第四章 文法与语法分析

1. 形如 A-->Aa 的产生式称为左递归的，类似地称 B-->βB 的产生式为右递归的。证明如果一个非终极符既有左递归式又有右递归式，则文法一定有二义性。

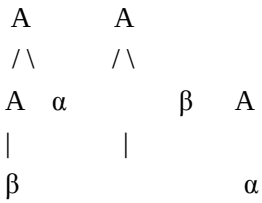
(答案)

设文法的非终极符 A，有产生式

A→Aα|α

A→βA|β

字符串 βα 符合文法的要求，但分析树有两种



因此非终极符既有左递归式，又有右递归式，则一定有二义性。

(关闭)

2. 形如 A-->B 的产生式称为单位产生式，其中 B 是其终级符。证明任何一个含单位产生式的文法均可转换成生式的等价文法。

(答案)

算法：

(1)对任一文法 G1，对 G1 中的每个非终极符 A，令 SpS(A)= {D | A =>* D, D 是 G1 中的非终极符}

(2)令 G1 中所有非单位产生式为 G2 中的产生式。

(3)若在 SpS(A)中有 D，且有 D->x(非单位产生式),则在 G2 中增加产生式:A->x。

(4)删除无用的产生式。

(关闭)

3. A-->λ 产生式为空产生式，证明任何含空产生式的文法均可转换成无空产生式的等价文法（可能只差一个空

(答案)

(1). 构造集合 β,令 β= {A|A→ε 是 G1 的产生式}

(2). 递归构造 β

β=β∪{D|D->x 是 G1 中的产生式，且 x 是集合 β 上的符号串}

直到 β 不再增加新元素
(3). 从 G1 中删去所有空产生式。
(4). 从 G1 中删去只能导出空串的非终极符。
(5). 设 β = {A1, A2, ..., An} , Vn~β={B1, B2, ..., Bm}
若有产生式 A->C1C2...Cp,则 Ci 有三种可能：
Ci 是终极符；Ci 是 β 中的元素；Ci 是非终极符并且不在集合 β 中
如果 Ci 是集合 β 中的元素，则因为删去了所有的空产生式，因此
应该扩充一产生式：A->C1...Ci-1Ci+1...Cp。
重复这个过程直至不出现新的产生式为止。

(关闭)

4. 试给出一个算法，它判定文法中有无非终极符，它不出现于任何句型中，称这种非终极符为不可到达符号

(答案)

算法：
(1)oldV={},newV={S},S 为文法开始符
(2)从 newV 中任取元素 A,oldV=oldV∪{A},newV=newV-{A}
对每个以 A 为左端的产生式,考察其右部的每一个非终极符 B，若 B 不在 oldV 中,newV=newV∪{B}
(3)若 newV 不为空转(2)；
否则 oldV 中即为所有可到达的非终极符，即若非终极符集!=oldV,则有不可到达的非终极符；

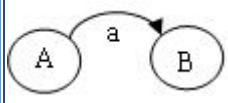
(关闭)

5. 写出正则文法 DFA，DFA 到正则文法的转换文法。

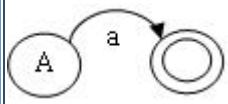
(答案)

Ⅰ. 正则文法→DFA
将正则文法中的非终极符作为 NFA 的状态，将正则文法中的初始非终极符作为 NFA 的初始状态，对于形如 A→a VT 的产生式，在 NFA 中加入一条从状态 A 到终止状态的有向边，并标记为 a. 对于形如 A→aB, A,B VN, a V 式，在 NFA 中加入一条从状态 A 到状态 B 的有向边，并标记为 a. 对文法中每个产生式应用上面的文法,即构造文法对应的 NFA. 随后使用第三章中给出的 NFA 转换成 DFA 的方法进行转换，即得正则文法对应的 NFA.

Ⅱ. DFA→正则文法
DFA 的状态集合 SS.作为正则文法的非终极符集; DFA 的初始状态 S0 作为正则文法的初始符集 S; DFA 的符号正则文法的终极符集 VT,正则文法的产生式集 P,如下构造:若 DFA 中有



(B 不是终止状态), 则 P 中加入产生式 A→aB; 若 DFA 中有



,则 P 中加入产生式 A→a;这样即构造出正则文法.

(关闭)

6. 下列那些文法是 LL(1)文法?

a.

S-->Abc
A-->a
A-->λ
B-->b
B-->λ

b.

S-->Ab
A-->a
A-->B
A-->λ
B-->b
B-->λ

c.

S-->ABBA
A-->a
A-->λ
B-->b
B-->λ

d.

S-->aSe
S-->B
B-->bBe
B-->C
C-->cCc
D-->d

(答案)

LL(1)文法的条件：对于任意非终极符 A,其任意两个产生式 A-> α 和 A-> β 都要满足下面条件：

predict(A-> α)∩predict(A-> β)=Φ;

(a) 因为非终极符 B 不可到达，所以最后两个产生式可以去掉。对具有相同左部的产生式 A->a 和 A-> λ , 它们的 Predict 集的交集为空：

predict(A->a)={a};Predict(A-> λ)={b}

故 a)所示文法是 LL(1)文法。

(b) 1)A,B 可导出空

2)first(A)={a,b,λ},first(B)={b,λ},first(S)={a,b}

follow(A)={b}, follow(B)={b}, follow(S)={#}

predict(S->Ab)={a,b},

predict(A->a)={a},predict(A->B)={b},predict(A->λ)={b}

predict(B->b)={b},predict(B->λ)={b}

因为 predict(A->B)∩predict(A->λ)={b}!=Φ

predict(B->b)∩predict(B->λ)={b}!=Φ

所以不是 LL(1)文法

(c) 1)S,A,B 可导出空

2)first(S)={a,b},first(A)={a,λ},first(B)={b,λ}

follow(S) ={#},follow(A) ={a,b,#},follow(B) ={a,b,#}
predict(S->ABBA)={a,b,#}
predict(A->a)={a},predict(A->λ)={a,b,#} (*)
predict(B->b)={b},predict(B->λ)={a,b,#} (**)

由(*)和(**)两行知，不是 LL(1)文法.

(d) 1)没有非终极符可导出空

2)first(C)={c,d},first(B)={b,c,c},first(S)={a,b,c,d}
follow(S)={e,#},follow(B)={e,#},follow(C)={c,e,#}
predict(S->aSe)={a},predict(S->B)={b,c,d}
predict(B->bBe)={b},predict(B->C)={c,d}
predict(C->cCc)={c},predict(C->d)={d}

因为满足 LL(1)文法的条件，所以是 LL(1)文法.

(关闭)

7. 对下列文法构造 LL(1)分析表：

E -->-E

E -->(E)
E -->Var Etail
Etail -->-E
Etail -->λ
Var -->id Vtail
Vtail -->(E)
Vtail -->λ

(答案)

First(E)={-, (, id} Follow(E)={＃,)}
First(Etail)={-, λ} Follow(Etail)={＃,)}
First(Var)={ id } Follow(Var)={-,＃,)}
First(Vtail)={ (, λ} Follow(Vtail){-,＃,)}
[1] predict (E→-E)={¨ }
[2] predict (E→(E))={(}
[3] predict (E→Var Etail)={id}
[4] predict (Etail→-E)={¨ }
[5] predict (Etail→λ)={＃,)}
[6] predict (Var→id Vtail)={id}
[7] predict (Vtail→(E))={(}
[8] predict (Vtail→λ)={¨,＃,)}
分析表:

	-	(	)	id	
E	1	2		3	
Etail	4		5		
Var				6	
Vtail	8	7	8		

(关闭)

8. 写出上述 LL(1)分析器分析 i--i(i)的过程，其中 i 是标志符。

(答案)

步骤	符 号 栈	输 入 流	动 作
1	E #	i--i(i)#	predict(3)
2	Var Etail#	i--i(i)#	predict(6)
3	id Vtail Etail#	i--i(i)#	Match
4	Vtail Etail#	--i(i)#	predict(8)
5	Etail#	--i(i)#	predict(4)

6	- E #	--i(i)#	Match
7	E #	-i(i)#	predict(1)
8	- E #	-i(i)#	Match
9	E #	i(i)#	predict(3)
10	Var Etail#	i(i)#	predict(6)
11	id Vtail Etail#	i(i)#	Match
12	Vtail Etail#	(i)#	predict(7)
13	(E) Etail#	(i)#	Match
14	E) Etail#	i)#	predict(3)
15	Var Etail) Etail#	i)#	predict(6)
16	id Vtail Etail) Etail#	i)#	Match
17	Vtail Etail) Etail#	)#	predict(8)
18	Etail) Etail#	)#	predict(5)
19	) Etail#	)#	Match
20	Etail#	#	predict(5)
21	#	#	接受

(关闭)

9. 对下列文法转换为 LL(1)文法：

DL -->DL；D
DL -->D
D -->idL: Type
idL -->id
idL -->idL, id
Type -->Stype
Type -->array (StypeL) of Type
Sttype -->id
STtype -->Bound..Bound
Bound -->Sign IntLiteral
Bound -->id
Sign -->+
Sign -->-
STypeL -->STypeL, Stype
STypeL -->Stype

(答案)

First(DL)={id}
First(D)={id}
First(idL)={id}
First(Type)={id, array, +, -}
First(SType)={id, +, -}
First(Bound)= {id, +, -}
First(Sign)={+, -}
First(SType L)= {id, +, -}
[1] DL→DL; D
[2] DL→D
[3] D→idL:Type
[4] idL→id
[5] idL→idL,id
[6] Type→Stype
[7] Type→array (STypeL) of Type
[8] Stype→id
[9] Stype→Bound
¡Bound
[10] Bound→Sign
IntLiteral
[11] Bound→id
[12] Sign→ +
[13] Sign→ -
[14] STypeL→STypeL, Stype
[15] STypeL→Stype
[1][2]的 Predict 集相同
[4][5]的 Predict 集相同
[8][9] 的 Predict 集不为空
[14][15] 的 Predict 集不为空
改为下面的样子即为 LL（1）文法
[1][2]去掉
[4][5] 去掉
[8][9] 去掉
[14][15] 去掉
添加： DL→D DLtail
DLtail→ ； DLtail|λ
idL→id idtail
idtail→， idLtail|λ
STypeL→Stype StypeLtail
STypeLtail→, STypeLtail|λ
SType→id StypeLtail |Sign IntLiteral..Bound
STypetail→..Bound|λ

(关闭)

10．称一个文法是 greibach 规范型(GNF)文法，如果每个产生式具有形式 A-->tβ，其中 t 是终极符，β 是任何符号是任一不含空句子的文法，试给出 G 到 GNF 的转换算法。

(答案)

提示：

因为该文法没有空句子，所以对文法开始符 S 不会导出空串，将可到达的非终极符保存起来，接下来分析合中的每个元素（设 A）为左部的产生式：

(1).若是空产生式，即 A->λ,则可利用消除空产生式算法，消除该空产生式；

(2).若第一个字符是终极符，A->ax,则符合，将其后的非终极符填入可到达的非终极符集合里；

(3).若第一个字符是非终极符 B，A->BX,将 B 加入可到达非终极符集合，并且 first(B)={b1,b2,...,bn}， 则将换为 A->b1X |b2X |... | bnX;若 first(B)中还包含 λ，则还需要处理产生式 A->X，返回(1).

(4).将 A 从可到达非终极符集中删除，取下一个元素，继续上面的步骤，直至该集合为空止。

(关闭)

11. 证明下列问题：

(1) 左递归文法不是 LL(1)文法。

(2) 证明 LL(1)文法是无二义性文法。

(3) 若文法 G 没有空产生式，且每个非终极符的产生式均以不同的终极符打头，则 G 定是 LL(1)文法。

(答案)

（1）证明： 先考虑直接左递归的情况

A→Aα

A→β

其中 α，β 为任意语法符号串，即得如下关系式

Predict（A→Aα） First（Aα） First（β）

Predict（A→B） First（β）

因此，A→Aα 和 A→B 的 Predict 集有交，即不满足 LL（1）文法条件。

再考虑间接左递归情况

A→Bα|β

B→Aγ|b

其中 α，β，γ，b 为任意语法符号串

Predict（A→Bα） First（Bα） First（Aγ） First（β）

Predict（A→β） First（β）

因此，A→Bα 和 A→β 的 Predict 集有交，即不满足 LL（1）文法条件。

（2）证明：LL（1）文法中任意两个产生式 Pi，Pj，（Pi，Pj 具有相同的左部非终极符）

Predict(Pi) ∩Predict(Pj) 为空

设 Pi: A→α1α2…αn

Pj: A→α11α21…αm1

(A∈VN, α1α2…αn, α11α21…αm1∈ VN∪VT)

因为 Predict(Pi) ∩Predict(Pj) 为空，因此 Pi，Pj 中的 A 经一步推导，

最左的终极符肯定不同，因此，对于一个字符串，不可能有两种方法推导。出它。

（3）证明：文法对于每个非终极符的产生式均以不同的终极符打头，因此每个产生式的 Predict 集均为打头的极符，所以 Predict 集是肯定不相交的，故，G 是 LL（1）文法。

(关闭)

12. 为下面文法构造 LR 状态机，并给出相应的 Goto 表。

Prog -->Block $

Block -->begin SL end

SL -->SL; S

SL -->S

S -->Block

S -->V:=E

V -->id

V -->id[E]

E -->E+T

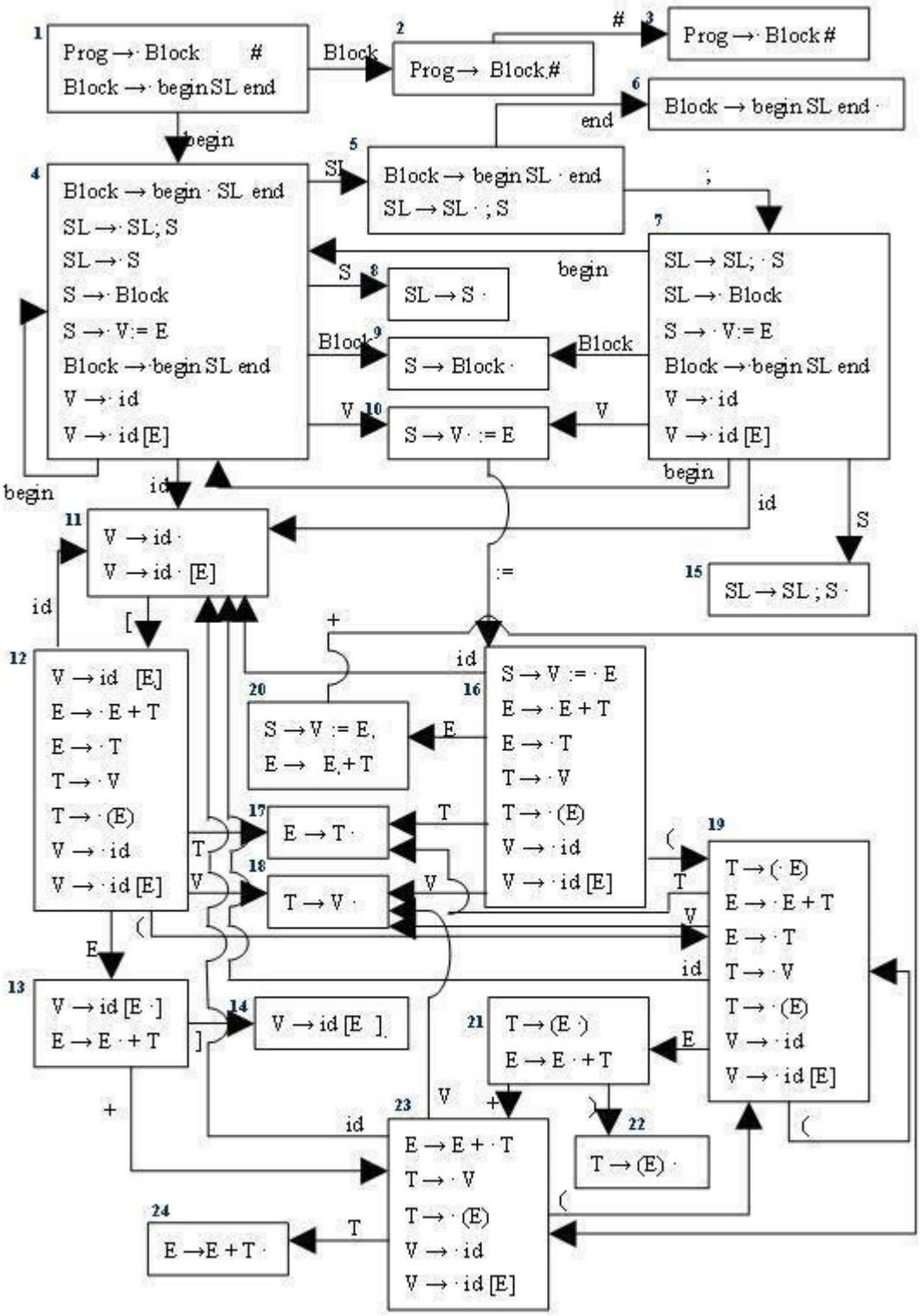
E -->T

T -->V

T -->(E)

(答案)

LR 状态机为：



Goto 表为：

	begin	end	id	:=	;	[	]	(	)	+	Block	SL	S	V	E	T
1	4										2					
2																
3																
4	4		11								9	5	8	10		
5		6			7											
6																
7	4		11								9		15	10		
8																
9																
10				16												
11						12										
12			11					19						18	13	17
13							14			23						
14																
15																
16			11					19						18	20	17
17																
18																
19			11					19						18	21	17
20										23						
21									22	23						
22																
23														18		24
24																

(关闭)

13. 以下文法中哪些不是 LR(0)？ 为什么？

(a) Q -->SL $

SL -->SL； S

SL -->S

S -->null

(b) Q -->SL $

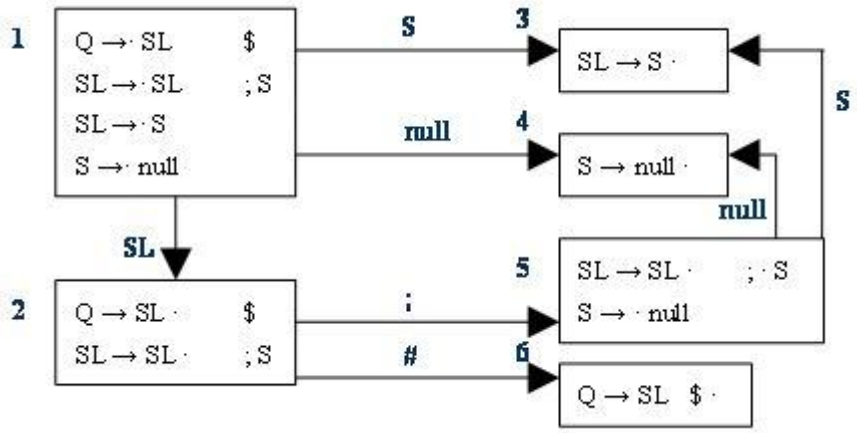
SL -->S； SL

SL -->S

SL -->null
(c) Q -->SL $
SL -->SL；SL
SL -->S
S -->null
(d) Q -->SL $
SL -->null SLtail
SLtail -->
SLtail -->；SL

(答案)

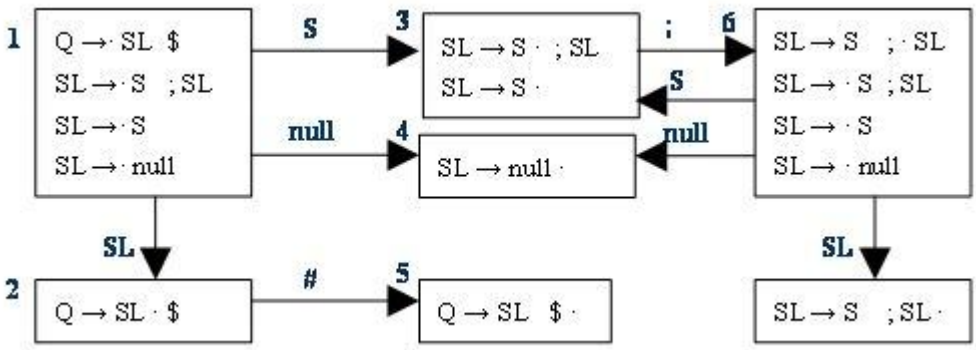
a)



状态	1	2	3	4	5	
动作	S	S	r	r	S	

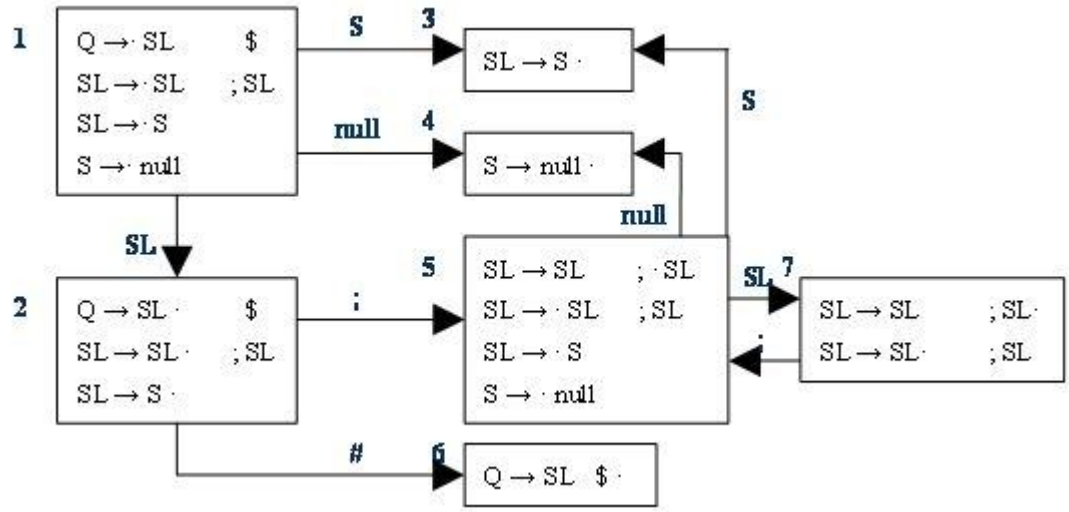
不存在冲突，故（a）文法是 LR(0)

b)



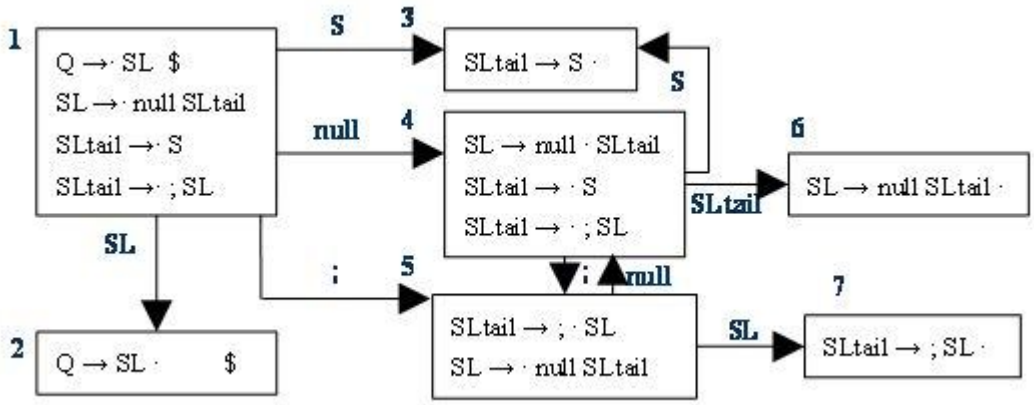
状态 3 存在移入 - 归约冲突，故（b）文法不是 LR(0)。

c)



状态 2，7 存在移入 - 归约冲突，故（c）文法不是 LR(0)

d)



状态	1	2	3	4	5	6	
动作	S	r	S	r	S	r	

不存在冲突，故（d）文法是 LR(0)。

(关闭)

14. 证明对应 LL(1)文法的 LR 状态机的每个状态的项目集恰有一个基本项目。

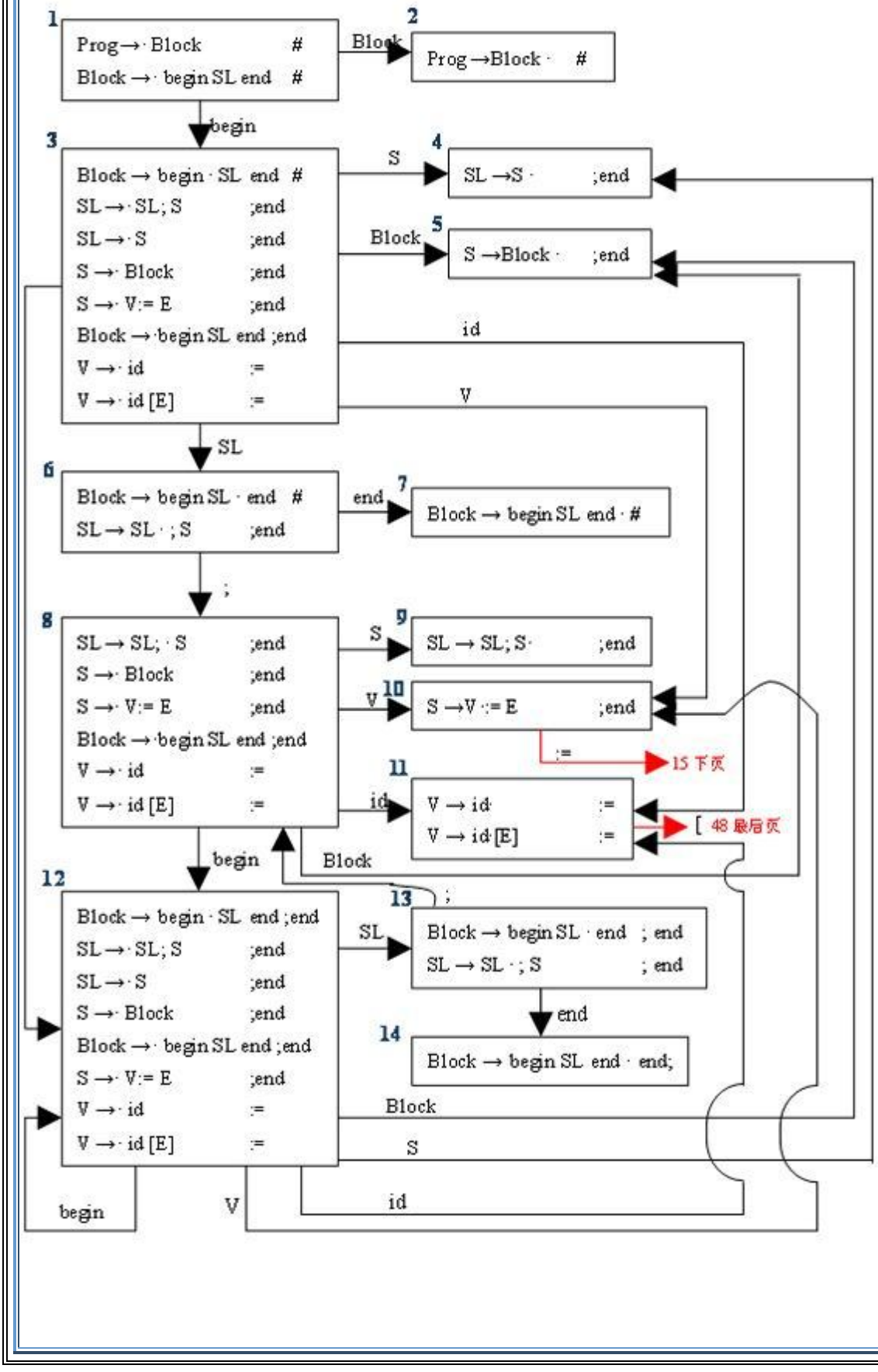
(答案)

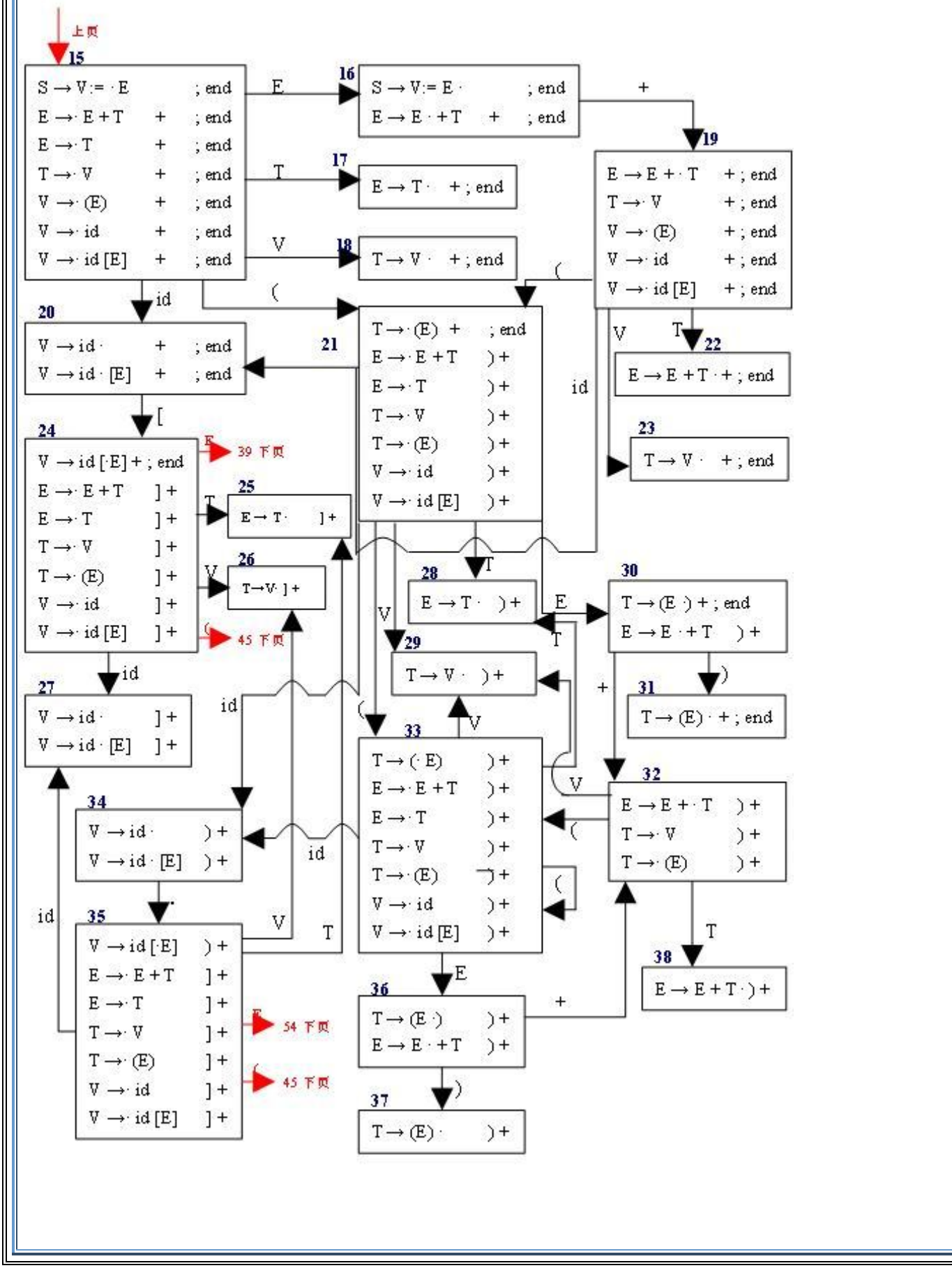
(无)

(关闭)

15. 为 12 题中给出的文法构造一个 LR(1)状态机。

(答案)





(关闭)

16. 下面的文法中哪一些是 LR(1)？LALR(1)？SLR(1)？说明你的理由。

(a)
S-->id:=E;
E-->E+P
E-->P
P-->id
P-->id:=E

(b)
S-->id:=A;
A-->id:=A
A-->E
E-->E+P
E-->P
P-->id
p-->(A)

(c)
S-->id:=A;
A-->id:=A
A-->E
E-->E+P
E-->P
P-->P+
P-->id
P-->(A)

(d)
S-->id:=A;
A-->PreE
pre-->Pre id:=
Pre--> ε
E-->E+P
E-->P
P-->id
P-->(A)

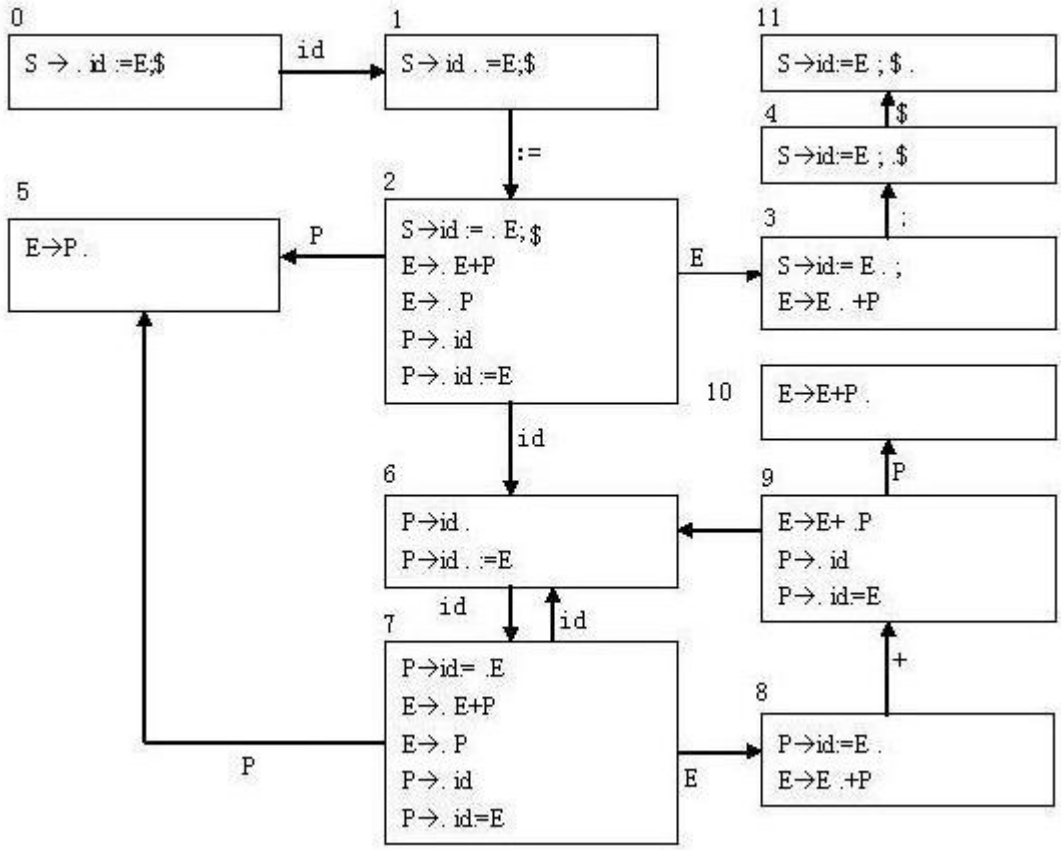
(e)
S-->id:=A;
A-->Pre E
Pre-->id:=Pre
Pre-->
E-->E+P
E-->P
P-->id
P-->(A)

(f)
S-->id:=A;
A-->id:=A
A-->E
E-->E+P
E-->P
P-->id
P-->(A;A)
p-->(V,V)
P-->A,A
P-->V;V
v-->id

(g)
S-->id:=A;
A-->id:=A
A-->E
E-->E+P
E-->P
P-->id
P-->(id;id)
P-->(A)

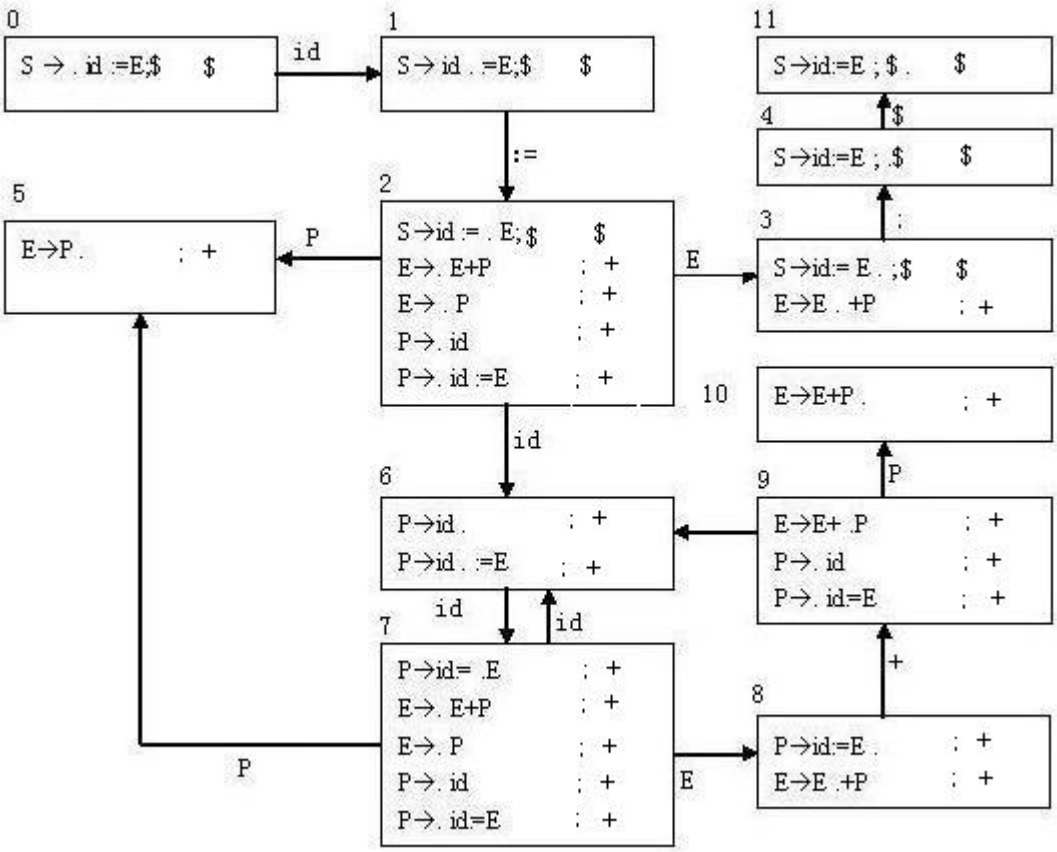
(答案)

(a)先画出 LR(0)状态机，如图：



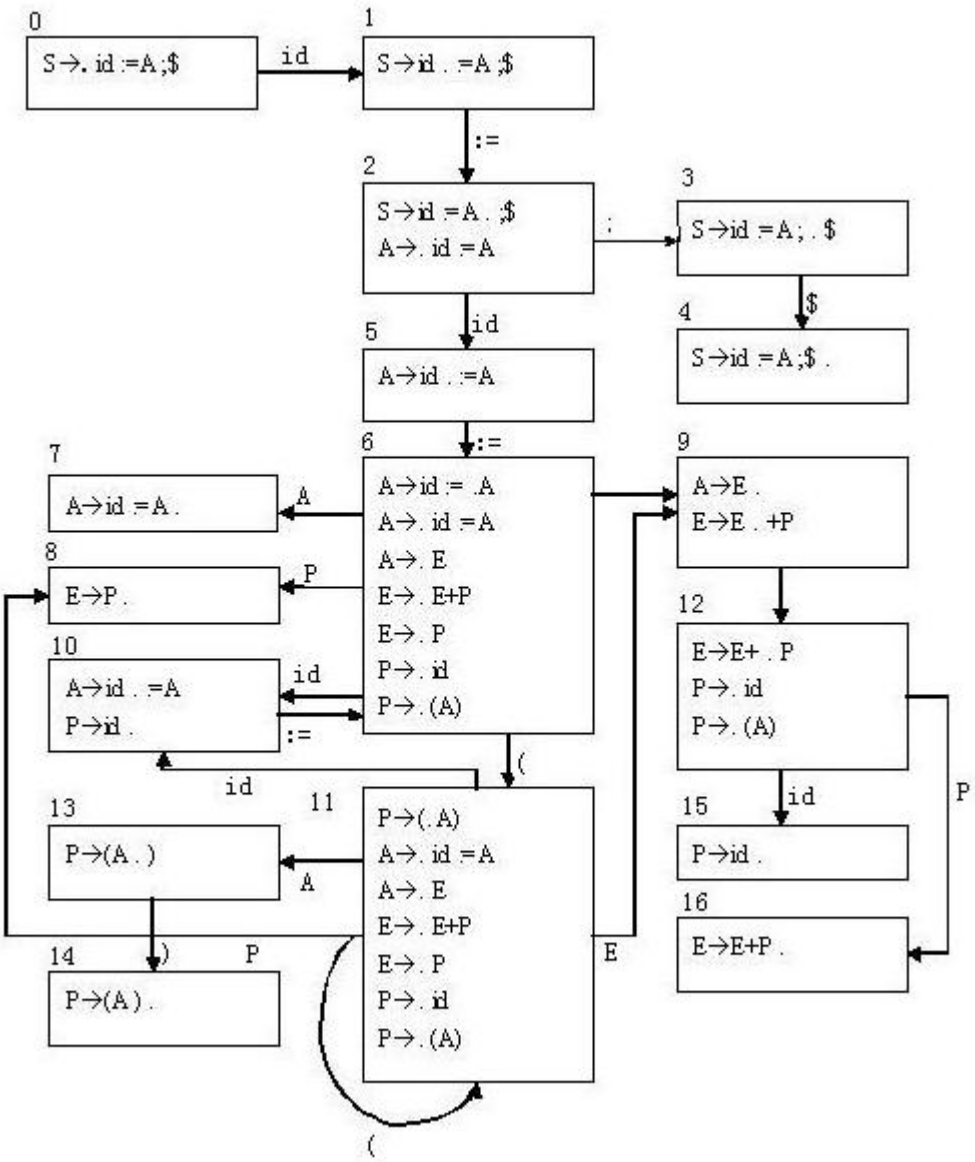
[1]因为状态(6)和状态(8)有移入和规约冲突，所以不是 LR(0)文法
[2]因为 follow(P)={+,;},所以看 follow 集,状态(8)中仍有冲突，所以也不是 SLR(1)文法

然后画出 LR(1)状态机,如图:



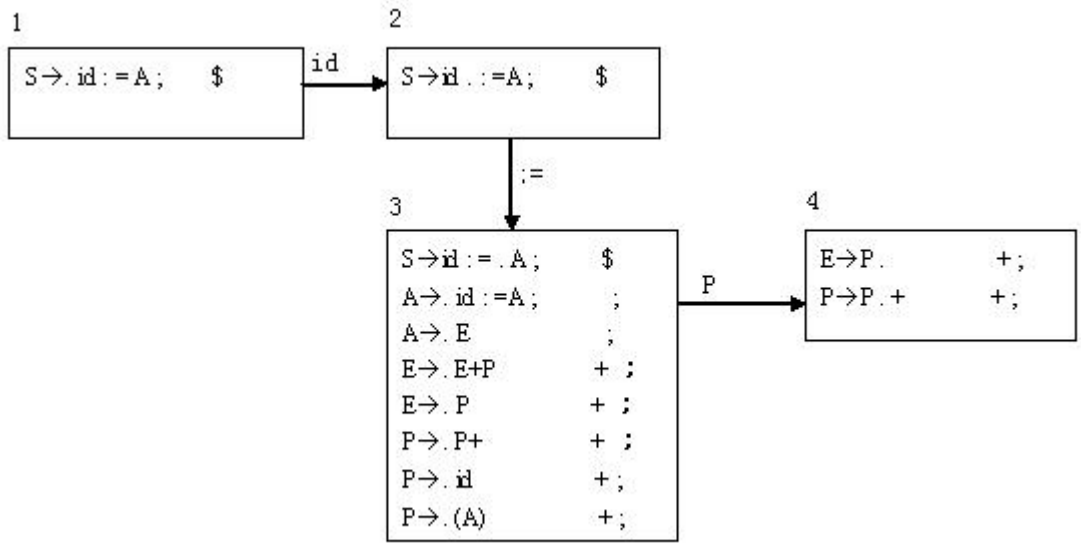
因为状态(8)中仍然有冲突，所以也不是 LR(1)文法；

(b)画出 LR(0)状态机，如图:



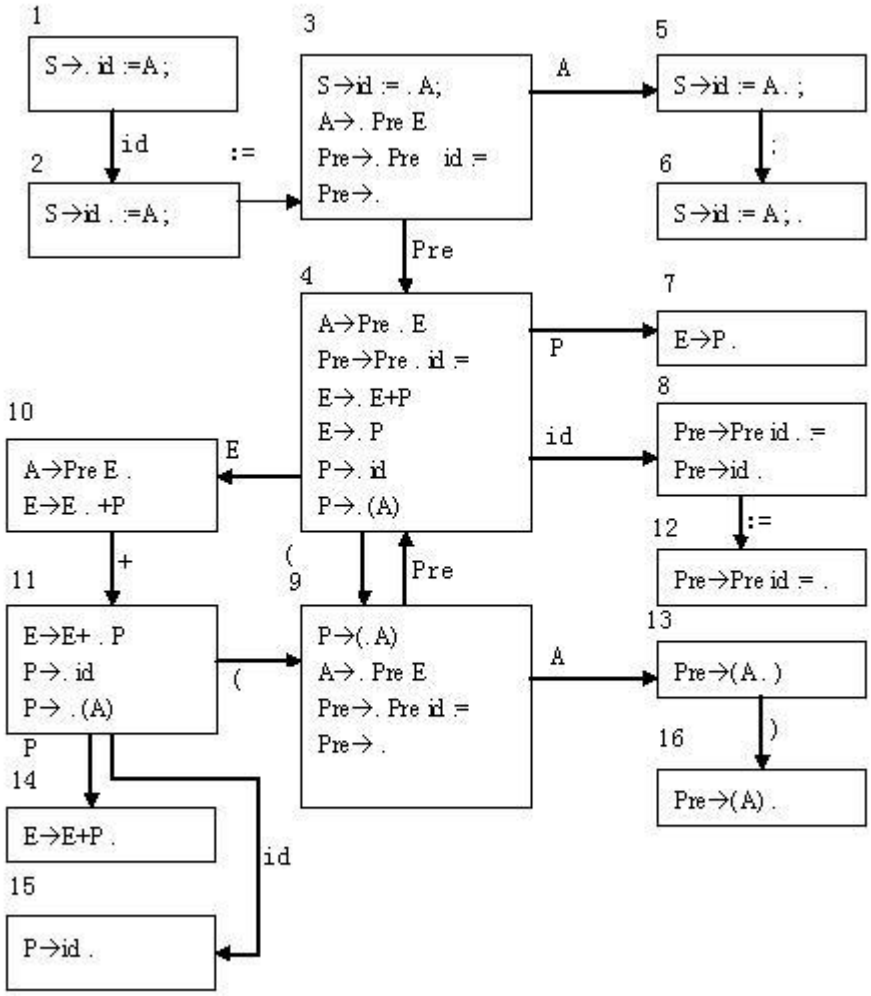
因为状态(5)和状态(7)有移入规约冲突，所以不是 LR(0)文法
但因为 follow(A)={;,)},follow(P)={+,),;},所以考虑 follow 集即可消除冲突，所以是 SLR(1)文法

(c)不是 LR(0)和 SLR(1)文法，因为状态机中必存在一个状态含项目 E->P.和 P->P.+,此时有移入规约冲突
即使看 follow 集，由于 follow(E)中包含+,不能消除冲突；
　下图是 LR(1)状态机的部分状态转换图：



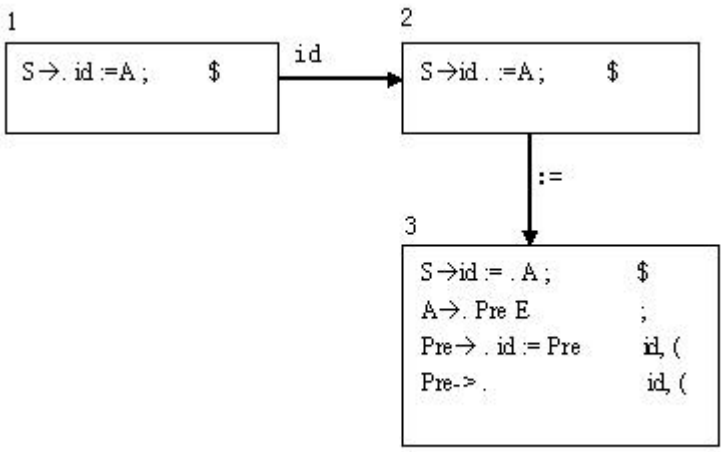
状态(4)中有移入规约冲突，所以不是 LR(1)文法.

(d)先画出 LR(0)状态机；



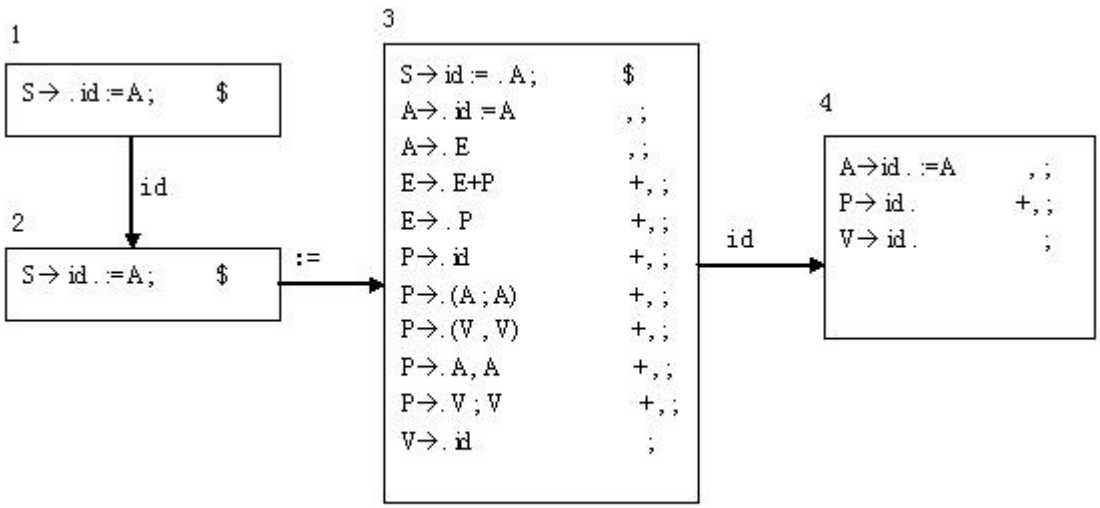
LR(0)状态机中状态(8)和状态(10)中有移入规约冲突，但考虑 follow 集时， follow(Pre)={id,(}
follow(A)={;,)}可以消除冲突，所以是 SLR(1)文法.

(e)下面是 LR(1)状态机的一部分；



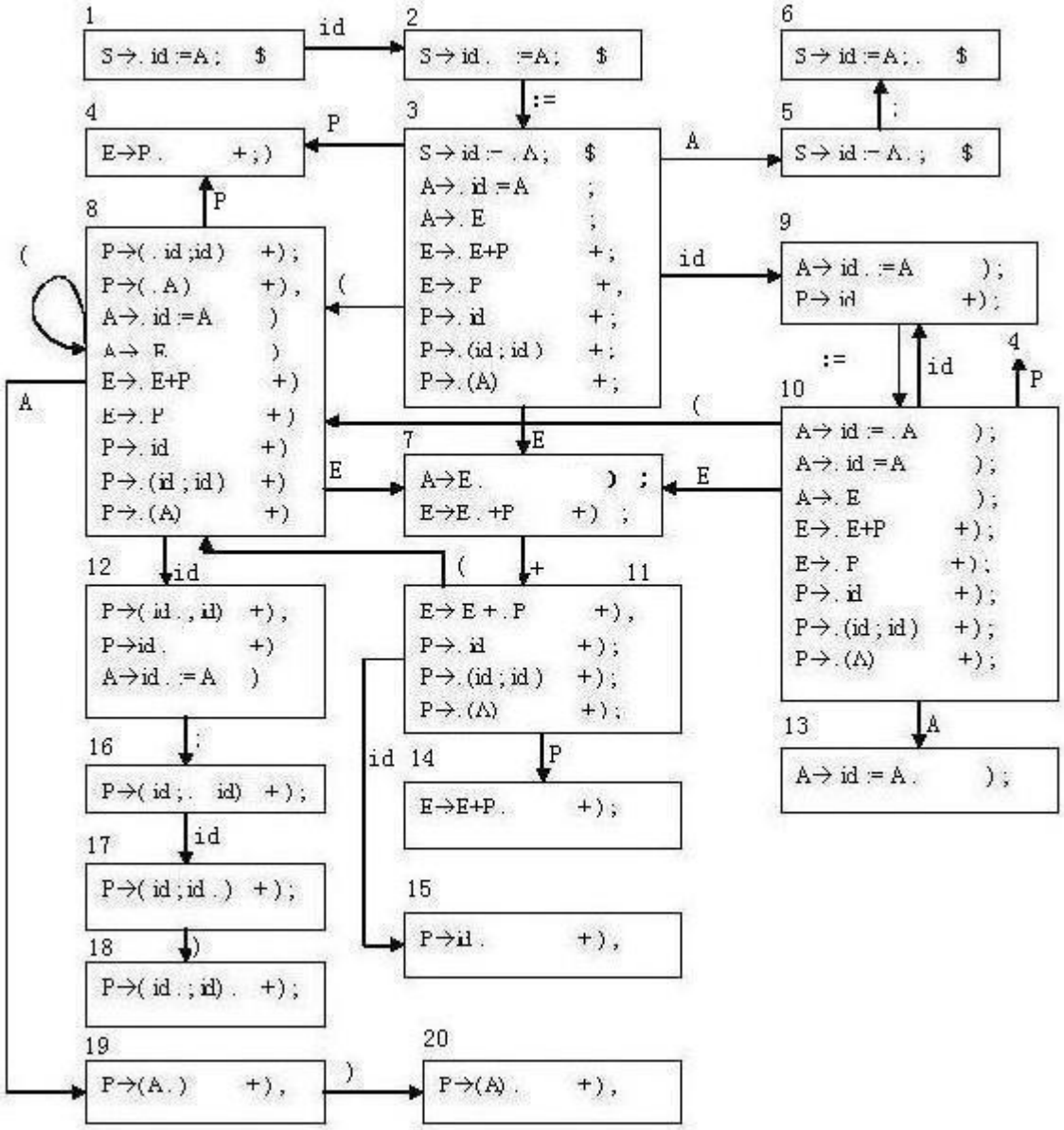
因为状态(3)中存在移入规约冲突(遇见 id 时)所以不是 LR(1)文法.

(f)下面是 LR(1)状态机的一部分:



因为状态(4)中存在规约规约冲突(遇见；时)所以不是 LR(1)文法

(g)下面是 LALR(1)状态机:



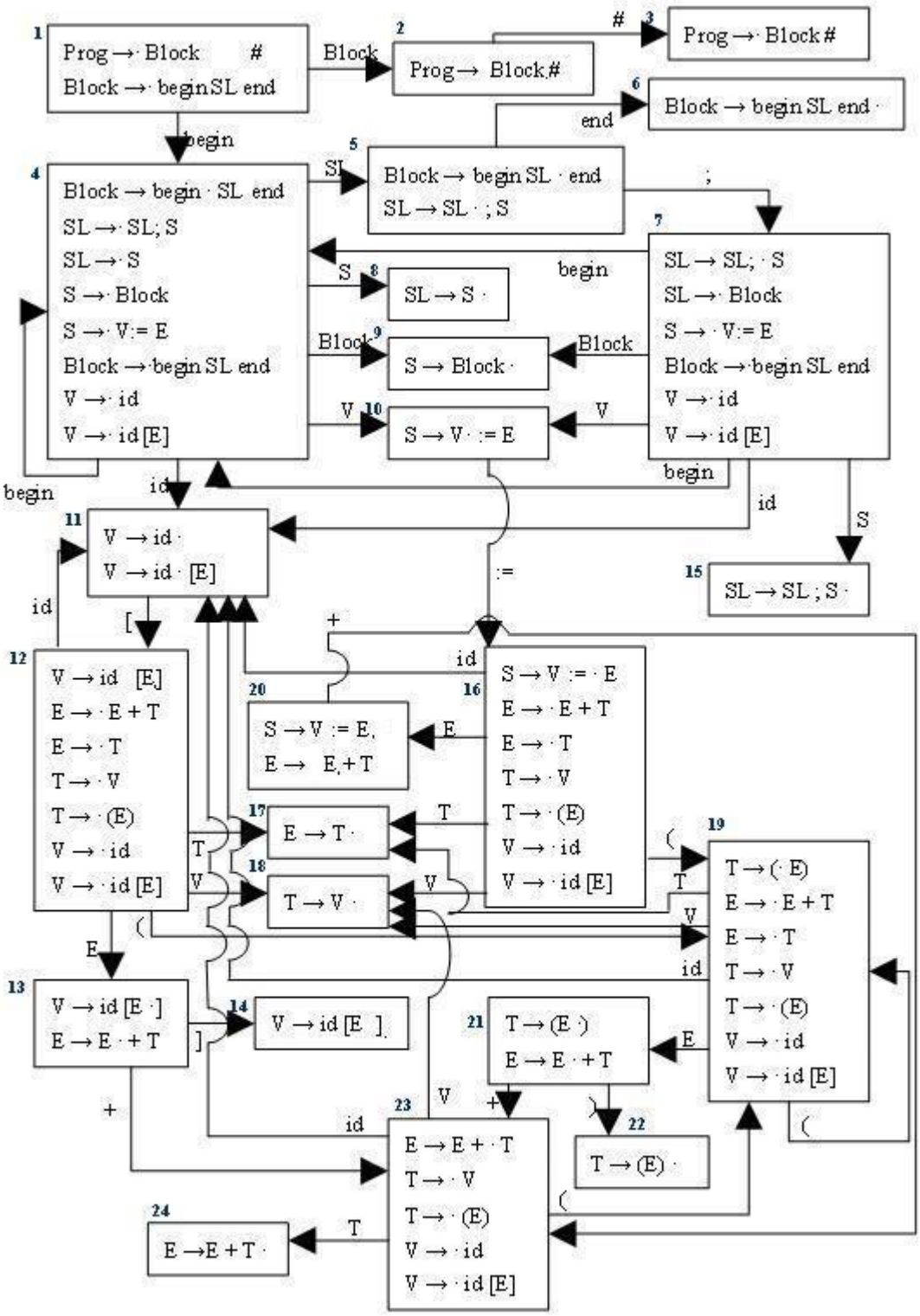
状态机中没有冲突的状态，所以是 LALR(1)文法；

它不是 SLR(1)文法，因为在 LR(0)状态机中状态(7)中存在移入规约冲突（遇见+时），考虑 follow 集也不能消除

(关闭)

17. 为 12 题中的文法构造 SLR(1)之 ACTION 表。

(答案)



First(Prog)={begin} Follow(Prog)={#}

First(Block)={begin} Follow(Block)={#, ;, end}
First(SL)= {begin, id} Follow(SL)={;, end}
First(S)={begin, id} Follow(S)= {;, end}
First(V)={id} Follow(V)={:=, ;, end,], +,)}
First(E)={(, id} Follow(E)={;, end,], +,)}
First(T)={id, (} Follow(T)= {;, end,], +,)}

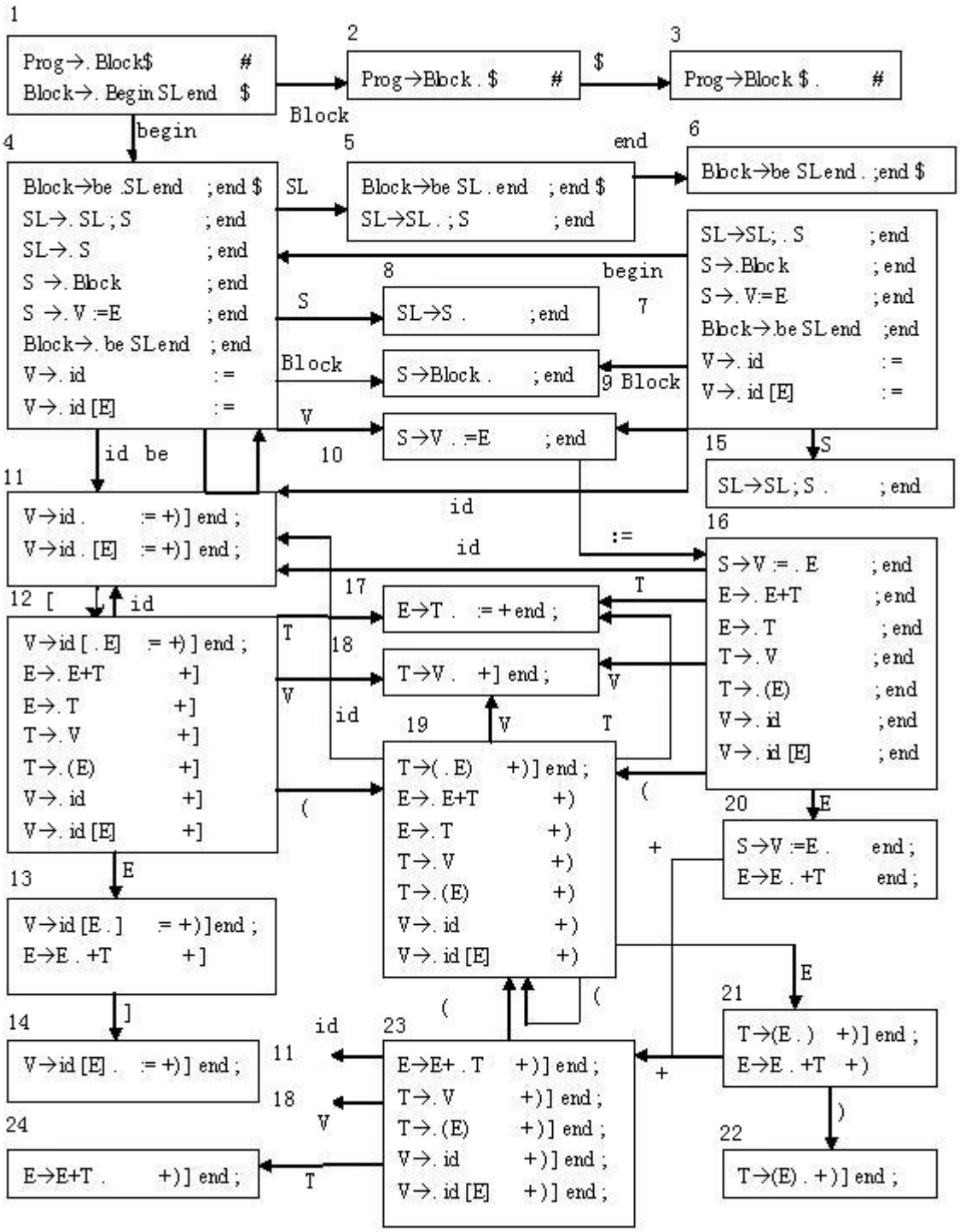
State	;	[	end	:=	]	+	)	#	begin	(
1									S2	
2								Succeed		
3										
4									S4	
5	S7		S6							
6	R2		R2					R2		
7									S4	
8	R4		R4							
9	R5		R5							
10				S16						
11		S12	R7	R7	R7	R7	R7			
12										S19
13					S14	S23				
14	R8		R8	R8	R8	R8	R8			
15	R3		R3							
16										
17	R10		R10		R10	R10	R10			
18	R11		R11		R11	R11	R11			
19									S19	
20	R6		R6			S23				
21						S23	S22			
22	R12		R12		R12	R12	R12			
23										S19
24	R9		R9		R9	R9	R9			

(关闭)

18. 以 15 题中为 12 题文法构造的 LR(1)状态机为基础，合并同心项，即合并有共同核心的项目，建立 LALR(1

(答案)

下图即为所求 LALR(1)状态机(其中有几处 begin 用 be 代替)：



(关闭)

19. 从为 12 题构造的 LR(0)状态机出发，利用向前看符号的传播与生成算法，确定 LALR(1)向前看符号集，并通过合并的方法产生的结果进行比较

(答案)

(无)

(关闭)

20. 证明任何无 λ-产生式的 LL(1)文法都是 LR(0)文法。

(答案)

(无)

(关闭)

21. 证明存在不是 LL(1)的 LR(0)文法，SLR 文法和 LALR(1)文法。

(答案)

考虑文法 G： M → aAc

A → bB | ba

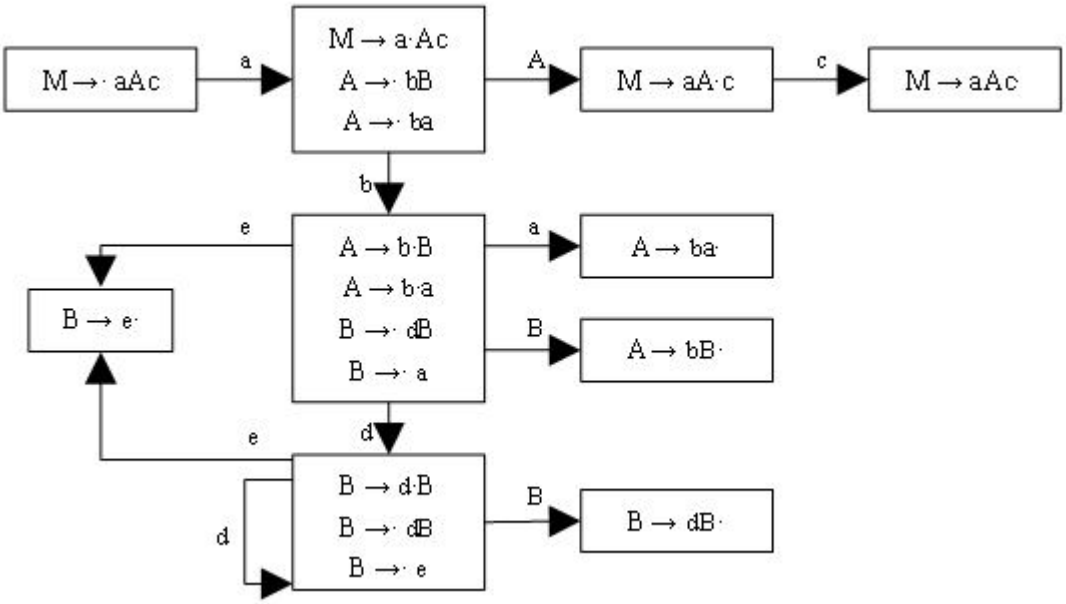
B → dB | e

First(M) = {a} Predict(M → aAc) = {a} Predict(B → dB) = {d}

First(A) = {b} Predict(A → bB) = {b} Predict(B → e) = {e}

First(B) = {d, e} Predict(A → ba) = {b}

由于 Predict(A → bB) ∩ Predict(A → ba) = {b} ≠ 空集，故文法 G 不是 LL(1)文法。



文法 G 不存在移入 ¨ 归约冲突，故文法 G 是 LR(0)文法。

考虑文法 G_j ： M → T

A → T*F | F

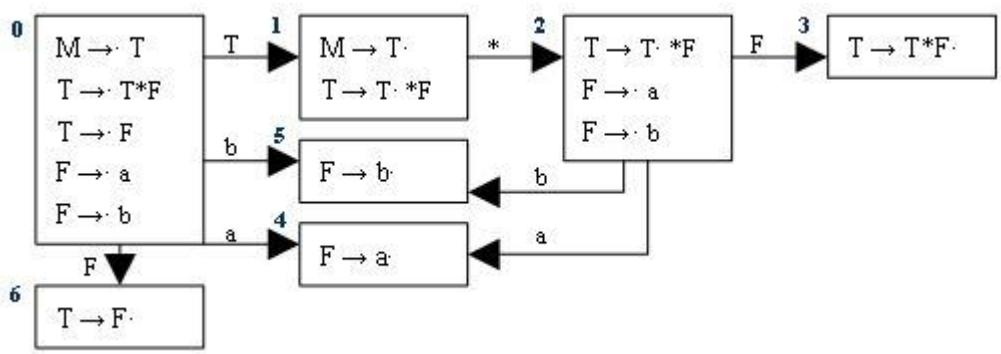
F → a | b

Predict(M → T) = {a, b} Predict(F → a) = {a}

Predict(T → T*F) = {a, b} Predict(F → b) = {b}

Predict(T → F) = {a, b}

由于 Predict(T → T*F) ∩ Predict(T → F) = {a, b} ≠ 空集，故文法 G_j 不是 LL(1)文法



	a	b	*	#	T	
0	S4	S5			1	
1			S2	OK		
2	S4	S5				
3			r2	r2		
4			r4	r4		
5			r5	r5		
6			r3	r3		

从状态 1 可以看出文法 G¡ 不是 LR(0)文法，但却是 SLR(1)文法。

(关闭)

22. 证明所有的 LL(1)文法都是 LR(1)文法。

(答案)

(无)

(关闭)

<<上一章◎ ◎回页首◎ ◎下一章>>