编译原理

田玲 教授、博导

lingtian@uestc.edu.cn

第七章 自上而下的语法分析

语法分析就是分析一个文法的句子结构；
主要有两类语法分析方法：

- ✓自上而下的语法分析
- ✓自下而上的语法分析

本章主要讨论自上而下的分析方法，并具体介绍递归下降分析法与预测分析法

语法分析器的功能

- 对经过词法分析得到的符号串，按文法规则进行判断，看它是否构成正确的句子；如果是不正确的句子，给出准确的错误信息，并给出相应的处理；对正确的句子，给出其**语法树**；

符号串 ⇒

从开始符号S出发，看能否找到一个最右推导，使得S⇒w；或从S出发，能否构成一个语法树，使得树的叶结点自左向右构成w；

从w开始，看能否找到一个规范归约，逐步向上归约，直到开始符号S。

- 给定文法G=(V_T,V_N,S，以
  - ✓**自上而下的语法分析**
  - ✓**自下而上的语法分析**

第二节 回溯分析法

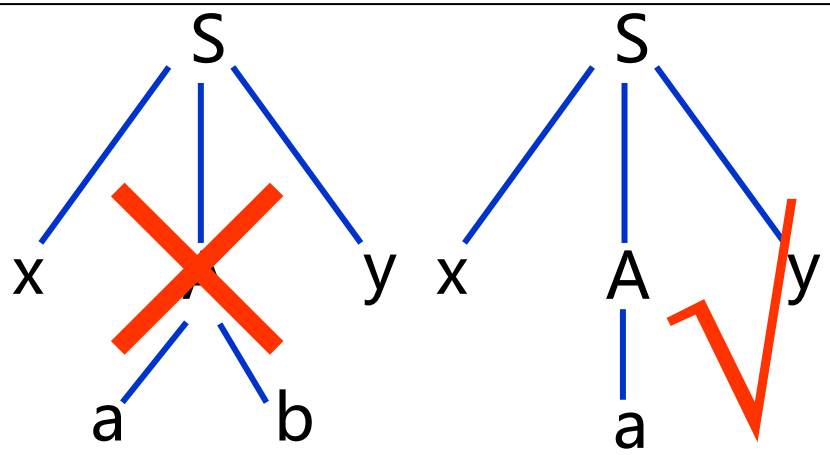
- 从文法的开始符号S出发，选取S的候选式进行推导，接着按最左推导进行下去；如果推导失败，再换用其他的候选式；若穷尽所有的候选式都失败，则表明w不是G的句子，w存在语法错语；

例：设有文法

S→xAy

A→ab | a

输入串为xay,它的自上而下分析过程如右图所示



注意：上述试探用所有的可能产生式进行推导的方法叫回溯，是一种低效的方法。

第二节 回溯分析法

- **回溯分析法的问题：**

  在文法G中，A有两个候选式A→ab和A→a，它们都能匹配输入串中的a，但选择前一个候选式造成虚假匹配；回溯的原因，在于候选式有公共左因子；

- **消除回溯：提取产生式的公共因子**

例：将文法

S→xAy

A→ab | a

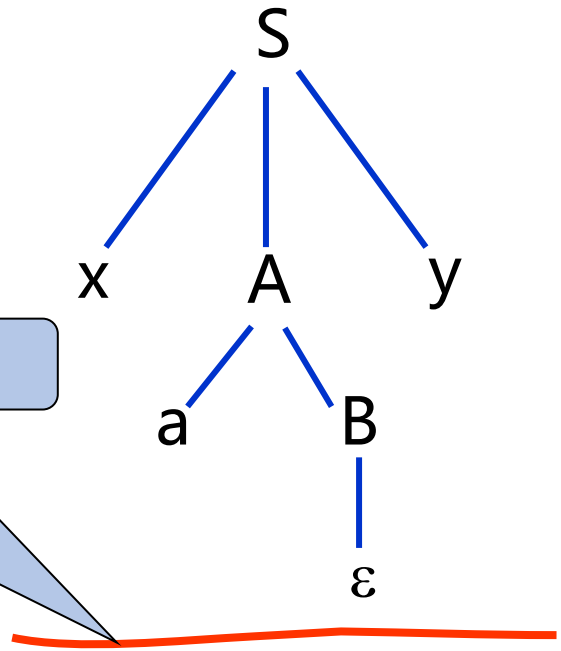
改造成：

S→xAy

唯一的推导过程，不用回溯。

B→b | ε

xay的推导过程如右图



第二节 回溯分析法

❑ 提取产生式公共因子的一般方法

产生式：

$A \rightarrow \alpha\beta_1|\alpha\beta_2|...|\alpha\beta_n|\delta$

提取公共左因子α

得到等价产生式：

$A \rightarrow \alpha B|\delta$

$B \rightarrow \beta_1|\beta_2|...|\beta_n$

注意：对一个文法G的所有非终结符的产生式提取公共左因子，可使该文法的分析过程没有回溯。

第二节 回溯分析法

另一个
严重问题

如果文法中有如下的产生式或推导时，怎么办：
$A \rightarrow A\alpha$ 或
$A \overset{+}{\Rightarrow} A\alpha$

它会产生如下的无限推导，而不能匹配任何输入字符:

$A \overset{+}{\Rightarrow} A\alpha \overset{+}{\Rightarrow} A\alpha\alpha \overset{+}{\Rightarrow} A\alpha\alpha\alpha \overset{+}{\Rightarrow} \ldots \overset{+}{\Rightarrow}$

- **左递归：以上的情况称为左递归；**
  - ✓**直接左递归：在文法中直接出现左递归，即有形如 $A \rightarrow A\alpha$ 的产生式；**
  - ✓**间接左递归：通过推导产生的左递归；**

例：文法$S \rightarrow Aa, A \rightarrow Sb|c$无直接左递归，有间接左递归：

$S \overset{+}{\Rightarrow} Aa \overset{+}{\Rightarrow} Sba \overset{+}{\Rightarrow} Aaba \overset{+}{\Rightarrow} Sbaba \overset{+}{\Rightarrow} \ldots$

第二节 回溯分析法

如何消除
左递归?

- 消除**直接左递归**:
  将左递归的产生式改写成等价的右递归的产生式;

例:

产生式:
$I \rightarrow IL|ID|L$

改写为

$I \rightarrow LI'$
$I' \rightarrow LI' |DI' |\varepsilon$

一般地，设文法G的直接左递归产生式为:

$$A \rightarrow A\alpha_1|A\alpha_2|...|A\alpha_n|\beta$$

其中为不含左递归的产生式，转换后的文法G′ 对应A的产生式为:

$$A \rightarrow \beta A'$$

$$A' \rightarrow \alpha_1 A' |\alpha_2 A' |...|\alpha_n A' |\varepsilon$$

第二节 回溯分析法

❑ 消除间接左递归：

算法

1.将文法G的所有非终结符按任一顺序排列，设为$A_1,\ldots,A_n$

2.执行下面算法，消除可能的左递归：

```
for i:=1 to n do
  for j:=1 to i-1 do
    begin
      把一个形如Ai→Ajα的产生式改写为
        Ai→δ1α|δ2α|...|δkα
          其中Aj→δ1|δ2|...|δk是Aj的所有产生式;
      消除Ai产生式的直接左递归;
    end
```

3.化简：删除多余产生式，即在从文法开始符号的任何推导中都不会出现的非终结符的产生式；

注意：上述算法不允许G中包含ε产生式，如有ε产生式，就应先消除它们。

第二节 回溯分析法

例：消除下面文法的左递归
S→Qc|c
Q→Rb|b
R→Sa|a

① i=1,j=0，没有操作

② i=2,j=1，
✓ A_2=Q,A_1=R,有Q→Rb|b
✓ 把R→Sa|a代入上式得:
Q→Sab|ab|b
✓ 该式无直接左递归

③ i=3,j=1，
✓ A_3=S,A_1=R，无相应产生式，
没有操作

注意：非终结符次序不同，最后的文法可能不同，但等价

步骤

1.**将非终结符按R，Q，S排列**

2.**执行算法**

④ i=3,j=2，
✓ A_3=S,A_2=Q，S→Qc|c
✓ 将上面得到的Q的产生式代入得:
S→Sabc|abc|bc|c
✓ 消除上式的直接左递归得:
S→abcS′ |bcS′ |cS′
S′ →abcS′ |ε

3.**化简**：经过上述算法，Q和R的产生式为多余的产生式，可删除。最后得:
S→abcS′ |bcS′ |cS′
S′ →abcS′ |ε

第三节 递归下降分析法

```
viod E()
{
   T();
   E’ ()
}

viod T()
{
   F();
   T’ ()
}

void F()
{
 if (sym== ‘(’ )
  {  advance();
     E();
     if (sym== ‘)’ ) advance();
     else error();
     }
     else if (sym== ‘i’ ) advance();
     else error(); }
```

```
viod E’ ()
{
if(sym== ‘+’ )
{
                    ();
                    ();
                    }
```

```
viod T’ ()
{
If(sym== ‘*’ )
{
   advance();
   F();
   T’ ();} }
```

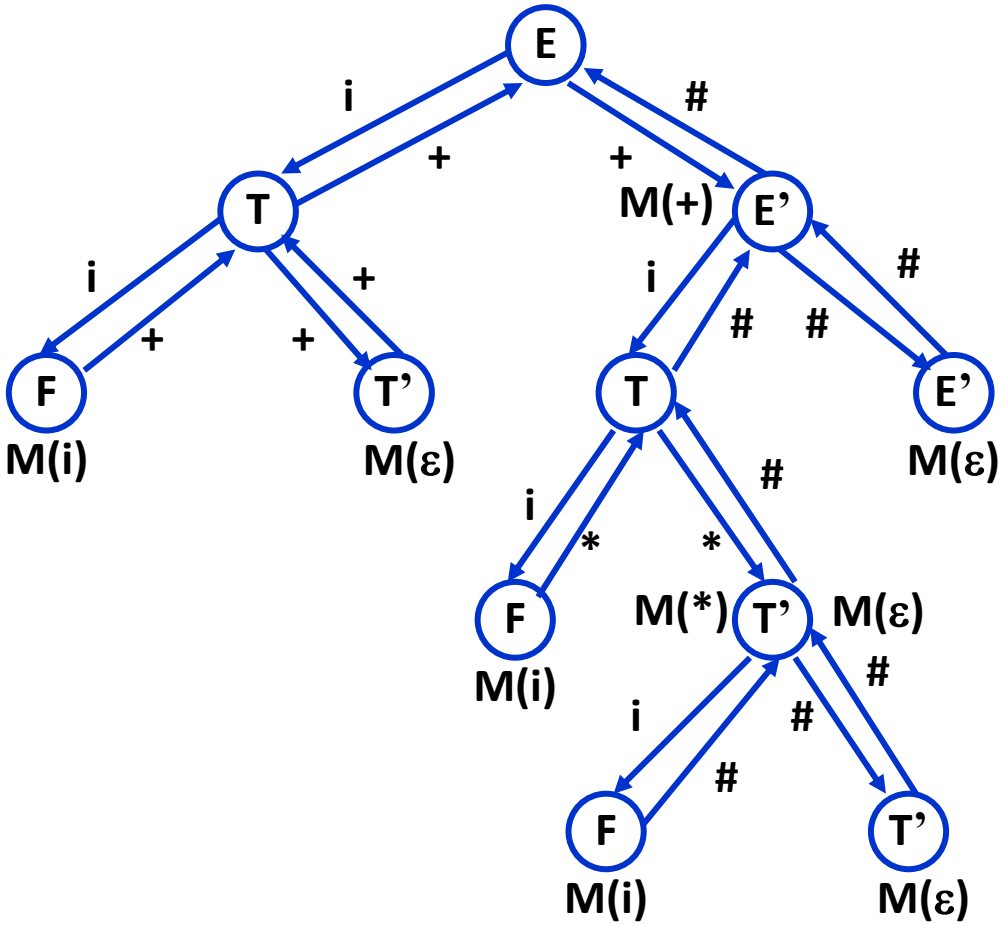
ip指向的那个输入符号

把输入串指针ip移向下一输入符号

error为出错处理程序

第三节 递归下降分析法

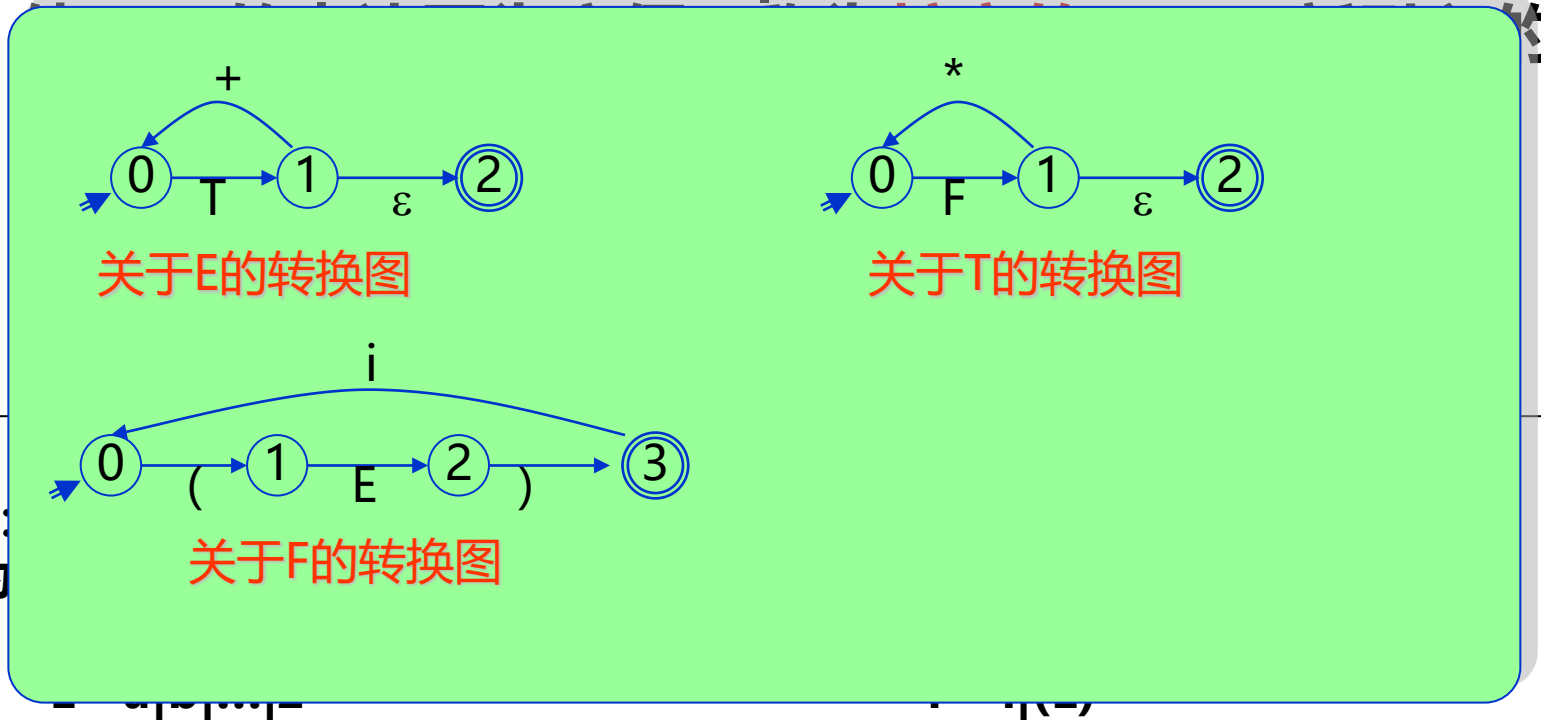
上面递归程序对输入串i+i*i的执行过程：



第三节 递归下降分析法

- **扩充的BNF**

  在原有元语言符号→,<,>,|的基础上增加新的元语言符号，



例1
示为

D→0|1|...|9

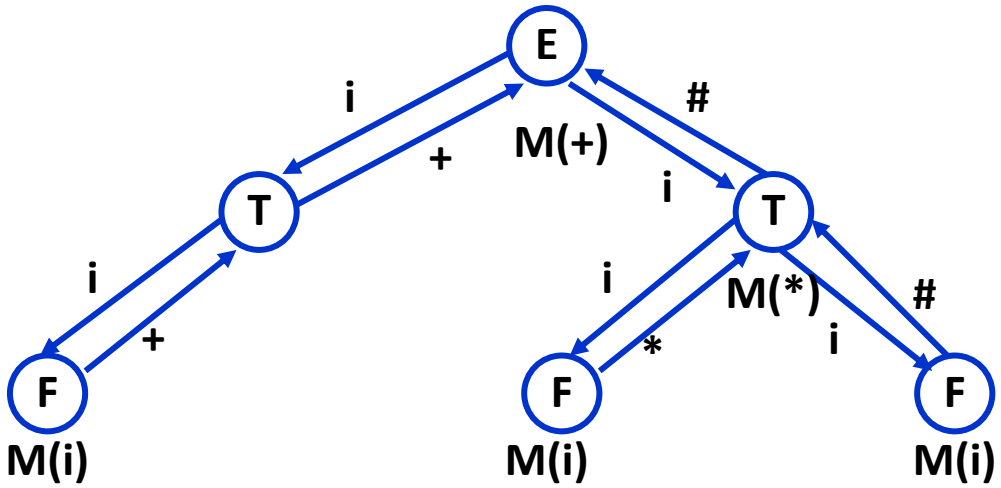
注意：扩充的BNF可以用转换图来表示

第三节 递归下降分析法

例：利用扩充BNF写成的递归下降分析器

```
procedure E;
 begin
   T;
   while sym='+' do
     begin advance; T end
 end

procedure T;
 begin
   F;
   while sym='*' do
     begin advance; F end
 end
```



i+i*i的分析过程

第四节 预测分析法

- **预测分析法**：预测分析（forecasting parse）是一种表驱动方法，由下推栈、预测分析表和控制程序组成。它实际上一种下推自动机的实现模型。

- **预测分析表**：
  - ✓形式:M[A,a]矩阵，$A \in V_N$,a
  - ✓内容:A→α或出错标志(空白)

表明A面临a时应采用该产生式进行推导

词法分析给每个语句加入的句末符

例：

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

第四节 预测分析法

❑预测分析器的执行算法：

预测分析程序总是按照栈顶符号和当前输入符号a行事。分析开始时，栈底先放一个#，然后放进文法的开始符号。对任何(X,a)，总控程序执行下述动作之一：

① 若X=a=‘#’,分析成功，且分析过程终止；

② 若X=a≠‘#’,把X从栈顶上托，并让a指向下一个输入符号；

③ 若X是非终结符，则查看分析表M，若M[X,a]中存放着X的一个产生式，则上托X，并把产生式右部符号按逆序推进栈；若M[X,a]是“出错”标志，则调用出错处理程序error;

例：文法G_2'

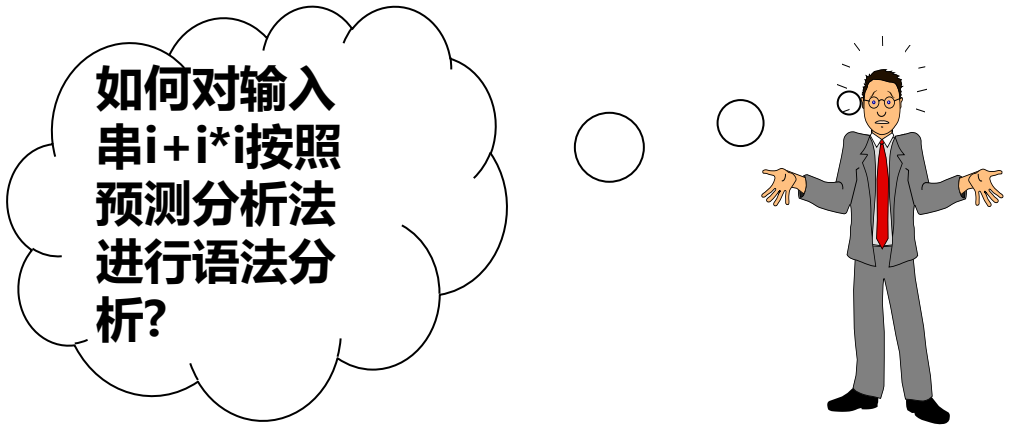
E→TE′

E′ →+TE′ |ε

T→FT′

T′ →*FT′ |ε

F→(E)|i



第四节 预测分析法

输入串i+i*i的预测分过程：

下推栈	输入串	查分析表
#E	i+i*i#	E→TE’
#E’T	i+i*i#	T→FT’
#E’T’F	i+i*i#	F→i
#E’T’i	i+i*i#	
#E’T’	+i*i#	T’→ε
#E’	+i*i#	E’→+TE’
#E’T+	+i*i#	
#E’T	i*i#	T→FT’
#E’T’F	i*i#	F→i
#E’T’i	i*i#	
#E’T’	*i#	T’→*FT’
#E’T’F*	*i#	
#E’T’F	i#	F→i
#E’T’i	i#	
#E’T’	#	T’→ε
#E’	#	E’→ε
#	#	结束

第四节 预测分析法

关键问题

给定一个文法，如何构造它的预测分析表？

- **FIRST集**：

  α所有可能推导出的首终结符（或ε）的集合

  定义:对$\alpha \in (V_T \cup V$
  $FIRST(\alpha) = \{a \mid \alpha \Rightarrow a\ldots, a \in V_T\}$
  若$\alpha \Rightarrow^{*} \varepsilon$,则$\varepsilon \in FIRST(\alpha)$

- **FOLLOW集**：

  所有句型中紧跟在A后面出现的终结符的集合

  定义:对$A \in V_N$,有
  $FOLLOW(A) = \{a \mid S \Rightarrow \ldots Aa\ldots, a \in V_T\}$
  若$S \Rightarrow^{+} \ldots A$, 则$\# \in FOLLOW(A)$,其中S为开始符号

注意：FIRST集一般称为**首符集**，FOLLO集一般称为**跟随符集**

第四节 预测分析法

- **FIRST集的构造方法**：

  对每个文法符号$X \in V_T \cup V_N$，连续使用如下规则，直到每个FIRST(X)不再增大。

  ①$X \in V_T$, 则FIRST(X)={X};

  ②$X \in V_N$，且有若$X \to a\ldots, a \in V_T$，则把a加入FIRST(X)；若$X \to \varepsilon$也是产生式，则把ε也加入FIRST(X)；

  ③若$X \to Y\ldots$，$Y \in V_N$，则把FIRST(Y)-{ε}加入FIRST(X)；

  ④若$X \to Y_1 Y_2 \ldots Y_k$，且Y_1，Y_2，…Y_{i-1}都是非终结符，$Y_1 \ldots Y_{i-1} \Rightarrow^{+} \varepsilon$，则把所有$FIRST(Y_j)$-{ε}加入FIRST(X),其中$1 \le j \le i-1$；

对文法G的任何符号串$\alpha = X_1 X_2 \ldots X_n$构造FIRST(α)的方法：

① 首先把$FIRST(X_1)$-{ε}加入FIRST(α)；

② 若对任何$1 \le j \le i-1$, $\varepsilon \in FIRST(X_j)$,则把$FIRST(X_i)$加入FIRST(α)中；

③ 如果所有$FIRST(X_j)$均包含ε，则把ε加入FIRST(α)中

第四节 预测分析法

- **FOLLOW集的构造方法**：
  对文法G的每个文法符号A，连续使用如下规则，直到FOLLOW(A)不再增大。
  ①对文法的开始符号，把#加入FOLLOW(S)；
  ②对于产生式A→αBβ，把FIRST(β)-{ε}加入FOLLOW(B)
  ③A→αB 或者 A→αBβ 且 $\beta \overset{+}{\Rightarrow} \varepsilon$， 将 FOLLOW(A) 加 入 FOLLOW(B)；

例：文法G_2'
E→TE'
E' →+TE' |ε
T→FT'
T' →*FT' |ε
F→(E)|i
求所有非终结符的FIRST集和FOLLOW集

FIRST集的构造

1.由规则2
FIRST（F）={(,i}
FIRST(T')={*,ε}
FIRST(E')={+,ε}

2.由E→TE',T→FT'及规则（3）有
FIRST(E)=FIRST(T)=FIRST(F)
={(,i}

第四节 预测分析法

FOLLOW集的构造

1.由规则1
　　FOLLOW（E）={#}

2.由规则2及F→(E)有
　　FOLLOW（E）={#,)}

3.由规则2及T→FT’有
　　FOLLOW（F）={*}

4.由规则2及E→TE’有
　　FOLLOW（T）={+}

5.由规则3及E→TE’有
　　FOLLOW（E’）={#,)}
　　FOLLOW(T)={#,),+}

6.由规则3及T→FT’有
　　FOLLOW（T’）={#,),+}
　　FOLLOW(F)={#,),+,*}

最后结果

	FIRST	FOLLOW
E	(,i	),#
E’	+,ε	),#
T	(,i	+,),#
T’	*,ε	+,),#
F	(,i	*, +,),#

第四节 预测分析法

- **LL(1)文法**：
  设有文法G，若它的任一产生式$A\to\alpha_1|\alpha_2|\ldots|\alpha_n$,均满足:
  ①$FIRST(\alpha_i)\cap FIRST(\alpha_j)=\varnothing$,其中$i\neq j,i,j=1,2,\ldots,n$
  ②若$\alpha_i\overset{+}{\Rightarrow}\varepsilon$,则$FIRST(\alpha_j)\cap FOLLOW(A)=\varnothing$,$j=1,2,\ldots,n$且$j\neq i$
  则文法G称为**LL(1)文法**。

- **LL(1)文法与自上而下分析法的关系**：
  在自上而下分析时，若当前输入字符为a,分析栈待匹配的非终结符为A，$A\to\alpha_1|\alpha_2|\ldots|\alpha_n$，则当:
  ①$a\in FIRST(\alpha_i)$，或
  ②若$\alpha_i\overset{+}{\Rightarrow}\varepsilon$,$a\in FOLLOW(A)$
  则$A\to\alpha_i$便是惟一与a匹配的产生式，即LL（1）文法中的两个条件保证了自上而下匹配的唯一性。

第四节 预测分析法

- **预测分析表的构造**：
  设有文法G，构造FIRST(α)和FOLLOW(A)，然后执行如下算法
  ①对文法G的每个产生式执行(2)和(3)；
  ②对每个终结符a∈FIRST(α)，则把A→α放入M[A,a]；
  ③若ε∈FIRST(α)，则对所有b∈FOLLOW(A)，把A→α放入M[A,b]；

注意：上述算法可以对任何文法G构造预测分析表。有些文法的预测分析表可能有多重入口。如果文法G的预测分析表没有多重入口，则G一定是LL(1)文法；反之，G不是LL(1)文法。

例：

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→i			F→(E)		

第四节 预测分析法

- **非LL(1)文法**：
  并非所有文法G都可以改写成LL(1)文法，即使提取左公因子和消除左递归后，也不是LL(1)文法。

例：文法G_4
S→iCtSliCtSeSla

提取左因子

S→iCtSS’ |a
S’ →eSlε

结论：该文法不是LL(1)文法。事实上，不必构造预测分析表也可得出该结论，因为S’ →eS|ε，同时FIRST(eS)∩FOLLOW(S’)={e}≠∅;

S	S→a			S→iCtSS’		
S’			S’→eS S’→ε			S’→ε
C		C→b				

预测分析表

FIRST(S)={i,a}
FIRST(S’)={e,ε}
FIRST(C)={b}
FOLLOW(S)=FOLLOW(S’)={e,#}
FOLLOW(C)={t}

求预测分析表

作业

- 必做题：7-2，7-3，7-5
- 思考题：其余

$A \rightarrow aAa|\varepsilon$

（1）该文法是LL（1）文法吗？为什么？

（2）若采用LL（1）方法进行语法分析，如何得到该文法的LL（1）分析表？

（3）若输入符号串为“aaaa”,请给出语法分析过程。

已知文法G（S）

E → i | (F)
F → ET
T → +ET | ε
求文法的FIRST集、FOLLOW集，并构造预测分析表。该文法是LL（1）文法吗？