编译原理

作业 3: 编译原理（第 5 章）

中国人民大学　信息学院　崔冠宇　2018202147

一、书中练习题

1. 令文法 G_1 为:

- $E \to E + T \mid T$
- $T \to T * F \mid F$
- $F \to (E) \mid i$

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。

解:

(1) 要证明是句型，只要给出一个推导即可：

$E \Rightarrow E + T \Rightarrow E + T * F$

(2) E + T * F 和 T * F 是短语。

因为 $E \Rightarrow^* E$，且 $E \Rightarrow^+ E + T * F$，所以 E + T * F 是关于 E 的短语。

因为 $E \Rightarrow^* E + T$，且 $T \Rightarrow^+ T * F$，所以 T * F 是关于 T 的短语。

(3) T * F 是直接短语。

因为 $E \Rightarrow^* E + T$，且 $T \Rightarrow T * F$，所以 T * F 是关于 T 的直接短语。

(4) T * F 是句柄。

因为它是最左直接短语，因此是句柄。

2. 考虑下面的表格结构文法 G_2:

- $S \to a \mid \wedge \mid (T)$

- $T \rightarrow T\ ,\ S \mid S$

(1) 给出 (a, (a, a)) 和 (((a, a), ∧, (a)), a) 的最左和最右推导。

(2) 指出 (((a, a), ∧, (a)), a) 的规范归约及每一步的句柄。根据这个规范归约，给出“移进-归约”的过程，并给出它的语法树自下而上的构造过程。

解: (1)

(a, (a, a)) 的最左推导：S ⇒ (T) ⇒ (T, S) ⇒ (S, S) ⇒ (a, S) ⇒ (a, (T)) ⇒ (a, (T, S)) ⇒ (a, (S, S))
⇒ (a, (a, S)) ⇒ (a, (a, a))

(a, (a, a)) 的最右推导：S ⇒ (T) ⇒ (T, S) ⇒ (T, (T)) ⇒ (T, (T, S)) ⇒ (T, (T, a)) ⇒ (T, (S, a)) ⇒ (T, (a, a))
⇒ (S, (a, a)) ⇒ (a, (a, a))

(((a, a), ∧, (a)), a) 的最左推导：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)

(((a, a), ∧, (a)), a) 的最右推导：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)), a) ⇒ ((S, ∧, (a)), a)
⇒ (((T), ∧, (a)), a) ⇒ (((T, S), ∧, (a)), a) ⇒ (((T, a), ∧, (a)), a) ⇒ (((S, a), ∧, (a)), a) ⇒ (((a, a), ∧, (a)), a)

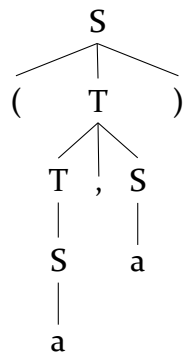
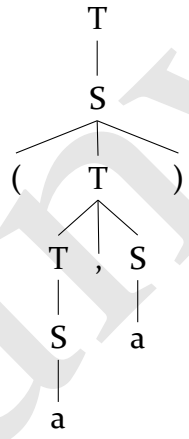
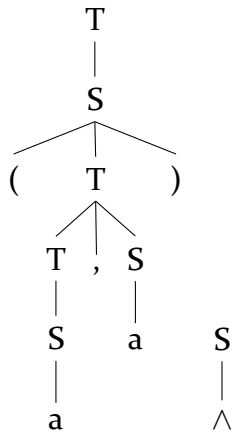
(2)

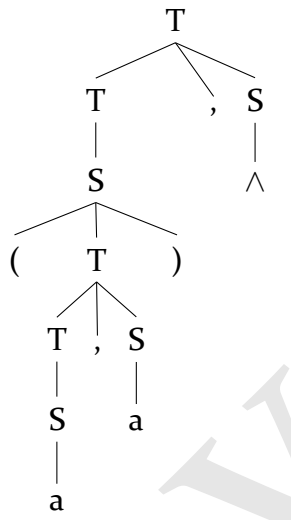
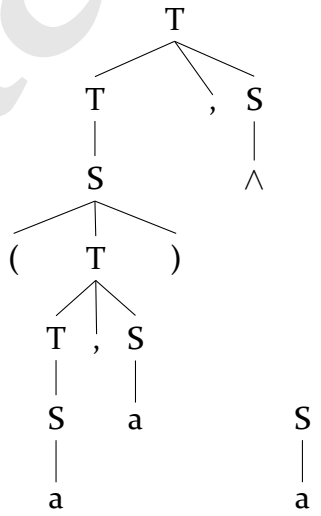
规范归约其实是最右推导的逆过程。根据上面 (((a, a), ∧, (a)), a) 的最右推导，可以写出它的规范归约和句柄（句柄为下划线所指部分）：

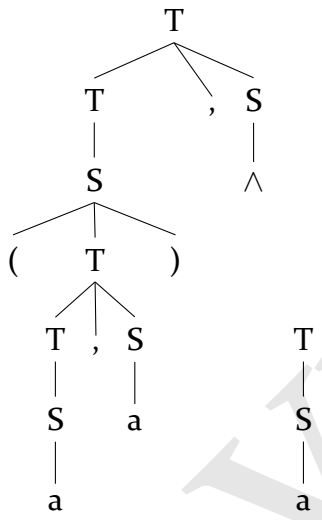
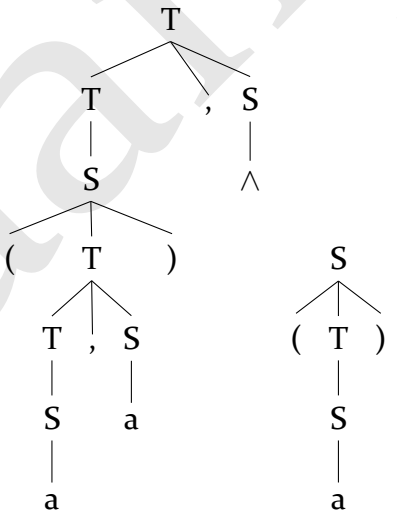
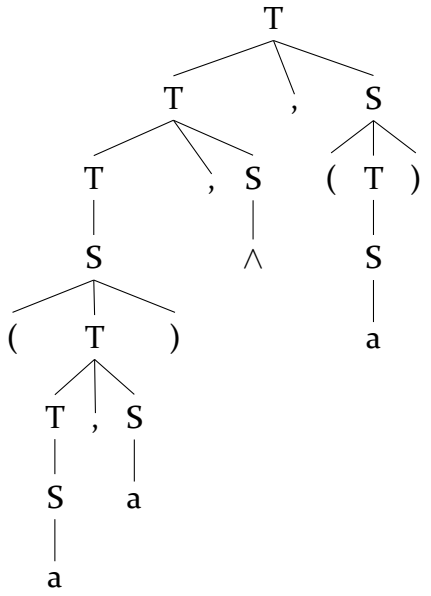
(((a, a), ∧, (a)), a)，(((S, a), ∧, (a)), a)，(((T, a), ∧, (a)), a)，(((T, S), ∧, (a)), a)，((((T), ∧, (a)), a)，
((S, ∧, (a)), a)，((T, △, (a)), a)，((T, S, (a)), a)，((T, (a)), a)，((T, (S)), a)，((T, (T)), a)，((T, S), a)，
((T), a)，(S, a)，(T, a)，(T, S)，(T)，S

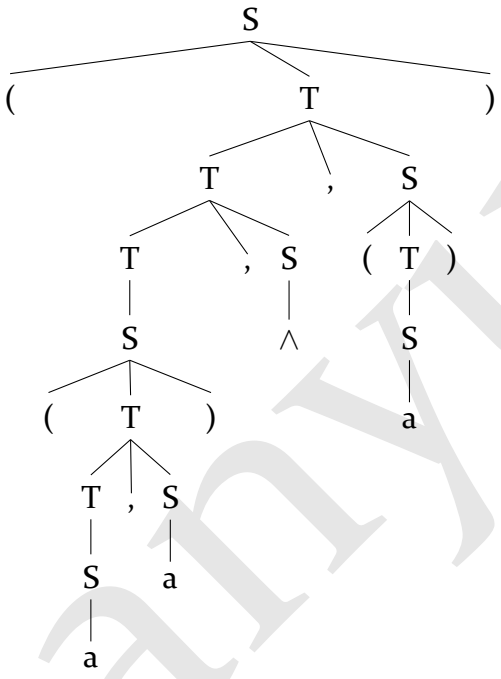
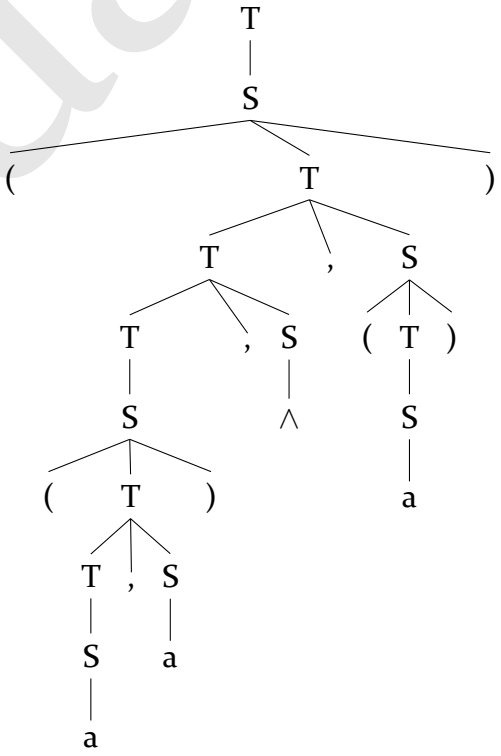
下面给出规范归约的“移进-规约”过程以及语法树自下而上的构造过程：

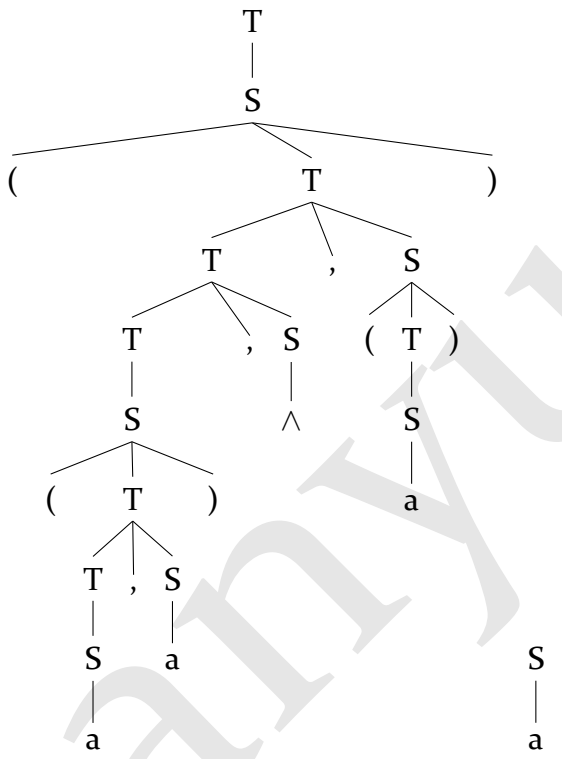
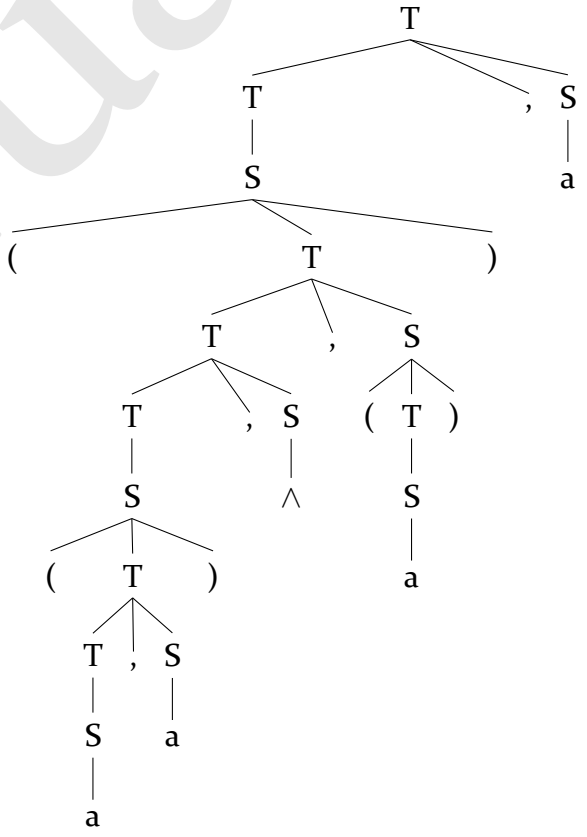
栈	输入串	动作	语法树
#	(((a, a), ∧, (a)), a)#	(初始化)	
#(	((a, a), ∧, (a)), a)#	进	
#((	(a, a), ∧, (a)), a)#	进	
#(((	a, a), ∧, (a)), a)#	进	
#(((a	, a), ∧, (a)), a)#	进	
#(((S	, a), ∧, (a)), a)#	归	
#(((T	, a), ∧, (a)), a)#	归	
#(((T,	a), ∧, (a)), a)#	进	
#(((T,a	), ∧, (a)), a)#	进	
#(((T,S	), ∧, (a)), a)#	归	
#(((T	), ∧, (a)), a)#	归	
#(((T)	, ∧, (a)), a)#	进	

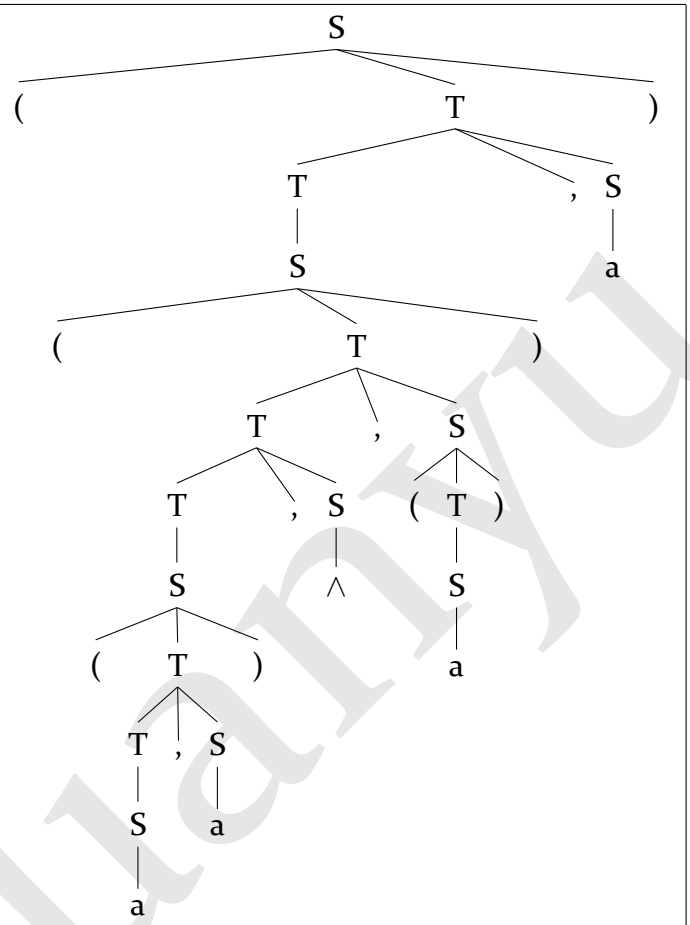
#((S	, ∧, (a)), a)#	归	
#((T	, ∧, (a)), a)#	归	
#((T,	∧, (a)), a)#	进	
#((T, ∧	, (a)), a)#	进	
#((T, S	, (a)), a)#	归	

#((T	, (a)), a)#	归	
#((T,	(a)), a)#	进	
#((T,(	a)), a)#	进	
#((T,(a	)), a)#	进	
#((T,(S	)), a)#	归	

#((T,(T	)), a)#	归	
#((T,(T)	), a)#	进	
#((T,S	), a)#	归	
#((T	), a)#	归	

#((T)	, a)#	进	
#(S	, a)#	归	
#(T	, a)#	归	
#(T,	a)#	进	
#(T, a	)#	进	

#(T, S	)#	归	
#(T	)#	归	
#(T)	#	进	

#S	#	归	
#S	#	(接受)	

3. (1) 计算练习 2 文法 G_2 的 FIRSTVT 和 LASTVT。

(2) 计算 G_2 的优先关系。G_2 是一个算符优先文法吗?

(3) 计算 G_2 的优先函数。

(4) 给出输入串 (a, (a, a)) 的算符优先分析过程。

解:

(1) G_2:

- S → a | ∧ | (T)
- T → T , S | S

1. FIRSTVT(S) = {a, ∧, (}, LASTVT(S) = {a, ∧,)}

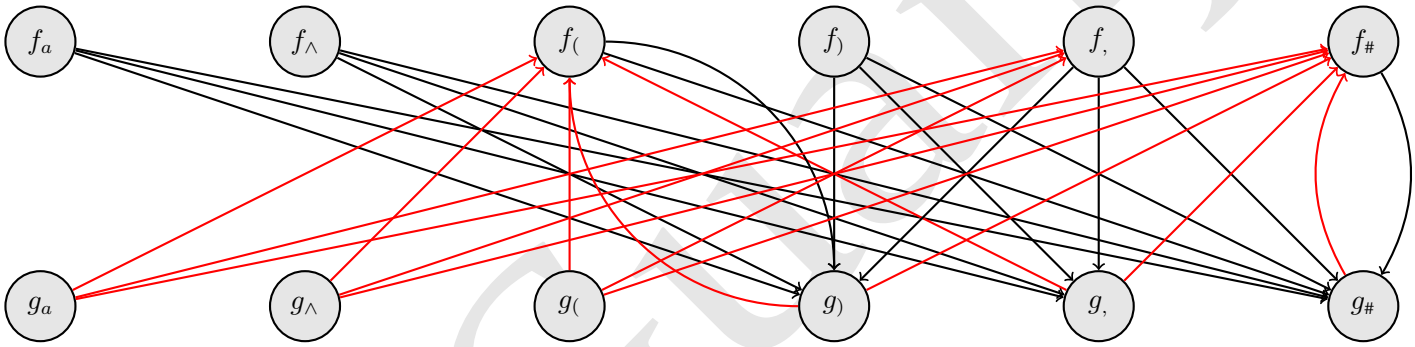
2. FIRSTVT(T) = {, , a, ∧, (}，LASTVT(T) = {, , a, ∧,)}

(2) 构造优先关系表：

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

没有发生冲突，可见这是一个算符优先文法。

(3) 先画出关系间的有向图：



然后根据某点可达的节点数，给出可能的优先函数：

	a	∧	(	)	,	#
f	6	6	4	6	6	2
g	7	7	7	4	5	2

验证：f(a) > g())，f(a) > g(,)，f(a) > g(#)，f(∧) > g())，f(∧) > g(,)，f(∧) > g(#)，f(() < g(a)，f(() < g(∧)，f(() < g((，f(() = g())，f(() < g(,)，f(() > g(#)，f()) > g())，f()) > g(,)，f()) > g(#)，f(,) < g(a)，f(,) < g(∧)，f(,) < g((，f(,) > g())，f(,) > g())，f(,) > g(,)，f(,) > g(#)，f(#) < g(a)，f(#) < g(∧)，f(#) < g((，f(#) < g())，f(#) < g(,)，f(#) = g(#)，没有冲突，因此是算符优先文法。

(4) 过程如下 (由于算符优先文法不能区分各种非终结符，因此统一用 N 表示)：

栈	输入	动作
#	(a, (a, a))#	(初始化)
#(	a, (a, a))#	f(#) < g(()，入栈
#(a	, (a, a))#	f(() < g(a)，入栈
#(N	, (a, a))#	f(a) > g(,)，归约
#(N,	(a, a))#	f(() < g(,)，进栈
#(N,(	a, a))#	f(,) < g(()，进栈
#(N,(a	, a))#	f(() < g(a)，进栈
#(N,(N	, a))#	f(a) > g(,)，归约
#(N,(N,	a))#	f(() < g(,)，进栈
#(N,(N,a	))#	f(,) < g(a)，进栈
#(N,(N,N	))#	f(a) > g())，归约
#(N,(N	))#	f(,) > g())，归约
#(N,(N)	)#	f(() = g())，入栈
#(N,N	)#	f()) > g())，归约
#(N	)#	f(,) > g())，归约
#(N)	#	f(() = g())，进栈
#S	#	f()) > g(#)，归约

4. 存在一种称为简单优先的自下而上分析法，这种分析法不会把错误句子当作为正确句子。一个文法 G，如果它不含 ε-产生式，也不含任何右部相同的不同产生式，并且它的任何符号对 (X, Y) ——X 和 Y 为终结符或非终结符——顶多存在下述三种关系 $\doteq$、$\lessdot$、$\gtrdot$ 之一，则称这个文法 G 是一个简单优先文法。这三种关系的定义是：

A. $X \doteq Y$ 当且仅当 G 中含有形如 $P \to \cdots XY \cdots$ 的产生式；

B. $X \lessdot Y$ 当且仅当 G 中含有形如 $P \to \cdots XQ \cdots$ 的产生式，其中 Q 为非终结符，而且 $Q \Rightarrow^+ Y \cdots$；

C. $X \gtrdot Y$ 当且仅当 Y 为文法 G 的终结符，且 G 含有形为 $P \to \cdots QR \cdots$ 的产生式，使得 $Q \Rightarrow^+ \cdots X$ 而 $Y \in \mathrm{FIRST}(R)$。例如，假定有规则 $S \to (\ T\ )$ 和推导 $T \Rightarrow S \Rightarrow a$ 则 $S \gtrdot)$ 和 $a \gtrdot)$ 成立。注意，上述 R 可能是终结符也可能是非终结符。

D. 对任何 X，若 $S \Rightarrow^+ X \cdots$，则 $\# \lessdot X$；若 $S \Rightarrow^+ \cdots X$ 则 $X \gtrdot \#$。

按简单优先文法的定义，回答以下问题：

(1) 构造文法 G_2 的简单优先分析表，辨明它是否为一个简单优先文法?

(2) 下面的文法产生和 $L(G_2)$ 相同的语言，

- S → a | ∧ | (R)
- T → S , T | S
- R → T

验证它的简单优先关系如下表所示：（注：红色部分疑似为题目有错的部分）

	R	S	T	a	∧	,	(	)	#
R								$\doteq$	
S						$\doteq$		$\gtrdot$	
T						$\gtrdot$		$\gtrdot$	$\gtrdot$
a						$\gtrdot$		$\gtrdot$	$\gtrdot$
∧		$\lessdot$	$\doteq$	$\lessdot$	$\lessdot$		$\lessdot$		
,		$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		$\lessdot$		
(	$\doteq$					$\gtrdot$		$\gtrdot$	$\gtrdot$
)									
#				$\lessdot$	$\lessdot$		$\lessdot$		

(3) 按上面的简单优先表构造优先函数。

(4) 证明简单优先文法是无二义的。

进一步说，简单优先文法的任何句型 $X_1X_2\cdots X_n$ 的句柄是满足条件 $X_{j-1} \lessdot X_j \doteq X_{j+1} \doteq \cdots \doteq X_i \gtrdot X_{i+1}$ 的最左子串 $X_jX_{j+1}\cdots X_i$。

(5) 构造简单优先分析器。

解:

(1) G_2:

- S → a | ∧ | (T)
- T → T , S | S

对比简单优先文法的简单优先关系的定义与算符优先文法的算符优先关系，可以发现二者具有相似之处，因此也可以类似定义 FIRSTV(X) 集合，表示跟在终结符或非终结符 X 后面的非终结符 Q 推导出来的第一个终结符或非终结符，以及 LASTV(Y) 集合表示某终结符或非终结符 R 的 FIRST 集合中的某终结符 Y 前面的非终结符 Q 推导出来的最后一个终结符或非终结符。

1. 根据情况 A，产生式右侧相邻的两个符号都是优先级相等的，可得：

   - $($ $\doteq$ T
   - T $\doteq$ $)$
   - T $\doteq$,
   - , $\doteq$ S

2. 根据情况 B ，可以先求各符号的 FIRSTV 集合：

   - FIRSTV(S) $= \emptyset$
   - FIRSTV(a) $= \emptyset$
   - FIRSTV($\wedge$) $= \emptyset$
   - FIRSTV(() $= \{$T, S, a, $\wedge$, ($\}$
   - FIRSTV(T) $= \emptyset$
   - FIRSTV()) $= \emptyset$
   - FIRSTV(,) $= \{$a, $\wedge$, ($\}$

   于是可得

   - $($ $\lessdot$ T
   - $($ $\lessdot$ S
   - $($ $\lessdot$ a
   - $($ $\lessdot$ $\wedge$
   - $($ $\lessdot$ $($
   - , $\lessdot$ a
   - , $\lessdot$ $\wedge$
   - , $\lessdot$ $($

3. 根据情况 C，可以先求各非终结符的 LASTV 集合：

- LASTV(a) $= \emptyset$
- LASTV($\wedge$) $= \emptyset$
- LASTV(() $= \emptyset$
- LASTV()) $= \{S, a, \wedge,)\}$
- LASTV(,) $= \{S, a, \wedge,)\}$

于是可得

- S $\gtrdot$)
- a $\gtrdot$)
- $\wedge \gtrdot$)
-) $\gtrdot$)
- S $\gtrdot$,
- a $\gtrdot$,
- $\wedge \gtrdot$,
-) $\gtrdot$,

4. 根据情况 D，有

- # $\lessdot$ a
- # $\lessdot \wedge$
- # $\lessdot$ (
- a $\gtrdot$ #
- $\wedge \gtrdot$ #
-) $\gtrdot$ #

然后构造简单优先分析表，如下：

	S	T	a	∧	(	)	,	#
S						⋗	⋗	
T						≍	≍	
a						⋗	⋗	⋗
∧						⋗	⋗	⋗
(	⋖	≍ ⋖	⋖	⋖	⋖			
)						⋗	⋗	⋗
,	≍		⋖	⋖	⋖			
#			⋖	⋖	⋖			

发现 (≍ T 和 (⋖ T 冲突，这不是一个简单优先文法。

(2)

- $S \to a \mid \wedge \mid (R)$
- $T \to S , T \mid S$
- $R \to T$

与上面类似，分几种情况

1. 根据情况 A，产生式右侧相邻的两个符号都是优先级相等的，可得：
   - $(\asymp R$
   - $R \asymp)$
   - $S \asymp ,$
   - $, \asymp T$
2. 根据情况 B，可以先求各符号的 FIRSTV 集合：
   - FIRSTV(S) $= \emptyset$
   - FIRSTV(a) $= \emptyset$
   - FIRSTV($\wedge$) $= \emptyset$
   - FIRSTV(() $= \{T, S, a, \wedge, (\}$

- FIRSTV(T) $= \emptyset$
- FIRSTV()) $= \emptyset$
- FIRSTV(,) $= \{$S, a, $\wedge$, ($\}$
- FIRSTV(R) $= \emptyset$

于是可得

- ($\lessdot$ T
- ($\lessdot$ S
- ($\lessdot$ a
- ($\lessdot \wedge$
- ($\lessdot$ (
- , $\lessdot$ S
- , $\lessdot$ a
- , $\lessdot \wedge$
- , $\lessdot$ (

3. 根据情况 C，可以先求各非终结符的 LASTV 集合：

- LASTV(a) $= \emptyset$
- LASTV($\wedge$) $= \emptyset$
- LASTV(() $= \emptyset$
- LASTV()) $= \{$T, S, a, $\wedge$,)$\}$
- LASTV(,) $= \{$a, $\wedge$,)$\}$

于是可得

- T $\gtrdot$)

- S ⋗)
- a ⋗)
- ∧ ⋗)
-) ⋗)
- a ⋗ ,
- ∧ ⋗ ,
-) ⋗ ,

4. 根据情况 D，有

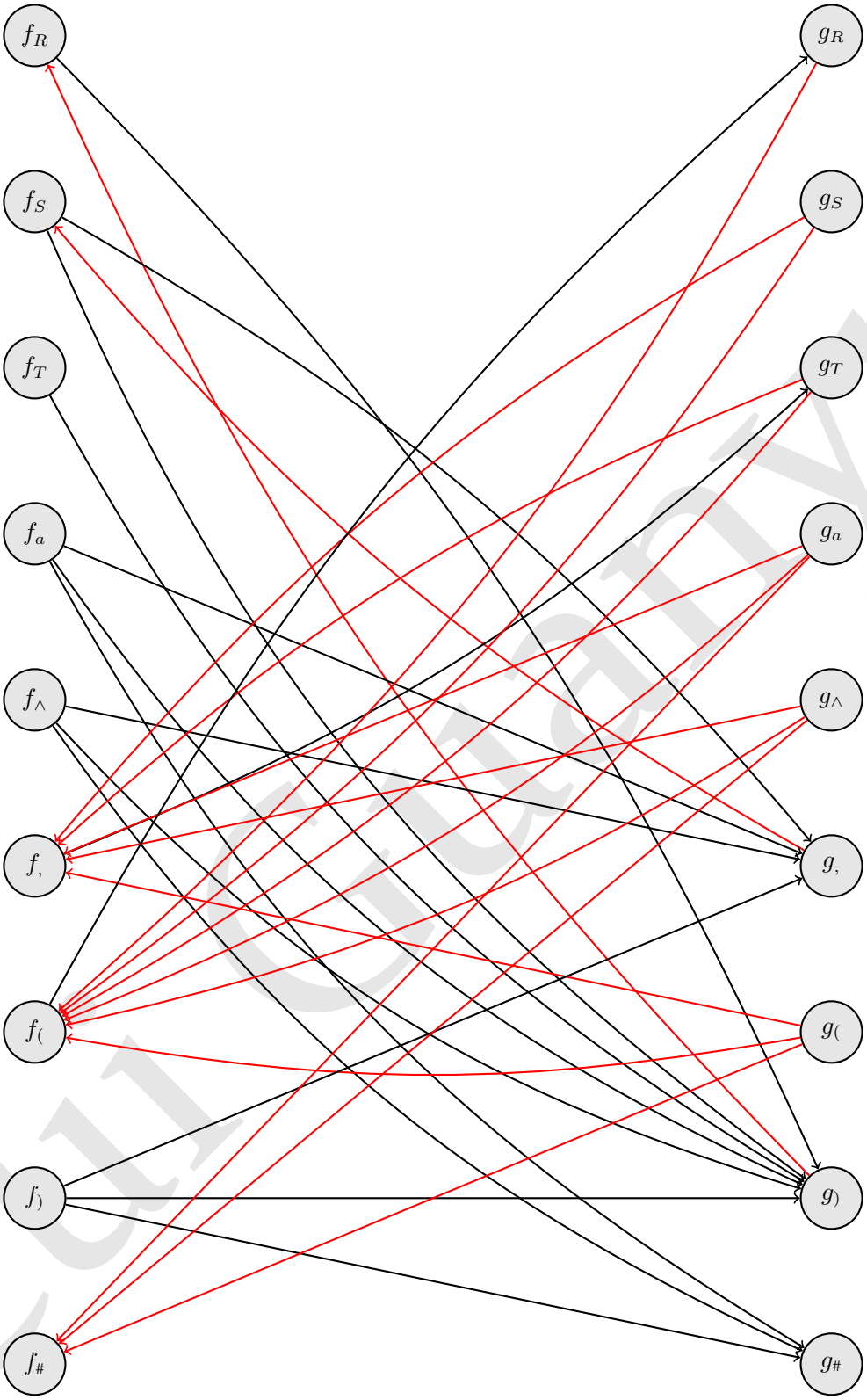
- # ⋖ a
- # ⋖ ∧
- # ⋖ (
- a ⋗ #
- ∧ ⋗ #
-) ⋗ #

然后构造简单优先分析表，如下：

	R	S	T	a	∧	,	(	)	#
R						⋗		⋗	
S						⋗		⋗	⋗
T						⋗		⋗	⋗
a						⋗		⋗	⋗
∧		⋖	⋗	⋖	⋖		⋖		
,		⋖	⋖	⋖	⋖		⋖		
(	⋗	⋖	⋖	⋖	⋖	⋗	⋖	⋗	⋗
)									
#				⋖	⋖		⋖		

发现没有冲突，因此是简单优先文法，而且分析表与题目所给的一致。

(3) 与算符优先文法的方法类似，先画出有向图：



然后根据每个节点可达的节点，计算优先函数表：

	R	S	T	a	∧	,	(	)	#
f	2	4	3	6	6	5	5	6	1
g	6	6	5	6	6	4	5	2	1

(4) 证明：

选择最左字串是因为这是最右推导的逆过程。下面只需要证明：

> 对句型 $X_1X_2\cdots X_{j-1}AX_{i+1}\cdots X_n$（设 A 是最右非终结符）用产生式 $A \to X_jX_{j+1}\cdots X_i$ 做最右推导，产生的新串 $X_1\ X_2\cdots\ X_n$ 满足 $X_{j-1}\lessdot X_j \doteq X_{j+1} \doteq \cdots \doteq X_i \gtrdot X_{i+1}$。

1. 根据情况 A，由于有产生式 $A \to X_jX_{j+1}\cdots\ X_i$，可得 $X_j \doteq X_{j+1} \doteq \cdots \doteq X_i$；
2. 由于 $X_1X_2\cdots X_{j-1}AX_{i+1}\cdots X_n$ 是 G 的一个句型，所以在 G 中添加一个产生式 $S \to X_1X_2\cdots X_{j-1}AX_{i+1}\cdots X_n$ 也不会改变这个语言，此时根据情况 B 可得 $X_{j-1}\lessdot X_j$；
3. 同理可得 $X_i \gtrdot X_{i+1}$。

(5) 可以利用 (4) 证得的性质构造简单分析器，分析算法如下：

Algorithm 1 简单优先分析的核心算法 `parse()`

```
// 初始化
stack.push(#)
// 指向第一个词
token = getNextToken()
while token != "#" do
    if f(stack.top()) <= g(token) then
        // 进栈
        stack.push(token)
        token = getNextToken()
    else
        // 根据性质找句柄
        while f(stack.top()) >= g(token) do
            t = stack.pop()
            rightPart.push_front(t)
        end while
        查找产生式右部为 rightPart 的产生式
        if 找到了产生式 A -> rightPart then
            将 rightPart 清空
            stack.push(A)
        else
            出错处理
            break
        end if
    end if
end while
```