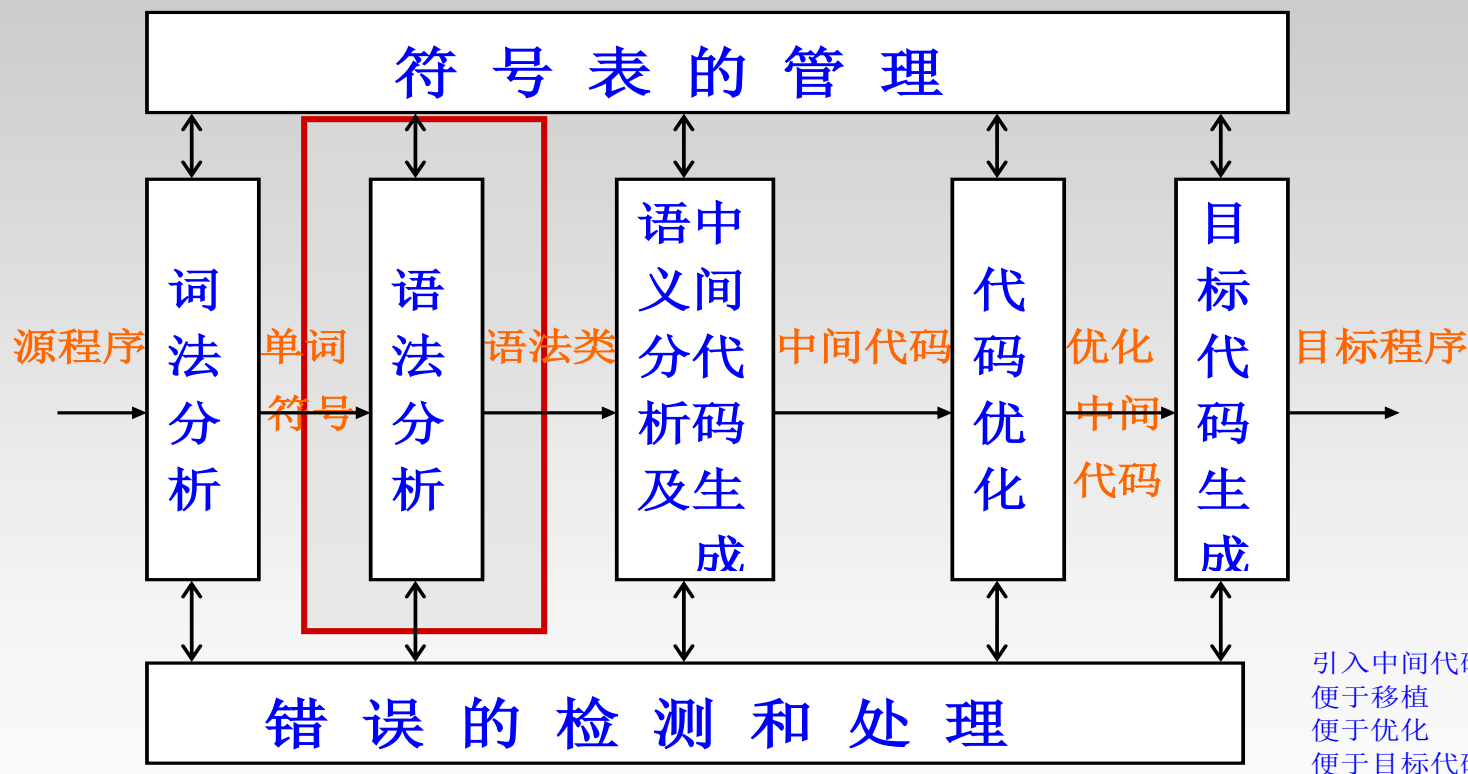


# 第四章 语法分析

## 典型编译程序的组成



# 语法分析程序

## ★ 基本任务

### 一、分析程序的语法结构

根据源语言的语法规则（CFG），分析语法结构，即分析如何由句子的单词组成各种语法范畴（如下标变量、各种表达式、各种语句、程序段或分程序，乃至整个源程序等

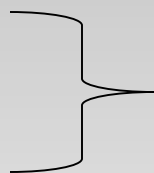
### 二、检查语法错误

### 三、输出语法树

# 语法分析的方法

## □ 自顶向下分析

- 递归下降分析
- LL (1) 分析



适用于LL(1)文法

## □ 自底向上分析

- 简单优先分析 → 适用于简单优先文法
- 算符优先分析 → 适用于算符优先文法
- LR分析法 → 适用于LR类文法

## ★效率低、代价高

- ①虚假匹配：如上例第一次a与a匹配是虚假的；
- ②影响语义分析：有些语法制导方式中，语义分析也得重来。

不能处理含左递归规则（直接或间接）的文法

例： $U \rightarrow Ua \mid b$

## 2、回溯问题

当规则中含有多个候选式时需一次次试探。

例： $S \rightarrow cAd$

$A \rightarrow ab \mid a$

匹配串cad

## 二、问题解决方法

### 1. 消除左递归（直接和间接）

#### (1) 直接左递归的消除：

用扩充的BNF表示{ },[ ],

引入新的非终结符号（提取公因子）

#### (2) 间接左递归的消除

消除左递归的算法（直接和间接均消除）

## (1) 直接左递归的消除

例： $U \rightarrow U\alpha | \beta$

方法一：扩充的BNF表示

$$U \rightarrow \beta \{ \alpha \}$$

方法二：引入新的非终结符

即  $U \rightarrow \beta U'$

$$U' \rightarrow \alpha U' | \varepsilon$$

## 提取公因子—使文法至多含有一个直接左递归的右部

$$U \rightarrow UV_1 | UV_2 | \dots | UV_n | x | y | \dots | z$$

$$U \rightarrow \underline{U(V_1 | V_2 | \dots | V_n)} \quad \underline{| x | y | \dots | z}$$

$$\text{设 } \alpha = (V_1 | V_2 | \dots | V_n)$$

$$\beta = x | y | \dots | z,$$

$$\text{即简写为 } U \rightarrow U\alpha | \beta$$

$$\begin{aligned} \text{则: } U &\rightarrow U' \{V_1 | V_2 | \dots | V_n\} \\ U' &\rightarrow x | y | \dots | z \end{aligned}$$

$$\text{或者: } U \rightarrow xU' | yU' | \dots | zU'$$

$$U' \rightarrow V_1 U' | V_2 U' | \dots | V_n U' | \varepsilon$$

★ 例1:  $T \rightarrow T * F | T / F | F$

变成:  $T \rightarrow T (*F | /F) | F$

消除左递归后得:  $T \rightarrow F \{ *F | /F \}$

★ 或:

$T' \rightarrow (*F | /F) T' | \varepsilon$

$T \rightarrow F T'$

★ 即:

$T \rightarrow F T'$

$T' \rightarrow *F T' | /F T' | \varepsilon$



## (2)间接左递归的消除

——消除左递归的算法(直接和间接均消除)

要求：文法不含  $A \rightarrow A$  和  $A \rightarrow \varepsilon$

原理： $S \rightarrow A\beta \mid \gamma$

$A \rightarrow S\alpha$

转换成  $S \rightarrow S\alpha\beta \mid \gamma$

间接左递归替换成直接左递归再消除

## 消除左递归的算法

①把G的非终结符号排序 $A_1, A_2, \dots, A_n$ ;

② for  $i:=1$  to  $n$  do

**Begin**

for  $j:=1$  to  $i-1$  do

把每个形如 $A_i \rightarrow A_j \gamma$ 的规则替换成

$A_i \rightarrow \delta_1 \gamma | \delta_2 \gamma | \dots | \delta_m \gamma,$

其中  $A_j \rightarrow \delta_1 | \delta_2 | \dots | \delta_m$  是当前 $A_j$ 的全部规则;

消除 $A_i$ 规则中的直接递归;

**End.**

③化简由②得的文法,即去掉多余规则;

**例：**  $G[S]: S \rightarrow Qc|c$

$Q \rightarrow Rb|b$

$R \rightarrow Sa|a$

**消除左递归**

**解：** (1)  $A_1=R, A_2=Q, A_3=S$

(2) ①  $i=1, A_i=A_1=R$  不含  $R \rightarrow Rr$  规则

②  $i=2, A_i=A_2=Q$

$j=1, A_j=A_1=R$  存在  $Q \rightarrow Rb$

**改写成**  $Q \rightarrow Sab|ab$  (因  $R \rightarrow Sa|a$ ) **无直接左递归**

**G中:**  $Q \rightarrow Sab|ab|b$  —— **原规则中的b**

例：  $G[S]: S \rightarrow Qc|c$   
 $Q \rightarrow Rb|b$   
 $R \rightarrow Sa|a$

③  $i=3, A_i=A_3=S$   
 $j=1, A_j=A_1=R$   
 $j=2, A_j=A_2=Q$

无  $S \rightarrow Rc$  规则

含  $S \rightarrow Qc$  规则

$S \rightarrow Sabc|abc|bc$

$G[S]$  中:  $S \rightarrow Sabc|abc|bc|c$  原规则中的

消除直接递归:  $S \rightarrow (abc|bc|c)\{abc\}$

思想：逐步迭代，改写 (EBNF)

例：  $G[S]: S \rightarrow Qc|c$   
 $Q \rightarrow Rb|b$   
 $R \rightarrow Sa|a$

$G[S]: S \rightarrow (abc|bc|c)\{abc\}$

$Q \rightarrow Sab|ab|b$  多余

$R \rightarrow Sa|a$  多余

$G[S]: S \rightarrow S'\{abc\}$

$S' \rightarrow abc|bc|c$

或  $S \rightarrow abcS'|bcS'|cS'$

$S' \rightarrow abcS' | \varepsilon$

★ 说明：

非终结符号排序不同，所得文法形式不同，但语言等价

# 消除左递归的矩阵表示方法

★ 设  $V_N$  中有  $n$  个非终结符号  $X_1, X_2, \dots, X_n$

将  $X_i \rightarrow \gamma_1 | \gamma_2 | \dots | \gamma_m$  表示成

$X_i = \gamma_1 + \gamma_2 + \dots + \gamma_m$  将  $\gamma_i$  分成两类

$$\begin{cases} \text{首符号} \in V_N \\ \text{首符号} \in V_T \end{cases}$$

$X_i = X_1 a_{1i} + X_2 a_{2i} + \dots + X_n a_{ni} + \beta_i$   $a_{ji}$  可能为  $\emptyset$

$\beta_i$  是以终结符号开头的候选式的“和”

求解  $X_i$

$$(X_1, X_2, \dots, X_n) = (X_1, X_2, \dots, X_n)$$

$X$

$$+ (\beta_1, \beta_2, \dots, \beta_n)$$

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n} \\ \vdots & & \vdots & \\ \alpha_{n1} & \alpha_{n2} & \dots & \alpha_{nn} \end{bmatrix}$$

$A$

$B$

或  $X = XA + B$  其解为  $X = BA^*$

$A^*$ ?

$$A^* = I_n + A + A^2 + \dots = I_n + AA^*$$

★ 解出 $A^*$ 很复杂，但我们的**目的只是消除左递归**，可绕开该问题。

$$I_n = \begin{pmatrix} \varepsilon & \Phi & \Phi & \cdots & \Phi \\ \Phi & \varepsilon & \Phi & \cdots & \Phi \\ \Phi & \Phi & \varepsilon & \cdots & \Phi \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Phi & \Phi & \Phi & \cdots & \varepsilon \end{pmatrix} \quad A^* = \begin{pmatrix} Z_{11} & Z_{12} & Z_{13} & \cdots & Z_{1n} \\ Z_{21} & Z_{22} & Z_{23} & \cdots & Z_{2n} \\ Z_{31} & Z_{32} & Z_{33} & \cdots & Z_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Z_{n1} & Z_{n2} & Z_{n3} & \cdots & Z_{nn} \end{pmatrix} = Z$$

即： $X=BZ$

$$Z=I_n+AZ$$

**将矩阵展开，得新文法，不含左递归！**



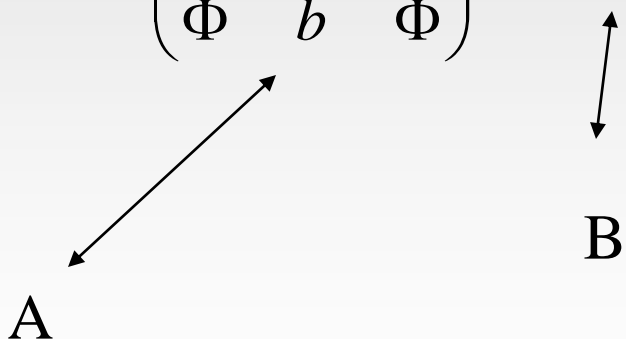
# 举 例

$$\text{上例: } S = Qc + c$$

$$Q = Rb + b$$

$$R = Sa + a$$

矩阵：

$$(S, Q, R) = (S, Q, R) \begin{pmatrix} \Phi & \Phi & a \\ c & \Phi & \Phi \\ \Phi & b & \Phi \end{pmatrix} + (c, b, a)$$


# 矩阵展开

$$(S, Q, R) = BZ = (c, b, a) \begin{pmatrix} Z_{11} & Z_{12} & Z_{13} \\ Z_{21} & Z_{22} & Z_{23} \\ Z_{31} & Z_{32} & Z_{33} \end{pmatrix}$$

$$Z = I_n + AZ = \begin{pmatrix} \varepsilon & \Phi & \Phi \\ \Phi & \varepsilon & \Phi \\ \Phi & \Phi & \varepsilon \end{pmatrix} + \begin{pmatrix} \Phi & \Phi & a \\ c & \Phi & \Phi \\ \Phi & b & \Phi \end{pmatrix} \begin{pmatrix} Z_{11} & Z_{12} & Z_{13} \\ Z_{21} & Z_{22} & Z_{23} \\ Z_{31} & Z_{32} & Z_{33} \end{pmatrix}$$

# 等价文法

$$(S, Q, R) = BZ = (c, b, a) \begin{pmatrix} Z_{11} & Z_{12} & Z_{13} \\ Z_{21} & Z_{22} & Z_{23} \\ Z_{31} & Z_{32} & Z_{33} \end{pmatrix}$$

$$S \rightarrow cZ_{11} | bZ_{21} | aZ_{31}$$

$$Q \rightarrow cZ_{12} | bZ_{22} | aZ_{32}$$

$$R \rightarrow cZ_{13} | bZ_{23} | aZ_{33}$$

$$Z = I_n + AZ = \begin{pmatrix} \varepsilon & \Phi & \Phi \\ \Phi & \varepsilon & \Phi \\ \Phi & \Phi & \varepsilon \end{pmatrix} + \begin{pmatrix} \Phi & \Phi & a \\ c & \Phi & \Phi \\ \Phi & b & \Phi \end{pmatrix} \begin{pmatrix} Z_{11} & Z_{12} & Z_{13} \\ Z_{21} & Z_{22} & Z_{23} \\ Z_{31} & Z_{32} & Z_{33} \end{pmatrix}$$

$$Z_{11} \rightarrow aZ_{31} | \varepsilon$$

$$Z_{21} \rightarrow cZ_{11}$$

$$Z_{31} \rightarrow bZ_{21}$$

$$Z_{12} \rightarrow aZ_{32}$$

$$Z_{22} \rightarrow cZ_{12} | \varepsilon$$

$$Z_{32} \rightarrow bZ_{22}$$

$$Z_{13} \rightarrow aZ_{33}$$

$$Z_{23} \rightarrow cZ_{13}$$

$$Z_{33} \rightarrow bZ_{23} | \varepsilon$$

## 2、回溯的消除

**问题：**不能准确地选定候选式

★ **回溯的原因：**

若当前符号为 $a$ ，需要对 $A$ 展开，而 $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ ，那么要知道哪个 $\alpha_i$ 是获得以 $a$ 开头串的候选式，即通过查看当前（第一个）符号来选择合适的候选式 $\alpha_i$

**根源：**候选式所产生的首字符的交集不为空，即可选定的候选式不唯一

**解决：**对文法的任何非终结符号，当要匹配输入串时，它  
能根据所面临的输入符号准确地指派**唯一**候选式。

## 2、回溯的消除

消除回溯的目标：

非终结符的所有候选式所产生的首符集两两不相交

方法：提取公共左因子

$$A \rightarrow \delta\beta_1 | \delta\beta_2 | \delta\beta_3 | \dots | \delta\beta_n | \gamma_1 | \gamma_2 | \dots | \gamma_m$$

（其中每个 $\gamma_m$ 不以 $\delta$ 开头）那么,可以改写成：

$$A \rightarrow \delta A' | \gamma_1 | \gamma_2 | \dots | \gamma_m$$
$$A' \rightarrow \beta_1 | \beta_2 | \beta_3 | \dots | \beta_n$$

注意：事实上并非消除，而是设计文法避免之。

## 2、回溯的消除

对每一  $A \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n$ , 要求  $\alpha_i$  可推导到的首字母 (终结符号) 不同即可。

如:  $A \rightarrow aB | bC | c$

若有  $\varepsilon$  出现, 则需考虑  $A$  后面的符号

如:  $A \rightarrow aB | bC | \varepsilon$

推导到  $S \xRightarrow{*} \alpha A x \beta$ , 仍然不知选用哪条候选式

(1) 首先定义两个集合：First集和Follow集

$\text{First}(\alpha_i) = \{a \mid \alpha_i \Rightarrow^* a\delta, \text{ 且 } a \in V_T, \alpha_i, \delta \in V^*\}$

当  $\alpha_i \Rightarrow^* \varepsilon$  时, 则  $\varepsilon \in \text{First}(\alpha_i)$

$\text{Follow}(A) = \{a \mid S\# \Rightarrow^* \alpha A a \delta, \text{ 且 } a \in V_T, \alpha, \delta \in V^*\}$

若  $S \Rightarrow^* \alpha A$  则  $\# \in \text{Follow}(A)$ 。

若S 为文法的开始符号, 则  $\# \in \text{Follow}(S)$  ? ✓

例：文法 $G(S)$ ： $S \rightarrow aA|d$      $A \rightarrow bAS|\epsilon$

解：  $\text{First}(aA)=\{a\}$ ,  $\text{First}(d)=\{d\}$

$\text{First}(bAS)=\{b\}$ ,  $\text{First}(\epsilon)=\{\epsilon\}$

$\text{Follow}(S)=?$      $\text{Follow}(S) = \{\#,a,d\}$

$\text{Follow}(A)=?$      $=\{\#\} \cup \text{First}(S)=\{\#,a,d\}$

$\text{First}(\alpha_i)=\{a \mid \alpha_i \Rightarrow^* a\delta, \text{ 且 } a \in V_T, \alpha_i, \delta \in V^*\}$

当 $\alpha_i \Rightarrow^* \epsilon$ 时, 则 $\epsilon \in \text{First}(\alpha_i)$

$\text{Follow}(A)=\{a \mid S\# \Rightarrow^* \alpha A a \delta, \text{ 且 } a \in V_T, \alpha, \delta \in V^*\}$

若 $S \Rightarrow^* \alpha A$  则  $\# \in \text{Follow}(A)$ 。



## (2) 无回溯的条件

对于G中的每一个 $A \in V_N$ ,  $A \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n$

$$\textcircled{1} \text{ First}(\alpha_i) \cap \text{First}(\alpha_j) = \Phi \quad (i \neq j \text{ 时})$$

( $\alpha_i$  和  $\alpha_j$  至多有一个能推出  $\varepsilon$ )

$\textcircled{2}$  若有  $\alpha_i$  能推导出  $\varepsilon$ , 则

$$\text{First}(\alpha_j) \cap \text{Follow}(A) = \Phi \quad (i \neq j \text{ 时})$$

# LL(1)文法

## ★ LL(1)文法

LL(1)文法使用的是确定的自顶向下的分析技术

## ★ LL(1)的含义

第一个L表明自顶向下分析是从左向右扫描输入串

第2个L表明分析过程中将使用最左推导

1表明只需向右看一个符号便可决定如何推导，即选择哪个产生式(规则)进行推导。

★ LL(1)文法的判别需要依次计算FIRST集、FOLLOW集,然后判断是否为LL(1)文法,最后再进行句子分析。

# LL(1)文法

**(1) 提取左公因子，文法不含左递归**

**(2) 对于G中的每一个  $A \in V_N$ ,  $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$**

**①  $\text{First}(\alpha_i) \cap \text{First}(\alpha_j) = \Phi$  (i≠j时)**

**② 若有  $\alpha_i$  能推导出  $\varepsilon$ , 则**

**$\text{First}(\alpha_j) \cap \text{Follow}(A) = \Phi$  (i≠j时)**

例：  $G(S): S \rightarrow aA | d \quad A \rightarrow bAS | \varepsilon$  是否是LL(1)文法？

解： 针对S:  $\text{First}(aA) \cap \text{First}(d) = \Phi$

针对A:  $\text{First}(bAS) \cap \text{First}(\varepsilon) = \Phi$

$\text{Follow}(S) = \{ \#, a, d \} \quad \text{Follow}(A) = \{ a, d, \# \}$

$\text{First}(bAS) \cap \text{Follow}(A) = \Phi$

该文法属于LL(1)文法

(1) 文法不含左递归

(2) 对于G中的每一个  $A \in V_N, A \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n$

①  $\text{First}(\alpha_i) \cap \text{First}(\alpha_j) = \Phi \quad (i \neq j \text{ 时})$

② 若有  $\alpha_i$  能推导出  $\varepsilon$ , 则

$\text{First}(\alpha_j) \cap \text{Follow}(A) = \Phi \quad (i \neq j \text{ 时})$

# LL(1)文法——自上而下无回溯分析法

假设面临的输入符号为 $a$ , 要用非终结符 $A$ 进行匹配,  $(A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n)$

(1) 若 $a$ 属于 $\text{First}(\alpha_i)$ , 则指派 $\alpha_i$ 执行匹配;

(2) 若 $a$ 不属于任何一个 $\text{First}(\alpha_i)$ , 则:

①若 $\epsilon$ 属于 $\text{First}(\alpha_i)$ 且 $a$ 属于 $\text{Follow}(A)$ , 则指派 $\alpha_i$ 执行匹配;

②否则,  $a$ 的出现是一个语法错误。

# 作业**P176**

4-1 (1)

4-31

4-3 (1) (2)

4-33

4-4

4-35 (1)

4-8

4-36

4-9

4-38 (1)

4-13

4-20