南京理工大学

编译原理

项欣光

计算机科学与工程学院

第4章 自顶向下语法分析方法

- 理解 “能使用自顶向下分析技术的文法必须是LL(1)文法”
- LL(1)文法的充要条件
- LL(1)文法的判别
- 某些 非LL(1)文法 到 LL(1)文法 的等价变换
  - 提取左公共因子
  - 消除左递归（直接左递归、间接左递归）
- 不确定的自顶向下分析思想
- 确定的自顶向下分析方法
  - 递归子程序法
  - 预测分析法[判别LL(1)文法；构造预测分析表；分析输入串]

预备知识

1. 语法分析的作用：识别单词符号序列是否是给定文法的正确程序。
2. 语法分析的方法：

自顶向下分析
- 确定的方法
- 不确定的方法

自底向上分析
- 算符优先分析
- LR分析

3. 自顶向下分析法："面向目标的分析方法"
   从开始符号出发，企图推导出与输入的单词串完全匹配的句子。
   （1）确定的方法：需要对文法有一定的限制。
   优点：简单、直观
   （2）不确定的方法：带回溯的分析方法——穷举的试探方法
   缺点：效率低、代价高

4.1 确定的自顶向下分析思想

主要思想：

从文法的开始符号出发，如何根据当前的单词符号，唯一地确定选用哪个产生式来替换相应的V_N向下推导。

文法例1

例1：文法

G1[S]：S→pA

S→qB

A→cAd

A→a

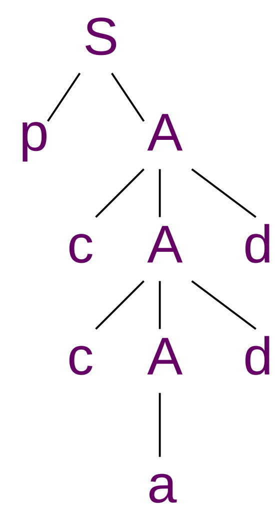
B→dB

B→b

W=pccadd 自顶向下的推导过程：

S ⇒pA ⇒pcAd ⇒pccAdd ⇒pccadd

对应的语法树：



这个文法的特点： （保证了推导过程唯一）

1. 每个产生式的右部都由终结符号开始。
2. 左部相同的产生式，它们的右部由不同的终结符开始。

文法例2

例2：

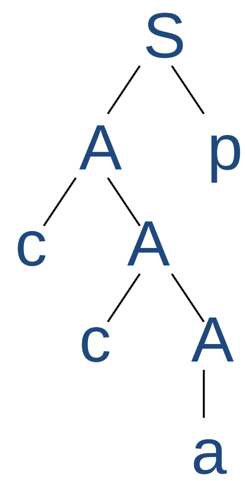
文法G[S]：

S→Ap
S→Bp
A→a
A→cA
B→b
B→dB

W = ccap 自顶向下的推导过程：

$$S \Rightarrow Ap \Rightarrow cAp \Rightarrow ccAp \Rightarrow ccap$$

语法树：



这个文法的特点： （保证了推导过程唯一）

1. 每个产生式的右部不全是由终结符号开始。
2. 左部相同的产生式，它们的右部由不同的终结符或非终结符开始。（引出First集合）
3. 文法中无空产生式。 （→ ε ）

First集合

为得到唯一的推导过程，条件为：

<u>左部相同的产生式，其“右部的首符号集合”不相交</u>。

定义：设 $G=(V_T, V_N, S, P)$ 是上下文无关文法，

$FIRST(\alpha)=\{a \mid \alpha \overset{*}{\Rightarrow} a\beta,\ a \in V_T,\ \alpha, \beta \in V^*\}$

若 $\alpha \overset{*}{\Rightarrow} \varepsilon$，则规定 $\varepsilon \in FIRST(\alpha)$

例2：文法G[S]：

S→Ap

S→Bp

A→a

A→cA

B→b

B→dB

求：

First(Ap) **={a,c}**

First(Bp) **={b,d}**

First(a) **={a}**

First(cA) **={c}**

First(b) **={b}**

First(dB) **={d}**

文法例3

- 例3：文法G[S]：

  S→aA

  S→d

  A→bAS

  A→ ε

W=abd 自顶向下的推导过程：

$S \Rightarrow aA \Rightarrow abAS \Rightarrow abS \Rightarrow abd$

这个文法的特点：

1. 文法中包含空产生式。（→ ε ）
2. 为得到唯一的推导过程，条件为：

   当某一V_N的产生式含空产生式，

   则它的非空产生式的First集两两互不相交，且

   与推导过程中紧跟该V_N可能出现的V_T集合也不相交。

Follow集

Select集

Follow集合

- 定义：设 $G=(V_T, V_N, S, P)$ 是上下文无关文法，$A \in V_N$，S是开始符号。

  $FOLLOW(A)=\{a \mid S \overset{*}{\Rightarrow} \mu A\beta$ 且 $a \in FIRST(\beta), \mu \in V_T^*, \beta \in V^+\}$

- 若 $S \overset{*}{\Rightarrow} \mu A\beta$，且 $\beta \overset{*}{\Rightarrow} \varepsilon$，则规定 $\# \in FOLLOW(A)$

  #作为输入串的结束符，或称为句子括号，

  如：#输入串#

通俗地讲：

$FOLLOW(A)=\{a \mid S \overset{*}{\Rightarrow} \ldots Aa \ldots, a \in V_T\}$

若S $\overset{*}{\Rightarrow} \ldots A$，则规定 $\# \in FOLLOW(A)$

举例，求例3中每个非终结符的Follow集

返回

SELECT集合

- 为得到唯一的推导过程，条件为：

  当 某一V_N的产生式含空产生式，

  则 它的非空产生式的First集两两互不相交 且

  与推导过程中紧跟该V_N可能出现的V_T集合也不相交

- 可得到唯一的推导过程的条件 等价的表示：

  若 $A \rightarrow \alpha$ $A \rightarrow \beta$ 其中$A \in V_N$，α，$\beta \in V_N{}^*$，α不推导出空，β能推导出空，

  则 $FIRST(\alpha) \cap ((FIRST(\beta) - \{\varepsilon\}) \cup FOLLOW(A)) = \Phi$

- **SELECT集 定义：**

  给定上下文无关文法的产生式$A \rightarrow \alpha$，$A \in V_N$，$\alpha \in V^*$，

  若$\alpha \not\overset{*}{\Rightarrow} \varepsilon$，则 $SELECT(A \rightarrow \alpha) = First(\alpha)$

  若$\alpha \overset{*}{\Rightarrow} \varepsilon$，则 $SELECT(A \rightarrow \alpha) = (First(\alpha) - \{\varepsilon\}) \cup Follow(A)$

举例，求例3中每个产生式的Select集

返回

LL(1)文法

可使用“自顶向下分析”的文法称为LL(1)文法。
必须满足的条件：

- 定义：

  一个上下文无关文法是LL(1)文法的充要条件：

  对每个V_N，A的两个不同产生式$A\rightarrow\alpha$，$A\rightarrow\beta$，

  满足 $SELECT(A\rightarrow\alpha) \cap SELECT(A\rightarrow\beta)=\phi$

  其中，α、β不能同时推导出ε。

L：scan from Left　从左向右扫描输入串
L：analyze from Left：分析过程是最左推导
1：只需向右看一个符号便可以决定选择哪个产生式进行推导。

LL(1)文法判别举例

例4：判断文法G[S]是否为LL(1)文法？

$$G[S]: \quad S \to aAS$$
$$S \to b$$
$$A \to bA$$
$$A \to \varepsilon$$

解：$Select(S \to aAS) = \{a\}$

$Select(S \to b) = \{b\}$

$Select(A \to bA) = \{b\}$

$Select(A \to \varepsilon) = (First(\varepsilon) - \{\varepsilon\}) \cup Follow(A)$

$\because Follow(A) = First(S) \cup Follow(A) = \{a, b\}$

$\therefore Select(A \to \varepsilon) = (First(\varepsilon) - \{\varepsilon\}) \cup Follow(A) = \{a, b\}$

由于 $Select(S \to aAS) \cap Select(S \to b) = \Phi$

$Select(A \to bA) \cap Select(A \to \varepsilon) \neq \Phi$

故该文法不是LL(1)文法，不能用自顶向下分析技术。

举例：对输入串 ab 进行推导就可能产生错误。

4.2 LL(1)文法的判别

判别步骤：

1. 求出能推出ε的非终结符
2. 计算FIRST集
3. 计算FOLLOW集
4. 计算SELECT集
5. 判别是否是LL(1)文法

例5：设文法G[S]为：

S→AB

S→bC

A→ ε

A→b

B→ ε

B→aD

C→AD

C→b

D→aS

D→c

判别步骤：

1. 求出能推出ε的非终结符
2. 计算FIRST集
3. 计算FOLLOW集
4. 计算SELECT集
5. 判别是否是LL(1)文法

判断它是否是LL(1)文法。

NEXT

1. 求出能推出ε的非终结符

1) 建立一个以V_N的个数为上限的一维数组X[ ]，数组元素为V_N，对应每个V_N有一<u>标志位</u>；（该标志位记录能否推出ε，其值为：“未定”、“是”、“否”）

2) 置初值——将数组X[ ]中对应的每一个V_N的标记置为“未定”；

3) 删除所有右部含V_T的产生式，若某一V_N为左部的产生式全被删除，则将数组中对应的标记值改为“否”；

4) 若某一的某产生式右部为ε，则数组中对应的标记值为“是”，并删除该V_N为左部的所有产生式；

5) 扫描产生式右部的每个V_N，若该V_N在数组中对应标志为“是”，则删去该V_N，转6；若该V_N在数组中对应标志为“否”，则转7；

6) 若该V_N删去后，所在产生式右部为空，则该产生式左部的V_N在数组中对应的标志改为“是”，并删去该V_N为左部的所有产生式；否则转8；

7) 删去该产生式，若该产生式左部在剩余的产生式中是唯一的左部（即A→α…，再无其它“A→β”的产生式），则把数组中该V_N对应的标志改为“否”；

8) 返回5，直至扫描完一边文法的产生式后，数组中的标志不再改变。

1. 求出能推出ε的非终结符

（1）建数组X[4]：X[0]——S； X[1]——A； X[2]——B； X[3]——C； X[4]——D

（2）X[0]=“未定” X[1]=“未定” X[2]=“未定” X[3]=“未定” X[4]=“未定”

（3）删除右部含V_T的产生式

- S → AB
- A → ε
- B → ε
- C → AD
- “D”被全部删除，则x[4]=“否”
- **即：X[0]=“未定” X[1]=“未定” X[2]=“未定” X[3]=“未定” X[4]=“否”**

1. 求出能推出ε的非终结符

（4）删除产生式右部为 ε

- S → AB
- C → AD
- x[1]=“是”　　x[2]=“是”
- 即：X[0]=“未定” X[1]=“是” X[2]=“是” X[3]=“未定” X[4]=“否”

（5）step5：S→B ➔step 8 ➔step5：S→　　➔step6：x[0]=“是”

- C→AD　　　　　　　　　　C→AD
- 即：X[0]=“是” X[1]=“是” X[2]=“是” X[3]=“未定” X[4]=“否”
- ➔ step8 ➔step5：C→D ➔step8 ➔step5：C→　　➔step7：x[3]=“否”
- 即：X[0]=“是” X[1]=“是” X[2]=“是” X[3]=“否” X[4]=“否”
- ➔step5 ➔step8 ➔ 结束

1. 求出能推出ε的非终结符

S→AB
S→bC
A→ε
A→b
B→ε
B→aD
C→AD
C→b
D→aS
D→c

非终结符	S	A	B	C	D
初值	未定	未定	未定	未定	未定
第1次扫描		是	是		否
第2次扫描	是			否	

返回

2. 计算FIRST集

方法一：根据定义计算
方法二：关系图法

定义：

$First(\alpha)=\{a \mid \alpha \overset{*}{\Rightarrow} a\beta,\ a \in V_T,\ \alpha, \beta \in V^*\}$

若 $\alpha \overset{*}{\Rightarrow} \varepsilon$，则 $\varepsilon \in First(\alpha)$

求First(x)的算法：

1) 若 $x \in V_T$，则 $first(x)=\{x\}$
2) 若 $X \in V_N$，且有产生式 $X \rightarrow a\ldots$， $a \in V_T$，则 $a \in first(X)$
3) 若 $X \in V_N$，$x \rightarrow \varepsilon$，则 $\varepsilon \in first(X)$
4) 若 $X \in V_N$，且有产生式 $X \rightarrow Y_1Y_2\ldots Y_n$，其中 $Y_1, Y_2, \ldots Y_n$ 都 $\in V_N$
   - 当 $Y_1, Y_2, \ldots Y_{i-1}$ 都能推导出 ε 时 $(1 \le i \le n)$，
     则 $first(Y_1) - \{\varepsilon\} \in first(X)$
     $first(Y_2) - \{\varepsilon\} \in first(X)$
     …
     $first(Y_{i-1}) - \{\varepsilon\} \in first(X)$， $first(Y_i) \in first(X)$
   - 当 $Y_1, Y_2, \ldots Y_n$ 都能推导出 ε 时，
     则 $first(X) = (first(Y_1)-\{\varepsilon\}) \cup (first(Y_2)-\{\varepsilon\}) \cup \ldots\ldots \cup (first(Y_n)-\{\varepsilon\}) \cup \{\varepsilon\}$

2. 计算FIRST集

- 对符号串求first集算法：
- $\alpha = X_1\ X_2 \ldots\ X_n$
- 1. 若X_1不能推导出ε，则$first(\alpha)=first(X_1)$
- 2. 若对任何j，$\varepsilon \in first(X_j)$ $(1<=j=i-1\ ,2<=i<=n)$
- 则$first(\alpha)=\cup_{j=1}^{i-1}(first(X_j)-\{\varepsilon\})\ \cup\ first(X_i)$
- 3. 若任何j，$\varepsilon \in first(X_j)$ $(1<=j<=n)$
- 则$first(\alpha)=\ \cup_{j=1}^{n}first(X_j)\ \cup\ first(\varepsilon)$

2. 计算FIRST集

S→AB
S→bC
A→ε
A→b
B→ε
B→aD
C→AD
C→b
D→aS
D→c

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

First (A)={b, ε}

First (B)={a, ε}

First (C)=(First(A)-{ε})∪First(D)∪{b}={a,b,c}

First (D)={a,c}

First (AB)={a,b,ε}
First (ε)={ε}
First (aD)={a}
First (aS)={a}

First (bC)={b}
First (b)={b}
First (AD)={a,b,c}
First (c)={c}

返回

3. 计算FOLLOW集

方法一：根据定义计算
方法二：关系图法

1） 设S为开始符号，把{#}加入Follow(S)中（#为句子括号）

2） 若$A \rightarrow \alpha B \beta$，则把 First($\beta$)−{$\varepsilon$} 加入Follow(B)中，

如果$\beta \stackrel{*}{\Rightarrow} \varepsilon$，则把 Follow(A)也加入Follow(B)中。

3） 反复2，直到每个V_N的Follow集不再增大为止。

3. 计算FOLLOW集

S→AB
S→bC
A→ε
A→b
B→ε
B→aD
C→AD
C→b
D→aS
D→c

Follow(S)={#}∪Follow(D)
Follow(A)={a}∪{a,c}∪Follow(S)
Follow(B)=Follow(S)
Follow(C)=Follow(S)
Follow(D)=Follow(B)∪Follow(C)

Follow(S)= {#}
Follow(A)= {a,c,#}
Follow(B)= {#}
Follow(C)= {#}
Follow(D)= {#}

返回

4. 计算SELECT集

First(S)={a, b, ε}
First(A)={b, ε}
First(B)={a, ε}
First(C)={a, b, c}
First(D)={a, c}
First(AB)={a, b, ε}
First(AD)={a, b, c}

Follow(S)= {#}
Follow(A)= {a, c, #}
Follow(B)= {#}
Follow(C)= {#}
Follow(D)= {#}

定义：

对于产生式 $A \rightarrow \alpha$，$A \in V_N$，$\alpha \in V^*$

若 $\alpha \not\Rightarrow^{*} \varepsilon$，则 $Select(A \rightarrow \alpha)=First(\alpha)$

若 $\alpha \Rightarrow^{*} \varepsilon$，则 $Select(A \rightarrow \alpha)=(First(\alpha)-\{\varepsilon\}) \cup Follow(A)$

续上例：

S→AB
S→bC
A→ε
A→b
B→ε
B→aD
C→AD
C→b
D→aS
D→c

Select(S→AB)=(First(AB)−{ε})∪Follow(S)={a, b, #}
Select(S→bC)=First(bC)={b}
Select(A→ε)=(First(ε)−{ε})∪Follow(A)={a, c, #}
Select(A→b) =First(b)={b}
Select(B→ε)=(First(ε)−{ε})∪Follow(B)={#}
Select(B→aD)=First(aD)={a}
Select(C→AD)=First(AD)={a, b, c}
Select(C→b) =First(b)={b}
Select(D→aS)=First(aS)={a}
Select(D→c) =First(c)={c}

返回

5. 判别

Select(S→AB)= {a, b, #}   Select(S→bC)= {b}
Select(A→ε)= {a, c, #}   Select(A→b) = {b}
Select(B→ε)= {#}   Select(B→aD)= {a}
Select(C→AD)= {a, b, c}   Select(C→b) = {b}
Select(D→aS)= {a}   Select(D→c) = {c}

LL(1)文法：
左部相同的产生式的SELECT集的交集均为空。

S→AB
S→bC
A→ε
A→b
B→ε
B→aD
C→AD
C→b
D→aS
D→c

Select(S→AB)∩Select(S→bC)={b}≠Φ

Select(A→ε)∩Select(A→b)=Φ

Select(B→ε)∩Select(B→aD)=Φ

Select(C→AD)∩Select(C→b)={b}≠Φ

存在交集非空的SELECT集合，所以该文法不是LL(1)文法。

返回

练习：设文法G[A]为:

A→A∪B

A→B

B→B∩C

B→C

C→!D

C→D

D→(A)

D→i

判断它是否是LL(1)文法。

4.3 某些非LL(1)文法到LL(1)文法的等价变换

非LL(1)文法

- 若文法含有左公共因子，一定不是LL(1)文法
- 若文法含有直接或间接左递归，一定不是LL(1)文法

非LL(1)文法→LL(1)文法的等价变换：

- 提取左公共因子
- 消除左递归

注：左公共因子　　　$S \to a\alpha \mid a\beta$ 其中a即为左公共因子

4.3.1.1 提取左公共因子

对形如 $A \rightarrow \alpha\beta \mid \alpha\gamma$ 进行等价变换为：$A \rightarrow \alpha(\beta \mid \gamma)$

进一步变换：$A \rightarrow \alpha A'$

$A' \rightarrow \beta \mid \gamma$

对形如 $A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \ldots \mid \alpha\beta_n$ 进行等价变换为：

$A \rightarrow \alpha(\beta_1 \mid \beta_2 \mid \ldots \mid \beta_n)$

进一步变换(引入新非终结符A’)：

$A \rightarrow \alpha A'$

$A' \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$

注：若 $A' \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$ 中仍含左公共因子，再次提取，直至所有的产生式不再有左公共因子。

例6：文法 G[S] 为:

$S \rightarrow aSb$

$S \rightarrow aS$

$S \rightarrow \varepsilon$

解：

$$\left.\begin{array}{l} S \rightarrow aSb \\ S \rightarrow aS \end{array}\right\} \quad S \rightarrow aS(b \mid \varepsilon) \quad \left\{\begin{array}{l} S \rightarrow aS\ S' \\ S' \rightarrow b \mid \varepsilon \end{array}\right. \quad \begin{array}{l} S \rightarrow aS\ S' \\ S \rightarrow \varepsilon \\ S' \rightarrow b \\ S' \rightarrow \varepsilon \end{array}$$

$$S \rightarrow \varepsilon \qquad S \rightarrow \varepsilon \qquad S \rightarrow \varepsilon$$

结论1：文法中<u>不含左公共因子</u>只是LL(1)文法的<u>必要条件</u>。

即：

- LL(1)文法一定不含左公共因子
- 不含左公共因子的文法不一定是LL(1)文法

4.3.1.2 提取隐含的左公共因子

1) 隐式 变 显式：右部以V_N开始的产生式，用该V_N对应的产生式进行相应替换

2) 再用一般形式进行提取：

$A \rightarrow \alpha\beta_1 | \alpha\beta_2 | \dots | \alpha\beta_n$ 等价变换为：

$A \rightarrow \alpha A'$

$A' \rightarrow \beta_1 | \beta_2 | \dots | \beta_n$

例7：文法G[S]为:

S→aSd	S→aSd	S→aS(d|c)	S→aSA’	S→aSA’
S→Ac	S→aSc	S→bc	S→bc	S→bc
A→aS	S→bc	A→aS	A’→d	A’→d
A→b	A→aS	A→b	A’→c	A’→c
	A→b		A→aS	
			A→b	

A是多余产生式！

4.3.1.3 不能在有限步骤内提取完左公共因子的文法

例8：文法G[S]为：

S→Ap	S→aApp	S→a(App|Bqq)	S→aS’
S→Bq	S→dp	S→dp	S’→App
A→aAp	S→aBqq	S→eq	S’→Bqq
A→d	S→eq	A→aAp	S→dp
B→aBq	A→aAp	A→d	S→eq
B→e	A→d	B→aBq	A→aAp
	B→aBq	B→e	A→d
	B→e		B→aBq
			B→e

继续替换S’产生式右部的A和B，
只能使产生式无限地增加下去。

结论2：不是所有文法，都能在有限的步骤内提取完“左公共因子”。

4.3.2 消除左递归

✓ 文法提取左公共因子后，并不一定是LL(1)文法。
只有不含空产生式，且无左公共因子，且无左递归时，文法才是LL(1)文法。否则需要进行判别。

左递归的形式：

- 直接左递归

  $A \rightarrow A\beta \qquad A \in V_N, \quad \beta \in V^*$

- 间接左递归

  $A \rightarrow B\beta$

  $B \rightarrow A\alpha \qquad A, B \in V_N, \quad \alpha, \beta \in V^*$

4.3.2.1 消除直接左递归

把“直接左递归”改为“右递归”：

$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_m$

$A \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$

改写为：

$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A'$

$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A' \mid$

$A' \rightarrow \varepsilon$

例9：消除文法G[S] 的左递归。

$S \rightarrow Sa$

$S \rightarrow b$

解：左递归文法改写为：

$S \rightarrow bS'$

$S' \rightarrow aS'$

$S' \rightarrow \varepsilon$

4.3.2.2 消除间接左递归

- 先把“间接左递归”变为“直接左递归”
- 再将“直接左递归”化为“右递归”：

$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_m$

$A \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$

改写为：

$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A'$

$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A' \mid$

$A' \rightarrow \varepsilon$

例10：消除文法G[A] 的间接左递归。

$A \rightarrow aB$
$A \rightarrow Bb$
$B \rightarrow Ac$
$B \rightarrow d$

“间接”变“直接”

$A \rightarrow aB$
$A \rightarrow Acb$
$A \rightarrow db$
$B \rightarrow Ac$
$B \rightarrow d$

“左递归”化为“右递归”

$A \rightarrow aBA' \mid dbA'$
$A' \rightarrow cbA' \mid \varepsilon$
$B \rightarrow Ac$
$B \rightarrow d$

3 消除文法中一切左递归

```
以某种顺序将V_N的排序为：A_1，A_2，…，A_n
for(i=1;i<=n;i++)
{   for(j=1;j<=i-1;j++)   //以A_1，…，A_{i-1}的产生式代入A_i产生式
    {     若Aj的产生式为：A_j→δ_1|δ_2|…|δ_k
          则形如A_i→A_j γ 的产生式变为：A_i→δ_1 γ|δ_2 γ|…|δ_k γ
    }
    消除A_i中的一切直接左递归
}
删除多余产生式
```

例11：消除文法G[A] 的一切左递归。

$S \rightarrow Qc|c$

$Q \rightarrow Rb|b$

$R \rightarrow Sa|a$

例11：消除文法G[A] 的一切左递归。

$$S \rightarrow Qc \mid c$$
$$Q \rightarrow Rb \mid b$$
$$R \rightarrow Sa \mid a$$

解：非终结符排序为：S,Q,R

(1)对于S：无直接左递归（不用消除）

(2)对于Q：右部不含S开头的产生式

无直接左递归（不用消除）

(3)对于R：右部含S开头的产生式，则：

$S \rightarrow Qc$	$Q \rightarrow Rb$	$R \rightarrow Qca$
$S \rightarrow c$	$Q \rightarrow b$	$R \rightarrow ca$
		$R \rightarrow a$

右部含Q开头的产生式：

$S \rightarrow Qc$	$Q \rightarrow Rb$	$R \rightarrow Rbca$
$S \rightarrow c$	$Q \rightarrow b$	$R \rightarrow bca$
		$R \rightarrow ca$
		$R \rightarrow a$

消除直接左递归：

$S \rightarrow Qc$	$Q \rightarrow Rb$	$R \rightarrow (bca \mid ca \mid a)R'$
$S \rightarrow c$	$Q \rightarrow b$	$R' \rightarrow bcaR' \mid \varepsilon$

(4) 考察是否存在无用产生式：没有“无用产生式”，所以不用删除。

S→Qc | c, Q→Rb | b, R→Sa | a

另解：非终结符排序为：R, Q, S

(1)对于R：无直接左递归（不用消除）

(2)对于Q：右部含R开头的产生式：

S→Qc **Q→Sab** **R→Sa**

S→c **Q→ab** **R→a**

Q→b 无直接左递归（不用消除）

(3)对于S：右部不含R开头的产生式，右部含Q开头的产生式

S→Sabc **Q→Sab** **R→Sa**

S→abc **Q→ab** **R→a**

S→bc **Q→b**

S→c

消除直接左递归：

S→(abc | bc | c)S’ **Q→Sab** **R→Sa**

S’→abcS’ | ε **Q→ab** **R→a**

Q→b

(4) 考察是否有“无用产生式”：Q、R的产生式是无用产生式，删除！！

最终的产生式为：

S→abcS’ **S’→abcS’**

S→bcS’ **S’→ε**

S→cS’

4.5 确定的自顶向下分析方法

1. 递归子程序法:

- 主要思想:

  对文法中每个非终结符编写一个递归过程，每个过程的功能是识别由该非终结符推出的串，当某非终结符的产生式有多个候选时能够按LL(1)形式可唯一地确定选择某个候选进行推导。

- 优点:

  简单、直观、易于构造。（PL/0的语法分析）

- 缺点:

  对文法要求高，必须满足LL(1)文法;

  由于递归调用多，速度慢，占用空间多。

4.5 确定的自顶向下分析方法

2. 预测分析法:

- 预测分析器的组成:

  预测分析程序、先进后出栈、预测分析表

- 预测分析法的步骤:

  （1）提取左公共因子，消除左递归

  （2）判断文法是否为LL(1)文法

  （3）若是，构造预测分析表;

  否则，不能进行“确定的自顶向下”分析

  （4）预测分析程序根据“预测分析表”并利用“分析栈”，对输入串进行分析

例12：文法G[E]：　　$E \to E+T \mid T$　　构造预测分析表。

$T \to T*F \mid F$

$F \to i \mid (E)$

解：

（1）消除左递归：

V_N排列为 E, T, F

消除E一切直接左递归：

$E \to TE'$	$T \to T*F$	$F \to i$
$E' \to +TE' \mid \varepsilon$	$T \to F$	$F \to (E)$

消除T的一切直接左递归：

$E \to TE'$	$T \to FT'$	$F \to i$
$E' \to +TE' \mid \varepsilon$	$T' \to *FT' \mid \varepsilon$	$F \to (E)$

F没有左递归。

文法无左公共因子。

所以，提取左公共因子和消除左递归后的文法为：

$E \to TE'$	$T \to FT'$	$F \to i$
$E' \to +TE'$	$T' \to *FT'$	$F \to (E)$
$E' \to \varepsilon$	$T' \to \varepsilon$	

（2）判断改写后的文法是否为LL(1)文法：

a) 可推导出 ε 的V_N表：

E	E'	T	T'	F
否	是	否	是	否

b) 求First集：

First(E)={ i , (}

First(E')={ + , ε }

First(T)={ i , (}

First(T')={ * , ε }

First(F)={ i , (}

First(TE')=First(T)={ i , (}

First(FT')=First(F)={ i , (}

c) 求Follow集：

Follow(E)= { # ,) }

Follow(E')=Follow(E)∪Follow(E') = {# ,)}

Follow(T)=(First(E')-{ ε }) ∪Follow(E')={+, #,)}

Follow(T')= Follow(T)∪Follow(T') = {+, #,)}

Follow(F)=(First(T')-{ ε })∪Follow(T)∪Follow(T')

={ *, +, #,) }

d) 求各产生式的SELECT集：

SELECT(E→TE’) = First(TE’) = { i , (}

SELECT(E’→+TE’) = First(+TE’) = { + }

SELECT(E’→ ε) = Follow(E’) ={ #,)}

SELECT(T→FT’) = First(FT’) = { i , (}

SELECT(T’→*FT’) = First(*FT’) = { * }

SELECT(T’→ ε)= Follow(T’)={ +, #,)}

SELECT(F→(E)) = First((E)) = { (}

SELECT(F→i) = First(i) = { i }

e) 判定：

SELECT(E’→+TE’) ∩ SELECT(E’→ ε) = ф

SELECT(T’→*FT’) ∩ SELECT(T’→ ε) = ф

SELECT(F→(E)) ∩ SELECT(F→i) = ф

所以该文法是LL(1)文法，可以使用预测分析法。

构造预测分析表的方法：
对每个V_T或“#”用符号a表示。
若a∈SELECT（A→α），则把A-
（所有空白的M[A,a]表示出错。

Select(E→TE’) = { i , (}
Select(E’→+TE’) = { + }
Select(E’→ ε) ={ #,)}
Select(T→FT’) = { i , (}
Select(T’→*FT’) = { * }
Select(T’→ ε) ={ +, #,)}
Select(F→(E)) = { (}
Select(F→i) = { i }

（3）构造预测分析表：

	+	*	(	)	i	#
E			E→TE’		E→TE’	
E’	E’→+TE’			E’→ε		E’→ε
T			T→FT’		T→FT’	
T’	T’→ε	T’→*FT’		T’→ε		T’→ε
F			F→(E)		F→i	

	+	*	(	)	i	#
E			E→TE’		E→TE’	
E’	E’→+TE’			E’→ε		E’→ε
T			T→FT’		T→FT’	
T’	T’→ε	T’→*FT’		T’→ε		T’→ε
F			F→(E)		F→i	

（4）预测分析输入串

#i+i*i#

步骤	分析栈	剩余输入串	所用产生式
1	#E	i+i*i#	E→TE’
2	#E’T	i+i*i#	T→FT’
3	#E’T’F	i+i*i#	F→i
4	#E’T’i	i+i*i#	i 匹配
5	#E’T’	+i*i#	T’→ε
6	#E’	+i*i#	E’→+TE’
7	#E’T+	+i*i#	+匹配
8	#E’T	i*i#	T→FT’
9	#E’T’F	i*i#	F→i
10	#E’T’i	i*i#	i 匹配
11	#E’T’	*i#	T’→*FT’
12	#E’T’F*	*i#	*匹配
13	#E’T’F	i#	F→i
14	#E’T’i	i#	i 匹配
15	#E’T’	#	T’→ε
16	#E’	#	E’→ε
17	#	#	接受

4.4 不确定的自顶向下分析思想

非LL(1)文法不能用“确定的”自顶向下分析，但可以使用“不确定的”自顶向下分析(“带回溯的”自顶向下分析)

引起回溯的原因：

1. 由于左部相同的产生式的右部First集交集不为空。
2. 由于左部相同V_N的右部能推导出ε，且该V_N的Follow集中含有其右部First集的元素。
3. 由于文法中含有左递归。

引起回溯的原因：

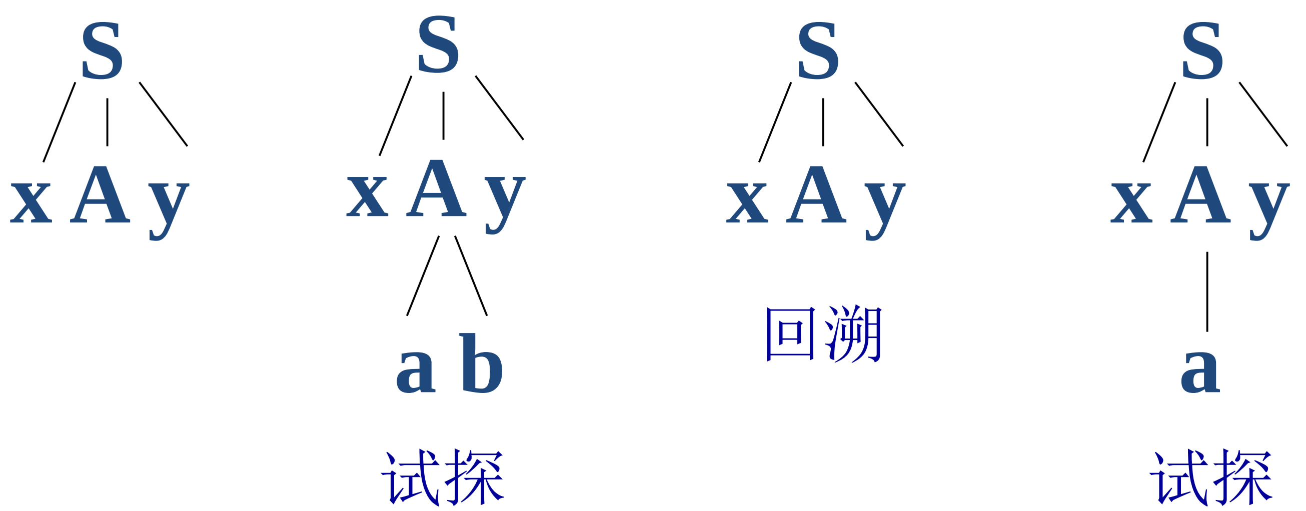
1. 由于左部相同的产生式的右部First集交集不为空。

例13：文法G[S]：

$S \rightarrow xAy$

$A \rightarrow ab \mid a$

分析输入串w=xay是否为该文法接受。



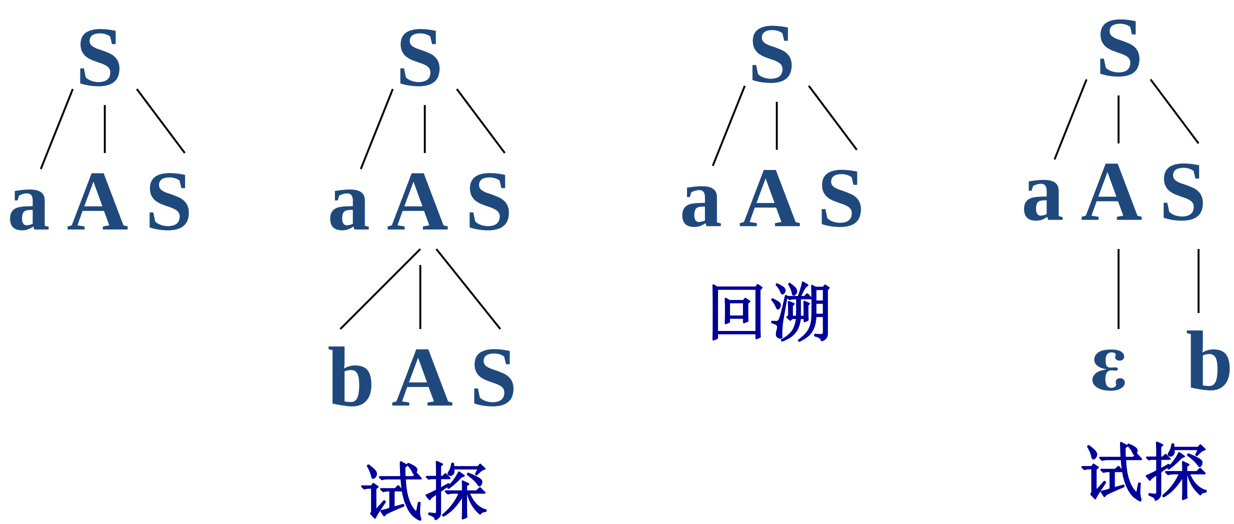
引起回溯的原因：

2. 由于左部相同V_N的右部能推导出 ε，
   且该V_N的Follow集中含有其右部First集的元素。

例14：设文法G[S] 为：

S→aAS
S→b
A→bAS
A→ ε

分析输入串 w=ab 是否为该文法接受。



引起回溯的原因：

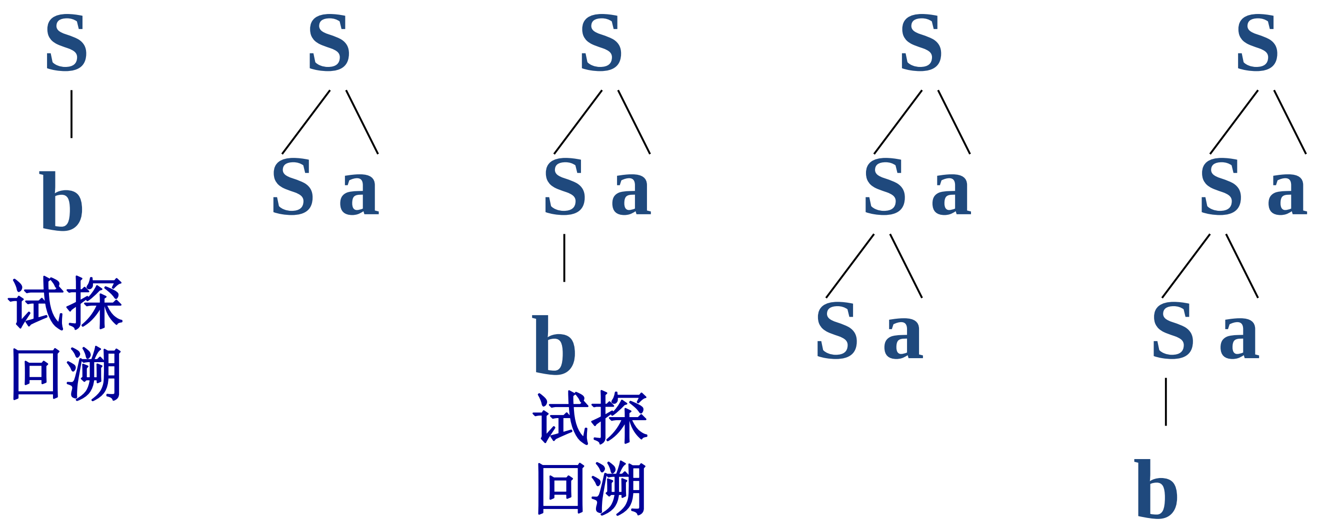
3. 由于文法中含有左递归。

例15：设文法G[S] 为:

S→Sa

S→b

分析输入串 w=baa 是否为该文法接受。



谢谢各位同学！