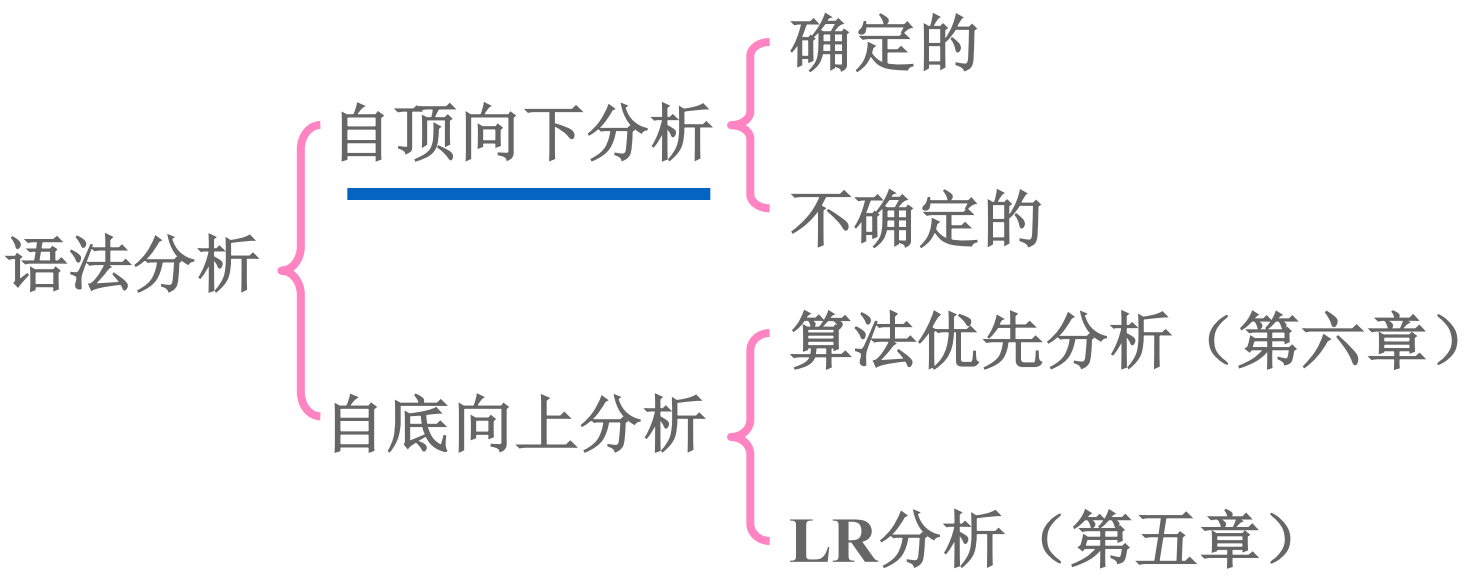
第四章

自顶向下语法分析方法

本章内容

- 语法分析的作用是识别由词法分析给出的单词序列是否是给定文法的正确句子

- 分类：



- 自顶向下基本思想：

  从文法的开始符出发企图推导出与输入的单词串完全相匹配的句子.

自上而下分析

- 定义：

  从文法的开始符号出发，反复使用文法的产生式，寻找与输入符号串匹配的推导。

- 语法树的构造：

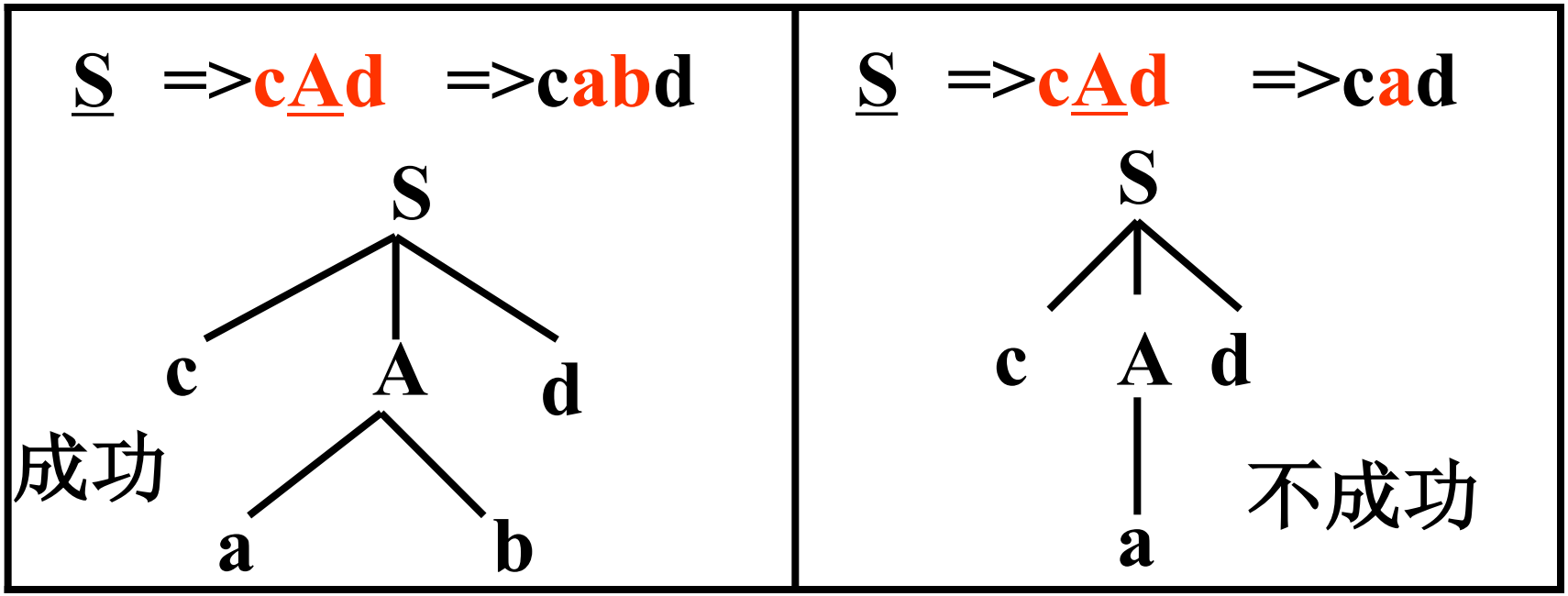
  将文法的开始符号作为语法树的根，向下逐步建立语法树，使语法树的末端结点符号串正好是输入符号串。

- 自上而下分析的主要问题

  选定产生式

自上而下分析

例 文法G：S → cAd
A → ab
A → a
识别输入串w=cabd是否为该文法的句子



4.1 确定的自顶向下分析思想

1 确定分析的条件

2 开始符号集FIRST(α)的定义

3 后跟符号集FOLLOW(A)的定义

4 选择集合SELECT(A→α)的定义

5 LL(1)文法的定义

1. 确定分析的条件

从文法的开始符出发，如能根据当前的输入符号（单词符号）唯一地确定选用哪个产生式进行推导，则分析是确定的。

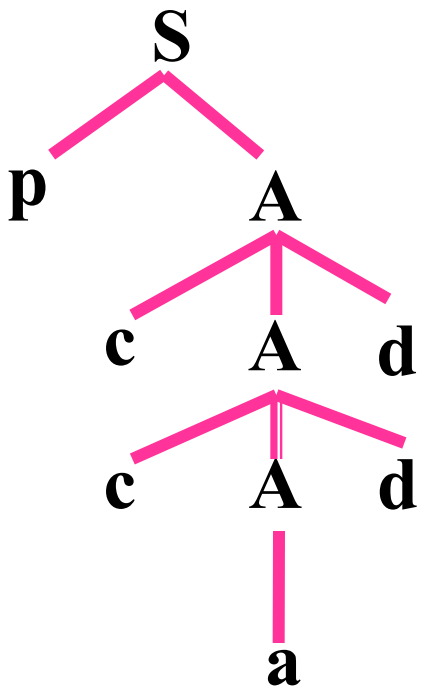
例1 设有文法$G_1[S]$:

S→pA|qB

A→cAd|a

B→dB|b

若输入串W=pccadd。[illegible]向下的推导过程为:



$G_1[S]$有如下特点：

(1) 每个产生式的右部由终结符开头;

(2) 同一非终结符的不同产生式的右部由不同的终结符开头。

对于这种文法，在推导过程可以根据当前的输入符号唯一确定选哪个产生式往下推导，即分析过程是确定的。

S =>pA =>pcAd =>pccAdd =>pccadd

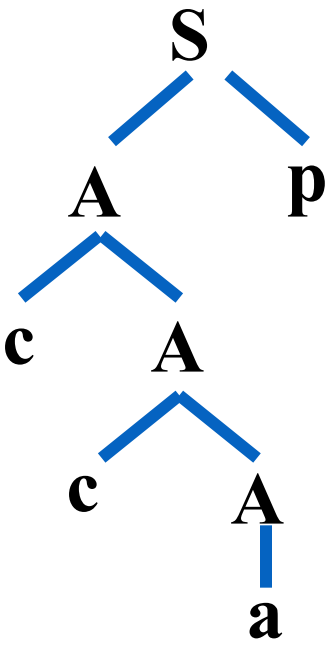
例2：设有文法$G_2[S]$为:

S→Ap|Bq

若输入串W=ccap,自顶向下的推导过程为：

A→a|cA

B→b|dB



该例说明，当

(1)产生式右部以终结符或非终结符开头（无空产生式）；

(2)同一非终结符的不同产生式的右部由不同的符号开头。

对于这种文法，在推导过程选用哪个产生式不直观，关键是判断产生式右部推出的开始符号（集），分析过程可能是确定的

S =>Ap =>cAp =>ccAp =>ccap

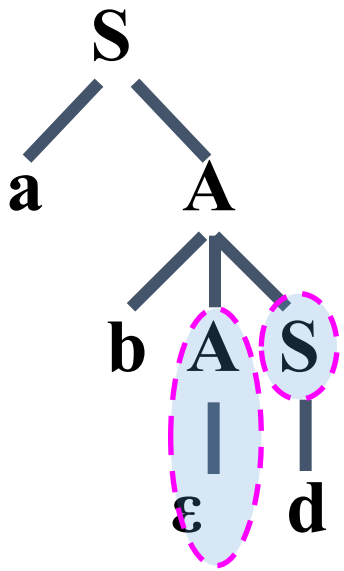
例3：设有文法$G_3[S]$

S→aA|d

A→bAS| ε

文法的特点——包含空产生式

若输入串W=abd，自顶向下的推导过程为：



对于空产生式左部的非终结符，关键是判断该非终结符的后跟符号（集），分析过程可能是确定的。

S =>aA =>abAS =>abS =>abd

- 要进行确定的自顶向下的分析，文法要满足一定的限制——即文法是LL(1)文法
- 先研究三个定义
  - 开始符号集FIRST
  - 后跟符号集FOLLOW
  - 选择集合SELECT

2. 开始符号集FIRST(β)的定义

- 定义:设G=(V_N, V_T, P, S)是上下文无关文法,

  $\alpha \rightarrow \beta$ ($\alpha \in V_N$, $\beta \in (V_N \cup V_T)^*$)

  FIRST(β) = {a | a $\in V_T$ 且$\beta \Rightarrow^*$ a.....}

  （若$\beta \Rightarrow^* \varepsilon$ 则规定$\varepsilon \in$FIRST(β)）

直观上说文法符号串β 的开始符号集是由β推导出的所有的终结符开头和可能的ε组成。

例文法$G_2[S]$:

S→Ap　　FIRST(Ap)={a,c}

S→Bq　　FIRST(Bq)={b,d}

A→a　　FIRST(a)={a }

A→cA　　FIRST(cA)={c}

B→b　　FIRST(b)={b}

B→dB　　FIRST(dB)={d}

结论一

针对无空产生式的文法G，同一非终结符的任两个产生式的右部串的First集无交集，即可进行确定的自顶向下分析。

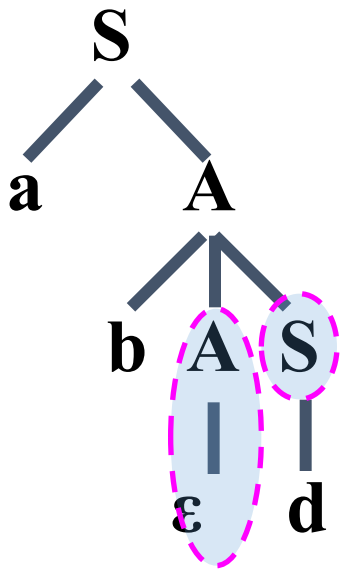
例3：设有文法$G_3[S]$

S→aA|d

A→bAS| ε

文法的特点——包含空产生式

若输入串W=abd，自顶向下的推导过程为：



对于空产生式左部的非终结符，关键是判断该非终结符的后跟符号（集），分析过程可能是确定的。

S =>aA =>abAS =>abS =>abd

3. 后跟符号集FOLLOW(A)的定义

- 定义

设G=(V_T, V_N, P, S)是上下文无关文法，

$B \rightarrow xAy$ $(A,B \in V_N \quad x,y \in (V_N \cup V_T)^*)$

FOLLOW(A)={a|S=>*…Aa…， $a \in V_T$}，

若有S=>* …A，则规定 # ∈FOLLOW(A)

（注：# 输入串#， ‘#’ 做为输入串的结束符）

直观上说,非终结符A的后跟符号集是由句型中紧跟A后的那些终结符（包括#）组成。

[例] 文法G_3[S]:　S→aA|d　　A→bAS|ε

- **FOLLOW(A)**
- **由S=>* aA 得　# ∈FOLLOW(A)**

  由S=>* abAS　=>* abAaA 得 a ∈FOLLOW(A)

  … =>* abAd　得 d ∈FOLLOW(A)

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

- **FOLLOW(S)**
- **由 S=>* S 得　# ∈FOLLOW(S)**

  由S=>aA=>abAS=>abbASS =>abbASaA 得 a ∈FOLLOW(S)

  …=>abbASd　得 d ∈FOLLOW(S)

  FOLLOW(S)={#,a,d}

解释

当A面对输入符a，在自顶向下的分析中应选择这样的产生式A→βi(βi可导出空串)进行推导：

- 若a ∈ First(βi),则 A→βi 可选
- 因 βi 可能导出空串，A自动获得匹配，输入符a有可能与A后的一个符号匹配，故当 a ∈ Follow(A) 时，产生式A→ βi 亦可选

结论一

针对无空产生式的文法G，同一非终结符的任两个产生式的右部串的First集无交集，即可进行确定的自顶向下分析。

结论二 针对有空产生式的文法G，对形如A→α，A→β的产生式（α，β不同时为空），如β⇏*ε，β⇒*ε，则α的开始符号集与β的开始符号集和后跟符号集均无交集，也可进行确定的自顶向下分析

[例] 文法$G_3[S]$:　S→aA|d　　A→bAS|ε

{S→aA} = First(aA) = { a }

{S→d} = First(d) = { d }

{A→bAS} = First(bAS) = { b }

{A→ε} = First(ε) + Follow(A) = {ε} + { #,a,d }

= { # , a , d }

回顾——开始符号集FIRST(β)的定义

- 定义:设G=(V_N, V_T, P, S)是上下文无关文法,

  $A \rightarrow \beta$ $(A \in V_N , \beta \in (V_N \cup V_T)^*)$

  $FIRST(\beta) = \{a \mid a \in V_T$ 且$\beta \Rightarrow^* a.....\}$

  （若$\beta \Rightarrow^* \varepsilon$, 则规定$\varepsilon \in FIRST(\beta)$）

直观上说文法符号串β 的开始符号集是由β推导出的所有的终结符开头和可能的ε组成。

回顾——后跟符号集FOLLOW(A)的定义

- 定义

设$G=(V_T, V_N, P, S)$是上下文无关文法，

$B \rightarrow xAy \quad (A,B \in V_N \quad x,y \in (V_N \cup V_T)^*)$

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

若有$S=>^* \ldots A$，则规定 $\# \in FOLLOW(A)$

（注：# 输入串#，　‘#’ 做为输入串的结束符）

直观上说,非终结符A的后跟符号集是由句型中紧跟A后的那些终结符（包括#）组成。

4. 选择集合SELECT(A→ β)的定义

- 定义

  对给定的上下文无关文法的产生式

  $A \rightarrow \beta$ ($A \in V_N$, $\beta \in V^*$)

- 若 $\beta \not\Rightarrow^* \varepsilon$,则SELECT(A→ β)=FIRST(β)
- 若 $\beta \Rightarrow^* \varepsilon$, 则SELECT(A→ β)

  $=(FIRST(\beta)-\{\varepsilon\}) \cup FOLLOW(A)$

解释

A的产生式$A\rightarrow\beta_1|\beta_2|\beta_3|\ldots$(A面对输入符a)在自顶向下的分析中：

- 对于$A\rightarrow\beta i$且$\beta i \neq>^*\varepsilon$, 若$a \in First(\beta i)$,

  则$A\rightarrow\beta i$可选

- 对于$A\rightarrow\beta j$且$\beta j =>^*\varepsilon$,

  若$a \in (First(\beta j)-\{\varepsilon\})\cup Follow(A)$,则$A\rightarrow\beta j$可选

[例] $G_3[S]$:

S→aA

S→d

A→bAS

A→ε

若β ≠>*ε,则SELECT(A→ β)=FIRST(β)

若β =>*ε, 则SELECT(A→ β)

=(FIRST(β)-{ε})∪FOLLOW(A)

SELECT(S→aA)=FIRST(aA)={a}

SELECT(S→d)=FIRST(d)={d}

SELECT(A→bAS)=FIRST(bAS)={b}

SELECT(A→ε)

=(FIRST(ε)-{ε})+ FOLLOW(A)={#,a,d}

结论三

同一非终结符的不同产生式$A\rightarrow\alpha$与$A\rightarrow\beta$，若$SELECT(A\rightarrow\alpha)\cap SELECT(A\rightarrow\beta)=\Phi$，则一定可以进行确定的自顶向下分析

5. LL(1)文法的定义

- 定义：

  上下文无关文法为LL(1)文法的充分必要条件是，对每个非终结符A的两个不同产生式$A\rightarrow\alpha$与$A\rightarrow\beta$，满足$SELECT(A\rightarrow\alpha)\cap SELECT(A\rightarrow\beta)=\Phi$

- LL(1)文法的含义是：

  - 第一个L——从左到右扫描输入串
  - 第二个L——分析过程用最左推导
  - (1)——表明只需向前看 1 个输入符号便可以决定选哪个产生式进行推导（类似地，LL(k) 文法则需要向前看 k 个输入符号才可以确定选用哪个产生式）

例 有文法G[S]为：

S→aAS	**SELECT(S→aAS)= {a}**
S→b	**SELECT(S→b)= {b}**
A→bA	**SELECT(A→bA)= {b}**
A→ε	**SELECT(A→ε)= {a,b}**

由于SELECT(A→bA)∩SELECT(A→ε)={b}≠Φ，所以文法G[S]不是LL(1)文法，当A遇输入符b时，不能确定选A→bA还是A→ε去推导。

判别一个上下文无关文法是否是LL(1)文法

1. 先找出ε的非终结符集T
2. 计算每个产生式右部β的FIRST(β)集
3. 计算每个非终结符A的FOLLOW(A)集
4. 计算每个产生式A→β的SELECT(A→β)集
5. 按LL(1)文法的定义判别

4.3 非LL(1)文法到LL(1)文法的变换

- 非LL(1)文法
  - 含有左公共因子的文法

若文法中含有形如：$A\rightarrow\alpha\beta|\alpha r$ 的产生式，称文法含有左公共因子。

显然，

$SELECT(A\rightarrow\alpha\beta)\cap SELECT(A\rightarrow\alpha r)\neq\phi$,文法不是LL(1)文法

- 含有左递归的文法

文法中只要含有下列形式的产生式,则称文法含有左递归：

a)$A \rightarrow A\beta$

b)$A \rightarrow B\beta$　　$B \rightarrow A\alpha$

在a)中含有左递归的产生式，称为直接左递归

在b)中虽然没有含左递归的产生式，

但$A \Rightarrow B\beta \Rightarrow A\alpha\beta$ 即$A \Rightarrow^{+} A\ldots$,称为间接左递归

以直接左递归为例，若有如下产生式

$A \rightarrow A\,\alpha \mid \beta$

其中α和β为任意语法符号串。

不难证明有下面关系式：

$Select(A \rightarrow A\,\alpha) \supseteq First(A\,\alpha) \supseteq First(\beta)$

$Select(A \rightarrow \beta) \supseteq First(\beta)$

故$A \rightarrow A\,\alpha$和$A \rightarrow \beta$ 的Select集相交，不是LL(1)文法

- 对非LL(1)文法进行等价变换
  - 提取左公共因子
  - 消除左递归

注意：变换后的文法不一定是LL(1)文法，文法不含左递归和左公共因子只是LL(1)文法的必要条件

1. 提取左公共因子

- 将产生式$A\rightarrow\alpha\beta|\alpha r$ 等价变换为:$A\rightarrow\alpha(\beta|\ r)$，$(\beta|\ r)$引入一新非终结符A’表示, 即得

$$A\rightarrow\alpha A'$$

$$A'\rightarrow\beta|\ r$$

- 一般形式：若$A\rightarrow\alpha\beta_1|\alpha\beta_2|\ldots|\alpha\beta_n|\gamma$

提取左公共因子, 即得

$$A\rightarrow\alpha A'\ |\gamma$$

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

若在β_i中仍含有左公共因子,可再次提取.

例　文法G1：

$S \rightarrow aSb|aS|\varepsilon$

提取左因子得: $S \rightarrow aS(b|\varepsilon)|\varepsilon$

引进新非终结符S’, 得:

$S \rightarrow aSS'|\varepsilon$

$S' \rightarrow b|\varepsilon$

2. 消除左递归

1) 消除直接左递归

- 文法G: $S \rightarrow Sa|b$

  可改写成 $S \rightarrow bS'$

  $S' \rightarrow aS'|\varepsilon$

- 一般情形:

  含直接左递归的文法G:

  $A \rightarrow A\alpha_1|A\alpha_2|\ldots|A\alpha_m|\beta_1|\beta_2|\ldots|\beta_n$

  消除左递归后改写成:

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

  $A' \rightarrow \alpha_1 A'|\alpha_2 A'|\ldots|\alpha_m A'|\varepsilon$

消除左递归

例:E → E+T|T

T →T*F|F

F →(E)| i

G[E]: (1) E → TE’ (2) E’ → +TE’| ε

(3) T → FT’ (4) T’ → *FT’ | ε

(5) F → (E)|i

2) 消除间接左递归

将间接左递归变成直接左递归，再消除

算法步骤：

1. 把文法的所有非终结符按任一顺序排列，

   如：A_1，A_2，…，A_i，…，A_k，…，A_n

2. 从A_1开始，按以下顺序处理A_k

① 消除左部为A_k的产生式的直接左递归

② 若左部为A_k的产生式的右部为非终结符$A_i(i<k)$开头（$A_k \rightarrow A_i\ldots$），则用左部为$A_i$的所有产生式的右部分别代替$A_k \rightarrow A_i\ldots$ 中的A_i，转至①

（*直至$A_1 \ldots A_n$的消除工作全部完成,退出2*）

3. 去掉无用产生式。

例 文法G：(1)S→Qc|c (2)Q→Rb|b (3)R→Sa|a

1. 将非终结符排序：R，Q，S

做为不一样，结果不一样，但表达的语言一致

2. 对R：产生式(3)不含直接左递归，所以保持不变

   对Q：把(3)代入(2)得(2’)Q→Sab|ab|b，无直接左递归

   对S：把(2’)代入(1)得 (1’) S→Sabc|abc|bc|c，有直接左递归，消除直接左递归得

   S→abcS’|bcS’|cS’　　S’ →abcS’|ε

   处理结果为：R→Sa|a　　Q→Sab|ab|b

   S→abcS’|bcS’|cS’　　S’ →abcS’|ε

3. 由于Q，R是不可到达的非终结符，其产生式应删除。

4. 最终得文法G’：S→abcS’|bcS’|cS’　S’→abcS’|ε

示例说明：

- 当非终结符顺序为R，Q，S，消除左递归的最终结果为：

  $S \rightarrow abcS' | bcS' | cS'$

  $S' \rightarrow abcS' | \varepsilon$

- 若非终结符顺序为S，Q，R，则消除左递归的最终结果为：

  $S \rightarrow Qc | c$

  $Q \rightarrow Rb | b$

  $R \rightarrow bcaR' | caR' | aR'$

  $R' \rightarrow bcaR' | \varepsilon$

- 结论：当非终结符的排序不同时，结果的产生式形式不同，但它们是等价的。

- **LL(1)文法**

  任两个产生式$A \to \alpha \mid \beta$都满足下列条件：
  - $\mathbf{FIRST}(\alpha) \cap \mathbf{FIRST}(\beta) = \varnothing$
  - 若$\beta \Rightarrow^* \varepsilon$，那么$\mathbf{FIRST}(\alpha) \cap \mathbf{FOLLOW}(A) = \varnothing$

- **LL(1)文法有一些明显的性质**
  - 没有公共左因子
  - 不是二义的
  - 不含左递归

4.4 确定的自顶向下分析方法

确定的自顶向下分析方法有：

1. LL(1)递归下降分析(recursive-descent parser)
2. LL(1)预测分析 (predictive parser)

两种方法都要求文法是LL(1)文法

4.4.1递归下降分析法

递归下降分析器：在消除了左递归和提取左公因子后，可以构造一个不带回溯的自上而下的分析程序。

这个程序由一组递归过程组成，每个过程对应文法的一个非终结符。

例：**G[E]:**

(1) E → TE’ (2) E’ → +TE’| ε

(3) T → FT’ (4) T’ → *FT’ | ε

(5) F → (E)|i

```
Procedure E;
    Begin
        T;E'
    End;
Procedure T;
    Begin
        F;T'
    End;
Procedure F;
   IF sym= 'i'  then advance
   else
       IF sym= '('  then
           Begin
               advance;
               E
       IF sym= ')'  then advance
               else  ERROR
           End;
```

```
Procedure E' ;
    IF sym= '+'  then
      Begin
          advance;
          T;E'
      End;
Procedure T' ;
    IF sym= '*'  then
      Begin
          advance;
          F;T'
      End;
```

特点：

- 优点：简单直观、易于构造
- 缺点：对文法要求高，必须满足LL(1)文法；

  递归调用多，速度慢，占用空间多
- 实用性：许多高级语言，如C，C++等编译系统常常采用此方法。

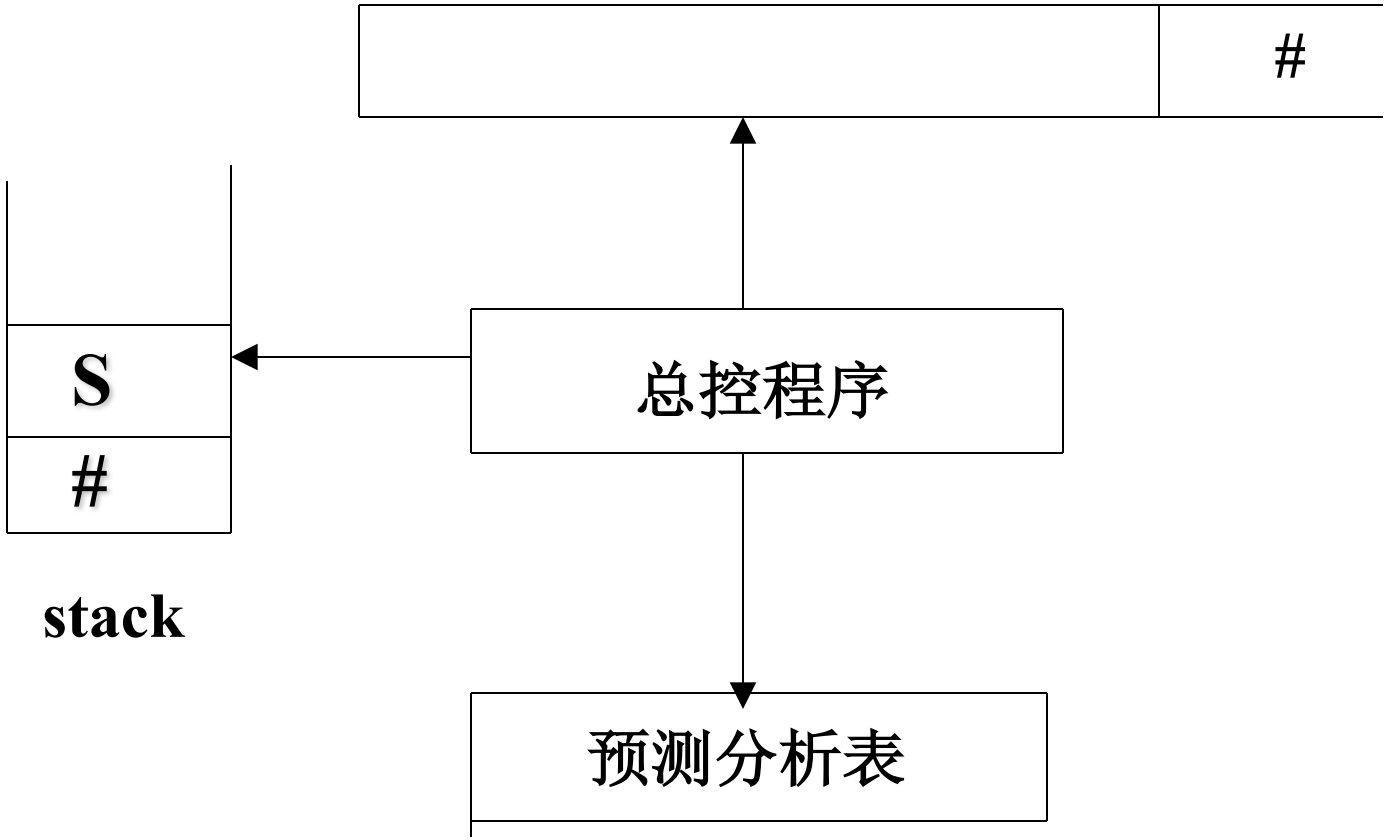
4.4.2 LL(1)预测分析器

一个预测分析器由三个部分组成：

- 预测分析程序：控制分析过程的进行。
- 分析栈：存放从文法开始符号出发的自顶向下推导过程中等待匹配的文法符号。开始时放入 '#' 和文法开始符，结束时栈应是空的。
- 预测分析表：是一个二维表，元素*M[A,a]*的内容是当非终结符A面临输入符号a（终结符或＃）时应选取的产生式；当无产生式时，元素*M[A,a]*为出错处理(error)。

预测分析方法（LL（1））

一、总控程序



1. 构造预测分析表

步骤：

(1) 把文法转变为LL(1)文法

(2) 求出每条产生式的SELECT集

(3) 依照SELECT集把产生式填入分析表

- 横坐标——终结符与‘#’
- 纵坐标——非终结符
- 交点*M[A,a]*
  - (A→β)放入*M[A,a]*——若a∈ SELECT(A→β)
  - 无产生式的*M[A,a]*——标记出错

[例] 算术表达式文法G

E→E+T | T

T→T*F | F

F→(E) | i

（1）消除G的左递归得到文法 G‘

E→TE '

E'→+TE' | ε

T→FT '

T'→*FT' | ε

F→(E) | i

（2）求出每个产生式的select集,G’是LL(1)文法

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 }

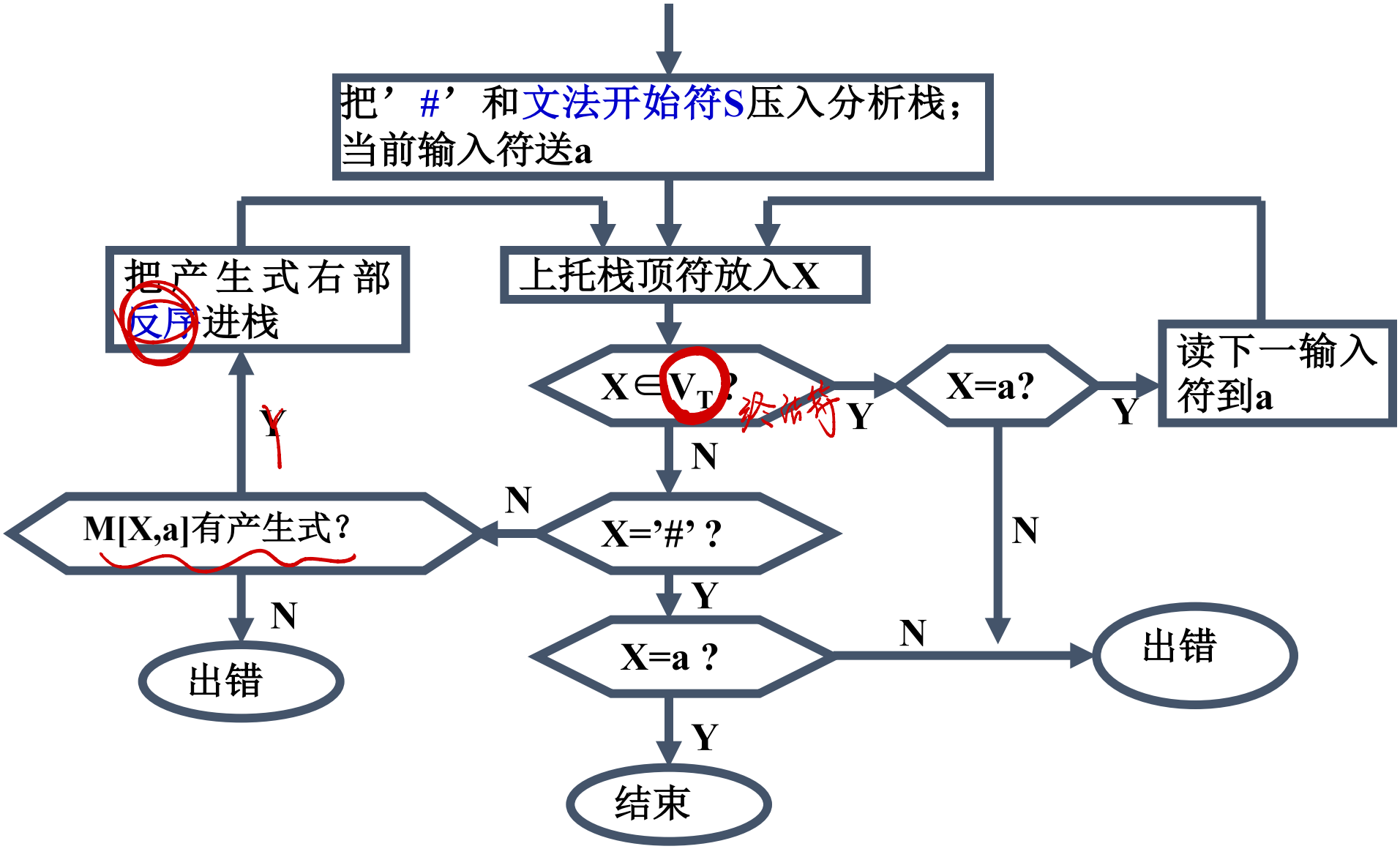
具有一定的二义性

（3）依照选择集合把产生式填入分析表

	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→ i			F→ (E)		

注：表中空白处为出错

2. 预测分析程序



预测分析程序工作过程

3. 输入串i+i*i#的分析过程

	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→ i			F→ (E)		

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

	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→ i			F→ (E)		

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

本章小结

- 两种自顶向下分析方法：
  - 递归子程序法
  - 预测分析法
- 一种文法：LL(1)文法
  - 非LL(1)到LL(1)的转换