



# 编译原理

---

武汉大学计算机学院  
编译原理课程组



# 前述内容回顾

---

- 编译程序
- 编译方式与解释方式的根本区别
- 编译程序的工作过程
- 编译程序的结构
- 编译程序的组织方式
- 编译程序的构造



# 本章内容简介

---

- 文法的形式定义
- 语言的形式定义
- 为语言构造文法
- 和语法分析有关的概念
- 文法的实用限制



## 第2章 文法和语言的形式定义

---

编译程序使得高级语言源程序所描述的功能得以在计算机上实现。编译程序的设计者就是高级语言的**实现者**，源程序的编写者就是高级语言的**使用者**，他们必须**遵循同样的准则**——高级语言程序的构成规则，才能使写出的源程序能够被成功地翻译，**文法**描述的就是高级语言程序的构成规则。

## 第2章 文法和语言的形式定义

- 在20世纪50年代，Noam Chomsky首先对语言的描述问题进行了探讨。他提出了一种用来描述语言的数学系统，并以此定义了四类性质不同的语言，称为**语言（文法）的Chomsky分类**。



2010.8.13 乔姆斯基在北大



Noam Chomsky, 1928—

人们把用一组数学符号和规则来描述语言的方式称为形式描述，把所用的数学符号和规则称为**形式语言**。



## 2.1 字母表与符号串

---

- 语言成份

- 字母表：符号的有穷集，符号构成了语言的句子
- 文法：结构规则的有穷集
- 语义：操作规则的有穷集

- 符号和符号串在形式语言中是很重要的概念，任何一种语言都是由该语言的基本符号组成的符号串集合。

如：英文、Pascal语言



## 2.1 字母表与符号串

### 2.1.1 字母表 (alphabet)

字母表是元素的非空有穷的集合。记为 $\Sigma$ 。

字母表中的元素称为符号。

例如  $\Sigma = \{a, b, \dots, y, z\}$ ,  $\Sigma = \{0, 1\}$   
 $\Sigma = \{\text{BEGIN}, \text{END}, \text{FOR}, \text{WHILE}\}$

- 非空性
- 有穷性



## 2.1 字母表与符号串

### 2.1.2 符号串与符号串集合

符号串是字母表中的符号所组成的任何有穷序列，通常用小写的字母表示。不包含任何符号的符号串为空串，记为  $\varepsilon$ 。

注意：  $\varepsilon \neq$  空格符号

符号串相等       $ab \neq ba$       顺序

符号串的前缀、后缀、子串

设  $\alpha, \beta, \delta$  是符号串，若  $x = \alpha\beta\delta$ ，则称  $\beta$  是  $x$  的子串。

特别地，当  $\alpha = \varepsilon$  时，称  $\beta$  是  $x$  的前缀。

当  $\delta = \varepsilon$  时，称  $\beta$  是  $x$  的后缀。





## 2.1 字母表与符号串

### ① 符号串的长度

符号串的长度 = 符号串中所含符号的个数

$$|\varepsilon| = 0$$

例： $aba$  的长度为3。记为： $|aba|=3$

问：若  $\Sigma = \{\text{BEGIN, END, FOR, WHILE}\}$ ,  $|\text{BEGINEND}| = ?$



## 2.1 字母表与符号串

---

### ② 符号串的连接

设 $x, y$ 是符号串，将 $y$ 直接地拼接接到 $x$ 之后所得的新符号串称为 $x$ 与 $y$ 的连接，记为 $xy$ 。

注意：一般说来， $xy$ 不等于 $yx$

$$\varepsilon X = X\varepsilon = X$$



## 2.1 字母表与符号串

### ③ 符号串的方幂

符号串 $x$ 与其自身的 $n-1$ 次连接, 称为 $x$ 的 $n$ 次方幂, 记为 $x^n$ 。

$$x^0 = \varepsilon$$

$$x^1 = x$$

$$x^2 = xx$$

...

$$x^n = x^{n-1}x = xx^{n-1} = \underbrace{xx \dots x}_{n \text{ 个}}$$

例:  $x=abc$  求  $x^k=?$



## 2.1 字母表与符号串

### ④符号串的逆

符号串 $x$ 的倒置，记为 $x^{-1}$ 。

性质：

1.  $\varepsilon^{-1} = \varepsilon$
2.  $(x^{-1})^{-1} = x$
3. 若 $x = \alpha\beta$ ，则 $x^{-1} = \beta^{-1}\alpha^{-1}$



## 2.1 字母表与符号串

### ⑤ 符号串集合的乘积

设A, B为两个符号串集合, 定义

和  $A+B$  (或  $A \cup B$ )  $= \{w \mid w \in A, \text{ 或 } w \in B\}$

积  $A \cdot B$  (或  $AB$ )  $= \{xy \mid x \in A, y \in B\}$

显然,  $A + \emptyset = \emptyset + A = A$  ;

$A\emptyset = \emptyset A = \emptyset$  ;

$\{\varepsilon\}A = A\{\varepsilon\} = A$



## 2.1 字母表与符号串

---

### ⑥符号串集合的方幂

A为符号串集合，则A的幂运算为：

$$A^0 = \{\epsilon\}$$

$$A^1 = A$$

$$A^2 = AA$$

...

$$A^n = A^{n-1}A \quad (n > 0)$$



## 2.1 字母表与符号串

---

⑦符号串集合的正闭包 $A^+$

$$A^+ = A + A^2 + \cdots = \bigcup_{i=1}^{\infty} A^i$$



## 2.1 字母表与符号串

⑧符号串集合的闭包 $A^*$

$$A^* = A^0 + A + A^2 + \cdots = \bigcup_{i \geq 0}^{\infty} A^i$$

性质：

$$A^* = A^0 \cup A^+ = A^+ + \{\varepsilon\}$$

$$A^+ = AA^* = A^*A$$





# 如何描述语言？

---

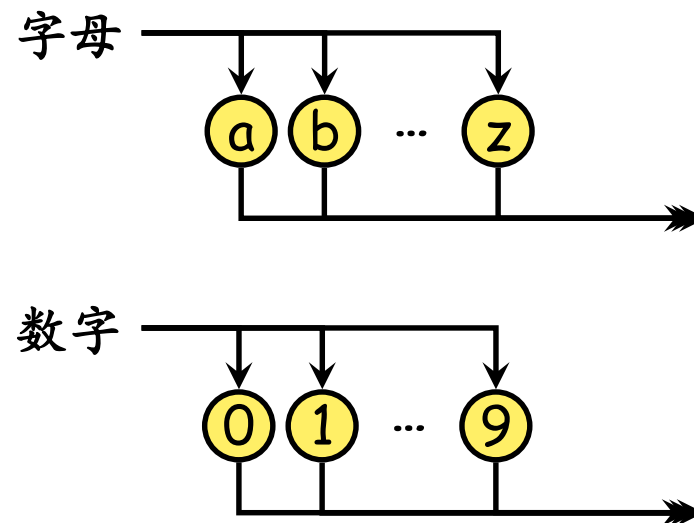
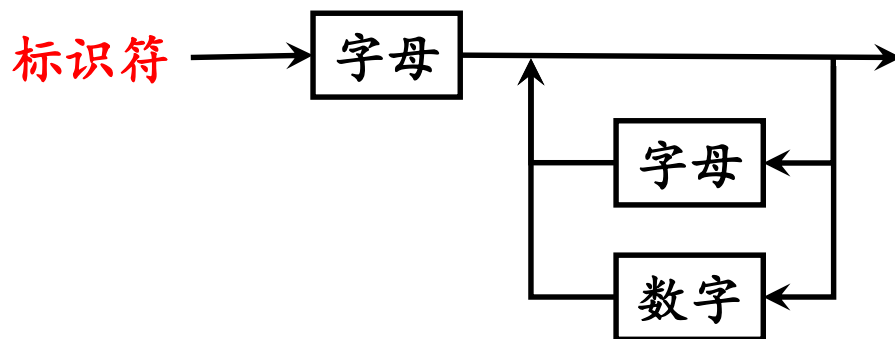
如何描述包含**无穷多个句子**的语言？

举例：〈标识符〉的语法描述

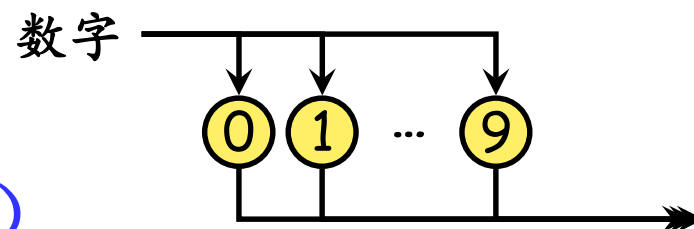
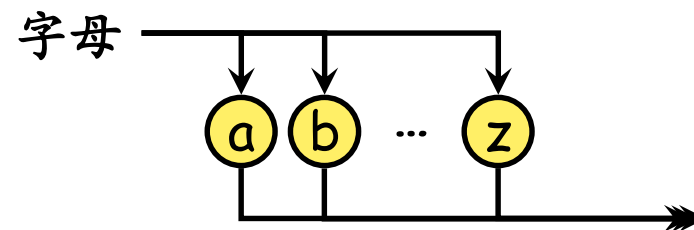
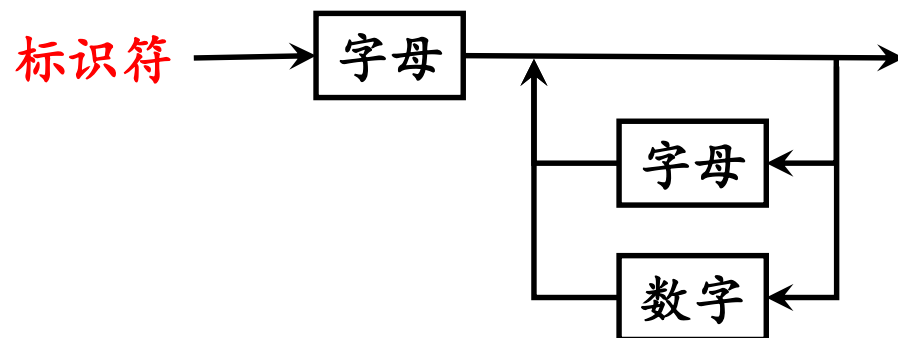
a, b, x1, sum, ...

# 语法描述

- 自然语言
  - 如：<标识符>
  - “是由字母后跟若干个( $\geq 0$ )字母或数字的符号串组成”
- 语法图 (Syntax Graph)



# 语法描述



- BNF范式 (Backus-Naur Form)

标识符 ::= 字母 | 标识符字母 | 标识符数字

- 扩充的BNF (EBNF: Extended BNF)

标识符 ::= 字母 { 字母 | 数字 }



## 2.2 文法及其分类

### 2.2.1 文法

为深入研究语言的内在性质，需要寻找构造语言的方法——**文法**。  
文法是对语言结构的定义和描述。

可以证明：**给定一个文法，就能从结构上唯一地确定其语言。**

例如：程序设计语言中的**无符号整数**是由下列规则构成的：

$\langle \text{无符号整数} \rangle \rightarrow \langle \text{无符号整数} \rangle \langle \text{数字} \rangle$

$\langle \text{无符号整数} \rangle \rightarrow \langle \text{数字} \rangle$

$\langle \text{数字} \rangle \rightarrow 0|1|2|3|4|5|6|7|8|9$

无符号整数365



## 2.2 文法及其分类

### 2.2.1 文法

#### 1. 终结符号:

终结符号是组成语言的基本符号，如保留字、标识符、常数、运算符、界限符等。终结符号是语言的不可再分的基本符号。终结符号形成的集合记为 $V_T$ 。

#### 2. 非终结符号:

非终结符号用来表示语言的语法成分（或语法范畴、语法单位），例如“赋值语句”。非终结符号所形成的集合记为 $V_N$ 。

$$V_T \cap V_N = \emptyset$$



## 2.2 文法及其分类

3. **产生式**: 产生式 (规则) 是一个有序对  $(\alpha, \beta)$ , 通常写作

$$\alpha \rightarrow \beta \quad (\text{或 } \alpha ::= \beta)$$

其中  $\alpha$  称为产生式的**左部**,  $\beta$  称为产生式的**右部**。  $\alpha \in (V_T \cup V_N)^+$ ,  $\beta \in (V_T \cup V_N)^*$ 。

**产生式是用来定义一个语法成分的**。它描述了一个语法成分**的形成规则**。例如标识符的构成规则可描述为:

$$\langle \text{标识符} \rangle \rightarrow \langle \text{字母} \rangle | \langle \text{标识符} \rangle \langle \text{字母} \rangle | \langle \text{标识符} \rangle \langle \text{数字} \rangle$$

假如有若干条规则有相同的左部, 通常写作:  $\alpha \rightarrow \beta_1 | \beta_2 | \dots | \beta_n$



## 2.2 文法及其分类

---

文法 $G$ 是一个四元组,  $G[S] = (V_N, V_T, P, S)$ 。

$V_N$  ——非终结符号集。

$V_T$  ——终结符号集。

$P$  ——表示产生式的有穷非空的集合。

$S$  ——开始符号。至少要在一条产生式中作为左部出现。



# 例 定义标识符的文法

---

$G[\langle \text{标识符} \rangle] = ( \{ \langle \text{标识符} \rangle, \langle \text{字母} \rangle, \langle \text{数字} \rangle \},$   
 $\{ A, B, \dots, Y, Z, 0, 1, \dots, 9 \}, P, \langle \text{标识符} \rangle)$

P 定义为:

$\langle \text{标识符} \rangle \rightarrow \langle \text{字母} \rangle | \langle \text{标识符} \rangle \langle \text{字母} \rangle | \langle \text{标识符} \rangle \langle \text{数字} \rangle$

$\langle \text{字母} \rangle \rightarrow A | B | C | D | E | F | G | \dots | U | V | W | X | Y | Z$

$\langle \text{数字} \rangle \rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$





## 2.2 文法及其分类

---

### 2.2.2 文法分类

乔姆斯基（Chomsky）把文法分成四种类型，即0型、1型、2型和3型。

这四类文法的区别在于：对产生式规则的形式上施加不同的限制。



## 2.2 文法及其分类

---

0型文法     $\alpha \rightarrow \beta$      $\alpha \in V^*V_NV^*$  ,  $\beta \in (V_N \cup V_T)^*$

1型文法     $\alpha \rightarrow \beta$      $1 \leq |\alpha| \leq |\beta|$      $\alpha \in V^*V_NV^*$  ,  $\beta \in (V_N \cup V_T)^+$

2型文法     $A \rightarrow \beta$      $A \in V_N$ ,  $\beta \in (V_N \cup V_T)^+$

3型文法     $A \rightarrow a$  或  $A \rightarrow aB$      $A, B \in V_N$ ,  $a \in V_T$



## 2.2 文法及其分类——举例

例0. 文法 $G_0[S]$ :

$G_0[S] = (\{S, A, B, C, D, E\}, \{a\}, P, S)$ , 其中 $P$ 为

$S \rightarrow ACaB$

$aD \rightarrow Da$

$Ca \rightarrow aaC$

$AD \rightarrow AC$

$CB \rightarrow DB$

$aE \rightarrow Ea$

$CB \rightarrow E$

$AE \rightarrow \varepsilon$

$\alpha \rightarrow \beta$

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



## 2.2 文法及其分类——举例

例1. 文法 $G_1[S]$ :

$G_1[S] = (\{S, B, C, D\}, \{a, b, c\}, P, S)$ , 其中 $P$ 为

$S \rightarrow aSBC \mid aBC$

$aB \rightarrow ab$

$CB \rightarrow CD$

$bB \rightarrow bb$

$CD \rightarrow BD$

$bC \rightarrow bc$

$BD \rightarrow BC$

$cC \rightarrow cc$

$\alpha \rightarrow \beta \quad 1 \leq |\alpha| \leq |\beta|, \quad \alpha \in (V_N \cup V_T)^+, \quad \beta \in (V_N \cup V_T)^+$



## 2.2 文法及其分类——举例

例2. 文法 $G_2[Z]$ :

$$G_2[Z] = (\{Z, S, A, B, C\}, \{a, b, c\}, P, Z),$$

其中P为:

$$Z \rightarrow SC$$

$$S \rightarrow aAc$$

$$A \rightarrow aAc | bBb$$

$$C \rightarrow aCb | \varepsilon$$

$$B \rightarrow bB | \varepsilon$$

扩充的2型文法:  $A \rightarrow \beta$        $A \in V_N, \beta \in (V_N \cup V_T)^*$



## 2.2 文法及其分类——举例

例3. 文法 $G_3[Z]$ :

$G_3[Z] = (\{Z, U, V\}, \{0, 1\}, P, Z)$ , 其中 $P$ 为

$Z \rightarrow U0|V1$

$U \rightarrow Z1|1$

$V \rightarrow Z0|0$

左线性文法:  $A \rightarrow \alpha$  或  $A \rightarrow B\alpha$ ,  $A, B \in V_N$ ,  $\alpha \in V_T^*$

右线性文法:  $A \rightarrow \alpha$  或  $A \rightarrow \alpha B$ ,  $A, B \in V_N$ ,  $\alpha \in V_T^*$



## 2.2 文法及其分类

---

四种类型文法的关系：PSG – CSG – CFG – RG

区别：产生式规则的形式

- 限制逐渐增强
- 能力逐渐减弱
- 定义（生成）的语言集逐渐减小

👉 为什么要对文法进行分类？



## 2.3 语言和语法树

---

语法成分的构成可用**文法**予以描述。给定文法后，可以通过**推导**得到该文法所描述的语言。





## 2.3 语言和语法树——推导

### 1. 直接推导

如果 $\alpha \rightarrow \beta$ 是文法 $G$ 的一条产生式，而 $\gamma, \delta$ 是 $(V_T \cup V_N)^*$ 中任意一个符号串，则将 $\alpha \rightarrow \beta$ 作用于符号串 $\gamma\alpha\delta$ 上得到符号串 $\gamma\beta\delta$ ，称符号串 $\gamma\beta\delta$ 是符号串 $\gamma\alpha\delta$ 的**直接推导**，记为

$$\gamma\alpha\delta \Rightarrow \gamma\beta\delta$$

直接推导的逆过程称为**直接归约**，即由符号串 $\gamma\beta\delta$ 可直接归约到 $\gamma\alpha\delta$ 。



# 直接推导举例

---

文法G[E]:

$E \rightarrow E+T | T$

$T \rightarrow T * F | F$

$F \rightarrow (E) | i$

$* T F \Rightarrow * T * F F$



## 2.3 语言和语法树——推导

### 2. 推导

设  $\alpha_0, \alpha_1, \dots, \alpha_n$  ( $n > 0$ ) 均为  $(V_T \cup V_N)^*$  中的符号串, 且有

$$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_{n-1} \Rightarrow \alpha_n$$

则称以上序列是长度为  $n$  的推导, 即  $\alpha_0$  可经过  $n$  步推导得到  $\alpha_n$ 。

$$\alpha_0 \xRightarrow{+} \alpha_n$$

推导的逆过程称为归约, 即  $\alpha_n$  可归约到  $\alpha_0$ 。



## 2.3 语言和语法树——推导

### 2. 推导

$$\alpha \xRightarrow{*} \beta$$

表示  $\alpha \xRightarrow{+} \beta$  或  $\alpha = \beta$

即从  $\alpha$  出发, 经过0步或多步推导, 可推导出  $\beta$ 。

例:  $\langle \text{无符号整数} \rangle \rightarrow \langle \text{无符号整数} \rangle \langle \text{数字} \rangle \mid \langle \text{数字} \rangle$   
 $\langle \text{数字} \rangle \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

无符号整数23



## 2.3 语言和语法树——语言

### 1. 句型

设有文法 $G[S]$ ，如果  $S \xRightarrow{*} u$  ，则称符号串 $u$ 为文法 $G[S]$ 的句型。

### 2. 句子

设有文法 $G[S]$ ，如果  $S \xRightarrow{*} u$  ，且  $u \in V_T^*$  ，

则称符号串 $u$ 为文法 $G[S]$ 的句子。



## 2.3 语言和语法树——语言

### 3. 语言

设有文法 $G[S]$ ，则文法 $G[S]$ 描述的语言为：

$$L(G[S]) = \{ u \mid S \xRightarrow{*} u, \text{ 且 } u \in V_T^* \}$$

举例：  $G_1[E]: E \rightarrow E+T \mid T \qquad T \rightarrow T * F \mid F \qquad F \rightarrow (E) \mid i$

$G_2[N]: N \rightarrow ND \mid D \qquad D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$



# 语言 $\xrightarrow{\text{凑规则}}$ 文法

---

1. 找出语言的若干典型句子
2. 分析句子特点
3. 根据句子特点凑规则
4. 得到文法
5. 检查文法, 应满足:
  - (1) 语言的所有句子都能由S推导得到;
  - (2) S推导得到的所有终结符号串都是语言的句子.



# 语言 $\xrightarrow{\text{凑规则}}$ 文法

【例1】  $L(G_1) = \{ a^n b^n \mid n > 0 \}$ , 求  $G_1$ .

【例2】  $L(G_2) = \{ a^n b^n \mid n \geq 0 \}$ , 求  $G_2$ .

【例3】  $L(G_3) = \{ a^n b^m \mid n \geq 0, m > 0 \}$ , 求  $G_3$ .

【例4】  $L(G_4) = \{ a^n b^{2m-1} \mid n \geq 1, m \geq 1 \}$ , 求  $G_4$ .

【例5】  $L(G_5) = \{ a^n b^{2n-1} \mid n \geq 1 \}$ , 求  $G_5$ .

【例6】 分别用右线性文法、左线性文法、正规文法来描述语言：  
“所有以0开头，后接0个或多个10组成的符号串的集合”。即：

$$L = \{ 0(10)^n \mid n \geq 0 \}$$





# 语言 $\xrightarrow{\text{凑规则}}$ 文法

【例6】  $L = \{ 0(10)^n \mid n \geq 0 \}$

右线性文法  $P_1$ :  $S \rightarrow 0A$   
 $A \rightarrow 10A \mid \varepsilon$

左线性文法  $P_2$ :  $S \rightarrow S10 \mid 0$

正规文法  $P_3$ :  $S \rightarrow 0A \mid 0$   
 $A \rightarrow 1S$

$L = \{ (01)^n 0 \mid n \geq 0 \}$

左线性文法  $P_4$ :  $S \rightarrow A0$   
 $A \rightarrow A01 \mid \varepsilon$

右线性文法  $P_5$ :  $S \rightarrow 01S \mid 0$

左线性文法  $P_6$ :  $S \rightarrow A0 \mid 0$   
 $A \rightarrow S1$

$$0(10)^n = 0\underbrace{101010\dots1010}_{n\text{个}10} = 0\underbrace{0101010\dots0101}_{n\text{个}01} = (01)^n 0$$



# 文法和语言的关系

---

- 给定一个语言，能确定相应的文法，但**不是唯一的**。
- 给定一个文法，就能从结构上**唯一**地确定其语言。
- 给定一个文法 $G$ 总能产生一个语言 $L(G)$ ，它可能包含一些句子，也可能不包含任何句子（空语言）。

**空语言**: 不包含任何句子。



# 文法和语言的关系——文法等价

## 【定义】文法等价

设 $G$ 与 $G'$ 是两个文法，如果 $L(G)=L(G')$ ，  
则称文法 $G$ 与 $G'$ 是等价的。

➤ 文法等价是语法等价，而非语义等价。

例：  $G_1[S]: S \rightarrow A \mid S-A \quad A \rightarrow a \mid b \mid c$

$G_2[S]: S \rightarrow A \mid A-S \quad A \rightarrow a \mid b \mid c$



# 与语法分析有关的几个概念

## 【定义】最左推导和最右推导

如果在某个推导过程中的任何一步直接推导 $\alpha \Rightarrow \beta$ 中，都是对符号串 $\alpha$ 的最左(右)非终结符号进行替换，则称其为最左(右)推导。最右推导又叫做规范推导。由规范推导得到的句型称为规范句型。

最左直接推导： $xUy \xRightarrow{lm} xuy, \quad x \in V_T^*, U \in V_N$

最右直接推导： $xUy \xRightarrow{rm} xuy, \quad y \in V_T^*, U \in V_N$



# 与语法分析有关的几个概念

## 【定义】最右归约和最左归约

最左推导的逆过程称为最右归约。

最右推导（规范推导）的逆过程称为最左归约。

例：  $G[S] = (\{S, A\}, \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, P, S)$

P:  $S \rightarrow SA \mid A$

$A \rightarrow 0|1|2|3|4|5|6|7|8|9$

- 每个句子都有一个规范推导
- 每个句型不一定存在规范推导



# 递归

---

如果文法的产生式呈  $U \rightarrow xUy$  形式，则称其为规则递归，也称直接递归。

如果文法中有推导  $U \xRightarrow{*} xUy$ ，则称其为文法递归，也称间接递归。



# 左递归

---

如果文法的产生式呈  $U \rightarrow Uy$  形式，则称其为规则左递归，也称直接左递归。

如果文法中有推导  $U \xRightarrow{*} Uy$ ，则称其为文法左递归，也称间接左递归。



# 右递归

---

如果文法的产生式呈  $U \rightarrow xU$  形式，则称其为规则右递归，也称直接右递归。

如果文法中有推导  $U \xRightarrow{*} xU$ ，则称其为文法右递归，也称间接右递归。





# 直接递归（规则递归）

---

$G_1[S]: S \rightarrow Sa | Ab | b | c$

$A \rightarrow Bc | a$

$B \rightarrow Sb | b$

直接左递归

$G_2[S]: S \rightarrow a | \varepsilon | aTb$

$T \rightarrow S, T | S$

直接右递归



# 间接递归（文法递归）

---

文法 $G_3[S]$ :  $S \rightarrow Aa|c$

$A \rightarrow Bc|a$

$B \rightarrow Sb|b$

间接左递归



# 递归

---

## 文法递归的作用：

用较少的产生式产生无穷多个句子，实现“用有穷表示无穷”。

➤ 若一个语言的句子有无穷多个，则其对应的文法必定是递归的。

【例】  $G[\langle \text{无符号整数} \rangle]$ ：

$\langle \text{无符号整数} \rangle \rightarrow \langle \text{数字} \rangle \mid \langle \text{无符号整数} \rangle \langle \text{数字} \rangle$

$\langle \text{数字} \rangle \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$



## 2.3 语言和语法树——语法分析树

设文法  $G = (V_N, V_T, P, S)$ ,

满足以下条件的树称为一棵**语法分析树** (parse tree)。

- (1) 树中的每个结点都有**标记**, 该标记是  $V_N \cup V_T$  中某一个符号;
- (2) **树根的标记**是识别符号  $S$ ;
- (3) 若一个结点至少有一个后继, 则该结点上的标记必为**非终结符号**;
- (4) 若一个标记为  $U$  的结点, 它有标记依次为  $x_1, x_2, \dots, x_n$  的**直接后继结点**, 则  $U \rightarrow x_1 x_2 \dots x_n$  必定是文法  $G$  的一条产生式。





# 语法分析树与推导

## ➤ 由推导生成语法分析树

依直接推导（产生式）增添分枝，直至推导结束。

若推导过程不同，语法树的生长过程也不同，但最终生成的语法树完全相同。

## ➤ 由语法分析树构造推导

由分枝建立直接推导，剪枝直至无分枝可剪。

按照这个构造方法，每个语法树必定至少存在一个推导，当剪枝次序不同时，将得到不同的推导。



# 语法分析树举例

【例】已知表达式文法 $G[E]$ :

$$E \rightarrow -EE$$
$$E \rightarrow -E$$
$$E \rightarrow a$$
$$E \rightarrow b$$
$$E \rightarrow c$$

试问 **-- a - bc** 是不是 $L(G)$ 的句子? 若是, 请给出该句子所有可能的语法树; 若不是, 请说明理由。

# 自然语言的二义性

《论语·泰伯篇》 民可使由之不可使知之

- ▶ 民可使由之，不可使知之。
- ▶ 民可，使由之；不可，使知之。
- ▶ 民可使，由之；不可使，知之。





# 二义性问题

---

## ➤ 句子的二义性

- ⇒ 一个句子有两棵（或以上）不同的语法树
- ⇒ 存在两个（或以上）不同的最左推导
- ⇒ 存在两个（或以上）不同的最右推导

## ➤ 文法的二义性

- ⇒ 文法G中的某一句子具有二义性

## ➤ 语言的二义性

- ⇒ 不存在任何无二义性的文法来描述某语言L



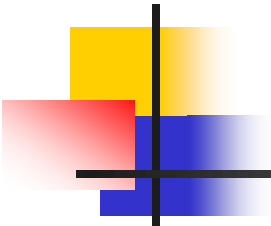


# 二义性举例

---

【例】语言  $\{a^i b^i c^j \mid i, j \geq 1\} \cup \{a^i b^j c^j \mid i, j \geq 1\}$   
是先天二义性的。

存在必为二义性的句子  $a^k b^k c^k$  ( $k \geq 1$ )。



# 二义性问题的不可判定性

二义性问题是不可判定的。即：不存在一个算法，它能在有限的步骤内，确切地判定一个文法是否为二义性的。

【例】已知文法G[S]：

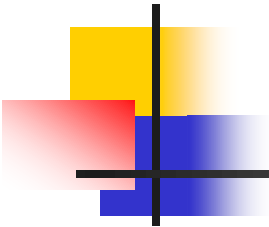
$S \rightarrow SaS \mid SbS \mid cSd \mid eS \mid f$

请证明该文法是二义性文法。

方法1：找一句子（句型），存在两棵（以上）不同的语法树。

方法2：找一句子（句型），存在两种（以上）不同的最左推导过程。

方法3：找一句子（句型），存在两种（以上）不同的最右推导过程。



## 2.4 文法的实用限制

---

在实际使用文法时，经常会对文法有所限制，使之满足某种具体的编译方法的要求。

- 消除文法的二义性
- 文法的压缩（化简）
- 消除单规则
- 消除空产生式
- 消除左递归



## 2.4 文法的实用限制——二义性

如果文法G中的某一个句子存在两棵(包括两棵)以上不同的语法树(即有两个不同的最左或最右推导), 则称该文法是二义性的。

例: Pascal语言中关于if语句的文法:

$\langle \text{if语句} \rangle \rightarrow \text{if} \langle \text{条件} \rangle \text{then} \langle \text{语句} \rangle$

$\quad \quad \quad | \text{if} \langle \text{条件} \rangle \text{then} \langle \text{语句} \rangle \text{else} \langle \text{语句} \rangle$

$S \rightarrow \langle \text{if语句} \rangle \mid i := E$

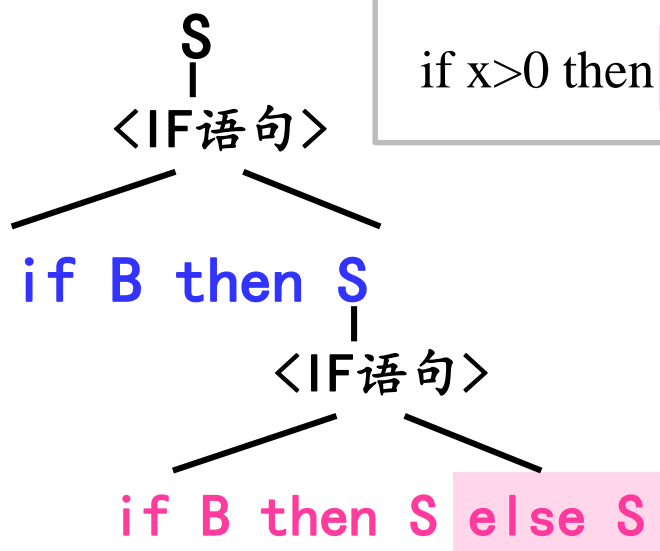
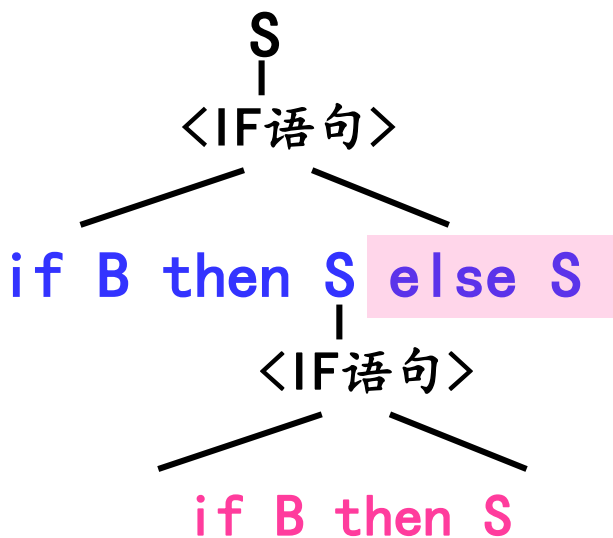
$\langle \text{if语句} \rangle \rightarrow \text{if } B \text{ then } S \mid \text{if } B \text{ then } S \text{ else } S$

## 2.4 文法的实用限制——二义性

例:  $S \rightarrow \langle \text{if语句} \rangle \mid i:=E$

$\langle \text{if语句} \rangle \rightarrow \text{if } B \text{ then } S \mid \text{if } B \text{ then } S \text{ else } S$

句型  $\text{if } B \text{ then if } B \text{ then } S \text{ else } S$  就有两棵不同的语法树。



if x>0 then if y>0 then x:=0 else y:=0

if x>0 then if y>0 then x:=0 else y:=0



最近嵌套规则

## 2.4 文法的实用限制——二义性

排除文法二义性通常有两种方法：

悬挂else问题

(1) 设定消除二义性规则：在语义上加些限制。最近嵌套规则

(2) 重写文法：重新构造一个等价的无二义性文法。

【例】if语句：等价的无二义性文法

$$stmt \rightarrow matched\_stmt \mid unmatched\_stmt$$
$$matched\_stmt \rightarrow \text{if } expr \text{ then } matched\_stmt \text{ else } matched\_stmt \mid i:=E$$
$$unmatched\_stmt \rightarrow \text{if } expr \text{ then } stmt \mid$$
$$\text{if } expr \text{ then } matched\_stmt \text{ else } unmatched\_stmt$$
$$expr \rightarrow 0 \mid 1$$



## 2.4 文法的实用限制——二义性

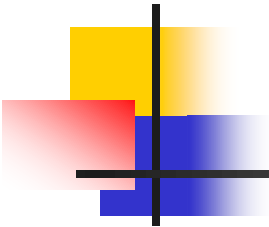
排除文法二义性通常有两种方法：运算符的优先级和结合性

- (1) 设定消除二义性规则：在语义上加些限制。
- (2) 重写文法：重新构造一个等价的无二义性文法。

【例】 $G[E]: E \rightarrow E+E \mid E-E \mid E * E \mid E/E \mid (E) \mid i$

等价的无二义性文法  $G'[E]$ :

$$\begin{aligned} E &\rightarrow E+T \mid E-T \mid T \\ T &\rightarrow T * F \mid T/F \mid F \\ F &\rightarrow (E) \mid i \end{aligned}$$



## 2.4 文法的实用限制

---

在实际使用文法时，经常会对文法有所限制，使之满足某种具体的编译方法的要求。

- 消除文法的二义性
- 文法的压缩（化简）
- 消除单规则
- 消除空产生式
- 消除左递归





## 2.4 文法的实用限制——文法的压缩

### 1. 文法不能含有**多余产生式**:

无法推导出终结符号串的产生式。

从开始符号出发的所有推导都不会用到的产生式。

【例】文法G[S]:

$$\begin{aligned} S &\rightarrow Ab \\ A &\rightarrow f \mid Db \\ B &\rightarrow f \end{aligned}$$



## 2.4 文法的实用限制——文法的压缩

2. 文法不能含有有害产生式:  $U \rightarrow U$

【例】  $G[S] = (\{S, A\}, \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, P, S)$

P:  $S \rightarrow SA \mid A \mid S$

$A \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$



## 2.4 文法的实用限制——文法的压缩

【定义】若文法 $G[S]$ 的所有产生式都满足下列实用限制条件：

1. 没有多余产生式
2. 没有有害产生式

则称文法 $G[S]$ 是压缩或化简的。



# 文法压缩举例

【例】化简（压缩）下面的文法

$G_1[S]: S \rightarrow Aa|cc$

$A \rightarrow Ae|Ca|a|A$

$C \rightarrow Cb$

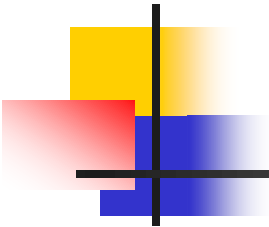
$D \rightarrow b|Db$



化简后的文法

$G_2[S]: S \rightarrow Aa|cc$

$A \rightarrow Ae|a$



## 2.4 文法的实用限制

---

在实际使用文法时，经常会对文法有所限制，使之满足某种具体的编译方法的要求。

- 消除文法的二义性
- 文法的压缩（化简）
- 消除单规则
- 消除空产生式
- 消除左递归

## 2.4 文法的实用限制——删除单规则

删除单规则（单产生式）： $A \rightarrow B$

【例】 $G_1[E]: E \rightarrow E+T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid i$

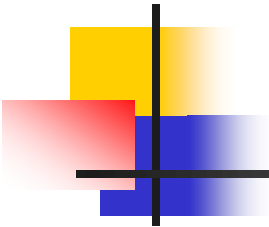


删除单产生式后的等价文法

$G_2[E]: E \rightarrow E+T \mid T * F \mid (E) \mid i$

$T \rightarrow T * F \mid (E) \mid i$

$F \rightarrow (E) \mid i$



## 2.4 文法的实用限制

---

在实际使用文法时，经常会对文法有所限制，使之满足某种具体的编译方法的要求。

- 消除文法的二义性
- 文法的压缩（化简）
- 消除单规则
- 消除空产生式
- 消除左递归

## 2.4 文法的实用限制——删除空产生式

删除空产生式 ( $\epsilon$  规则) :  $U \rightarrow \epsilon$

【例】  $G_1[S]$ :  $A \rightarrow aBbD$

$B \rightarrow DD$

$D \rightarrow b \mid \epsilon$



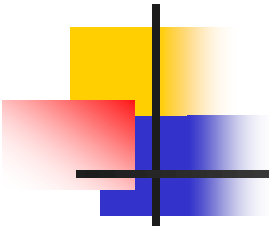
删除空产生式后的等价文法

$G_2[S]$ :  $A \rightarrow aBbD \mid aBb \mid abD \mid ab$

$B \rightarrow DD \mid D$

$D \rightarrow b$





## 2.4 文法的实用限制

---

在实际使用文法时，经常会对文法有所限制，使之满足某种具体的编译方法的要求。

- 消除文法的二义性
- 文法的压缩（化简）
- 消除单规则
- 消除空产生式
- 消除左递归

## 2.4 文法的实用限制——消除直接左递归

消除直接左递归  $U \rightarrow Uy$

- 采用EBNF表示

$[x]$ —— $x$ 可以出现零次或一次

$\{x\}$ —— $x$ 可以出现零次到多次

$x(y|z)$ ——等价于  $xy$  或  $xz$

(1) 提取公因子

$$A \rightarrow ux|uy|\dots|uz \quad \Rightarrow \quad A \rightarrow u(x|y|\dots|z)$$

(2) 消除直接左递归

$$A \rightarrow x|y|\dots|z|Au \quad \Rightarrow \quad A \rightarrow (x|y|\dots|z)\{u\}$$

## 2.4 文法的实用限制——消除直接左递归

消除直接左递归  $U \rightarrow Uy$

- 直接改写法

引进新的非终结符号，将左递归改写为右递归。

设有产生式  $U \rightarrow Ux_1 | Ux_2 | \dots | Ux_m | y_1 | y_2 | \dots | y_n$

其中， $y_1, \dots, y_n$  均不以符号  $U$  为首， $x_1, \dots, x_m$  均不为  $\varepsilon$ ，

增加新非终结符号  $U'$ ，将上述产生式变换为

$$U \rightarrow y_1 U' | y_2 U' | \dots | y_n U'$$

$$U' \rightarrow x_1 U' | x_2 U' | \dots | x_m U' | \varepsilon$$

## 2.4 文法的实用限制——消除直接左递归

消除直接左递归

【例】  $G_1[S]: S \rightarrow Sa|b|c$

 EBNF:

$G_2[S]: S \rightarrow (b|c)\{a\}$

直接改写法:

$G_3[S]: S \rightarrow bS'|cS'$

$S' \rightarrow aS' | \epsilon$

## 2.4 文法的实用限制——消除直接左递归

消除直接左递归

【例】 $G_1[E]: E \rightarrow E+T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid i$



EBNF:

$G_2[E]: E \rightarrow T\{+T\}$

$T \rightarrow F\{ * F \}$

$F \rightarrow (E) \mid i$

直接改写法:

$G_3[E]:$

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \varepsilon$

$F \rightarrow (E) \mid i$



## 2.5 分析方法简介

文法 (生成)		自动机 (识别)	语言
0型文法	PSG	TM	PL的语法规则
1型文法	CSG	LBA	
2型文法	CFG	PDA	
3型文法	RG	FSA	PL的词法规则



## 2.5 分析方法简介

---

1. 自上而下分析方法

2. 自下而上分析方法





## 2.5 分析方法简介

### 1. 自上而下分析方法的基本思想

- 构造语法树角度:

从S出发,利用文法产生式向下构造语法树,使得叶子结点从左至右的排列恰好与给定的待检查串匹配。

- 推导角度:

从S出发,利用文法产生式,为串寻找一个最左推导序列。

**【例】**  $G_{16}[S]: S \rightarrow aAbc|aB, A \rightarrow ba, B \rightarrow beB|d$   
待检查串 abed



## 2.5 分析方法简介

问题：当  $A \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n$  时，A 具有多个候选式，如何恰好选择其中一个使得分析工作能够继续？

- 带回溯的自上而下分析方法（不确定的自上而下分析方法）

- 确定的自上而下分析方法

限制文法：若某非终结符有多个候选式，当候选式首符号是终结符时，应保证它们互不相同。

【例】 $G_{17}[S]: S \rightarrow aBc | bCd, B \rightarrow eB | f, C \rightarrow dC | c$

待检查串 aefc



## 2.5 分析方法简介

### 2. 自下而上分析方法的基本思想

- 构造语法树角度:

从待检查串出发, 利用产生式向上构造语法树, 直至到达S。

- 推导角度:

从待检查串出发, 寻找一个最左归约, 直至到达S。

【例】  $G[S]: S \rightarrow SA|A, A \rightarrow 0|1|2|3|4|5|6|7|8|9$

待检查串 365

问题1: 如何选择合适的产生式?

问题2: 将它应用到构造好的语法树的哪个部分?

(去替换符号串的哪个子串?)



## 第2章内容小结

---

- 文法的形式定义
- 语言的形式定义
- 为语言构造文法
- 与语法分析有关的概念
- 文法的实用限制



# 作业

---

分编译学习小组：  
6人为一组，自由组合。





# 下章内容简介——第3章

---

- DFA、NDFA
- NDFA到DFA的转换
- 正规文法与FA
- 正规表达式与FA