



第二章 语言及其文法

基本概念

哈尔滨工业大学 陈鄞



字母表 (*Alphabet*)

复习时间：2.28

- 字母表 Σ 是一个有穷符号集合
- 符号：字母、数字、标点符号、...

例：

- 二进制字母表：{ 0,1 }
- *ASCII* 字符集
- *Unicode* 字符集

字母表上的运算

➤ 字母表 Σ_1 和 Σ_2 的 **乘积** (*product*)

$$\Sigma_1 \Sigma_2 = \{ab \mid a \in \Sigma_1, b \in \Sigma_2\}$$

例： $\{0, 1\} \{a, b\} = \{0a, 0b, 1a, 1b\}$

字母表上的运算

➤ 字母表 Σ_1 和 Σ_2 的乘积(*product*)

➤ 字母表 Σ 的 n 次幂(*power*)

$$\begin{cases} \Sigma^0 = \{ \epsilon \} & n \text{ 可以为 } 0, \text{ 表示空串, 用 } \epsilon \text{ (yi fuxi lou) 表示} \\ \Sigma^n = \Sigma^{n-1} \Sigma, n \geq 1 \end{cases}$$

例: $\{0, 1\}^3 = \{0, 1\} \{0, 1\} \{0, 1\}$
 $= \{000, 001, 010, 011, 100, 101, 110, 111\}$

字母表的 n 次幂: 长度为 n 的符号串构成的集合

字母表上的运算

- 字母表 Σ_1 和 Σ_2 的乘积(*product*)
- 字母表 Σ 的 n 次幂(*power*)
- 字母表 Σ 的正闭包(*positive closure*)
 - $\Sigma^+ = \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \dots$ 长度为1, 2, ..., n的符号串集合

例： $\{a, b, c, d\}^+ = \{a, b, c, d,$
 $aa, ab, ac, ad, ba, bb, bc, bd, \dots,$
 $aaa, aab, aac, aad, aba, abb, abc, \dots\}$

字母表的正闭包：长度正数的符号串构成的集合

字母表上的运算

- 字母表 Σ_1 和 Σ_2 的**乘积**(*product*)
 - 字母表 Σ 的 **n 次幂**(*power*)
 - 字母表 Σ 的**正闭包**(*positive closure*)
 - 字母表 Σ 的**克林闭包**(*Kleene closure*)
- $$\Sigma^* = \Sigma^0 \cup \Sigma^+ = \Sigma^0 \cup \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \dots$$

例： $\{a, b, c, d\}^* = \{\epsilon, a, b, c, d, aa, ab, ac, ad, ba, bb, bc, bd, \dots, aaa, aab, aac, aad, aba, abb, abc, \dots\}$

字母表的克林闭包：任意符号串（长度可以为零）构成的集合

串 (*String*)

- 设 Σ 是一个字母表, $\forall x \in \Sigma^*$, x 称为是 Σ 上的一个 **串**
 - 串是字母表中 **符号** 的一个 **有穷序列**
- 串 s 的 **长度**, 通常记作 $|s|$, 是指 s 中 **符号的个数**
 - 例: $|aab|=3$
- **空串** 是 **长度为0** 的串, 用 ε (*epsilon*) 表示
 - $|\varepsilon|=0$

串上的运算——连接

- 如果 x 和 y 是串，那么 x 和 y 的 **连接** (*concatenation*) 是把 y 附加到 x 后面而形成的串，记作 xy
- 例如，如果 $x=dog$ 且 $y=house$ ，那么 $xy=doghouse$
- **空串** 是连接运算的 **单位元** (*identity*)，即，对于任何串 s 都有， $\epsilon s = s\epsilon = s$

设 x, y, z 是三个字符串，如果 $x=yz$ ，则称 y 是 x 的 **前缀**， z 是 x 的 **后缀**

串上的运算——幂

➤ 串 s 的幂运算

$$\begin{cases} s^0 = \varepsilon, \\ s^n = s^{n-1}s, n \geq 1 \end{cases}$$

➤ $s^1 = s^0 s = \varepsilon s = s, s^2 = ss, s^3 = sss, \dots$

➤ 例：如果 $s = ba$ ，那么 $s^1 = ba, s^2 = baba, s^3 = bababa, \dots$

串 s 的 n 次幂：将 n 个 s 连接起来



第二章 语言及其文法

基本概念

哈尔滨工业大学 陈鄞





第二章 语言及其文法

文法的定义

哈尔滨工业大学 陈鄞



自然语言的例子——句子的构成规则

- <句子> → <名词短语> <动词短语>
- <名词短语> → <形容词> <名词短语>
- <名词短语> → <名词>
- <动词短语> → <动词> <名词短语>
- <形容词> → *little*
- <名词> → *boy*
- <名词> → *apple*
- <动词> → *eat*

未用尖括号括起来部分表示
语言的基本符号

尖括号括起来部分称为语法成分

文法的形式化定义

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

➤ V_T : 终结符集合

终结符 (*terminal symbol*) 是文法所定义的语言的**基本符号**, 有时也称为*token*

➤ 例: $V_T = \{ apple, boy, eat, little \}$

文法的形式化定义

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

➤ V_T : 终结符集合

➤ V_N : 非终结符集合

非终结符(*nonterminal*) 是用来表示语法成分的符号,
有时也称为“语法变量”

➤ 例: $V_N = \{ \langle \text{句子} \rangle, \langle \text{名词短语} \rangle, \langle \text{动词短语} \rangle, \langle \text{名词} \rangle, \dots \}$

文法的形式化定义

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

➤ V_T : 终结符集合

$$V_T \cap V_N = \Phi$$

➤ V_N : 非终结符集合

$$V_T \cup V_N: \text{文法符号集}$$

➤ P : 产生式集合

产生式(*production*)描述了将终结符和非终结符组合成串的方法
产生式的一般形式:

$$\alpha \rightarrow \beta$$

读作: α 定义为 β

➤ $\alpha \in (V_T \cup V_N)^+$, 且 α 中至少包含 V_N 中的一个元素: 称为产生式的**头**
(*head*) 或**左部**(*left side*) α 不为空

➤ $\beta \in (V_T \cup V_N)^*$: 称为产生式的**体**(*body*)或**右部**(*right side*)

文法的形式化定义

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

➤ V_T : 终结符集合

➤ V_N : 非终结符集合

➤ P : 产生式集合

产生式(*production*)描述了将终结符和非终结符组合成串的方法
产生式的一般形式:

$$\alpha \rightarrow \beta$$

➤ 例: $P = \left\{ \begin{array}{l} \langle \text{句子} \rangle \rightarrow \langle \text{名词短语} \rangle \langle \text{动词短} \rangle, \\ \langle \text{名词短语} \rangle \rightarrow \langle \text{形容词} \rangle \langle \text{名词短语} \rangle, \\ \dots \end{array} \right\}$

文法的形式化定义

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

- V_T : 终结符集合
- V_N : 非终结符集合
- P : 产生式集合
- S : 开始符号

$S \in V_N$ 。开始符号 (start symbol) 表示的是该文法中最大的语法成分

➤ 例: $S = \langle \text{句子} \rangle$

前面的例子

文法的形式化定义

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

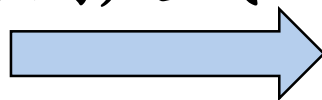
- V_T : 终结符集合
- V_N : 非终结符集合
- P : 产生式集合
- S : 开始符号

例: $G = (\{ \text{id}, +, *, (,) \}, \{E\}, P, E)$

$$P = \{ \begin{aligned} E &\rightarrow E + E, \\ E &\rightarrow E * E, \\ E &\rightarrow (E), \\ E &\rightarrow \text{id} \end{aligned} \}$$

约定:

不引起歧义的前提下, 可以只写产生式


$$\begin{aligned} G : E &\rightarrow E + E \\ E &\rightarrow E * E \\ E &\rightarrow (E) \\ E &\rightarrow \text{id} \end{aligned}$$

产生式的简写

➤ 对一组有相同左部的 α 产生式

$$\alpha \rightarrow \beta_1, \alpha \rightarrow \beta_2, \dots, \alpha \rightarrow \beta_n$$

可以简记为:

$$\alpha \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$$

读作： α 定义为 β_1 ，或者 β_2 ，...，或者 β_n 。

$\beta_1, \beta_2, \dots, \beta_n$ 称为 α 的候选式(Candidate)

➤ 例

$$E \rightarrow E + E$$

$$E \rightarrow E * E$$

$$E \rightarrow (E)$$

$$E \rightarrow \text{id}$$



$$E \rightarrow E + E \mid E * E \mid (E) \mid \text{id}$$

符号约定

➤ 下述符号是终结符

➤ (a) 字母表中排在前面的小写字母，如 a 、 b 、 c

➤ (b) 运算符，如 $+$ 、 $*$ 等

➤ (c) 标点符号，如括号、逗号等

➤ (d) 数字 0 、 1 、 \dots 、 9

➤ (e) 粗体字符串，如 `id`、`if`等

符号约定

➤ 下述符号是终结符

➤ 下述符号是非终结符

➤ (a) 字母表中排在前面的大写字母，如 A 、 B 、 C

➤ (b) 字母 S 。通常表示开始符号

➤ (c) 小写、斜体的名字，如 $expr$ 、 $stmt$ 等

➤ (d) 代表程序构造的大写字母。如 E (表达式)、 T (项)

和 F (因子)

factor

expression

term

符号约定

- 下述符号是**终结符**
- 下述符号是**非终结符**

终结符	a, b, c	终结符号串	u, v, \dots, z
非终结符	A, B, C		
文法符号	X, Y, Z	文法符号串	α, β, γ

- 字母表中**排在后面的大写字母** (如 X 、 Y 、 Z)
表示**文法符号** (即终结符或非终结符)
- 字母表中**排在后面的小写字母** (主要是 u 、 v 、 \dots 、 z)
表示**终结符号串** (包括空串)
- **小写希腊字母**, 如 α 、 β 、 γ , 表示**文法符号串** (包括空串)
- 除非特别说明, **第一个产生式的左部**就是**开始符号**



第二章 语言及其文法

文法的定义

哈尔滨工业大学 陈鄞





第二章 语言及其文法

语言的定义

哈尔滨工业大学 陈鄞



自然语言的例子

文法：

- ① <句子> → <名词短语> <动词短语>
- ② <名词短语> → <形容词> <名词短语>
- ③ <名词短语> → <名词>
- ④ <动词短语> → <动词> <名词短语>
- ⑤ <形容词> → *little*
- ⑥ <名词> → *boy*
- ⑦ <名词> → *apple*
- ⑧ <动词> → *eat*

有了文法（语言规则），如何判定一个词串是否是满足文法的句子？

单词串： *little boy eats apple*

推导 (Derivations) 和归约 (Reductions)

- 给定文法 $G=(V_T, V_N, P, S)$, 如果 $\alpha \rightarrow \beta \in P$, 那么可以将符号串 $\gamma\alpha\delta$ 中的 α 替换为 β , 也就是说, 将 $\gamma\alpha\delta$ 重写 (rewrite) 为 $\gamma\beta\delta$, 记作 $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$ 。此时, 称文法中的符号串 $\gamma\alpha\delta$ 直接推导 (directly derive) 出 $\gamma\beta\delta$
- 简而言之, 就是用产生式的右部替换产生式的左部

推导 (Derivations) 和归约 (Reductions)

➤ 如果 $\alpha_0 \Rightarrow \alpha_1, \alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \dots, \alpha_{n-1} \Rightarrow \alpha_n$, 则可以记作 $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \dots \Rightarrow \alpha_{n-1} \Rightarrow \alpha_n$, 称符号串 α_0 经过 n 步推导出 α_n , 可简记为 $\alpha_0 \Rightarrow^n \alpha_n$

➤ $\alpha \Rightarrow^0 \alpha$ 一步推导称为直接推导

➤ \Rightarrow^+ 表示 “经过正数步推导”

➤ \Rightarrow^* 表示 “经过若干 (可以是0) 步推导”

推导 (Derivations) 和 归约 (Reductions)

例

文法:

- ① <句子> → <名词短语> <动词短语>
- ② <名词短语> → <形容词> <名词短语>
- ③ <名词短语> → <名词>
- ④ <动词短语> → <动词> <名词短语>
- ⑤ <形容词> → *little*
- ⑥ <名词> → *boy*
- ⑦ <名词> → *apple*
- ⑧ <动词> → *eat*

<句子> ⇒ <名词短语> <动词短语>
⇒ <形容词> <名词短语> <动词短语>
⇒ *little* <名词短语> <动词短语>
⇒ *little* <名词> <动词短语>
⇒ *little boy* <动词短语>
⇒ *little boy* <动词> <名词短语>
⇒ *little boy eats* <名词短语>
⇒ *little boy eats* <名词>
⇒ *little boy eats apple*

推导

归约

回答前面的问题

➤ 有了文法（语言规则），如何判定某一词串是否是该语言的句子？

➤ 句子的**推导**（派生）- 从**生成**语言的角度

从开始符号能推导出该词串

➤ 句子的**归约**

- 从**识别**语言的角度

从词串能归约出开始符号

} 均根据规则

句型 and 句子

➤ 如果 $S \Rightarrow^* \alpha$, $\alpha \in (V_T \cup V_N)^*$, 则称 α 是 G 的一个 **句型**

(*sentential form*)

➤ 一个句型中既可以包含**终结符**, 又可以包含**非终结符**, 也可能是**空串**

➤ 如果 $S \Rightarrow^* w$, $w \in V_T^*$, 则称 w 是 G 的一个 **句子** (*sentence*)

➤ 句子是**不包含非终结符的句型**

例

<句子>⇒<名词短语><动词短语>

⇒<形容词><名词短语><动词短语>

⇒ *little* <名词短语><动词短语>

⇒ *little* <名词><动词短语>

⇒ *little boy* <动词短语>

⇒ *little boy* <动词><名词短语>

⇒ *little boy eats* <名词短语>

⇒ *little boy eats* <名词>

句子 → ⇒ *little boy eats apple*

句型

语言的形式化定义

➤ 由文法 G 的开始符号 S 推导出的所有句子构成的集合称为**文法 G 生成的语言**，记为 $L(G)$ 。
即

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

文法 $E \rightarrow E+E \mid E * E \mid (E) \mid \text{id}$
生成的语言中包含多少个句子？

无穷多个

例

➤ 文法 G

$$\textcircled{1} S \rightarrow L \mid LT$$

$$\textcircled{2} T \rightarrow L \mid D \mid TL \mid TD$$

$$\textcircled{3} L \rightarrow a \mid b \mid c \mid \dots \mid z \quad \text{letter}$$

$$\textcircled{4} D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid \dots \mid 9 \quad \text{digit}$$

该文法生成的语言是：标识符

请写出无符号整数
和浮点数的文法

$$T \Rightarrow TL$$

$$\Rightarrow TDL$$

$$\Rightarrow TDDL$$

$$\Rightarrow TLDDL$$

...

$$\Rightarrow TD\dots LDDL$$

$$\Rightarrow DD\dots LDDL$$

T : 字母数字串

语言上的运算

运算	定义和表示
L 和 M 的并	$L \cup M = \{s \mid s \text{ 属于 } L \text{ 或者 } s \text{ 属于 } M\}$
L 和 M 的连接	$LM = \{st \mid s \text{ 属于 } L \text{ 且 } t \text{ 属于 } M\}$
L 的幂	$\begin{cases} L^0 = \{ \varepsilon \} \\ L^n = L^{n-1}L, n \geq 1 \end{cases}$
L 的Kleene闭包	$L^* = \bigcup_{i=0}^{\infty} L^i$
L 的正闭包	$L^+ = \bigcup_{i=1}^{\infty} L^i$

例：令 $L = \{A, B, \dots, Z, a, b, \dots, z\}$, $D = \{0, 1, \dots, 9\}$ 。则 $L(L \cup D)^*$ 表示的语言是标识符

字母打头的标识符



第二章 语言及其文法

语言的定义

哈尔滨工业大学 陈鄞





提纲

2.1 基本概念


2.2 文法的定义

2.3 语言的定义

2.4 文法的分类

2.5 CFG的语法分析树

2.6 非上下文无关的语言构造



第二章 语言及其文法

文法的分类

哈尔滨工业大学 陈勤



Chomsky 文法分类体系

➤ 0型文法 (*Type-0 Grammar*)

➤ 1型文法 (*Type-1 Grammar*)

➤ 2型文法 (*Type-2 Grammar*)

➤ 3型文法 (*Type-3 Grammar*)

0型文法 (*Type-0 Grammar*)

$$\alpha \rightarrow \beta$$

➤ 无限制文法(*Unrestricted Grammar*) / 短语结构文法

(*Phrase Structure Grammar, PSG*)

➤ $\forall \alpha \rightarrow \beta \in P$, α 中至少包含1个非终结符

➤ 0型语言

➤ 由0型文法 G 生成的语言 $L(G)$

1型文法 (Type-1 Grammar)

$$\alpha \rightarrow \beta$$

➤ 上下文有关文法 (Context-Sensitive Grammar, CSG)

➤ $\forall \alpha \rightarrow \beta \in P, |\alpha| \leq |\beta|$

➤ 产生式的一般形式: $\alpha_1 \overset{\text{非终结符}}{A} \alpha_2 \rightarrow \alpha_1 \overset{\text{文法符号串}}{\beta} \alpha_2 (\beta \neq \varepsilon)$

➤ 上下文有关语言 (1型语言)

➤ 由上下文有关文法 (1型文法) G 生成的语言 $L(G)$

CSG 中不包含 ε -产生式

2型文法 (*Type-2 Grammar*)

$$\alpha \rightarrow \beta$$

➤ 上下文无关文法 (*Context-Free Grammar, CFG*)

➤ $\forall \alpha \rightarrow \beta \in P, \alpha \in V_N$ 非终结符

➤ 产生式的一般形式: $A \rightarrow \beta$

例:

$$S \rightarrow L \mid LT$$

$$T \rightarrow L \mid D \mid TL \mid TD$$

$$L \rightarrow a \mid b \mid c \mid d \mid \dots \mid z$$

$$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid \dots \mid 9$$

2型文法 (Type-2 Grammar)

$$\alpha \rightarrow \beta$$

➤ 上下文无关文法 (Context-Free Grammar, CFG)

➤ $\forall \alpha \rightarrow \beta \in P, \alpha \in V_N$

➤ 产生式的一般形式: $A \rightarrow \beta$

➤ 上下文无关语言 (2型语言) 下推自动机

➤ 由上下文无关文法 (2型文法) G 生成的语言 $L(G)$

3型文法 (Type-3 Grammar)

$$\alpha \rightarrow \beta$$

➤ 正则文法 (Regular Grammar, RG)

2种分类

➤ 右线性 (Right Linear) 文法: $A \rightarrow wB$ 或 $A \rightarrow w$ 右侧最多1个非终结符, 且在同一侧

➤ 左线性 (Left Linear) 文法: $A \rightarrow Bw$ 或 $A \rightarrow w$

➤ 左线性文法和右线性文法都称为正则文法

例 (右线性文法)

字母打头的标识符

① $S \rightarrow a \mid b \mid c \mid d$

② $S \rightarrow aT \mid bT \mid cT \mid dT$

③ $T \rightarrow a \mid b \mid c \mid d \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5$

④ $T \rightarrow aT \mid bT \mid cT \mid dT \mid 0T \mid 1T \mid 2T \mid 3T \mid 4T \mid 5T$

文法G (上下文无关文法)

① $S \rightarrow L \mid LT$

② $T \rightarrow L \mid D \mid TL \mid TD$

③ $L \rightarrow a \mid b \mid c \mid d$

④ $D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5$

3型文法 (*Type-3 Grammar*)

$$\alpha \rightarrow \beta$$

- **正则文法** (*Regular Grammar, RG*)
 - **右线性** (*Right Linear*) **文法**: $A \rightarrow wB$ 或 $A \rightarrow w$
 - **左线性** (*Left Linear*) **文法**: $A \rightarrow Bw$ 或 $A \rightarrow w$
 - 左线性文法和右线性文法都称为正则文法
- **正则语言 (3型语言)** 有限状态自动机
- 由正则文法 (3型文法) G 生成的语言 $L(G)$

正则文法能描述程序设计语言的多数单词

四种文法之间的关系

➤ 逐级限制

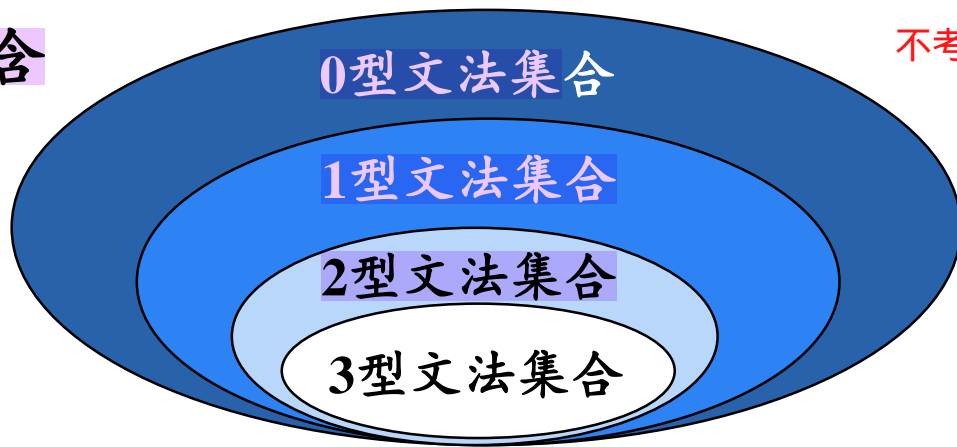
➤ 0型文法： α 中至少包含1个非终结符

➤ 1型文法 (CSG) : $|\alpha| \leq |\beta|$ 上下文有关


➤ 2型文法 (CFG) : $\alpha \in V_N$

➤ 3型文法 (RG) : $A \rightarrow wB$ 或 $A \rightarrow w$ ($A \rightarrow Bw$ 或 $A \rightarrow w$)

➤ 逐级包含



不考虑空产生式时



第二章 语言及其文法

文法的分类

哈尔滨工业大学 陈鄞





第二章 语言及其文法

CFG的分析树

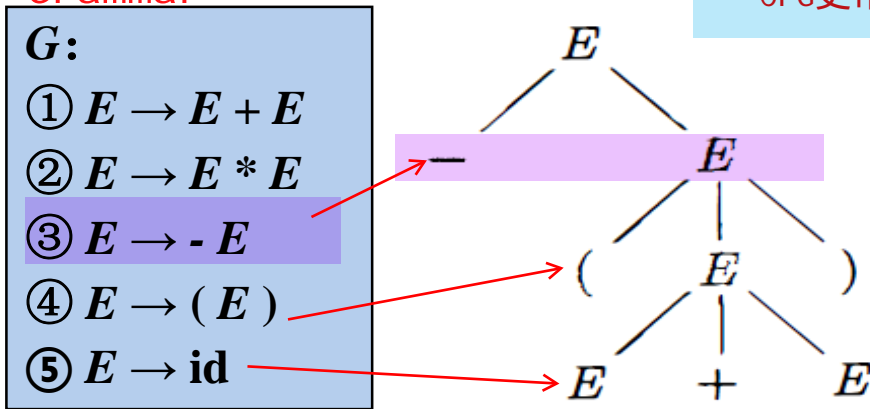
哈尔滨工业大学 陈鄞



CFG 的分析树

Context-Free Grammar

正则文法可以描述单词，但几乎描述不了句子构造
CFG更常用



- 根节点的标号为文法开始符号
- 内部结点表示对一个产生式 $A \rightarrow \beta$ 的应用，该结点的标号是此产生式左部 A 。该结点的子结点的标号从左到右构成了产生式的右部 β
- 叶结点的标号既可以是非终结符，也可以是终结符。从左到右排列叶节点得到的符号串称为是这棵树的产出(yield)或边缘(frontier)

分析树是推导的图形化表示

- 给定一个推导 $S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_n$ ，对于推导过程中得到的每一个句型 α_i ，都可以构造出一个边缘为 α_i 的分析树

推导过程: $E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(\text{id}+E) \Rightarrow -(\text{id}+\text{id})$

文法:

① $E \rightarrow E + E$

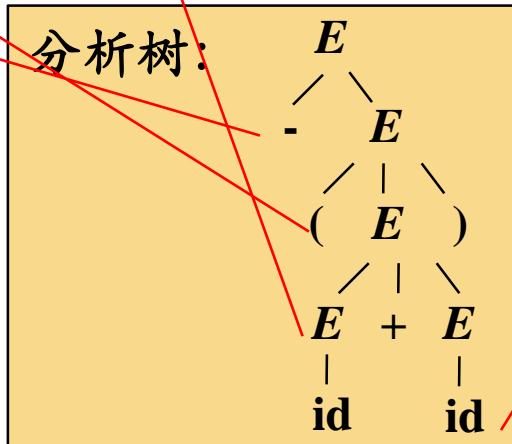
② $E \rightarrow E * E$

③ $E \rightarrow - E$

④ $E \rightarrow (E)$

⑤ $E \rightarrow \text{id}$

分析树:



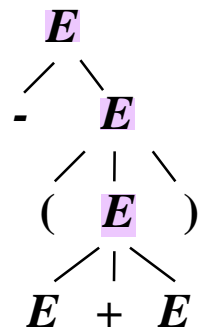
(句型的) 短语

- 给定一个句型，其分析树中的每一棵子树的边缘称为该句型的一个短语(*phrase*)
- 如果子树只有父子两代结点，那么这棵子树的边缘称为该句型的一个直接短语(*immediate phrase*)

文法:

- ① $E \rightarrow E + E$
- ② $E \rightarrow E * E$
- ③ $E \rightarrow - E$
- ④ $E \rightarrow (E)$
- ⑤ $E \rightarrow \text{id}$

分析树:



短语:

- $-(E+E)$
- $(E+E)$
- $E+E$

直接短语:

- $E+E$

直接短语一定是某产生式的右部

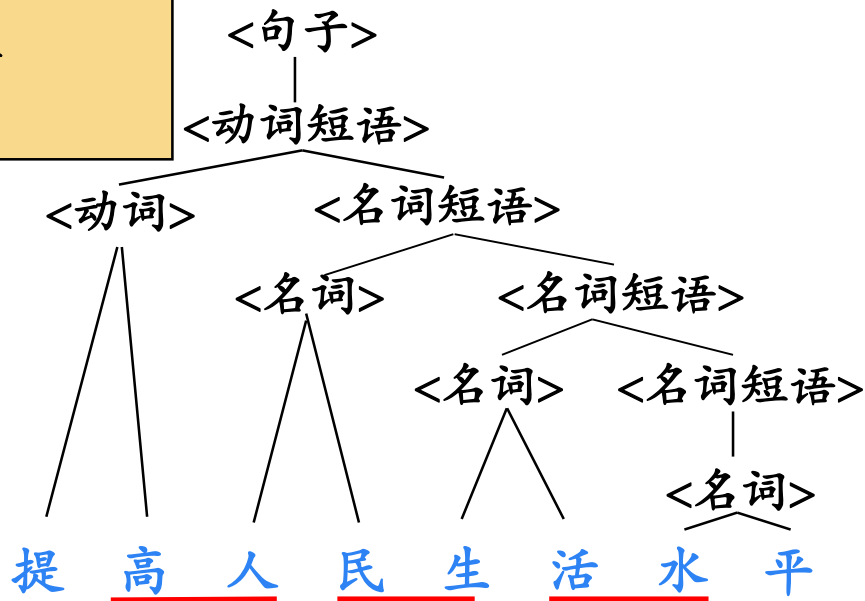
但产生式的右部不一定是给定句型的直接短语

例

文法:

- ① <句子> → <动词短语>
- ② <动词短语> → <动词> <名词短语>
- ③ <名词短语> → <名词> <名词短语> | <名词>
- ④ <动词> → 提 高
- ⑤ <名词> → 高 人 | 人 民 | 民 生 | 生 活
| 活 水 | 水 平

输入: 提高人民生活水平



二义性文法 (*Ambiguous Grammar*)

- 如果一个文法可以为某个句子生成多棵分析树，则称这个文法是二义性的

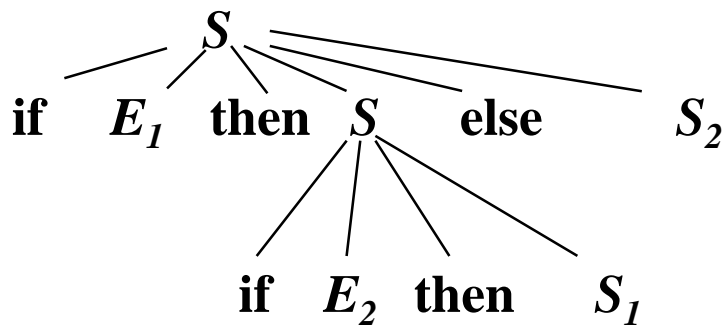
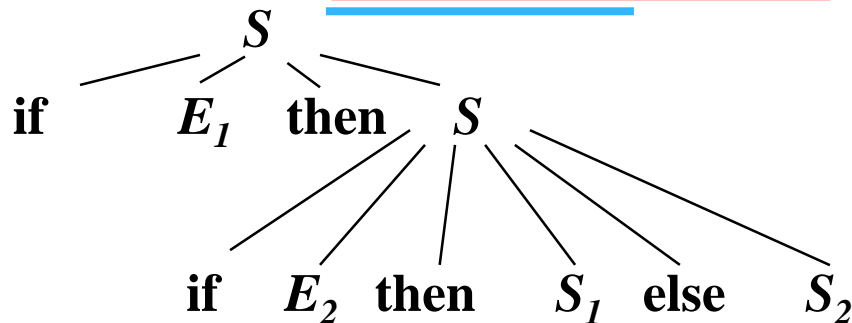
► 文法

➤ $S \rightarrow$ $\left. \begin{array}{l} \text{if } E \text{ then } S \\ \text{if } E \text{ then } S \text{ else } S \end{array} \right\}$ 条件语句

$\left| \text{other} \right. \leftarrow$ 其他语句

► 句型

➤ **if E_1 then if E_2 then S_1 else S_2**



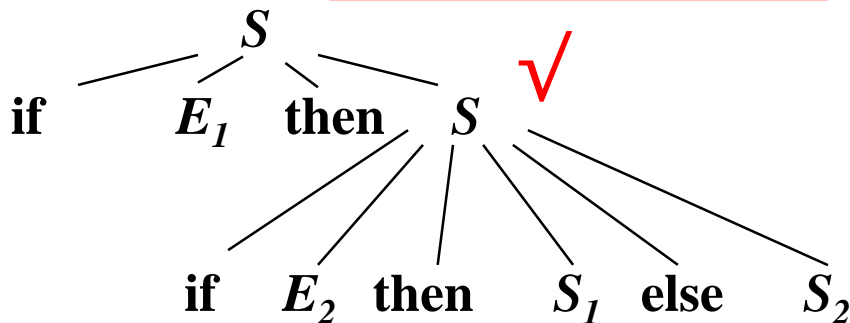
例

➤ 文法

➤ $S \rightarrow$ if E then S
| if E then S else S
| *other*

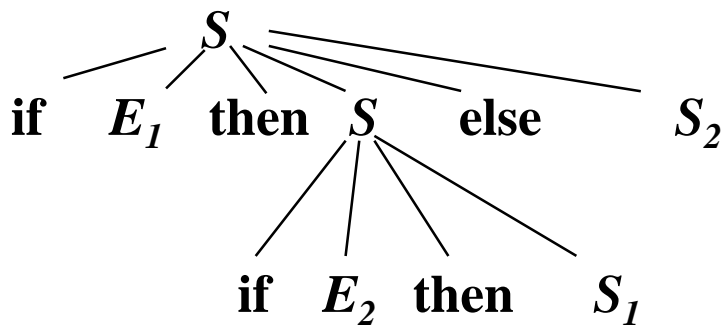
➤ 句型

➤ if E_1 then if E_2 then S_1 else S_2



解决

消歧规则：每个else和最近的尚未匹配的if匹配



二义性文法的判定

- 对于任意一个上下文无关文法，不存在一个算法，判定它是无二义性的；但能给出一组充分条件，满足这组充分条件的文法是无二义性的
- 满足，肯定无二义性
- 不满足，也未必就是有二义性的



第二章 语言及其文法

CFG的分析树

哈尔滨工业大学 陈鄞

