第 4 章语法分析习题答案

1．判断

（1）由于递归下降分析法比较简单，因此它要求文法不必是 LL(1)文法。(×)

（2）某些左递归文法可能是 LL(1)文法。(×)

（3）任何 LL(1)文法都是无二义性的。(√)

（4）存在一种算法，能判定任何上下文无关文法是否是 LL(1) 文法。(√)

（5）算符优先分析过程和规范归约过程都是最右推导的逆过程。(×)

（6）每一个 SLR(1)文法都是 LR(1)文法。(√)

（7）任何一个 LL(1)文法都是一个 LR(1)文法，反之亦然。(×)

（8）由于 LALR 是在 LR(1)基础上的改进方法，所以 LALR 的能力强于 LR(1)。(×)

（9）所有 LR 分析器的总控程序都是一样的，只是分析表各有不同。(√)

（10）算符优先分析法很难完全避免将错误的句子得到正确的归约。(√)

2．文法 G[E]：

E→E+T|T

T→T*F|F

F→(E)|i

试给出句型(E+F)*i 的短语、简单短语、句柄和最左素短语。

答案：

画出语法树，得到：

短语：(E+F)*i ，(E+F) ，E+F ，F ，i

简单短语：F ，i

句柄：F

最左素短语：E+F

3．文法 G[S]：

S→SdT | T

T→T<G | G

G→(S) | a

试给出句型(SdG)<a 的短语、简单短语、句柄和最左素短语。

答案：

画出语法树，得到：

短语：(SdG)<a 、(SdG) 、SdG 、G 、a

简单(直接)短语：G 、a

句柄：G

最左素短语：SdG

4．对文法 G[S]提取公共左因子进行改写，判断改写后的文法是否为 LL(1)文法。

S→if E then S else S

S→if E then S

S→other

E→b

答案：

提取公共左因子；文法改写为：

S→if E then S S'|other

S'→else S| ε

E→b

LL(1)文法判定：

① 文法无左递归

② First(S)={if,other}， First(S')={else, ε}

First(E)={b}

Follow(S)= Follow(S')={else,#}

Follow(E)={then}

First(if E then S S')∩First(other)= Φ

First(else S)∩First(ε)= Φ

③First(S')∩Follow(S')={else}不为空集

故此文法不是 LL(1)文法。

5．对于给定文法 G[bexpr]：

bexpr→bexpr or bterm | bterm

bterm→bterm and bfactor | bfactor

bfactor→not bfactor|(bexpr) |true |false

（1）用 EBNF 改写该文法，消除左递归。

（2）试用类 C 语言为其构造一个递归下降分析程序。

答案：

（1）用 EBNF 改写该文法，消除左递归：

bexpr→bterm {or bterm }

bterm→bfactor {and bfactor}

bfactor→not bfactor | (bexpr) | true |false

（2） 用类 C 语言写出其递归下降分析程序

```
bexpr(){
    bterm();
    while (sym == ”or”){
        getsym();
        bterm();
    }
}
bterm(){
    bfactor();
    while (sym == ”and”){
        getsym();
        bfactor();
    }
}
bfactor(){
    if (sym == ”not”){
```

```
            getsym();
            bfactor();
        }
        else if (sym == ”(”){
            getsym(); E();
            if (sym == ”)”)  getsym();
            else error();
        }
        else if (sym == ”true”) getsym();
        else if (sym == ”false”) getsym();
        else error();
    }
```

6．对文法 G[S]:（清华教材99页第1题）

S→a|∧|(T)

T→T,S|S

（1）给出(a,(a,a))和(((a,a),∧,(a)),a)的最左推导。

（2）消除文法的左递归，将其改写为右递归文法，然后对每个非终结符写出不带回溯的递归子程序。

（3）经改写后的文法是否是 LL(1)的？给出它的预测分析表。

（4）给出输入串(a,a)#的分析过程，并说明该串是否为 G 的句子。

答案：

（1）对(a,(a,a)的 左推导为：

S⇒(T)

⇒(T,S)

⇒(S,S)

⇒(a,S)

⇒(a,(T))

⇒(a,(T,S))

⇒(a,(S,S))

⇒(a,(a,S))

⇒(a,(a,a))

对(((a,a),∧,(a)),a) 的 左推导为：

S⇒(T)

⇒(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),∧,(a)),a)

（2）改写文法为：

0) S→a

1) S→∧

2) S→(T)

3) T→S N

4) N→, S N

5) N→ε

非终结符	FIRST 集	FOLLOW 集
S	{a,∧,(}	{#,,,)}
T	{a,∧,(}	{)}....
N	{,,ε}.	{)}....

依据 LL(1)文法的判别条件，检查变换后的文法，可以得到：

① 变换后的文法不含左递归

② 对左部为 S 的产生式，右部多个候选式的 FIRST 集两两不相交。

③ 对左部为 N 的产生式可知：

FIRST （→, S N）={，}

FIRST （→ε）={ε} FOLLOW （N）={)}

由于 FIRST 集合中含有 ε 的非终结符的 Follow 集与其 First 集交集为空，FIRST(N)∩FOLLOW(N) ={，}∩{)}=∅ ，文法没有二义性，所以文法是 LL(1)的。

预测分析表如下：

	a	∧	(	)	,	#
S	→a	→∧	→(T)			
T	→S N	→S N	→S N			
N				→ε	→, S N	

也可由预测分析表中无多重入口判定文法是 LL(1)的。

（3）对输入串（a,a）#的分析过程为：

步骤	分析栈	读入符号	剩余输入串	推导所用的产生式或匹配
1	#S	(	a,a)#...	S→(T)
2	#)T(	(	a,a)#...	‘(’ 匹配
3	#)T	a	,a)#...	T→SN
4	#)NS	a	,a)#...	S→a
5	#)Na	a	,a)#...	‘a’ 匹配
6	#)N	,	a)#...	N→,SN
7	#)NS,	,	a)#...	‘,’ 匹配
8	#)NS	a	)#...	S→a
9	#)Na	a	)#...	‘a’ 匹配
10	#)N	)	#...	N→ ε
11	#)	)	#...	‘)’ 匹配
12	#	#		接受

可见输入串（a,a）#是文法的句子。

7．对下面的文法 G：（清华教材 100 页第 2 题）

E→TE'

E' →＋E｜ε

T→FT '

T'→T｜ε

F→PF '

F'→*F '｜ε

P→(E)｜a｜b｜ ^

（1）计算这个文法的每个非终结符的 FIRST 集和 FOLLOW 集。

（2）证明这个文法是 LL（1）的。

（3）构造它的预测分析表。

（4）构造它的递归下降分析程序。

	FIRST	FOLLOW
E	(, a, b, ^	), #
E'	+, ε	), #
T	(, a, b, ^	+,), #
T'	(, a, b, ^, ε	+,), #
F	(, a, b, ^	(, a, b, ^, +,), #
F'	*, ε	(, a, b, ^, +,), #
P	(, a, b, ^	*,(, a, b, ^, +,), #

答案：

（1）每个非终结符的 FIRST 集和 FOLLOW 集如下：

FIRST(P)={ (, a , b , ^ }

FIRST(F')={ * , ε}

FIRST(F)=FIRST(P) ={ (, a , b , ^ }

FIRST(T')=FIRST(T)∪{ε}={ (, a , b , ^ , ε}

FIRST(T)=FIRST(F) ={ (, a , b , ^ }

FIRST(E')={ + , ε }

FIRST(E)=FIRST(T)={ (, a , b , ^ }

FOLLOW(E)={) , # }

FOLLOW(E')=FOLLOW(E)={) , # }

FOLLOW(T)=FIRST(E')\ ε ∪FOLLOW(E)={+,) , # }

FOLLOW(T')=FOLLOW(T)=={+,) , # }

FOLLOW(F)=FIRST(T')\ ε ∪FOLLOW(T)={ (,a,b,^ , +,) , # }

FOLLOW(F')=FOLLOW(F)={(, a , b , ^ , +,) , # }

FOLLOW(P)=FIRST(F')\ ε ∪FOLLOW(F)={*,(,a, b ,^,+ ,) ,# }

（2）证明：

依据 LL(1)文法的判别条件，检查变换后的文法 G，可以得到：

①变换后的文法不含左递归

②对于有多个候选式的产生式，P→(E)｜a｜b｜^ ，有

FIRST((E))∩FISRT(a) ∩FIRST(b) ∩FIRST(^)=Φ

③对候选式的终结首符号集包含 ε 的产生式

对产生式 E'→＋E｜ε

FIRST(+E) ∩FOLLOW(E')={+} ∩ {) , # }=Φ

对产生式 T'→T｜ε

FIRST(T) ∩FOLLOW(T')={ (, a , b , ^ } ∩ {+,) , # } =Φ

对产生式 F'→*F'｜ ε

FIRST（*F') ∩FOLLOW(F')={*} ∩ {(, a , b , ^ , +,) , # }=Φ

综上可知，文法 G 是 LL（1）的。

（3）预测分析表如下：

	(	a	b	^	)	+	*	#
E	→TE’	→TE’	→TE’	→TE’				
E’					→ε	→+E		→ε
T	→FT’	→FT’	→FT’	→FT’				
T’	→T	→T	→T	→T	→ε	→ε		→ε
F	→PF’	→PF’	→PF’	→PF’				
F’	→ε	→ε	→ε	→ε	→ε	→ε	→*F’	→ε
P	→(E)	→a	→b	→^				

（4）递归下降分析程序如下：

E 的递归下降分析子程序

```
E(){
    T();
    E'( );
}
```

T 的递归下降分析子程序

```
T(){
    F();
    T'( );
}
```

F 的递归下降分析子程序

```
F(){
    P();
    F'( );
}
```

P 的递归下降分析子程序

```
P(){
    if (sym in[a, b, ^])
        getsym();
    else if (sym == ‘(’){
        getsym();
        E();
        if (sym = ‘)’)  getsym();
        else  error();
    }
    else  error();
}
```

E’ 的递归下降分析子程序

```
E'( ){
if (sym == ‘ + ’){
    getsym();
    E();
}
else
if (sym in[#, )])
  return;
else error();
}
```

T’ 的递归下降分析子程序

```
T'( ){
 if (sym in[), +, # ] )
  return;
else
 T()
}
```

F’ 的递归下降分析子程序

```
F'( ){
 if (sym == ‘*’){
    getsym();
    F'( );
}
else
if (sym in[a, b, (, ), ^, +, # ])
  return;
else
  error();
}
```

8. 已知文法 G[S]:

S→BA

A→BS| d

B→aA| bS | c

（1）构造其预测分析表。

求解其 FIRST 集：

符号	FIRST	FOLLOW
S	{a,b,c}	{#,a,b,c,d}
A	{a,b,c,d}	{a,b,c,d}
B	{a,b,c}	{a,b,c,d}
a		{a,b,c,d}
b		{a,b,c}
c		{a,b,c,d}
d		{a,b,c,d}

根据上表构建预测分析表，得到预测分析表如下：

	a	b	c	d	#
S	→BA	→BA	→BA		
A	→BS	→BS	→BS	→d	
B	→aA	→bS	→c		

（2）给出输入串 adccd 的 LL(1)分析过程，并说明该串是否为文法 G[S]的句子。

步骤	分析栈	读入符号	剩余输入串	推导所用的产生式或匹配
1	#S	a	dccd#	S→BA
2	#AB	a	dccd#	B→aA
3	#AAa	a	dccd#	‘a’匹配
4	#AA	d	ccd#	A→d
5	#Ad	d	ccd#	‘d’匹配
6	#A	c	cd#	A→BS
7	#SB	c	cd#	B→c
8	#Sc	c	cd#	‘c’ 匹配
9	#S	c	d#	S→BA
10	#AB	c	d#	B→c
11	#Ac	c	d#	‘c’ 匹配
12	#A	d	#	A→d
13	#d	d	#	‘d’ 匹配
14	#	#		接受

分析成功，该串是文法 G[S]的句子。

9．有文法 G[S]：（清华教材 122 页第 3 题）

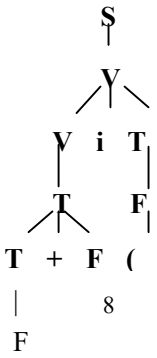
S→V

V→T|ViT

T→F|T+F

F→V*|(

（1）给出(+(i(的规范推导。



S=>V=>ViT=>ViF=>Vi(=>T i(=>T+F i(=>T+(i(=>F+(i(=>(+(i(

（2）指出句型 F+Fi(的短语，句柄，素短语。

短语：F，F+F，(，F+Fi(　　句柄：F　　素短语：(

（3）G[S]是否为 OPG？若是，给出（1）中句子的分析过程。

FIRSTVT 和 LASTVT

	FIRSTVT	LASTVT
S	i,+,),(	i,+,*,(
V	i,+,),(	i,+,*,(
T	+,),(	+,(,*
F	),(,	*,(

算符优先关系

	i	+	*	(	)	#
i	⋗	⋖	⋗	⋖	⋖	⋗
+	⋗	⋗	⋗	⋖	⋖	⋗
*	⋗	⋗	⋗			⋗
(	⋗	⋗	⋗			⋗
)	⋖	⋖		⋖	⋖	
#	⋖	⋖		⋖	⋖	≡

因为该文法是 OP，同时任意两个终结符的优先关系唯一，所以该文法为 OPG。

(+(i(的分析过程

步骤	栈	当前符号	剩余输入串	移进或归约
1	#	(	(+(i(#	移进
2	#(	+	(i(#	归约
3	#F	+	(i(#	移进
4	#F+	(	i(#	移进
5	#F+(	i	(#	归约
6	#F+F	i	(#	归约
7	#F	i	(#	移进
8	#Fi	(	#	移进
9	#Fi(	#		归约
10	#FiF	#		归约
11	#F	#		接受

10．文法 G［S］:（清华教材 122 页第 4 题）

S→S;G｜G

G→G(T)｜H

H→a｜(S)

T→T+S｜S

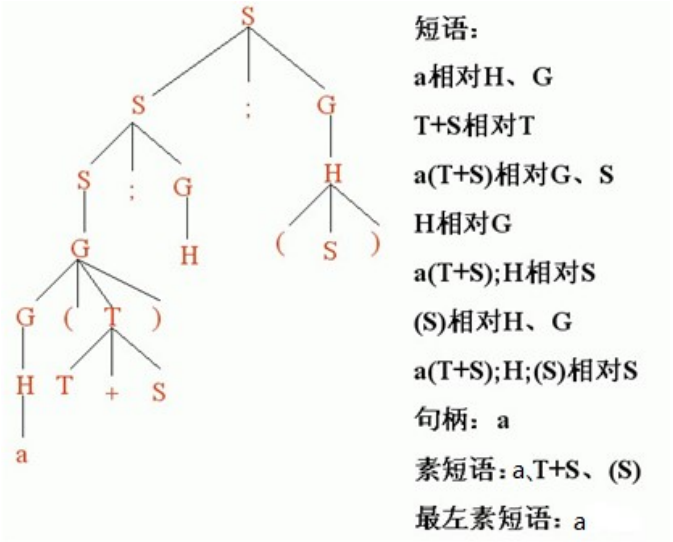
（1）构造 G［S］的算符优先关系表，并判断 G［S］是否为算符优先文法。

构造文法 G［S］的算符优先关系矩阵：

	;	(	)	a	+	#
;	·>	<·	·>	<·	·>	·>
(	<·	<·	=·	<·	<·	
a	·>	·>	·>		·>	·>
+	·>	·>	·>		·>	·>
#	<·	<·	·>	<·	·>	
	<·	<·		<·		=·

在上表中可看出终结符之间的优先关系是唯一的，或称 G［S］的算符优先关系矩阵不含多重入口，因此，G［S］是一个算符优先文法。

（2）给出句型 a(T+S);H;(S)的短语、句柄、素短语和最左素短语。



短语：

a相对H、G

T+S相对T

a(T+S)相对G、S

H相对G

a(T+S);H相对S

(S)相对H、G

a(T+S);H;(S)相对S

句柄：a

素短语：a、T+S、(S)

最左素短语：a

（3）给出 a;(a+a)和(a+a)的分析过程，说明它们是否为 G［S］的句子。

步骤	栈	当前符号	剩余输入串	移进或归约
（1）	#	(	a+a)#	移进
（2）	#(	a	+a)#	移进
（3）	#(a	+	a)#	归约
（4）	#(N	+	a)#	移进
（5）	#(N+	a	)#	移进
（6）	#(N+a	)	#	归约
（7）	#(N+N	)	#	归约
（8）	#(N	)	#	移进
（9）	#(N)	#		归约
（10）	#N	#		分析成功

说明是它的句子。

（4）给出（3）中输入串的最右推导，分别说明两输入串是否为 G[S]的句子。

S⇒ G⇒H⇒（S）由此往下 S 不可能推导出 a+a，所以 (a+a)不是 G［S］的句子。

（5）由（3）和（4）说明了算符优先分析的哪些缺点。

由于算符优先分析法去掉了单非终结符之间的归约，尽管在分析过程中，当决定是否为句柄时采取一些检查措施，但仍难完全避免把错误的句子得到正确的归约。

（6）算符优先分析过程和规范归约过程都是最右推导的逆过程吗？

算符优先分析过程不是最右推导的逆过程。规范归约过程是最右推导的逆过程。

11. 若有定义二进制数的文法 G[S]：（清华教材 166 页第 2 题）

S→L.L|L

L→LB|B

B→0|1

（1）试为该文法构造其 SLR 分析表，并判断该文法是否为 SLR(1)文法。

文法：

S→L.L|L

L→LB|B

B→0|1

拓广文法为 G′，增加产生式 S′→S　若产生式排序为：

0　S’→S

1　S →L.L

2　S →L

3　L →LB

4　L →B

5　B →0

6　6　B →1

由产生式知：

First (S’) = {0,1}

First (S) = {0,1}

First (L) = {0,1}

First (B) = {0,1}

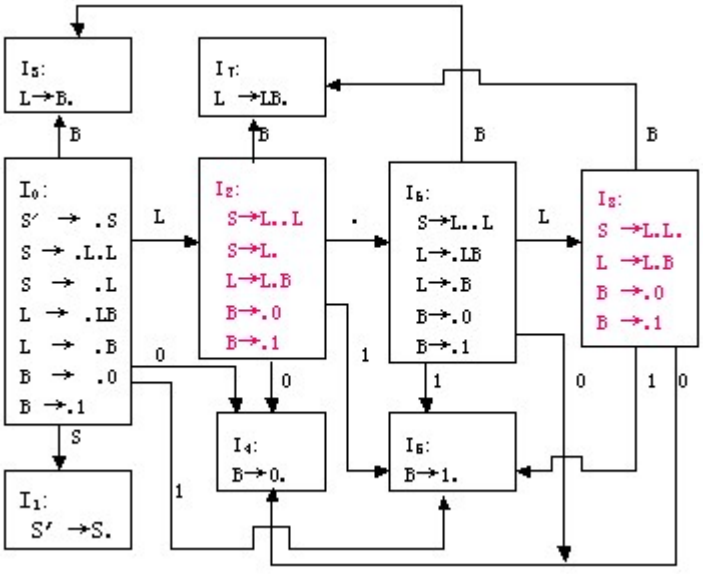
Follow(S’) = {#}

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 I_2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 I_2、I_8中：

Follow(s) ∩{0，1}= { #} ∩{0，1}=∅

所以在 I_2 、I_8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：题目 2 的 SLR(1)

分析表

状态	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

（2）给出输入串 101.110 的 LR 分析过程。

步骤	状态栈	符号栈	剩余输入串	动作	GOTO
1	0	#	101.110#	S5	
2	0 5	#1	01.110#	r6	3
3	0 3	#B	01.110#	r3	2
4	0 2	#L	01.110#	S4	
5	0 2 4	#L0	1.110#	r5	7
6	0 2 7	#LB	1.110#	r3	2
7	0 2	#L	1.110#	S5	
8	0 2 5	#L1	.110#	r6	7
9	0 2 7	#LB	.110#	r3	2
10	0 2	#L	.110#	S6	
11	0 2 6	#L.	110#	S5	
12	0 2 6 5	#L.1	10#	r6	3
13	0 2 6 3	#L.B	10#	r4	8
14	0 2 6 8	#L.L	10#	S5	
15	0 2 6 8 5	#L.L1	0#	r6	7
16	0 2 6 8 7	#L.LB	0#	r3	8
17	0 2 6 8	#L.L	0#	S4	
18	0 2 6 8 4	#L.L0	#	r5	7
19	0 2 6 8 7	#L.LB	#	r3	8
20	0 268	#L.L	#	r1	1
21	01	#S	#	acc	

分析成功，说明输入串 101.110 是题目 2 文法的句子。

12.已知文法 G[A] （清华教材 165 页第 1 题）

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。

答案：

拓广文法为 G′，增加产生式 S′→A

若产生式排序为：

0 S'→A

1 A →aAd

2 A →aAb

3 A → ε

由产生式知：

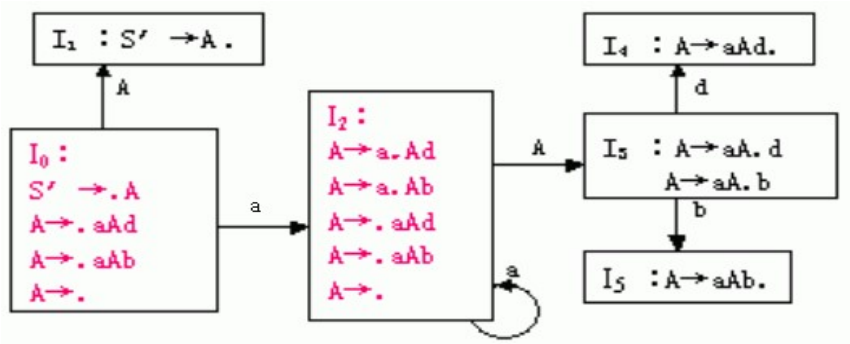
FIRST（S'）= {ε,a}

FIRST（A）= {ε,a}

FOLLOW(S'）= {#}

FOLLOW(A) = {d,b,#}

G′ 的 LR(0)项目集族及识别活前缀的 DFA 如下图所示：



在 I0 中：

A →.aAd 和 A →.aAb 为移进项目，A →.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 I0、I2 中：

Follow(A) ∩{a}= {d，b，#} ∩{a}=**Φ**

即 I0、I2 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
	a	d	b	#	A
0	S2	r3	r3	r3	1
1				acc	
2	S2	r3	r3	r3	3
3		S4	S5		
4		r1	r1	r1	
5		r2	r2	r2	

对输入串 ab#的分析过程

步骤	状态栈	符号栈	剩余输入串	ACTION	GOTO
1	0	#	ab#	S2	
2	0 2	#a	b#	r3	3
3	0 2 3	#aA	b#	S5	
4	0 2 3 5	#aAb	#	r2	1
5	0 1	#A	#	acc	

分析成功，说明输入串 ab 是文法的句子。