第2章 前后文无关文法和语言

- 主讲：王慧娇
- 办公室：金鸡岭3301-1
- 电话：13978321977
- QQ：248886622
- Email：whj7667@qq.com

- 答疑地点：5507
- 辅导时间：周三1、2节

本章主要内容

- 语言及文法概述
- 文法(Grammar)和语言的形式定义
- 句型分析（语法分析树、文法的二义性、短语和句柄）
- 文法的分类（正规文法、前后文无关文法）
- 文法的构造（文法表示语言）

问题：

- 1.如何确切地描述和定义一种程序设计语言？
- 2.如何识别和分析一种程序设计语言？

2.1 语言概述

- 什么是语言
  - 自然语言(Natural Language)
    - 是人与人的通讯工具
    - 语义(Semantics):环境、背景知识、语气、二义性———难以精确定义、难以形式化
  - 计算机语言(Computer Language)
    - 严格的语法(Grammar)、语义(Semantics) ———
    - 易于形式化，可以使用BNF范式定义语法
  - 语言是用来交换信息的工具———功能性描述

2.1 语言概述

- 语言——形式化的内容提取
  - 单词(Token)：满足一定规则字符(Character)串
  - 句子(Sentence)：满足一定规则单词序列
  - 语言(Language)：满足一定条件的句子集合
- 语言是字和组合字的规则——结构性描述
  - 例：一译开天第课今始编节上
  - 今天开始上第一节编译课

2.1 语言概述

- 语言描述形式
  - 集合法
    - 将语言中的语句全部枚举出来
    - 例：L={I am a teacher,You are my students}
  - 文法
    - 使用有限规则，产生描述语言的全部句子
    - 例：文法 $S \rightarrow aS|a$，则 $L=\{a^n|n>0\}$
  - 自动机
    - 一种算法或过程，识别某一字母表上的全部句子

2.2 符号和符号串

- 字母表（Alphabet）是一个非空有穷集合，字母表中的元素称为该字母表的一个字母（Letter），也叫字符（Character）
- 例 以下是不同的字母表

(1) {a，b，c，d}

(2) { a，b，c，……，z}

(3) {0，1}

- 相当于高级语言的字符集

2.2 符号和符号串

- 符号串

  (1) ε是字母表A上的一个符号串，叫做空串；

  (2) 若x是字母表A上的符号串，而a是A的元素，则xa是A上的符号串。

  (3) y是A上的符号串，当且仅当它由(1)和(2)导出。

- 换句话说：由字母表中的符号所组成的的任何有穷序列被称之为该字母表上的符号串

- 符号串的长度：是该符号串中的符号的数目。例如$|aab|=3$，$|\varepsilon|=0$。

2.2 符号和符号串

- 符号串的前缀、后缀及子串

前缀：移走s的尾部的零个或多于零个符号

后缀：删去s的头部的零个或多于零个符号

子串：从s中删去一个前缀和一个后缀之后的剩余部分

- 设s是符号串：s=abcd，则求符号串s的前缀、后缀及子串

2.2 符号和符号串

符号串的连接和幂

1.连接：设x和y是符号串，它们的连接xy是把y的符号写在x的符号之后得到的符号串。

例如， x=ba,y=nana, xy=banana, yx=nanaba

2.幂：$x^0=\varepsilon$; $x^1=x$; $x^2=xx$;

……;$x^n=x^{n-1}x$;

例如, x=ba,

$x^1=$ ba, $x^2=$baba, $x^3=$bababa,……$x^n=(ba)^n$

2.2 符号和符号串

- **符号串集合的和与积**
  - 设A，B为两个符号串集合，定义

    和 A+B（或$A\cup B$）$=\{w \mid w\in A$，或 $w \in B\}$

    积 A•B（或 AB）$= \{ xy \mid x \in A, y \in B\}$
- **用$\varnothing$表示空集**

  $A+\varnothing = \varnothing+A = A$

  $A\varnothing = \varnothing A = \varnothing$

  $\{\varepsilon\}A = A\{\varepsilon\} = A$

2.2 符号和符号串

- 符号串集合的方幂
  - 符号串集合A
  - $A^0=\{\varepsilon\}$, $A^1=A$, $A^2=AA$, $A^3=A^2A$, …, $A^n=A^{n-1}A=AA^{n-1}$，$n>0$
- 符号串集合的正闭包
  - $A^+=A^1 \cup A^2 \cup \cdots \cup A^n \cdots = \bigcup_{i=1}^{\infty} A^i$
- 符号串集合的自反传递闭包
  - $A^*=A^0 \cup A^1 \cup A^2 \cup \cdots \cup A^n \cdots = \bigcup_{i=0}^{\infty} A^i = \{\varepsilon\} \cup A^+$

2.2 符号和符号串

- 例：A={a，b，c}
  - A^1={a，b，c}
  - A^2={aa, ab, ac, ba,bb,bc,ca,cb,cc}
  - A^3={aaa，aab，aac，aad，aba，abb，abc…}
  - ……
  - A^+={a，b，c，aa，ab，ac，ba，bb, bc,…}
  - A^*={ ε ,a，b，c，aa，ab，ac，ba，bb, bc,…}

2.3文法和语言的形式化定义

- 描述语言的形式——文法
- 编写程序与产生语言

如何产生程序？

产生句子的规则——从产生语言的角度

(1) <句子>→<主语短语><动词短语>

(2) <主语短语>→ the <名词>

(3) <动词短语>→<动词><宾语短语>

(4) <宾语短语>→<冠词><名词>

(5) <名词>→ monkey

(6) <名词>→ banana

(7) <动词>→ ate

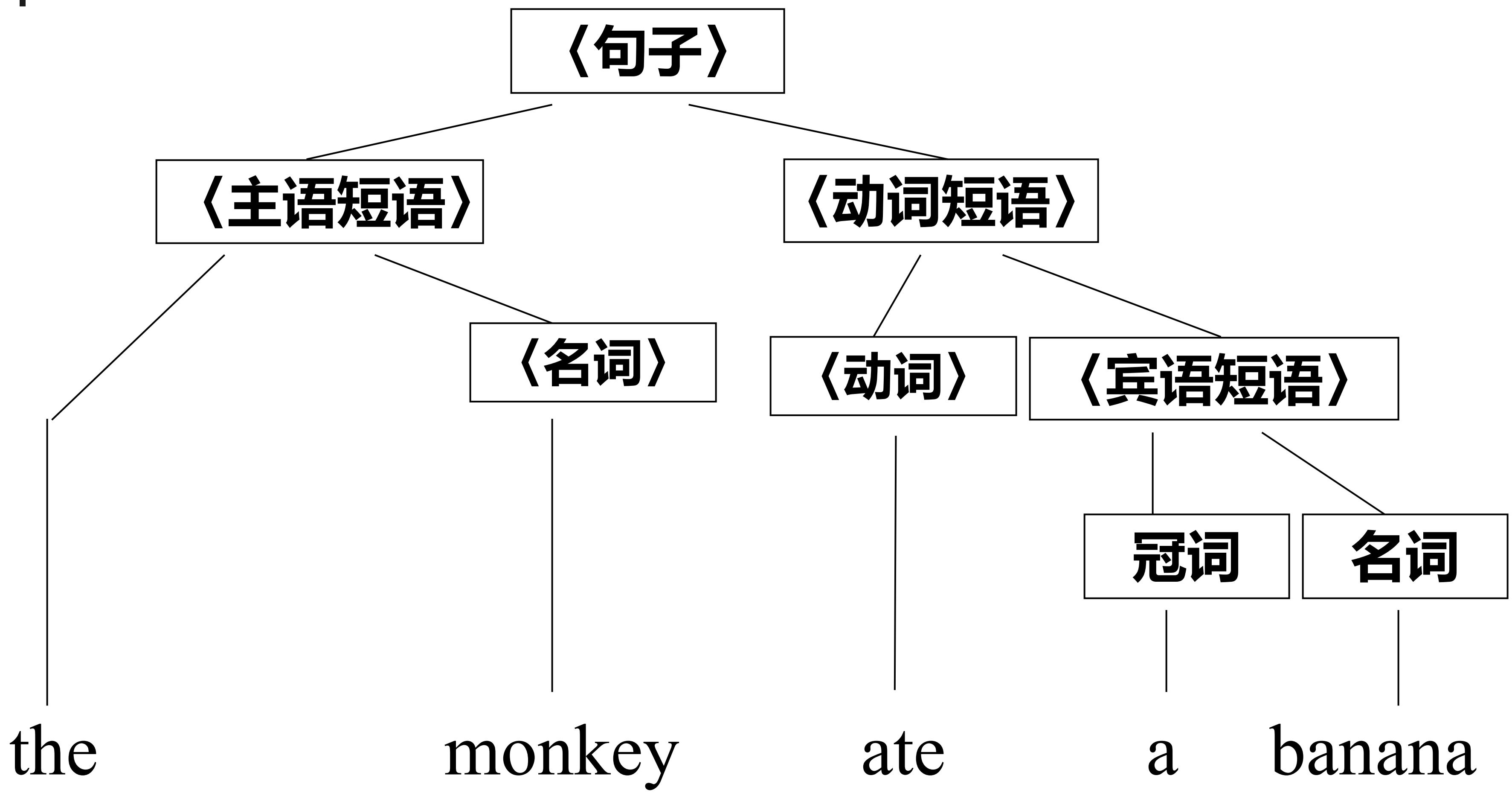
(8) <动词>→ has

(9) <冠词>→ the

(10) <冠词>→ a

考虑一个句子——文法要素的提取

分析： the monkey ate a banana



句子的派生（推导）___根据规则

<句子>⇒<主语短语> <动词短语>

⇒ <主语短语> <动词> <宾语短语>

⇒ the <名词> <动词> <宾语短语>

⇒ the <名词> <动词> < 冠词> <名词>

⇒ the monkey <动词> < 冠词> <名词>

⇒ the monkey ate <冠词> <名词>

⇒ the monkey ate a <名词>

⇒ the monkey ate a banana

产生句子的规则

(1) <句子>→<主语短语><动词短语>

(2) <主语短语>→ the <名词>

(3) < 动词短语>→< 动词><宾语短语>

(4) <宾语短语>→< 冠词><名词>

(5) <名词>→ monkey | banana

(6) <动词>→ ate | has

(7) <冠词>→ the | a

句子的语法组成

——终结符号集，非终结符号集，语法规则，开始符号

终结符号集V_T：基本符号集，不需要进一步定义

V_T={the， monkey, a,banana, ate, has}

非终结符号集V_N：需要进一步定义的语法范畴

V_N＝{<句子>，<主语短语>，<动词短语语>，<宾语短语>，<冠词>，<名词>，<动词>}

语法规则集P＝{<句子>→<主语短语><动词短语>，……}

开始符号S＝<句子>

2.3 文法的形式化定义

- **1.文法G 的形式化定义**
- 一个文法G[S]表示为形如 (V_T, V_N, P, S) 的四元式。
- V_T：终结符(Terminal)集
- V_N：非终结符(Variable)集，$V_T \cap V_N = \Phi$
  - 语法范畴——某个语言结构
- S：开始符号(Start Symbol)，$S \in V_N$
  - 至少在产生式左侧出现一次

2.3 文法的形式化定义

- 文法G 的形式定义
- P：产生式(Product)集合α→β，读作：α定义为β。α称为产生式α→β的左部(Left Part)，β称为产生式α→β的右部(Right Part)。
- $V= V_T \cup V_N$

2.3 文法的形式化定义

- **2.推导**
- 设 $\alpha_0, \alpha_1, \alpha_2 \cdots \alpha_n \in V^*$, 且有

$$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$$

称以上序列是长度为n的推导

- 记为 $\alpha_0 \overset{n}{\Rightarrow} \alpha_n$ (n步推导)
- $\alpha_0 \overset{+}{\Rightarrow} \alpha_n$ （至少一步的推导）
- $\alpha_0 \overset{*}{\Rightarrow} \alpha_n$ （若干步:零步或多步推导）

2.3 文法的形式化定义

- **2.推导**
- 从文法开始符号出发，不断将某个非终结符号替换为该非终结符号的某个产生式体

2.3 文法的形式化定义

- 3.直接推导
- 根据产生式对符号串进行变换的过程
  - $A \rightarrow \gamma$ 是文法G的一个产生式，
  - 且 α、$\beta \in V^*$，
  - 称 $\alpha A \beta$ 直接推导/派生(Derive)出 $\alpha \gamma \beta$，也称 $\alpha \gamma \beta$ 直接归约(Reduce)为 $\alpha A \beta$。
  - 记为 $\alpha A \beta \Rightarrow \alpha \gamma \beta$

例 算术表达式的文法

考虑用文法表示这个定义

标识符 i 是表达式 ⟶ $F \to i$

表达式加一个表达式是表达式 ⟶ $E \to E + T$

表达式乘一个表达式是表达式 ⟶ $T \to T * F$

表达式加上括号后是表达式 ⟶ $F \to (E)$

例 算术表达式的文法

- P: $E \rightarrow E + T$
- $E \rightarrow T$
- $T \rightarrow T * F$
- $T \rightarrow F$
- $F \rightarrow (E)$
- $F \rightarrow i$

- G =（{i， +， *， (，)}， E， P， {E,T,F}）
- 约定：可以只写产生式
- 简写G[E] :$E \rightarrow E + T \mid T \quad T \rightarrow T * F \mid F \quad F \rightarrow (E) \mid i$

产生式的简写

- 对一组有相同左部的产生式

$A \rightarrow \beta_1$，$A \rightarrow \beta_2 \ldots$，$A \rightarrow \beta_n$

简单地记为：

$A \rightarrow \beta_1 | \beta_2 | \ldots | \beta_n$

读作：A定义为或者β_1，或者β_2，…，或者β_n。并且称它们为**A产生式**。β_1，β_2，…，β_n称为**候选式**(Candidate)

- 文法如何实现对语言的刻画？产生式很关键！

推导与直接推导

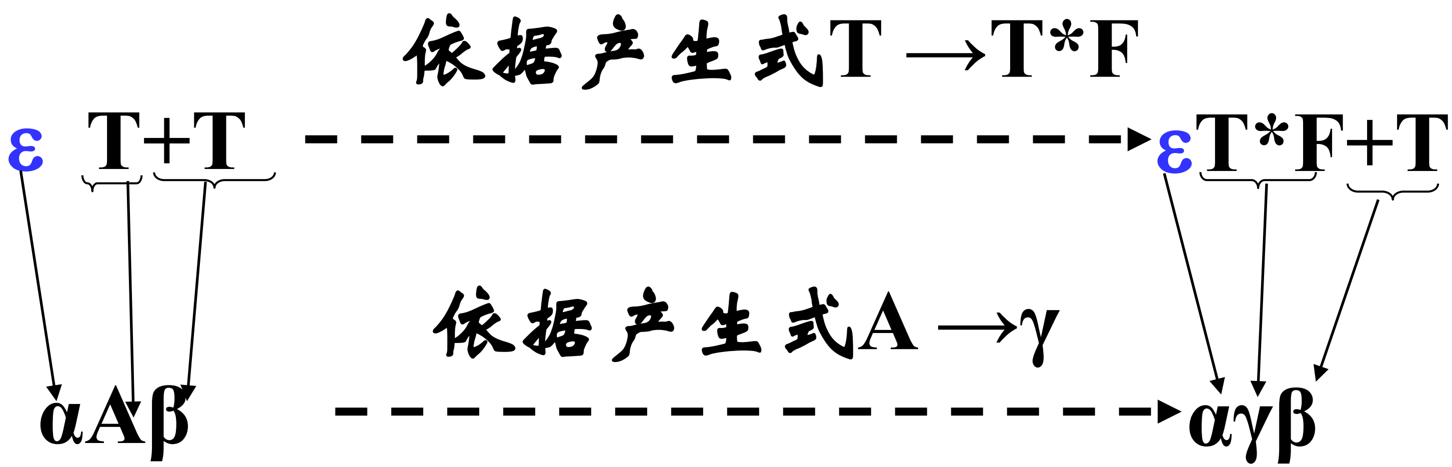
1 $E \rightarrow E+T$　　4 $T \rightarrow F$
2 $E \rightarrow T$　　5 $F \rightarrow (E)$
3 $T \rightarrow T*F$　　6 $F \rightarrow i$

- E可以变成E+T　　$E \Rightarrow E+T$　（1）
- E+T中的第一个E变成T　　$\Rightarrow T+T$　（2）
- T+T 变成T*F+T　　$\Rightarrow T*F+T$　(3)
- T*F+T变成F * F + T　　$\Rightarrow F*F+T$　（4）
- F * F + T变成i * F + T　　$\Rightarrow i*F+T$　（5）
- i * F + T变成i * i + T　　$\Rightarrow i*i+T$　（6）
- i * i+ T变成i*i+F　　$\Rightarrow i*i+F$　（7）
- i*i+F变成i* i +i　　$\Rightarrow i*i+i$　（8）

E经8步变换变成i*i+i　　$E \overset{8}{\Rightarrow} i*i+i$

变换的分析

- 实质是从E（文法的开始符）开始依据产生式对所得串中的**特定部分**进行变换，不断获得新的串，最终得到目标



2.3 文法的形式化定义

- 4. 句型与句子
- 定义: $S \overset{*}{\Rightarrow} \alpha$，$\alpha \in V^*$，则称 α 是G的一个句型

例：$E \Rightarrow E+T \Rightarrow T+T \Rightarrow T*F+T$

$E \overset{3}{\Rightarrow} T*F+T$

- 定义: $S \overset{*}{\Rightarrow} \alpha$，$\alpha \in V_T^*$，则称 α 是G产生的一个句子
  - $E \overset{8}{\Rightarrow} i+i*i$

还可以“得出”其他的句子

<句子>⇒ the monkey ate a monkey

或the banana ate a monkey

或the banana ate a banana

……

符合语法且符合“语义”的句子：

the monkey ate a banana

2.3 文法的形式化定义

- 5.文法G产生的语言
- $L(G)=\{x|S \overset{*}{\Rightarrow} x，\ x\in V_T^*\}$
- 文法 G[E] :E→ E + T | T T→ T * F | F F→ (E) | i
- 产生的语言如何?
  - 文法G的作用——语言的有穷描述
    - 以有限的规则描述无限的语言
  - 有限:产生式集合、终结符集合、非终结符集合
  - 无限：可以导出无穷多个句子
    - (注：L也可是有穷)

2.3 文法的形式化定义

- **6.递归文法**

当语言是无限集时，能否用有限的规则来描述呢？

回答是肯定的，只需使用**文法的递归定义**即可。

例如，文法G1[<标识符>]：

<标识符>→<字母>|<标识符><字母>|<标识符><数字>

<字母>→A|B|…|Z

<数字>→0|1|2|…|9

2.3 文法的形式化定义

6. 递归文法

文法G2[E]：

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

显然，G1，G2都是递归定义的。所谓递归定义，指在定义一个语法成分时，直接或间接地使用了语法成分自身。

定义2.6　设G为文法，$A \rightarrow \alpha \in P$，若$\alpha$具有$\upsilon A\delta$的形式，$\upsilon\delta \neq \varepsilon$，则称产生式$A \rightarrow \alpha$是直接递归的；若存在推导$A \Rightarrow \alpha \overset{*}{\Rightarrow} \upsilon A\delta$，(且$\upsilon = \varepsilon / \delta = \varepsilon$)则称$A \rightarrow \alpha$是(左/右)递归的.称$A$为(左/右)递归的非终结符号.一文法至少含有一个递归的非终结符则称为递归文法.

2.3 文法的形式化定义

- 7. 文法等价
- 若L(G1)=L(G2)，则称文法G1和G2是等价的。也就是说，如果两个文法定义的语言一样，则称这两个文法是等价的。
  例如

  文法G[A]：A→0R　A→01　R→A1
  文法G[S]：S→0S1，S→01
- 上述两个文法等价。

文法的构造——为了更好地理解文法

- 目的：给出语言的有穷描述
- 途径：刻画语言的结构
- 做法：
  - 给出定义的形式化描述
  - 根据经验给出描述

文法举例

- 给出能够产生下列语言的文法
- {x|x是长度为偶数的0、1串}

  $S \rightarrow 00S|01S|10S|11S|\varepsilon$

- $\{0^m 1^n | m,n \geq 1\}$

  $S \rightarrow 0S|0A$　　　　$A \rightarrow 1A|1$

- $\{0^n 1^n | n \geq 1\}$

  $S \rightarrow 0S1|01$

文法举例

- 一个文法的几种写法

① $G=(\{S,A\}，\{a,b\}，P，S)$

其中P：$S\rightarrow aAb$

$A\rightarrow ab$

$A\rightarrow aAb$

$A\rightarrow \varepsilon$

② G：$S\rightarrow aAb$

$A\rightarrow ab$

$A\rightarrow aAb$

$A\rightarrow \varepsilon$

③ G[S]：$S\rightarrow aAb \quad A\rightarrow ab \quad A\rightarrow aAb \quad A\rightarrow \varepsilon$

④ G[S]：$S\rightarrow aAb \quad A\rightarrow ab \mid aAb \mid \varepsilon$

文法构造小结

- 明确描述对象——语言
  - 合法的语言结构
- 确定基本符号集V_T
- 引入非终结符
  - 各种句子结构
- 定义句子的组成规则
  - BNF范式或产生式

习题

- 构造文法生成下列语言
  - (1) $\{a^n \mid n \geq 1\}$
  - (2) $\{a^n \mid n \geq 0\}$
  - (3) $\{a^{2n+1} \mid n \geq 0\}$
- 给出下面文法的语言的特点

(1) $S \rightarrow aS \mid \varepsilon$

(2) $S \rightarrow A \mid AB \quad A \rightarrow 0 \mid 0A \quad B \rightarrow 1 \mid 11$

(3) $S \rightarrow aSS \mid a$

(4) $S \rightarrow 1S0 \quad S \rightarrow aA \quad A \rightarrow bA \quad A \rightarrow a$

小结

- 文法和语言的表示
- 前后文无关文法的定义
- 基本概念
- 语言和文法的相关概念
- 文法的递归与无限语言
- 文法推导出语言的特点
- 根据语言的特点给出文法

2.4 文法的分类(Chomsky体系)

- 文法G 的形式定义
  - 一个文法G[S]表示为形如（V_T, V_N, P, S）的四元式。

（$V_T \cap V_N = \Phi$　　　$V_T \cup V_N = V$）

Chomsky对产生式的形式加以限制，得到四类基本文法：0型文法、1型文法、2型文法、3型文法

2.4 文法的分类(Chomsky体系)

- 0型文法(短语结构文法PSG)
  - 如果G满足文法定义的要求，则G是0型文法，（短语结构文法PSG: Phrase Structure Grammar），L(G)称为PSL。
  - 若P中任一产生式都有一般形式:

    $\alpha \rightarrow \beta \quad \alpha \in V^{+}, \beta \in V^{*}$ 且对α，β不加任何限制

2.4 文法的分类(Chomsky体系)

0型文法(短语结构文法PSG)

- 由0型文法生成（或者说：定义）的语言称为0型（递归可枚举）语言,它可由图灵（Turing）机识别。
- 例如：S→ACaB Ca→aaC CB→DB CB→E aD→Da AD→AC aE→Ea AE→ε 就是一个0型文法，它所产生的语言

$$L_0 = \{ a^{2^i} \mid i \in I_+ \} = \{aa, aaaa, aaaaaaaa, \cdots\}$$

2.4 文法的分类(Chomsky体系)

- 1型文法（前后文有关文法）
  - 若一个0型文法G所有产生式具有形式：

    $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$

    其中， $\alpha_1,\alpha_2 \in V^*$ $\beta \in V^+$ $A \in V_N$,

    则称G 为1型(前后文有关)文法，记为CSG（Context Sensitive Grammar)。
  - 1型文法产生的语言称为前后文有关语言CSL，它可由线性限界自动机识别。
  - 命名的由来：只有当非终结符A的前后分别为α_1,α_2 时，才能将A替换为β。

2.4 文法的分类(Chomsky体系)

- 1型文法（前后文有关文法）
  - 1型文法还有另一种定义形式：G的每个产生式形为$\alpha \to \beta$，且满足：$|\alpha| \le |\beta| \quad \alpha, \beta \in V^+$，则G是1型文法。
  - 例 文法G：$S \to \varepsilon \mid A \quad A \to aABC \quad A \to abC \quad CB \to BC \quad bB \to bb \quad bC \to bc \quad cC \to cc$
  - 因G含有ε-产生式，所以它不是一个严格意义下的1型文法。它所产生的语言为

$$\boxed{L(G) = \{a^i b^i c^i \mid i \ge 0\}}$$

2.4 文法的分类(Chomsky体系)

- 2型文法（前后文无关文法）
  - 若1型文法G中所有产生式具有形式：

    $A \rightarrow \beta \qquad \beta \in V^{+} \quad A \in V_N$

    则称G为2型（前后文无关）文法，记为CFG(Context Free Grammar)。
  - 2型文法产生的语言称为前后文无关语言（CFL），它可由下推自动机识别。

2.4 文法的分类(Chomsky体系)

- 2型文法（前后文无关文法）
  - 若允许ε-产生式存在，则CFG产生式形式为

    $A \to \beta \qquad \beta \in V^* \quad A \in V_N$
  - 例：G[S]=({S},{a,b},{S→aSb　S→ab},S)产生的语言为
  - $L(G)=\{a^i b^i \mid i \geq 1\}$

2.4 文法的分类(Chomsky体系)

- 正规文法(RG)
  - 若一个2型文法中仅有形如以下的产生式
    $A \rightarrow aB$ 或 $A \rightarrow a$　A、$B \in V_N$，$a \in V_T \cup \{\varepsilon\}$
  - 则称为右线性(Right Linear)文法。

  - 若一个2型文法中仅有形如以下的产生式
    $A \rightarrow Ba$ 或 $A \rightarrow a$　A、$B \in V_N$，$a \in V_T \cup \{\varepsilon\}$
  - 则称为左线性(Right Linear)文法。

  - 左线性和右线性文法都是3型文法(正规文法 Regular Grammar -RG)

2.4 文法的分类(Chomsky体系)

- 正规文法(RG)
  - L(G)为3型/正规集/正则集/正则语言（RL）
    - 例：程序设计语言的多数词法特性
  - 左、右线性文法不可混用
- 使用有限自动机(FA)识别正规语言

Chomsky体系——总结

0型文法	1型文法	2型文法	3型文法	3型文法
(PSG)	(CSG)	(CFG)	S→a|b	S→a|b
S→aBC	S→aBC	E→E+E	S→aT|bT	S→Ha|Hb
S→aSBC	S→aSBC	E→E*E	T→a|b	S→H1|H2
CB→BC	CB→BC	E→(E)	T→1|2	H→Ha|Hb
aB→d	aB→ab	E→id	T→aT|bT	H→H1|H2
bB→bb	bB→bb	E→E-E	T→1T|2T	H→a|b
bC→b	bC→bc	E→E/E		
cC→ cc	cC→ cc			

Chomsky体系——总结

$G=(V_T, V_N, P, S)$是一个文法,$\alpha\to\beta\in P$

* G是0型文法，L(G)是0型语言；

  ---其能力相当于图灵机(TM)

* $|\alpha|\le|\beta|$:G是1型文法，L(G)是1型语言(除$S\to\varepsilon$);

  ---其识别系统是线性界限自动机(LBA)

* $\alpha\in V_N$: G是2型文法，L(G)是2型语言;

  ---其识别系统是不确定的下推自动机(PDA)

* $A\to aB$或$A\to a$: G是右线性文法，L(G)是3型语言

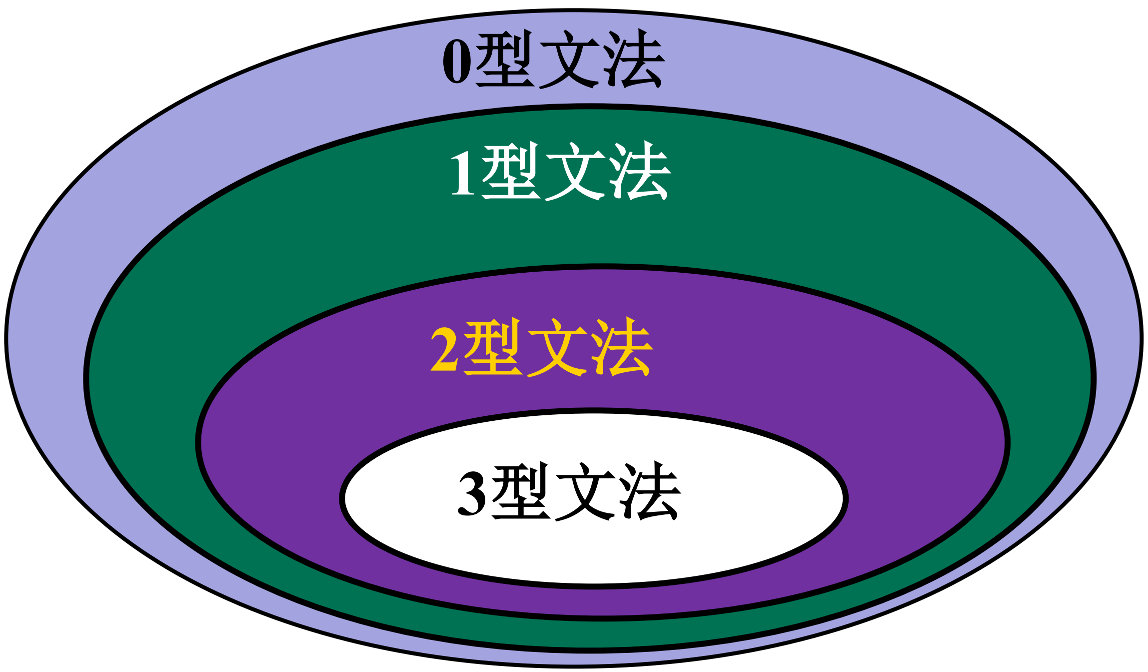
  $A\to Ba$或$A\to a$: G是左线性文法，L(G)是3型语言

  ---其识别系统是有穷自动机(FA)

Chomsky体系——总结

四种文法之间的关系是将产生式作进一步限制而定义的

四种文法之间的逐级"包含"关系如下：



BNF范式——Backus-Naur Form
Backus-Normal Form

- $\alpha \rightarrow \beta$表示为$\alpha ::= \beta$
- 非终结符用“<”和“>”括起来
- 终结符：基本符号集

BNF范式——Backus Normal Form

- 例 简单算术表达式(只写产生式)
  - <简单表达式>::=<简单表达式>+<简单表达式>
  - <简单表达式>::=<简单表达式>*<简单表达式>
  - <简单表达式>::=(<简单表达式>)
  - <简单表达式>::=id
- 即：<简单表达式>::=<简单表达式>+<简单表达式>|

  <简单表达式>*<简单表达式>|(<简单表达式>)|id
- 哪些是终结符？哪些是变量？

小结

- 四类文法的基本形式
- 四类文法的语言
- *BNF*范式的概念

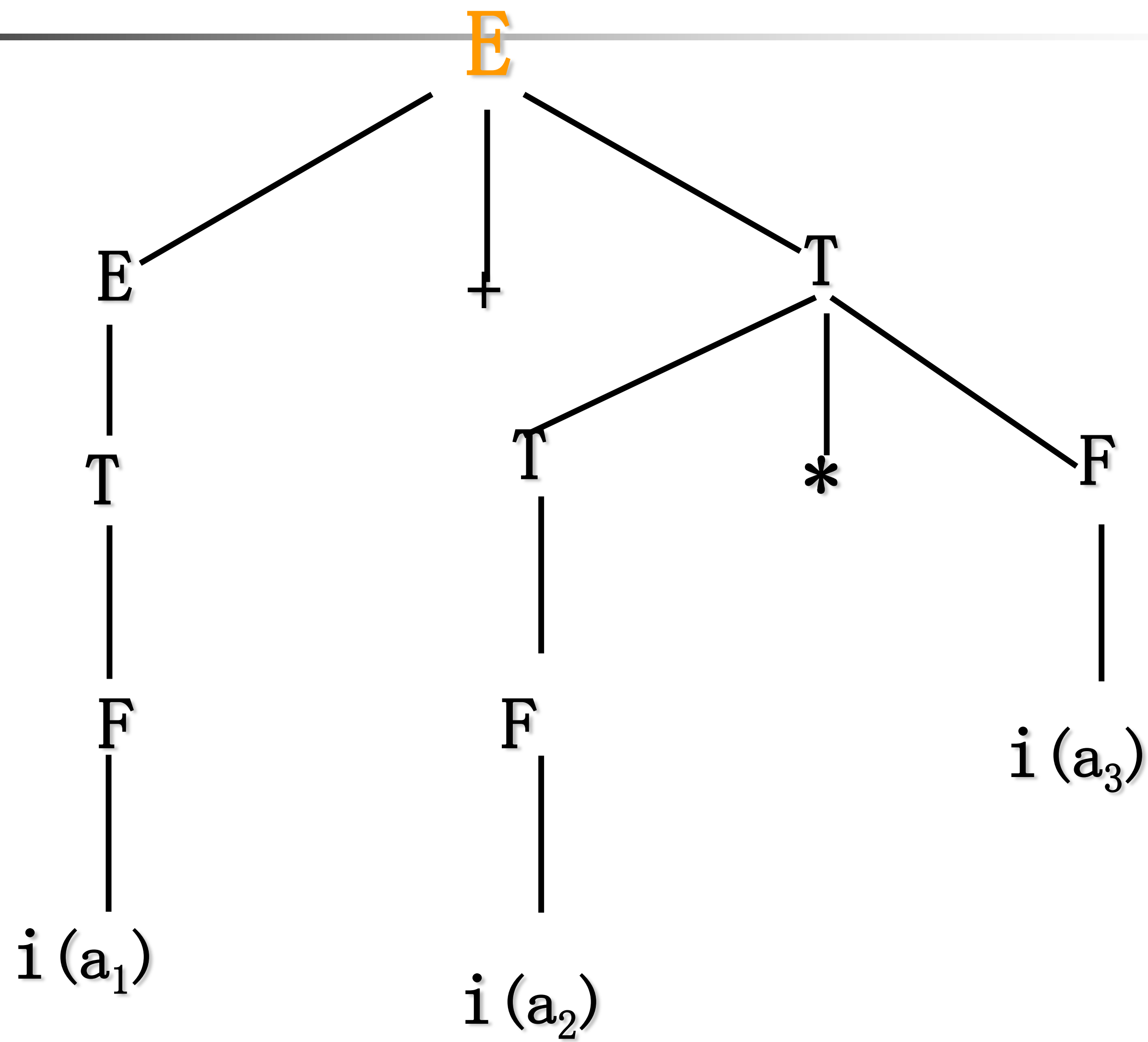
2.5 语法树

- 什么是语法树？
- 用树的形式表示句型（句子）的语法结构，又称为语法分析树
- 设$G=(V_N,V_T,P,S)$是一文法，则满足下述条件的树称为语法树：
  - 1）每个结点有一标记X，$X\in V$；
  - 2）根的标记为S（开始符）；
  - 3）若结点X有后继，则$X\in V_N$；
  - 4）A有k个后继，自左至右为$X_1, X_2, \ldots, X_k$，则$A\rightarrow X_1X_2\ldots X_k \in P$

例:语法(分析)树　i+i*i

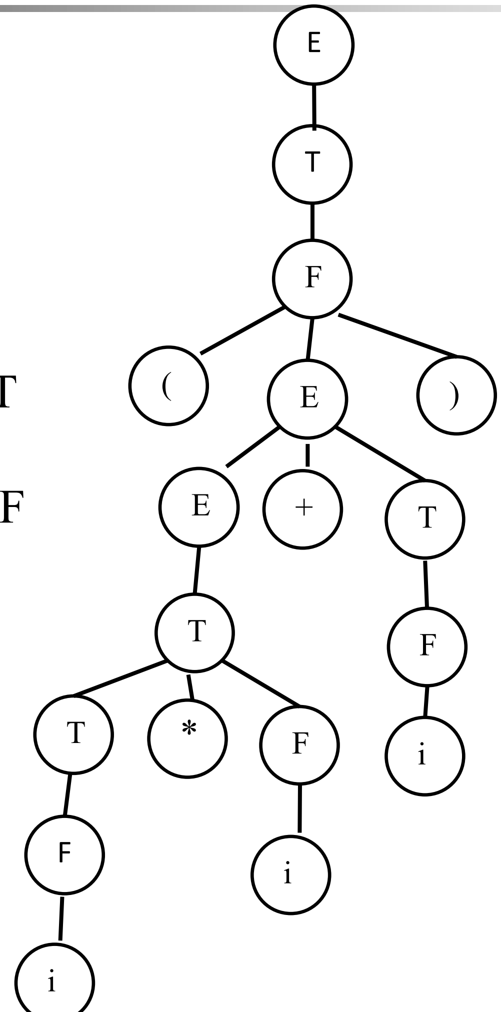
文法:　$E \to E + T \mid T$　　$T \to T * F \mid F$

$F \to (E) \mid i$



(i*i+i)的语法树

E→E + T | T T→T* F | F F→(E) | i



最左推导

E ⇒ T 产生式：E →T
⇒ F 产生式：T →F
⇒(E) 产生式：F →(E)
⇒(E+T) 产生式：E →E+T
⇒(T+T) 产生式：E →T
⇒(T*F+T) 产生式：T →T*F
⇒ (F*F+T) 产生式：T →F
⇒(i*F+T) 产生式：F →i
⇒(i*i+T) 产生式：F →i
⇒(i*i+F) 产生式：T →F
⇒(i*i+i) 产生式：F →i

最右推导

E ⇒ T 产生式：E →T
⇒ F 产生式：T →F
⇒(E) 产生式：F →(E)
⇒(E+T) 产生式：E →E+T
⇒(E+F) 产生式：T →F
⇒(E+i) 产生式：F →i
⇒(T+i) 产生式：E →T
⇒(T*F+i) 产生式：T →T*F
⇒(T*i+i) 产生式：F →i
⇒(F*i+i) 产生式：T →F
⇒(i*i+i) 产生式：F →i

2.5 语法树

- 关于语法树的结论
- 1.对于一个句子（句型），总能为其构造一棵语法树，语法树的末端从左到右标记起来就是该句子（句型）
- 2.对于同一语法树，可对应不同的推导序列，但仅有一个最左推导和一个最右推导

2.6 句型分析

- 句型分析：构造一算法，用以判断所给的符号串是否为某文法的句型（句子）
- 常见分析方法有自顶向下分析和自底向上分析两类；
- 自顶向下 从文法开始符出发试图推导出给定的符号串；
- 自底向上 推导的逆过程（称归约）：从已给的符号串出发，试图将其归约为开始符。

例 算术表达式的文法

P: $E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow i$

i+i*i的不同推导

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

不做限制

句型（sentential Form）

（归约）

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

施于最左变量

左句型(left-~)

（最右归约）

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

施于最右变量

右句型/规范句型 (canonical ~)

(最左/规范归约)

2.6.1规范推导与规范归约

- 最左/右推导(Left/Right-most Derivation)
  - 每次推导都施加在句型的最左(右)边的非终结符上——与最右（左）归约对应

- 规范推导
  - 最右推导

2.6.1规范推导与规范归约

- 形式上，从符号串α到符号串β的推导序列 $\alpha \overset{*}{\Rightarrow} xUy \Rightarrow xuy \overset{*}{\Rightarrow} \beta$ 总有 $x \in V_T^*$ ($y \in V_T^*$) 时，称为最左（右）推导

- 定义：最左（右）推导所得句型称为左（右）句型；右句型称为规范句型。

2.6.1规范推导与规范归约

- 自顶向下的分析法（推导方法）
  - 试图建立从开始符S到w最左推导:$S \overset{*}{\Rightarrow} w$
  - 显然，每步推导时，对应于最左非终结符相应的产生式可能会有多个，若无特殊的办法，只能一个一个地试探。（回溯）
  - 存在左递归：$E \rightarrow E + T$，导致分析陷入死循环

2.6.1规范推导与规范归约

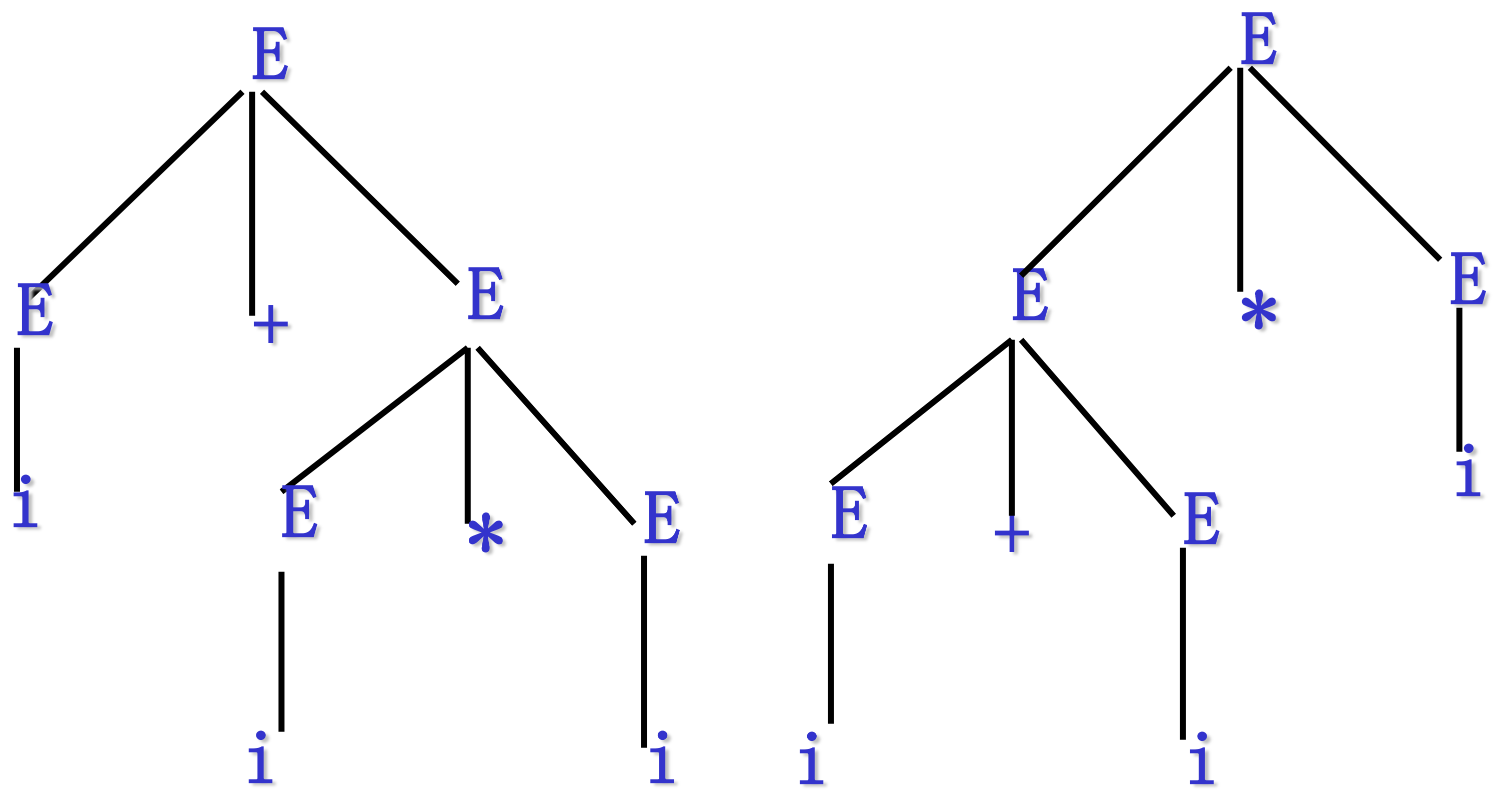
- 归约：推导的逆过程
- 最左归约（规范归约）：最右推导的逆过程。
- 自底向上的语法分析
  - 从已给的符号串w出发，试图以相反的方向为w建立一个规范推导，最终得到文法的开始符。
  - 建立最左归约的过程。

2.6.2 二义性

- 文法的二义性(歧义性/ambiquity)
  - 1.文法中存在一个句子w，w的语法树不只一个；
  - 2.或w有多个不同的最左推导和最右推导。
  - 3.如果一个文法包含二义性的句子,则称这个文法是二义性的；否则，该文法是无二义性的
  - 考虑表达式下面的文法 G[E],其产生式如下：
  - $E \to E+E \mid E*E \mid (E) \mid i$

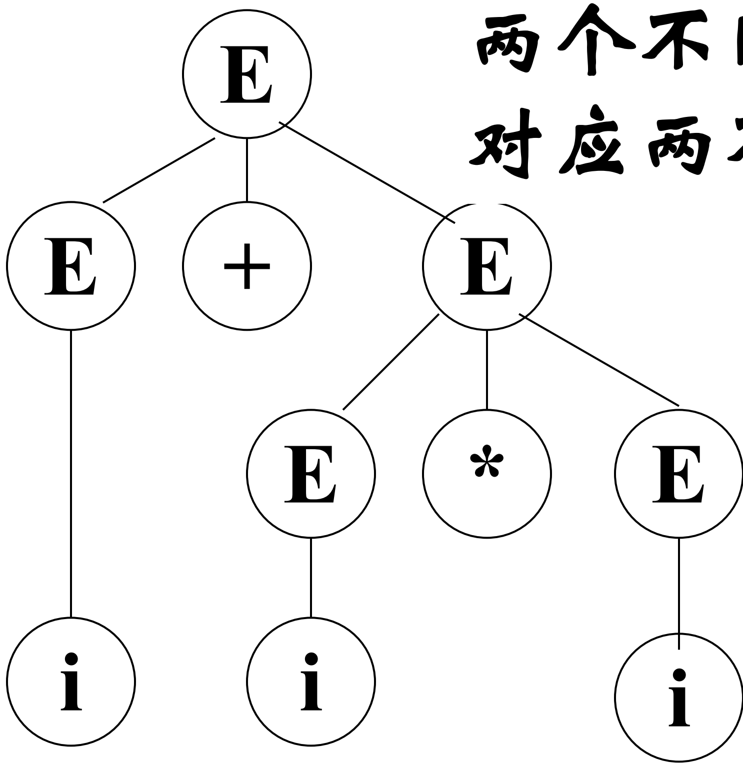
2.6.2 文法的二义性

- 一个句子 i+i*i 有两棵不同的语法树

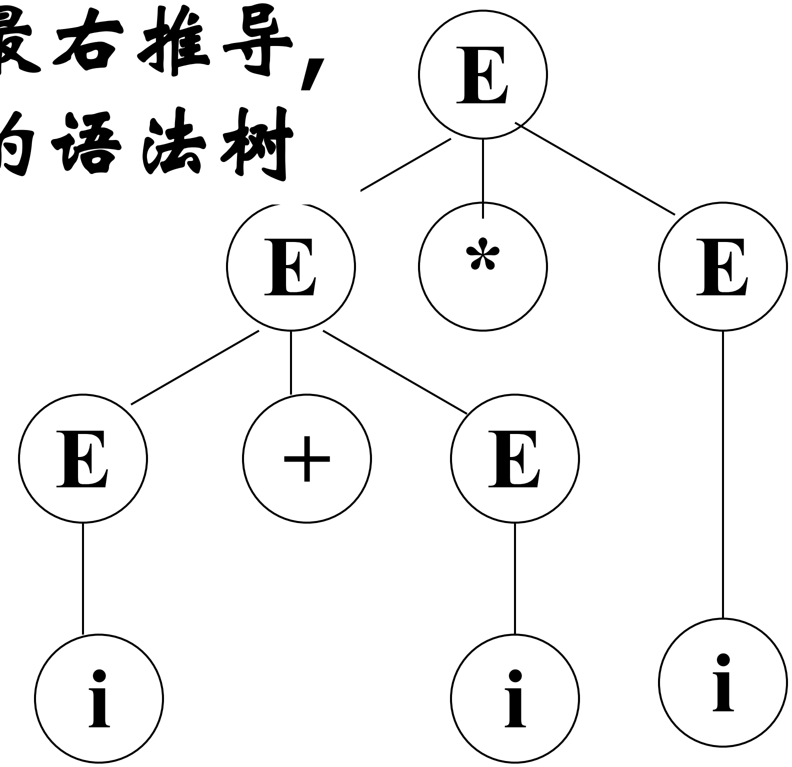


$$
\begin{aligned}
E &\Rightarrow E+E \\
&\Rightarrow E+E*E \\
&\Rightarrow E+E*i \\
&\Rightarrow E+i*i \\
&\Rightarrow i+i*i
\end{aligned}
$$

$$
\begin{aligned}
E &\Rightarrow E*E \\
&\Rightarrow E*i \\
&\Rightarrow E+E*i \\
&\Rightarrow E+i*i \\
&\Rightarrow i+i*i
\end{aligned}
$$



两个不同的最右推导，
对应两不同的语法树

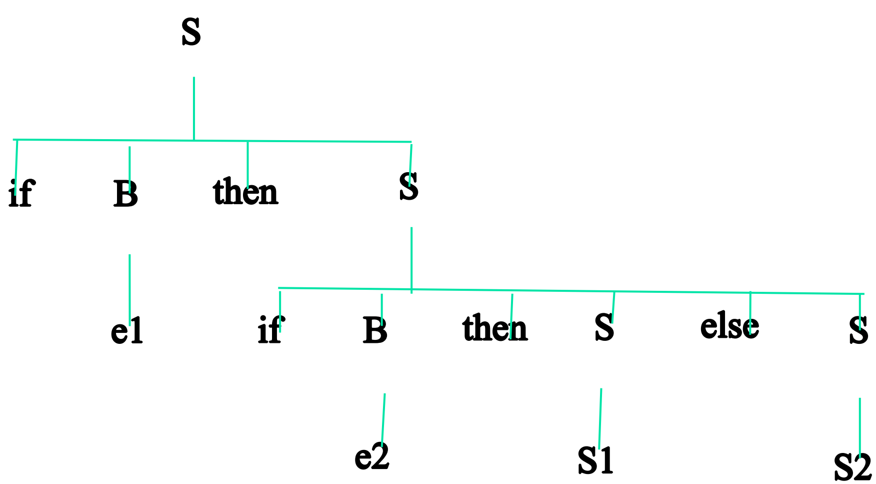


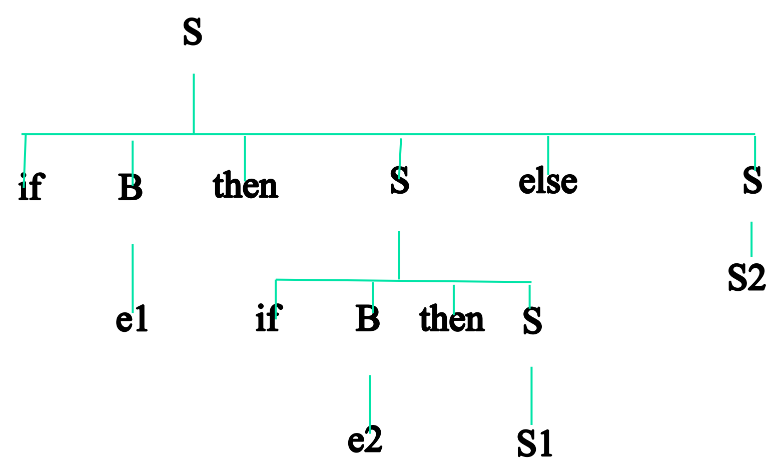
2.6.2 文法的二义性

- 一般来说,高级程序设计语言存在无二义性文法,但有时用二义性文法。如：表达式文法、条件语句文法
- $S \rightarrow$ if B then S
  | if B then S else S
  | other
  - 二义性的句子: if e_1 then if e_2 then s_1 else s_2

文2.6.2法的二义性

- 二义性的句子：if e_1 then if e_2 then s_1 else s_2





2.6.2文法的二义性

- 对于任意一个CFG（前后文无关文法），不存在算法判定它是无二义性的；但能给出一组充分条件，满足这组充分条件的文法是无二义性的

- 存在先天二义性语言。例如，语言

$$\{a^ib^ic^j \mid i,j\geq 1\} \cup \{a^ib^jc^j \mid i,j\geq 1\}$$

  存在二义性的句子$a^kb^kc^k$

- 一个语言是否为先天二义性的，在理论上不可判定

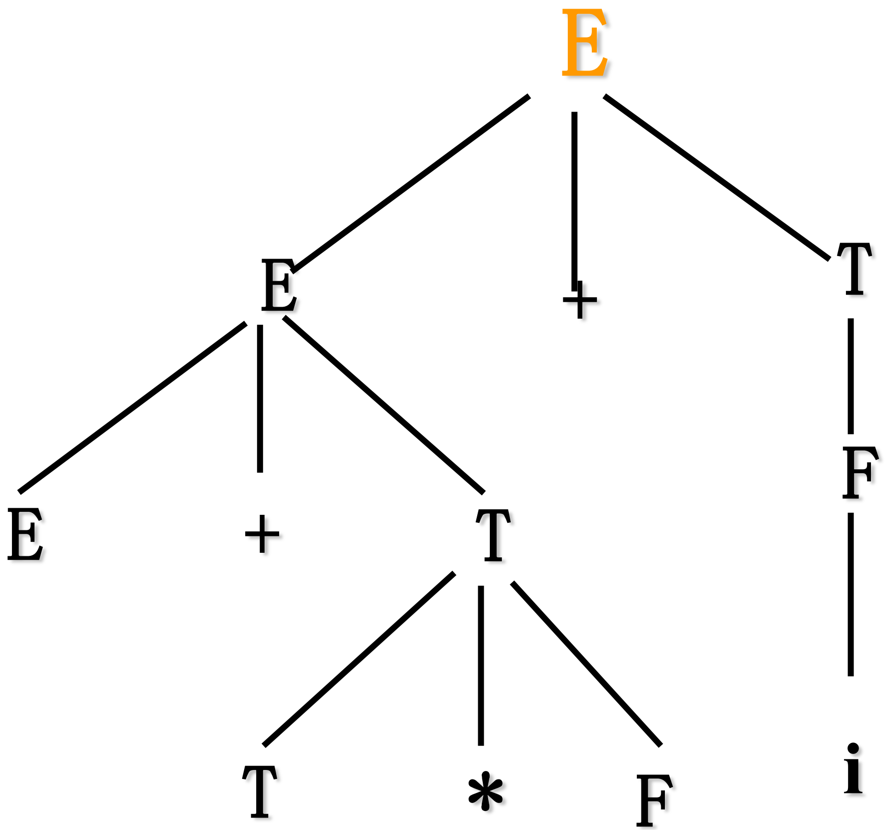
2.6.3短语(Phrase)和句柄(Handle)

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid i$

句型E+T*F+i的语法树



一、短语(Phrase)

- 短语：在句型（句子）的语法树中，子树的末端符号串是子树相对于根的短语。
- 直接短语：仅有父子两代的子树的情形。

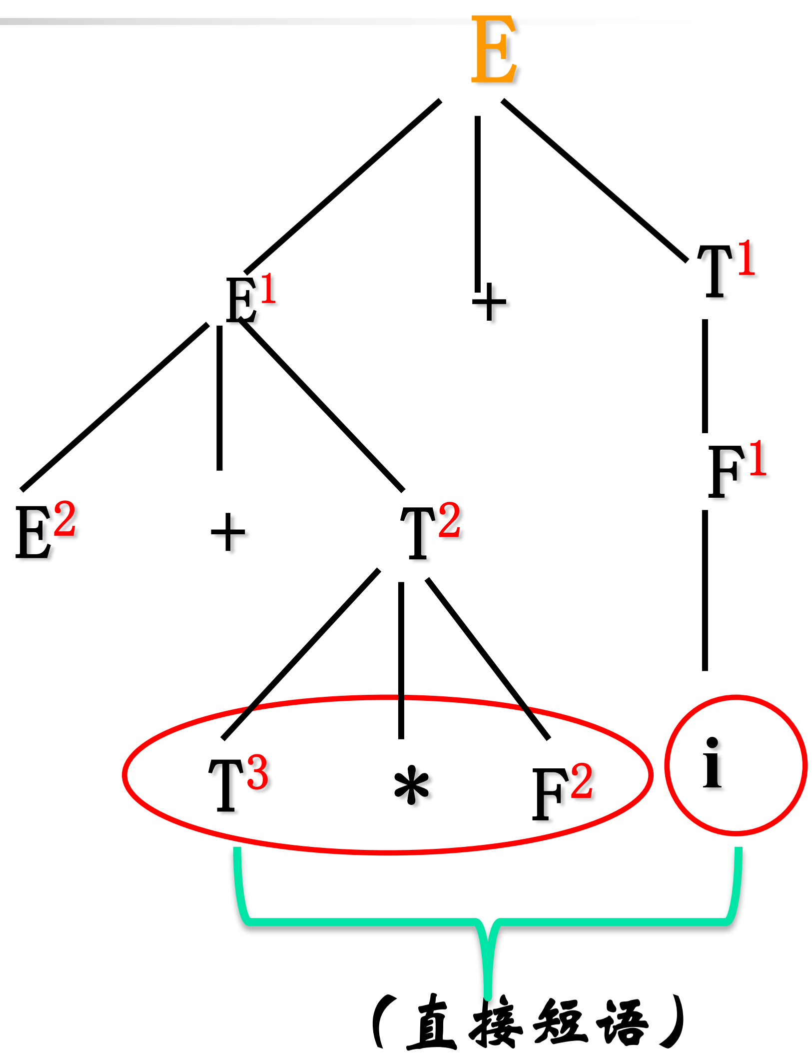
T*F是句型相对于T^2的短语

E+T*F是句型相对于E^1的短语

i是句型相对于T^1的短语

i是句型相对于F^1的短语

E+T*F+i是句型相对于E的短语



一、短语(Phrase)

- 短语的形式化定义：
- 如果 $S \Rightarrow^{*} \alpha A \beta$ 且 $A \Rightarrow^{+} \gamma$，则称γ是句型$\alpha\gamma\beta$的相对于变量A的短语
- 如果 $S \Rightarrow^{*} \alpha A \beta \;\&\; A \Rightarrow \gamma$

则称γ是句型$\alpha\gamma\beta$的相对于变量A的直接短语

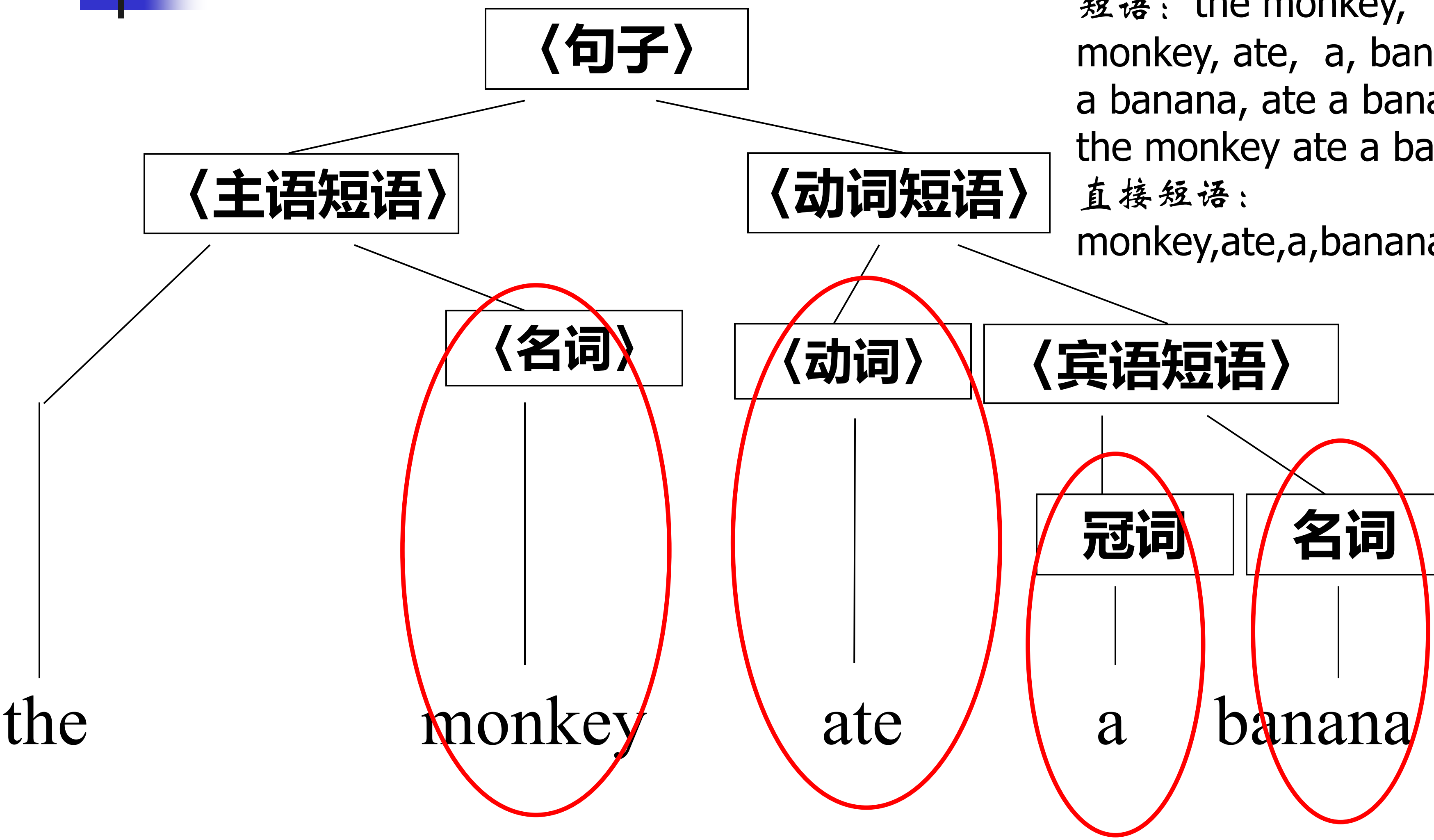
- 短语——一棵子树的叶子！

一、短语(Phrase)

the monkey ate a banana

短语： the monkey, monkey, ate, a, banana, a banana, ate a banana the monkey ate a banana

直接短语：
monkey,ate,a,banana



例： 句型的短语与直接短语（i*i+i）

$E \rightarrow E+T \mid T$

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

$F \rightarrow (E) \mid i$

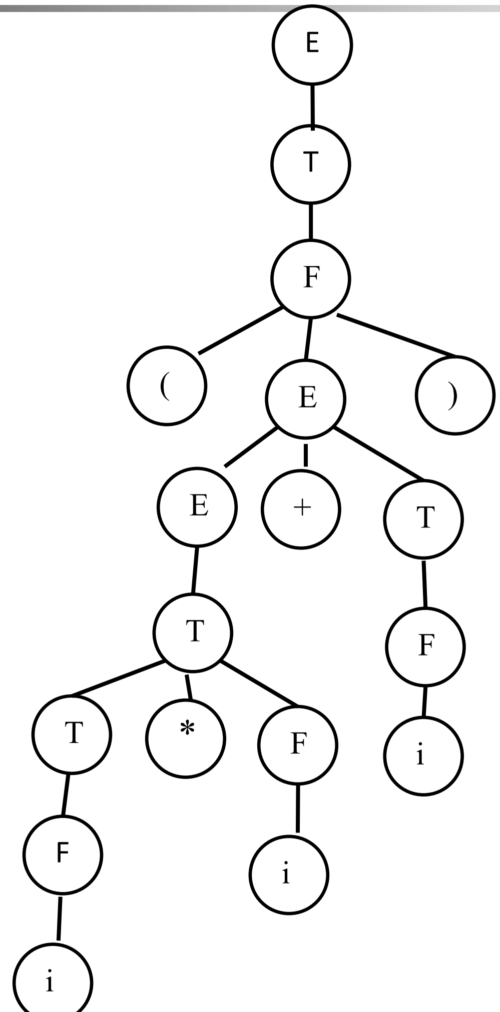
全部短语：

i

i*i

i*i+i

(i*i+i)



最右推导

$E \Rightarrow T$ 产生式： $E \rightarrow T$

$\Rightarrow F$ 产生式： $T \rightarrow F$

$\Rightarrow (E)$ 产生式： $F \rightarrow (E)$

$\Rightarrow (E+T)$ 产生式： $E \rightarrow E+T$

$\Rightarrow (E+F)$ 产生式： $T \rightarrow F$

$\Rightarrow (E+i)$ 产生式： $F \rightarrow i$

$\Rightarrow (T+i)$ 产生式： $E \rightarrow T$

$\Rightarrow (T*F+i)$ 产生式： $T \rightarrow T*F$

$\Rightarrow (T*i+i)$ 产生式： $F \rightarrow i$

$\Rightarrow (F*i+i)$ 产生式： $T \rightarrow F$

$\Rightarrow (i*i+i)$ 产生式： $F \rightarrow i$

直接短语： i

二、句柄

- 自底向上的语法分析法——归约
- 句柄(Handle)：规范归约中的**可归约串**；语法树中的**最左直接子树的叶子串**；**最左直接短语**
- 例如，句型E+T*F+i中，直接短语（T*F和i），句柄（T*F）。

句型的句柄(Handle)——最左直接短语

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid i$

句子：i+i*i+i

短语：

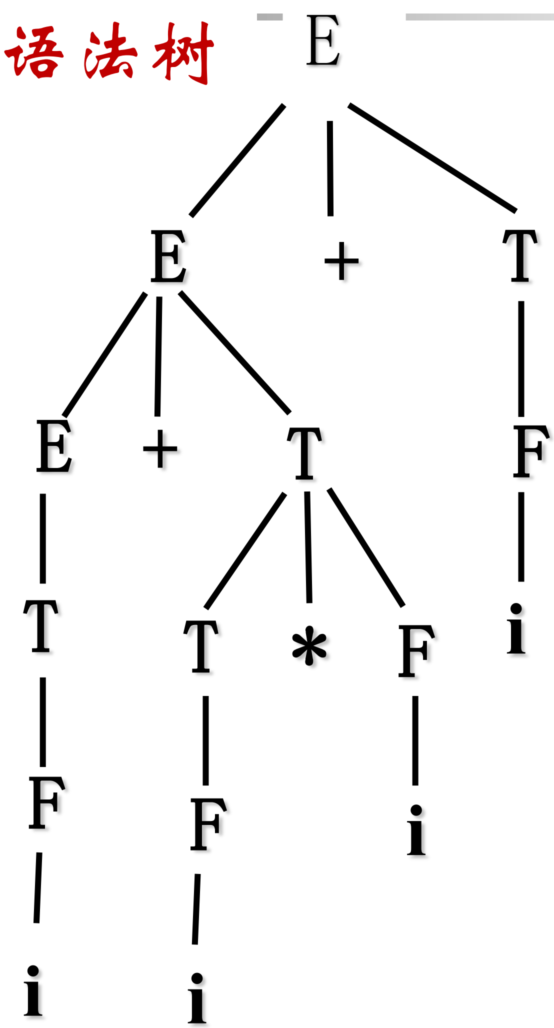
i （直接短语）

i*i

i+i*i

i+i*i+i

语法树



最右推导	句柄
$E \Rightarrow E+T$	E+T
$\Rightarrow E+F$	F
$\Rightarrow E+i$	i
$\Rightarrow E+T+i$	E+T
$\Rightarrow E+T*F+i$	T*F
$\Rightarrow E+T*i+i$	i
$\Rightarrow E+F*i+i$	F
$\Rightarrow E+i*i+i$	i
$\Rightarrow T+i*i+i$	T
$\Rightarrow F+i*i+i$	F
$\Rightarrow i+i*i+i$	i

二、句柄

- 句柄的应用背景
  - 自下而上的语法分析过程中（优先分析法，LR类分析法）
- 问题
  - （1）如何确定一个规范句型的句柄？
  - （2）将句柄规约为哪个非终结符？

课堂举例及提问

- 1.以下文法是否具有二义性？

  文法的产生式如下：

  $P=\{S \rightarrow AB,\ A \rightarrow a \mid ab,\ B \rightarrow c \mid bc\}$

- 2.文法G[S]: $S \rightarrow SS+ \mid SS^* \mid a$

给出句子aa*a+a+的语法树、最右推导，并写出各步推导所得句型的句柄以及句子的全部短语，指出直接短语。

小结（本章的重点内容之一）

- 句型分析的两种方法
- 最左（右）推导
- 最左（右）归约
- 规范推导、规范归约
- 自底向上的语法分析方法
- 自顶向下的语法分析方法
- 语法树和二义性
- 短语和句柄