第三章：语法分析

文法与语言
文法的分类
文法的相关概念

内容介绍

- 文法与语言
- 文法的相关概念
- 文法的分类

1.1 语言的语法

- 文法的产生：自然语言与程序设计语言的区别
- 语言的三个基本要素：语法、语义、语用

1.2 高级语言语法的表示形式

- BNF：

  语法成分::=组成部分

  赋值语句::=变量=表达式

  变量::=标识符|下标变量|指针变量...

  表达式::=算术表达式|逻辑表达式...

- 上下文无关文法

  产生式规则形容规则：

  P(语法符号)$\rightarrow X_1X_2...X_n$

  赋值语句$\rightarrow$变量=表达式

1.3 文法的定义

- **文法的定义**

一个文法G是一个四元组：G = (V_N, V_T, S, P) 其中：

- V_T *(Terminal Vocabulary)*是一个非空的有限集合，它的每个元素称为终极符号或终极符，一般用小写字母表示。 从语法分析的角度看，终极符号是一个语言不可再分的基本符号。
- V_N (*Nonterminal Vocabulary*) 是一个非空的有限集合，它的每个元素称为非终极符号、非终极符或中间符，一般用大写字母表示。非终极符是一个语法范畴，表示一类具有某种性质的符号。

注： 设V是文法G的符号集，则有：$V = V_N \cup V_T, V_N \cap V_T = \varnothing$

- S(*Start*)是特殊的非终极符号，称为文法的开始符号，$S \in V_N$ 。
- P (*production*)是产生式的有限集合。所谓的产生式，也称为产生规则（简称为规则），是按照一定格式书写的定义语法范畴的文法规则。

1.4 上下文无关文法

产生式的形式为： $A \rightarrow \alpha$的文法。其中：

① A：称为产生式的左部(头)， $A \in V_N$；

② α：称为产生式的右部（体）， $\alpha \in (V_T \cup V_N)^*$ ；

③ “→”或“::=”：读作“定义为”或“由…组成”；

④ 开始符号S必须至少在某个产生式的左部出现一次。

⑤为了书写方便，若干个左部相同的产生式，如 ：

$P \rightarrow \alpha 1$　　$P \rightarrow \alpha 2$　　……　　$P \rightarrow \alpha n$

可合并为一个，缩写为: $P \rightarrow \alpha 1 \mid \alpha 2 \mid \ldots \mid \alpha n$

其中，每个αi 称为 P 的一个候选式，符号“|”读作“或” 。

1.4 上下文无关文法

- 例子：
- $G=\{V_N, V_T, P, S\}$

  $V_N=\{N, D\}$ $V_T=\{0, 1, 2, \ldots, 9\}$

  $P=\{N\rightarrow D,\ N\rightarrow ND,\ D\rightarrow 0|1|\ldots|9\}$

  $S=N$

基本概念(1)

- 直接推导：如果$A \to \beta$是一个产生式，则有$\alpha_1 A \alpha_2 \Rightarrow \alpha_1 \beta \alpha_2$，其中$\Rightarrow$表示一步推导。这时称$\alpha_1 \beta \alpha_2$是由$\alpha_1 A \alpha_2$直接推导的，又称$\alpha_1 \beta \alpha_2$直接归约到 $\alpha_1 A \alpha_2$ 。

  $\Rightarrow$的含义是，使用一条规则，代替$\Rightarrow$左边的符号，产生$\Rightarrow$右端的符号串。

- $A \Rightarrow^+ \beta$：表示A通过一步或多步可推导出β
- $A \Rightarrow^* \beta$：表示A通过零步或多步可推导出β

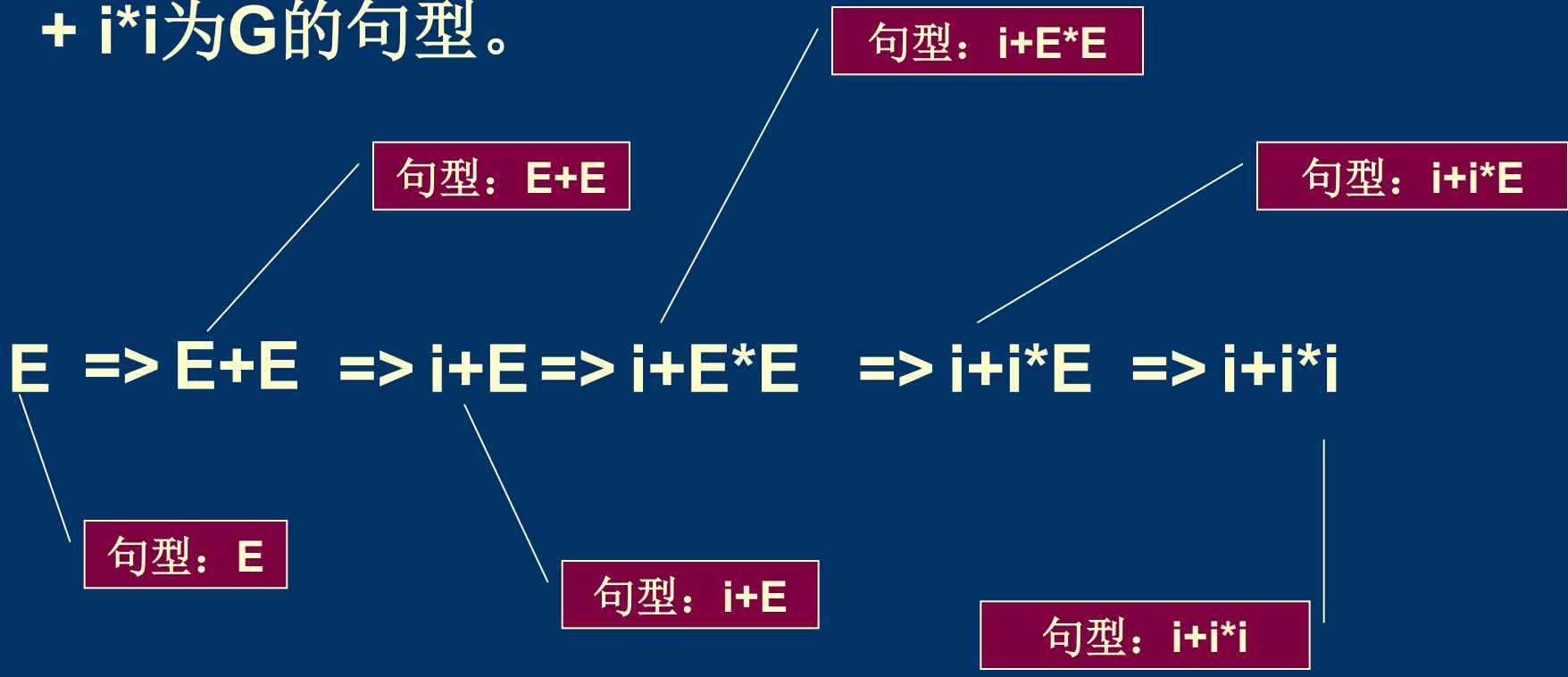
基本概念(1)

- 句型：设有文法G，如果有$S \Rightarrow^* \beta$，则称符号串β为G的句型 。我们用SF(G)表示文法G的所有句型的集合
- 句子：设β为文法G的一个句型，且β只包含终极符，则称β为G的句子
- 语言：$L(G)=\{ u| S \Rightarrow^+ u, u \in V_T^* \}$。 文法G所定义的语言是其所有句子的集合。

例：对于表达式文法G:

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

E 、 E+E 、 i+E 、 i + E*E 、 i + i*E 、 i + i*i为G的句型。



1.5 基本概念与高级语言语法

- 每一种高级程序设计语言都有自己的语法
- 符合高级语言文法的程序就是该语法的句子
- 所有程序组成的集合就构成了语言。

1.5 基本概念与高级语言文法

- PASCAL语言语法例子：

<程序> → <首部><声明部分><程序体>
<首部> → progarm（<运行环境>）
<声明部分> → var <标号声明><常量声明><类型声明><过程函数声明>
<程序体> → begin<语句序列>end
<语句序列> → <语句><语句序列>|<语句>
<语句> → <条件语句>|<循环语句>|<赋值语句>|<输入输出语句>|<过程调用语句>...

1.6 验证程序是不是语法的句子

- 最基本的方法：采用类搜索算法，从开始符出发进行推导
- 用剪枝来提高效率

两种基本思想

- 自顶向下的语法分析：从开始符出发，根据定义看是否可以推导出程序。
- 自底向上的语法分析：从程序出发，用规则的左部替换规则的右部，一直到规约成开始符。

基本概念(2)

- 短语

  设S是文法的开始符，有$S \Rightarrow^* \alpha_1 A \alpha_2$，如果有$A \Rightarrow^+ \beta$ 则称 β 是句型$\alpha_1 \beta \alpha_2$的一个短语。

- 简单短语

  设S是文法的开始符，有$S \Rightarrow^* \alpha_1 A \alpha_2$，如果有$A \Rightarrow \beta$ 则称 β 是句型$\alpha_1 \beta \alpha_2$的一个简单短语。

- 句柄

  句型中的最左简单短语称为句柄。

例：对于表达式文法G:

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

i*i 为句型 i + i*i 的一个短语。

因为：

$$E \Rightarrow E+E \Rightarrow i+E$$

$$i+E \Rightarrow i+E*E \Rightarrow i+i*E \Rightarrow i+i*i$$

思考题：i+i*i 亦为 句型 i + i*i 的一个短语。

例：对于表达式文法G:

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

i*i 不是句型 i + i*i 的简单短语。

因为：

$$E \Rightarrow E+E \Rightarrow i+E$$

$$E \Rightarrow E*E \Rightarrow i*E \Rightarrow i*i$$

例：设有文法 G[S]: S → cAdBf

A → ab | a

B → e |e B

考虑符号串$=cabdef.

因：S => cAdBf => cAdef

A=> ab

又因：S => cAdBf => cabdBf

B => e

所以：e，ab分别为cabdef的简单短语；

其中：ab为cabdef的句柄。

基本概念(3)

设有文法G

- 直接左递归：若有$E \rightarrow E\alpha$形式的规则，则称E是直接左递归。
- 直接右递归：若有$E \rightarrow \alpha E$形式的规则，则称E是直接右递归。
- 左递归：若有$E \Rightarrow^{+} E\alpha$，则称E是左递归。
- 右递归：若有$E \Rightarrow^{+} \alpha E$，则称E是右递归。
- 递归：若有$E \Rightarrow^{+} \alpha_1 E \alpha_2$

基本概念(3)

- **最左（右）推导：**

  如果进行推导时选择的是句型中的最左(右）非终极符，则称这种推导为最左(右)推导，并用符号 $\Rightarrow_{lm}$（$\Rightarrow_{rm}$）表示最左（右）推导。

- **左（右）句型：**

  用最左推导方式导出的句型，称为左句型，而用最右推导方式导出的句型，称为右句型(规范句型)。

- **结论：**

  每个句子都有相应的最右和最左推导（但对句型此结论不成立）

2. 文法分类

1956年，著名的语言学家乔姆斯基（Noam（Chomsky）将文法分为四类，四类文法对应四种类型的语言。

- 0型文法（短语型文法）

如果对文法G中的任一产生式$\alpha \rightarrow \beta$不加任何限制，即：$\alpha \in (V_T \cup V_N)^+$，并且至少含有一个非终极符，$\beta \in (V_T \cup V_N)^*$，则称G为0型文法或短语型文法。

2. 文法分类

- 1型文法（上下文相关文法 ）

  如果对文法G中的任一产生式均限制为形如：

  $\alpha A\beta \rightarrow \alpha\gamma\beta$

  其中$A \in V_N$，α，$\beta \in (V_T \cup V_N)^*$，

  $\gamma \in (V_T \cup V_N)^*$，

则称文法G为1型文法或上下文相关文法。

- 2 型文法（又称上下文无关文法context-free Grammer CFG）

  如果对文法G中的任一产生式均限制为形如：

  $A \rightarrow \alpha$

  其中$A \in V_N$, $\alpha \in (V_T \cup V_N)^*$，

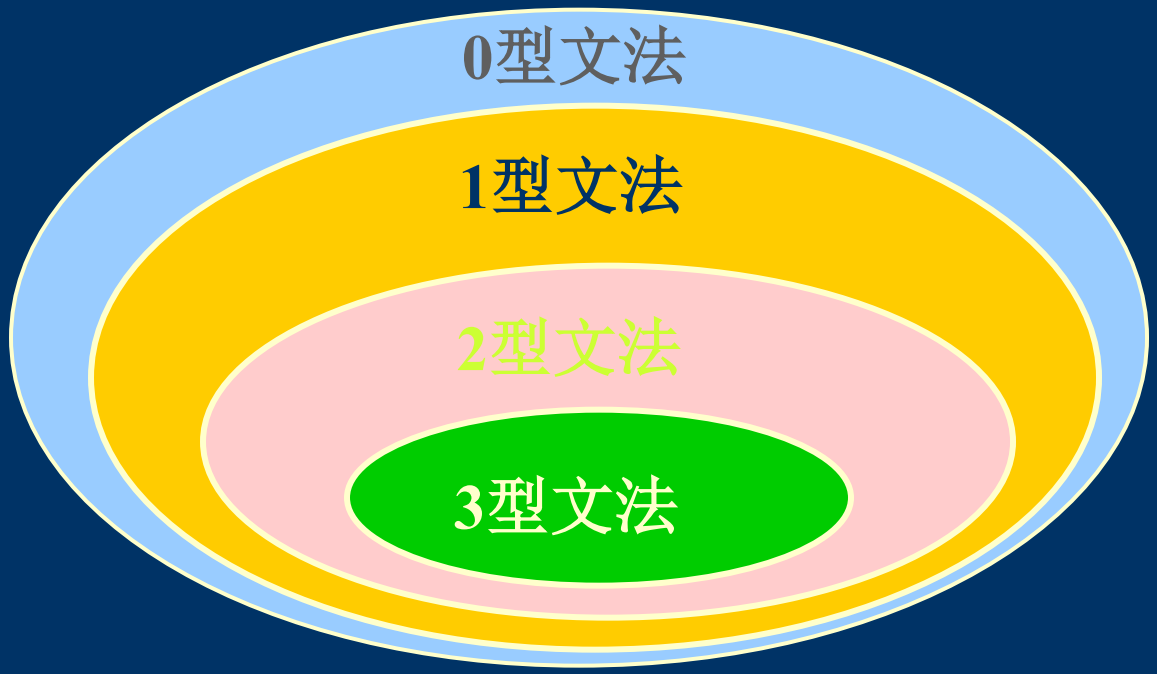
则称G为2型文法或上下文无关文法。

2. 文法分类

- 3型文法（又称线性文法、正则文法、正规文法）
  - 如果对文法G中的任一产生式均限制为形如：

    $A \rightarrow aB$ 或 $A \rightarrow a$

    其中： $A, B \in V_N$ ， $a \in V_T$ 则称文法G为3型文法。



0型文法　1型文法　2型文法　3型文法

产生式限制越来越严

描述功能越来越强

2. 文法分类

在编译技术中通常用3型文法来描述高级程序设计语言的词法部分，2型文法，或上下文无关文法描述的程序设计语言的语法结构，比如描述算术表达式，描述各种语句等等。

对“文法”一词如无特殊说明则均指上下文无关文法。

3 类型题

已知文法G的产生式规则如下：

E →E+T| T

T → T *F| F

F → （ E ）| i（标识符）

问：(T+i)*F+i 是不是文法G的句型，如果是，该句型有哪些短语，简单短语，句柄是哪个？

3 类型题

E⇒ E+T
⇒ T+T
⇒ T*F+T
⇒ F*F+T
⇒ (E) * F+T
⇒ (E + T) * F+T
⇒ (T + T) * F + T
⇒ (T + F) * F +T
⇒ (T + i) * F+ T
⇒ (T + i) * F + i

3 类型题

- 短语

(T+i)*F+i ，整个的肯定是短语由E推导出来

(T+i)*F 是由T推出来的

两个i ； T+i ； （T+i）；T;

- 简单短语

T 两个i

- 句柄

T