第2讲 上下文无关文法和分析

学习的主要内容和目标

➢学习的主要内容

- ✧ 形式语言概念
- ✧ 上下文无关文法
- ✧ 二义性

➢学习的目标

- ✧ 理解文法、语言、语法树、推导、归约、二义性、文法等价等基本概念；
- ✧ 理解文法的分类；
- ✧ 掌握文法的运用；

引例

➢如何抽象地表示程序设计语言的规则?

- y=y-1
- sum=a+b
- z=y+y

```
int b=3;
 int main()
 {
 int a=2,sum;
 sum=a+b;
 return 0;
 }
```

引例

- 如何抽象地表示程序设计语言的规则？
  - 赋值语句→变量=代数表达式
  - 代数表达式→代数表达式+代数表达式
  - 代数表达式→代数表达式*代数表达式
  - 代数表达式→（代数表达式）
  - 代数表达式→变量
  - 代数表达式→常量

文法

引例

- 文法的用途
  - 判定

    sum=a*b+
  - 生成（构造）

    x1=a+c

    x2=a+c2

形式语言的基础概念

- 字母表 (Alphabet)
  - 形式符号的集合
  - 常用 Σ 表示
  - 举例

    英文字母表 { a, b, ..., z, A, B , ..., Z }

    汉字表 { ..., 自, ..., 动 , ..., 机, ... }

    $\Sigma = \{ aa, bb \}$

    $\Sigma = \{ a, n, y,$ 任,意 $\}$

形式语言的基础概念

- 字符串 (String)
  - ✧ 字母表Σ上的一个字符串（串），或称为字，为Σ中字符构成的一个有限序列。空串常用ε表示，不包含任何字符。
  - ✧ 字符串 w 的长度，记为 $|w|$，是包含在 w 中字符的个数

    举例　$|\varepsilon| = 0$，　$|a| = 1$　　$|aabb| = ?$

形式语言的基础概念

➢字符串的连接运算

✧ 设 x, y 为串，且 $x = a_1a_2 \ldots a_m$, $y = b_1b_2 \ldots b_n$, 则 x 与 y 的连接

$$x y = a_1a_2 \ldots a_m b_1b_2 \ldots b_n$$

✧ 连接运算的性质

$$(x y) z = x (y z)$$

$$x = x \varepsilon = x$$

$$|x y| = |x| + |y|$$

形式语言的基础概念

➢字符串的幂运算

✧设 x 为串，则 x 的 n 次幂

$$x^n = \underbrace{x \ldots x\,x\,x}_{n}$$

✧ 幂运算的性质

$x^0 = \varepsilon$

$X^n = x^{n-1}x$

$x^n x^m = x^{n+m}$

形式语言的基础概念

➢字母表上的运算

✧幂运算：设 Σ 为字母表，n 为任意自然数，

- ✓ （1） $\Sigma^0 = \{ \varepsilon \}$
- ✓ （2）设 $x \in \Sigma^{n-1}$，$a \in \Sigma$， 则 $a\,x \in \Sigma^n$
- ✓ （3） Σ^n 中的元素只能由（1）和（2）生成

✧* 闭包 $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$

✧+ 闭包 $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

上下文无关文法

➢ 引例

- ✧ 赋值语句→变量=代数表达式
- ✧ 代数表达式→代数表达式+代数表达式
- ✧ 代数表达式→代数表达式*代数表达式
- ✧ 代数表达式→（代数表达式）
- ✧ 代数表达式→变量
- ✧ 代数表达式→常量

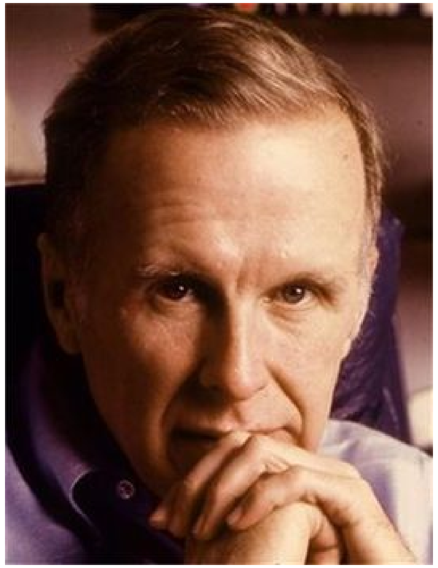
- ✧ $S \to v=E$
- ✧ $E \to E+E$
- ✧ $E \to E*E$
- ✧ $E \to (E)$
- ✧ $E \to v$
- ✧ $E \to d$

上下文无关文法

➢Backus-Naur form （巴科斯-瑙尔范式）

- ✧采用::=，而不是→
- ✧除了“|”更多的元符号
  - ✓ 尖括号“<”和“>”包含必选项；
  - ✓ 方括号“[”和“]”包含可选项；
  - ✓ 大括号“{”和“}”包含可重复0至无数次的项。
- ✧例如：
  - ✓ S::=1{0}
  - ✓ 该范式表示的语法规则定义1开头后面若干个0的序列

上下文无关文法



- 人物传记
  - ✧ 约翰·巴克斯（John Backus ）
  - ✧ Fortran 语言之父
  - ✧ 1977年获得图灵奖
    - ✓ 获奖理由：高级编程系统，程序设计语言规范的形式化定义
  - ✧ 巴科斯范式：
    - ✓ 以美国人巴科斯(Backus)和丹麦人诺尔(Naur)的名字命名的一种形式化的语法表示方法，用来描述语法的一种形式体系，是一种典型的元语言。又称巴科斯-诺尔形式(Backus-Naur form)

上下文无关文法

- 上下文无关文法的四个基本要素
  - ✧ 终结符的集合:有限符号集，相当于字母表
  - ✧ 非终结符的集合:有限变量符号的集合
  - ✧ 开始符号：一个特殊的非终结符
  - ✧ 产生式的集合

开始符号

$$
\begin{aligned}
S &\rightarrow v=E \\
E &\rightarrow E+E \\
E &\rightarrow E*E \\
E &\rightarrow (E) \\
E &\rightarrow v \\
E &\rightarrow d
\end{aligned}
$$

非终结符

终结符

上下文无关文法

➢ 上下文无关文法的形式定义

一个上下文无关文法 CFG (context-free grammars)是一个四元组

$$G = (V_N, V_T, P, S).$$

非终结符的集合 → V_N

终结符的集合 → V_T

产生式的集合 → P

开始符号 → S

满足

$V_N \cap V_T = \Phi$

$S \in V_N$

产生式形如 $A \rightarrow \alpha$，其中 $A \in V_N$, $\alpha \in (V_N \cup V_T)^*$

上下文无关文法

- **上下文无关文法实例**
  - ✧形如 $A\rightarrow \alpha_1, A\rightarrow \alpha_2, \ldots, A\rightarrow \alpha_n$ 的产生式集合可简缩记为 $A\rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$

$$
\begin{aligned}
S &\rightarrow v=E \\
E &\rightarrow E+E \\
E &\rightarrow E*E \\
E &\rightarrow (E) \\
E &\rightarrow v \\
E &\rightarrow d
\end{aligned}
\quad\Longrightarrow\quad
\begin{aligned}
S &\rightarrow v=E \\
E &\rightarrow E+E \mid E*E \mid (E) \mid v \mid d
\end{aligned}
$$

上下文无关文法

文法的运用

➢ 推导与归约

- ✧ 判定字符串是否属于文法所定义的语言；
- ✧ 推导过程
  - ✓ 将产生式的左部替换为产生式的右部
  - ✓ 一种是自上而下的方法
- ✧ 归约过程
  - ✓ 将产生式的右部替换为产生式的左部
  - ✓ 一种是自下而上的方法

上下文无关文法

➢推导实例

$$S \rightarrow v{=}E$$

$$E \rightarrow E{+}E \mid E{*}E \mid (E) \mid v \mid d$$

$v = v * (v + d)$ 的一个推导过程

$$S \xRightarrow{(1)} v{=}E \xRightarrow{(2)} v{=}E*E \xRightarrow{(3)} v{=}E*(E) \xRightarrow{(4)} v{=}v*(E)$$

$$\xRightarrow{(5)} v{=}v*(E+E) \xRightarrow{(6)} v{=}v*(E+d) \xRightarrow{(7)} v{=}v*(v+d)$$

上下文无关文法

- 归约实例

$$S \rightarrow v{=}E$$

$$E \rightarrow E{+}E \mid E{*}E \mid (E) \mid v \mid d$$

$v = v *(v+d)$ 的一个归约过程

$$v{=}v*(v+d) \xRightarrow{(1)} v{=}v*(v+E) \xRightarrow{(2)} v{=}E*(v+E) \xRightarrow{(3)} v{=}E*(E+E)$$

$$\xRightarrow{(4)} v{=}E*(E) \xRightarrow{(5)} v{=}E*E \xRightarrow{(6)} v{=}E \xRightarrow{(7)} S$$

上下文无关文法

➢推导

✧对于 CFG $G=(V_N, V_T, P, S)$，上述推导过程可用关系$\underset{G}{\Rightarrow}$描述.

✓ 设 $\alpha,\beta\in(V_N\cup V_T)^*$，$A\to\gamma$ 是一个产生式，则定义

$$\alpha A\beta \underset{G}{\Rightarrow} \alpha\gamma\beta .$$

✓ 若 G 在上下文中是明确的，则简记为$\alpha A\beta \Rightarrow \alpha\gamma\beta$.

✧ n步推导（忽略过程，只表示结论）

✓ $\alpha \overset{*}{\Rightarrow} \beta \qquad n\geq 0$

✓ $\alpha \overset{+}{\Rightarrow} \beta \qquad n\geq 1$

上下文无关文法

➢最左推导

✧推导过程中每一步总是替换出现在最左边的非终结符;

✧最左推导关系用 $\underset{lm}{\Rightarrow}$ 表示，其传递闭包用 $\overset{*}{\underset{lm}{\Rightarrow}}$ 表示;

$$S \to v{=}E$$

$$E \to E{+}E \mid E{*}E \mid (E) \mid v \mid d$$

✧关于 $v*(v+d)$ 的一个最左推导:

$$S \xRightarrow{(1)} v{=}E \xRightarrow{(2)} v{=}E*E \xRightarrow{(3)} v{=}v*E \xRightarrow{(4)} v{=}v*(E)$$

$$\xRightarrow{(5)} v{=}v*(E+E) \xRightarrow{(6)} v{=}v*(v+E) \xRightarrow{(7)} v{=}v*(v+d)$$

上下文无关文法

➢ 最右推导

- ✧ 推导过程的每一步总是替换出现在最右边的非终结符
- ✧ 最右推导关系用$\underset{rm}{\Rightarrow}$表示，其传递闭包用$\overset{*}{\underset{rm}{\Rightarrow}}$表示.
- ✧ 关于 $v*(v+d)$ 的一个最右推导：

$$S \rightarrow v{=}E$$

$$E \rightarrow E{+}E \mid E{*}E \mid (E) \mid v \mid d$$

$$S \xRightarrow{(1)} v{=}E \xRightarrow{(2)} v{=}E*E \xRightarrow{(3)} v{=}E*(E) \xRightarrow{(4)} v{=}E*(E{+}E)$$

$$\xRightarrow{(5)} v{=}E*(E{+}d) \xRightarrow{(6)} v{=}E*(v{+}d) \xRightarrow{(7)} v{=}v*(v{+}d)$$

上下文无关文法

➢句型

✧ 设 CFG $G=(V_N, V_T, P, S)$，称 $\alpha\in(V_N\cup V_T)^*$ 为 G 的一个句型，当且仅当 $S\overset{*}{\Rightarrow}\alpha$.

- ✓ 若 $S\underset{lm}{\overset{*}{\Rightarrow}}\alpha$，则 α 是一个左句型;
- ✓ 若 $S\underset{rm}{\overset{*}{\Rightarrow}}\alpha$，则 α 是一个右句型;
- ✓ 若句型 $\alpha\in V_T^*$，则称 α 为一个句子。

上下文无关文法

➢上下文无关文法的语言

✧设 $CFG\ \ G=(V_N, V_T, P, S)$，定义 G 的语言为

$$L(G)=\{\,w \mid w\in V_T{}^{*}\wedge S \stackrel{*}{\Rightarrow} w\,\}$$

➢上下文无关语言

✧ 如果一个语言 L 是某个 CFG G 的语言，则L是上下文无关语言.

➢文法等价

✧两个不同的文法G_1和G_2，他们定义的语言相同，即 $L(G_1)=L(G_1)$

上下文无关文法

- 归纳小结：语言的上下文无关文法构造
  - ✧重复型语言
  - ✧任意组合型语言
  - ✧对称型语言
  - ✧顺序型语言

上下文无关文法

- 实例分析和设计

$$\{a^n c b^m d^n \mid n,m \geq 0\}$$

Chomsky文法分类

- 0 型文法
  - ✧ 0 型文法 $G=(V_N, V_T, P, S)$ 的产生式形如 $\alpha \to \beta$，其中 $\alpha, \beta \in (V_N \cup V_T)^*$，但 α 中至少包含一个非终结符;
  - ✧ 能够用 0 型文法定义的语言称为0 型语言或递归可枚举语言.

Chomsky文法分类

- 1 型文法
  - ✧ 1 型文法 $G = (V_N, V_T, P, S)$的产生式形如$\alpha \rightarrow \beta$，满足 $|\alpha| \leq |\beta|$，只有 $S \rightarrow \varepsilon$是右部包含$\varepsilon$例外，但此时S不能出现在任意产生式右部；
  - ✧ 1 型文法也称为上下文有关文法
  - ✧ 能够用 1 型文法定义的语言称为 1 型语言或 上下文有关语言.

Chomsky文法分类

➢2 型文法

✧ 2 型文法 $G=(V_N, V_T, P, S)$的产生式形如 $\alpha\rightarrow\beta$, 其中$\alpha\in V_N$，$\beta\in(V_N\cup V_T)^*$.

✧ 2 型文法也称为上下文无关文法.

✧ 能够用 2 型文法定义的语言称为2 型语言，或上下文无关语言.

Chomsky文法分类

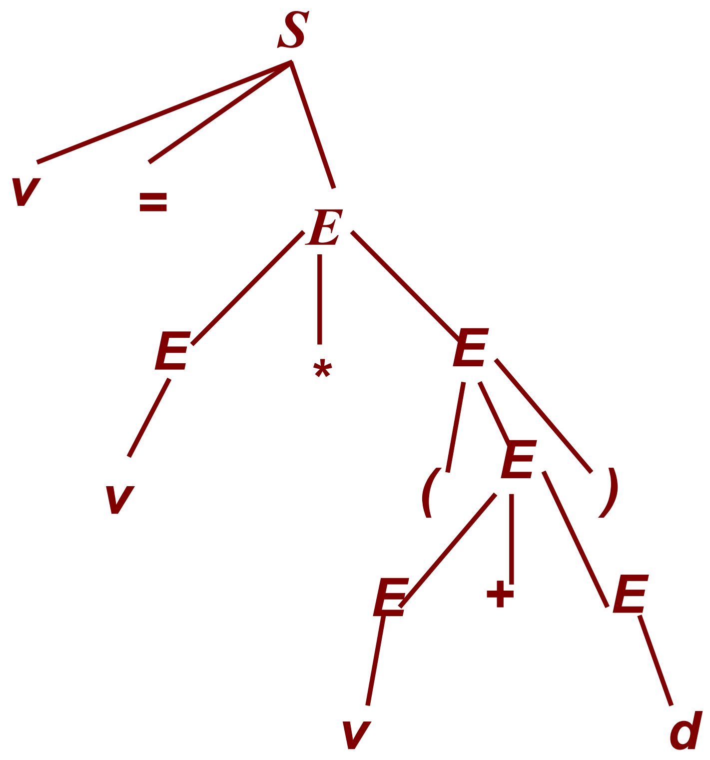
- 3 型文法
  - ✧ 3 型文法 $G=(V_N, V_T, P, S)$也称为正规文法，产生式形如 $\alpha \to \beta$
    - ✓ 右线性的产生式满足$\alpha \in V_N$， $\beta=aA$或者a或者ε，$a\in V_T$，$A\in V_N$
    - ✓ 左线性的产生式满足$\alpha\in V_N$，$\beta=Aa$, a或者ε，且$a\in V_T$，$A\in V_N$
  - ✧ 能用 3 型文法定义的语言称为3 型语言，或正规语言.

语法分析树

- 对于 CFG $G=(V_N, V_T, P, S)$，语法分析树是满足下列条件的树
  - ✧ 每个内部结点由一个非终结符标记;
  - ✧ 每个叶结点由一个非终结符，或一个终结符，或 ε 来标记； 当标记为 ε 时，它必是其父结点唯一的孩子;
  - ✧ 如果一个内部结点标记为 A，而其孩子从左至右分别标记为$X_1,X_2, \ldots,X_k$，则 $A \rightarrow X_1X_2\ldots X_k$ 是 P 中的一个产生式. 注意：只有 k=1 时上述 X_i才有可能为ε，此时结点 A 只有唯一的孩子，且 $A \rightarrow \varepsilon$ 是P 中的一个产生式。

语法分析树

➢ 引例



$$S \rightarrow v{=}E$$

$$E \rightarrow E{+}E \mid E{*}E \mid (E) \mid v \mid d$$

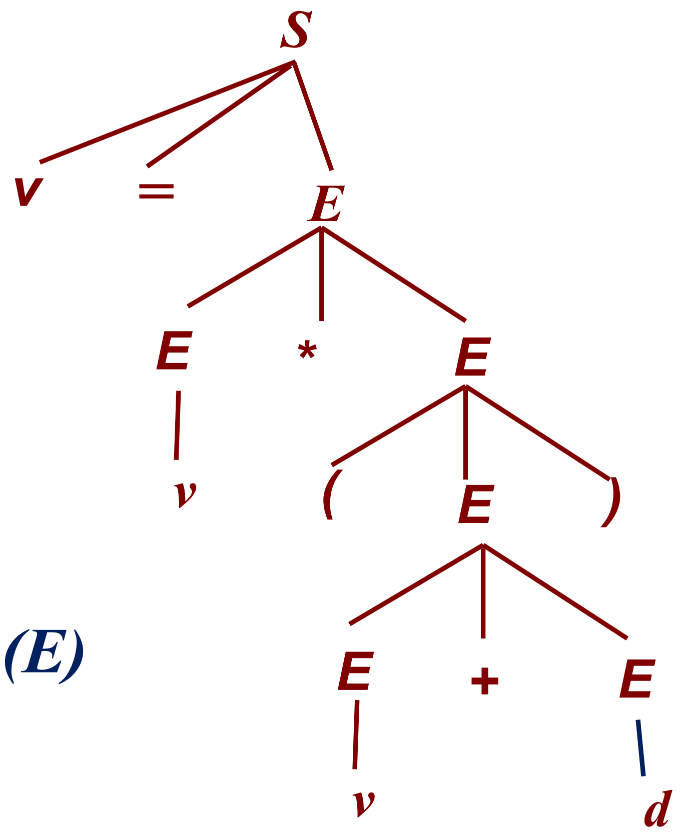
语法分析树

- 推导过程自下而上构造了一棵树

$$S \rightarrow v=E$$

$$E \rightarrow E+E \mid E*E \mid (E) \mid v \mid d$$

句型 $v=v*(v+d)$



$$S \xRightarrow{(1)} v=E \xRightarrow{(2)} v=E*E \xRightarrow{(3)} v=E*(E) \xRightarrow{(4)} v=v*(E)$$

$$\xRightarrow{(5)} v=v*(E+E) \xRightarrow{(6)} v=v*(E+d) \xRightarrow{(7)} v=v*(v+d)$$

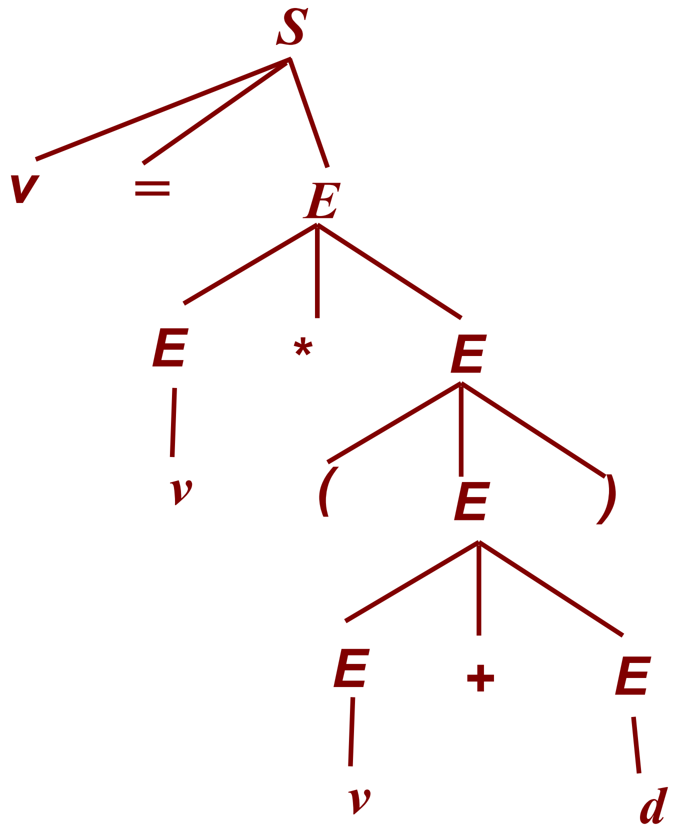
语法分析树

- 归约过程自下而上构造了一棵树

$S \rightarrow v{=}E$

$E \rightarrow E{+}E \mid E{*}E \mid (E) \mid v \mid d$

句型 $v{=}v*(v+d)$



$v{=}v*(v+d) \xrightarrow{(1)} v{=}v*(v+E)$

$v{=}v*(v+E) \xrightarrow{(2)} v{=}E*(v+E)$

$v{=}E*(v+E) \xrightarrow{(3)} v{=}E*(E+E)$

$v{=}E*(E+E) \xrightarrow{(4)} v{=}E*(E)$

$v{=}E*(E) \xrightarrow{(5)} v{=}E*E$

$v{=}E*E \xrightarrow{(6)} v{=}E$

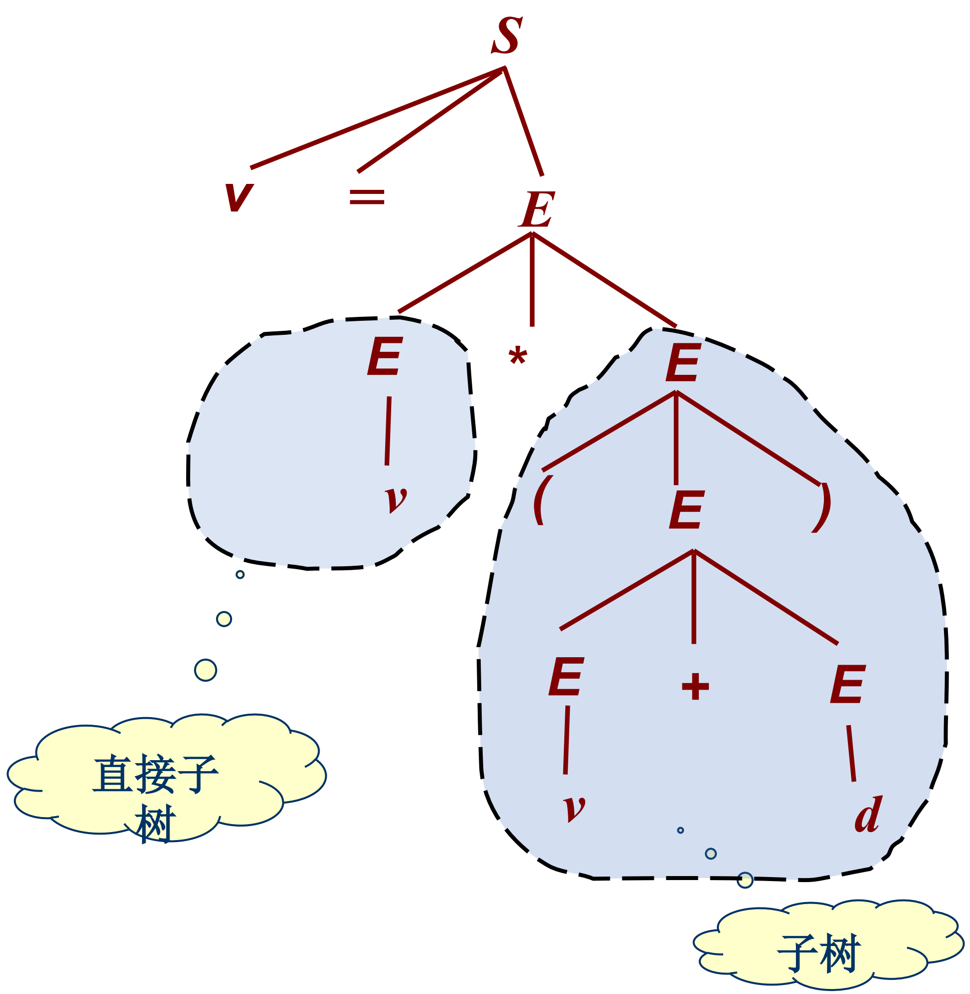
$v{=}E \xrightarrow{(7)} S$

语法分析树

- 子树和直接子树
  - ✧ 句型： $v=v*(v+d)$

$$S \rightarrow v=E$$

$$E \rightarrow E+E \mid E*E \mid (E) \mid v \mid d$$



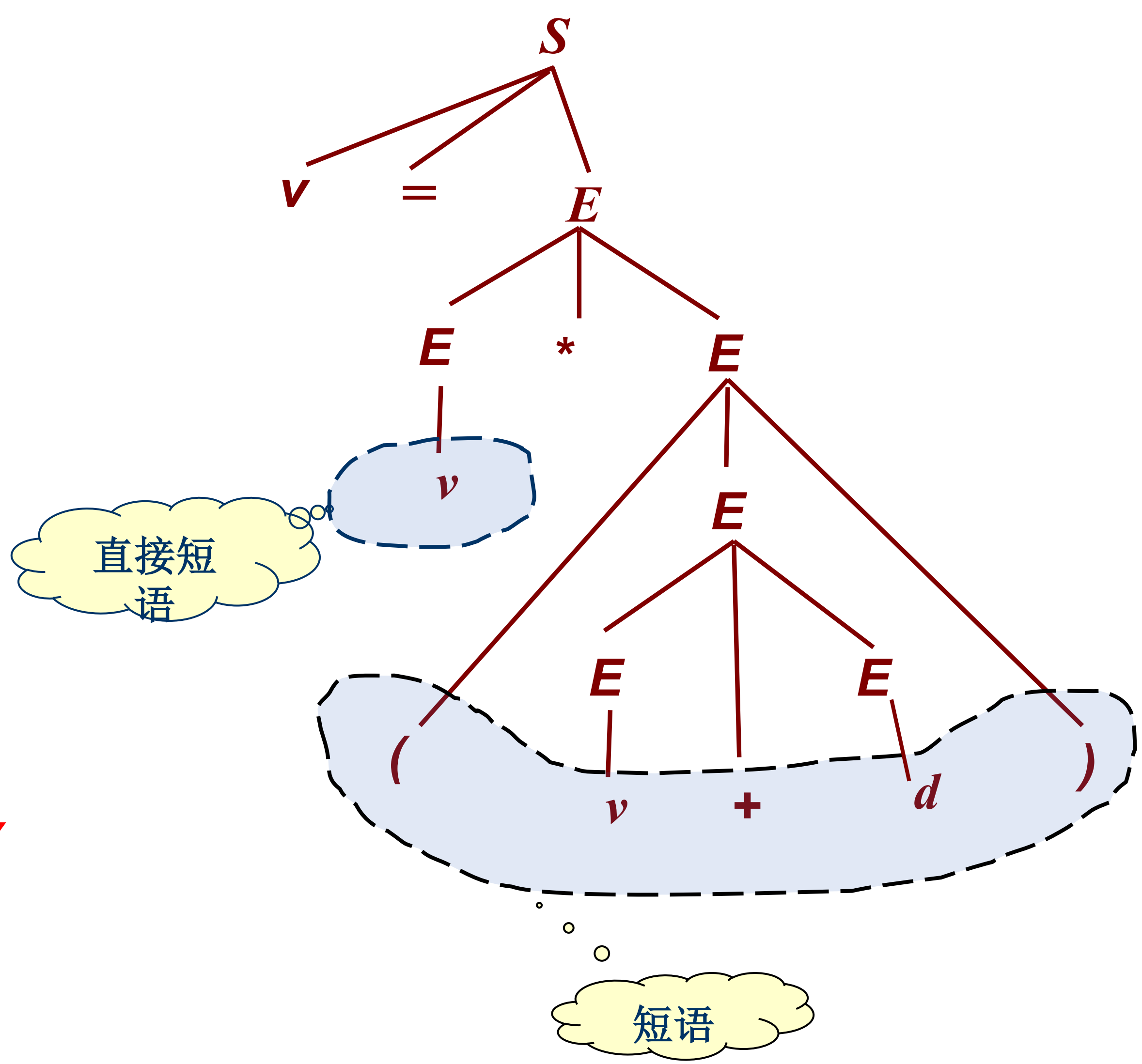
语法分析树

- ➢ 短语、直接短语
  - ✧ 句型： v= v *(v+d)

$$S \rightarrow v{=}E$$

$$E \rightarrow E{+}E \mid E{*}E \mid (E) \mid v \mid d$$

α是句型w的短语当且仅当 α是w的子串且$A \overset{+}{\Rightarrow} \alpha$



语法分析树

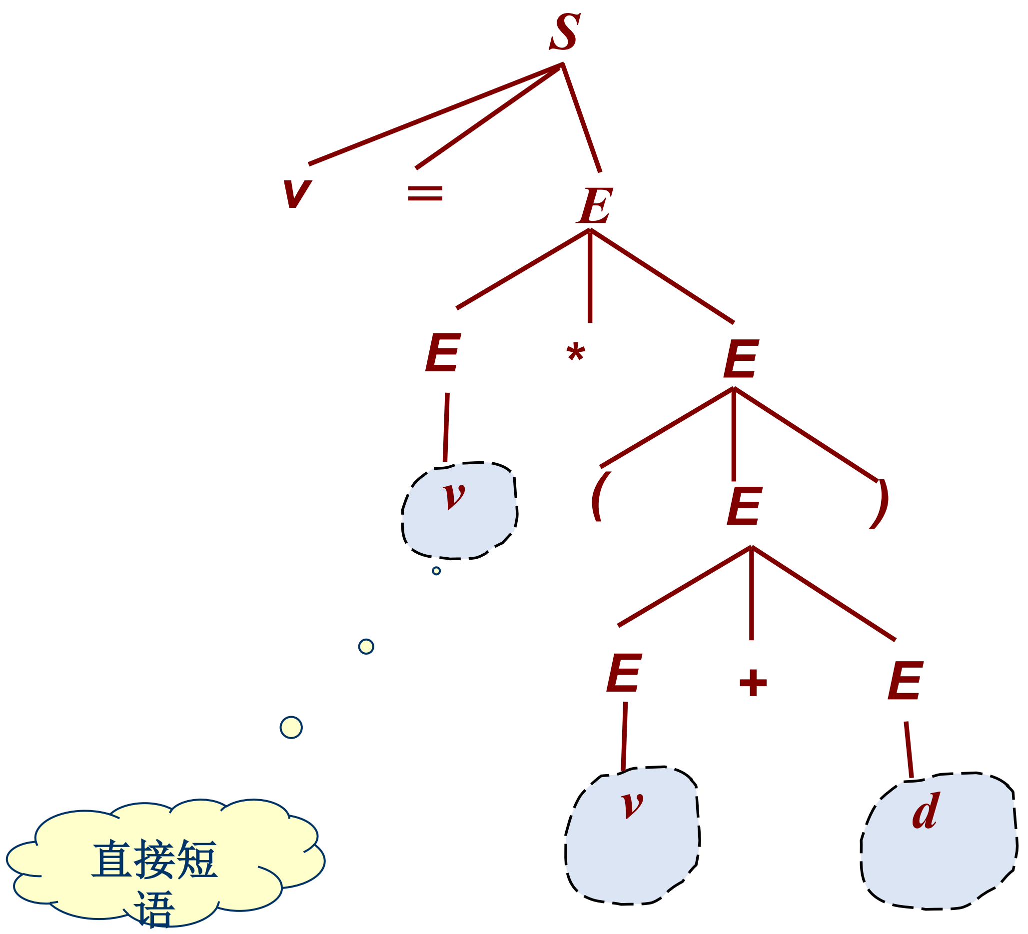
➢ 句柄

✧ 一个句型的最左的直接短语称之为句柄;

✧ 句型： $v = v * (v + d)$

$S \rightarrow v=E$

$E \rightarrow E+E \mid E*E \mid (E) \mid v \mid d$



语法分析树

➢归约、推导与分析树之间关系

✧ 设 CFG $G=(V_N, V_T, P, S)$. 以下命题是相互等价的:

① 字符串 $w \in V_T^*$ 可以归约到非终结符A;

② $A \overset{*}{\Rightarrow} w$;

③ $A \underset{lm}{\overset{*}{\Rightarrow}} w$;

④ $A \underset{rm}{\overset{*}{\Rightarrow}} w$;

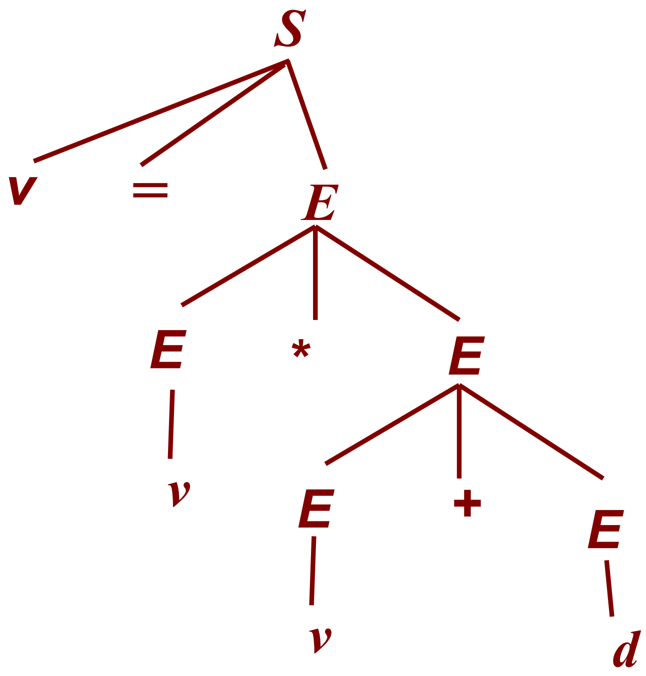
⑤ 存在一棵根结点为 A 的分析树，其果实为 w.

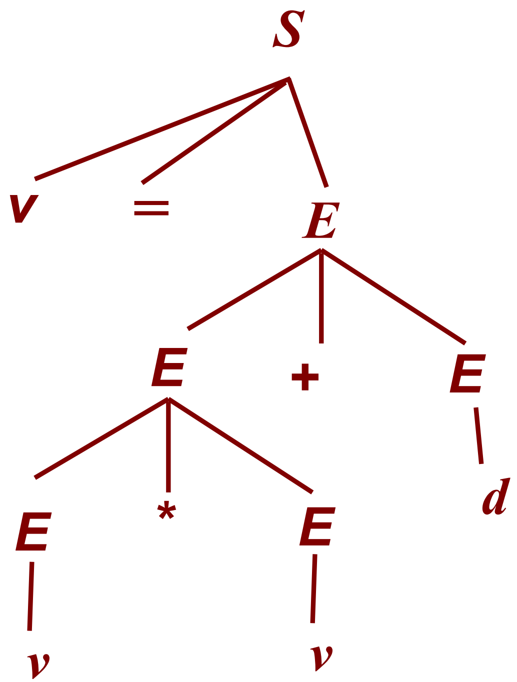
文法的二义性

➢ 二义性

$$S \rightarrow v{=}E$$

$$E \rightarrow E{+}E \mid E{*}E \mid (E) \mid v \mid d$$





文法的二义性

- 二义性
  - ✧ 二义性的危害?

文法的二义性

- ➢CFG $G=(V_N, V_T, P, S)$ 为二义的，如果对某个 $w \in V_T^*$，存在两棵不同的分析树，它们的根结点都为开始符号 S，果实都为 w.
  - ✧ 如果对每一 $w \in T^*$，至多存在一棵这样的分析树，则 G 为无二义的.

文法的二义性

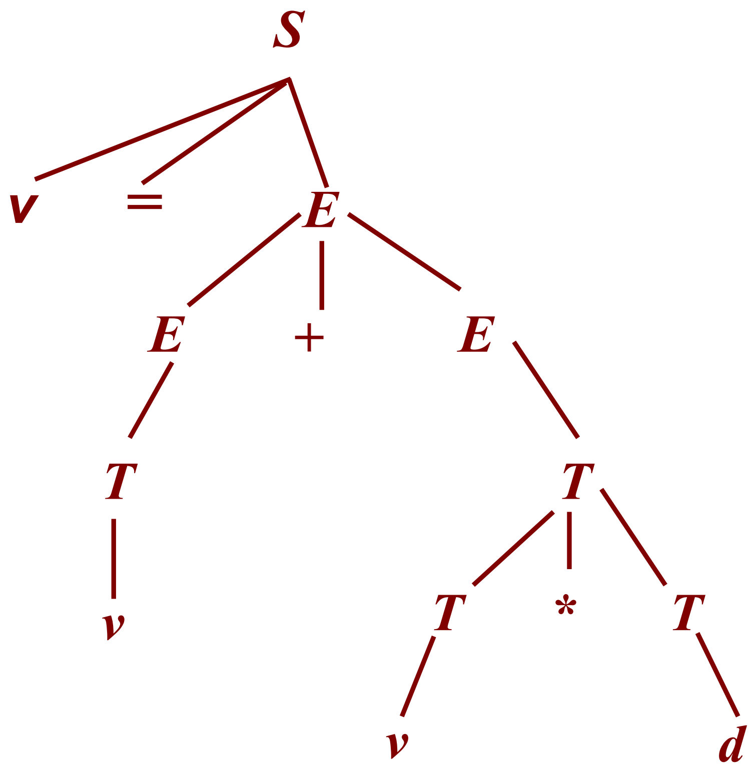
➢ 消除二义性

✧ 采用 算符优先级联方法变 换文法

$$S \rightarrow v{=}E$$
$$E \rightarrow E{+}E \mid E{*}E \mid (E) \mid v \mid d$$

⇓

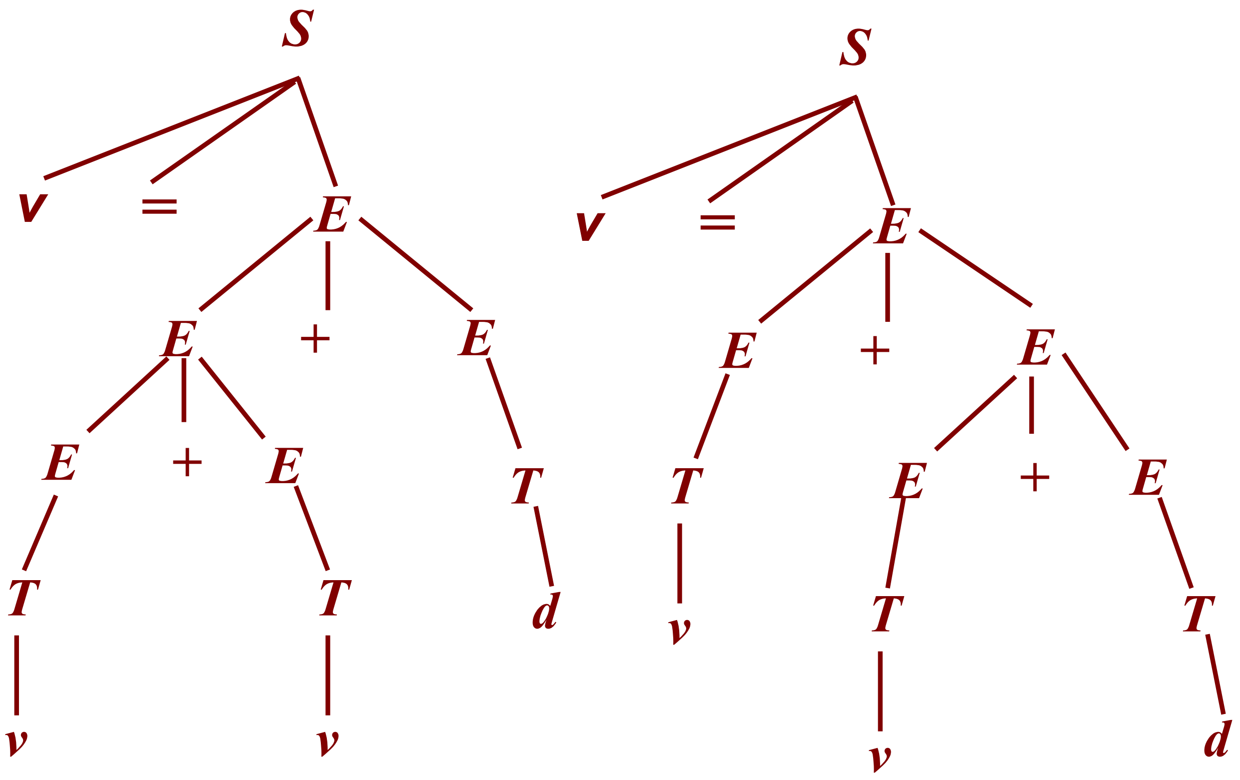
$$S \rightarrow v{=}E$$
$$E \rightarrow E + E \mid T$$
$$T \rightarrow T * T \mid (E) \mid v \mid d$$



文法的二义性

- 消除二义性
  - ✧ 串v= $v + v + d$ 存在 不同的分析树

$$
\begin{aligned}
&S \rightarrow v{=}E \\
&E \rightarrow E + E \mid T \\
&T \rightarrow T * T \mid (E) \mid v \mid d
\end{aligned}
$$



文法的二义性

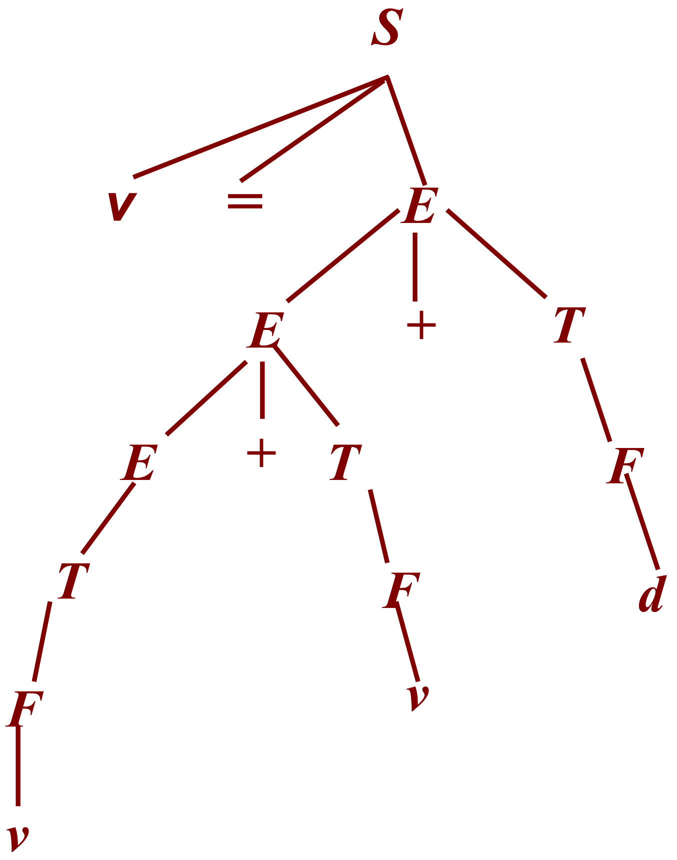
- 消除二义性
  - 采用左结合方法
  - 串 $v=v+v+d$ 存在唯一的 分析树

$$S \rightarrow v=E$$

$$E \rightarrow E+T \mid T$$

$$T \rightarrow T*F \mid F$$

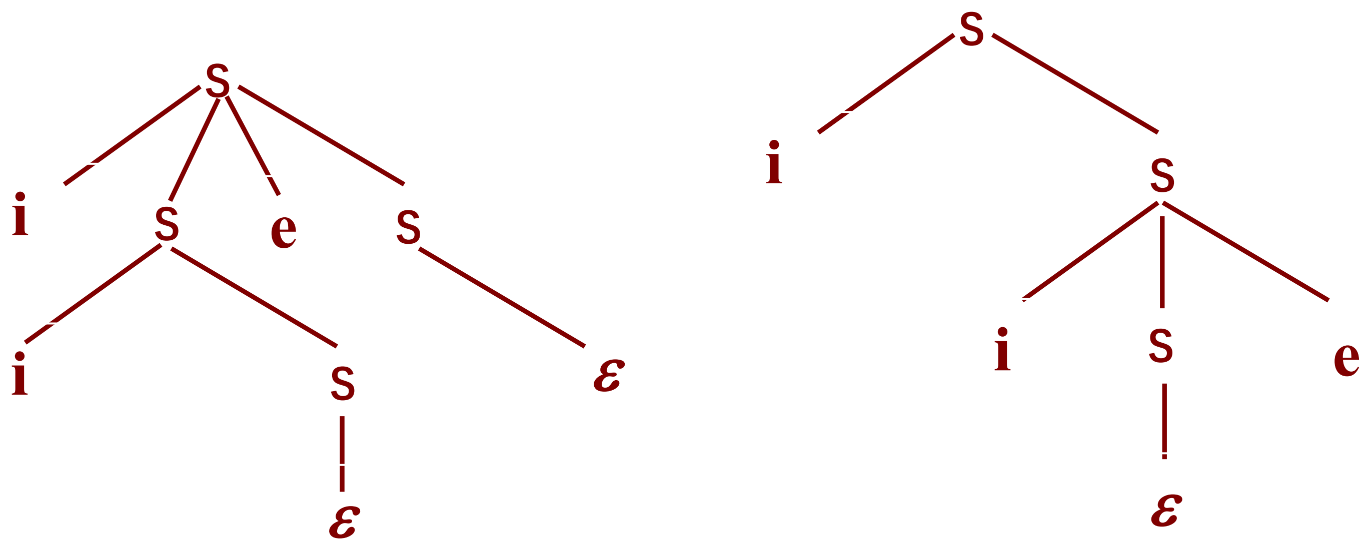
$$F \rightarrow (E) \mid v \mid d$$



文法的二义性

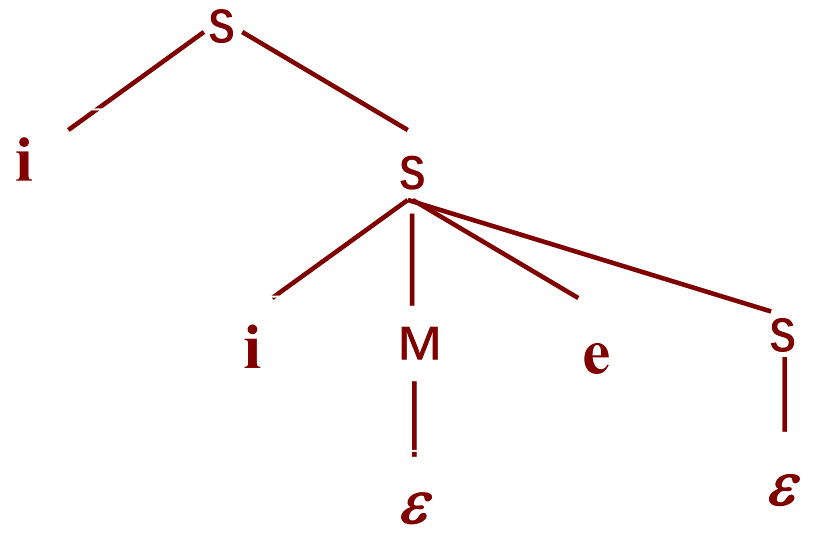
➢ **消除二义性**

- ✧ 采用最近嵌套方法消除 悬挂 else 二义性
- ✧ $S \rightarrow \varepsilon \mid iS \mid iSeS$
- ✧ iie



文法的二义性

- 消除二义性
  - 采用最近嵌套方法消除 悬挂 else 二义性
  - $S \to \varepsilon \mid iS \mid iSeS$
  - iie



$$S \to \varepsilon \mid iS \mid iMeS$$

$$M \to \varepsilon \mid iMeM$$

That's all for today.