SLR 方法解决移进-归约冲突

I7：存在移进-归约冲突，因为 Follow（S）={#，a，b}，因此无法解决移进-归约冲突

所以不是 SLR(1)文法。

构造 LR（1）项目集规范族：



检查 I5，[A→SA.，a/b]，要求输入为 a 或者 b 使用 A→SA 归约，而[S→.b，a/b]及[A→.a，a/b]要求移进，因此存在移进-归约冲突，所以不是 LR（1）文法。

P135-8

解答：

不存在左递归；

因为 Fist(AaAb)={a}，First(BbBa)={b}所以交集为空

所以该文法是 LL(1)文法。

I0={S→.AaAb，S→.BbBa，A→.，B→.}

I1=GO(I0，A)={ S→A.aAb }

I2=GO(I0，B)={ S→B.bBa }

I3=GO(I1,a)={ S→Aa.Ab，A→.}

I4=GO(I2,b)={ S→Bb.Ba，B→.}

I5=GO(I3,A)={ S→AaA.b }

I6=GO(I4,B)={ S→BbB.a }

I7=GO(I5,b)={ S→AaAb.}

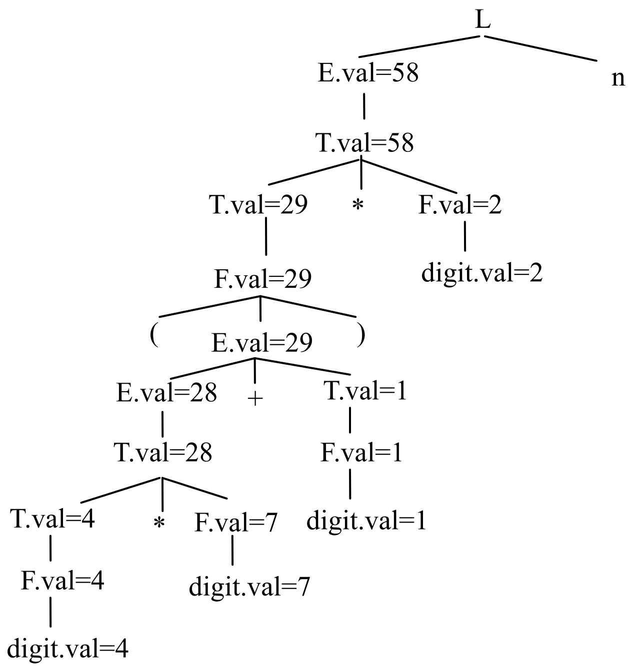
I8=GO(I6,a)= { S→BbBa.}

考虑：I0:存在两个归约项目，A→.,B→.,Follow(A)={a,b}，Follow(B)={a,b}，所以冲突不能解决，不是 SLR(1)文法。

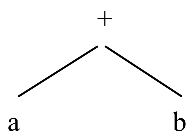
第六章

P164-1

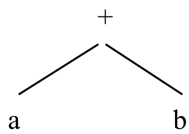
解答：表达式（4*7+1）*2 的附注语法树：



P164-2



(1)



(2)

p165-5

(1)

$E \rightarrow E_1+T$ { if (E_1.type = int) and (T.type = int) then E.type = int
Else E.type = real }

$E \rightarrow T$ { E.type = T.type }

$T \rightarrow$num. num { T.type = real }

$T \rightarrow$num { T.type = int }

(2)

$E \rightarrow E_1+T$ { if (E1.type = int) and (T.type = int) then

E.type = int

E.code = E_1.code || T.code || +

Else if (E_1.type = real) and (T.type = real)

E.type = real

E.code = E_1.code || T.code || +

Else if E_1.type = int then

E.type = real

E.code = E_1.code ||inttoreal || T.code || +

Else

$E.type = real$

$E.code = E_1.code \parallel T.code \parallel inttoreal \parallel +$

End if

E→T { E.type = T.type
E.code = T.code}

T→num. num { T.type = real
E.code = num.num }

T→num { T.type = int
E.code = num }

P164-7

$S \to L_1.L_2$ { $S.val = L_1.val + L_2.val / 2^{L_2.length}$ }

S→L { S.val = L.val }

$L \to L_1B$ { $L.val = 2*L_1.val + B.c$
$L.length = L_1.length + 1$ }

L→B { L.val = B.c
L.length = 1 }

B→0 { B.c = 0}

B→1 { B.c = 1}

P165-11

对 D，L，T 设置综合属性 type。过程 addtype（id.entry，type）用来将标识符 id 的类型 type 填入到符号表中。

（1） 翻译模式：

D→id L { addtype（id.entry，L.type）}

L→, id L_1 { $L.type = L_1.type$ ； addtype（id.entry，L_1.type）}

L→:T { L.type = T.type }

T→integer { T.type = integer }

T→real { T.type = real }

(2) 假设 Ttype 为已定义的表示“类型”的数据结构，预测翻译器如下：

```
procedure D；
  var l_type：Ttype
  begin
    if sym = “id” then
      begin
        advance ;
        l_type = L ;
        addtype(id.entry , l_type)
    end
    else error
end；
```

```
procedure L；
      var l_type :Ttype；
      begin
         if sym =   “ , ”   then
               begin
                  advance;
                  if sym = “id” then
                        begin
                           advance ;
                           l_type = L ;
                           adddtype(id.entry，l_type)
                        end
                  else error ;
               end
         else if sym = “:” then
            begin
               advance ;
               l_type = T ;
               end
         else error ;
   return (l_type) ；
end；

procedure T ;
  var t_type: Ttype ;
  begin
    if sym = “integer” then
        begin
          advance ;
          t_type = integer ;
        end
    else if sym = “real” then
        begin
          advance ;
          t_type = real ;
        end
    else error
    return(t_type);
end;
```

第七章

P217-1

a*（-b+c）　　　后缀式：ab-c+*

a+b*（c+d/e）　　后缀式：abcde/+*+

-a+b*（-c+d）　　后缀式：a-bc-d+*+

not A or not（C or not D）　后缀式：A not C D not or not or

（A and B）or（not C or D）　后缀式：A B and C not D or or

（A or B）and（C or not D and E）后缀式：　A B or C D not E and or and

if　（x + y）*z = 0 then　（a+b）↑c else a↑b↑c　后缀式：xy+z*0= ab+c↑ abc↑↑ if-then-else

P217-3

-（a+b）*（c+d）-（a+b+c）

三元式：

（1）+，a，b

（2）-，（1），-

（3）+，c，d

（4）*，（2），（3）

（5）+，a，b

（6）+，（5），c

（7）-，（4），（6）

间接三元式：

三元式表：

（1）+，a，b

（2）-，（1），-

（3）+，c，d

（4）*，（2），（3）

（5）+，（1），c

（6）-，（4），（5）

间接码表：（1），（2），（3），（4），（1），（5），（6）

四元式序列：

（1）+，a，b，T1

（2）-，T1，-，T2

（3）+，c，d，T3

（4）*，T2，T3，T4

（5）+，a，b，T5

（6）+，T5，c，T6

（7）-，T4，T6，T7

P218-8

自下而上分析过程中把赋值语句 A := B * （-C + D）翻译成四元式的步骤：

步骤	输入串	栈	PLACE	四元式
（1）	A := B * （-C + D）			
（2）	:= B * （-C + D）	i	A	
（3）	B * （-C + D）	i :=	A-	
（4）	* （-C + D）	i := i	A-B	
（5）	* （-C + D）	i := E	A-B	
（6）	（-C + D）	i := E*	A-B-	
（7）	-C + D）	i := E*（	A-B--	
（8）	C + D）	i := E*（-	A-B---	
（9）	+ D）	i := E*（-i	A-B---C	
（10）	+ D）	i := E*（-E	A-B---C	（-，C，-，T1）
（11）	+ D）	i := E*（E	A-B--T1	
（12）	D）	i := E*（E+	A-B--T1-	
（13）	）	i := E*（E+i	A-B--T1-D	
（14）	）	i := E*（E+E	A-B--T1-D	（+，T1，D，T2）
（15）	）	i := E*（E	A-B--T2	
（16）		i := E*（E）	A-B--T2-	
（17）		i := E*E	A-B-T2	（*，B，T2，T3）
（18）		i := E	A-T3	（:=，T3，-，A）
（19）		A		

P218-5

设 A、B 为 10 x 20 的数组，C、D 大小为 10 的数组，数组每维下届为 1，每个数据项宽度为 4，则：

A[i，j] := B[i，j] + C[A[k，1]] + D[i+j]

T1 := i * 20

T1 := T1 + j

T2 := A – 84

T3 := 4 * T1

T4 := i * 20

T4 := T4 + j

T5 := B – 84

T6 := 4 * T4

T7 := T5[T6]

T8 := k*20

T8 = T8 + 1

T9 := A – 84

T10 := 4 * T8

T11 := T9[T10]

T12 := C – 4

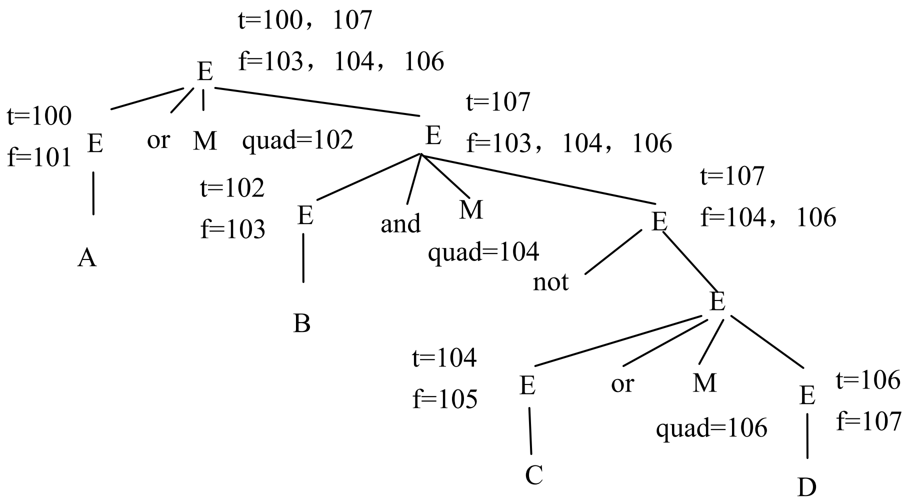
T13 := 4 * T11

T14 := T12[T13]

T15 := T7 + T14
T16 := i + j
T17 := D – 4
T18 := 4*T16
T19 :=T17[T18]
T20 :=T15 + T19
T2[T3] := T20

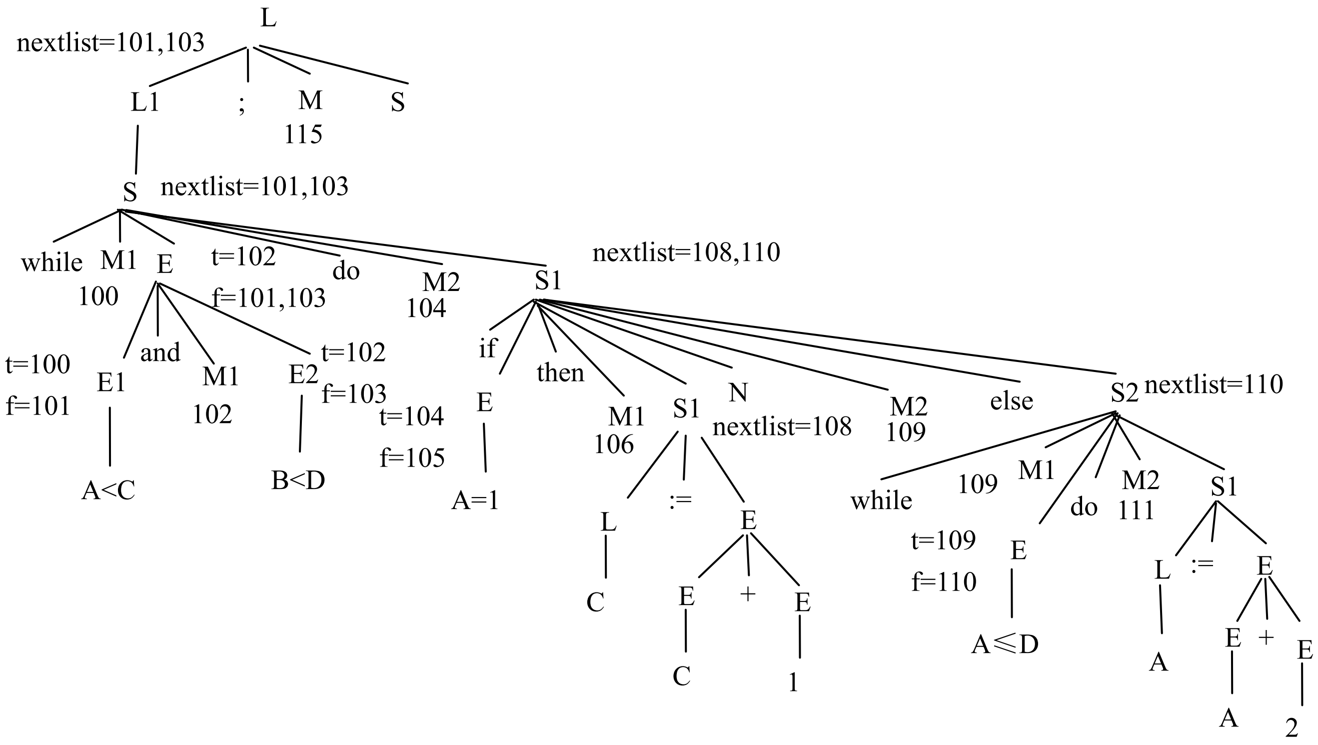
P218-6

A or （B and not（C or D））：



100：（jnz，A，-，0）
101：（j，-，-，102）
102：（jnz，B，-，104）
103：（j，-，-，0）
104：（jnz，C，-，0）
105：（j，-，-，106）
106：（jnz，D，-，0）
107：（j，-，-，0）

P218-7



100：（j<，A，C，102）
101：（j，-，-，115）
102：（j<，B，D，104）
103：（j，-，-，115）
104：（j=，A，‘1’，106）
105：（j，-，-，109）
106：（+，C，‘1’，T1）
107：（:=，T1，-，C）
108：（j，-，-，100）
109：（j⩽，A，D，111）
110：（j，-，-，100）
111：（+，A，‘2’，T2）
112：（:=，T2，-，A）
113：（j，-，-，109）
114：（j，-，-，100）
115：

P219-12

（1） 如果该程序执行，则先会打印出：

MAXINT-5

MAXINT-4

MAXINT-3

MAXINT-2

MAXINT-1

MAXINT

然后对于有些可能出现的整型数溢出而出现运行时的异常。

（2） 根据其语义，先确定 PASCAL 语言 for 语句的中间代码结构如下：

```
    t1 := initial
    t2 := final
    if t1 > t2 goto L2
    v := t1
L1：S 的代码
    if v = t2 goto L2
    v := v + 1
    goto L1
L2：
```

为了便于语法制导的翻译，将 PASCAL 语言的 for 语句：

S→for V := E1 to E2 do S1

改写成如下产生式：

S→F do S1

F→for v := E1 to E2

翻译模式如下：

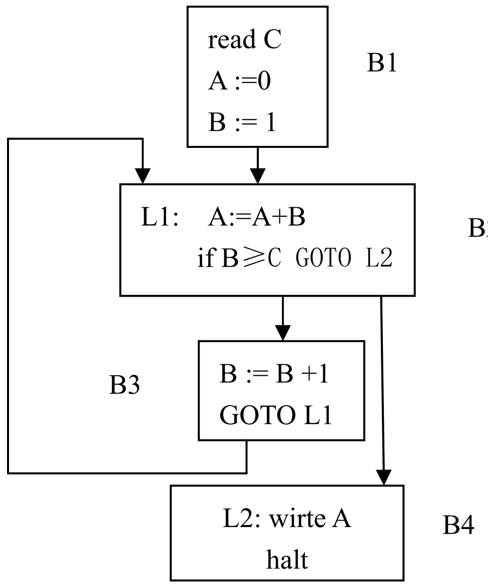
```
F→for v := E1 to E2
  { F.nextlist := makelist(nextquad);
    emit (j>, E1.place, E2.place, 0);
    emit (:=, E1.place, -, v.place);
    F.quad := nextquad;
    F.place1 := E2.place;
    F.place2 := entry (v); }
S→F do S1
  { backpatch (S1.nextlist, F.quad);
    S.nextlist := merge (F.nextlist, makelist (nextquad));
    emit(j=, F.place1, F.place2, 0);
    emit (+, F.place2, 1, F.place2);
    emit (j, -, -, F.quad) }
```

第十章

P306-1：

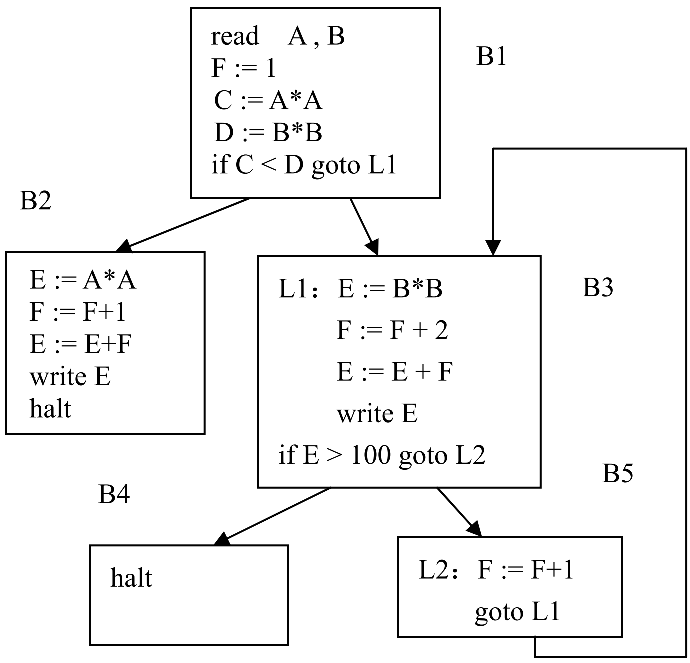
```
    read C
    A := 0
    B := 1
L1: A := A + B
    if B ≥ C GOTO L2

    B := B +1
    GOTO L1
L2: write A
    halt
```



P306-2

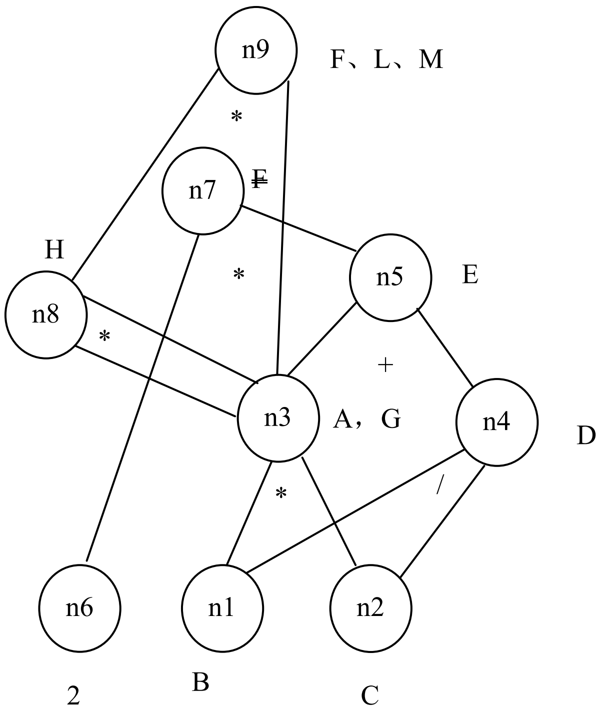
```
    read  A , B
    F := 1
    C := A*A
    D := B*B
    if C < D goto L1
    E := A*A
    F := F+1
    E := E+F
    write E
    halt
L1：E := B*B
    F := F + 2
    E := E + F
    write E
    if E > 100 goto L2
    halt
L2：F := F+1
    goto L1
```



P306-3 基本块：

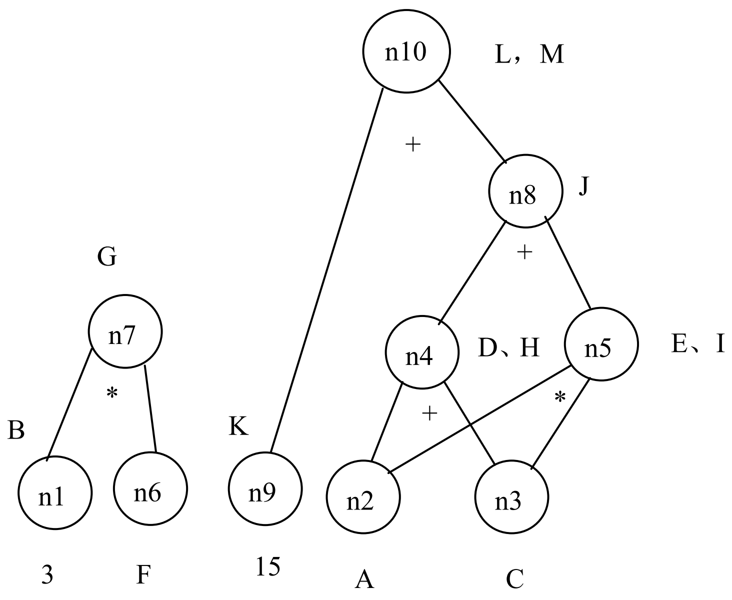
B1： A := B*C
D := B/C
E :=A+D
F :=2*E
G:=B*C
H :=G*G
F:=H*G
L := F
M :=L

B2： B := 3
D := A+C
E :=A*C
G:=B*F
H:=A+C
I:=A*C
J:=H+I
K:= B*5
L:=K+J
M:=L



如果只有 G、L、M 在基本块后还要被引用，则优化为：
G :=B*C
S1 :=G*G
L := S1*G
M := L
（S1 为临时变量）

如果只有 L 在基本块后还要被引用，则优化为：
S1 :=B*C
S2 :=S1*S1
L := S2*S1
（S1、S2 为临时变量）



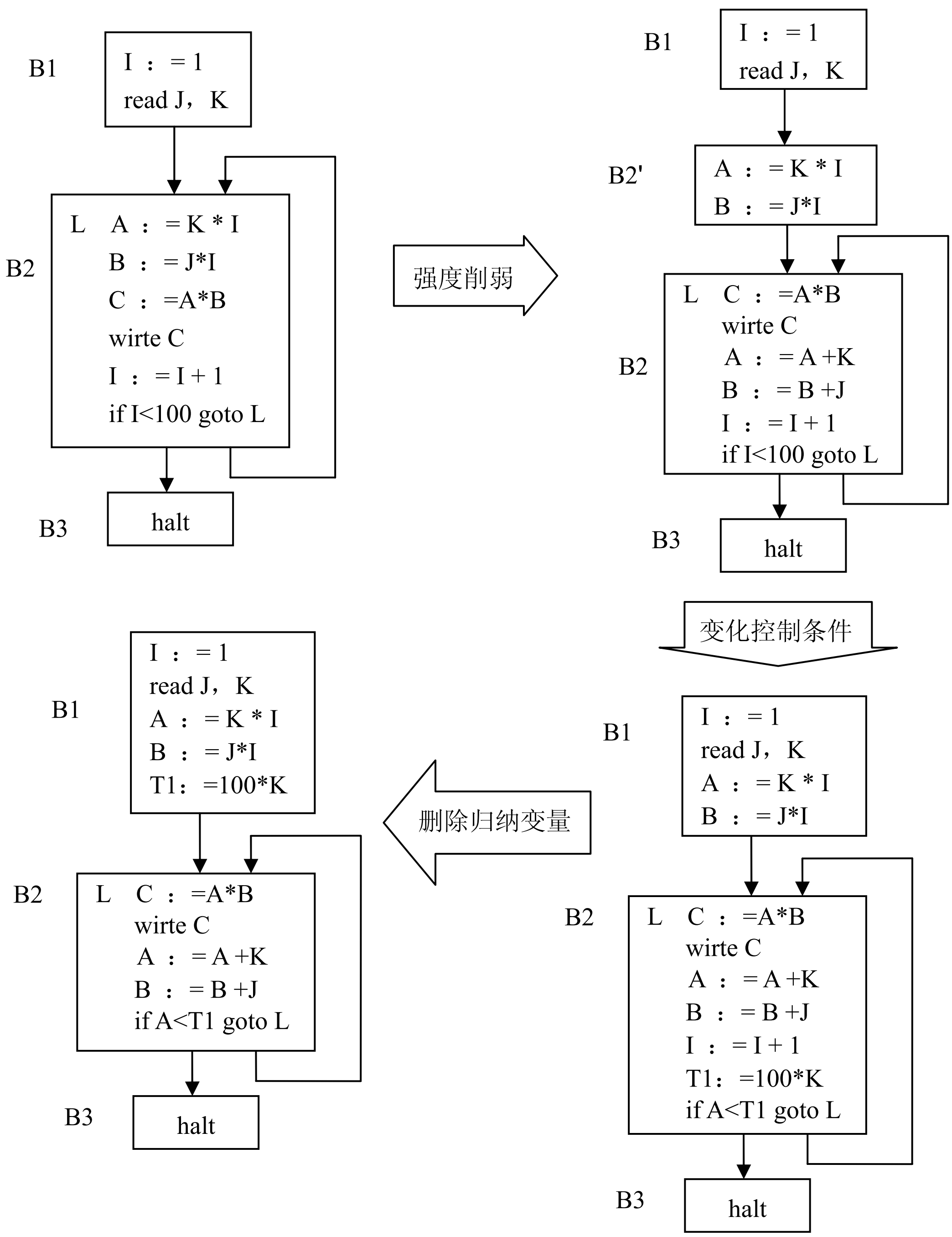
如果只有 G、L、M 在基本块后还要被引用，则优化为：
G :=3*F
S1:=A+C
S2:=A*C
S3:=S1+S2
L := 15 + S3
（S1、S2、S3 为临时变量）

如果只有 L 在基本块后还要被引用，则优化为：
S1:=A+C
S2:=A*C
S3:=S1+S2
L := 15 + S3
（S1、S2、S3 为临时变量）

P307-4 对一下四元式程序，对其中循环进行循环优化：

```
        I ：=1
        read J，K
L:      A ：=K* I
        B ：=J*I
        C ：=A* B
        write C
        I ：=I + 1
        if I < 100 goto L
        halt
```

解答：首先进行基本块划分，画出程序流图：从图中可以看出需要优化的循环块为 B2



P307-5 以下是某程序的最内循环模式对其进行循环优化。

A ：= 0
I ：= 1
L1： B：= J + 1
C：= B + I
A：=C+A
if I = 100 goto L2
I ：= I + 1
goto L1
L2：

解答：首先做出程序流图，然后进行优化：