



# 5 上下文无关文法2025

赵伟

## 第5章 概述

5.1.1

- 介绍另一类语言，上下文无关语言（**CFL**），使用了我们熟悉的术语：句子、句型、语法、语法树等，这容易让人产生联想而不感到陌生。
- 在**Chomsky**体系中，上下文无关语言是比正则语言范围更大的一类语言。
- **CFL**有两种语言识别器，一种是上下文无关文法（**CFG**）类似于正则表达式但更强，另一种是堆栈自动机（**PDA**）类似于有穷自动机但也更强。
- 对应于教材第五章，我们从**CFG**开始，介绍与**CFG**有关的定义、推导、语法树、歧义性、**CFG**化简等概念。目的是为语法分析做必要准备。

## 5.1 文法如何定义语言

- 注意到一些非正则语言它们可能是上下文无关语言。
  - $L = \{0^n 1^n \mid n \geq 1\}$
  - 回文 =  $\{w \in \{0, 1\}^* \mid w = w^R\}$
  - $L^{\text{DUP}} = \{ww \mid w \in L, L \subseteq \{0, 1\}^*\}$
  - ...
- 如何识别上下文无关语言？
  - 上下文无关文法定义CFL，是CFL的语言识别器。

# 定义语言的思路

- ▶ 例，给定  $L = \{ 0^n 1^n \mid n \geq 1 \}$
- 对于什么样的串  $x$  属于  $L$ ，有一个自然的递归定义：
  - 基础：01 属于  $L$ ；
  - 归纳：若  $w$  属于  $L$ ，那么  $0w1$  属于  $L$ 。
- 从上述定义中产生出两条规则：
  - $01 \in L$ ；
  - 若  $w \in L$ ，那么  $0w1 \in L$ 。
- 则  $L$  是满足这两条（约束）规则的串的集合。
- 形式化这种递归定义方式得到 CFG。

# 用变元指代语言

2018

- 我们用变元 $A$ 指代语言 $L$ , 那么 $L(A)=L$ , 借此建立构造 $L$ 的若干规则。
- 例: 借用变元定义 $L = \{ 0^n 1^n \mid n \geq 1 \}$ 的两条规则是:
  - $L(A) \supseteq \{01\}$
  - $L(A) \supseteq \{0\} L(A) \{1\}$
- 在CFG中, 上述规则被表示成产生式规则形如:
  - $A \rightarrow 01$
  - $A \rightarrow 0A1$
- 左部是单个变元, 右部为一个符号串, 变元也可以出现在产生式规则右部。

# 利用产生式规则产生语言

2016

- 对于由产生式规则构成的CFG,
  - $A \rightarrow 01$
  - $A \rightarrow 0A1$
- $A$  被称为变元,  $0$ 和 $1$ 被称为终结符 (除了变元以外的符号)
- 如果我们将变元 $A$ 的语言记为 $L(A)$ , 那么,
  - 根据第一个产生式规则知道  $01 \in L(A)$ ;
  - 根据第二个产生式规则, 如果  $x \in L(A)$  那么  $0x1 \in L(A)$ 。
- ▶ 利用归纳法得到  $L(A) = \{ 0^n 1^n \mid n \geq 1 \}$

# CFG允许多个变元

2018

- 当CFG中有多个变元时，通过产生式规则，一些变元的语言借助于另一些变元的语言得到定义。
- $B \rightarrow AA$
- $A \rightarrow 01$
- $A \rightarrow 0A1$
- 第一条规则解释为：若  $x, y \in L(A)$ ，那么  $xy \in L(B)$
- 容易证明：  $L(B) = \{xy \mid x, y \in L(A)\}$ ,  $L(A) = \dots$
- CFG中有一个变元是文法初始符号，它的语言也就是这个文法所定义的语言。
- 因此，我们说CFG识别的语言是它的文法初始符号所指代的语言，这是上下文无关语言。

# CFG的形式定义

- 至此我们得到CFG的关键元素：变元、终结符、产生式规则、文法初始符号。
- 形式定义：CFG  $G = (V, T, \mathcal{P}, S)$ ，其中，
  - $G$ 为文法标识（可省略），
  - $V$ 是变元的有穷集合，
  - $T$ 是终结符的有穷集合（字母表，也习惯用 $\Sigma$ ），
  - $\mathcal{P}$ 是产生式规则的有穷集合，
  - $S$ 是文法初始符号， $S \in V$ 。
- 文法 $G$ 所定义的语言， $L(G) = L(S) \subseteq T^*$ ，是 $T$ 上的语言。



# CFG形式定义举例

2018

- 语言  $\{0^n 1^n \mid n \geq 1\}$  的CFG:
- $A \rightarrow 01$
- $A \rightarrow 0A1$
- 写成代数形式, CFG  $G = (\{A\}, \{0, 1\}, \{(A, 01), (A, 0A1)\}, A)$
- 有  $L(G) = L(A) = \{0^n 1^n \mid n \geq 1\}$
  
- 语言  $\{xy \mid x, y \in \{0^n 1^n \mid n \geq 1\}\}$  的CFG  $G'$ :
- $B \rightarrow AA$
- $A \rightarrow 01$
- $A \rightarrow 0A1$
- 代数形式: CFG  $(\{B, A\}, \{0, 1\}, \{(B, AA), (A, 01), (A, 0A1)\}, B)$
- $L(G') = L(B) = \{xy \mid x, y \in L(G)\}$

# 对产生式规则的解释

2016.12

- CFG  $G = (V, T, \mathcal{P}, S)$
- 对于  $\mathcal{P}$  中产生式规则
$$A \rightarrow \alpha,$$
其中  $A \in V$ ,  $\alpha \in (V \cup T)^*$ , 有:
  - $L(A) \supseteq L(\alpha)$
  - 如果  $\alpha \in T^*$  那么有  $L(\alpha) = \{\alpha\}$ ;
  - 如果  $\alpha = \beta N \gamma$ ,  $\beta, \gamma \in (V \cup T)^*$ ,  $N \in V$ , 则  $L(\alpha) = L(\beta)L(N)L(\gamma)$
- 再考察:
  - $B \rightarrow AA$       $L(B) \supseteq L(A)L(A)$      若  $x, y \in L(A)$  则  $xy \in L(B)$
  - $A \rightarrow 01$       $L(A) \supseteq L(01)$      若  $x \in L(01)$  则  $x \in L(A)$
  - $A \rightarrow 0A1$       $L(A) \supseteq L(0)L(A)L(1)$  若  $x \in L(A)$  则  $0x1 \in L(A)$

# CFG形式定义（续）

2016.12

- 对于任意CFG  $G = (V, T, \mathcal{P}, S)$ , 有,
- $V \cap T = \emptyset$  若  $(A, \alpha) \in \mathcal{P}$ , 那么  $A \in V$ , 并且  $\alpha \in (V \cup T)^*$
- 一般直观地写成:  $A \rightarrow \alpha$
- $A$  是产生式左部,  $\alpha$  是右部, 为了方便  $\alpha$  又称为  $A$  的候选式
- 如果  $A$  的全部候选式为  $\alpha_1 \dots \alpha_k$ ,  $k > 1$ , 那么可简写为,
  - $A \rightarrow \alpha_1 | \dots | \alpha_k$
- $S \in V$  (恰有一个变元是文法初始符号)
- $L(G) = L(S) \subseteq T^*$

符号使用习惯:

$A, B, C, \dots$  为变元

$a, b, c, \dots$  为终结符

$\dots, X, Y, Z$  为终结符或者变元

符号使用习惯:

$\dots, w, x, y, z$  只是终结符串

$\alpha, \beta, \gamma, \dots$  终结符和变元组成的串

第一个产生式的左部变元为文法初始符号

- $E \rightarrow E + E \mid E * E \mid (E) \mid d$
  - 变元集合  $\{E\}$
  - 终结符集合  $\{+, *, (, ), d\}$
  - 产生式规则集合, 4个如上所示
  - 文法初始符号  $E$
- 
- $L(E) \supseteq L(E) \{+\} L(E)$
  - $L(E) \supseteq L(E) \{*\} L(E)$
  - $L(E) \supseteq \{() L(E) \}$
  - $L(E) \supseteq \{d\}$
- 
- CFG  $(\{E\}, \{+, *, (, ), d\}, \{(E, E+E), (E, E*E), (E, \backslash(E\backslash)), (E, d)\}, E)$

- $\check{D} \rightarrow \varepsilon \mid D; \check{D}$
- $D \rightarrow T d$
- $T \rightarrow \text{int} \mid \text{float}$
- 回顾词法记号与词法单元概念
  - $d, \text{int}, \text{float}$  和  $;$  构成文法的终结符集合。
- 用到的语言?
  - $\check{D}, D, T$  构成文法的变元集合，其中  $\check{D}$  为文法初始符号。
- 代数表示为  
 CFG  $(\{\check{D}, D, T\},$   
      $\{d, \text{int}, \text{float}, ;\},$   
      $\{(\check{D}, \varepsilon), (\check{D}, D; \check{D}), (D, Td), (T, \text{int}), (T, \text{float})\},$   
      $\check{D})$
- $L(T)=?$ ;  $L(D)=?$ ;  $L(\check{D})=?$ 
  - RE:  $((\text{int } d + \text{float } d);)^*$ ; 采用推导  $\check{D} \Rightarrow \dots$

- 对于CFG  $G = (V, T, \mathcal{P}, S)$ ,
- 符号串 $\eta$ 推导 $\delta$ 写为,  $\eta \Rightarrow \delta$ , 其中 $\eta, \delta \in (V \cup T)^*$
- 基本思想是将产生式规则看作重写规则, 如果串 $\eta$ 中某个变元的一次出现被它的某个候选式替换, 得到串 $\delta$ , 就说 $\eta$ 推导 $\delta$
- 形式地, 如果 $\eta = \alpha A \beta$ ,  $\delta = \alpha \gamma \beta$ ,  $(A, \gamma) \in \mathcal{P}$ , 那么 $\eta \Rightarrow \delta$
- 例: CFG  $G = (\{A, B\}, \{0, 1\}, \{(B, AA), (A, 01), (A, 0A1)\}, B)$
- $00A11 \Rightarrow 000111$
- $00A11 \Rightarrow 000A111$
- $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 00001111$
- $B \Rightarrow AA \Rightarrow 0A1A \Rightarrow 0011A \Rightarrow 00110A1 \Rightarrow 00110011$

- 这种推导可以一直进行下去，串中任何变元的一次出现都可以用它的某个候选式来替换，直到串中只剩下终结符为止。
- $B \Rightarrow AA \Rightarrow 0A1A \Rightarrow 0011A \Rightarrow 00110A1 \Rightarrow 00110011$
- $E \Rightarrow E+E \Rightarrow E+E^*E \Rightarrow E+d^*E \Rightarrow E+d^*d \Rightarrow d+d^*d$
- $\check{D} \Rightarrow D;\check{D} \Rightarrow D;D;\check{D} \Rightarrow Td;D;\check{D} \Rightarrow Td;Td;\check{D} \Rightarrow Td;Td; \Rightarrow \text{int } d;Td; \Rightarrow \text{int } d;\text{float } d;$
- 如果从文法初始符号开始这个推导过程，终将得到该文法定义的语言的每一个成员。

# 推导与直接推导

- 对于CFG  $G = (V, T, \mathcal{P}, S)$ ,
- 如果  $\eta = \alpha A \beta$ ,  $\delta = \alpha \gamma \beta$ ,  $(A, \gamma) \in \mathcal{P}$ ,  $\alpha, \beta \in (V \cup T)^*$ , 那么  $\eta \Rightarrow \delta$ 。
- $\Rightarrow$  称为 **直接推导**。
- **推导**  $\Rightarrow^*$  是指 “0 到多个直接推导步骤”
- 基础:  $\eta \Rightarrow^* \eta$ , 对任意  $\eta \in (V \cup T)^*$
- 归纳: 若  $\eta \Rightarrow^* \alpha$ ,  $\alpha \Rightarrow \delta$ , 那么  $\eta \Rightarrow^* \delta$ , 对任意  $\eta, \alpha, \delta \in (V \cup T)^*$



# 例子：推导与直接推导

邵伟

- $S \rightarrow 01; S \rightarrow 0S1。$ 
  - $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000111。$
- 所以
  - $S \Rightarrow^* S;$
  - $S \Rightarrow^* 0S1;$
  - $S \Rightarrow^* 00S11;$
  - $S \Rightarrow^* 000111。$
- $B \Rightarrow AA \Rightarrow 0A1A \Rightarrow 0011A \Rightarrow 00110A1 \Rightarrow 00110011$
- $E \Rightarrow E+E \Rightarrow E+E^*E \Rightarrow E+d^*E \Rightarrow E+d^*d \Rightarrow d+d^*d$
- $\check{D} \Rightarrow D;\check{D} \Rightarrow D;D;\check{D} \Rightarrow Td;D;\check{D} \Rightarrow Td;Td;\check{D} \Rightarrow Td;Td; \Rightarrow \text{int } d;Td; \Rightarrow \text{int } d;\text{float } d;$

# 推导与直接推导的性质

2016.12

- 推导 $\Rightarrow^*$ 
  - 自反性
  - 传递性
- 直接推导 $\Rightarrow$ 
  - 重写性质，即，

对任意文法符号串 $\alpha$ 和 $\beta$ ， $\alpha A \beta \Rightarrow \alpha \gamma \beta$ 当且仅当 $(A, \gamma) \in \mathcal{P}$

注意，替换 $\alpha A \beta$ 中的 $A$ 为 $\gamma$ 得到 $\alpha \gamma \beta$ 与 $\alpha$ 和 $\beta$ 都无关。

# 归约与直接归约

2016

- 从产生式的右部到左部来运用，一步一步地进行，并最终将给定的终结符号串转变为文法初始符号，或者因所有子串都不是候选式而结束。
- 在过程的每一步，当前符号串中的某个子串与某个候选式相同，则可以将这个子串用那个候选式的变元替代，从而当前符号串发生改变。
- 对于CFG  $G = (V, T, \mathcal{P}, S)$ ，如果  $(A, \gamma) \in \mathcal{P}$ ， $\alpha, \beta \in (V \cup T)^*$ ，那么有  $\alpha\gamma\beta \leftarrow \alpha A \beta$ 。 $\leftarrow$ 称为直接归约，是直接推导的逆过程。
- 连续归约  $\leftarrow^*$ 是指“0到多个直接归约步骤”，定义为，
  - 基础：  $\eta \leftarrow^* \eta$ ，对任意  $\eta \in (V \cup T)^*$
  - 归纳：若  $\eta \leftarrow^* \alpha$ ， $\alpha \leftarrow \delta$ ，那么  $\eta \leftarrow^* \delta$ ，对任意  $\eta, \alpha, \delta \in (V \cup T)^*$

# 例子：归约与直接归约

邵伟

- $S \rightarrow 01 \quad S \rightarrow 0S1$ 。
  - $000111 \Leftarrow 00S11 \Leftarrow 0S1 \Leftarrow S$ 。
- 所以
  - $S \Leftarrow^* S$ ;
  - $0S1 \Leftarrow^* S$ ;
  - $00S11 \Leftarrow^* S$ ;
  - $000111 \Leftarrow^* S$ 。

- 除了自反性，推导关系的传递性还包括以下内容：
  - $\alpha \Rightarrow^* \gamma$  如果有  $\beta$  使得  $\alpha \Rightarrow \beta$  且  $\beta \Rightarrow \gamma$
  - $\alpha \Rightarrow^* \gamma$  如果有  $\beta$  使得  $\alpha \Rightarrow^* \beta$  且  $\beta \Rightarrow \gamma$
  - $\alpha \Rightarrow^* \gamma$  如果有  $\beta$  使得  $\alpha \Rightarrow^* \beta$  且  $\beta \Rightarrow^* \gamma$
- 借助于推导关系的语言定义。
- 定义：给定CFG  $G=(V, T, \mathcal{P}, S)$ ，该文法定义的**语言**  $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$ 。
- 借助于归约关系的语言定义。
- 定义：给定CFG  $G=(V, T, \mathcal{P}, S)$ ，该文法定义的**语言**  $L(G) = \{w \in T^* \mid w \Leftarrow^* S\}$ 。

# 例：文法生成符号串

2016

- 示例：文法通过推导来生成符号串  $aabbbcc$
- 替换策略：随机选取任意变元的任意一次出现替换之（推导）；固定选择最左（右）边的变元用它的某个候选式替换之（最左、最右推导）

$T \rightarrow R$   
 $T \rightarrow aTc$   
 $R \rightarrow \varepsilon$   
 $R \rightarrow RbR$

$\underline{T}$   
 $\Rightarrow a\underline{T}c$   
 $\Rightarrow aa\underline{T}cc$   
 $\Rightarrow aa\underline{R}cc$   
 $\Rightarrow aaRb\underline{R}cc$   
 $\Rightarrow aa\underline{R}bcc$   
 $\Rightarrow aaRb\underline{R}bcc$   
 $\Rightarrow aaRbR\underline{R}bcc$   
 $\Rightarrow aa\underline{R}bbRbcc$   
 $\Rightarrow aabb\underline{R}bcc$   
 $\Rightarrow aabbbcc$

$\underline{T}$   
 $\Rightarrow a\underline{T}c$   
 $\Rightarrow aa\underline{T}cc$   
 $\Rightarrow aa\underline{R}cc$   
 $\Rightarrow aa\underline{R}bRcc$   
 $\Rightarrow aa\underline{R}bRbRcc$   
 $\Rightarrow aab\underline{R}bRcc$   
 $\Rightarrow aab\underline{R}bRbRcc$   
 $\Rightarrow aabb\underline{R}bRcc$   
 $\Rightarrow aabbbb\underline{R}cc$   
 $\Rightarrow aabbbcc$

# 句型与句子

2016年

- 这里借用了自然语言的术语。
- **句型**：从文法初始符号推导出来的任意串。
- **句子**：从文法初始符号推导出来的终结字符串。
- 句子是句型的一个特例。
- 形式地：
  - 对于文法  $G=(V, T, \mathcal{P}, S)$ ,
  - $\alpha$  是一个句型, 当且仅当  $S \Rightarrow^* \alpha$
  - $w$  是一个句子, 当且仅当  $S \Rightarrow^* w$  且  $w \in T^*$
- 上页例子中每一步推导得到的都是句型, 只有最后一步是句子。
- 思考: CFG的语言是所有句子的集合。

# 最左（右）推导的符号表示

2018

- 推导允许我们替换串中的任意变元，并允许使用它的任意候选式。
- 这种重写策略会导致同一个句型有多个推导存在。
- 最左（右）推导确定化了推导过程采用的重写策略。
- 最左推导： $\Rightarrow_{lm}$ 、 $\Rightarrow_{lm}^*$
- 最右推导： $\Rightarrow_{rm}$ 、 $\Rightarrow_{rm}^*$



# 例：平衡括号文法

王浩

- 平衡括号文法：

$$S \rightarrow SS \mid (S) \mid ()$$

- $S \Rightarrow_{\text{lm}} SS \Rightarrow_{\text{lm}} (S)S \Rightarrow_{\text{lm}} (())S \Rightarrow_{\text{lm}} (())()$
- 因此,  $S \Rightarrow_{\text{lm}}^* (())()$
- $S \Rightarrow SS \Rightarrow S() \Rightarrow (S)() \Rightarrow (())()$  是一个推导, 但不是最左推导
- $S \Rightarrow_{\text{rm}} SS \Rightarrow_{\text{rm}} S() \Rightarrow_{\text{rm}} (S)() \Rightarrow_{\text{rm}} (())()$
- 因此,  $S \Rightarrow_{\text{rm}}^* (())()$
- $S \Rightarrow SS \Rightarrow SSS \Rightarrow S()S \Rightarrow ()()S \Rightarrow ()()()$  既不是最左推导也不是最右推导

# 最左、最右推导

$$E \rightarrow E + E \mid E^*E \mid (E) \mid F$$

$$F \rightarrow aF \mid bF \mid 0F \mid 1F \mid \varepsilon$$

问句子  $a^*(ab+10)$  的最左、最右推导？

$$\begin{aligned} E & \\ \Rightarrow_{lm} E^*E & \\ \Rightarrow_{lm} F^*E & \\ \Rightarrow_{lm} a^*E & \\ \Rightarrow_{lm} a^*(E) & \\ \Rightarrow_{lm} a^*(E+E) & \\ \Rightarrow_{lm} a^*(F+E) & \\ \Rightarrow_{lm} a^*(aF+E) & \\ \Rightarrow_{lm} a^*(abF+E) & \\ \Rightarrow_{lm} a^*(ab+E) & \\ \Rightarrow_{lm} a^*(ab+F) & \\ \Rightarrow_{lm} a^*(ab+1F) & \\ \Rightarrow_{lm} a^*(ab+10F) & \\ \Rightarrow_{lm} a^*(ab+10) & \end{aligned}$$

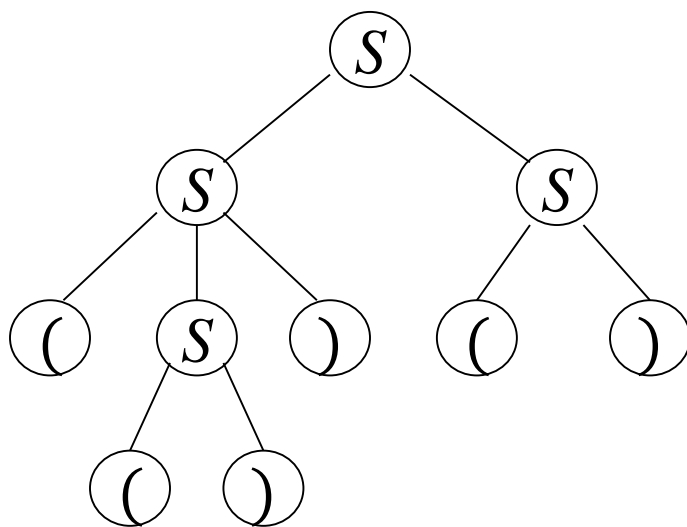
$$\begin{aligned} E & \\ \Rightarrow_{rm} E^*E & \\ \Rightarrow_{rm} E^*(E) & \\ \Rightarrow_{rm} E^*(E+E) & \\ \Rightarrow_{rm} E^*(E+F) & \\ \Rightarrow_{rm} E^*(E+1F) & \\ \Rightarrow_{rm} E^*(E+10F) & \\ \Rightarrow_{rm} E^*(E+10) & \\ \Rightarrow_{rm} E^*(F+10) & \\ \Rightarrow_{rm} E^*(aF+10) & \\ \Rightarrow_{rm} E^*(abF+0) & \\ \Rightarrow_{rm} E^*(ab+10) & \\ \Rightarrow_{rm} F^*(ab+10) & \\ \Rightarrow_{rm} aF^*(ab+10) & \\ \Rightarrow_{rm} a^*(ab+10) & \end{aligned}$$

## 5.2 语法分析树

- 可以将推导写成树形表示。树根是文法初始符号。每一次直接推导所替换的变元，就是在树中给它加上孩子节点，即对于所用候选式，依序一个符号一个节点，这些节点都作为孩子。

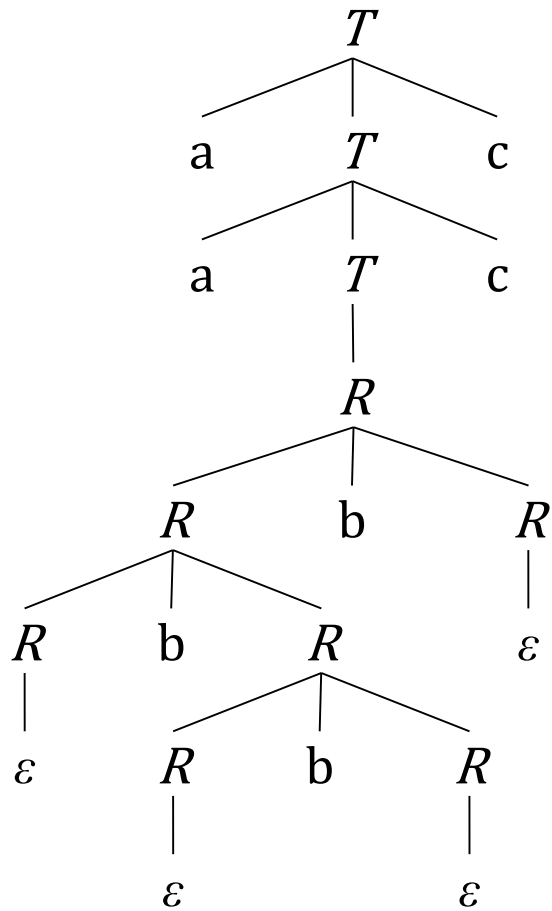
对于文法：  $S \rightarrow SS \mid (S) \mid ()$ ,

推导  $S \Rightarrow SS \Rightarrow (S)S \Rightarrow (())S \Rightarrow (())()$  的语法树



# 语法树示例

2018



$$T \rightarrow R$$

$$T \rightarrow aTc$$

$$R \rightarrow \varepsilon$$

$$R \rightarrow RbR$$

$$\begin{aligned}
 &\underline{T} \\
 \Rightarrow &a\underline{T}c \\
 \Rightarrow &aa\underline{T}cc \\
 \Rightarrow &aa\underline{R}cc \\
 \Rightarrow &aaR\underline{b}Rcc \\
 \Rightarrow &aaR\underline{b}bcc \\
 \Rightarrow &aaRb\underline{R}bcc \\
 \Rightarrow &aaRb\underline{R}bRbcc \\
 \Rightarrow &aaRb\underline{R}bRbcc \\
 \Rightarrow &aaRbb\underline{R}bcc \\
 \Rightarrow &aabb\underline{R}bcc \\
 \Rightarrow &aabbbcc
 \end{aligned}$$

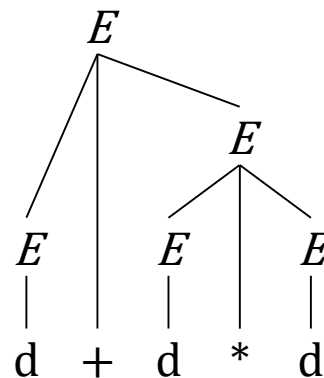
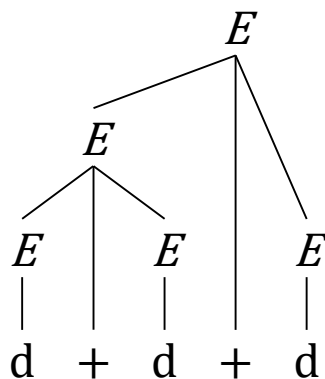
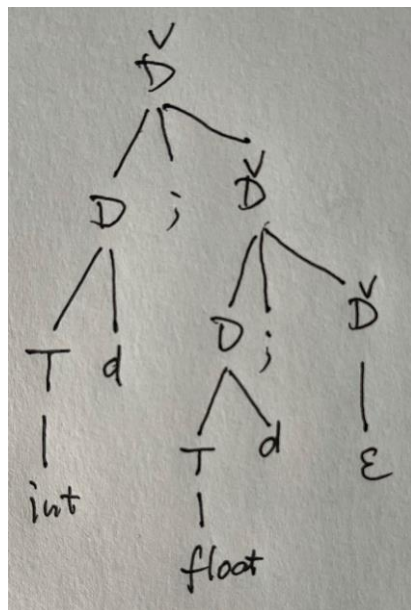
$$\begin{aligned}
 &\underline{T} \\
 \Rightarrow &a\underline{T}c \\
 \Rightarrow &aa\underline{T}cc \\
 \Rightarrow &aa\underline{R}cc \\
 \Rightarrow &aaR\underline{b}Rcc \\
 \Rightarrow &aaRb\underline{R}Rcc \\
 \Rightarrow &aab\underline{R}bRcc \\
 \Rightarrow &aabR\underline{b}RbRcc \\
 \Rightarrow &aabb\underline{R}bRcc \\
 \Rightarrow &aabbb\underline{R}cc \\
 \Rightarrow &aabbbcc
 \end{aligned}$$

- 语法树的叶子是终结符或 $\epsilon$ ，将语法树的叶子从左往右依序连接成串，该串是句子。
- 结论：CFL的每个句子都有自己的语法树，都是有序树。
  - 叶子：用终结符或 $\epsilon$ 标记
  - 内节点：用变元标记
  - 内节点的孩子对应变元的候选式（从左往右组成串）
  - 根：用文法初始符号标记
- 若允许语法树的叶子还可以是变元的话，我们就能得到句型的语法树。句型的语法树的叶子是变元、终结符或 $\epsilon$ 。

# 语法树的产物

2018

- 一个句子的语法树的产物就是该句子（每个句子都有它的语法树）。
- 一个句型的语法树的产物就是该句型。
- 语法树给它的产物增加了结构信息，将为编译所使用。
- $\check{D} \Rightarrow D; \check{D} \Rightarrow D;D; \check{D} \Rightarrow Td;D; \check{D} \Rightarrow Td;Td; \check{D} \Rightarrow Td;Td; \Rightarrow \text{int } d;Td; \Rightarrow \text{int } d;\text{float } d;$



# 与CFG判定性质有关的等价性

2016

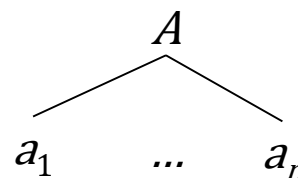
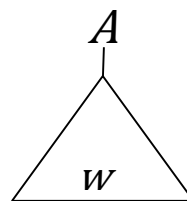
- CFG  $G=(V, T, P, S)$  的判定过程的等价性。
- 结论：对于任意  $A \in V$ ,  $w \in T^*$ , 以下彼此是等价的。
  1. (归约)  $w \leftarrow^* A$  根据归约知道  $w \in L(A)$ 。
  2. (推导)  $A \Rightarrow^* w$  根据推导知道  $w \in L(A)$ 。
  3. (最左推导)  $A \Rightarrow_{lm}^* w$  根据最左推导知道  $w \in L(A)$ 。
  4. (最右推导)  $A \Rightarrow_{rm}^* w$  根据最右推导知道  $w \in L(A)$ 。
  5. (语法树) 存在根节点为  $A$ , 产物为  $w$  的语法树。

# 从判定性1（归约）到5（语法树）

5.16.2

- **定理5.1** 设文法  $G = (V, T, P, S)$  是一个CFG，对任意终结符串  $w$ ，如果  $w$  归约为  $A$ ，则一定存在一颗以  $A$  为根、产物为  $w$  的语法分析树。
- 证明，对归约步数进行归纳。
  - 基础：若该归约步数只有一步，那么是直接归约，即存在产生式规则  $A \rightarrow w$ ， $w \Leftarrow^* A$ 。那么有一颗以  $w$  个符号标记的叶子，以  $A$  标记的根，组成树，即是  $w$  的语法树，产物为  $w$ 。

$$A \rightarrow w, w \Leftarrow^* A$$





# 从归约到语法树

2016

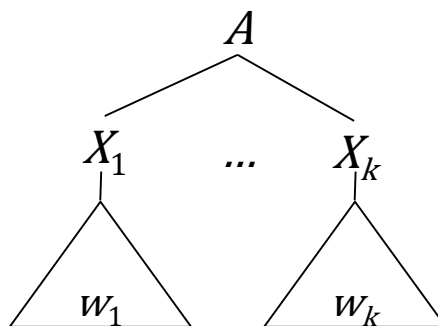
- 归纳：假设对于步数小于 $n$ 的归约 $w \leftarrow^* A$ ，都存在以 $A$ 为根以 $w$ 为产物的语法树。那么归约步数为 $n$ 时，观察最后一步归约，即归约为 $A$ ，一定使用了 $A$ 的产生式，不失一般性假设为 $A \rightarrow X_1 \dots X_k$ ，其中 $X_i \in V$ 或者 $X_i \in T$ ，对于 $i=1, \dots, k$ 。那么 $w$ 一定可以分成 $k$ 段，即 $w = w_1 \dots w_k$ ，并且 $w_i \leftarrow^* X_i$ ，对于 $i=1, \dots, k$ 。因此存在根为 $A$ ， $A$ 的孩子为 $X_1 \dots X_k$ ，产物为 $w$ 的语法树。

$$A \rightarrow X_1 \dots X_k$$

$$X_1 \dots X_k \leftarrow^* A$$

$$w = w_1 \dots w_k$$

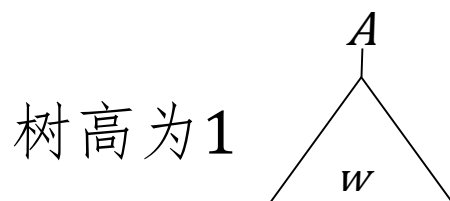
$$w_i \leftarrow^* X_i, \quad i=1, \dots, k$$



# 从判定性5（语法树）到3（推导）

5.16.2

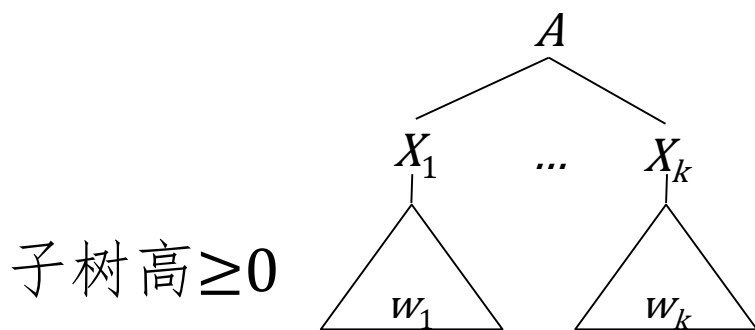
- ▶ 定理5.2。设文法  $G = (V, T, \mathcal{P}, S)$  是一个CFG，若有一个语法树它的根标记为变元  $A$  并且产物是  $w$ ， $w \in T^*$ ，那么一定存在一个  $G$  中的最左推导  $A \Rightarrow_{\text{lm}}^* w$ 。
- ▶ 证明，对语法树的高度进行归纳。
- ▶ 基础：树高为1，那么树的叶子只能是终结符，树根为  $A$ 。那么树的叶子从左往右连接起来为  $w$ 。由于这是一个语法树，所以  $A \rightarrow w$  一定是  $\mathcal{P}$  中产生式，因此有最左推导  $A \Rightarrow_{\text{lm}}^* w$ 。



有  $A \rightarrow w$ ,  
 $A \Rightarrow_{\text{lm}}^* w$

# 从语法树到推导

- 归纳：假设对于高度小于 $n$ 的语法树，其中根为 $X$ 产物为 $x$ ，都存在 $X \Rightarrow_{\text{lm}}^* x$ 。那么，当树高为 $n$ 时，我们总可以把它分解为树根及其孩子，以及每个以孩子为根的子树。不失一般性，假设树根为 $A$ ，孩子为 $X_1 \dots X_k$ ，由于这是语法树，所以必定有产生式 $A \rightarrow X_1 \dots X_k$ ，故 $A \Rightarrow_{\text{lm}}^* X_1 \dots X_k$ 。另每个以 $A$ 的孩子为根的子树的高度都小于 $n$ ，根据归纳假设，它们对应有最左推导 $X_i \Rightarrow_{\text{lm}}^* w_i$ ，对于 $i=1, \dots, k$ 。其中 $w = w_1 \dots w_k$ 。
- 因此， $A \Rightarrow_{\text{lm}} X_1 \dots X_k \Rightarrow_{\text{lm}}^* w_1 X_2 \dots X_k \Rightarrow_{\text{lm}}^* w_1 w_2 X_3 \dots X_k \Rightarrow_{\text{lm}}^* \dots \Rightarrow_{\text{lm}}^* w_1 \dots w_k$



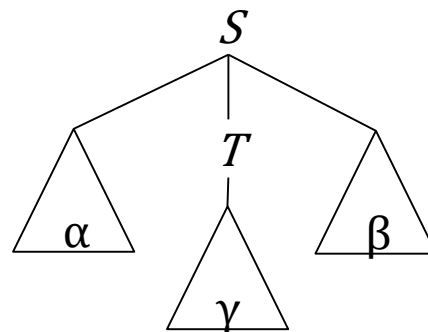
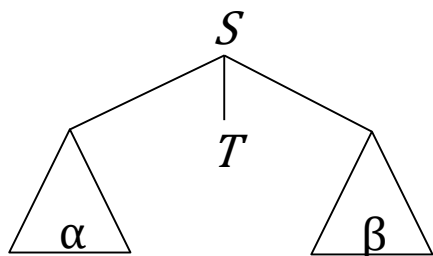
$$\begin{aligned}
 &A \rightarrow X_1 \dots X_k \\
 &A \Rightarrow^* X_1 \dots X_k \Rightarrow^* w = \\
 &\quad w_1 \dots w_k \\
 &X_i \Rightarrow^* w_i, \quad i=1, \dots, k
 \end{aligned}$$

# 推导步骤与扩展语法树步骤一一对应

2016.12

- 在过程步骤上，推导过程与扩展语法树过程一一对应。
- 从根 $S$ 扩展产物为 $w$ 的语法树的过程，初始时 $S$ 为可扩展节点，
  - 对于当前可扩展节点，用它的候选式作为孩子节点；
  - 若当前节点为未扩展过的变元则它为可扩展节点；
  - 扩展语法树直到没有可扩展节点时结束。
  - 当语法树与推导策略无关时，从给定的推导得到与语法树扩展的对应关系

$\alpha T \beta \Rightarrow \alpha \gamma \beta$  如果有产生式  $T \rightarrow \gamma$



# 归约步骤与构建语法树步骤一一对应

邵伟

- $w$  归约为  $S$ , 对应语法树根  $S$  和产物  $w$
- 符号串  $w \in T^*$  的归约是一个序列  $\alpha_1, \dots, \alpha_n, n > 1$ 。其中  $\alpha_1 = w, \alpha_n = S, \alpha_{i+1} \Rightarrow \alpha_i, 1 \leq i < n$ 。记为  $w \Leftarrow^* \alpha_n$ 。
- 语法树的生成过程:
- 从根向叶子方向扩展语法树: 对应推导。
- 从叶子往根方向构建语法树: 对应归约。

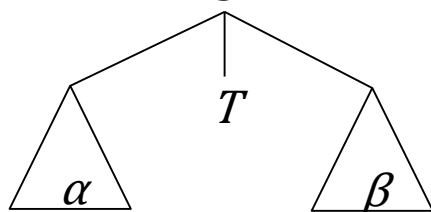
# 推导、归约与语法树对应关系

邵伟

$S$

$\vdots$

$S$

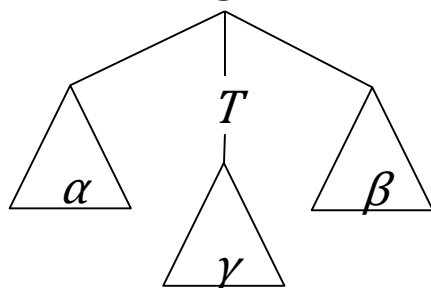


$$S \Rightarrow^* w$$

$$\begin{aligned} S &\Rightarrow^* \alpha T \beta \\ &\Rightarrow \alpha \gamma \beta \\ &\Rightarrow^* w \end{aligned}$$

自上而下  
地扩展

$S$



$\vdots$

$S$



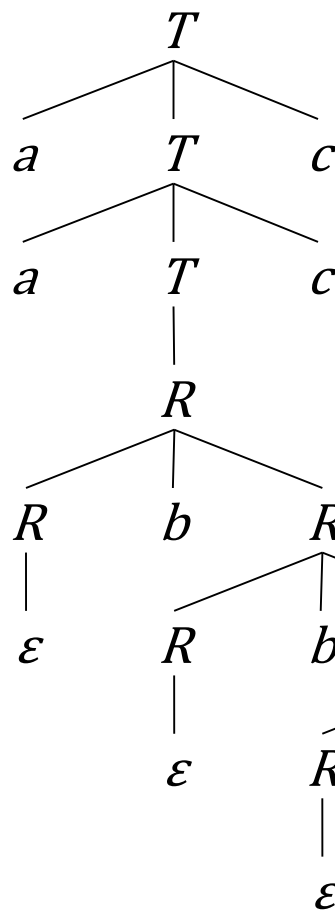
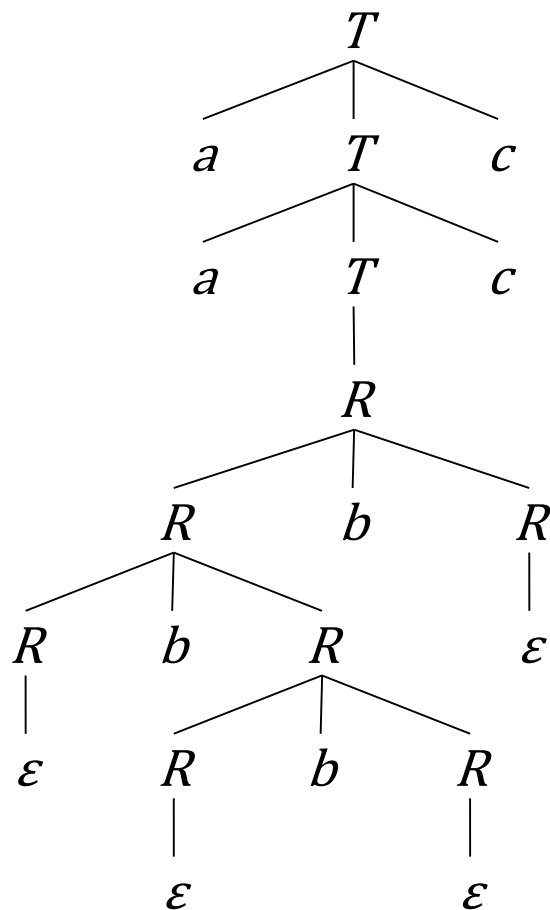
$$w \Leftarrow^* S$$

$$\begin{aligned} w &\Leftarrow^* \alpha \gamma \beta \\ &\Leftarrow \alpha T \beta \\ &\Leftarrow^* S \end{aligned}$$

自下而上  
地构建

## 5.4 歧义性文法

- 选择不同的候选式常常会推导出不同的句子，但有时不尽然。如果真出现同一个句子有多个语法树的情况，表明文法是有歧义的。

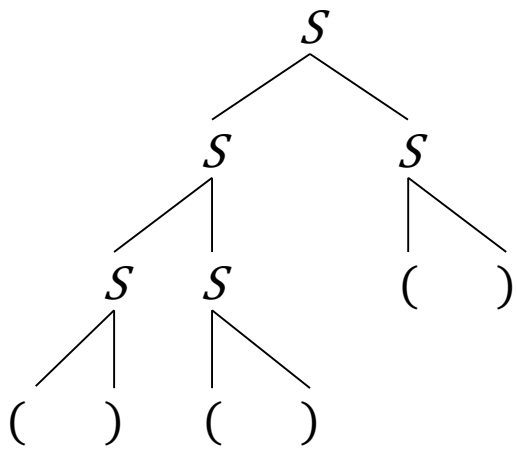


$$\begin{aligned} T &\rightarrow R \\ T &\rightarrow aTc \\ R &\rightarrow \epsilon \\ R &\rightarrow RbR \end{aligned}$$

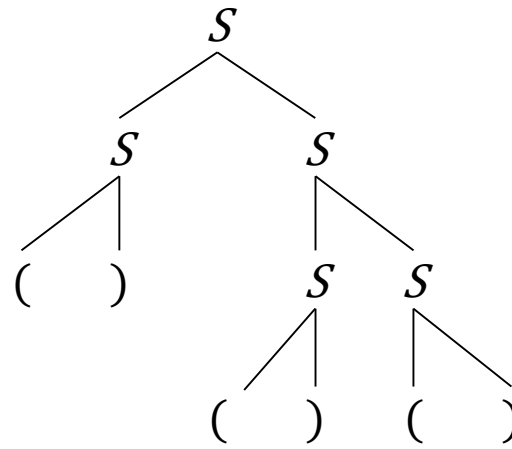
# 歧义性文法

石川

- 一个 CFG 是歧义的 若它的语言有元素至少是两个语法树的产物。
- 例：平衡括号语言  $S \rightarrow SS \mid (S) \mid ()$
- 容易找到句子它有两个语法树



(a)



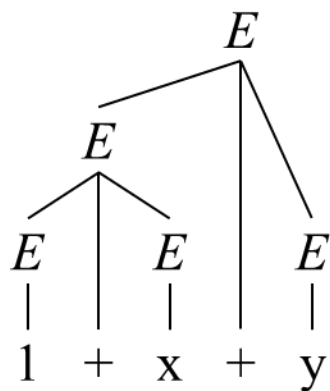
(b)



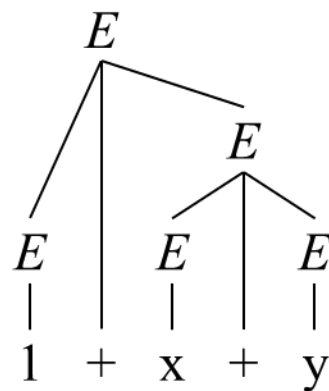
# 歧义性文法

2018

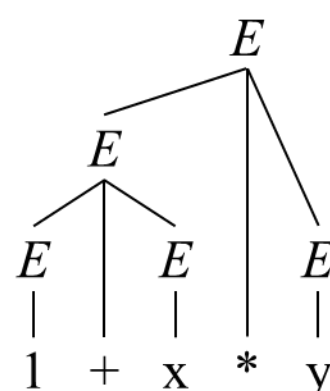
- $E \rightarrow E + E \mid E * E \mid (E) \mid d \mid i$



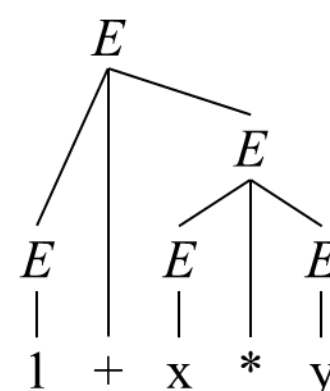
(a)



(b)



(d)

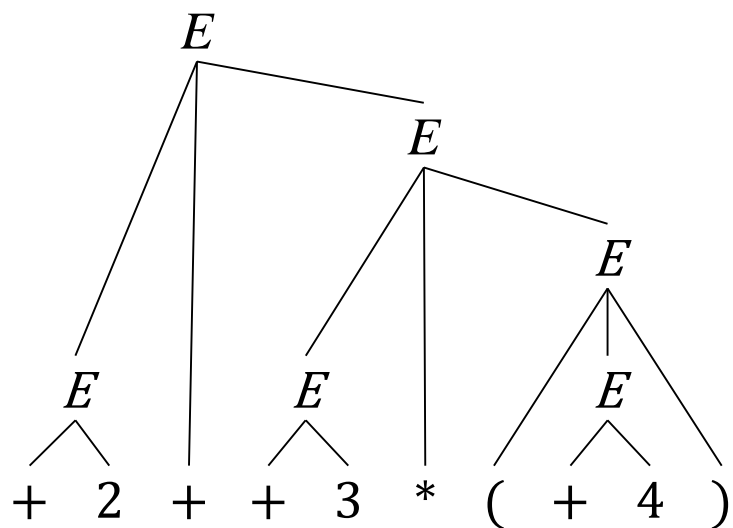


(e)

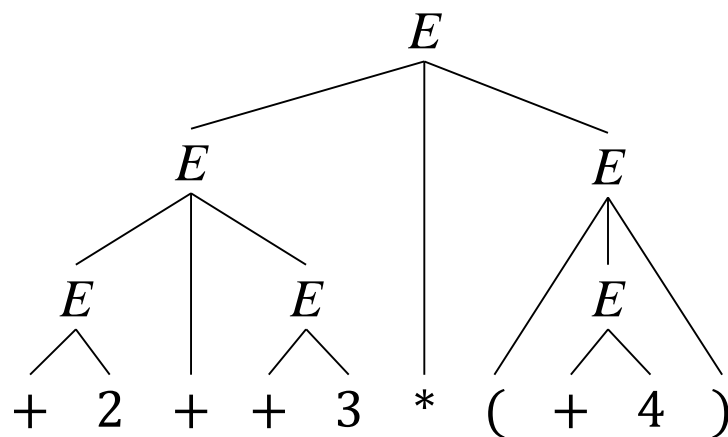
# 歧义性文法

2018

- $E \rightarrow E + E \mid E * E \mid + i \mid (E) \mid d \mid i$



(a)



(b)

# 歧义性与最左/最右推导

- **定理5.5** 对于CFG的句子 $w$ 有两个语法树当且仅当有两个最左推导。
- 如果 $w$ 有两个语法树，根据定理证明中的构造方法必定存在两个不同的最左推导。
- 相反地，根据证明中的另一部分，两个不同的最左推导产生不同的语法树。
- 最右推导类似。

# 歧义性的来源

- $2+3*4$ 
  - ①  $2+(3*4)$
  - ②  $(2+4)*4$
- 操作符的优先级
- $2+3+4$ 
  - ①  $(2+3)+4$
  - ②  $2+(3+4)$
- 中缀操作符的结合性
- if  $e_1$  then if  $e_2$  then  $s_1$  else  $s_2$ 
  - ① if  $e_1$  then {if  $e_2$  then  $s_1$  else  $s_2$ }
  - ② if  $e_1$  then {if  $e_2$  then  $s_1$ } else  $s_2$

# 歧义性是文法的性质

邵伟

- 歧义性是文法的性质不是语言的性质
- 对于平衡括号语言，有另一个CFG，这个没有歧义性。

$$B \rightarrow (RB \mid \varepsilon$$

$$R \rightarrow ) \mid (RR$$

- $B$  文法初始符号，从它推导出平衡括号串。
- $R$  生成右括号比左括号多一个的串。

# 例：把歧义文法转换为无歧义文法

2016

- 歧义文法：
  - $B \rightarrow BB \mid (B) \mid ()$
- 无歧义文法：
  - $B \rightarrow (RB \mid \varepsilon$
  - $R \rightarrow ) \mid (RR$
- 对于给定的平衡的括号的串通过从左到右扫描该串，构造出唯一最左推导。
- 如果我们需要扩展  $B$ ：
  - 若下一个符号是 “(” 使用  $B \rightarrow (RB$  扩展
  - 若输入串为空使用  $\varepsilon$  扩展
- 如果我们需要扩展  $R$ ：
  - 若下一个符号是 “)” 使用  $R \rightarrow )$  扩展
  - 若下一个符号是 “(” 使用  $R \rightarrow (RR$  扩展



剩余输入串：

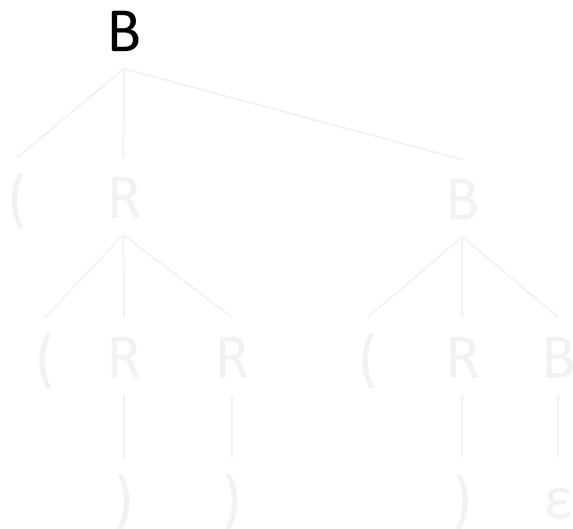
**(( )) (**



当前  
输入  
符号

最左推导步骤：

**B**



$B \rightarrow (RB \mid \varepsilon \quad R \rightarrow ) \mid (RR$



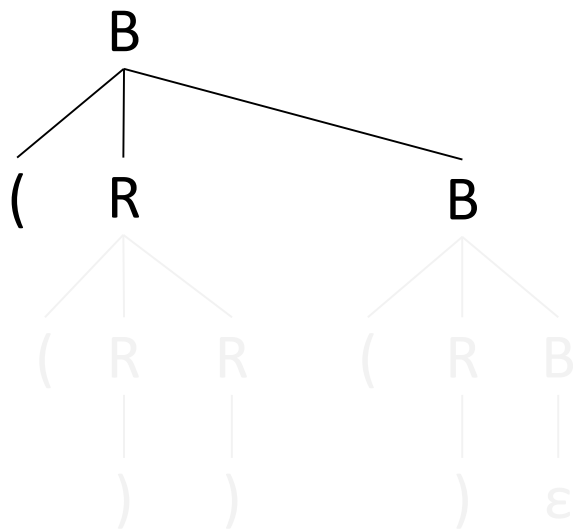
# The Parsing Process

剩余输入串：

**(( )) (**



当前  
输入  
符号



$B \rightarrow (RB \mid \varepsilon$        $R \rightarrow ) \mid (RR$

最左推导步骤:

B

**(RB**





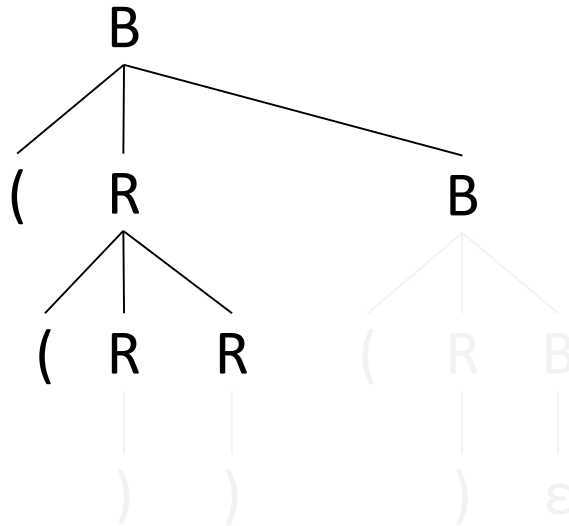
# The Parsing Process

剩余输入串：

(( )) ( )



当前  
输入  
符号



$B \rightarrow (RB \mid \varepsilon$        $R \rightarrow ) \mid (RR$

最左推导步骤：

B

(RB

( (RRB



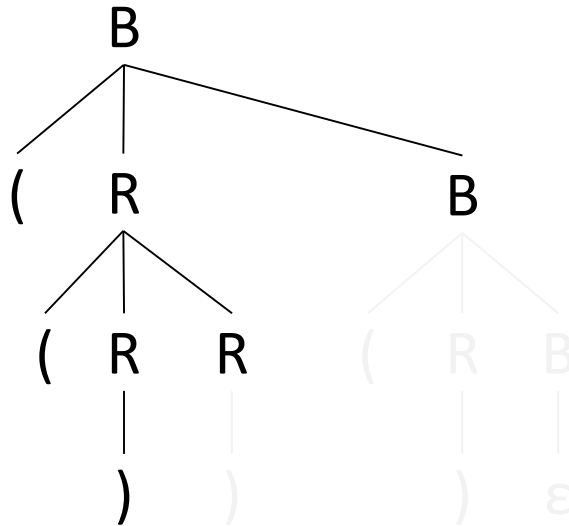
# The Parsing Process

剩余输入串：

$(( )) ( )$



当前  
输入  
符号



最左推导步骤：

$B$

$(RB$

$( (RRB$

$( ( ) RB$

$B \rightarrow (RB \mid \varepsilon \quad R \rightarrow ) \mid (RR$



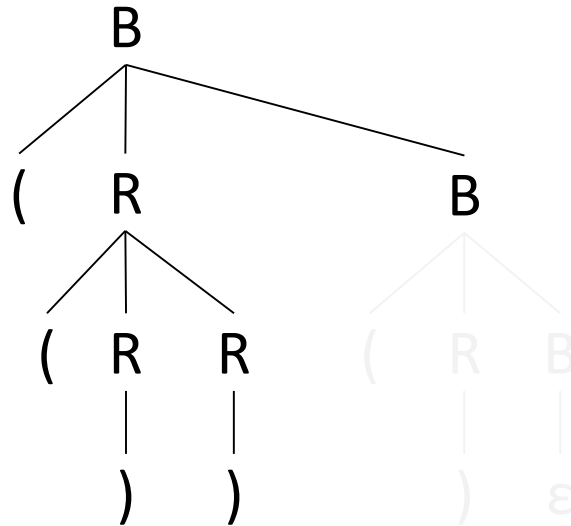
# The Parsing Process

剩余输入串：

**(( )) (**



当前  
输入  
符号



最左推导步骤：

**B**

**(RB**

**( (RRE**

**( ( ) RE**

**( ( ) ) B**

$B \rightarrow (RB \mid \varepsilon \quad R \rightarrow ) \mid (RR$

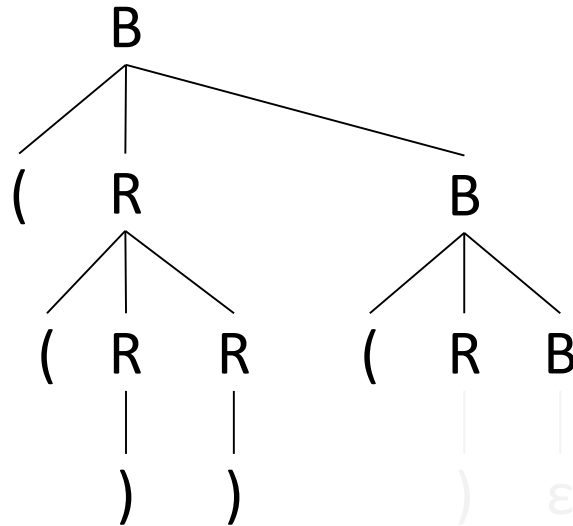


# The Parsing Process

剩余输入串：

(( )) ( )  
^

当前  
输入  
符号



最左推导步骤：

B

(RB

( (RRB

( ( ) RB

( ( ) ) B

( ( ) ) (RB

$B \rightarrow (RB \mid \varepsilon$        $R \rightarrow ) \mid (RR$

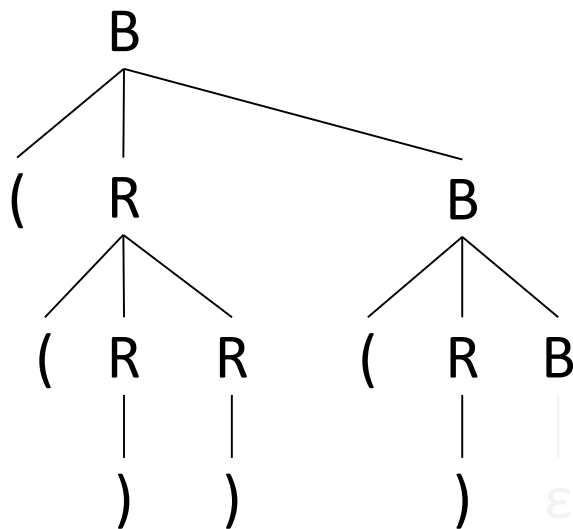


# The Parsing Process

剩余输入串：

^

当前  
输入  
符号



最左推导步骤：

B

(RB

( (RRB

( ( ) RB

( ( ) ) B

( ( ) ) (RB

( ( ) ) ( ) B

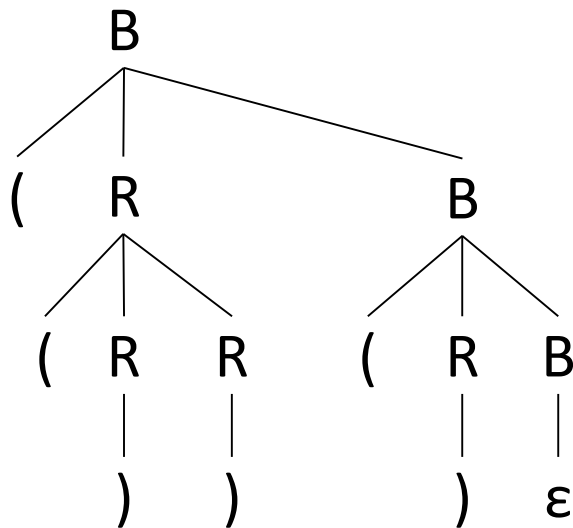
$B \rightarrow (RB \mid \varepsilon$        $R \rightarrow ) \mid (RR$



## 剩余输入串：



当前  
输入  
符号



## 最左推导步骤:

B

(RB

( ( RRB

( ) RB

( ( ) ) B

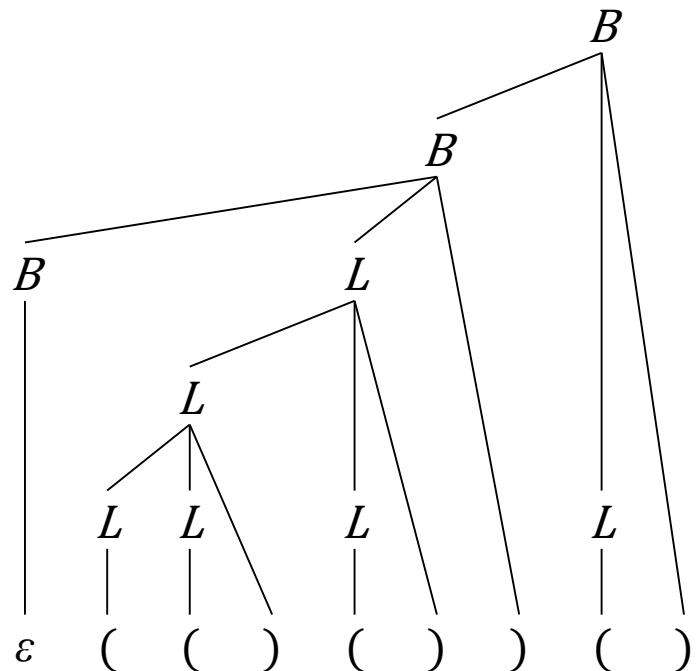
( ) (RB

( ) ( ) B

( ) ( ) ( ) ( )

$$B \rightarrow (RB \mid_{\varepsilon} R \rightarrow) \mid (RR$$

王明

$$R \rightarrow ) \mid (RR$$
$$L \rightarrow ( \mid LL )$$


# 固有的歧义性

- 消除歧义性：通过分析产生歧义性的原因，针对性地修剪文法得到无歧义性文法。
- 分析定义CFL的多个文法：
  - ① 都是无歧义的
  - ② 有一些是无歧义的，其余都是歧义的
  - ③ 都是歧义的
- 选择无歧义的文法来进行CFL的语法分析，但不幸的是，有一些CFL是固有的歧义的，即对它不存在无歧义文法。
- 歧义性文法是语法分析需要考虑的情形。



# Example:固有的歧义性

石川

- 语言  $\{0^i 1^j 2^k \mid i = j \text{ or } j = k\}$  是固有的歧义的。
- 直觉地，至少一些串形如  $0^n 1^n 2^n$  必定是由两个不同语法树产生的，一个基于对0和1的个数进行检测，而另一个基于对1和2的个数进行检测。
- -是左结合+是右结合的表达式  $2-3+4$ （同一优先级运算符的结合性不一致）

# 中缀运算符 $\oplus$ 的结合性

2018

- $\oplus$ 是左结合的，如果 $a \oplus b \oplus c = (a \oplus b) \oplus c$
- $\oplus$ 是右结合的，如果 $a \oplus b \oplus c = a \oplus (b \oplus c)$
- $\oplus$ 是无结合性的，如果 $a \oplus b \oplus c$ 则为非法
- 数学上，-和/是左结合的，+和\*是结合的，左右都可以
- 计算上，因为满足结合律的运算符它的左、右结合性表现在精度、是否溢出等方面会有不同，所以左、右结合性只能二者选一。
- 一些程序设计语言如C选则左结合的+和\*，这样就与数学上的-和/的左结合性保持一致，否则就会顶牛，对消除歧义性没有帮助。
- 但是C中的赋值操作是右结合的， $x=y=z$ 被解释为 $x=(y=z)$
- 另一些程序设计语言如SML有一些右结合操作
- PASCAL语言的<和>都是无结合性的。即 $a < b < c$ 出错

# 消除文法歧义性 ( $\bar{a}$ 修剪操作符)

王

CFG的等价性质:  $\bar{a} \cdot G = G$

- $E \rightarrow E \oplus E \mid d$
- $\oplus$ 是左结合的,
  - $E \rightarrow E \oplus E' \mid E'$
  - $E' \rightarrow d$
- $\oplus$ 是右结合的,
  - $E \rightarrow E' \oplus E \mid E'$
  - $E' \rightarrow d$
- 扩展到多个运算符时保持结合性一致
  - $E \rightarrow E + E' \mid E - E' \mid E'$
  - $E' \rightarrow d$

# 解决不同优先级操作符带来的歧义性

2016

- ▶ 每个优先级对应一个非终结符
- ▶ 如果一个表达式使用了某优先级的运算符，那么它的子表达式就不能使用较低优先级的运算符（除非出现在括号里）。
- ▶ 因此一个优先级的变元的候选式中不能出现那些对应于比它优先级低的变元（除非出现在括号里）。
- $E \rightarrow E + T \mid T$
- $T \rightarrow T * F \mid F$
- $F \rightarrow (E) \mid d \mid i$

# 一种有歧义性的文法

王作龙

$$S \rightarrow AB \mid CD$$
$$A \rightarrow 0A1 \mid 01$$
$$B \rightarrow 2B \mid 2$$
$$C \rightarrow 0C \mid 0$$
$$D \rightarrow 1D2 \mid 12$$

$A$  产生相等个数0和1

$B$  产生任意个数的 2

$C$  产生任意数目的 0

$D$  产生相等个数1和2

每个有相等数目的0, 1, 和2的串都有两个最左推导.如:

$$S \Rightarrow AB \Rightarrow 01B \Rightarrow 012$$
$$S \Rightarrow CD \Rightarrow 0D \Rightarrow 012$$

# 小结：文法、语言、程序

2016.12

$P \rightarrow \check{D} \check{S}$

$\check{D} \rightarrow \varepsilon \mid \check{D} D;$

$D \rightarrow T d \mid T d(\check{A}) \{\check{D} \check{S}\}$

$T \rightarrow \text{int} \mid \text{void}$

$\check{A} \rightarrow \varepsilon \mid \check{A} A;$

$A \rightarrow T d \mid T d()$

$\check{S} \rightarrow S \mid \check{S}; S$

$S \rightarrow \text{if } (B) S \mid \text{if } (B) S \text{ else } S \mid \text{while } (B) S \mid$   
 $\text{return } E \mid \text{print } d \mid \{\check{S}\} \mid d(\check{R})$

$B \rightarrow B \wedge B \mid B \vee B \mid E r E \mid E$

$E \rightarrow d = E \mid i \mid d \mid d(\check{R}) \mid E o E \mid (E)$

$\check{R} \rightarrow \varepsilon \mid R$

$R \rightarrow E \mid d()$

词法单元SCO, ID, LPA, RPA, LBR,  
RBR, INT, VOID, IF, ELSE, WHILE,  
RETURN, AND, OR, ROP={<, <=, >},  
ASG, NUM, AOP={+, \*, /}

$\mathcal{L} = \{\text{SCO, ID, LBK, RBK, LPA, RPA, LBR, RBR, INT, VOID, NUM, CMA, IF, ELSE, WHILE, RETURN, PRINT, ADD, MUL, DIV, AND, OR, ROP}\}$

```
int raw(int x) {
```

```
    y = x+5;
```

```
    return y};
```

```
void foo(int y) {
```

```
    int z;
```

```
    void bar(int x; int soo();) {
```

```
        if(x>3) bar(x/3, soo(),)
```

```
        else z = soo(x);
```

```
        print z};
```

```
    bar(y, raw(),);
```

```
foo(6,)
```

## 5.5 CFG 化简

- 对文法进行哪方面的化简？

5.5.1 去除无用符号

5.5.2 去除 $\varepsilon$ 产生式

5.5.3 去除单位产生式

### CFG的几种等价性质

对于CFG  $G$ 有

- $a \cdot G$ 的文法符号都是可达的且 $L(a \cdot G) = L(G)$
- $g \cdot G$ 的变元都是产出的且 $L(g \cdot G) = L(G)$
- $\bar{e} \cdot G$ 不含 $\varepsilon$ 产生式且 $L(\bar{e} \cdot G) = L(G)$
- $\bar{u} \cdot G$ 不含单位产生式且 $L(\bar{u} \cdot G) = L(G)$

## 5.5.1 有用与无用的文法符号

5.5.1

- 出现在一个句子的成功推导中的文法符号都是有用符号
  - 每个句子的始于初始符号的推导都是它的一个成功推导
- 对CFG  $(V, T, P, S)$ , 若有  $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$ ,  $\alpha, \beta \in (V \cup T)^*$ ,  $w \in T^*$  那么  $X$  是有用的文法符号 (有用符号)
- 分为两个方面来看:
  - ①  $S \Rightarrow^* \alpha X \beta$        $X$  是可达符号;  $X$  是**可达的**
  - ②  $X \Rightarrow^* w$        $X$  是有产出的符号;  $X$  是**产出的**

结论:

1. 由文法初始符号推导出的任意串中的文法符号都是可达的。
2. 能推导出终结字符串的文法符号都是产出的。
3. 既是可达的又是产出的文法符号是有用的。
4. 一个文法的文法符号不是有用的便是无用的。



# 例：有用、无用的文法符号

- 终结符总是产出的，但不一定是可达的；
- 对变元而言，各种情形都有可能。

①  $S \Rightarrow^* \alpha X \beta$        $X$ 是可达的

②  $X \Rightarrow^* w$        $X$ 是产出的

- 例：  $S \rightarrow AB$     $A \rightarrow aA \mid a$     $B \rightarrow AB$
- 可达符号：  $S, A, B, a$
- 不可达符号：无
- 有产出的符号：  $A, a$
- 无产出的符号：  $S, B$
- 因为文法初始符号  $S$  是无产出的，所以该语言是空（不考虑）

# 去除无用符号

2016

- 对于CFG  $G = (V, T, P, S)$ ,  $L(G) \neq \emptyset$
- 从 $G$ 中去除无产出变元得到 $gG = (V_g, T, P_g, S)$ , 其中:
  - $V_g$ 是 $V$ 中去掉无产出成员所得;
  - $P_g$ 是 $P$ 中去掉含有无产出变元的成员 (即产生式)。
- 从 $G$ 中去除不可达符号得到CFG  $aG = (V_a, T_a, P_a, S)$ , 其中:
  - $T_a$ 是从 $T$ 中去掉不可达成员;
  - $V_a$ 是 $V$ 中去掉不可达成员;
  - $P_a$ 是 $P$ 去掉含有不可达符号的成员。
- 例 $G: S \rightarrow AB|a \quad A \rightarrow b$ 
  - $V_g = ?$  ;  $P_g = ?$
- 则 $gG: S \rightarrow a \quad A \rightarrow b$
- 那么 $agG: S \rightarrow a$
- 对于 $G': S \rightarrow AB|B \quad A \rightarrow b$  会怎样?  $V_g$ 中不能没有 $S$ !
- 注意,  $L(G') = \emptyset$

## 例：去除无用符号

思考

- 从文法中先去除无产出变元，进而去除不可达符号。
- 例 $G$ :  $S \rightarrow AB|C$   $A \rightarrow aA|a$   $B \rightarrow bB$   $C \rightarrow c$ 
  - $V = \{S, A, B, C\}$ ;  $T = \{a, b, c\}$ ;  $P$ ;  $S$
- $gG$ :  $S \rightarrow C$   $A \rightarrow aA|a$   $C \rightarrow c$ 
  - $V_g = \{S, A, C\}$ ;  $T = \{a, c\}$ ;  $P_g$ ;  $S$
- $agG$ :  $S \rightarrow C$   $C \rightarrow c$ 
  - $V_{ga} = \{S, C\}$ ;  $T_a = \{c\}$ ;  $P_a$ ;  $S$
- 思考：从 $G$ 中去除不可达符号，进而去除无产出变元，如何？

# 定理证明

2018

- 定理5.6 对于CFG  $G = (V, T, P, S)$ ,  $L(G) \neq \emptyset$ , 令  $g \cdot G = (V_g, T, P_g, S)$ ,  $ag \cdot G = (V_{ga}, T_a, P_{ga}, S)$ , 则  $ag \cdot G$  中没有无用符号, 且  $L(ag \cdot G) = L(G)$ 。

证明: (1)  $ag \cdot G$  中没有无用符号。

- 对于每一个  $X \in T \cup V_g$ , 则  $X$  也属于  $T \cup V$ , 如果  $X \Rightarrow_G^* w, w \in T^*$ , 那么这个推导中所有符号都是产出的, 所以  $X \Rightarrow_{gG}^* w$ 。
- 对于每一个  $X \in T \cup V_g$ , 如果  $S \Rightarrow_{gG}^* \alpha X \beta, \alpha, \beta \in (V_g \cup T)^*$ , 那么这个推导中所有符号都是可达的, 所以有  $S \Rightarrow_{agG}^* \alpha X \beta$ , 并且  $X \in V_{ga} \cup T_{ga}, \alpha, \beta \in (V_{ga} \cup T_{ga})^*$ 。

证明: (2) 证明等价性质  $L(ag \cdot G) = L(G)$ 。

- 由于  $T_a \subseteq T, V_{ga} \subseteq V$ , 并且  $P_{ga} \subseteq P$ , 所以  $L(ag \cdot G) \subseteq L(G)$ 。
- 又由于  $S \Rightarrow_G^* w, w \in T^*$  中所有出现的符号都是有用的, 所以有  $S \Rightarrow_{agG}^* w, w \in T_a^*$ 。因此  $L(G) \subseteq L(ag \cdot G)$ 。

# 发现有产出变元的算法

2016

- 若  $X \Rightarrow^* w$ ，则  $X$  是产出的。
- 根据CFG的产生式判断变元  $A$  是否是产出的：
  - 基础：若存在产生式  $A \rightarrow w$ ，其中  $w$  不含变元，那么  $A$  是产出的。
  - 归纳：若存在产生式  $A \rightarrow \alpha$ ，其中  $\alpha$  仅由终结符和有产出变元组成，那么  $A$  是产出的。
- 证明：根据推导  $X \Rightarrow^* w$  的步数归纳，0步成立；假定  $n-1$  步成立；那么  $n$  步时， $X \Rightarrow \alpha \Rightarrow^* w$  中第一步对应产生式  $A \rightarrow \alpha$ ，且  $\alpha$  中每个符号都推导出  $w$  的一个子串，并且推导步数小于  $n$ ，那么这些符号都是产出的，因此算法能发现  $A$  是产出的。

## 例：发现有产出变元

思考

- $S \rightarrow AB|C$     $A \rightarrow aA|a$     $B \rightarrow bB$     $C \rightarrow c$
- 基础：A 和 C 可以被确定有产出，因为  $A \rightarrow a$  和  $C \rightarrow c$ 。
- 归纳：S 可被确定有产出，因为  $S \rightarrow C$ 。
- 没有别的可以被发现的了。

# 发现可达符号的算法

2016.12

- 若  $S \Rightarrow^* \alpha X \beta$ , 则  $X$  是可达的。
- 根据CFG的产生式判断某变元  $A$  是否是可达的:
  - 基础: 文法初始符号  $S$  可达。
  - 归纳: 如果  $A$  可达, 并且存在产生式  $A \rightarrow \alpha$ , 那么所有  $\alpha$  中的符号均可达。
- 证明: 对从  $S$  开始推导的步数归纳。0步成立; 设小于  $n$  步成立; 那么  $n$  步时如果有  $S \Rightarrow^* \alpha A \beta \Rightarrow \alpha \gamma \beta$ , 其中  $A \rightarrow \gamma$ 。除最后一步直接推导, 前面的步数小于  $n$ , 所以  $A$  是可以被发现的, 那么  $\gamma$  中的符号也能被发现。

## 例：发现可达符号

思考

- 例：  $S \rightarrow AB$   $A \rightarrow C|\epsilon$   $C \rightarrow c$   $B \rightarrow bB$   $D \rightarrow \epsilon$
  - 基础：  $S$  被发现，因为文法初始符号是可达的。
  - 归纳：  $A$  和  $B$  被发现，因为有产生式  $S \rightarrow AB$ ，即有  $S \Rightarrow^* AB$ ；
  - 接着  $C$  被发现，因为  $A \rightarrow C$ ；
  - $b$  被发现因为  $B \rightarrow bB$ ；
  - 接着  $c$  被发现因为  $C \rightarrow c$ ；
  - 再没有别的可发现的了。
- 
- 思考：  $\epsilon$  是不是可达的？



## 5.5.2 去除 $\epsilon$ 产生式

- $\epsilon$ 产生式 $A \rightarrow \epsilon$ 是有用的但不是必须的。如果语言中不考虑空串的话我们定义它时就不需要 $\epsilon$ 产生式了，或者含有 $\epsilon$ 产生式的CFG可以等价转化为无 $\epsilon$ 产生式的CFG。
- 发现可空变元。
- 可空变元在候选式中不出现的话，事实上就反映了它推导 $\epsilon$ 这种情况，如此就已经起到了 $\epsilon$ 产生式的作用。
- 去除 $\epsilon$ 产生式。
- 定义：CFG  $(V, T, P, S)$ ， $A \in V$ 是**可空变元**，当且仅当 $A \Rightarrow^* \epsilon$ 。
- 例： $S \rightarrow AcB$     $A \rightarrow aA | \epsilon$     $B \rightarrow bB | A$
- 可空变元：A； B

# 发现可空变元的算法

王

- CFG  $G=(V, T, P, S)$  的  $A \in V$  是可空变元，当且仅当  $A \Rightarrow^* \varepsilon$ 。
- 基础：若  $A \rightarrow \varepsilon$  是  $G$  的产生式，那么  $A$  可空。
- 归纳：若  $G$  有产生式  $A \rightarrow C_1 \dots C_k$ ，其中对于  $i=1, 2, \dots, k$ ，有  $C_i \in V$  且是可空的，那么  $A$  是可空的。
- 例：  $S \rightarrow AB$     $A \rightarrow aA | \varepsilon$     $B \rightarrow bB | A$
- 基础：  $A$  可空因为  $A \rightarrow \varepsilon$ 。
- 归纳：  $B$  可空因为  $B \rightarrow A$ ；再者，  $S$  可空因为  $S \rightarrow AB$ 。

# 定理证明

邵永亮

定理5.7 在任何文法 $G$ 中，上面的算法生成的恰好是全部的可空符号。

- 变元 $A$  被算法找出当且仅当  $A \Rightarrow^* \varepsilon$
- 证明：根据推导 $A \Rightarrow^* \varepsilon$ 的步数归纳，第1步成立即算法发现 $A$ 可空；假定小于 $n$ 步时成立；那么 $n$ 步时，不失一般性，有 $A \Rightarrow C_1 C_2 \cdots C_k \Rightarrow^* \varepsilon$ ，其中对于 $i=1, 2, \dots, k$ ， $C_i$ 小于 $n$ 步推导出 $\varepsilon$ ，根据归纳假设 $C_i$ 可空，因此 $A$ 可空。
- 若算法发现 $A$ 可空，那么 $A \Rightarrow^* \varepsilon$ ，证明对应算法中归纳过程（留作练习）。

# 去除 $\varepsilon$ 产生式

ε产生式

- **变形产生式**：若产生式 $A \rightarrow \alpha$ 的候选式 $\alpha$ 中有 $m$ 个符号是可空变元， $m \geq 0$ ，那么这 $m$ 个变元的每一个出现和不出现，就形成了 $\alpha$ 的 $2^m$ 个变形版本。其中 $\alpha$ 又称为自己的原形版本。
- $m=0$ 时？只有原形版本
- $m=|\alpha|$ 时？有个 $A \rightarrow \varepsilon$ 版本

**性质**：如果使用变形候选式推导出 $w$ ，那么，使用原形候选式亦推导出 $w$ 。

- 举例：若 $A$ 可空，那么产生式 $B \rightarrow CAD$ 的变形版本为：
- $B \rightarrow CAD ; B \rightarrow CD$
- 举例：若 $A$ 和 $B$ 可空，那么产生式 $S \rightarrow AaBDAbC$ 的变形版本：
- $S \rightarrow aDbbC | aDbAbC | aBDbbC | aBDbAbC | AaDbbC | AaDbAbC | AaBDbbC | AaBDbAbC$



# 算法：去除 $\epsilon$ 产生式

输入：  $G = (V, T, P, S)$

输出：  $\bar{\epsilon}G = (V, T, P_{\bar{\epsilon}}, S)$

- $P_{\bar{\epsilon}}$ 初始化为空集合；
- 找出 $G$ 的所有可空变元；
- 对 $P$ 中每个非 $\epsilon$ 产生式，将它的变形全部加入 $P_{\bar{\epsilon}}$ ；
- $P_{\bar{\epsilon}}$ 中去掉以 $\epsilon$ 为候选式的所有产生式。

## 例：去除 $\epsilon$ 产生式

ε产生式

$G: S \rightarrow ABC \mid A \rightarrow aA \mid \epsilon \mid B \rightarrow bB \mid \epsilon \mid C \rightarrow \epsilon$

- 其中 $A$ 、 $B$ 、 $C$ 和 $S$ 都是可空的。

$\bar{e}G:$

$$S \rightarrow ABC \mid AB \mid AC \mid A \mid BC \mid B \mid C$$
$$A \rightarrow aA \mid a$$
$$B \rightarrow bB \mid b$$

## 例：去除 $\epsilon$ 产生式

ε产生式

$G': S \rightarrow AB \mid A \mid B \mid \epsilon \quad A \rightarrow aAA \mid \epsilon \quad B \rightarrow bBB \mid \epsilon$

- 其中 $A$ 和 $B$ 都是可空的。

$\bar{e}G'$ :

$S \rightarrow AB \mid A \mid B$

$A \rightarrow aAA \mid aA \mid a$

$B \rightarrow bBB \mid bB \mid b$

- $L(G') = L(\bar{e}G') \cup \{\epsilon\}$

## 定理5.8

证明

- 如果文法 $\bar{e}G$ 是利用去除 $\varepsilon$ 产生式的算法从 $G$ 得出的, 则 $L(\bar{e}G) = L(G) \setminus \{\varepsilon\}$ 。
- 令 $G = (V, T, P, S)$
- 令 $\bar{e}G = (V, T, P_{\bar{e}}, S)$
- 证明对任意变元 $A \in V$ ,  $A \Rightarrow_{\bar{e}G}^* w$ 当且仅当 $A \Rightarrow_G^* w$ 且 $w \neq \varepsilon$ 。
- 对推导步数进行归纳。



# 定理证明

邵永亮

- (当) 若  $w \neq \varepsilon$  且  $A \Rightarrow_G^* w$ , 则  $A \Rightarrow_{\bar{e}G}^* w$ 。
- 基础: 第1步时,  $(A, w) \in P$ , 由于  $w \neq \varepsilon$ , 所以  $(A, w) \in P_{\bar{e}}$ 。
- 归纳: 假设步数小于  $n$  时结论成立, 那么  $n$  步时, 不失一般性有  $A \Rightarrow_G X_1 X_2 \cdots X_k \Rightarrow_G^* w$ 。  $w$  可被分段成  $w = w_1 w_2 \cdots w_k$ , 对于  $i = 1, 2, \dots, k$ , 有  $X_i \Rightarrow_G^* w_i$ , 并且其步数小于  $n$ 。那么根据归纳假设, 若  $w_i \neq \varepsilon$ , 则有  $X_i \Rightarrow_{\bar{e}G}^* w_i$ 。
- 我们把推导为  $w_j = \varepsilon$  的  $X_j$  去掉, 对于所有  $j$ , 结果记为  $\alpha$ 。那么  $\alpha$  是  $X_1 X_2 \cdots X_k$  的一个变形, 则有  $\alpha \Rightarrow_{\bar{e}G}^* w$ , 同时  $A \rightarrow X_1 X_2 \cdots X_k$  和  $A \rightarrow \alpha$  都在  $\bar{e} \cdot G$  中, 因此,  $A \Rightarrow_{\bar{e}G} \alpha \Rightarrow_{\bar{e}G}^* w$ 。

# 定理证明

证明

- (仅当) 若  $A \Rightarrow_{\bar{e}G}^* w$  则  $A \Rightarrow_G^* w$  且  $w \neq \varepsilon$ 。
- 基础：推导步数为1时，必有  $(A, w) \in P_a$ ，这是原形产生式所以  $(A, w) \in P$  而且  $w \neq \varepsilon$ 。
- 归纳：假设推导步数小于  $n$  时结论成立，那么推导步数为  $n$  时，
- $A \Rightarrow_{\bar{e}G} \alpha \Rightarrow_{\bar{e}G}^* w$ ，由于  $\alpha \Rightarrow_{\bar{e}G}^* w$  的步数小于  $n$ ，所以根据归纳假设  $\alpha \Rightarrow_G^* w$ 。
- 我们找出  $(A, \alpha) \in P_{\bar{e}}$  的原形产生式  $(A, \delta) \in P_{\bar{e}}$ ，就有  $(A, \delta) \in P$  并且  $A \Rightarrow_G \delta \Rightarrow_G^* w$ 。

## 5.5.3 去除单位产生式

- **单位产生式**：右部为单个变元的产生式。
- 思路：若  $A \Rightarrow^* B$  是通过一系列单位产生式完成，并且  $B \rightarrow \alpha$  是一个非单位产生式，那么给文法中增加产生式  $A \rightarrow \alpha$ 。
- 此项工作完成后，删掉所有单位产生式。
- $S \rightarrow A|a$
- $A \rightarrow B/b/Cc$
- $B \rightarrow d/e$
- $S \rightarrow d/e/b/Cc/a$
- $A \rightarrow d/e/b/Cc$
- $B \rightarrow d/e$

# 发现单位产生式的算法

2016.12

- **关键点：**找出所有单位对 $(A, B)$ 使得 $A \Rightarrow^* B$ 仅通过一系列单位产生式得到。
- **基础：**对于任何变元， $(A, A)$ 是单位对。
- **归纳：**若已知 $(A, B)$ 是单位对且 $B \rightarrow C$ 是一个单位产生式，那么 $(A, C)$ 是单位对。
- 例： $\alpha, \beta, \gamma, \delta, \eta$ 都不是单个变元，
- $A \rightarrow B \mid \alpha$
- $B \rightarrow C \mid \beta$
- $C \rightarrow D \mid \gamma$
- $D \rightarrow \delta \mid \eta$
- 单位对： $(A, B), (A, C), (A, D), (B, C), (B, D), (C, D)$

# 证明算法正确工作

证明

- 基础：对于任何变元， $(A, A)$ 是单位对。
- 归纳：若已知 $(A, B)$ 是单位对且 $B \rightarrow C$ 是一个单位产生式，那么 $(A, C)$ 是单位对。
- 证明算法能发现给定文法的所有单位对。
- 按照找出单位对 $(A, B)$ 的次序归纳，能表明 $A \Rightarrow^* B$ 是通过单位产生式得到。
- 相反地，对经过单位产生式得到的 $A \Rightarrow^* B$ 的推导步数进行归纳，能够表明单位对 $(A, B)$ 被发现。

# 去除单位产生式的算法

2016

- 输入: CFG  $G = (V, T, P, S)$
- 输出:  $\bar{u} G = \text{CFG}(V, T, P_{\bar{u}}, S)$

求出 $G$ 的所有单位对;

对每个单位对 $(A, B)$ , 把所有的 $(A, \gamma) \in P$ 加入 $P_{\bar{u}}$ , 其中 $\gamma$ 不是单个变元且 $(B, \gamma) \in P$ 。

证明: 对任意 $T$ 上的符号串 $w$ 有 $w \in L(G)$ , 当且仅当 $w \in L(\bar{u} G)$

- (当)  $S \Rightarrow_{\bar{u} G}^* w$  为1步时有 $(S, w) \in P$ 或者存在单位对 $(S, A)$ 且 $(A, w) \in P$ 。假设步数小于 $n$ 时结论成立, 那么第 $n$ 步时,  $S \Rightarrow_{\bar{u} G} X_1 X_2 \cdots X_k \Rightarrow_{\bar{u} G}^* w$ , 其中对于 $i=1, 2, \dots, k$ , 有 $X_i \Rightarrow_{\bar{u} G}^* w_i$ , 那么根据归纳假设有 $X_i \Rightarrow_G^* w_i$ 。因此, 对于 $(S, X_1 X_2 \cdots X_k) \in P_{\bar{u}}$ , 或者有 $(S, X_1 X_2 \cdots X_k) \in P$ , 或者存在单位对 $(S, A)$ 且 $(A, X_1 X_2 \cdots X_k) \in P$
- (仅当) 仿照上面思考。

## 例：去除单位产生式

王

1. 求出 $G$ 的所有单位对；
2. 对每个单位对 $(A, B)$ ，把所有的 $(A, \delta) \in P$ 加入 $P_{\bar{u}}$ ，其中 $\delta$ 不是单个变元且 $(B, \delta) \in P$ 。

- 例： $\alpha, \beta, \gamma, \delta, \eta$ 都不是单个变元，

$$A \rightarrow B \mid \alpha \quad B \rightarrow C \mid \beta \quad C \rightarrow D \mid \gamma \quad D \rightarrow \delta \mid \eta$$

单位对： $(A, B)$ ， $(A, C)$ ， $(A, D)$ ， $(B, C)$ ， $(B, D)$ ， $(C, D)$

从而得到 $\bar{u}G$ ：

$$A \rightarrow \delta \mid \eta \mid \gamma \mid \beta \mid \alpha$$

$$B \rightarrow \delta \mid \eta \mid \gamma \mid \beta$$

$$C \rightarrow \delta \mid \eta \mid \gamma$$

$$D \rightarrow \delta \mid \eta$$

# 例：去除单位产生式

2018

- 例：  $G_{\text{ETF}}$ :

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

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

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

- 单位对：  $(E, T)$ ,  $(E, F)$ ,  $(T, F)$ 。

- $\bar{u}G_{\text{ETF}}$ :

$$E \rightarrow E+T \mid T^*F \mid (E) \mid d$$

$$T \rightarrow T^*F \mid (E) \mid d$$

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



- 定理5.10 若 $L$ 是一个 CFL, 那么有一个 $L \setminus \{\varepsilon\}$ 的CFG有如下特征:
  - 不含无用符号。
  - 不含 $\varepsilon$ 产生式。
  - 不含单位产生式。
- 每个文法符号都是有用的; 每个候选式或者是单个终结符或者是长度大于1的文法符号串。

证明: 由于 $L$ 是一个CFL, 那么必存在语言识别器, 设为CFG  $G$ 。那么  $ag\bar{u}\bar{e}G$  就是具备定理描述特征的文法。理由陈述为:  $\bar{e} \cdot G$  的每个候选式长度大于等于1;  $\bar{u}\bar{e} \cdot G$  的每个候选式或为终结符或长度大于1, 因为使用算法5.5对  $\bar{e} \cdot G$  去除单位产生式不会引入 $\varepsilon$ 产生式。最后对文法  $\bar{u}\bar{e} \cdot G$  去除无用符号不会增加产生式, 所以结果文法  $ag\bar{u}\bar{e} \cdot G$  具备定理所述特征。

由 $ag\bar{u}\bar{e}\cdot G$ 得到清理文法 $G$ 的步骤:

- 首先消除 $\varepsilon$ 产生式
  - 对 $G$ 做 $\bar{e}$ 修剪操作得到 $\bar{e}G$
- 其次消除单位产生式
  - 对 $\bar{e}G$ 做 $\bar{u}$ 修剪操作得到 $\bar{u}\bar{e}G$
- 接着消除无产出变元
  - 对 $\bar{u}\bar{e}G$ 做 $g$ 修剪操作得到 $g\bar{u}\bar{e}G$
- 最后消除不可达变元
  - 对 $g\bar{u}\bar{e}G$ 做 $a$ 修剪操作得到 $ag\bar{u}\bar{e}G$

- CFL语言识别器CFG，代数定义，变元与终结符、列表表示，
  - 判定性质：推导和归约、直接推导、最左和最右推导、语言、句子、句型、语法树、产物、
  - 推导与语法树扩展对应，归约与语法树构建对应
  - 歧义性与消除文法歧义性（运算符优先级原因）
  - 等价性质：四种修剪操作
  - 产生式左部与右部、候选式，符号串替换
- 
- 习题5.1(1)、5.4、5.6~8、5.10、5.13