中山大學
SUN YAT-SEN UNIVERSITY

国家超级计算广州中心
NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

DCS290

Compilation Principle

编 译 原 理

第二章：语言和文法基础

郑馥丹
zhengfd5@mail.sysu.edu.cn

CONTENTS

目 录

1．0型文法

- 通过对产生式施加不同的限制，Chomsky将文法分为**四种类型**：
  - **0型**
  - **1型（上下文有关文法）**
  - **2型（上下文无关文法）**
  - **3型（正规文法）**

- **0型文法(短语文法)**：对任一产生式$\alpha \rightarrow \beta$，都有$\alpha \in (V_N \cup V_T)^+$且至少含有一个非终结符，$\beta \in (V_N \cup V_T)^*$**（与原文法定义一致）**

- **任何文法都是0型文法。**

2．1型文法(上下文有关文法[Context-Sensitive Grammar])

- **1型文法（上下文有关文法）**：0型文法的特例
  - 设文法G=(V_N,V_T,P,S)，对P中的任一产生式α→β，都有**|β|≥|α|**，仅S→ε除外

    例　文法G[S]:

    S→aSBE　　S→aBE　　EB→BE

    aB→ab　　bB→bb　　bE→be　　eE→ee

  - 1型文法产生式的一般形式是**αAγ→αβγ**，α,γ∈V*，A∈V_N，β∈V^+，它表示当A的上文为α且下文为γ时可把A替换成β，因此称1型文法为**上下文有关文法**。

3. 2型文法(上下文无关文法[Context-Free Grammar])

- **2型文法（上下文无关文法[Context-Free Grammar，CFG]）**：1型文法的特例
  - 对任一产生式α→β，都有α∈V_N， β∈(V_N∪V_T)*

    例 文法G[S]：S→AB　A→BS|0　B→SA|1

  - 2型文法产生式的一般形式是：**A→β**，它表示不管A的上下文如何即可把A替换成β，因此被称为上下文无关文法。
  - **通常程序设计语言的文法——2型文法**

4. 3型文法 (正规文法[Regular Grammar])

- **3型文法(正规文法)：** 2型文法的特例
  - 任一产生式α→β的形式都为A→aB或A→a，其中A，B∈V_N，a∈V_T

    例如 文法G[S]：　S→0A|1B|0

    A→0A|1B|0S

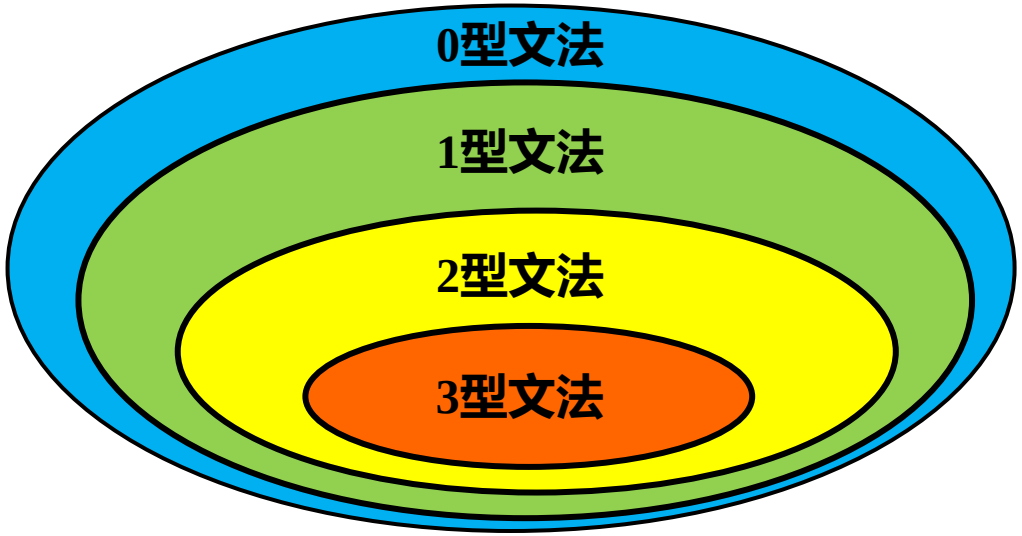
    B→1B|1|0

  - 在程序设计语言中，3型文法通常用来描述单词的结构

小结

文法类别	产生式形式	说明
0型文法 (短语文法)	$\alpha \rightarrow \beta$， $\alpha \in V^+$，且至少含一个非终结符，$\beta \in V^*$	对产生式基本无限制
1型文法 (上下文有关文法)	$\alpha \rightarrow \beta$， $\lvert\beta\rvert \geqslant \lvert\alpha\rvert \geqslant 1$或 $\alpha A \gamma \rightarrow \alpha\beta\gamma, \alpha,\gamma \in V^*$， $A \in V_N$， $\beta \in V^+$	将**A**替换成β时，必须考虑A的上下文α，γ
2型文法 (上下文无关文法)	$A \rightarrow \beta$， $A \in V_N$ ， $\beta \in V^*$	无需考虑**A**在上下文中的出现情况
3型文法 (正规文法)	$A \rightarrow aB$或$A \rightarrow a$， $A, B \in V_N$ ， $a \in V_T$	产生式全部是规定的形式

四种文法之间的逐级“包含”关系



CONTENTS

目 录

1．上下文无关文法 （2型）

- 上下文无关文法有足够的能力描述现今程序设计语言的语法结构
- 因此，我们只关心上下文无关文法形成的语言中的句子的分析

例1：算术表达式：E→i|E+E|E*E|(E)

例2：<赋值语句>→i:=E

<条件语句>→if <条件>then <语句>

| if <条件> then <语句> else <语句>

2. 规范推导和规范句型

- 如果在推导的任何一步α→β，其中α、β是句型，都是对α中的最左非终结符进行替换，则称这种推导为**最左推导**；同理，如果是总对最右的非终结符进行替换，则称为**最右推导**
- **最右推导**被称为**规范推导**
- 由规范推导所得的句型称为**规范句型（右句型）**

例 文法G：E→E+T|T

T→T×F|F

F→(E)|i

句子i+i×i的推导过程如下：

最左推导： E⇒E+T⇒T+T⇒F+T⇒i+T⇒i+T×F⇒i+F×F⇒i+i×F⇒i+i×i

最右推导： E⇒E+T⇒E+T×F⇒E+T×i⇒E+F×i⇒E+i×i⇒T+i×i⇒F+i×i⇒i+i×i

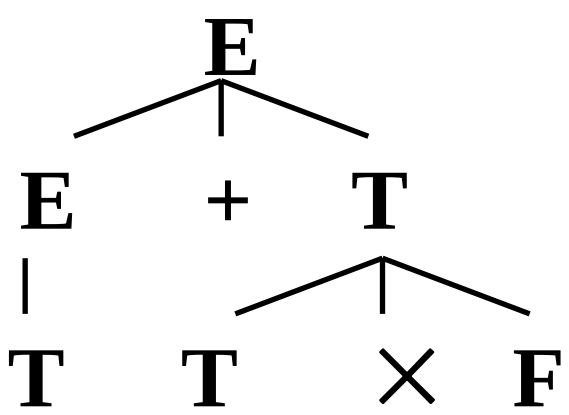
3. 语法树(推导树)[Parse Tree]

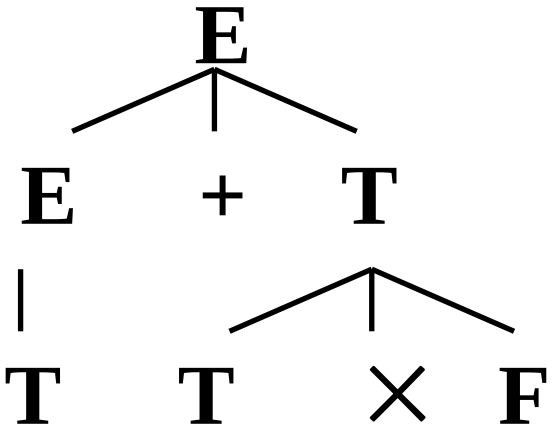
- 作用：直观地描述上下文无关文法的**句型推导过程**。给定文$G=(V_N,V_T,P,S)$，对于G的任何句型都能构造与之关联的语法树

文法G：E→E+T|T
T→T×F|F
F→(E)|i

E=>E+T=>E+T×F=>T+T×F

E=>E+T =>T+T =>T+T×F





从左到右读出叶子结点得到的符号串，为文法的句型。也把该语法树称为该句型的语法树。

从语法树中看不出句型中的符号被替代的顺序

3. 语法树(推导树)[Parse Tree]

- 语法树定义:
  - 给定文法G=(V_N,V_T,P,S)，若一棵树满足下列4个条件，则称此树为G的语法树:

1. 每个结点都有一个标记，此标记是V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，则肯定A∈V_N
4. 如果结点n有标记A，其直接子孙结点从左到右的次序是n_1，n_2，…，n_k，其标记分别为A_1，A_2，…，A_k，那么A→$A_1A_2…A_k$一定是P中的一个产生式

随堂练习(2)

S→SS*|SS+|a

通过此文法如何生成串**aa+a***，并为该串构造**语法树**。

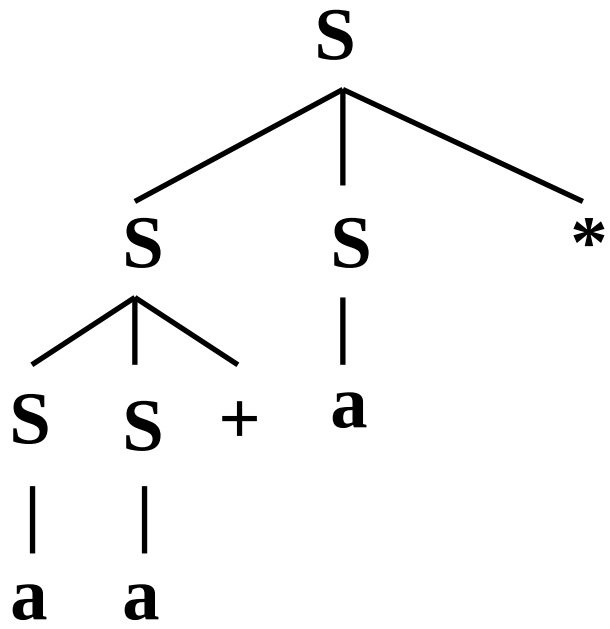
S ⇒SS*

⇒SS+S*

⇒aS+S*

⇒aa+S*

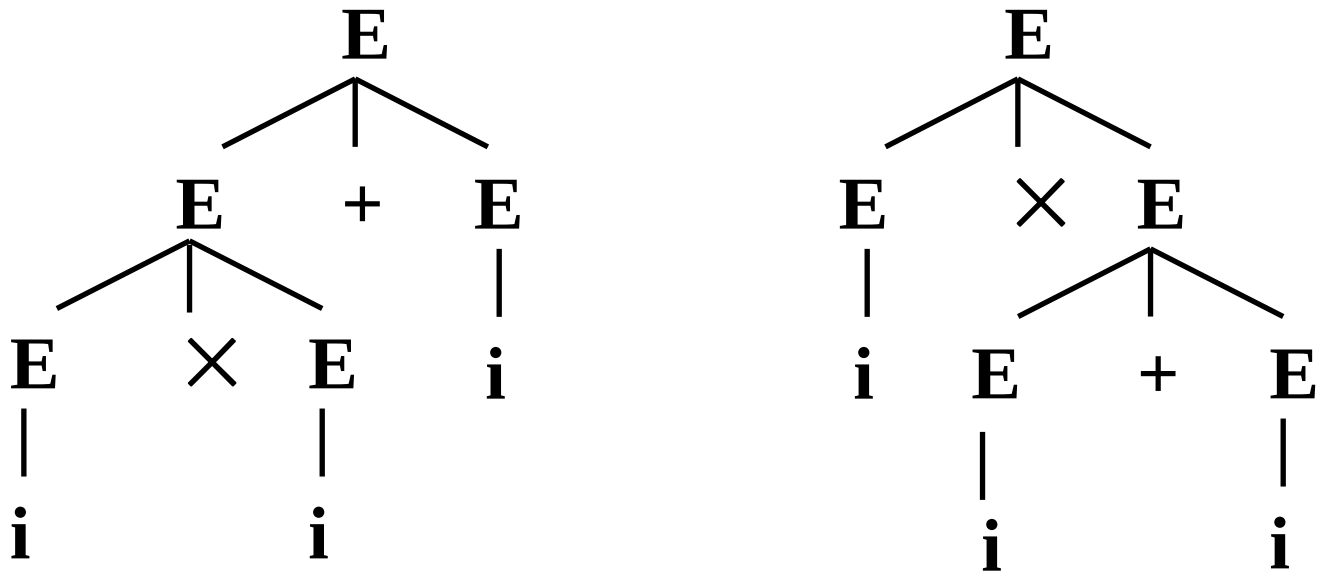
⇒aa+a*



4. 文法的二义性[Ambiguity]

- 文法G：E→E+E|E×E|（E）|i

  句子 i×i+i 对应的语法树



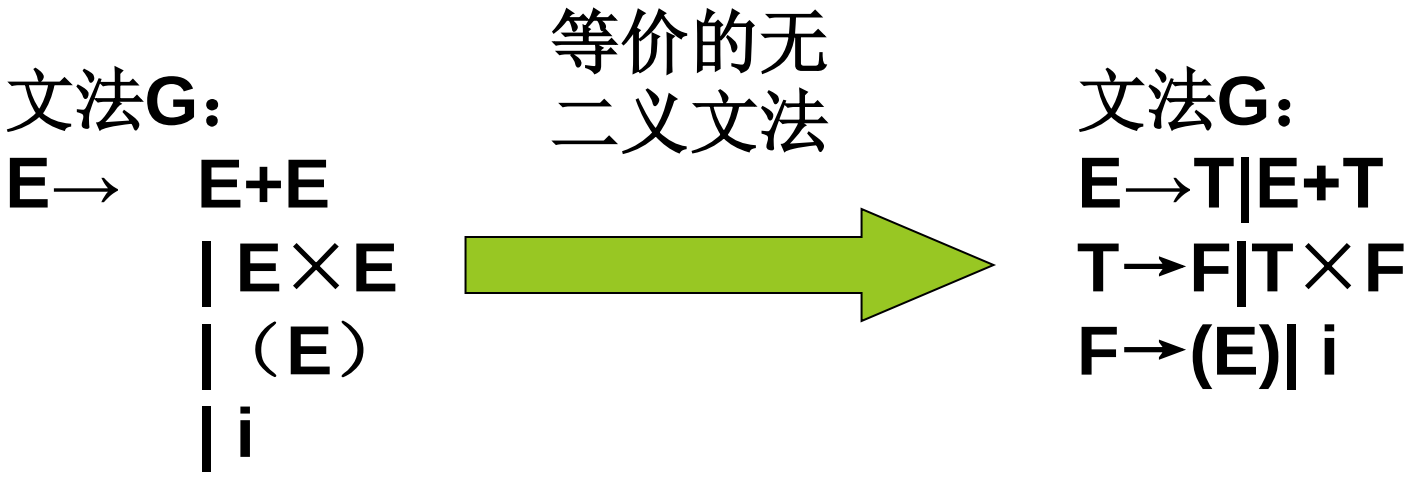
存在两个不同的最左推导：

推导1：$\underline{E} \Rightarrow \underline{E}+E \Rightarrow \underline{E}\times E+E \Rightarrow i\times\underline{E}+E \Rightarrow i\times i+\underline{E} \Rightarrow i\times i+i$

推导2：$\underline{E} \Rightarrow \underline{E}\times E \Rightarrow i\times\underline{E} \Rightarrow i\times\underline{E}+E \Rightarrow i\times i+\underline{E} \Rightarrow i\times i+i$

4. 文法的二义性[Ambiguity]

- 定义：如果一个文法存在某个句子对应**两棵不同的语法树**，则说这个文法是**二义的**
- 二义性文法存在某个句子，它有**两个不同的最左（右）推导**



任何一个二义性的文法，都可以转换成一个等价的无二义性文法。

随堂练习(3)

- 考虑文法 S→aSbS|bSaS|ε，试说明此文法是二义性的。
- 提示：可以从对于句子abab有两个不同的最左推导来说明。
- 解：
  - **S⇒aSbS⇒aεbS⇒aεbaSbS⇒aεbaεbS⇒aεbaεbε⇒abab**
  - **S⇒aSbS⇒abSaSbS⇒abεaSbS⇒abεaεbS⇒abεaεbε⇒abab**
  - **存在两个不同的最左推导。**
  - **因此文法是二义的。**

CONTENTS

目 录

1. 句型分析

- 对于程序设计语言来说，句型分析就是一个识别输入符号串是否为语法上正确的程序的过程。
- **任务**：句型分析就是**识别一个符号串是否为某文法的句型**，是某个推导的构造过程。
- 由文法开始符号S推导出的符号串α（即S = > α），称为文法G[S]的句型。
- **若能根据该文法构造出该符号串的语法树，则该符号串就是该文法的句型。**
- 句型分析算法采用**从左到右的分析算法**，即总是从左到右地识别输入符号串
- 句型的分析算法分类
  - **自上而下分析法 (Top-Down parsing)**
  - **自下而上分析法 (Bottom-Up parsing)**

2. 自上而下的分析方法[Top-Down parsing]

- 定义：
  - **从文法的开始符号**出发，**反复使用文法的产生式**，寻找与输入符号串匹配的推导。
- 语法树的构造：
  - 将文法的**开始符号作为语法树的根**，**向下**逐步建立语法树，使语法树的末端结点符号串正好是输入符号串。

例：文法G：

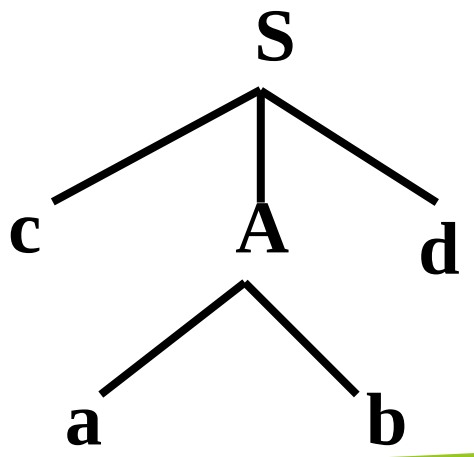
S → cAd

A → ab

A → a

识别输入串w=cabd是否为该文法的句子

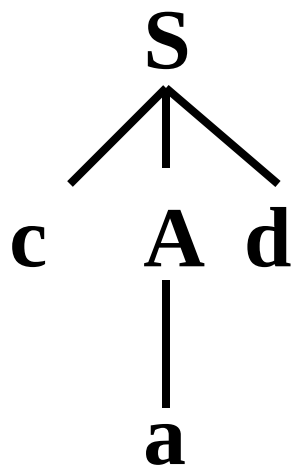
推导过程：**S =>cAd =>cabd**



2. 自上而下的分析方法[Top-Down parsing]

- 自上而下方法的主要问题

例：对输入串cabd自上而下构造语法树的另一过程



不成功！

不成功的原因是

选错产生式A→a

S → cAd
A → ab
A → a

自上而下分析的**主要问题是选择产生式**:

假定要被代换的最左非终结符号是B，且有n条规则：$B \rightarrow A_1|A_2|...|A_n$，那么如何确定用哪个右部去替代B?

3. 自下而上的分析方法 [Bottom-Up parsing]

- 定义：
  - **从输入符号串开始**，逐步进行**归约**，直至归约到文法的开始符号。
- 语法树的构造：
  - 从输入符号串开始，以它**作为语法树的末端结点符号串**，**自底向上**的构造语法树。

归约过程：　**cabd |-cAd |-S**

用“|-”表示归约，下划线部分为被归约符号

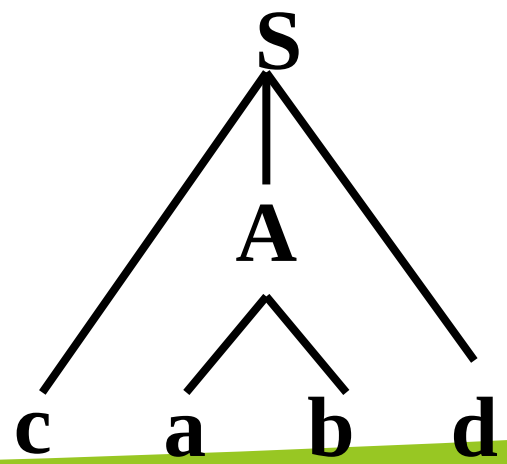
例：文法G：

S → cAd

A → ab

A → a

识别输入串w=cabd是否为该文法的句子



3. 自下而上的分析方法 [Bottom-Up parsing]

- 自下而上分析的主要问题

例：对输入串cabd的两种归约过程

（1）**cabd|-cAd|-S**　　归约到开始符

（2）**cabd|-cAbd**　　不能归约到开始符

S → cAd
A → ab
A → a

– 在自下而上的分析方法中，每一步都是从当前串中选择一个子串加以归约，该子串暂称“**可归约串**”。

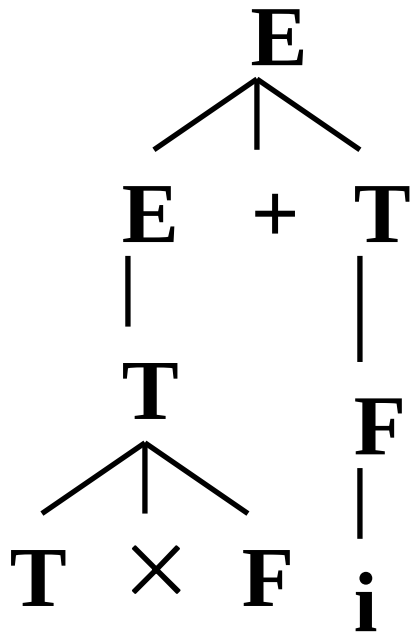
– 自下而上分析的**主要问题：如何确定“可归约串”——“句柄”。**

4. 短语[phrase]，直接短语和句柄[handel]（重要！）

- 设αβδ是文法G[S]中的一个句型，如果有S=*>αAδ且**A=+>β**，则称β是句型αβδ相对于非终结符A的**短语[phrase]**。
- 特别的如有A=>β，则称β是句型αβδ相对于规则A→β的**直接短语（简单短语）**。
- 一个句型的最左直接短语称为该句型的**句柄[Handle]**。
- **句柄就是“可归约串”。**
- 注意：在短语的定义中包括了三个条件，都必须满足：
  - （1）αβδ是文法的一个句型；
  - （2）S=*>αAδ;
  - （3）A=+>β 。
    - ✓ （1）（2）说明αβδ 、αAδ都必须是句型；
    - ✓ （2）（3）说明，将αβδ中的β归约A后，得到的αAδ一定要是句型。假如符号串β ，将其归约成A后得到的符号串不能由开始符号推出，则β不是短语。

4. 短语[phrase]，直接短语和句柄[handel]（重要！）

例：文法G[E]: E→E+T|T，T→T×F|F，F→（E）|i的一个句型是**T×F+i**，相应的语法树见右图：



1. 因为E=*> T+i 且 T=>T×F，所以**T×F**是句型相对于T的短语，且是相对于T→T×F的直接短语；
2. 因为E=*> T×F+F 且 F=>i，所以**i**是句型相对于F的短语，且是相对于F→i的直接短语；
3. 因为E=*> E 且E=⁺>T×F+i，所以**T×F+i**是句型相对于E的短语；
4. **T×F**是最左直接短语，即句柄。

注意：虽然**F+i**是句型T×F+i的一部分，但**不是短语**，因为尽管有E=⁺>F+i，但是**不存在从文法开始符E=* >T×E的推导**

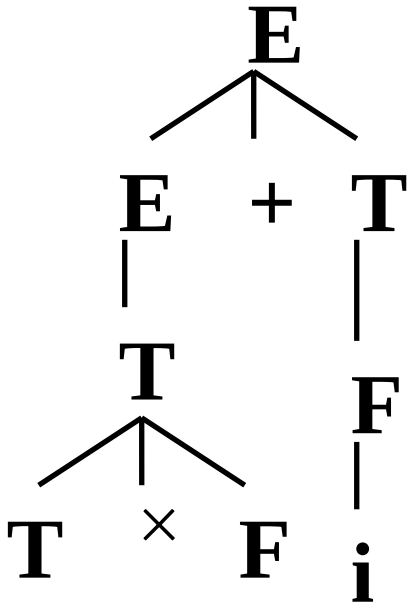
若有S=*>αAδ且A=⁺>β，则称β是句型αβδ相对于非终结符A的短语。

5. 短语与语法树

- 从句型的语法树上很容易找出句型的短语——语法树中每棵子树的末端结点构成相对于子树根的短语

对于句型**T × F+i**来说：

- ✓ 五棵子树对应五个**短语**（其中2个重复）：T × F，i，T × F+i
- ✓ 两层子树的末端结点构成直接短语，两棵两层子树对应两个**直接短语**：T×F，i
- ✓ 位于最左边的两层子树的末端结点构成**句柄**：T×F



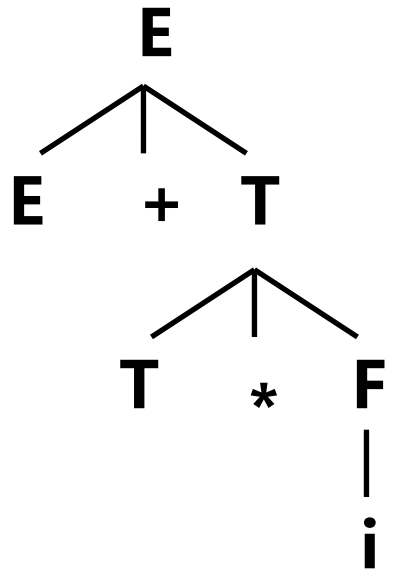
5. 短语与语法树

文法G[E]:

- **E→T|E+T|E-T**
- **T→F|T*F|T/F**
- **F→(E)|i 的一个句型是E+T*i**

3棵子树，对应3个短语:

- **E+T*i**
- **T*i**
- **i （直接短语、句柄）**



5. 短语与语法树

文法G[E]:E→E+T|T，T→T×F|F，F→ (E)|i的一个句型是F×i + i

- **7棵子树,对应有7个短语:**

F×i + i (E=*>E, E=*>F×i + i)

F×i (E=*>E+T, E=*>F×i)

F×i (E=*>T+T, T=*>F×i)

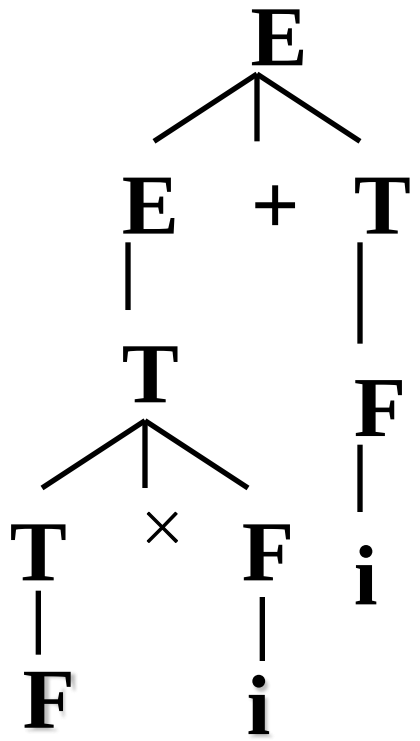
句柄 ←

F (E=*>T×i + i, T=*>F)(直接短语)

i (E=*>F×F+i, F=*>i)(直接短语)

i (E=*>F×i+T, T=*>i)

i (E=*>F×i+F, F=*>i)(直接短语)



5. 短语与语法树

- 小结：找短语、直接短语、句柄的方法：
  - 找短语、直接短语、句柄时，可以通过**语法树**来进行。
  - 句型或句子才有所谓的短语等，所以要**先确定句型或句子**，然后再找它的短语、直接短语、句柄。
  - **要求某一句型的句柄等，语法树只需要分析到该句型。**

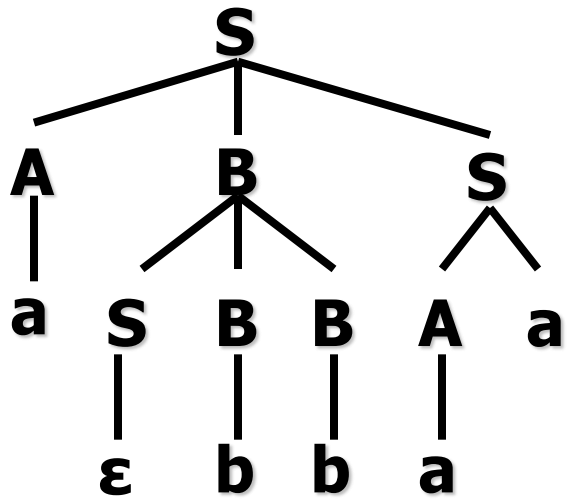
随堂练习(4)

求句子abbaa的短语、直接短语、句柄。

8棵子树，对应8个短语：

- abbaa
- a (直接短语)
- bb
- aa
- ε (直接短语)
- b (直接短语)
- b (直接短语)
- a (直接短语)

(句柄)



CONTENTS

目 录

1. 有关文法的实用限制

- 有害规则和多余规则
  - 有害规则：U→U，无用且引起文法的二义性
  - 多余规则：所有句子推导都用不到的规则，表现在：
    - ✓ 不可到达：文法中某些非终结符不在任何规则的右部出现，所以任何句子的推导中都无法用到它。
    - ✓ 不可终止：不可推导出终结符号串的非终结符。

例：文法G[S]：

（1）S→Be　~~（2）B→Ce~~　（3）B→Af

（4）A→Ae　（5）A→e　~~（6）C→Cf~~

~~（7）D→f~~

多余规则为：

- 不可到达（7）
- 不可终止（6）
- （2）也是不可终止的

1. 有关文法的实用限制

- 为避免有害规则和多余规则：
  - 非终结符A必须在某句型中出现。即有$S=*>\alpha A\beta$，其中α,β均属于$(V_N \cup V_T)^*$
  - 必须能从A推出终结符号串t来。即$A=*>t$,其中$t \in V_T^*$

2. 上下文无关文法中的ε规则

- 上下文无关文法中允许使用A→ε产生式，A→ε称为ε规则

第二章课后作业

- 1. 令文法G[E]为：

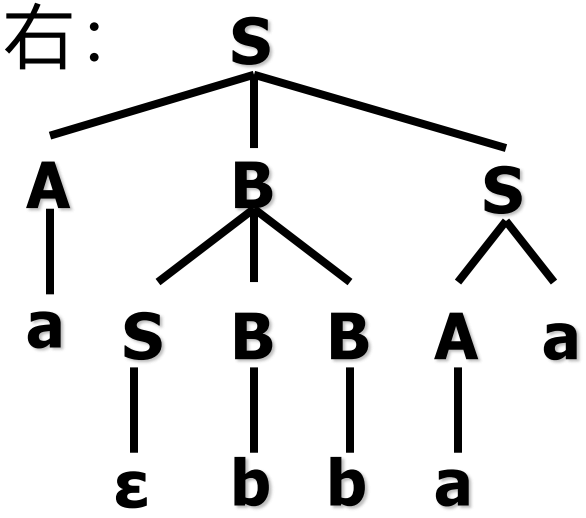
  E→T|E+T|E-T

  T→F|T*F|T/F

  F→(E)|i

  证明E+T*F是它的一个规范句型，指出这个句型的所有短语、直接短语和句柄。

- 2. 一个上下文无关文法生成句子abbaa的唯一语法树如右：



  （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、简单短语和句柄。

第二章课后作业

- 提交要求：
  - 文件命名：学号-姓名-第二章作业；
  - 文件格式：.pdf文件；
  - 手写版、电子版均可；若为手写版，则拍照后转成pdf提交，但**须注意将照片旋转为正常角度，且去除照片中的多余信息**；电子版如word等转成pdf提交；
  - 提交到超算习堂（第二章作业）处；
  - 提交ddl：**3月11日晚上12:00**；
  - **重要提示：不得抄袭！**

课前/课后 复习/提问

1. 根据右侧的文法推导出句子“他是王明”

〈句子〉

⇒ 〈主语〉〈谓语〉

⇒ 〈代词〉〈谓语〉

⇒ 他〈谓语〉

⇒ 他〈动词〉〈直接宾语〉

⇒ 他是〈直接宾语〉

⇒ 他是〈名词〉

⇒ 他是王明

- <句子>::=<主语><谓语>
- <主语>::=<代词>｜<名词>
- <代词>::= 我｜你｜他
- <名词>::= 王明｜大学生｜工人｜英语
- <谓语>::=<动词><直接宾语>
- <动词>::= 是｜学习
- <直接宾语>::=<代词>｜<名词>

2. 进行如下字符串运算

（1）x=“abc”, y=“defg”, 则xy=?

xy=“abcdefg”

（2）x=“abc”, 则x^2 =?

x^2=“abcabc”

（3）A={a, b, c} B={d, e}, 则AB=?

AB={ad, ae, bd, be, cd, ce}

（4）B={d, e}, 则B^2=?

B^2 =BB={d, e} {d, e}={dd, de, ed, ee}

课前/课后 复习/提问

3. 文法G： S→0S1， S→01

由文法G推到出句子00000011111

S ⇒0S1

⇒00S11

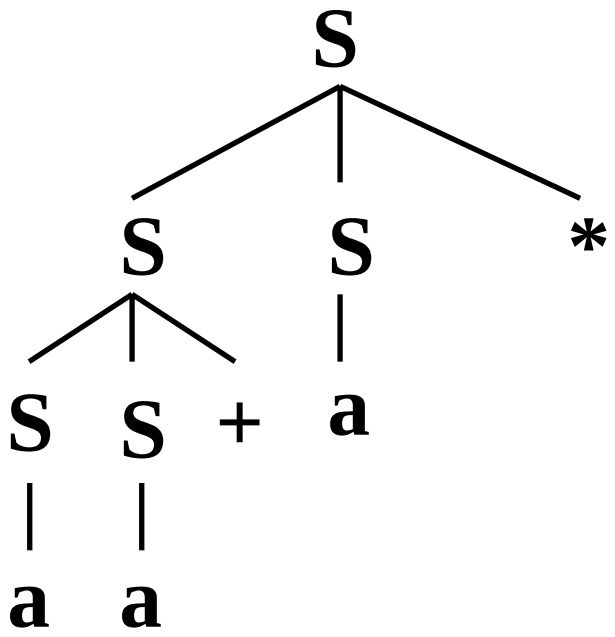
⇒000S111

⇒0000S1111

⇒00000011111

课前/课后 复习/提问

- 4. 找出句型aa+a*的短语、直接短语和句柄



5棵子树，对应5个短语：

- ✓ aa+a*
- ✓ aa+
- ✓ a（直接短语）
- ✓ a（直接短语）**（句柄）**
- ✓ a（直接短语）