广东工业大学计算机学院

第二章 文法和语言

§2.4 文法的类型

§2.5 上下文无关及其语法树

编译原理

复习

- 文法产生的目的是什么？
- 文法的定义？
- 文法与语言的关系？
- 写一文法G，使得 $L(G) = \{ a^m b^n \mid m \geq 0, n \geq 1 \}$
- 写出以下文法G所定义的语言L(G)。

  G：$S \rightarrow SaS$

  $S \rightarrow b$

  $S \rightarrow d$

本课内容

- 2.4 文法的类型
- 2.5 上下文无关文法及其语法树

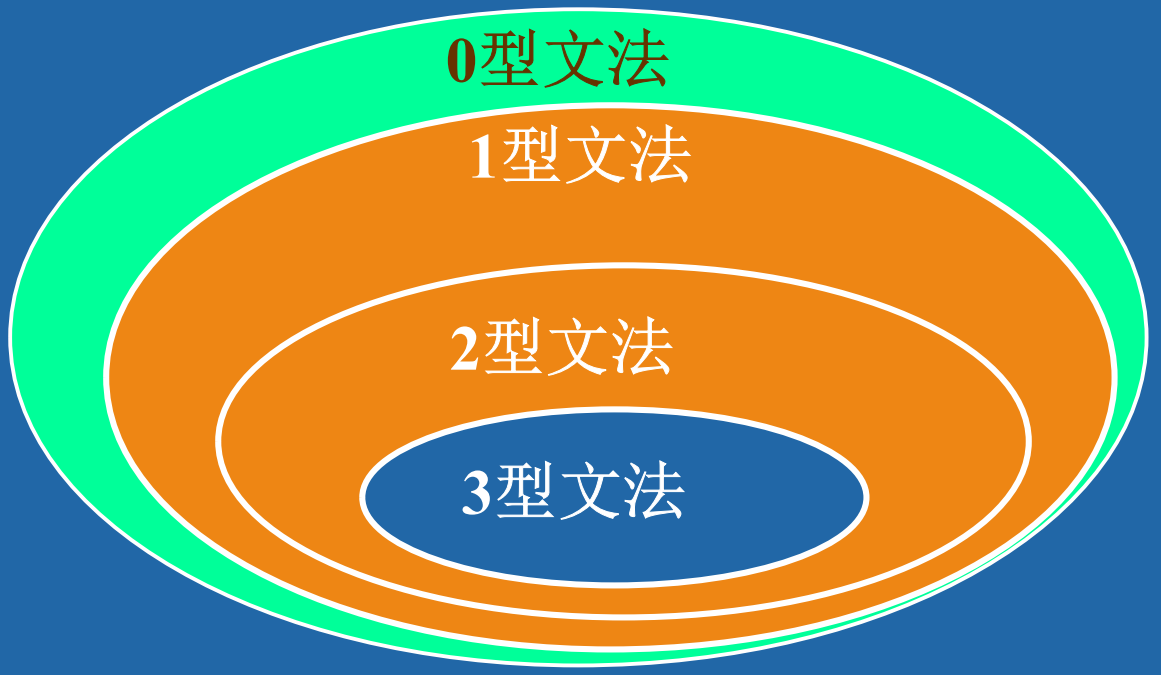
文法的分类

- 自从乔姆斯基(Chomsky)于1956年建立形式语言的描述以来，形式语言的理论发展很快。
- 这种理论对计算机科学有着深刻的影响，特别是对程序设计语言的设计、编译方法和计算复杂性等方面更有重大的作用。
- 乔姆斯基把文法分成四种类型，即0型、1型、2型和3型。这几类文法的差别在于对产生式施加不同的限制。

文法的类型

四种文法之间存在“逐级包含关系”

0型文法的条件限制最为宽松，其余的文法都是在0型文法之上增加条件限制，定义而成的。



0型文法

- 0型文法也称短语文法。设G = (V_N，V_T，P，S)，如果对于每个产生式α→β，都有：α∈ (V_N ∪ V_T)*且至少含有一个非终结符，而β∈ (V_N ∪ V_T)*，则G是一个0型文法。

- 立即应用：
- 下列文法是否0型文法？
- 文法G[A]：

  A → 0R
  A → 01
  R → A1

问题分解：

1. 对G[A]，(V_N∪V_T) = ？ 再检验(V_N∪V_T) *

Answer: (V_N ∪ V_T) = {0, 1, R, A}

2. 每个产生式的左部是否都含有非终结符？

Answer: 是

- 一个非常重要的理论结果是：0型文法的能力相当于图灵机(Turing)。或者说，任何0型语言都是递归可枚举的；反之，递归可枚举集必定是一个0型语言。

1型文法

- 对0型文法产生式的形式作某些限制，即可给出1，2和3型文法的定义。
- 设$G = (V_N, V_T, P, S)$为一文法，若P中的每一个产生式$\alpha \rightarrow \beta$均满足$|\beta| \geq |\alpha|$，仅仅$S \rightarrow \varepsilon$除外，则文法G是1型或上下文有关的(context-sensitive)。
- 立即应用：下列文法是不是1型文法？
- 文法G[A]：

  (1) $A \rightarrow 0R1$　(2) $0R1 \rightarrow 00R11$　(3) $R \rightarrow 0 \mid 1$ √
- 某些文献给将上下文有关文法的产生式的形式描述为：

  $$\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$$
- 其中α_1、α_2、$\beta \in (V_N \cup V_T)^*$，$\beta \neq \varepsilon$，$A \in V_N$
- 这种定义与前边的定义等价。但后者更能体现“上下文有关”这一术语，因为只有A出现在α_1和α_2的上下文中，才允许用β取代A。

1型文法举例

- 例：设有文法G[S]：
- G[S]：S→CD　　Ab→bA　C→aCA
  Ba→aB　C→bCB　　Bb→bB AD→aD
  C→a　BD→bD　　D→b　Aa→bD

- 解：对每一条产生式α→β，判断其是否满足|β| ≥ |α| 即可(S →ε除外)。

- ∴ G[S]是1型文法。

2型文法

- 设G = (V_N，V_T，P，S)，若P中的每一个产生式α→β满足：α ∈ V_N，β∈ (V_N ∪ V_T)*，则此文法称为2型的或上下文无关的(context-free) 。
- 有时将2型文法的产生式表示为形如：A →β，其中A ∈ V_N，也就是说用β取代非终结符A时，与A所在的上下文无关，因此取名为上下文无关文法。

1型文法仅要求 |β| ≥ |α|

- 立即应用：下列文法是不是2型文法？
- 例1：G = ({S, A, B}, {a, b}, P, S)，其中P是：

S → aB	A → bAA
S → bA	B → b
A → a	B → bS
A → aS	B → aBB

例2：文法G[A]：A → 0R1
0R1 → 00R11　R → 0 | 1
是1型文法，它是2型文法吗？

- 答：是。G的语言是由相同个数的a和b所组成的{a，b}*上的串。

思考：2型文法比1型文法多了什么限制？

3型文法

- 正规文法
  - 右线性文法
    - 设G = (V_N，V_T，P，S)，若P中的每一个产生式的形式都是A → aB或A → a （A→ε），其中A和B都是非终结符，a是终结符，则G是3型文法。
- 特征：产生式的左部必定是非终结符，而右部的首字符必定是终结符。
  - 左线性文法
    - 设G = (V_N，V_T，P，S)，若P中的每一个产生式的形式都是A → Ba或A → a （A→ε），其中A和B都是非终结符，a是终结符，则G是3型文法。

- 立即应用：下面的文法是不是3型文法？
- G = ({S, A, B}, {a, b}, P, S)，其中P由下列产生式组成：

  S → aB　A → bA　S → bA　　B → b

  A → a　B → bS　　A → aS　　B → aB

- 答：是。G的语言是由相同个数的a和b所组成的{a，b}*上的串。

2、3文法类型举例

- 例：下列文法是2型文法吗？是3型文法吗？
- 文法G[S]：
- S → AB　　A → BS|0　　B → SA|1

- 解：(1) 判断其是否符合2型文法定义：对每一条产生式，A →β其中A ∈ V_N
- ∴ G[S]是2型文法。

- (2) 判断其是否符合3型文法定义的依据是什么？
- 对A → aB或A → a，有A ∈ V_N，B ∈ V_N，a ∈ V_T

- ∴由S → AB　A → BS　B → SA可知，G[S]不是3型文法。

正规文法举例：定义标识符

- 例：定义标识符的3型(正规)文法
- 文法G[S]：
  $S \rightarrow lT$
  $S \rightarrow l$
  $T \rightarrow lT$
  $T \rightarrow dT$
  $T \rightarrow l$
  $T \rightarrow d$
- 其中l表示a～z中的任何一英文字母，d表示0～9中的任一数字。
- 文法G[S]可以用来表示标识符定义：以字母开头，而且数字可以出现在除开头处之外的任何地方。

各文法之间的关系

- 四个文法类的定义是逐渐增加限制得到的。因此每一种正规文法都是上下文无关的，每一种上下文无关文法都是上下文有关的，而每一种上下文有关文法都是0型文法。
- 各文法的特点：

文法	特点
0型文法	每个产生式$\alpha\rightarrow\beta$都有：$\alpha\in (V_N \cup V_T)^*$且至少含有一个非终结符，而$\beta\in (V_N \cup V_T)^*$
1型文法	每一个产生式$\alpha\rightarrow\beta$均满足$\lvert\beta\rvert \geq \lvert\alpha\rvert$，仅仅S (单个非终结符）$\rightarrow\varepsilon$除外。(即$\alpha_1 A\alpha_2\rightarrow\alpha_1\beta\alpha_2$)
2型文法	每一个产生式$\alpha\rightarrow\beta$满足：α是一非终结符，$\beta\in (V_N \cup V_T)^*$，(即$A\rightarrow\beta$)
3型文法	每一个产生式的形式都是$A\rightarrow aB$或$A\rightarrow a$，其中A和B都是非终结符，a是终结符。

练习

- 乔姆斯基把文法分成4种类型：0型也叫___；1型也叫__；2型也叫__；3型也叫__。
- 文法G包括四个组成部分：一组终结符号，一组_____，一组产生式，以及一个_____
- 文法可分为几类？各举一例。
- 正规文法产生的语言都可以用上下文无关文法来描述.

单选题 2分

设置

此题未设置答案，请点击右侧设置按钮

文法分为四种类型，即0型、1型、2型、3型。其中3型文法是

A 短语文法

B 上下文有关文法

C 正规文法

D 上下文无关文法

提交

单选题 2分

设置

此题未设置答案，请点击右侧设置按钮

给定语言L为：所有以0开头，后接零个或多个10组成的符号串的集合，则描述它的正规文法G[S]应为

A S→0A A→10A |ε

B S→S10 | 0

C S→0B | 0 B→1S

D 以上都是

提交

本课内容

- 2.4 文法的类型
- 2.5 上下文无关文法及其语法树

上下文无关文法的应用举例1

- 上下文无关文法有足够的能力描述现代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。

- 例如：文法G = ({E}, {+, *, i, (,)}, P, E)，其中P为：
  $E \rightarrow i$　　$E \rightarrow E + E$　　$E \rightarrow E * E$　　$E \rightarrow (E)$

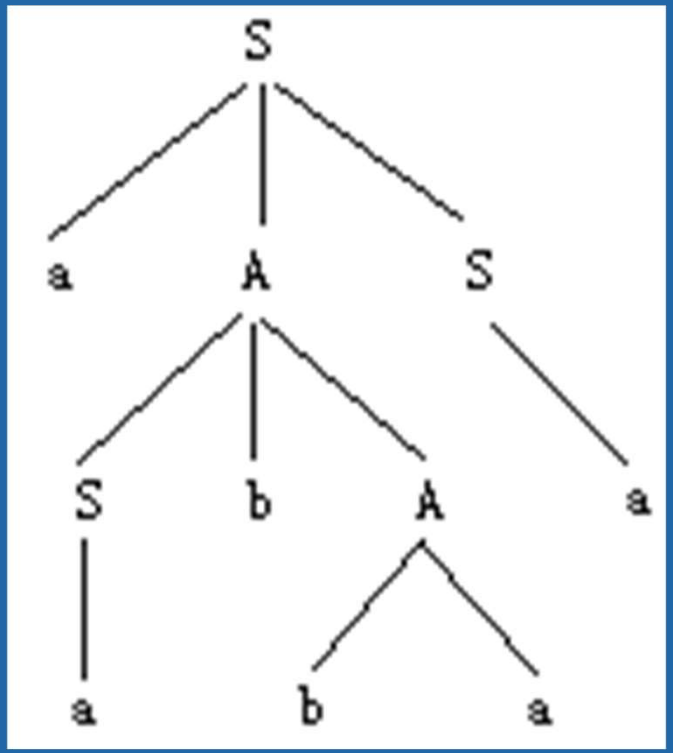
- 这里的非终结符E表示一类算术表达式。i表示程序设计语言中的“变量”，该文法定义了(描述了)由“变量，+，*，(和)”组成的算术表达式的语法结构，即：
  - (1) 变量是算术表达式；
  - (2) 若E_1和E_2是算术表达式，则$E_1 + E_2$，$E_1 * E_2$和(E_1)也是算术表达式。

上下文无关文法的应用举例2

- 又例如：
- 1. 描述语句的产生式：
- <语句> → <条件语句> | <赋值语句> | <循环语句>…
- 2. 描述一种简单赋值语句的产生式为：
- <赋值语句> → i := E
- 3. 描述条件语句的产生式：
- <条件语句> → if <条件> then <语句> | if <条件> then <语句>else<语句>
- 正因为上下文无关文法有足够的能力描述现代程序设计语言的语法结构，所以我们着重关心上下文无关文法形成的语言的句子的分析和分析方法的研究。
- 今后，对"文法"一词若无特别说明，则均指上下文无关文法。

语法树(推导树)

- 前面我们介绍了句型、推导等概念，现在介绍一种描述上下文无关文法的句型推导的直观方法，即语法树(推导树)。
- 给定文法G = (V_N，V_T，P，S)，对于G的任何句型都能构造与之关联的语法树(推导树)。这棵树满足下列4个条件：
- ① 每个结点都有一个标记，此标记是V ($V = V_N \cup V_T$)中的一个符号。
- ② 根的标记是S。
- ③ 若一结点n至少有一个它自己除外的子孙，并且有标记A，则A肯定在V_N中。
- ④ 考察结点n的直接子孙，如果它们的从左到右的次序是结点n_1，n_2，…，n_k，其标记分别为A_1，A_2，…，A_k，那么$A \rightarrow A_1A_2$，…，A_k一定是P中的一个产生式。



语法树举例

•从一个例子来说明某语法树的构造。

•标记**S**的顶端结点是树根，它的直接子孙为**a**、**A**和**S**三个结点，所以**S → aAS**是一个产生式。

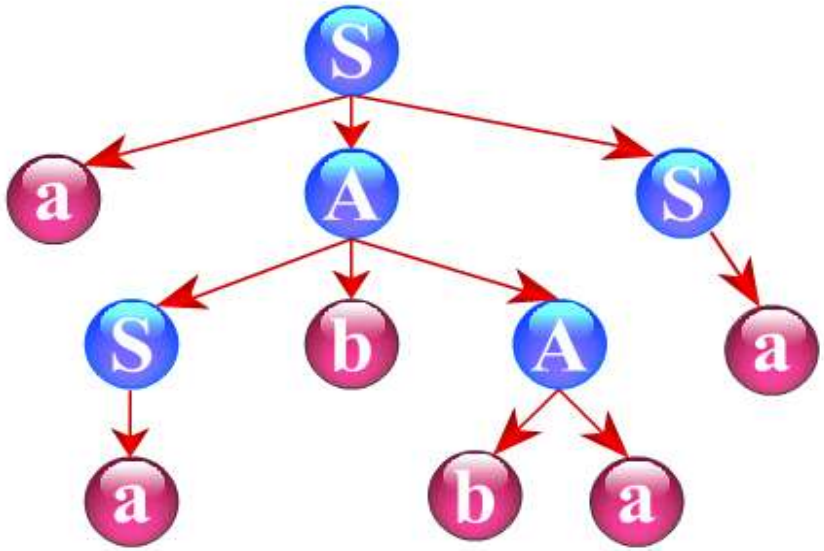
上下文无关文法的语法树

用于描述上下文无关文法句型推导的直观方法

例：G[S]

S→aAS
A→SbA
A→SS
S→a
A→ba

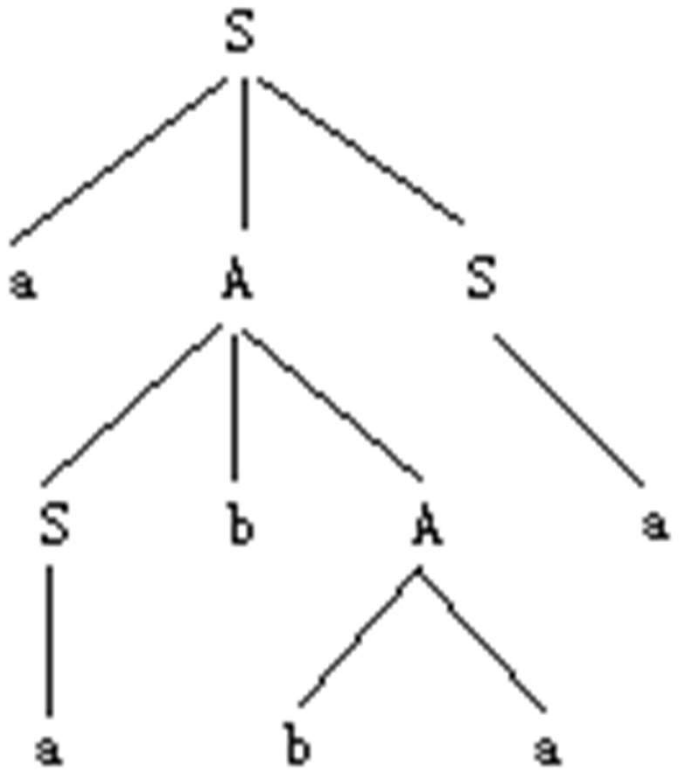
句型aabbaa的语法树（推导树）



叶子结点：树中没有子孙的结点。从左到右读出推导树的叶子标记连接成的文法符号串，为G[S]的句型。也把该推导树称为该句型的语法树。

- A结点至少有一个除它自己以外的子孙，(A的直接子孙为S，b和A)，A肯定是非终结符。
- 从左至右读出推导树的叶子标记，得到的就是句型aabbaa。常常把aabbaa叫做推导树的结果，也把推导树叫做句型aabbaa的语法树。

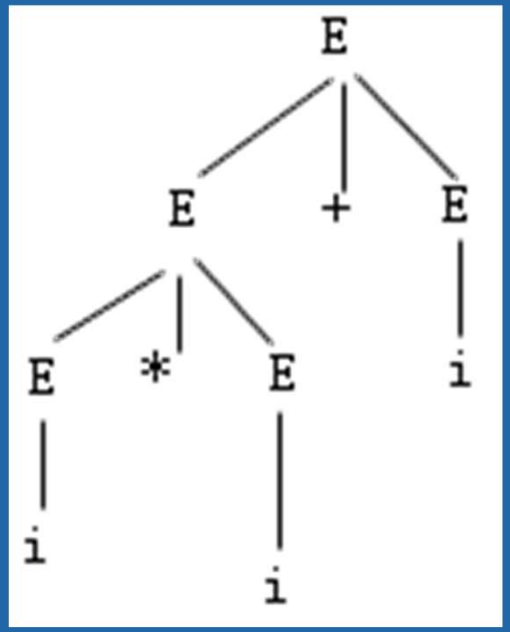
最左推导与最右推导

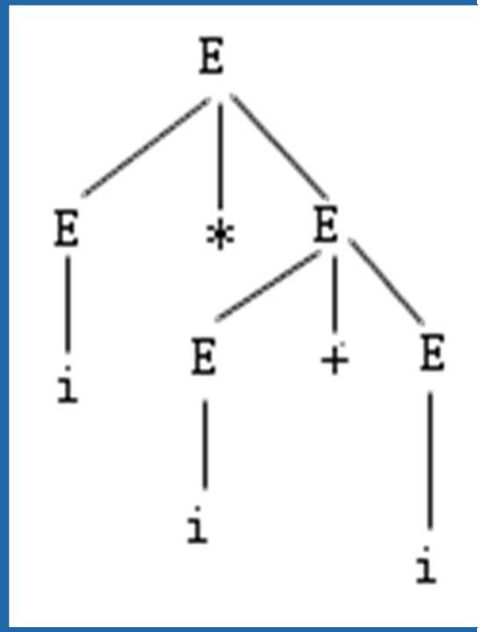


- 设G为上下文无关文法，对于α≠ε，有 $S \overset{*}{\Rightarrow} \alpha$，当且仅当文法G有以α为结果的一棵语法树(推导树)。
- 如果在推导的任何一步“$\alpha \Rightarrow \beta$”中，α和β是句型，都是对α中的最左(最右)非终结符进行替换，则称这种推导为最左(最右)推导。
- 在形式语言中，最右推导常被称为规范推导。由规范推导所得的句型称为规范句型。
- 回顾上例文法G的句型aabbaa的推导过程可能有：
  - 过程1：$S \Rightarrow aAS \Rightarrow aAa \Rightarrow aSbAa \Rightarrow aSbbaa \Rightarrow aabbaa$
  - 过程2：$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$
  - 过程3：$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aSbAa \Rightarrow aabAa \Rightarrow aabbaa$
    - 哪个过程是最左推导？哪个过程是最右推导？

文法的二义性

- 例如，对于文法G = ({E}, {+, *, i, (,)}, P, E)，其中P为：
  E → i　　E → E + E　　E → E * E　　E → (E)
- 句型i * i + i就有两个不同的最左推导一和二，它们所对应的语法树分别如图所示。





- 在一个文法中，如果存在某个句子对应两棵不同的语法树，则这个文法是二义的。
- 或者说，若一个文法中存在某个句子有两个不同的最左(最右)推导，则这个文法是二义的。

- 推导一：E ⇒ E + E ⇒ E * E + E ⇒ i * E + E ⇒ i * i + E ⇒ i * i + i
- 推导二：E ⇒ E * E ⇒ i * E ⇒ i * E + E ⇒ i*i + E ⇒ i * i + i

二义性讨论

- 文法的二义性和语言的二义性是两个不同的概念。
- 因为可能有两个不同的文法G和H，其中G是二义的，但是却有L(G) = L(H)。
- 也就是说，两个文法可能产生的语言是相同的，与文法本身有无二义性无关。

- 如果产生上下文无关语言的每一个文法都是二义的，则说此语言是先天二义的。（习题指导书P28）

- 对于一个程序设计语言来说，常常希望它的文法是无二义的，因为希望对它的每个语句的分析是唯一的，以利于编译器的实现。

无二义性文法的构造

- 人们已经证明，不存在一个算法，它能在有限步骤内，确切判定任给的一个文法是否为二义的。

- 所以需要为无二义性寻找一组充分条件(当然它们未必都是必要的)。例如：在下面的文法中，假若规定了运算符‘+’与‘*’的优先顺序和结合规则：‘*’的优先性高于‘+’，且它们都服从左结合。那么就可以构造出一个无二义文法：

•文法G = ({E}, {+, *, i, (,)}, P, E)，其中P为

- E → i
- E → E + E
- E → E * E
- E → (E)

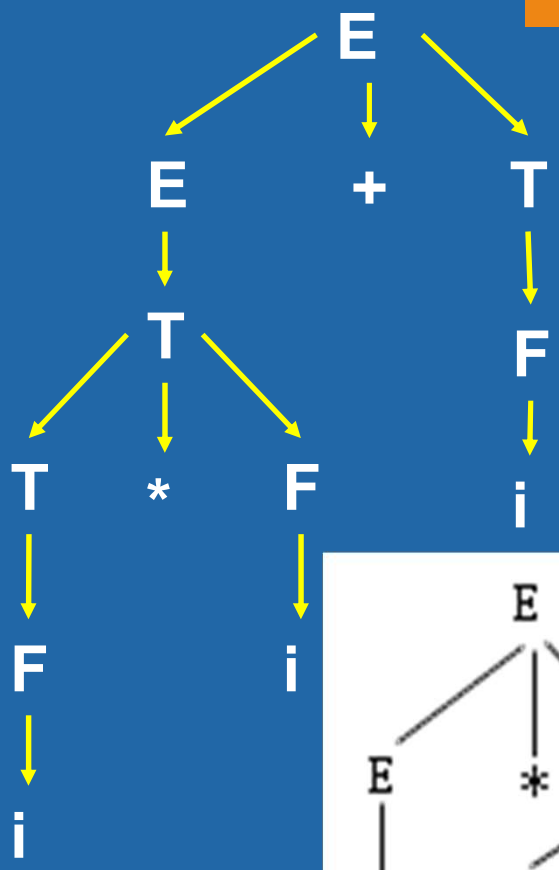
•表达式的无二义文法G[E]：

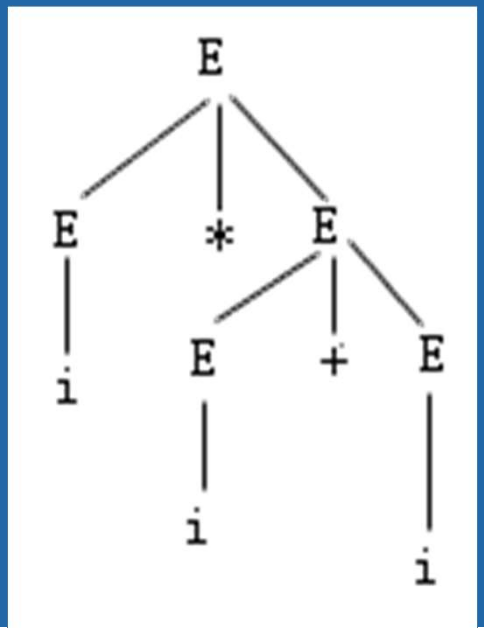
- E → T | E + T
- T → F | T * F
- F → (E) | i

G[E]就是一个无二义性的文法。

无二义性文法的实例检验

- 表达式的无二义文法G[E]：
- E → T | E + T
- T → F | T * F
- F → (E) | i
- 且规定运算符‘+’与‘*’的优先顺序和结合规则：‘*’的优先性高于‘+’，且它们都服从左结合。
- 句型i * i + i是否只存在一棵语法树？





•对比有二义性的文法G[E]

- E → i　E → E + E
- E → E * E　E → (E)

主观题 10分

设置

证明下述文法G［S］是二义的。
S->iSeS|iS|i

正常使用主观题需2.0以上版本雨课堂

作答

课后作业

- PPT后面3页
- 作业格式：
- (1) 在每一次的作业开头，需要写上日期和页码；
- (2) 每道题目的题号要写清楚

主观题 10分

设置

符号串xxyyyx是如下文法G[S]的句子，

S → xB | yA

A → xS | yAA | x

B → yS | xBB | y

(1)构造该句子的分析树（语法树）；

(2)写出生成该句子的最左推导；

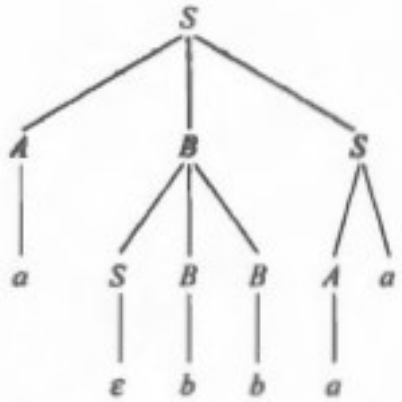
正常使用主观题需2.0以上版本雨课堂

作答

主观题 10分

设置

一个上下文无关文法生成句子abbaa的唯一语法树如下：



(1) 给出该句子相应的最左推导和最右推导。

(2) 该文法的产生式集合 P 可能有哪些元素？

(3) 找出该句子的所有短语、简单短语、句柄。

正常使用主观题需2.0以上版本雨课堂

作答