



## 第2章

# 文法和语言

本章目的：为语言的语法描述寻求工具。

通过该工具，可以：

- ❑ 对源程序给出精确无二义的语法描述。（严谨、简洁、易读）
- ❑ 根据语言文法的特点来指导语法分析的过程。
- ❑ 从描述语言的文法可以自动构造出可用的分析程序。
- ❑ 制导语义翻译。

习题P34页2, 4, 11, 12(1)(2)(6), 15, 18

- 一个程序设计语言是一个记号系统，它的完整定义应包括两个方面：
  - 语法  
指一组规则，用它可以形成和产生一个合适的程序。
  - 语义
    - 静态语义  
是一系列限定规则，并确定哪些合乎语法的程序是合适的；
    - 动态语义  
也称作运行语义或执行语义，表明程序要做些什么，要计算什么。
- 文法是阐明语法的一个工具

## 2.1 文法的直观概念

<句子> ::= <主语> <谓语>

<主语> ::= <代词> | <名词>

<代词> ::= 我 | 你 | 他

<名词> ::= 王明 | 大学生 | 工人 | 英语

<谓语> ::= <动词> <直接宾语>

<动词> ::= 是 | 学习

<直接宾语> ::= <代词> | <名词>

# PL/0 语言的符号说明

$\langle \text{变量说明部分} \rangle ::= \text{VAR } \langle \text{标识符} \rangle \{ , \langle \text{标识符} \rangle \};$

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

$\langle \text{字母} \rangle ::= a | b | \dots | X | Y | Z$

$\langle \text{数字} \rangle ::= 0 | 1 | 2 | \dots | 8 | 9$

# 形式语言

- 如果不考虑语义和语用，即只从语法这一侧面来看语言，这种意义下的语言称作形式语言。形式语言抽象地定义为一个数学系统。
- “形式”是指这样的事实：语言的所有规则只以什么符号串能出现的方式来陈述。
- 形式语言理论是对符号串集合的表示法、结构及其特性的研究。是程序设计语言语法分析研究的基础。

## 2.2 符号和符号串

- 字母表 $\Sigma$ 
  - 字母表是元素的非空有穷集合
- 符号
  - 字母表中的元素称为符号
- 符号串
  - 由符号组成的任何有穷序列
  - 符号串 $x$ 的长度:  $x$ 所包含的符号个数, 记作 $|x|$
  - 空符号串  $\varepsilon$

例如:  $\Sigma = \{a, b\}$

- 符号串的头、尾、固有头、固有尾
- 符号串的连接
  - 设 $x$ 和 $y$ 是符号串，它们的连接 $xy$ 是把 $y$ 的符号写在 $x$ 的符号之后得到的符号串。
- 符号串的方幂
  - 设 $x$ 是符号串，把 $x$ 自身连接 $n$ 次得到符号串 $z$ ，即 $z=xx...xx$ ，称为符号串 $x$ 的方幂。  
 $\alpha^0=\varepsilon$ ,  $\alpha^n=\alpha\alpha^{n-1}=\alpha^{n-1}\alpha$  ( $n>0$ )

例如:  $s=abc$



## ■ 符号串集合及其运算

- 若集合**A**中的一切元素都是字母表上的符号串，则称**A**为该字母表上的符号串集合。
- 合并：字符串集合**A**和**B**的合并 **$A \cup B = \{\alpha | \alpha \in A \text{ 或 } \alpha \in B\}$** 。
- 乘积：字符串集合**A**和**B**的乘积 **$AB = \{\alpha\beta | \alpha \in A \text{ 且 } \beta \in B\}$** 。  
显然 **$\{\varepsilon\}A = A\{\varepsilon\} = A$** 。
- 幂： **$A^n = A^{n-1}A = AA^{n-1}$  ( $n > 0$ )**，并规定 **$A^0 = \{\varepsilon\}$** 。
- 正闭包： **$A^+ = A^1 \cup A^2 \cup \dots \cup A^n \cup \dots$** 。
- 闭包： **$A^* = A^0 \cup A^+$** 。  
显然 **$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots \cup \Sigma^n \cup \dots$** 。  
 **$\Sigma^*$ 表示 **$\Sigma$** 上的所有有穷长的串的集合**

■ 例  $\Sigma = \{0, 1, 2, 3, \dots, 9\}$

则：

$1^3$

$\{1, 0\}^3$

$\{1, 0\}^+$

$\{1, 0\}^*$

$\{10\}^*$

$L = \{a, b, \dots, y, z\},$

$M = \{1, 2, \dots, 8, 9\}$

则求解:

$(L \cup M)$

$(L^1 \cup M^1) \quad *$

$L \quad (L \cup M) \quad *$

## 2.3 文法和语言的形式定义

— 概念 设  $\Sigma$  为字母表，则任何集合  $L \subseteq \Sigma^*$  是字母表  $\Sigma$  上的一个语言

■ 如何来描述一种语言？

□ 如果语言是有穷的（只含有有穷多个句子），可以将句子逐一列出来表示

□ 如果语言是无穷的，找出语言的有穷表示。两个途经：

生成方式（文法）：语言中的每个句子可以用严格定义的规则来构造。

识别方式（自动机）：用一个过程，当输入的一任意串属于语言时，该过程经有限次计算后就会停止并回答“是”，若不属于，要么能停止并回答“不是”，（要么永远继续下去。）

## ■ 定义2.1-文法

文法**G**定义为四元组( $V_N$ ,  $V_T$ ,  $P$ ,  $S$ )。其中

$V_N$ : 非终结符的非空有穷集;

$V_T$ : 终结符的非空有穷集;

$P$ : 产生式 (也称规则) 的非空有穷集;

$S$ : 开始符号, 它是一个非终结符, 至少要在一条规则中作为左部出现。

通常用 $V$ 表示 $V_N \cup V_T$ ,  $V$ 称为文法**G**的文法符号集。

$$V_N \cap V_T = ?$$

# 文法示例1

- 例2.1 文法  $G = (V_N, V_T, P, S)$

$$V_N = \{ A \}, V_T = \{ 0, 1 \}$$

$$P = \{ A \rightarrow 0A1, A \rightarrow 01 \}$$

$A$  为文法  $G$  的开始符号

- 习惯上只将产生式写出。并有如下约定：

- 第一条产生式的左部是开始符号

- 用尖括号括起的是非终结符，否则为终结符。或者大写字母表示非终结符，小写字母表示终结符

- $G$  可写成  $G[S]$ ， $S$  是开始符号

例：  $G$ :  $A \rightarrow 0A1$   
 $A \rightarrow 01$

或  $G[A]$ :  $A \rightarrow 0A1$   
 $A \rightarrow 01$

或  $G[A]$ :  $A \rightarrow 0A1|01$

# 文法示例2

- 例2.2 文法  $G = (V_N, V_T, P, S)$

$V_N = \{\text{标识符}, \text{字母}, \text{数字}\}$

$V_T = \{a, b, c, \dots, x, y, z, 0, 1, \dots, 9\}$

$P = \{$   
     $\langle \text{标识符} \rangle \rightarrow \langle \text{字母} \rangle$   
     $\langle \text{标识符} \rangle \rightarrow \langle \text{标识符} \rangle \langle \text{字母} \rangle$   
     $\langle \text{标识符} \rangle \rightarrow \langle \text{标识符} \rangle \langle \text{数字} \rangle$   
     $\langle \text{字母} \rangle \rightarrow a, \dots, \langle \text{字母} \rangle \rightarrow z$   
     $\langle \text{数字} \rangle \rightarrow 0, \dots, \langle \text{数字} \rangle \rightarrow 9 \}$

$S = \langle \text{标识符} \rangle$

# 文法示例3

- $\Sigma=\{a\}, A=\{a^n|n\geq 1\}$
- $\Sigma=\{a,b\}, A=\{a^n b^m|n,m\geq 1\}$
- $\Sigma=\{a,b\}, A=\{a^n b^n|n\geq 1\}$



# 文法和语言的掌握要求

- 已知语言描述，写出文法

例：若语言由0、1符号串组成，串中0和1的个数相同，构造其文法。

- 已知文法，写出语言描述

例：  $G[E]: E \rightarrow E+T | T$   
 $T \rightarrow T * F | F$   
 $F \rightarrow (E) | a$

## ■ 例2.3

**G[S]:**

$S \rightarrow aSBE \mid aBE$

$EB \rightarrow BE$

$aB \rightarrow ab$

$bB \rightarrow bb$

$bE \rightarrow be$

$eE \rightarrow ee$

## -课堂思考问题

- 给出下列语言  $L$  的一个文法:

$$L = \{ a^n b^m \mid m \geq n \geq 1 \}$$

*a a ... a b... b ... b b*

*a a ... a b... b b... b*

*a a ... a b... b... b*

## ■ 定义2.2-直接推导、直接归约

设  $\alpha \rightarrow \beta$  是文法  $G=(V_N, V_T, P, S)$  的规则,  $\gamma$  和  $\delta$  是  $V^*$  中的任意符号串。若有符号串  $v, w$  满足:  $v=\gamma\alpha\delta$ ,  $w=\gamma\beta\delta$ , 则说  $v$  (应用规则  $\alpha \rightarrow \beta$ ) 直接产生  $w$ , 或说  $w$  是  $v$  的直接推导, 或说  $w$  直接归约到  $v$ , 记作  $v \Rightarrow w$ 。

■ 定义2.3 -推导

若存在  $v \Rightarrow w_0 \Rightarrow w_1 \Rightarrow \dots \Rightarrow w_n = w$  ( $n > 0$ ), 则说  $v$  推导出  $w$ , 或说  $w$  归约到  $v$ , 记为

$$v \stackrel{+}{\Rightarrow} w.$$

■ 定义2.4-星推导

若有  $v \stackrel{+}{\Rightarrow} w$ , 或  $v = w$ , 则记为  $v \stackrel{*}{\Rightarrow} w$ .

## ■ 最左（最右）推导

- 如果在推导的任何一步  $\alpha \Rightarrow \beta$ ，其中  $\alpha \in V^*$ ，都是对  $\alpha$  中的最左（最右）非终结符进行替换，则称这种推导为最左（最右）推导

## ■ 规范推导

- 在形式语言中，最右推导常被称为规范推导。

# id+id\*id的不同推导 $E \rightarrow E+E | E * E | (E) | id$

$E \Rightarrow E * E$   
 $\Rightarrow E + E * E$   
 $\Rightarrow E + id * E$   
 $\Rightarrow id + id * E$   
 $\Rightarrow id + id * id$

不做限制

句型 (sentential Form)

(归约)

$E \Rightarrow^* id + id * id$

$E \Rightarrow E + E$   
 $\Rightarrow id + E$   
 $\Rightarrow id + E * E$   
 $\Rightarrow id + id * E$   
 $\Rightarrow id + id * id$

施于最左变量

左句型 (left-~)

(最右归约)

$E \Rightarrow^5 id + id * id$

$E \Rightarrow E + E$   
 $\Rightarrow E + E * E$   
 $\Rightarrow E + E * id$   
 $\Rightarrow E + id * id$   
 $\Rightarrow id + id * id$

施于最右变量

右句型/规范句型  
(canonical ~)

(最左/规范归约)

$E \Rightarrow^+ id + id * id$

## ■ 定义2.5-句型、句子

设有文法 $G$ 。若 $S \xRightarrow{*} x$ ，则称 $x$ 是文法 $G$ 的句型；

若 $S \xRightarrow{*} x$ ，且 $x \in V_T^*$ ，则称 $x$ 是文法 $G$ 的句子。

$G[A]: A \rightarrow 0A1, A \rightarrow 01$

$A \Rightarrow 0A1$

$\Rightarrow 00A11$

$\Rightarrow 000A111$

$\Rightarrow 00001111$



## ■ 定义2.6- 语言

由文法**G**生成的语言记为**L(G)**,它是文法**G**的一切句子的集合。

例如:  $G[A]: A \rightarrow 0A1, A \rightarrow 01$

$$L(G) = \{0^n 1^n | n \geq 1\}$$

## ■ 定义2.7-文法等价

若**L(G1) = L(G2)**, 则称文法**G1**和**G2**是等价的。

文法**G<sub>1</sub>[A]: A → 0R** 与 **G<sub>2</sub>[A]: A → 0A1** 等价

$A \rightarrow 01$

$A \rightarrow 01$

$R \rightarrow A1$

## 2.4 文法的类型

### ■ Chomsky分类

#### □ 0型文法—短语文法

对任一产生式 $\alpha \rightarrow \beta$ ，都有 $\alpha \in (V_N \cup V_T)^+$ 且至少含有一个非终结符， $\beta \in (V_N \cup V_T)^*$

#### □ 1型文法-上下文有关文法 (CSG)

对任一产生式 $\alpha \rightarrow \beta$ ，都有 $|\beta| \geq |\alpha|$ ，仅仅 $S \rightarrow \epsilon$ 除外

#### □ 2型文法-上下文无关文法 (CFG)

对任一产生式 $\alpha \rightarrow \beta$ ，都有 $\alpha \in V_N$ ， $\beta \in (V_N \cup V_T)^*$

#### □ 3型文法-正规文法 (RG)

任一产生式 $\alpha \rightarrow \beta$ 的形式都为 $A \rightarrow aB$ 或 $A \rightarrow a$ ，其中 $A \in V_N$ ， $B \in V_N$ ， $a \in V_T$

例: 文法G[S]:

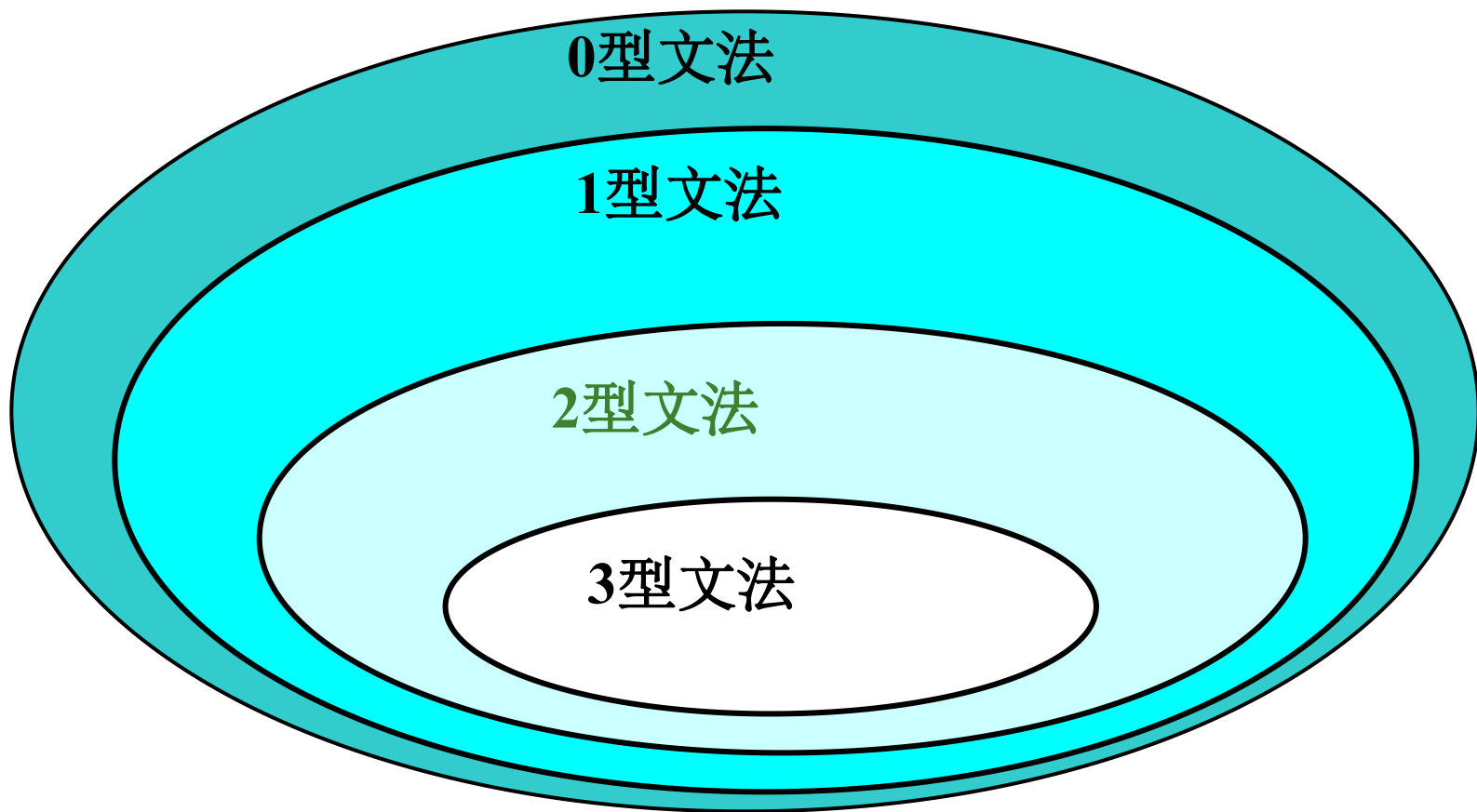
$$\begin{aligned} S &\rightarrow aSBE \\ S &\rightarrow aBE \\ EB &\rightarrow BE \\ aB &\rightarrow ab \\ bB &\rightarrow bb \\ bE &\rightarrow be \\ eE &\rightarrow ee \end{aligned}$$

文法G[S]:

$$\begin{aligned} S &\rightarrow aB|bA \\ A &\rightarrow a|aS|bAA \\ B &\rightarrow b|bS|aBB \end{aligned}$$

文法G[S]:

$$\begin{aligned} S &\rightarrow 0A|1B|0 \\ A &\rightarrow 0A|1B|0S \\ B &\rightarrow 1B|1|0 \end{aligned}$$



## ■ 例

给出一个正规文法**G**, 使  
 **$L(G) = \{a^n b^m \mid n, m \geq 1\}$**

## 2.5 上下文无关文法及其语法树

### ■ 引例

**G[S]:**

**$S \rightarrow aAS \mid a$**

**$A \rightarrow SbA \mid SS \mid ba$**

写出**aabbbaa**的最左推导和最右推导。

■ 给定文法  $G=(V_N, V_T, P, S)$ ，对于  $G$  的任何句型都能够造与之关联的语法树。这棵树满足下列4个条件：

- 每个结点都有一个标记，此标记是  $V$  的一个符号。
- 根的标记是  $S$ 。
- 若一结点  $n$  至少有一个它自己除外的子孙，并且有标记  $A$ ，则肯定  $A \in V_N$ 。
- 如果结点  $n$  有标记  $A$ ，其直接子孙结点从左到右的次序是  $n_1, n_2, \dots, n_k$ ，其标记分别为  $A_1, A_2, \dots, A_k$ ，那么  $A \rightarrow A_1 A_2 \dots A_k$  一定是  $P$  中的一个产生式。

- 一棵语法树表示了一个句型的种种可能的(但未必是所有的)不同推导过程, 包括最左(最右)推导。

例1  $G[E]$ :

$E \rightarrow E + E | E * E | (E) | id$

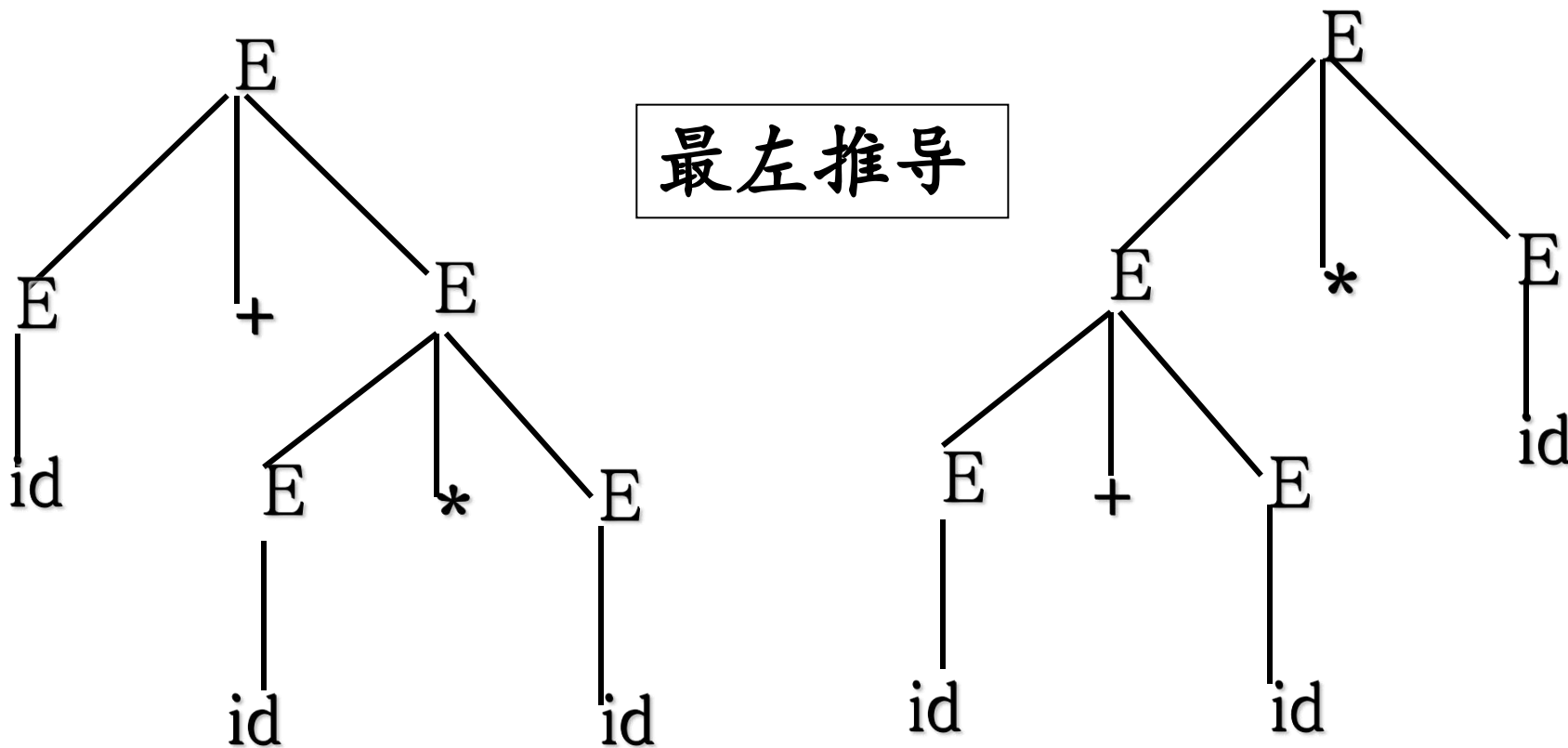
推导  $id + id * id$



# ■ 对同一句子存在两棵语法分析树

□ 在理论上不可判定

最左推导



- 若一个文法存在某个句子对应两棵不同的语法树，则称这个文法是二义的。或者说，若一个文法存在某个句子有两个不同的最左（右）推导，则称这个文法是二义的。

文法：

<条件语句>  $\rightarrow$  if <条件> then <语句>  
          | if <条件> then <语句> else <语句>

二义性的句子：

if  $e_1$  then if  $e_2$  then  $s_1$  **else**  $s_2$

■ 例2  $G[E]$ :  
 $E \rightarrow T \mid E + T$   
 $T \rightarrow F \mid T * F$   
 $F \rightarrow (E) \mid i$

- 文法的二义性和语言的二义性是两个不同的概念。因为可能有两个不同的文法**G**和**G'**，其中**G**是二义的，但是却有 $L(G)=L(G')$ ，也就是说，这两个文法所产生的语言是相同的。如果产生上下文无关语言的每个文法都是二义性，则说此语言是天生二义的。

## 2.6 句型的分析

- 从左到右读出推导树的叶子标记连接成的文法符号串为**G**的句型。
- 句型分析
  - 句型分析就是识别一个符号串是否为某文法的句型，是某个推导的构造过程。
- 分析程序（识别程序）
  - 在语言的编译实现中，把完成句型分析的程序称为分析程序或识别程序。分析算法又称识别算法。
  - 从左到右的分析算法，即总是从左到右地识别输入符号串，首先识别符号串中的最左符号，进而依次识别右边的一个符号。

## ■ 分析算法

- 自上而下分析法
- 自下而上分析法

考虑文法**G[S]**:

**$S \rightarrow cAd$**

**$A \rightarrow ab$**

**$A \rightarrow a$**

识别输入串 **$w=cabd$** 是否该文法的句子。

- 自上而下分析法的主要问题:

假定要被替换的最左非终结符是 $V$ 且有 $n$ 条产生式:  
 $V \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ , 那么如何确定用哪个右部去替换 $V$ ? (回溯的方式)

- 自下而上分析法的关键问题:

从当前串中选择一个可以归约到某个非终结符的子串 (称为“可归约串”)。

## ■ 定义2.8-短语, 直接短语, 句柄 设文法G[S]

如果有 $S \xRightarrow{*} \alpha A \delta$ 且 $A \xRightarrow{+} \beta$ , 则称 $\beta$ 是句型 $\alpha \beta \delta$ 相对于非终结符 $A$ 的短语。

如果有 $A \Rightarrow \beta$ , 则称 $\beta$ 是句型 $\alpha \beta \delta$ 相对于非终结符 $A$ 的直接短语 (简单短语) 。

一个句型的最左直接短语称为该句型的句柄。



■ 例

设文法**G[E]**:

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid i$

求句型 $i*i+i$ 的短语、直接短语和句柄

## 2.7 有关文法实用中的一些说明

- 在实用中，我们将限制文法中不得含有有害规则和多余规则。
  - **有害规则**是形如  $U \rightarrow U$  的产生式（可能引起二义性）
  - **多余规则**是文法中连一个句子的推导都用不到的规则。一种是没有任何规则的右部出现的非终结符（称为**不可到达的非终结符**）；另一种是不能从它推出终结符号串的非终结符（称为**不可终止的非终结符**）。
  - 对于文法  $G[S]$ ，为了保证任一非终结符  $A$  在句子推导中出现，必须满足如下两个条件：
    - 1)  $A$  必须在某句型中出现。
    - 2) 必须能从  $A$  推出终结符号串  $t$  来。

# 化简文法

■ 例:  $G[S]$

1)  $S \rightarrow Be$

2)  ~~$B \rightarrow Ce$~~

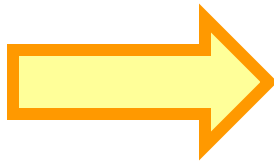
3)  $B \rightarrow Af$

4)  $A \rightarrow Ae$

5)  $A \rightarrow e$

6)  ~~$C \rightarrow Cf$~~

7)  ~~$D \rightarrow f$~~



1)  $S \rightarrow Be$

2)  $B \rightarrow Af$

3)  $A \rightarrow Ae$

4)  $A \rightarrow e$

# 上下文无关文法中的 $\varepsilon$ 规则

- 具有形式  $A \rightarrow \varepsilon$  的规则称为  **$\varepsilon$  规则**，其中  $A \in V_N$ 。
- 某些著作和讲义中限制这种规则的出现。因为  $\varepsilon$  规则会使有关文法的一些讨论和证明变得复杂。
- 两种定义的唯一差别是  $\varepsilon$  句子在不在语言中。