SHANDONG UNIVERSITY

编译原理

第二章 文法与语言设计

授 课 教 师 ：郑艳伟
手 机 ：18614002860（微信同号）
邮 箱 ：zhengyw@sdu.edu.cn

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

第二章 文法与语言设计

第二章 文法与语言设计

2.1.1 基本概念

- 设Σ是一个有穷字母表，它的每个元素称为一个符号。
- Σ上的一个符号串是指由Σ中的符号所构成的一个有穷序列。
  - $\Sigma = \{a, b\}$，则a, b是符号， $a, b, aa, ab, bb, aaa, \ldots$都是符号串
- 符号在不同的编译阶段，所代表的意义有所不同
  - 词法分析阶段，每个字符是一个符号，符合某种规则的符号串构成单词；
  - 语法分析阶段，每个单词是一个符号，符合某种规则的符号串构成句子；
  - 语义分析和中间代码生成阶段，语义动作也被看做是一个符号。

郑艳伟，zhengyw@sdu.edu.cn

2.1.1 基本概念

- 不包含任何符号的序列称为空字（空串），记为ε（/'epsɪlɒn/）。
- 用Σ^*表示Σ上所有符号串的全体，空字ε也包括在其中，称为Σ的闭包。
  - $\Sigma = \{a, b\}$，则$\Sigma^* = \{\varepsilon, a, b, aa, ab, bb, aaa, \dots\}$。
- 用ϕ表示不含任何元素的空集$\{\}$。
  - 注意ε、$\{\}$、$\{\varepsilon\}$的区别。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.1.1 基本概念

- 符号串x和y的连接，指x和y的符号按先后顺序串联在一起组成的新符号串，记作xy。
  - 特别的，对任意符号串x，有$x\varepsilon = \varepsilon x = x$。
- 例：若$\Sigma = \{a, b\}, x = ab, y = abb$，则$xy = ababb, yx = abbab$，显然$xy \neq yx$。
- 符号串的方幂：设x是一个符号串，则$x^0 = \varepsilon, x^n = xx^{n-1} = x^{n-1}x$，其中$n = 1,2, \ldots$。
- 符号串的长度：指符号串x中符号的个数，用$|x|$表示。
  - $|\varepsilon| = 0, |a| = 1, |ab| = 2, |ababb| = 5$
- 符号串的前缀和后缀：符号串的左部任意子串，称为符号串的前缀；符号串的右部任意子串，称为符号串的后缀。
  - 符号串$aabb$的前缀有$\varepsilon, a, aa, aab, aabb$，后缀有$\varepsilon, b, bb, abb, aabb$。

2.1.1 基本概念

- Σ^*的子集（符号串集合）U和V的连接（积）定义为：$UV=\{\alpha\beta|\alpha\in U,\beta\in V\}$(逗号表示与)；特别地，$U\{\varepsilon\}=\{\varepsilon\}U=U$
  - 设$\Sigma=\{a,b,c\},A=\{aa,bb\},B=\{ac,c\}$，则$AB=\{aaac,aac,bbac,bbc\},BA=\{acaa,acbb,caa,cbb\}$
  - 一般而言，$UV\neq VU$，但$(UV)W=U(VW)$。
- V自身的n次连接（方幂）记为：$V^n=\underbrace{VV\ldots V}_{n}$。
  - 规定：$V^0=\{\varepsilon\}$。
- V的闭包：$V^*=V^0\cup V^1\cup V^2\cup\cdots$。
  - 闭包V^*中的每个符号串都是由V中的符号串经过有限次连接而成的，即闭包中每个字符串长度有限。
- V的正则闭包（正闭包）：$V^+=VV^*$。

第二章 文法与语言设计

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.1.2 文法

【例2.1】He gave me a book.

- <句子> → <主语> <谓语> <间接宾语> <直接宾语>.
- <主语> → <代词>
- <谓语> → <动词>
- <间接宾语> → <代词>
- <直接宾语> → <冠词> <名词>
- <代词> → He
- <代词> → me
- <冠词> → a
- <动词> → gave
- <名词> → book

2.1.2 文法

- 反复利用规则，将→左边的符号替换成右边，推导出句子：

<句子>⇒ <主语> <谓语> <间接宾语> <直接宾语>.

⇒ <代词> <谓语> <间接宾语> <直接宾语>.

⇒ He <谓语> <间接宾语> <直接宾语>.

⇒ He <动词> <间接宾语> <直接宾语>.

⇒ He gave <间接宾语> <直接宾语>.

⇒ He gave <代词> <直接宾语>.

⇒ He gave me <直接宾语>.

⇒ He gave me <冠词> <名词>.

⇒ He gave me a <名词>.

⇒ He gave me a book.

- 显然也推导出句子：He gave He a book. / me gave He a book. / me gave me a book.

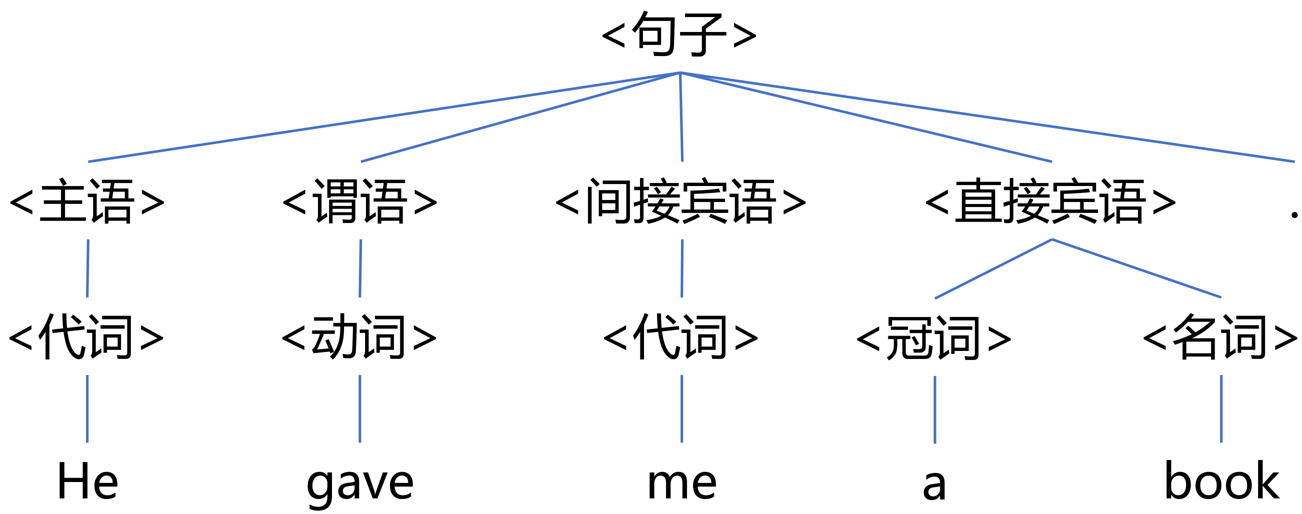
郑艳伟，zhengyw@sdu.edu.cn

2.1.2 文法

【例2.1】He gave me a book.

- <句子> → <主语> <谓语> <间接宾语> <直接宾语>.
- <主语> → <主格代词>
- <谓语> → <动词>
- <间接宾语> → <宾格代词>
- <直接宾语> → <冠词> <名词>
- <主格代词> → He
- <宾格代词> → me
- <冠词> → a
- <动词> → gave
- <名词> → book

2.1.2 文法



上下文无关文法要素

- **终结符号**：是组成语言的基本符号，是语言的一个不可再分的单位。
- **非终结符号**：也称语法变量，用来代表语法范畴，是一个类的记号，而不是个体记号。
- **产生式**：也称产生规则或简称规则，是定义语法范畴的一种书写规则。
- **开始符号**：是一个特殊的非终结符号，代表所定义的语言中我们最感兴趣的语法范畴。

2.1.2 文法

- 形式化定义：一个文法G是一个四元式(V_N, V_T, P, S)
  - V_N是一个非空有限集合，它的每个元素称为非终结符号；
  - V_T是一个非空有限集合，它的每个元素称为终结符号，$V_T \cap V_N = \phi$；
  - P是一个产生式集合（有限），每个产生式的形式是$\alpha \to \beta$，其中$\alpha \in (V_T \cup V_N)^* V_N (V_T \cup V_N)^*, \beta \in (V_T \cup V_N)^*$；
  - S是一个非终结符号，称为开始符号；其至少在某个产生式左部出现一次。
  - 特别地，若$\alpha \in V_N$，则称该文法为上下文无关文法（Context-Free Grammar）。
- 为书写方便，经常采用如下简写，每个α_i也称为是P的一个候选式
  - $$\left.\begin{array}{l} P \to \alpha_1 \\ P \to \alpha_2 \\ \quad \cdots\cdots \\ P \to \alpha_n \end{array}\right\} \Leftrightarrow P \to \alpha_1 | \alpha_2 | \ldots | \alpha_n$$
- 箭头 $\to$ 读为“定义为”，直竖| 读为“或”，它们是元语言符号。

2.1.2 文法

约定

- 用大写字母A、B、C…，或带尖括号的词组如<算术表达式>，代表非终结符号；
- 用小写字母a、b、c…代表终结符号；
- 用希腊字母α、β、γ等代表由终结符号和非终结符号组成的字符串。
- 为简便起见，当引用具体文法的例子时，仅列出产生式和指出开始符号。

【例】G[E]：

$$E \rightarrow i \mid EAE$$

$$A \rightarrow + \mid *$$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

产生式

- 产生式：$\alpha \rightarrow \beta$
  - 左边的α称为产生式的左部
  - 右边的β称为产生式的右部
  - 有时也说$\alpha \rightarrow \beta$是关于A的一条产生规则。
  - 有的书上，→也用::=表示，这种表示方法也称为巴科斯范式 (BNF)。

【例】递归产生式定义加乘算术表达式G[E]:

$E \rightarrow i$

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow (E)$

第二章 文法与语言设计

2.1.3 推导和归约

- 【例2.4】有如下文法：$E \rightarrow E + E \mid E * E \mid (E) \mid i$
  - 推导：$E \Longrightarrow (E) \Longrightarrow (E + E) \Longrightarrow (i + E) \Longrightarrow (i + i)$
  - 这个推导提供了一个证明，证明$(i + i)$是这个文法所定义的一个算术表达式。
- 我们称$\alpha A\beta$直接推导出$\alpha\gamma\beta$，即$\alpha A\beta \Rightarrow \alpha\gamma\beta$，当且仅当$A \rightarrow \gamma$是一个产生式，且$\alpha$、$\beta \in (V_T \cup V_N)^*$
  - 如果$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n$，则称这个序列是从$\alpha_1$到$\alpha_n$的一个推导；
  - 若存在一个从α_1到α_n的推导，则称α_1可推导出α_n，其逆过程称为归约；
  - 用$\alpha_1 \overset{+}{\Rightarrow} \alpha_n$表示从$\alpha_1$出发，经一步或若干步，可推导出$\alpha_n$；
  - 用$\alpha_1 \overset{*}{\Rightarrow} \alpha_n$表示从$\alpha_1$出发，经0步或若干步，可推导出$\alpha_n$，即$\alpha_1 = \alpha_n$或$\alpha_1 \overset{+}{\Rightarrow} \alpha_n$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

规范推导和规范归约

- 一个句型到另一个句型的推导过程往往不唯一：$E \to E + E \mid E * E \mid (E) \mid i$
  - $E \Longrightarrow E + E \Longrightarrow i + E \Longrightarrow i + i$
  - $E \Longrightarrow E + E \Longrightarrow E + i \Longrightarrow i + i$
- 约束
  - 若推导过程中，总是最先替换最右(左)的非终结符，则称为最右(左)推导；
  - 若归约过程中，总是最先归约最右(左)的非终结符，则称为最右(左)归约。
- 规范
  - 句型的最右推导称为规范推导，其逆过程最左归约称为规范归约；
  - $i * i + i$的规范推导：$E \Rightarrow E + E \Rightarrow E + i \Rightarrow E * E + i \Rightarrow E * i + i \Rightarrow i * i + i$
  - $i * i + i$的规范归约：$i * i + i \Leftarrow E * i + i \Leftarrow E * E + i \Leftarrow E + i \Leftarrow E + E \Leftarrow E$
  - $i * i + i$的最右推导2：$E \Rightarrow E * E \Rightarrow E * E + E \Rightarrow E * E + i \Rightarrow E * i + i \Rightarrow i * i + i$

第二章 文法与语言设计

SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

语言

- 假定G是一个文法，S是它的开始符号，如果$S \overset{*}{\Rightarrow} \alpha$，则称$\alpha$是一个句型。
- 如果一个句型中只包含终结符号，则称其为一个句子。
- 文法G所产生的句子的全体是一个语言，记为：$L(G) = \{\alpha | S \overset{+}{\Rightarrow} \alpha, \alpha \in V_T^*\}$。
- 【例】有文法G：$E \rightarrow E + E \mid E * E \mid (E) \mid i$
  - $(i * i + i)$是该文法的一个句子，因为有推导：$E \Rightarrow (E) \Rightarrow (E + E) \Rightarrow (E * E + E) \Rightarrow (i * E + E) \Rightarrow (i * i + E) \Rightarrow (i * i + i)$；
  - $E, (E), (E + E), (i * E + E), \ldots, (i * i + i), i * i + i$都是这个文法的句型。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

语言

- 【例】有文法$G_1=(V_N,V_T,P,S)$，其中：

$V_N=\{<数字串>,<数字>\};V_T=\{0,1,2,3,4,5,6,7,8,9\};$

$P=\{<数字串>\to<数字串><数字>\mid<数字>,$

$<数字>\to 0|1|2|3|4|5|6|7|8|9\};$

$S=\{<数字串>\}$

确定G_1对应的语言。

  - 【分析】由$<数字串>\to<数字>$，可得数字串为0~9的任意数字

    每次用$<数字串>\to<数字串><数字>$推导，末尾就增加一个0~9的数字，直到使用$<数字串>\to<数字>$推导为止。

  - 【结论】$L(G_1)$表示十进制非负整数。

形如$A\to A$ ...的产生式称为左递归产生式；形如$A\to\cdots A$的产生式称为左递归产生式

语言

- 【例】有文法$G_2[S]: S \rightarrow bA, A \rightarrow aA|a$，确定$G_2$对应的语言。
  - $S \Rightarrow bA \Rightarrow ba$
  - $S \Rightarrow bA \Rightarrow baA \Rightarrow baa$
  - $S \Rightarrow bA \Rightarrow baA \Rightarrow baaA \Rightarrow baaa$
  - $\ldots$
  - $S \Rightarrow bA \Rightarrow baA \Rightarrow \cdots \Rightarrow ba \ldots a$
  - 归纳得：$L(G_2) = \{ba^n|n \geq 1\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

语言

- 【例】有文法$G_3[S]: S \to AB, A \to aA|a, B \to bB|b$，确定$G_3$对应的语言。
  - $S \Rightarrow AB \Rightarrow ab$
  - $S \Rightarrow AB \Rightarrow aAB \Rightarrow aaB \Rightarrow aab$
  - $\ldots$
  - $S \Rightarrow AB \Rightarrow AB \Rightarrow aB \Rightarrow abB \Rightarrow abb$
  - $S \Rightarrow AB \Rightarrow AB \Rightarrow aB \Rightarrow abB \Rightarrow abbB \Rightarrow abbb$
  - $\ldots$
  - $S \Rightarrow AB \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow a \ldots aB \Rightarrow a \ldots abB \Rightarrow a \ldots ab \ldots b$
  - 归纳得：$L(G_3) = \{a^m b^n | m \geq 1, n \geq 1\}$

语言

- 【例】构造文法：$L(G_4) = \{a^n b^n | n \geq 1\}$。
  - ab
  - $aabb$
  - $aaabbb$
  - $\dots$
- 从另一个角度看：
  - ab
  - $aabb$
  - $aaabbb$
  - $\dots$
  - 归纳得：$G_4[S]$: $S \to aSb|ab$

郑艳伟，zhengyw@sdu.edu.cn

语言

- 【例】构造文法G_5，使其描述的语言为正奇数集合。
- 【分析】正奇数要求，要么是一位奇数数字，要么是以奇数数字结尾的十进制数字。
- 【解】令$G_5=(V_N,V_T,\mathcal{P},<\text{正奇数}>)$; $V_T=\{0,1,2,3,4,5,6,7,8,9\}$
  - P: $<\text{一位奇数}>\rightarrow 1|3|5|7|9$; $<\text{一位数字}>\rightarrow<\text{一位奇数}>|0|2|4|6|8$
  - $<\text{正奇数}>\rightarrow<\text{一位奇数}>|<\text{数字串}><\text{一位奇数}>$
  - $<\text{数字串}>\rightarrow<\text{一位数字}>|<\text{数字串}><\text{一位数字}>$
  - $V_N=\{<\text{正奇数}>,<\text{数字串}>,<\text{一位数字}>,<\text{一位奇数}>\}$

第二章 文法与语言设计

- 2.1 文法和语言
  - 2.1.1 基本概念
  - 2.1.2 文法
  - 2.1.3 推导和归约
  - 2.1.4 语言
  - 2.1.5 文法的Chomsky分类
- 2.2 语法树与二义文法
  - 2.2.1 短语和句柄
  - 2.2.2 语法树
  - 2.2.3 二义文法
- 2.3 程序语言设计
  - 2.3.1 正规式
  - 2.3.2 正规式的等价变换
  - 2.3.3 基本运算的文法设计
  - 2.3.4 连接-闭包和闭包-连接
  - 2.3.5 拆分括号对
  - 2.3.6 表达式优先级与结合性
- 2.4 文法的等价变换
  - 2.4.1 消除无用产生式
  - 2.4.2 消除单非产生式
  - 2.4.3 消除空符产生式

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.1.5 文法的Chomsky分类

- Chomsky于1956年建立了形式语言的描述，并将文法划分为4种类型。

定义 2.12 (0 型文法)

对文法 $G=(V_N,V_T,P,S)$，称 G 为 0 型文法（Type 0 Grammar）或短语文法（Phrase Structure Grammar，PSG）。

$L(G)$ 称为 0 型语言或短语结构语言 (PSL)、递归可枚举集（Recursively Enumerable Set）。♣

定义 2.13 (1 型文法)

对文法 $G=(V_N,V_T,P,S)$，产生式形式为 $\alpha A\beta\rightarrow\alpha\gamma\beta$，其中 $A\in V_N,\alpha\in V^*,\beta\in V^*,\gamma\in V^*$，称 G 为 1 型文法（Type 1 Grammar）或上下文有关文法（Context Sensitive Grammar，CSG）。

$L(G)$ 称为 1 型语言（Type 1 Language）或上下文有关语言（Context Sensitive Language，CSL）。♣

2.1.5 文法的Chomsky分类

定义 2.14 (2 型文法)

对文法 $G=(V_N,V_T,P,S)$，任何产生式 $A\to\beta\in P$，均有 $A\in V_N,\beta\in V^*$，称 G 为 2 型文法（Type 2 Grammar）或上下文无关文法（Context Free Grammar，CFG）。

$L(G)$ 称为 2 型语言（Type 2 Language）或上下文无关语言（Context Free Language，CFL）。♣

定义 2.15 (3 型文法)

对文法 $G=(V_N,V_T,P,S)$，产生式形式均为 $A\to aB$ 或 $A\to b$，其中 $A\in V_N,B\in V_N,a\in V_T,b\in V_T\cup\{\varepsilon\}$，称 G 为 3 型文法（Type 3 Grammar），也称为正则文法或正规文法（Regular Grammar，RG）。

$L(G)$ 称为 3 型语言（Type 3 Language）或正则语言、正规语言（Regular Language，RL）。♣

- 3型文法还有另外一种形式，称为左线性文法，如果产生式形式为：$A\to Ba$或$A\to b$，其中，$a\in V_T,b\in V_T\cup\{\varepsilon\},A\in V_N,B\in V_N$。

几个有趣的结论

- 正规文法(3)不能产生语言$L(G) = \{a^n b^n | n \geq 1\}$，上下文无关文法(2)则可以：$S \rightarrow aSb|ab$。
- $L(G) = \{a^n b^n c^i | i \geq 1, n \geq 1\}$是一个上下文无关语言：$S \rightarrow AB, A \rightarrow aAb|ab$，$B \rightarrow Bc|c$。
- 语言$L(G) = \{a^n b^n c^n | n \geq 1\}$只能用上下文有关文法(1)产生：
  - $S \rightarrow aSBA|abB,$
  - $BA \rightarrow BA', \quad BA' \rightarrow AA', \quad AA' \rightarrow AB,$
  - $bA \rightarrow bb, \quad bB \rightarrow bc, \quad cB \rightarrow cc$
- 语言$L(G) = \{\alpha c \alpha | \alpha \in (a|b)^*\}$只能用0型文法产生。

第二章 文法与语言设计

- 2.1 文法和语言
  - 2.1.1 基本概念
  - 2.1.2 文法
  - 2.1.3 推导和归约
  - 2.1.4 语言
  - 2.1.5 文法的Chomsky分类
- 2.2 语法树与二义文法
  - 2.2.1 短语和句柄
  - 2.2.2 语法树
  - 2.2.3 二义文法
- 2.3 程序语言设计
  - 2.3.1 正规式
  - 2.3.2 正规式的等价变换
  - 2.3.3 基本运算的文法设计
  - 2.3.4 连接-闭包和闭包-连接
  - 2.3.5 拆分括号对
  - 2.3.6 表达式优先级与结合性
- 2.4 文法的等价变换
  - 2.4.1 消除无用产生式
  - 2.4.2 消除单非产生式
  - 2.4.3 消除空符产生式

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.2.1 短语和句柄

- **短语**：对文法$G[S]$，如果有$S \overset{*}{\Rightarrow} \alpha A\delta$且$A \overset{+}{\Rightarrow} \beta$，则称$\beta$是句型$\alpha\beta\delta$相对于非终结符号$A$的短语。
  - 特别地，如果有$A \Rightarrow \beta$，则称β是句型$\alpha\beta\delta$相对于A的直接短语。
  - 一个句型的最左直接短语称为该句型的句柄。

2.2.1 短语和句柄

【例】对文法$G[E]$

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

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

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

- 句型$i * i + i$
  - $E \Rightarrow E + T \Rightarrow E + F \Rightarrow E + i \Rightarrow T + i \Rightarrow F + i \Rightarrow i + i$
  - 但$i + i$不是该句型的一个短语，因为$E \not\Rightarrow i * E$。
- 句型$E + T * F + i$
  - $E \Rightarrow E + T \Rightarrow E + F \Rightarrow E + i \Rightarrow E + T + i \Rightarrow E + T * F + i$
  - $E + T * F + i$是句型$E + T * F + i$相对于E的短语。
  - $E + T * F$是句型$E + T * F + i$相对于E的短语。
  - $T * F$是句型$E + T * F + i$相对于T的短语，且是直接短语，也是句柄。
  - i是句型$E + T * F + i$相对于F的短语，且是直接短语。

2.2.1 短语和句柄

【例】对文法$G[E]$

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

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

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

- 句型 $i * i + i$
  - $E \Rightarrow E + T \Rightarrow E + F \Rightarrow E + i \Rightarrow T + i \Rightarrow F + i \Rightarrow i + i$
  - 但$i + i$不是该句型的一个短语，因为$E \nRightarrow i * E$。

【作为对比】对文法$G[E]$：$E \rightarrow E + E \mid E * E \mid (E) \mid i$

- $E \Rightarrow E * E \Rightarrow i * E \Rightarrow i * E + E \Rightarrow i * i + E \Rightarrow i * i + i$
- $E \overset{*}{\Rightarrow} i * E$，$E \overset{+}{\Rightarrow} i + i$，因此 $i + i$ 是 $i * i + i$ 的一个短语。

第二章 文法与语言设计

2.2.2 语法树

- 语法分析树，简称语法树(Syntax Tree)，用树形图表示一个句型的推导过程
  - 一棵语法树表示了句型的种种可能的不同推导过程（但未必是全部），包括最左（最右）推导，即一棵语法树是不同推导过程的共性抽象。
- 语法树根结点为开始符号，叶结点从左到右为所要推导的句型，内部结点是推导过程中用到的非终结符。
  - 对同一个句型，可能有不同的推导次序可以到达这个句型，语法树隐藏了替换次序的信息，表现了一个静态的推导结构。

郑艳伟，zhengyw@sdu.edu.cn

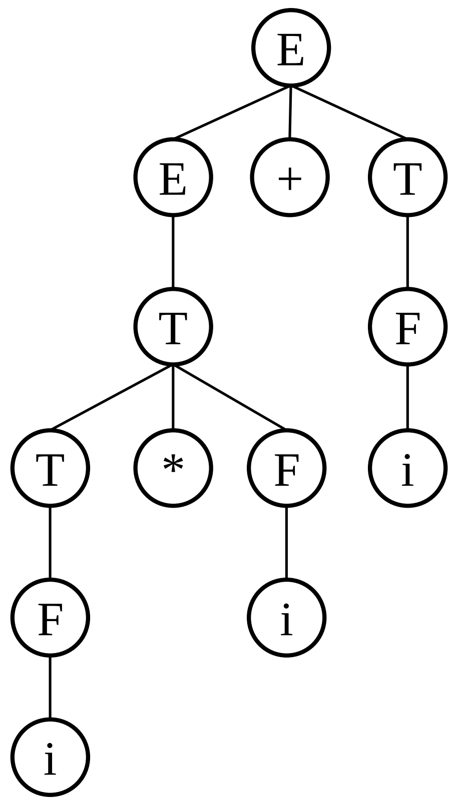
2.2.2 语法树

【例】对文法$G[E]$

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

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

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



2.2.2 语法树

- 语法树的子树叶结点构成短语，二层子树叶结点构成直接短语，最左二层子树叶结点构成句柄。
  - 因为二层子树是针对这个句型的最后一步推导，因此是直接短语；
  - 最左二层子树父结点直接推出子节点，且最左，即对应最左直接短语；

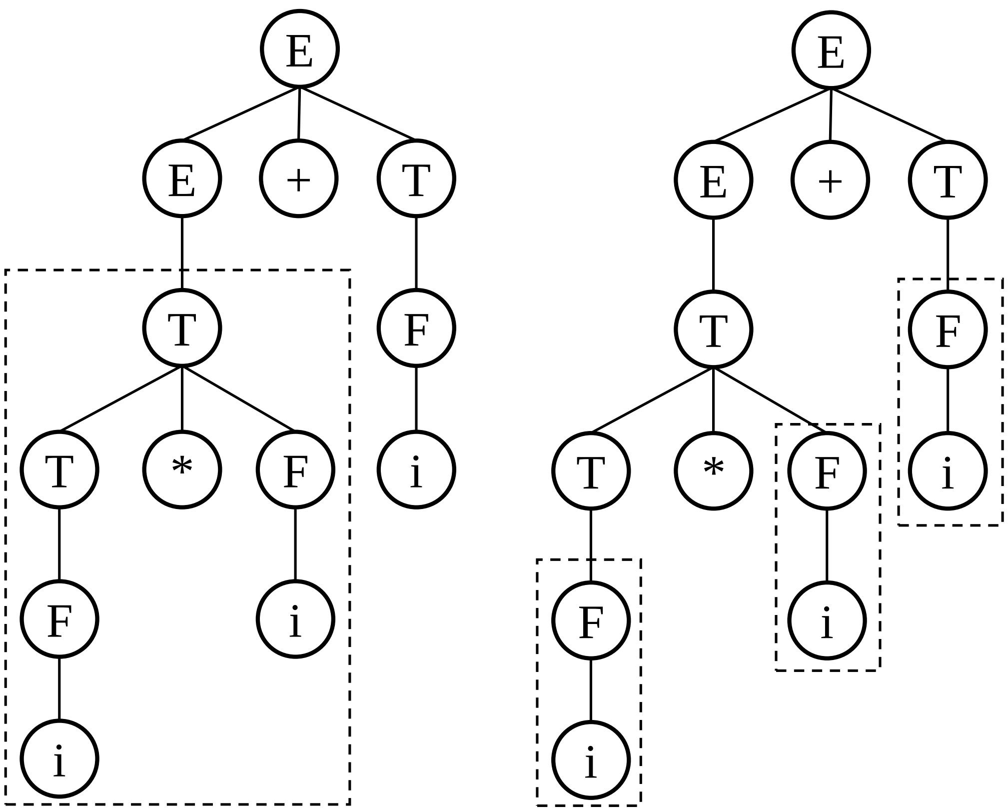
【例】对文法$G[E]$

$$E \to T \mid E + T$$

$$T \to F \mid T * F$$

$$F \to i \mid (E)$$

这一点其实对寻找句柄没有帮助，因为语法分析完成才能构造出语法树。

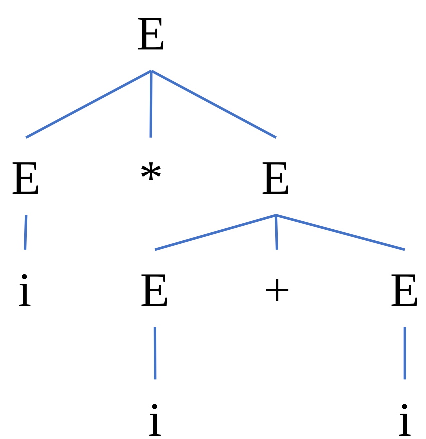


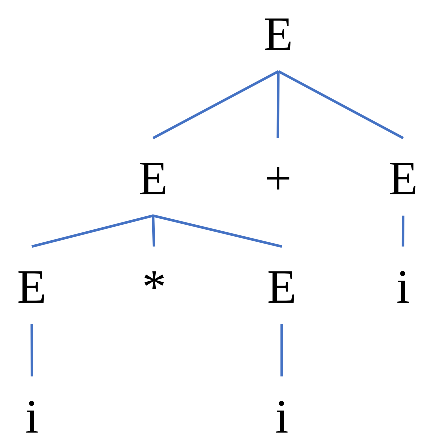
第二章 文法与语言设计

郑艳伟，zhengyw@sdu.edu.cn

二义文法

- **二义文法：**如果一个文法的某个句型对应两棵不同的语法树，即其最左（最右）推导不唯一，称该文法为二义文法。
- 【例】$E \rightarrow E + E \mid E * E \mid (E) \mid i$，关于句子i*i+i的最右推导：
  - $E \Rightarrow E * E \Rightarrow E * E + E \Rightarrow E * E + i \Rightarrow E * i + i \Rightarrow i * i + i$
  - $E \Rightarrow E + E \Rightarrow E + i \Rightarrow E * E + i \Rightarrow E * i + i \Rightarrow i * i + i$





2.2.3 二义文法

- 文法的二义性和语言的二义性是不同的概念
  - 可能有两个不同的文法G和G'，其中一个是二义的而另一个是无二义的，但是有$L(G)=L(G')$；
  - 对程序设计语言来说，常常希望它的文法是无二义的，因为我们希望对它每个语句的分析是唯一的；
  - 但是，只要能控制和驾驭文法的二义性，有时候存在二义性并不一定是坏事；
  - 目前已经证明，二义性问题是不可判定的，即不存在一个算法，它能在有限步骤内确切的判定一个文法是否为二义性的。

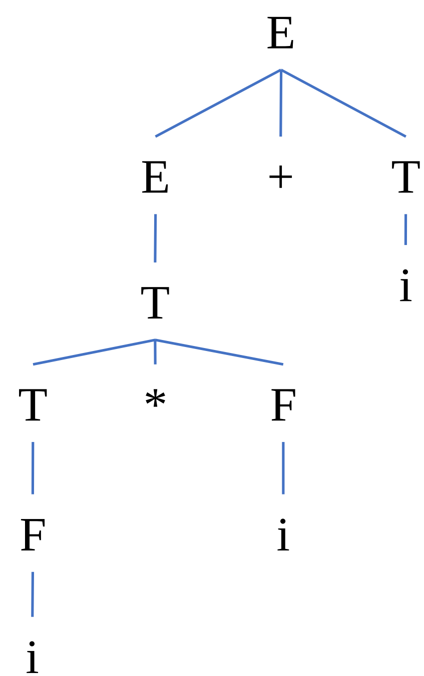
郑艳伟，zhengyw@sdu.edu.cn

2.2.3 二义文法

- 【例】$E \rightarrow E + E \mid E * E \mid (E) \mid i$，构造该文法的无二义文法，使它们表示的语言相同，并给出句子i*i+i的最右推导。

【解】 $E \rightarrow T \mid E + T$, $T \rightarrow F|T * F, F \rightarrow (E)|i$ (优先级越高越远离开始符号)

- $E \Rightarrow E + T \Rightarrow E + i \Rightarrow T + i \Rightarrow T * F + i \Rightarrow T * i + i \Rightarrow F * i + i \Rightarrow i * i + i$

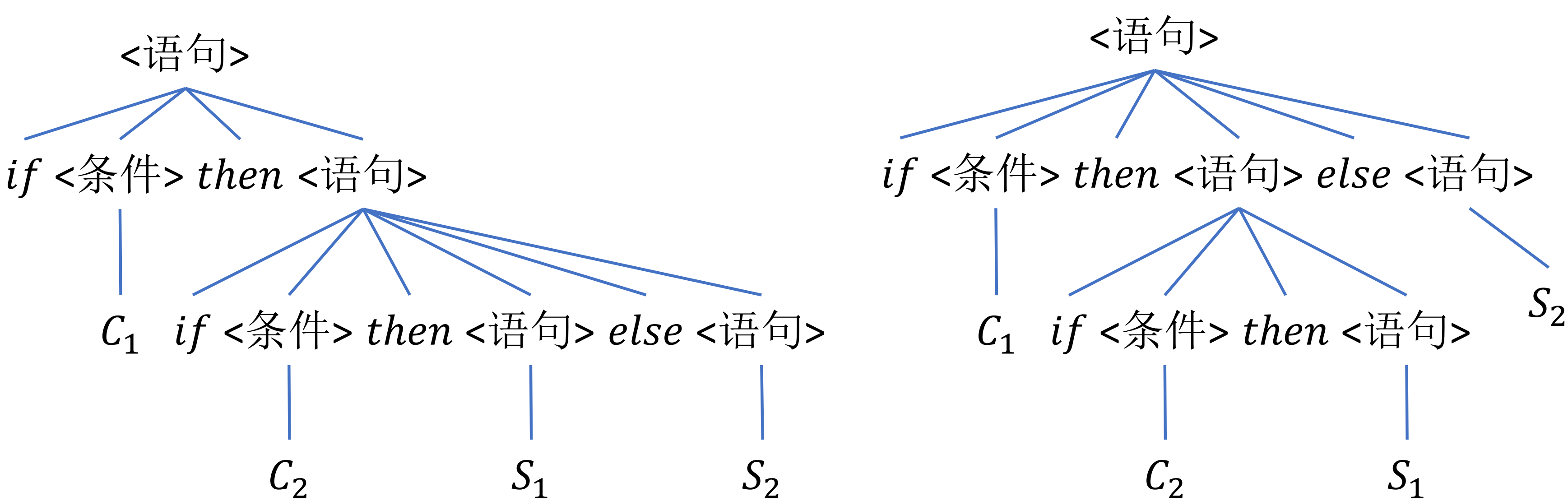


郑艳伟，zhengyw@sdu.edu.cn

几个有趣的结论

- **上下文无关文法**表示**条件语句**：
  - $<\text{语句}> \rightarrow if < \text{条件} > then < \text{语句} >$

    $\mid if < \text{条件} > then < \text{语句} > else < \text{语句} >$
  - 这是个二义文法，如句子：$if\ C_1\ then\ if\ C_2\ then\ S_1\ else\ S_2$



几个有趣的结论

- 上下文无关文法表示条件语句：
  - $<语句>\rightarrow if <条件> then <语句>$

    $| if <条件> then <语句> else <语句>$
  - 这是个二义文法，如句子：$if\ C_1\ then\ if\ C_2\ then\ S_1\ else\ S_2$
- 一般语言都规定：else必须匹配最后那个未得到匹配的then，即就近匹配
  - $<语句>\rightarrow <匹配句> | <非匹配句>$
  - $<匹配句>\rightarrow if <条件> then <匹配句> else <匹配句>$

    $| <其它语句>$
  - $<非匹配句>\rightarrow if <条件> then <语句>$

    $| if <条件> then <匹配句> else <非匹配句>$

郑艳伟，zhengyw@sdu.edu.cn

约定

- 对上下文无关文法，对其施加以下限制，满足这两个条件的文法也称化简了的文法：
  - 文法不含产生式$P \rightarrow P$，因为这种产生除了引起二义性外没有任何用处。
  - 每个非终结符P必须都有用处，这意味着必须存在推导：$(1) S \overset{*}{\Rightarrow} \alpha P \beta$，以及 $(2)\ P \overset{+}{\Rightarrow} \gamma, \gamma \in V_T^*$。

第二章 文法与语言设计

2.3.1 正规式

正规式 (Regular Expression) 递归定义

① ε和ϕ都是Σ上的正规式，它们所表示的正规集分别为$\{\varepsilon\}$和ϕ。

② 任何$a \in \Sigma$，a是Σ上的一个正规式，它所表示的正规集是$\{a\}$。

③ 假定U和V都是Σ上的正规式，它们所表示的正规集分别记为$L(U)$和$L(V)$，那么$(U|V), (U \cdot V), (U)^*$也都是正规式，它们所表示的正规集分别为$L(U) \cup L(V)$, $L(U)L(V), (L(U))^*$。

- 仅由有限次使用上述三步骤得到的表达式才是Σ上的正规式，仅由这些正规式所表示的字集才是Σ上的正规集。

正规式符号

- “|”读“或”；“·”读“连接”，可以省略；“*”读“闭包”。
- 在不致混淆时，括号可以省略；
- 优先级由高到低顺序：*, ·, |

2.3.1 正规式

【例】令$\Sigma=\{a,b\}$，下面是Σ上的正规式和相应的正规集：

- ba^* Σ上所有以b为首后跟任意多个a的字
- $a(a|b)^*$ Σ上所有以a为首的字
- $(a|b)^*(aa|bb)(a|b)^*$ Σ上含有两个连续a或两个连续b的字

【例】令$\Sigma=\{A,B,0,1\}$，下面是Σ上的正规式和相应的正规集：

- $(A|B)(A|B|0|1)^*$ Σ上“标识符”的全体
- $(0|1)(0|1)^*$ Σ上“数”的全体

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.3.1 正规式

【例】令$\Sigma = \{0,1\}$，构造正规式，使其表示包含偶数个0和偶数个1的字

- 把两个字符划分为一组，有以下三种情形

① 00，这种已满足要求

② 11，也已满足要求

③ 10或01开头，则中间经历任意多00|11，最后必须以10或01结尾，即：

$(10|01)(00|11)^*(10|01)$

- 由以上三种任意组合，即可满足要求：$(((10|01)(00|11)^*(10|01)) \mid 00 \mid 11)^*$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.3.1 正规式

【例】常用正规式

- 整数：$(0|1|2|\ldots|9)(0|1|2|\ldots|9)^*$
- 浮点数： $(0|1|2|\ldots|9)(0|1|2|\ldots|9)^*.(0|1|2|\ldots|9)(0|1|2|\ldots|9)^*$
- 浮点数指数表示法： $<$ 浮点数 $> E <$ 整数 $>$
- 标识符：$(a|b|\ldots|z|A|B|\ldots|Z|\_)(a|b|\ldots|z|A|B|\ldots|Z|\_|0|1|\ldots|9)^*$

- 有时用“+”表示至少1次出现
  - 整数：$(0|1|2|\ldots|9)^+$
  - 浮点数： $(0|1|2|\ldots|9)^+.(0|1|2|\ldots|9)^+$

第二章 文法与语言设计

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.3.2 正规式的等价变换

- 若两个正规式表示的正规集相同，则认为两个正规式等价
  - 两个等价的正规式U和V，记为$U = V$
  - 例如，$b(ab)^* = (ba)^*b, (a|b)^* = (a^*b^*)^*$
- 正规式的计算：假设U, V, W是正规式
  - 交换律：$U \mid V = V \mid U$，$UV \neq VU$
  - 结合律：$(U|V) \mid W = U \mid (V|W)$，$(UV)W = U(VW)$
  - 分配律：$(U|V)W = UW \mid VW$，$U(V|W) = UV \mid UW$
  - 乘法单位元：$\varepsilon U = U\varepsilon = U$

第二章 文法与语言设计

- 2.1 文法和语言
  - 2.1.1 基本概念
  - 2.1.2 文法
  - 2.1.3 推导和归约
  - 2.1.4 语言
  - 2.1.5 文法的Chomsky分类
- 2.2 语法树与二义文法
  - 2.2.1 短语和句柄
  - 2.2.2 语法树
  - 2.2.3 二义文法
- 2.3 程序语言设计
  - 2.3.1 正规式
  - 2.3.2 正规式的等价变换
  - 2.3.3 基本运算的文法设计
  - 2.3.4 连接-闭包和闭包-连接
  - 2.3.5 拆分括号对
  - 2.3.6 表达式优先级与结合性
- 2.4 文法的等价变换
  - 2.4.1 消除无用产生式
  - 2.4.2 消除单非产生式
  - 2.4.3 消除空符产生式

2.3.3 基本运算的文法设计

正规式设计文法的理论问题

- 正规式和3 型文法等价，可以设计出自动生成3 型文法的算法；
- 使用正规式生成2 型文法显然不存在理论上的障碍；
- 在语义分析和中间代码生成中，经常需要文法符号在推导或归约过程中传递信息，这就对文法提出了各种额外的要求，本节介绍从正规式到2型文法的手工设计方法。

2.3.3 基本运算的文法设计

- 连接运算：$\alpha\beta$
  - $A \to \alpha B, B \to \beta$
  - $A \to B\beta, B \to \alpha$
  - $A \to BC, B \to \alpha, C \to \beta$
- 或运算：$\alpha|\beta$
  - $A \to \alpha, A \to \beta$
- 闭包运算：α^*
  - $A \to A\alpha|\varepsilon$，推导先右后左，归约先左后右
  - $A \to \alpha A|\varepsilon$，推导先左后右，归约先右后左

2.3.3 基本运算的文法设计

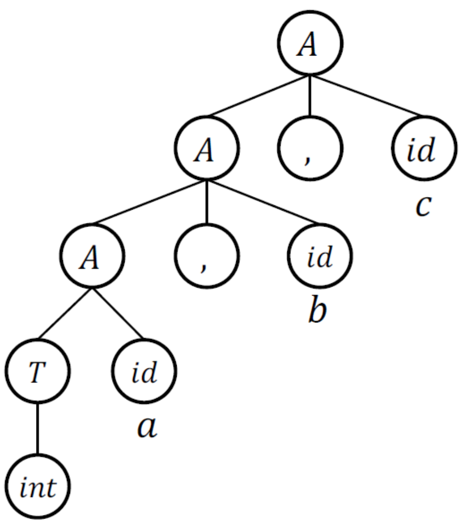
- 过程定义：int sum(int x, int y) {...}
  - $T\ id(T\ id,\ T\ id)$
  - $T\ id(T\ id(,\ T\ id)^*)$
  - 从左向右归约：$S \rightarrow T\ id(T\ id\ A), A \rightarrow A, T\ id | \varepsilon, T \rightarrow int$
- 过程调用：sum(x, y)
  - $id(id,\ id)$
  - $id(id\ (, id)^*)$
  - 从右向左归约：$S \rightarrow id(id\ A), A \rightarrow , id\ A | \varepsilon, T \rightarrow int$

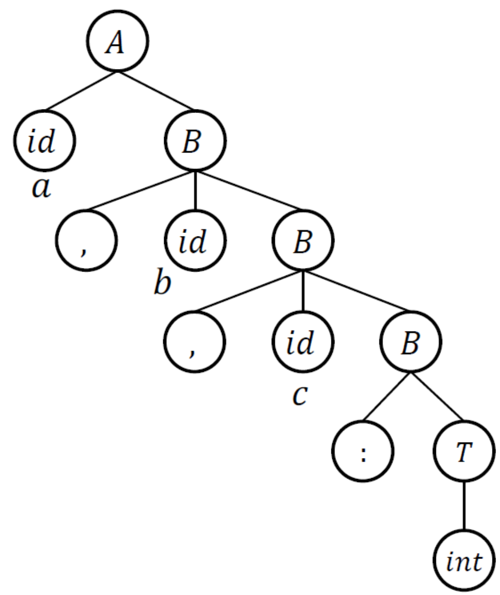
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

第二章 文法与语言设计

2.3.4 连接-闭包和闭包-连接

- 连接-闭包运算：$\alpha\beta^*$
  - $A \to A\beta|\alpha$
  - $A \to \alpha B, B \to \beta B|\varepsilon$
- 闭包-连接运算：$\alpha^*\beta$
  - $A \to B\beta, B \to B\alpha|\varepsilon$
  - $A \to \alpha A|\beta$
- C风格声明语句：int a, b, c
  - $T\ id\ (,id)^*$
  - $A \to A, id\ |\ T\ id, T \to int$
- Pascal风格声明语句：a, b, c: int
  - $id\ (,id)^* : T$
  - $A \to id\ B, B \to, id\ B|: T, T \to int$





第二章 文法与语言设计

2.3.5 拆分括号对

- 数组引用：a[2, 3, 4]
  - $id[num(,num)^*]$ 或 $id[(num,)^*num]$
  - 中括号内部是一个连接-闭包结构或者闭包-连接结构。
- 下标计算：假设每维长度为n_i，对$id[k_1, k_2, \cdots, k_m]$，递归计算偏移量
  - $e_1 = k_1$
  - $e_2 = e_1 \times n_2 + k_2$
  - $e_3 = e_2 \times n_3 + k_3$
  - ……
  - $e_m = e_{m-1} \times n_m + k_m$
- 每个n_i 要根据id查表得到，id至少需要和数组下标第1维度关联起来
  - $A \to L], L \to id[num \mid L, id$

第二章 文法与语言设计

2.3.6 表达式优先级与结合性

- 考虑加（+）、乘（*）、幂（^）、负（−）、括号5 个操作
  - 不同优先级特性，优先级由高到低为：括号、幂、负、乘、加。
  - 涵盖左右结合，加、乘为左结合，幂、负为右结合。
  - 涵盖一元二元操作，加、乘、幂为二元操作符，负为一元操作符。
  - 可以通过括号改变计算次序。
- 二义文法
  - $E \rightarrow E + E \mid E * E \mid E \hat{} E \mid -E \mid (E) \mid i$
- 非二义文法
  - $E \rightarrow E + T \mid T$

2.3.6 表达式优先级与结合性

非二义文法

- $E \rightarrow E + T \mid T$
- $T \rightarrow T * F \mid F$
- $F \rightarrow -F \mid M$
- $M \rightarrow P \hat{} M \mid P$
- $P \rightarrow (E) \mid i$

表达式设计要点

- 优先级越低，越接近开始符号；优先级越高，越远离开始符号。
- 相同优先级的算符，使用同一个左部符号，不同优先级使用不同左部符号。
- 在规范归约下，左结合的运算符使用左递归。
- 在规范归约下，右结合的运算符使用右递归。
- 一元运算符左部、右部使用同一符号。

第二章作业

【作业2-1】令文法为

$$E \rightarrow T|E+T|E-T$$

$$T \rightarrow F|T*F|T/F$$

$$F \rightarrow (E)|i$$

(1) 给出$i+i*i$、$i*(i+i)$的最左推导和最右推导。

(2) 给出$i+i+i$、$i-i*i$、$i-i-i$的语法树。

(3) 写出$i-i*i$的所有短语、直接短语和句柄

【作业2-2】证明下面的文法是二义的：$S \rightarrow iSeS|iS|i$

【作业2-3】把下面的文法改为无二义的：$S \rightarrow SS|(S)|()$

第二章 文法与语言设计

- 2.1 文法和语言
  - 2.1.1 基本概念
  - 2.1.2 文法
  - 2.1.3 推导和归约
  - 2.1.4 语言
  - 2.1.5 文法的Chomsky分类
- 2.2 语法树与二义文法
  - 2.2.1 短语和句柄
  - 2.2.2 语法树
  - 2.2.3 二义文法
- 2.3 程序语言设计
  - 2.3.1 正规式
  - 2.3.2 正规式的等价变换
  - 2.3.3 基本运算的文法设计
  - 2.3.4 连接-闭包和闭包-连接
  - 2.3.5 拆分括号对
  - 2.3.6 表达式优先级与结合性
- 2.4 文法的等价变换
  - 2.4.1 消除无用产生式
  - 2.4.2 消除单非产生式
  - 2.4.3 消除空符产生式

2.4 文法的等价变换

- 如果文法G_1和G_2满足$L(G_1) = L(G_2)$，则称G_1和G_2是等价的
  - 通过等价变换使文法满足某个文法的特殊要求
  - 通过等价变换使文法满足某个算法的要求

第二章 文法与语言设计

2.4.1 消除无用产生式

- 无用符号有两种
  - 从开始符号无法推出的符号，这类符号永远使用不到，但在文法中占用计算资源，造成文法复杂臃肿。
  - 无法推导出句子的符号，这种符号除上述危害外，还会导致错误的推导，直到推导到该符号，无法再往下进行，使得发现错误的时间延迟。
- 对文法$G=(V_N,V_T,P,S)$，对$\forall X\in V_N\cup V_T$，若X同时满足如下两个条件则称符号X是有用的，否则称X是无用符号：
  - $X\overset{*}{\Rightarrow}\omega$，其中$\omega\in V_T^*$
  - $S\overset{*}{\Rightarrow}\alpha X\beta$
- 对文法$G=(V_N,V_T,P,S)$，若产生式$A\rightarrow\beta\in P$的左部或右部含有无用符号，则称此产生式为无用产生式。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.4.1 消除无用产生式

算法 2.1 消除无用产生式

输入: 文法 $G = (V_N, V_T, P, S)$

输出: 消除无用产生式后的文法 $G'' = (V_N'', V_T'', P'', S'')$

1 G' =removeEndlessSymbols(G);

2 G'' =removeStartlessSymbols(G');

郑艳伟，zhengyw@sdu.edu.cn

算法 2.2 消除不能推导出句子的无用符号和无用产生式

输入: 文法 $G = (V_N, V_T, P, S)$

输出: 消除不能推导出句子的无用符号和无用产生式后的文法 $G' = (V_N', V_T', P', S')$

```
function removeEndlessSymbols(G):
    V_T' = V_T, S' = S, V_N' = ∅, P' = ∅;
    do
        foreach A → x_1 x_2 ··· x_n ∈ P do
            isEndless = false;
            for i = 1 : n do
                if x_i ∉ V_N' ∪ V_T' ∪ {ε} then
                    isEndless = true;
                    break;
                end
            end
            if ¬isEndless then
                V_N' ∪= {A};
                P' ∪= {A → x_1 x_2 ··· x_n};
                P −= {A → x_1 x_2 ··· x_n};
            end
        end
    while V_N' 有新元素加入;
    return G';
end removeEndlessSymbols
```

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.4.1 消除无用产生式

算法 2.3 消除开始符号不能到达的无用符号和无用产生式

输入: 文法 $G' = (V_N', V_T', P', S')$

输出: 消除开始符号不能到达的无用符号和无用产生式后的文法 $G'' = (V_N'', V_T'', P'', S'')$

function removeStartlessSymbols(G'):
    $S'' = S', V_N'' = \{S''\}, V_T'' = \emptyset, P'' = \emptyset$;
    do
        foreach $A \to x_1x_2\cdots x_n \in P'$ do
            if $A \notin V_N''$ then continue;
            for $i = 1 : n$ do
                if $x_i \in V_N'$ then $V_N'' \cup= \{x_i\}$;
                if $x_i \in V_T'$ then $V_T'' \cup= \{x_i\}$;
            end
            $P'' \cup= \{A \to x_1x_2...x_n\}$;
            $P' -= \{A \to x_1x_2...x_n\}$;
        end
    while $V_N'' \cup V_T''$ 有新元素加入;
    return G'';
end removeStartlessSymbols

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.4.1 消除无用产生式

- 例：消除无用符号和无用产生式：$G[S]=(\{S,T,Q,R\},\{0,1\},P,S)$，其中$P$如下所示
  - 施行算法2.2，消除不能推导出句子的无用符号和无用产生式

$S\rightarrow 0S$
$S\rightarrow T$
$S\rightarrow R$
$T\rightarrow 0$
$Q\rightarrow 1Q$
$Q\rightarrow 0$
$R\rightarrow 1R$

V_N'	P'
T	$T\rightarrow 0$
Q	$Q\rightarrow 0$
S	$S\rightarrow T$
	$Q\rightarrow 1Q$
	$S\rightarrow 0S$

$$V_N'=\{S,T,Q\},P'=\{S\rightarrow 0S,S\rightarrow T,T\rightarrow 0,Q\rightarrow 1Q,Q\rightarrow 0\}$$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.4.1 消除无用产生式

- 例：消除无用符号和无用产生式：$G[S]=(\{S,T,Q,R\},\{0,1\},P,S)$，其中$P$如下所示
  - 施行算法2.3，消除开始符号不能到达的无用符号和无用产生式

$S\rightarrow 0S$
$S\rightarrow T$
$T\rightarrow 0$
$Q\rightarrow 1Q$
$Q\rightarrow 0$

V_N''	V_T''	P'
S	0	$S\rightarrow 0S$
T		$S\rightarrow T$
		$T\rightarrow 0$

$$V_N''=\{S,T\},V_T''=\{0\},P''=\{S\rightarrow 0S,S\rightarrow T,T\rightarrow 0\}$$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

第二章 文法与语言设计

2.4.2 消除单非产生式

- 单非产生式，指形如$A \to B, A \in V_N, B \in V_N$的产生式。
  - 这种产生式容易在自动机转正规文法时产生，导致转换成的文法不再是正规文法，需要消除这种产生式。
  - 消除思路：把B的产生式左部替换为A，然后删除产生式$A \to B$

2.4.2 消除单非产生式

算法 2.4 消除单非产生式

输入: 文法 $G=(V_N, V_T, P, S)$

输出: 消除单非产生式后的文法 $G'=(V_N', V_T', P', S')$

$S'=S, V_T'=V_T, V_N'=\emptyset, P'=\emptyset$;

foreach $A \in V_N$ do

    $W(A)=\{A\}$;

end

do

    foreach $A \in V_N$ do

        if $\exists A \rightarrow B \wedge B \in V_N$ then

            $W(A) \cup = W(B)$;

        end

    end

while $\exists W(A)$ 增大;

foreach $A \in V_N$ do

    foreach $B \in W(A)$ do

        $P' \cup = \{A \rightarrow \alpha | B \rightarrow \alpha \wedge \alpha \notin V_N\}$

    end

end

foreach $A \rightarrow \alpha \in P'$ do

    $V_N' \cup = \{A\}$

end

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.4.2 消除单非产生式

- 例：消除单非产生式：$G[S]=(\{S,Q,R\},\{0,1\},P,S)$，其中$P$如下所示

$S\rightarrow QR$
$S\rightarrow Q$
$S\rightarrow R$
$Q\rightarrow 0Q$
$Q\rightarrow 1$
$R\rightarrow 1R$
$R\rightarrow 0$

- （1）求W

$W(S)=\{S,Q,R\}$
$W(Q)=\{Q\}$
$W(R)=\{R\}$

- （2）计算产生式集

$S\rightarrow QR$	$Q\rightarrow 0Q$	$R\rightarrow 1R$
$S\rightarrow 0Q$	$Q\rightarrow 1$	$R\rightarrow 0$
$S\rightarrow 1$		
$S\rightarrow 1R$		
$S\rightarrow 0$		

- （3）消除无用符号和无用产生式，略

第二章 文法与语言设计

2.4.3 消除空符产生式

- 空符产生式，指形如$A \to \varepsilon$的产生式。
  - 有的算法要求文法除开始符号外，不能有空符产生式。
- 任给一文法$G[S]$
  - 如果$\varepsilon \notin L(G)$，则可以消除所有空符产生式；
  - 如果$\varepsilon \in L(G)$，则消除所有空符产生式后，再增加一个空符产生式$S \to \varepsilon$，称为规范空符产生式。

2.4.3 消除空符产生式

算法 2.5 计算可推导出空符的非终结符集

输入: 文法 $G = (V_N, V_T, P, S)$

输出: 可推导出空符的非终结符集 W

```
function getEmptableVN(G):
    W = ∅;
    foreach A → β ∈ P do
        if β = ε then  W ∪= {A} ;
    end
    do
        foreach A → X1X2···Xn ∈ P do
            isEmptable = true;
            for i = 1 : n do
                if Xi ∉ W then
                    isEmptable = false;
                    break;
                end
            end
            if isEmptable then  W ∪= {A} ;
        end
    while W 增大;
    return W;
end getEmptableVN
```

2.4.3 消除空符产生式

算法 2.6 消除空符产生式

输入: 文法 $G=(V_N,V_T,P,S)$

输出: 消除空符产生式后的文法 $G'=(V_N,V_T,P',S)$

function removeEmptyProduction(G,W):
    $P'=\emptyset$;
    foreach $A\to X_1X_2\cdots X_n\in P$ do
        derive($P',A,W,\varepsilon,X_1X_2\cdots X_n$);
    end
    return $G'=(V_N,V_T,P',S)$;
end removeEmptyProduction
function derive($\&P',A,W,left,right$):
    if $right=\varepsilon$ then
        if $left\neq\varepsilon$ then
            $P'\cup=\{A\to left\}$;
        end
    else
        derive($P',A,W,left+right[1],right[2:end]$);
        if $right[1]\in W$ then
            derive($P',A,W,left,right[2:end]$);
        end
    end
end derive

郑艳伟，zhengyw@sdu.edu.cn

2.4.3 消除空符产生式

算法 2.7 消除或规范空符产生式

输入: 文法 $G=(V_N,V_T,P,S)$

输出: 消除或规范空符产生式后的文法 $G'=(V_N,V_T,P',S)$

```
W =getEmptableVN(G);
G' =removeEmptyProduction(G, W);
P' -= {S → S};
if S ∈ W then
    P' ∪= {S → ε};
end
```

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.4.3 消除空符产生式

- 例：消除$G[S]$的空符产生式，其中P如下所示

$S \to AS$
$S \to AB$
$S \to \varepsilon$
$A \to a$
$A \to \varepsilon$
$B \to b$
$B \to AS$

- （1）求W $W = \{S, A, B\}$
- （2）消除空符产生式

$S \to AS, S \to S, S \to A, S \to \varepsilon$
$S \to AB, S \to A, S \to B, S \to \varepsilon$
$S \to \varepsilon$
$A \to a$
$A \to \varepsilon$
$B \to b$
$B \to AS, B \to A, B \to S, B \to \varepsilon$

- （3）求并集，去掉$S \to S$和空符产生式，加入$S \to \varepsilon$，得到：

$S \to AS \mid AB \mid A \mid B \mid \varepsilon, \quad A \to a, \quad B \to b \mid AS \mid S \mid A$

郑艳伟，zhengyw@sdu.edu.cn

第 2 章 文法与语言设计 内容小结

- 一个文法是一个四元组,包括非终结符集、终结符集、产生式集和开始符号。
- 用产生式右部替换左部符号的过程，称为推导，其逆过程称为归约。
- 最右推导称为规范推导，其逆过程最左归约称为规范归约。
- 开始符号能推导出的句子全体,称为语言。
- 短语是某个上下文中的一个语法单位，最左直接短语称为句柄。
- 语法树任意子树叶结点构成短语，二层子树叶结点构成直接短语，最左二层子树叶结点构成句柄。
- 如果文法中存在某个句型对应两棵不同语法树，则这个文法为二义文法。
- 文法的等价变换包括消除无用产生式、单非产生式和空符产生式。

第二章 文法与语言设计

The End

谢谢

授　课　教　师 ：郑艳伟
手　　　　　机 ：18614002860（微信同号）
邮　　　　　箱 ：zhengyw@sdu.edu.cn