山东大学
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

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 语言
  - 2.1.4 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 文法的等价变换
- 2.4 语言设计
  - 2.4.1 声明语句设计
  - 2.4.2 表达式与赋值语句
  - 2.4.3 控制语句设计
  - 2.4.4 程序单元设计
  - 2.4.5 程序设计

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 语言
  - 2.1.4 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 文法的等价变换
- 2.4 语言设计
  - 2.4.1 声明语句设计
  - 2.4.2 表达式与赋值语句
  - 2.4.3 控制语句设计
  - 2.4.4 程序单元设计
  - 2.4.5 程序设计

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

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。
- 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$\Sigma^* = \{\varepsilon, a, b, aa, ab, bb, aaa, \ldots\}$。
- 用ϕ表示不含任何元素的空集{}。
  - 注意ε、{}、{ε}的区别。

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

- 2.1 文法和语言
  - 2.1.1 基本概念
  - 2.1.2 文法
  - 2.1.3 语言
  - 2.1.4 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 文法的等价变换
- 2.4 语言设计
  - 2.4.1 声明语句设计
  - 2.4.2 表达式与赋值语句
  - 2.4.3 控制语句设计
  - 2.4.4 程序单元设计
  - 2.4.5 程序设计

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>.

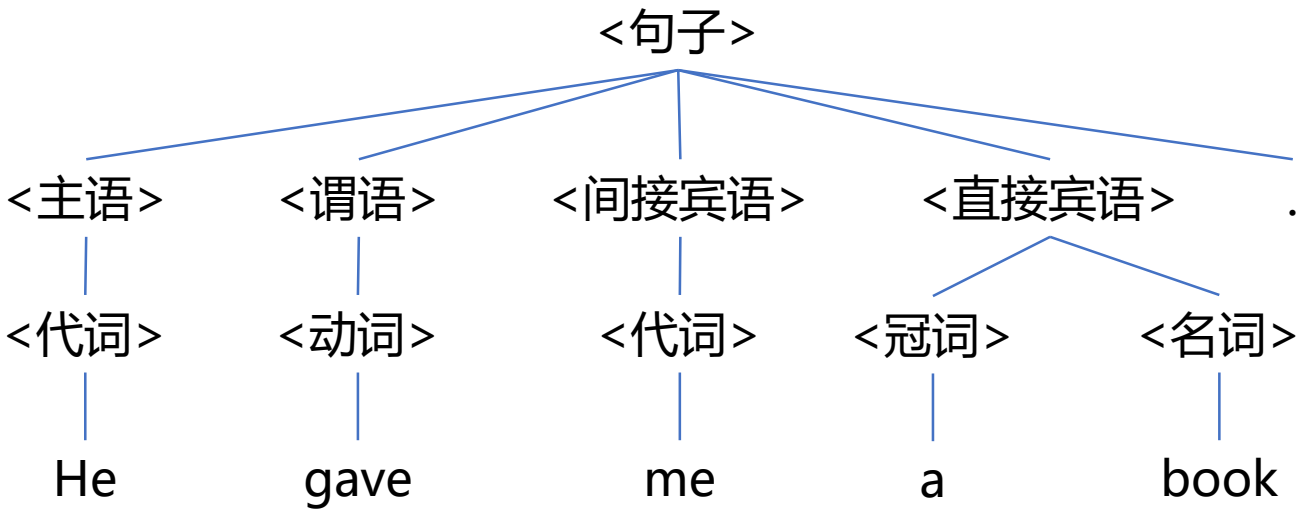
⇒ He gave me <冠词> <名词>.

⇒ He gave me a <名词>.

⇒ He gave me a book.

- 显然也推导出句子：me gave He a 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
  - $$\left.\begin{array}{l} A \to \alpha_1 \\ A \to \alpha_2 \\ \quad \ldots\ldots \\ A \to \alpha_n \end{array}\right\} \Leftrightarrow A \to \alpha_1 | \alpha_2 | \ldots | \alpha_n$$
- 箭头→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2.3】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：$\alpha \to \beta$
  - 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$\alpha \to \beta$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2.2】递归产生式定义加乘算术表达式G[E]:

$E \to i$

$E \to E + E$

$E \to E * E$

$E \to (E)$

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

上下文无关文法

- 上下文无关文法：它所定义的文法范畴（或语法单位）是完全独立于这种范畴可能出现的环境的。
  - 当碰到一个算术表达式，我们完全可以对它“就事论事”的进行处理，而不必考虑它的上下文。
  - 在自然语言中，一个句子、一个词乃至一个字，它的语法性质和所处的上下文往往都有密切关系。

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

推导

- 【例2.4】有如下文法：$E \to E+E \mid E*E \mid (E) \mid i$
  - 推导：$E \Rightarrow (E) \Rightarrow (E+E) \Rightarrow (i+E) \Rightarrow (i+i)$
  - 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$A \to \gamma$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：$E \to E+E \mid E*E \mid (E) \mid i$
  - $E \Rightarrow E+E \Rightarrow i+E \Rightarrow i+i$
  - $E \Rightarrow E+E \Rightarrow E+i \Rightarrow i+i$
- 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

- 2.1 文法和语言
  - 2.1.1 基本概念
  - 2.1.2 文法
  - 2.1.3 语言
  - 2.1.4 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 文法的等价变换
- 2.4 语言设计
  - 2.4.1 声明语句设计
  - 2.4.2 表达式与赋值语句
  - 2.4.3 控制语句设计
  - 2.4.4 程序单元设计
  - 2.4.5 程序设计

山东大学 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2.5】有文法G：$E \to E + E \mid E * E \mid (E) \mid i$
  - $(i * i + i)$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2.6】有文法$G_1 = (V_N, V_T, P, S)$，其中：

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

$P = \{< 数字串 > \rightarrow < 数字串 >< 数字 > | < 数字 >,$

$< 数字 > \rightarrow 0|1|2|3|4|5|6|7|8|9\};$

$S = \{< 数字串 >\}$

确定G_1对应的语言。

- 【分析】由< 数字串 >→< 数字 >，可得数字串为0~9的任意数字

每次用< 数字串 >→< 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 >→< 数字 >推导为止。

- 【结论】$L(G_1)$表示十进制非负整数。

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

语言

- 【例2.7】有文法$G_2[S]: S \to bA, A \to aA|a$，确定$G_2$对应的语言。
  - $S \Rightarrow bA \Rightarrow ba$
  - $S \Rightarrow bA \Rightarrow baA \Rightarrow baa$
  - $S \Rightarrow bA \Rightarrow baA \Rightarrow baaA \Rightarrow baaa$
  - …
  - $S \Rightarrow bA \Rightarrow baA \Rightarrow \cdots \Rightarrow ba \ldots a$
  - 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2.8】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。
  - $S \Rightarrow AB \Rightarrow ab$
  - $S \Rightarrow AB \Rightarrow aAB \Rightarrow aaB \Rightarrow aab$
  - …
  - $S \Rightarrow AB \Rightarrow AB \Rightarrow aB \Rightarrow abB \Rightarrow abb$
  - $S \Rightarrow AB \Rightarrow AB \Rightarrow aB \Rightarrow abB \Rightarrow abbB \Rightarrow abbb$
  - …
  - $S \Rightarrow AB \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow a \ldots aB \Rightarrow a \ldots abB \Rightarrow a \ldots ab \ldots b$
  - 归纳得：$L(G_3) = \{a^m b^n | m \geq 1, n \geq 1\}$

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

语言

- 【例2.9】构造文法：$L(G_4) = \{a^n b^n | n \geq 1\}$。
  - ab
  - $aabb$
  - $aaabbb$
  - …
  - 从另一个角度看：
  - ab
  - $aabb$
  - $aaabbb$
  - …
  - 归纳得：$G_4[S]: S \rightarrow aSb|ab$

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

语言

- 【例2.10】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 语言
  - 2.1.4 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 文法的等价变换
- 2.4 语言设计
  - 2.4.1 声明语句设计
  - 2.4.2 表达式与赋值语句
  - 2.4.3 控制语句设计
  - 2.4.4 程序单元设计
  - 2.4.5 程序设计

2.1.4 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。
- 0型文法：我们说文法$G = (V_N, V_T, P, S)$是一个0型文法，如果它的每个产生式$\alpha \to \beta$满足：$\alpha \in (V_N \cup V_T)^* V_N (V_N \cup V_T)^*$，$\beta \in (V_N \cup V_T)^*$。
  - 0型文法也称短语文法。
  - 0型文法的能力相当于图灵(Turing)机，或者说任何0型语言都是递归可枚举的；反之，递归可枚举必定是一个0型语言。
- 图灵机相关知识参考
  - 对一个句子，可以做出某语言接受或拒绝该句子的判断，这个语言称为图灵可判断语言。
  - 除了接受、拒绝，如果还存在不停机可能，称为图灵可识别语言。
  - 图灵可识别语言等价于递归可枚举语言，同时也被认为是半可判定的，它是可以被图灵机识别的。

2.1.4 文法的Chomsky分类

- 0型文法：我们说文法$G = (V_N, V_T, P, S)$是一个0型文法，如果它的每个产生式$\alpha \to \beta$满足：$\alpha \in (V_N \cup V_T)^* V_N (V_N \cup V_T)^*$，$\beta \in (V_N \cup V_T)^*$。
- 1型文法：在满足0型文法基础上，除$S \to \varepsilon$外，每个产生式$\alpha \to \beta$满足$|\alpha| \leq |\beta|$，且S不能出现在任何产生式的右部。
  - 1型文法也称上下文有关文法，即对非终结符号进行替换时必须考虑上下文，并且一般不允许替换成空串ε。
  - 例如，假如$\alpha A \beta \to \alpha \gamma \beta$是1型文法的一个产生式，$\alpha$和$\beta$均不空，则非终结符号$A$只有在$\alpha$和$\beta$这个上下文环境中才能替换为$\gamma$。

2.1.4 文法的Chomsky分类

- 0型文法：我们说文法$G = (V_N, V_T, P, S)$是一个0型文法，如果它的每个产生式$\alpha \to \beta$满足：$\alpha \in (V_N \cup V_T)^* V_N (V_N \cup V_T)^*$，$\beta \in (V_N \cup V_T)^*$。
- 2型文法：在满足0型文法基础上，每个产生式满足：$A \to \beta, A \in V_N, \beta \in (V_N \cup V_T)^*$。
  - 2型文法也称上下文无关文法。
  - 上下文无关文法对应下推自动机，使用下推表（先进后出栈）的有限自动机是分析上下文无关文法的基本手段。

2.1.4 文法的Chomsky分类

- 0型文法：我们说文法$G = (V_N, V_T, P, S)$是一个0型文法，如果它的每个产生式$\alpha \to \beta$满足：$\alpha \in (V_N \cup V_T)^* V_N (V_N \cup V_T)^*$，$\beta \in (V_N \cup V_T)^*$。
- 3型文法：在满足0型文法基础上，每个产生式满足：$A \to aB$或$A \to b$，其中，$a \in V_T, b \in V_T \cup \{\varepsilon\}, A \in V_N, B \in V_N$。
  - 3型文法也称右线性文法。
  - 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，$a \in V_T, b \in V_T \cup \{\varepsilon\}, A \in V_N, B \in V_N$。
  - 3型文法等价于正规式，所以也成为正规文法。
  - 正规式就是正则表达式，英文为Regular Expression；正规文法也称为正则文法，英文为Regular Grammar。

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

几个有趣的结论

- 正规文法(3)不能产生语言$L(G)=\{a^nb^n|n\geq 1\}$，上下文无关文法(2)则可以：$S\rightarrow aSb|ab$。
- $L(G)=\{a^nb^nc^i|i\geq 1,n\geq 1\}$是一个上下文无关语言：$S\rightarrow AB, A\rightarrow aAb|ab,\ B\rightarrow Bc|c$。
- 语言$L(G)=\{a^nb^nc^n|n\geq 1\}$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 语言
  - 2.1.4 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 文法的等价变换
- 2.4 语言设计
  - 2.4.1 声明语句设计
  - 2.4.2 表达式与赋值语句
  - 2.4.3 控制语句设计
  - 2.4.4 程序单元设计
  - 2.4.5 程序设计

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5.2】对文法$G[E]$

$$E \to T|E+T$$

$$T \to F|T*F$$

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

- 句型$i_1 * i_2 + i_3$
  - $E \Rightarrow E+T \Rightarrow E+F \Rightarrow E+i_3 \Rightarrow T+i_3 \Rightarrow F+i_3 \Rightarrow i_2+i_3$
  - 但$i_2 + i_3$不是该句型的一个短语，因为$E \not\Rightarrow i_1 * E$。
- 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。

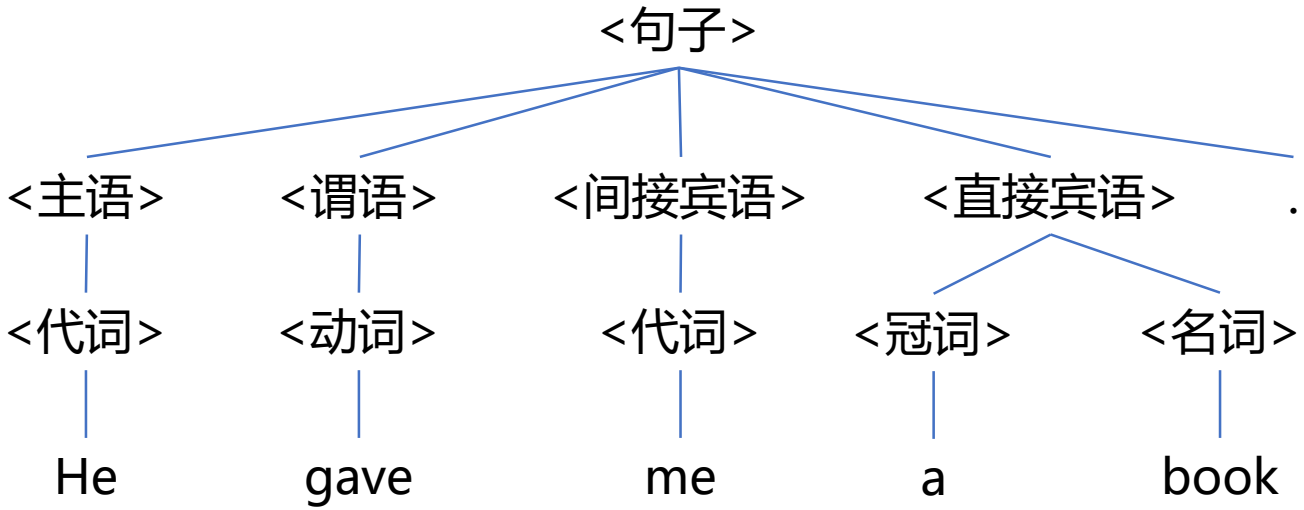
第二章 文法与语言设计

- 2.1 文法和语言
  - 2.1.1 基本概念
  - 2.1.2 文法
  - 2.1.3 语言
  - 2.1.4 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 文法的等价变换
- 2.4 语言设计
  - 2.4.1 声明语句设计
  - 2.4.2 表达式与赋值语句
  - 2.4.3 控制语句设计
  - 2.4.4 程序单元设计
  - 2.4.5 程序设计

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

2.2.2 语法树

- 语法分析树，简称语法树(Syntax Tree)，即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。
  - 如果坚持使用最左（最右）推导，那么一棵语法树就完全等价于一个最左（最右）推导。



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

2.2.2 语法树

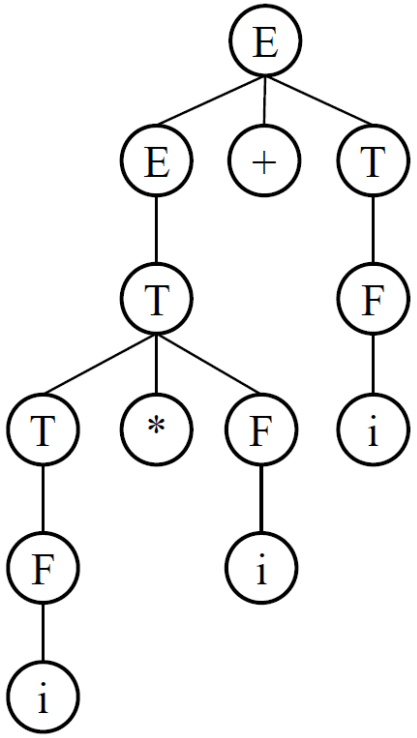
- 语法树根结点为开始符号，叶结点从左到右为所要推导的句型，内部结点是推导过程中用到的非终结符。
  - 对同一个句型，可能有不同的推导次序可以到达这个句型，语法树隐藏了替换次序的信息，表现了一个静态的推导结构。

【例5.2】对文法$G[E]$

$$E \to T|E+T$$

$$T \to F|T*F$$

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



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

2.2.2 语法树

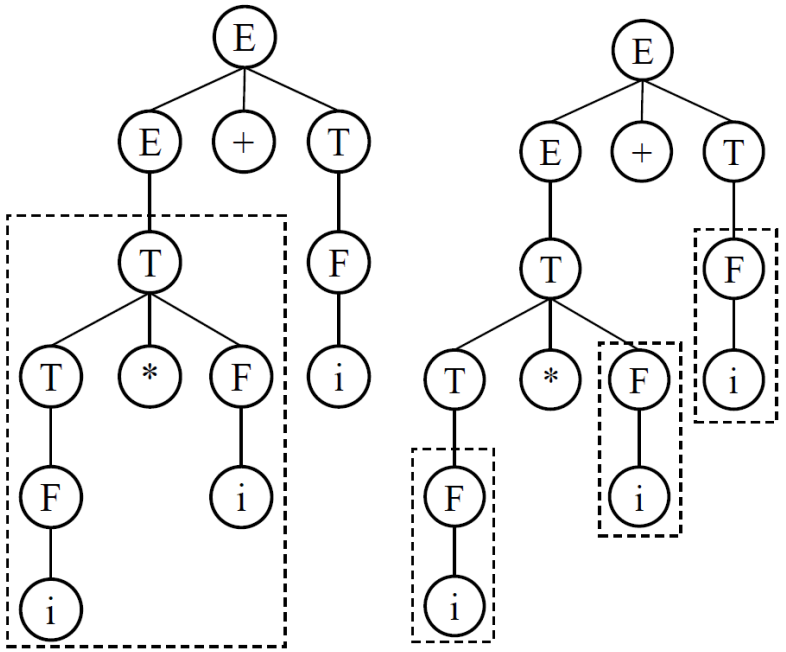
- 语法树的子树叶结点构成短语，二层子树叶结点构成直接短语，最左二层子树叶结点构成句柄。
  - 因为二层子树是针对这个句型的最后一步推导，因此是直接短语；
  - 最左二层子树父结点直接推出子节点，且最左，即对应最左直接短语；
  - 这一点其实对寻找句柄没有帮助，因为语法分析完成才能构造出语法树。

【例5.2】对文法$G[E]$

$$E \to T|E+T$$

$$T \to F|T*F$$

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



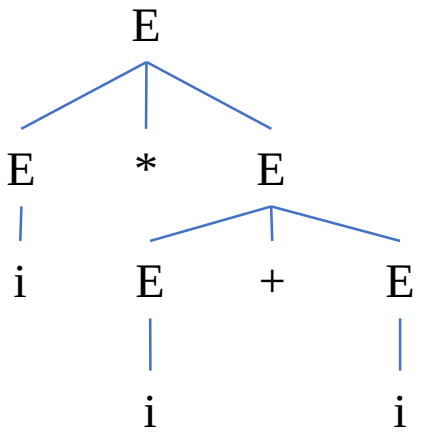
第二章 文法与语言设计

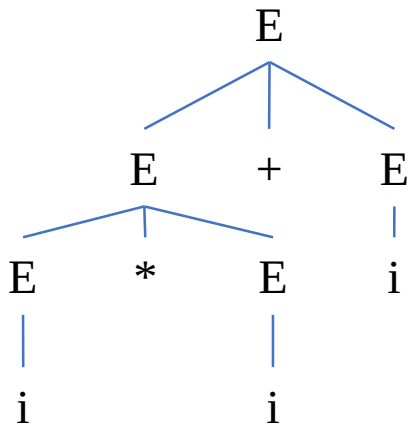
- 2.1 文法和语言
  - 2.1.1 基本概念
  - 2.1.2 文法
  - 2.1.3 语言
  - 2.1.4 文法的Chomsky分类
- 2.2 语法树与二义文法
  - 2.2.1 短语和句柄
  - 2.2.2 语法树
  - 2.2.3 二义文法
- 2.3 文法的等价变换
- 2.4 语言设计
  - 2.4.1 声明语句设计
  - 2.4.2 表达式与赋值语句
  - 2.4.3 控制语句设计
  - 2.4.4 程序单元设计
  - 2.4.5 程序设计

山东大学 SHANDONG UNIVERSITY 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子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2.11】$E \to E+E \mid E*E \mid (E) \mid i$，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

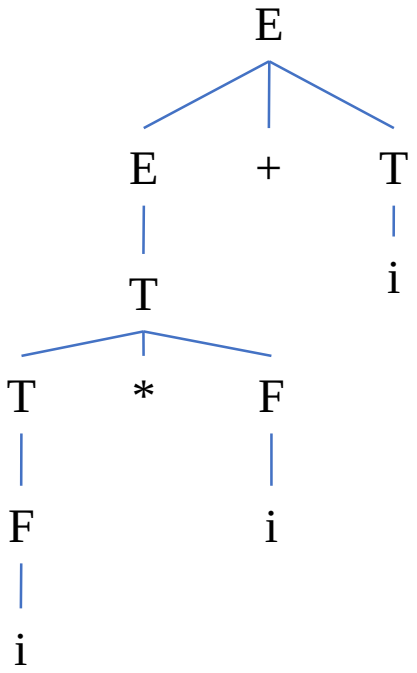
- 文法的二义性和语言的二义性是不同的概念
  - 可能有两个不同的文法G和G'，其中一个是二义的而另一个是无二义的，但是有$L(G)=L(G')$；
  - 对程序设计语言来说，常常希望它的文法是无二义的，因为我们希望对它每个语句的分析是唯一的；
  - 但是，只要能控制和驾驭文法的二义性，有时候存在二义性并不一定是坏事；
  - 目前已经证明，二义性问题是不可判定的，即不存在一个算法，它能在有限步骤内确切的判定一个文法是否为二义性的。

2.2.3 二义文法

- 【例2.12】$E \to E+E \mid E*E \mid (E) \mid i$，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】 $E \to T|E+T,\ T \to F|T*F, F \to (E)|i$ (优先级越高越远离开始符号)

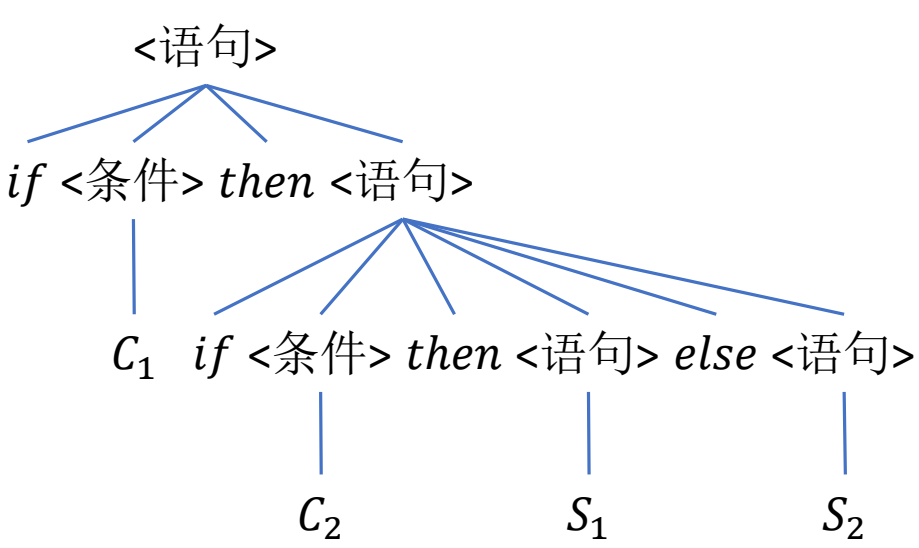
- $E \Rightarrow E+T \Rightarrow E+i \Rightarrow T+i \Rightarrow T*F+i \Rightarrow T*i+i \Rightarrow F*i+i \Rightarrow i*i+i$

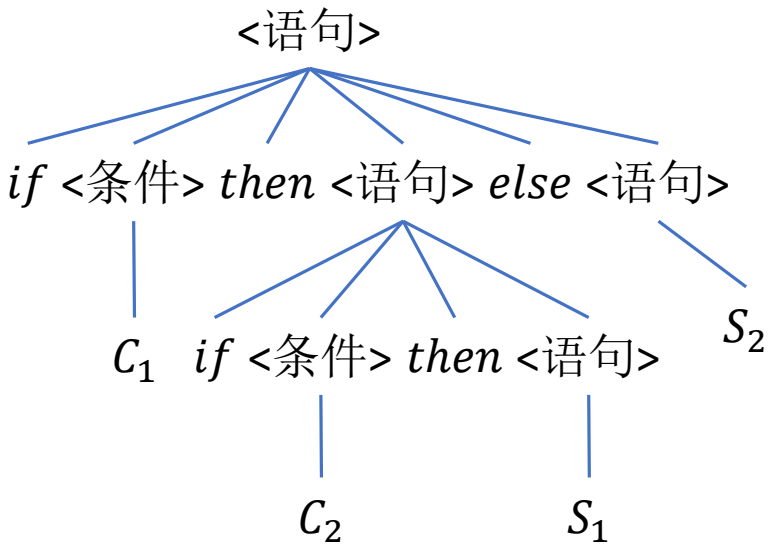


山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

几个有趣的结论

- 上下文无关文法表示条件语句：
  - $< 语句 > \rightarrow if < 条件 > then < 语句 >$
    $| if < 条件 > then < 语句 > else < 语句 >$
  - 这是个二义文法，如句子：$if\ C_1\ then\ if\ C_2\ then\ S_1\ else\ S_2$





山东大学 SHANDONG UNIVERSITY 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：
  - $< 语句 > \rightarrow if < 条件 > then < 语句 >$
    $| if < 条件 > then < 语句 > else < 语句 >$
  - 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
  - $< 语句 > \rightarrow < 匹配句 > | < 非匹配句 >$
  - $< 匹配句 > \rightarrow if < 条件 > then < 匹配句 > else < 匹配句 >$
    $| < 其它语句 >$
  - $< 非匹配句 > \rightarrow if < 条件 > then < 语句 >$
    $| if < 条件 > then < 匹配句 > else < 非匹配句 >$

山东大学 SHANDONG UNIVERSITY 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$P \to P$，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

- 2.1 文法和语言
  - 2.1.1 基本概念
  - 2.1.2 文法
  - 2.1.3 语言
  - 2.1.4 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 文法的等价变换
- 2.4 语言设计
  - 2.4.1 声明语句设计
  - 2.4.2 表达式与赋值语句
  - 2.4.3 控制语句设计
  - 2.4.4 程序单元设计
  - 2.4.5 程序设计

2.4 语言设计

H语言（Heterogeneous Language）

- 语言核心是C语言的精简，主要是为了覆盖编译原理和编译器技术的核心概念。
- 有些其它语言比较重要的特征，也吸纳进了这个语言。

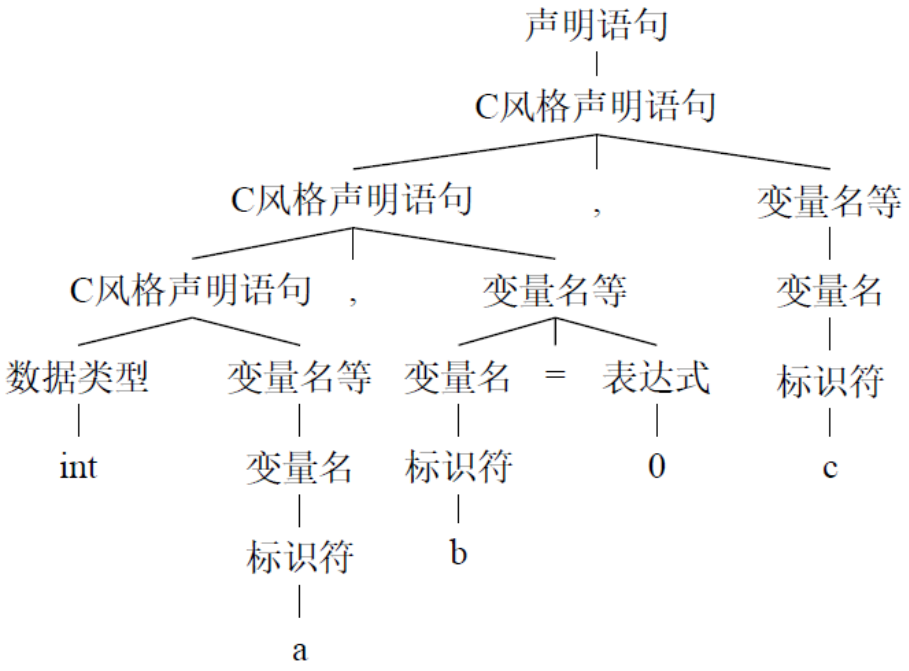
2.4.1 声明语句设计

```
1  <声明语句> → <C风格声明语句> | <Pascal风格声明语句>
2  <C风格声明语句> → <C风格声明语句>, <变量名等> | <数据类型><变量名等>
3  <Pascal风格声明语句> → <变量名等><Pascal右侧变量表>
4  <Pascal右侧变量表> → ,<变量名等><Pascal右侧变量表> | :<数据类型>
5  <变量名等> → <变量名>=<表达式> | <变量名>
6  <变量名> → <标识符> | <标识符>[<上界列表>]
7  <上界列表> → <上界列表>, <常数> | <常数>
8  <表达式> → <算术表达式> | <布尔表达式> | <字符表达式>
9  <数据类型> → byte | ubyte | char | bool | short | ushort | int | uint | float | double |
       real
```

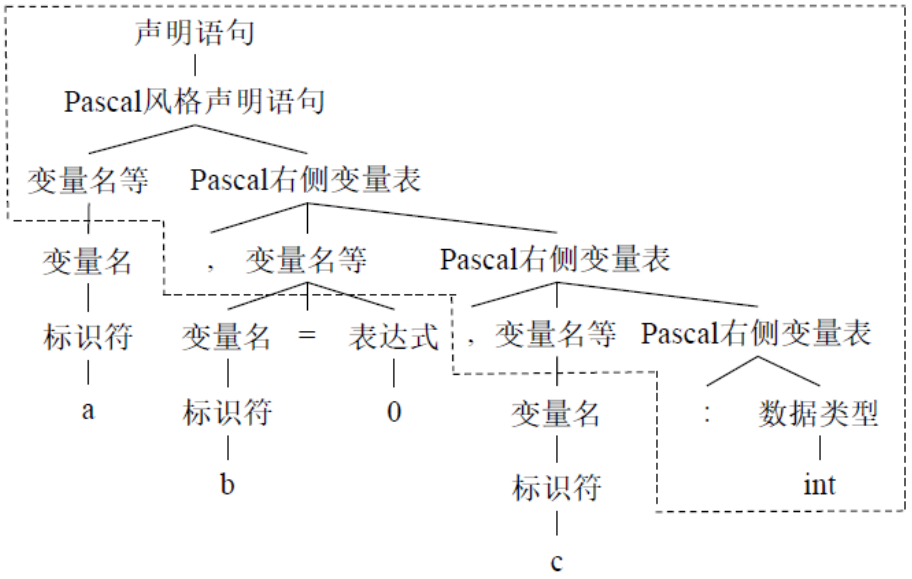
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

2.4.1 声明语句设计

- 根据上述产生式，分别给出声明语句int a, b=0, c和a, b=0, c : int的语法树，<表达式>部分可以直达叶结点。



(a) C 风格声明语句



(b) Pascal 风格声明语句

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

H语言二义和非二义算术表达式

```
1  <数字表达式> → <算术表达式>
2  <算术表达式> → <算术表达式><算术运算符><算术表达式>
3                | (<算术表达式>) | <运算量>
4  <运算量> → <变量> | <常量> | <函数调用>
5  <变量> → <标识符> | <标识符>[<下标列表>]
6  <下标列表> → <下标列表>, <算术表达式> | <算术表达式>
7  <算术运算符> → + | - | * | / | % | **
```

```
1  <数字表达式> → <算术表达式>
2  <算术表达式> → <算术表达式> + <加减项> | <算术表达式> - <加减项> | <加减项>
3  <加减项> → <加减项> * <乘除项> | <加减项> / <乘除项> | <加减项> % <乘除项> | <乘除项>
4  <乘除项> → +<正负项> | -<正负项> | <正负项>
5  <正负项> → <幂项> ** <正负项> | <幂项>
6  <幂项> → (<算术表达式>) | <运算量>
```

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

H语言二义和非二义布尔表达式

```
1  <数字表达式> → <布尔表达式>
2  <布尔表达式> → ¬<布尔表达式>
3               | <布尔表达式> ∧ <布尔表达式>
4               | <布尔表达式> ∨ <布尔表达式>
5               | (<布尔表达式>)
6               | <关系表达式>
7               | <布尔常量>
8               | <算术表达式>
9  <关系表达式> → <算术表达式><关系运算符><算术表达式>
10 <关系运算符> → < | <= | > | >= | == | !=
11 <布尔常量> → true | false
```

```
1  <数字表达式> → <布尔表达式>
2  <布尔表达式> → <布尔表达式> ∨ <或项> | <或项>
3  <或项> → <或项> ∧ <与项> | <与项>
4  <与项> → ¬<非项> | <非项>
5  <非项> → (<布尔表达式>) | <关系表达式> | <布尔常量> | <算术表达式>
```

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

H语言二义位运算表达式

```
1  <算术表达式> → ~<算术表达式>
2              | <算术表达式> << <算术表达式>
3              | <算术表达式> >> <算术表达式>
4              | <算术表达式> & <算术表达式>
5              | <算术表达式> ^ <算术表达式>
6              | <算术表达式> \| <算术表达式>
7              | (<算术表达式>)
8              | <算术表达式>
```

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

H语言非二义完整表达式

```
1  <表达式> → <子表达式> ? <子表达式> : <子表达式> // 三目运算
2           | <子表达式>
3  <子表达式> → <子表达式> ∨ <或项> | <或项> // 布尔或
4  <或项> → <或项> ∧ <与项> | <与项> // 布尔与
5  <与项> → ¬<非项> | <非项>  // 布尔非
6  <非项> → <非项> \| <位或项> // 按位或
7  <位或项> → <位或项> ^ <位异或项> | <位异或项> // 按位异或
8  <位异或项> → <位异或项> & <位与项> | <位与项> // 按位与
9  <位与项> → <关系项><关系运算符><关系项> // 关系运算
10          | <关系项>
11 <关系项> → <关系项> << <位移项> | <关系项> >> <位移项> | <位移项> // 移位操作
12 <位移项> → <位移项> + <加减项> | <位移项> – <加减项> | <加减项> // 加减法
13 <加减项> → <加减项> * <乘除项> | <加减项> / <乘除项>
14          | <加减项> % <乘除项> | <乘除项> // 乘除余
15 <乘除项> → –<负项> | <负项> // 负号
16 <负项> → <负项> ** <幂项> | <幂项> // 幂运算
17 <幂项> → ~<位反项> | <位反项> // 按位取反
18 <位反项> → (<表达式>) | <运算量> // 返回顶层
```

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

H语言赋值语句

1 <赋值语句> → <变量>=<表达式>

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

H语言语句列表

```
1   <语句列表> → <语句列表><带分号语句>;
2           | <语句列表><控制语句>
3           | ε
4   <带分号语句> → <声明语句> | <赋值语句>
5   <控制语句> → <if语句>
6           | <while语句>
7           | <for语句>
8           | <goto语句>
9           | <switch语句>
10  // 为后续应用做定义
11  <单行语句> → <声明语句>; | <赋值语句>; | <控制语句>
12  <语句块> → {<语句列表>} | <单行语句>
13  <语句块2> → {<语句列表>} | <语句列表>
```

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

H语言控制语句

```
1  <if语句> → if (<布尔表达式>) <语句块>
2          | if (<逻辑表达式>) <语句块> else <语句块>
3  <while语句> → while (<布尔表达式>) <语句块>
4  <for语句> → for (<标识符> = <算术表达式> : <算术表达式>) <语句块>
5          | for (<标识符> = <算术表达式> : <算术表达式> : <算术表达式>) <语句块>
6          | for (<for初始化>; <for判定>; <多赋值语句>) <语句块>
7  <for初始化> → <声明语句> | <多赋值语句> | ε
8  <for判定> → <布尔表达式> | ε
9  <多赋值语句> → <多赋值语句>,<赋值语句> | <赋值语句>
```

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

H语言Goto语句

```
1  <goto语句> → goto <标号>
2  <赋值语句> → <标号> <赋值语句>
3  <控制语句> → <标号> <控制语句>
4  <标号> → <标识符>:
```

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

H语言Switch语句

```
1  <switch语句> → switch (<布尔表达式>) {<case列表><default部分>}
2  <case列表> → <一个case><语句块2>
3  <一个case> → case c: | <case列表> case c:
4  <default部分> → default: <语句块2>
```

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

H语言过程定义与过程调用

```
1  <过程定义> → <返回类型><标识符>(<形参列表>){<语句列表>}
2          | <标识符>(<形参列表>): <返回类型>{<语句列表>}
3  <返回类型> → <数据类型> | void
4  <形参列表> → <形参列表2> | ε
5  <形参列表2> → <形参列表2>,<形参> | <形参>
6  <形参> → <数据类型><标识符>
```

```
1  <过程调用> → <标识符>(<实参列表>)
2  <实参列表> → <实参列表2> | ε
3  <实参列表2> → <标识符>,<实参列表2> | <标识符>
```

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

H语言程序设计

```
1  <程序> → <分程序表><分程序> | <分程序>
2  <分程序> → <声明语句>; | <过程定义>
```

第二章作业

【作业2-1】令文法为

$$E \to T|E+T|E-T$$

$$T \to F|T*F|T/F$$

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

(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。

【作业2-2】证明下面的文法是二义的：$S \to iSeS|iS|i$

【作业2-3】把下面的文法改为无二义的：$S \to SS|(S)|()$

山东大学
SHANDONG UNIVERSITY

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