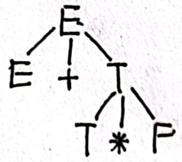
编译原理第5章作业

1. 解：∵ E ⇒ E+T ⇒ E+T*F；所以 E+T*F 是文法G1的一个句型.

画出该句型的语法分析树如下：



所有短语：E+T*F；T*F

直接短语：T*F

句柄：T*F

2. 解：(1) 最左推导：

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

最右推导：

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

(2)

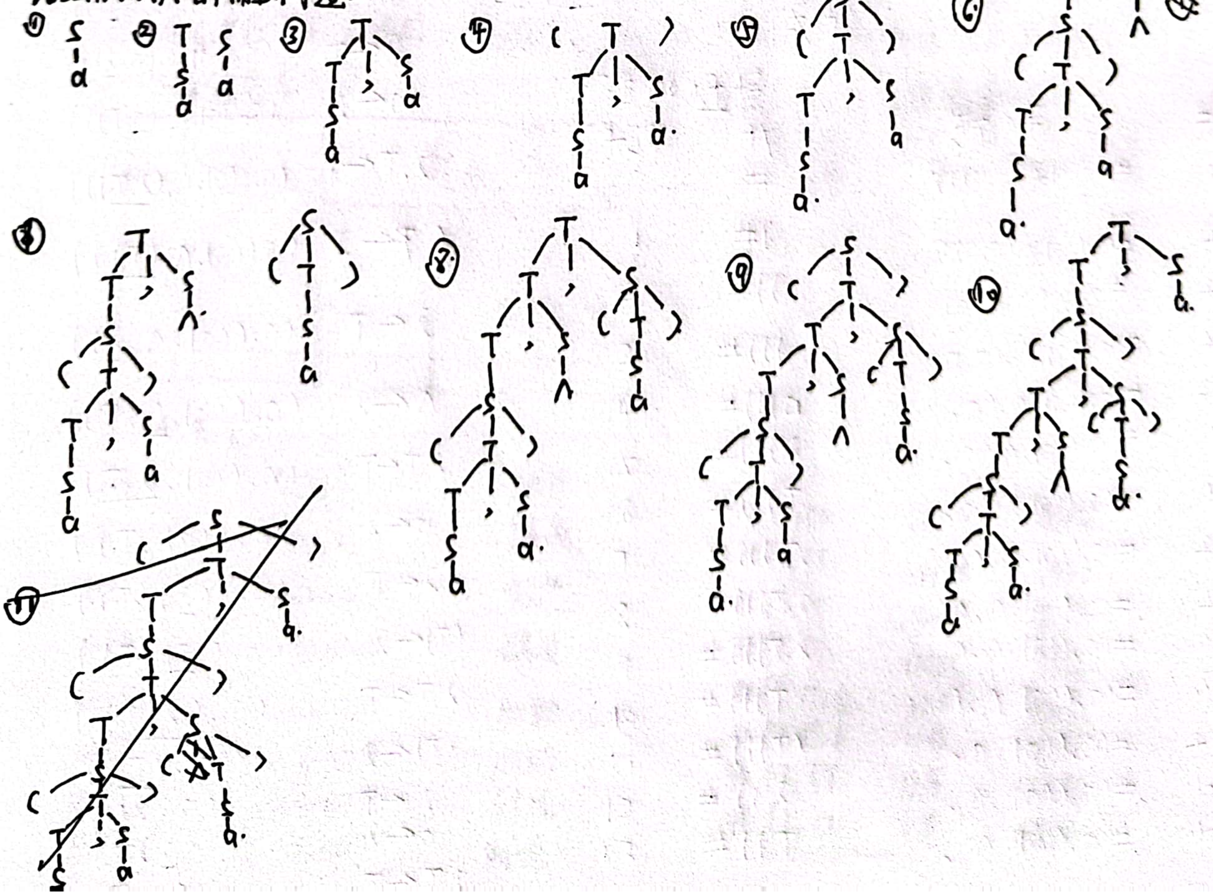
规范句型	归约规则
(((a,a),∧,(a)),a)	S→a
(((S,a),∧,(a)),a)	T→S
(((T,a),∧,(a)),a)	S→a
(((T,S),∧,(a)),a)	T→T,S
(((T),∧,(a)),a)	S→(T)
((S,∧,(a)),a)	T→S
((T,∧,(a)),a)	S→∧
((T,S,(a)),a)	T→T,S
((T,(a)),a)	S→a
((T,(S)),a)	T→S
((T,(T)),a)	S→(T)
((T,S),a)	T→T,S
((T),a)	S→(T)
(S, a)	T→S
(T, a)	S→a
(T,S)	T→T,S
(T)	S→(T)
S	归约成功

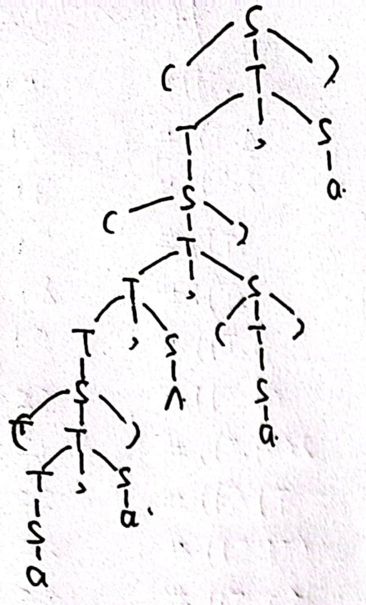
移进归约过程.

步骤	栈	输入串	动作
0	#	(((a,a),∧,(a)),a)#	准备
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

步骤	栈	输入串	动作
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)#	~~移进~~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
35	#S	#	接受

语法分析树自下而上构造:





解 (1) FirstVT(S) = {a, ∧, (}，LastVT(S) = {a, ∧,)}

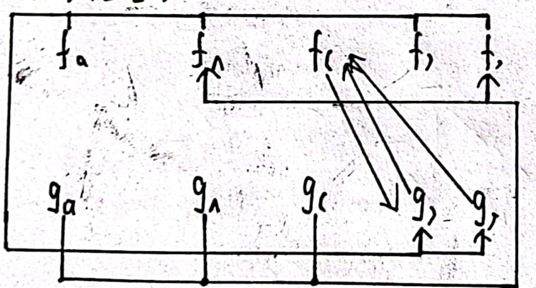
FirstVT(T) = {a, ∧, (, ,}，LastVT(T) = {a, ∧,), ,}

(2) 根据优先关系表的构造过程作出表如下:

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

根据 G_2 的文法产生式任一产生式右部含两个相邻的非终结符，所以是算符文法，又因为在 G_2 中任何终结符对至多只满足 <, >, = 三关系之一，所以是算符优先文法。

(3) 根据优先表作图.



得到 G_2 的优先函数

	a	∧	(	)	,
f	4	4	2	4	4
g	5	5	5	2	3

(4) 算符优先分析.

栈	字符串	动作
#	(a,(a,a))#	预备
#(	a,(a,a))#	移进
#(a	,(a,a))#	移进
#(S	,(a,a))#	归约
#(T	,(a,a))#	归约
#(T,	(a,a))#	移进
#(T,(	a,a))#	移进
#(T,(a	,a))#	移进
#(T,(S	,a))#	归约
#(T,(T	,a))#	归约
#(T,(T,	a))#	移进
#(T,(T,a	))#	移进
#(T,(T,S	))#	归约
#(T,(T	))#	归约
#(T,(T)	)#	移进

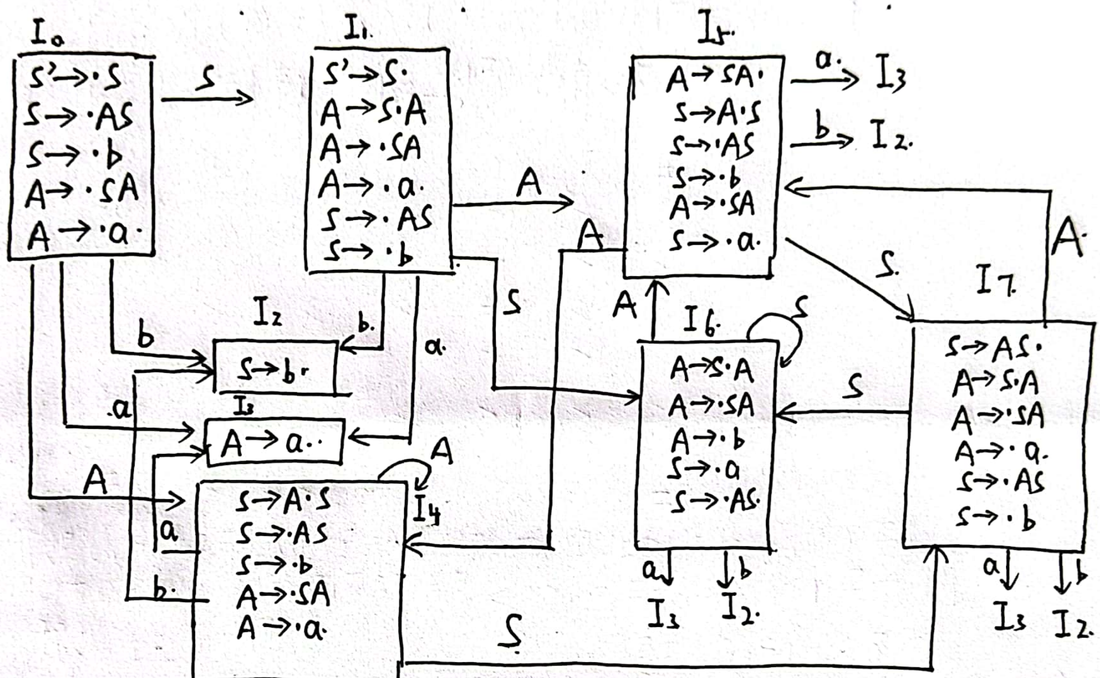
(T,S	)#	归约
#(T	)#	归约
#(T)	#	移进
#S	#	归约

5. 解:(1) 拓广文法.

S'→S
S→AS
S→b
A→SA
A→a

LR(0)项目: S'→·S ; S'→S· ; S→·AS ; S→A·S ; S→AS· ; S→·b ; S→b· ; A→·SA ; A→S·A ; A→SA· ; A→·a ; A→a· ;

(2) 我们ε-closure 闭包的方式来构造一个识别活前缀的DFA



由上图看出,I₁状态,I₅状态,I₇状态存在移进归约冲突,所以该文法不是LR(0)的

(3) 判断文法是否为SLR,即观察移进归约冲突能否被解决

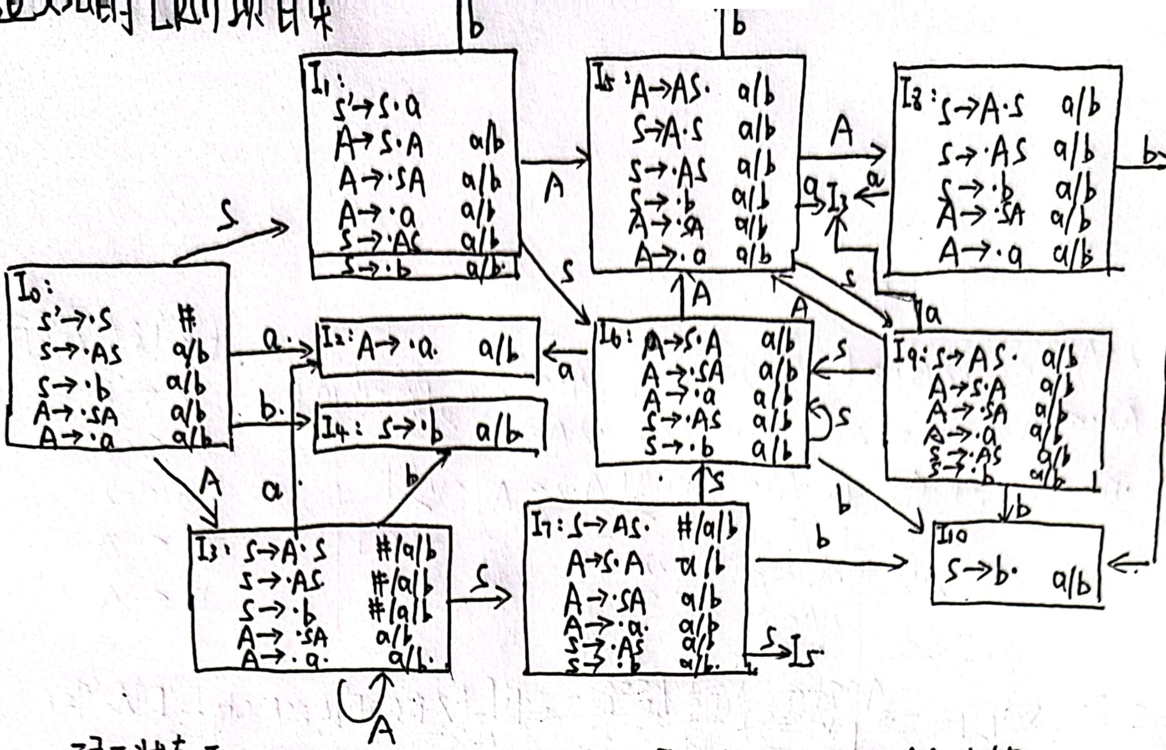
I_1中, S'→S·为归约, Follow(S')={#}与{a},{b}不交,可以解决

I_5中, A→SA·为归约, Follow(A)={b}与{b}相交.

I_7中, S→AS·为归约, Follow(S)={a,#}与{a}相交.

因此该文法不是SLR(1)文法.

的LR(1)项目集



对于状态I_5：因为 $A\to AS\cdot$ a/b 所以遇字符a时应用$A\to AS\cdot$归约

又因为 $A\to\cdot a$，a/b，所以遇字符a时应移进，

两者产生矛盾，该文法不是LR(1)文法。

因为LALR项目集是合并LR(1)项目集的同心集而形成的，从而存在移进-归约的矛盾的LR(1)项目集也会出现在LALR的某个项目集中，所以G也不是LALR文法。

8. $First(AaAb)=\{a\}$，$First(BbBa)=\{b\}$.

$First(AaAb)\cap First(BbBa)=\Phi$.

又因为文法不含左递归，又因为 $First(A)=\{\varepsilon\}$，$First(B)=\{\varepsilon\}$ $First(A)\cap Follow(A)=\Phi$

$Follow(A)=\{a,b\}$，$Follow(B)=\{a,b\}$ $First(B)\cap Follow(B)=\Phi$.

所以该文法是LL(1)的

构造该文法的LR(0)项目集

$I_0=\{S'\to\cdot S,\ S\to\cdot AaAb,\ S\to\cdot BbBa,\ A\to\cdot,\ B\to\cdot\}$

$I_1=\{S'\to S\cdot\}$

$I_2=\{S\to A\cdot aAb\}$

$I_3=\{S\to B\cdot bBa\}$

$I_4=\{S\to Aa\cdot Ab,\ A\to\cdot\}$

$I_5=\{S\to Bb\cdot Ba,\ B\to\cdot\}$

$I_6=\{S\to AaA\cdot b\}$

$I_7=\{S\to BbB\cdot a\}$

$I_8=\{S\to AaAb\cdot\}$

$I_9=\{S\to BbBa\cdot\}$.

I_0中存在移进-归约冲突

又∵$Follow(A)=\{a,b\}$与$\{a,b\}$有交

所以该文法非SLR(1).