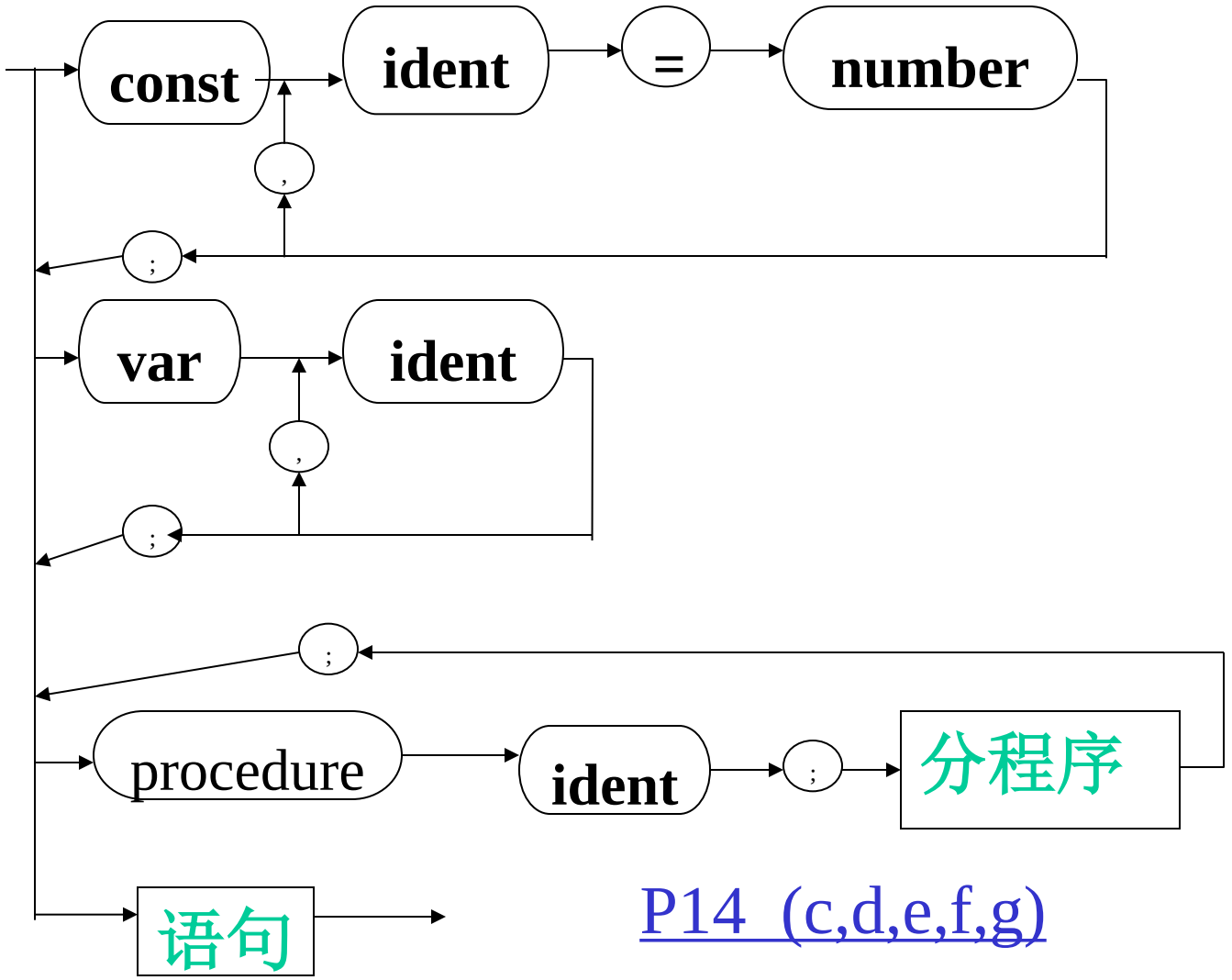
编译原理

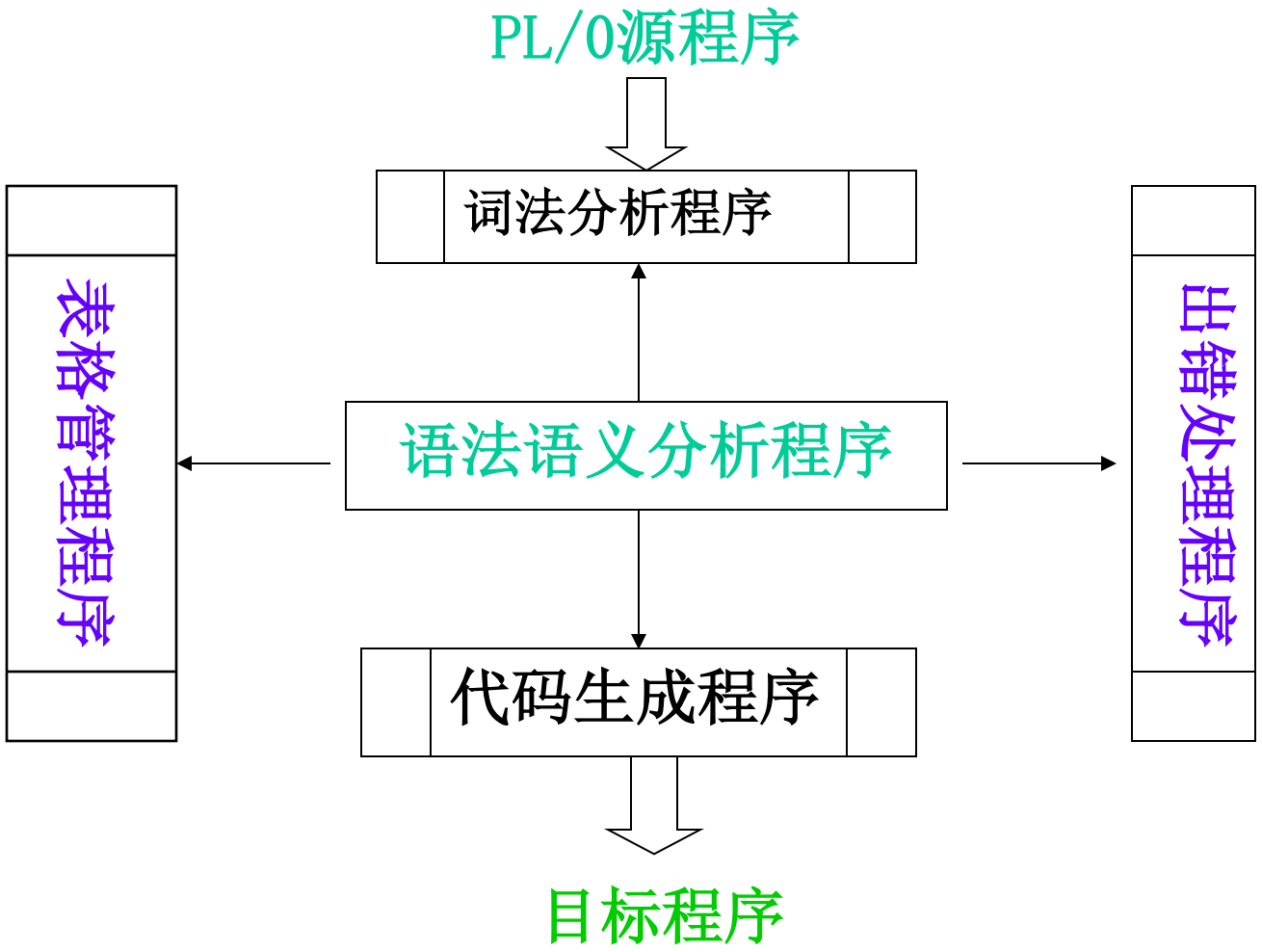
PL/0语言语法图描述

分程序



P14 (c,d,e,f,g)

PL/0编译程序的结构



PL/0编译程序的总体设计

- 其编译过程采用一趟扫描方式
- 以语法、语义分析程序为核心

  词法分析程序和代码生成程序都作为一个过程，当语法分析需要读单词时就调用词法分析程序，而当语法、语义分析正确，需要生成相应的目标代码时，则调用代码生成程序。
- 表格管理程序实现变量，常量和过程标识符的信息的登录与查找。
- 出错处理程序，对词法和语法、语义分析遇到的错误给出在源程序中出错的位置和与错误 性质有关的编号，并进行错误恢复。

PL/0编译程序的结构

- PL/0编译程序的组成：

  PL/0程序有18个过程或函数组成。（见表2.1）

- 过程、函数嵌套层次图（见图2.3）
- PL/0编译程序总流程图（见图2.4）

PL/0编译程序的词法分析

PL/0的词法分析程序Getsym是一个独立的过程

Getsym功能是 ：对单词进行分类分析

1、Getsym流程图（见图2.5）

2、流程图分析

3、Getch程序流程图（见图2.6）

PL/0编译程序的语法分析

PL/0的语法分析

1、方法：采用了自顶向下的递归子程序法

2、语法依据：文法的规则

3、任务：分析在词法分析基础上的单词符号结构是否符合给定的文法规则

4、分析步骤（粗）

开始→非终结符→调用相应处理程序→进入语法单元→ 沿语法图箭头方向分析→遇终结符→判断单词与图中匹配吗？ →匹配→单元翻译程序→下一个分析→程序结束符 ‘.’

5、PL/0语法调用关系图（见图2.7）

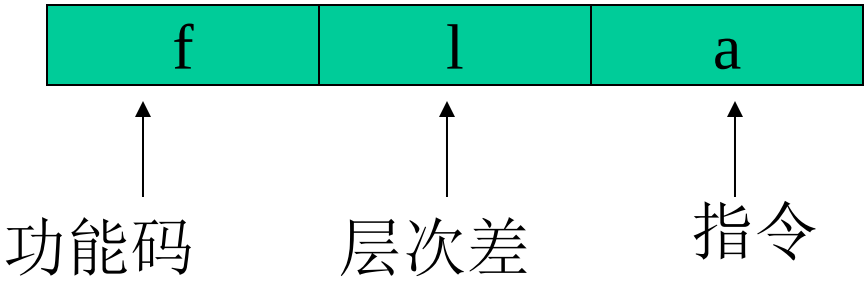
6、Block过程的流程图（见图2.8）

Block过程是处理说明部分和程序体部分

五、PL/0目标代码生成

1、PCODE代码

PCODE代码是PL/0所产生的目标代码。格式为：



变量、过程被调用的分程序与说明该变量、过程的分程序之间的层次差

2、源程序和目标程序清单

六、PL/0语法错误处理 （了解）

方法：
- 指出错误位置，并校正，如逗号、括号等
- 错误局限在语法当中。

七、PL/0存储分配

更详细的解释请参考教材

编译原理

文法和语言

- 编译程序研究如何将源语言程序翻译为目标语言程序；
- 让计算机熟悉和掌握源语言和目标语言；
- 怎样让计算机掌握语言的语法和语义？
- 对语法和语义进行形式化描述
- 文法是对语法进行形式化描述的工具
- 对文法和语言进行形式化定义

- 文法的形式化定义
- 语言的形式化定义

文法和语言

- 构造编译程序的算法是从研究源程序及目标程序产生的，首先找到源语言的形式描述，根据这种描述，构造出相应的分析加工程序。
- 程序设计语言包括语法和语义两方面。
- 语法是一组规则，可用来产生合乎语法的程序，也可用来分析一个程序是否合乎语法。
  - A:=B+C
- 程序设计语言的语义包括静态语义和动态语义。静态语义是一系列限定规则，用来确定哪些合乎语法的程序是正确的（类型匹配、变量作用域审查等）；动态语义称为运行语义或执行语义，表示程序要做什么，要计算什么。

文法和语言

一、文法的概念

二、符号和符号串

三、文法和语言的定义

四、文法的类型

五、上下文无关文法及其语法树

六、句型的分析

七、有关文法的一些限制

文法的概念

语言
- 语法：是一组规则，定义符号如何排列，排列与符号含义无关。
- 语义 ：研究语法的含义
  - 静态语义
  - 动态语义

文法是阐述语法的一个工具

一、文法的概念 （写出以下语言的文法）

“你是大学生” 对　　“我是教师”对

“我大学生是”错　　“我学习大学生”对

〈句子〉::=〈主语〉〈谓语〉

〈主语〉::=〈代词〉|〈名词〉

〈代词〉::=我|你|他

〈名词〉::=王明|大学生|教师|英语

〈谓语〉::=〈动词〉〈直接宾语〉

〈动词〉::=是|学习

〈直接宾语〉::=〈代词〉|<名词>

文法的概念

推导： 我是教师

〈句子〉 ⇒ 〈主语〉〈谓语〉

⇒ 〈代词〉〈谓语〉 ⇒ 我〈谓语〉 ⇒

我〈动词〉〈直接宾语〉 ⇒我是〈名词〉

⇒ 我是教师

符号和符号串

- 汉语是由所有合法的汉语句子组成的集合，英语是由所有合法的英语句子组成的集合；
- C、PASCAL等程序设计语言是由所有C、PASCAL程序组成的集合；
- 程序是由一些基本符号组成的；
- 从字面上看，每个程序都是一个基本符号串；
- 设有一个基本符号集， C、PASCAL等程序设计语言可看成是在这个基本符号集上定义的，按一定规则构成的一切基本符号串组成的集合。

符号和符号串

为给出语言的形式化定义，先讨论一些有关概念：

1、字母表—符号集：是字母的有穷非空集合。

汉语字母表包括：

汉字、数字、标点符号等

Pascal语言字母表包括：

字母、数字、若干专用符号以及Begin、if等保留字。

符号和符号串

2、符号串—字母表的符号组成的任何有穷序列。

例：$\Sigma=\{0, 1\}$ — 字母表

符号串有：

0，1，00，01，10，11….

例：$\Sigma=\{a, b, c\}$ — 字母表

符号串有：

a，b，c，ab，aaca

3、符号串的长度：符号串x有m个符号，则长度就为m，表示|x|=m

如： ababa 则长度是

5

4、空符号串：用 ε 表示，长度为0（不含任何符号）

5、符号串的运算：

（1） 符号串的头和尾

若z=xy，则x是z的头，y是z的尾。

例：设z=abc ，则z的头是

ε ， a， ab， abc

则z的尾是

ε ， c， bc， abc

（2）符号串的固有头和固有尾

若z=xy符号串，x非空，则y是固有尾；

若y非空，则x是固有头。

例：设z=abc，则z的固有头是

ε ， a， ab

则z的固有尾是

ε ， c ， bc

（3）符号串的连接：

设x，y是符号串，连接xy是y符号写在x符号之后。

例：x=ab，y=MN　　则xy=

abMN

若符号串x ，则 ε x =？　x ε = ？

显然： **ε x=x ε =x**

（4）符号串的方幂：

设x是符号串，则z=xx.....xx，称z为x的方幂，

记$z=x^n$。

因此 $x^0=\varepsilon$，$x^1=x$，$x^2=xx$，$x^3=xxx$

显然n>0时，有$x^n = x \cdot x^{n-1} = x^{n-1} \cdot x$

（5）符号串的集合：

若集合A中的一切元素都是某字母表上的符号串，则称A为该字母表上的符号串集合。

两个符号串集合A、B乘积定义：

$AB=\{xy \mid x\in A且y\in B\}$

例：A={a,b}，B={c,d}

则AB=

{ac，ad，bc，bd}

- （6）闭包（$\sum^*$）
- 字母表$\sum$，用$\sum^*$ 表示$\sum$上所有有穷长的串集合，$\sum^*$称为$\sum$的闭包。
- 例：字母表$\sum=\{0,1\}$
- 则$\sum^*$=

$=\sum^0 \cup \sum^1 \cup \sum^2 \cup \ldots \cup \sum^n$

$=\{\varepsilon,0,1,00,01,10,11,000,001,010\ldots\ldots\}$

〈7〉 正闭包(Σ^+)

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots \cup \Sigma^n$

Σ^+称Σ的正闭包。

显然：$\Sigma^* = \Sigma^0 \cup \Sigma^+$

$\Sigma^+ = \Sigma\Sigma^* = \Sigma^*\Sigma$

三、文法和语言的形式定义

（用以上术语对文法的概念进行形式化）

1、规则（重写规则、产生式、生成式）

形如$\alpha \rightarrow \beta$ 或$\alpha ::= \beta$

α 称规则的左部，β称规则的右部。

2、文法的定义

〈1〉文法G定义为四元组（V_N，V_T，P，S）

其中：V_N —— 非终结符号集

V_T —— 终结符号集

P —— 产生式（规则）

S —— 开始符号或称作识别符号，它是一个非终结符，至少要在一条规则中作为左部出现。

规定：（1）V_N，V_T，P是有穷非空集合；

（2）$V_N \cap V_T = \phi$　（不含公共元素）

例1　文法$G=(V_N, V_T, P, S)$，

其中 $V_N=\{S\}$，$V_T=\{0, 1\}$，

$P=\{S \rightarrow 0S1,\ S \rightarrow 01\}$。

非终结符集中只含一个元素S；

终结符集由两个元素0和1组成；

有两条产生式；开始符号是S。

想：从非终结符可推出哪些符号串？

答案：S＝{01，0011，000111，00001111…}

例2　文法G =（V_N ，V_T，P，S）

其中 V_N ={标识符，字母，数字}

V_T= {a,b ,c,…,x,y,z,0,1,…,9}

P = { <标识符> →〈字母〉

<标识符> → <标识符><字母〉

<标识符> → <标识符><数字〉

<字母> →a

<字母> →b

…

<字母> →z

<数字> →1

…

<数字> →9 }

S＝ <标识符>

“<　　>”表示非终结符

想：标识符可以是哪些字符串？

<2> 文法G的第二种表示法：

上例1改为： G: S → 0S1

S → 01

<3> 文法G的第三种表示法：

上例1改为： G[S]: S → 0S1

S → 01

一般约定，第一条产生式的左部是识别符号；用尖括号括起来的是非终结符号，不用尖括号括起来的是终结符号，或者用大写字母表示非终结符号，小写字母表示终结符号。

3、直接推导的定义

<1> $\alpha \rightarrow \beta$ 是文法G=（V_N，V_T，P，S）的规则，

γ和δ是V*中的任意符号，若有符号串V，W满足：

$V=\gamma\alpha\delta$，$W=\gamma\beta\delta$

则说 V是直接产生W

或 W是V的直接推导

或 W直接规约到V

记作 $V \Rightarrow W$

例3： 在例1中

V＝0S1，W=0011

W 是否V的直接推导

G: S → 0S1
S → 01

V=S，W＝0S1

W 是否V的直接推导

V=0S1，W=00S11

W 是否V的直接推导

例3： 在例1中

V＝0S1，W=0011

V是否W的直接推导

直接推导：0S1 ⇒**0011**

使用规则：S →01

γ＝0，δ＝1，

α＝S， β＝01

V=S，W＝0S1

V是否W的直接推导

直接推导：S ⇒ 0S1

V=0S1，W=00S11

V是否W的直接推导

直接推导： 0S1⇒**00S11**

规则： S → 0S1

γ＝ε ， δ＝ε

α＝S， β＝0S1

规则： S → 0S1

γ＝0 ， δ＝1

α＝S， β＝0S1

(2)若存在直接推导的序列：

$V=W_0 \Rightarrow W_1 \Rightarrow W_2 \Rightarrow \ldots W_n = W \quad (n>0)$

则 称V推导W （或W规约到V），记 $V \overset{+}{\Rightarrow} W$

(3)若有 $V \overset{+}{\Rightarrow} W$ ，或 V = W ，则记作 $V \overset{*}{\Rightarrow} W$

例子： 在例1中存在序列：

$V = 0S1 \Rightarrow 00S11 \Rightarrow 000S111 \Rightarrow 00001111 = W$

记作： $0S1 \overset{+}{\Rightarrow} 00001111$

$0S1 \overset{*}{\Rightarrow} 00001111$

一样的

4、句型的定义：

设G[S]是文法，如果符号串x是从识别符号（开始符号）推导出来的（即$S \overset{*}{\Rightarrow} x$）则称x是文法G[S]的**句型**。

若x仅由终结符号组成（$S \overset{*}{\Rightarrow} x$，$x \in V_T^*$）

则称x为G[S]的**句子**。

例： S，0S1，000111都是文法G的句型，000111是G的句子。

〖结论〗句子一定是句型，句型不一定是句子。

5、语言的定义： L(G)表示

文法G产生的语言定义为： G产生的句子的集合

$\{x \mid S \overset{*}{\Rightarrow} x,\ S$为文法开始符号，$\ x \in V_T^* \}$

该集合为语言，用L(G)表示。

从定义可知： x是句型且x是文法G的句子。

想： 例1的语言表示为什么？

5、语言的定义： L(G)表示

文法G产生的语言定义为：

集合 $\{x \mid S \Rightarrow x,\ S$为文法开始符号，$\ x \in V_T^*\}$

该集合为语言，用L(G)表示。

从定义可知： x是句型且x是文法G的句子。

想： 例1的语言表示为什么？

答案：$L(G)=\{0^n1^n \mid n \geq 1\}$

因为$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow \ldots \Rightarrow 0^n1^n$

重复利用规则 $S \Rightarrow 0S1$

例5：设G=（V_N，V_T，P，S），V_N＝｛S,B,E｝

V_T＝｛a,b,e｝ P产生式为：

(1) S →aSBE　(2) S →aBE　(3) EB → BE

(4) aB →ab　(5) bB →bb　(6) bE →be　(7) eE →ee

想：L(G)表示为什么？

例5：设**G=（V_N，V_T，P，S），V_N＝｛S,B,E｝**

V_T＝｛a,b,e｝P产生式为：

(1) S →aSBE　**(2) S** →aBE　**(3) EB** → BE

(4) aB →ab　**(5) bB** →bb　**(6) bE** →be　**(7) eE** →ee

想：L(G)表示为什么？

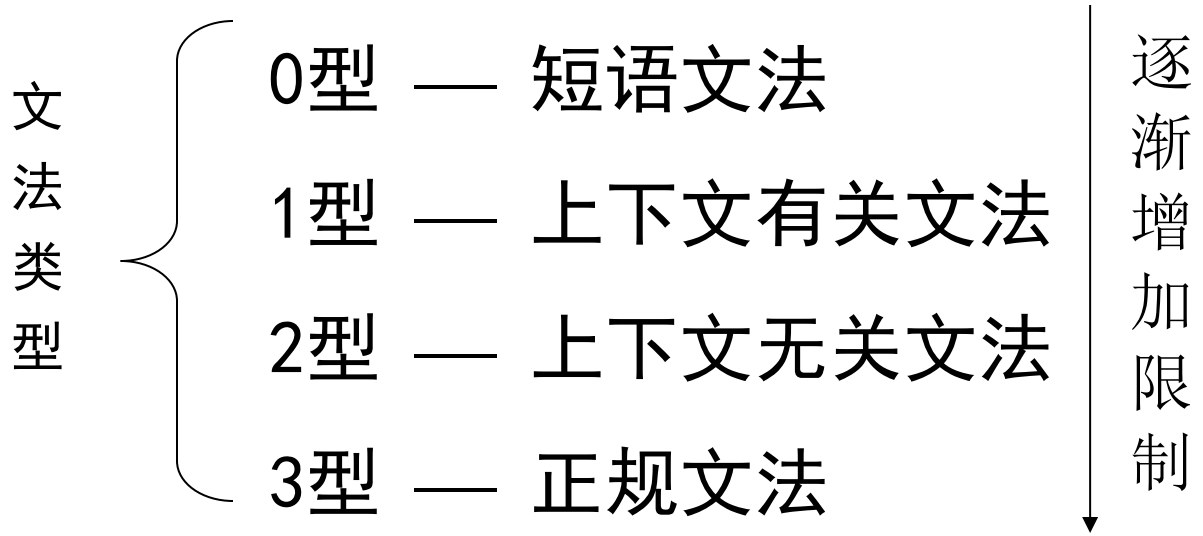
$L(G)=\{a^n b^n e^n \mid n\geq 1\}$

解：S ⇒aSBE ⇒aaSBEBE ⇒aaaSBEBEBE ⇒…⇒
aaaSBBEBEE⇒aaaSBBBEEE⇒aaaaBEBBBEEE⇒
aaaaBBEBBEEE⇒aaaaBBBEBEEE⇒aaaaBBBBEEEE
⇒aaaabBBBEEEE⇒aaaabbBBEEEE⇒aaaabbbBEEEE
⇒aaaabbbbEEEE⇒aaaabbbbeEEE⇒ aaaabbbbeeEE⇒
aaaabbbbeeeE ⇒ aaaabbbbeeeee ⇒ $a^4b^4e^4$

同理：$S\overset{*}{\Rightarrow}a^n b^n e^n$

四、文法的类型：

语言学家Chomsky把文法分成以下四种类型：



如果文法是正规文法⇒一定也是上下文无关文法

设$G=(V_N, V_T, P, S)$，如果它的每一个产生式$\alpha \rightarrow \beta$满足：

$\alpha \in (V_N \cup V_T)^*$且至少包含一个非终结符，

$\beta \in (V_N \cup V_T)^*$，则G是0型文法。

1型文法又称为上下文有关文法，它的每一个产生式也可描述为：$\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$，其中$\alpha_1 \alpha_2$ β都在$(V_N \cup V_T)^*$中，$\beta <> \varepsilon$，A在V_N中。只有A出现在$\alpha_1 \alpha_2$的上下文中，才允许用β取代A。

1、上下文无关文法：

设**G=（V_N，V_T，P，S），若P中的每一个产生式α→β满足：**

α是一非终结符，β∈$(V_N \cup V_T)^*$，则此文法称为上下文无关文法 (2型文法) 。（用β 取代α 与α 所在的上下文无关）

例6：G＝({S,A,B},{a,b},P,S)，P的产生式如下：

S→aB

S→bA

A→bAA

A→a

A→aS

B→b

B→bS

B→aBB

上下文无关文法

该文法可以写成：

S→aB | bA

A→ a | aS | bAA

B→ b | Bs | aBB

2、正规文法：

设**G=（V_N，V_T，P，S），若P中的每一个产生式都是A→aB**或A→a，A,B都是非终结符，a是终结符，则G是正规文法。

例6：G＝({S,A,B},{0,1},P,S)，P的产生式如下：

S→0A

S→1B

S→0

A→0S

A→0A

A→1B

B→1

B→0

B→1B

正规文法

该文法可以写成：

S→ 0 | 0A | 1B

A→ 0S | 0A | 1B

B→ 0 | 1 | 1B

- 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
- 描述一种简单赋值语句的产生式：

〈赋值语句〉→i：=E

- 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〉

- 但上下文无关文法不足以描述自然语言

- 举例说明：词义分析
  - 打→玩耍？
  - 打→ 12个？
  - 打 球类运动→玩耍 球类运动
  - 数词 打→数词 * 12个

给定一段程序，如何判断它是否符合程序设计语言的上下文无关文法？

如果能根据文法，为该程序构造出一棵语法树，则该程序符合当前文法。

五、上下文无关文法及其语法树

描述上下文无关文法的句型推导的直观方法。

1、语法树（推导树）的定义：

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，须满足条件为：

- 每个结点都有一个标记，此标记是V的一个符号。
- 根的标记是S。
- 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。
- 如果结点n的直接子孙，从左到右的次序是结点$n_1, n_2, \ldots, n_K$，其标记分别为$A_1, A_2, \ldots, A_K$，那么$A \rightarrow A_1A_2\ldots A_K$一定是P中的一个产生式。

例8：G=({S,A},{a,b},P,S)，其中P为：

$S \rightarrow aAS$

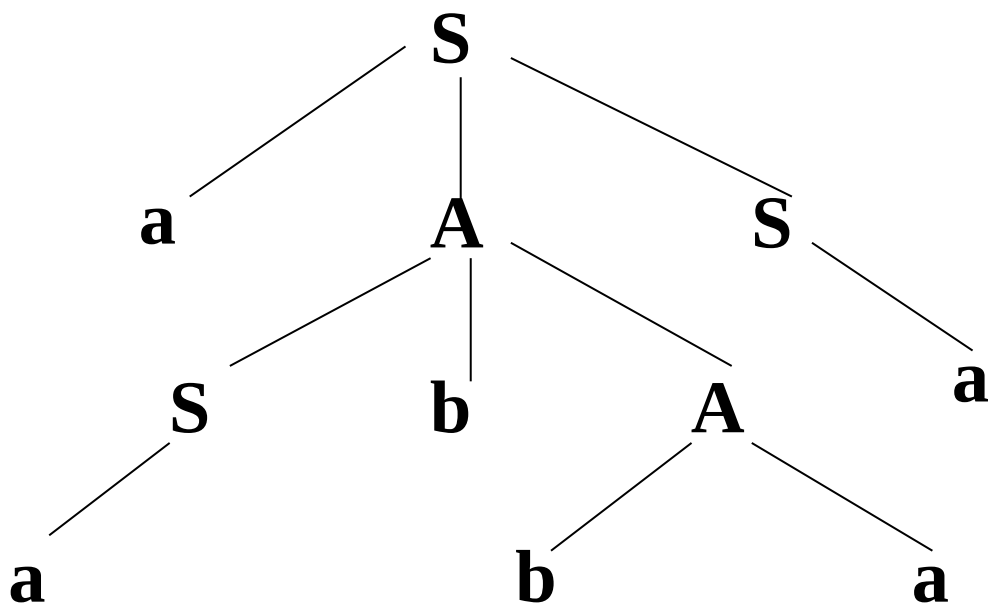
$A \rightarrow SbA$

$A \rightarrow SS$

$S \rightarrow a$

$A \rightarrow ba$

想：请构造aabbaa的语法树



说明：

语法树满足： ①树根是S。

② 每个节点的标记都是V的一个符号。

③ A有自己的子孙(S,b,A) ， $A \in V_N$中。

④(SbA) $A \rightarrow$ SbA是产生式，(aAS) $S \rightarrow$ aAS是产生式。

➢语法树的理解：

语法树表示了在推导过程中用到什么样的产生式和用到哪些非终结符，并没有表明采用的顺序。

只表示推导的结果，不表示推导的过程。

上式推导树是aabbaa句型的语法树。可以有多种推导过程如下：

➤语法树的理解：

语法树表示了在推导过程中用到什么样的产生式和用到哪些非终结符，并没有表明采用的顺序。

只表示推导的结果，不表示推导的过程。

上式推导树是aabbaa句型的语法树。可以有多种推导过程如下：

- （1）每次替换最右边非终结符
- S⇒aAS ⇒aAa ⇒aSbAa ⇒aSbbaa ⇒aabbaa
- （2）每次替换最左边非终结符
- S⇒aAS ⇒ aSbAS⇒aabAS⇒aabbaS⇒aabbaa
- （3）无固定的推导方向
- S⇒aAS⇒ aSbAS⇒aSbAa⇒aabAa⇒aabbaa

2、最左推导 （或最右推导）：

若在推导中的任何一步 $\alpha \Rightarrow \beta$（ α、β是句型），都是对α中的最左（最右）非终结符进行替换，则称为最左（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。

总结

- 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
- 给定句子S，如果能够根据文法G构造出该句子的语法树，则S为合乎文法G的句子
- 生成语法树可以有多种推导过程：最左推导、最右推导、无固定方向的推导

- 一棵语法树表示了一个句型的种种可能的不同推导过程，包括最左(最右)推导。但是，一个句型是否只对应唯一的一棵语法树呢?一个句型是否只有唯一的一个最左(最右)推导呢?

3、文法的二义性的定义：

- ✓ 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。
- ✓ 若一个文法中存在某个句子，它有两个不同的最左（最右）推导，则这个文法是二义的。

例10：文法G＝({ E}, {+，×，i，(，) }，P，E)，其中P为：

$E \rightarrow i$

$E \rightarrow E+E$

$E \rightarrow E \times E$

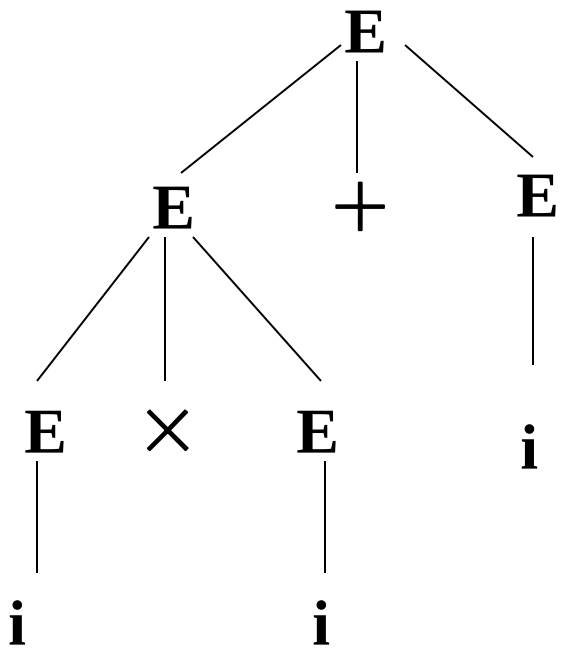
$E \rightarrow (E)$

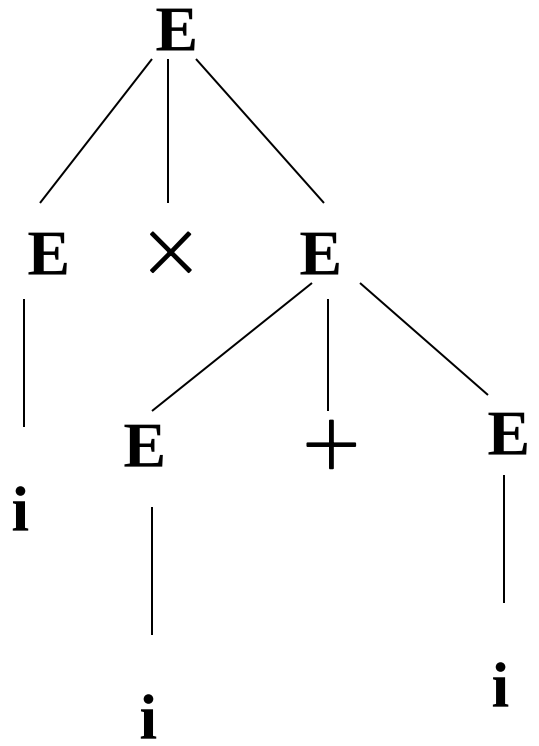
则句型i×i+i的推导：

推导1：E⇒E+E ⇒E×E+E ⇒i×E+E ⇒i×i＋E ⇒i×i＋i

推导2：E ⇒E×E ⇒i×E ⇒i×E＋E ⇒i×i＋E ⇒i×i＋i

产生两个不同的语法树：





该文法为二义性

如果规定“×”和“＋”的优先性，并服从左结合，上式就可以构出无二义性。

例如：文法G[E]:

$E \rightarrow T \mid E+T$

$T \rightarrow F \mid T \times F$

$F \rightarrow (E) \mid i$

$E \rightarrow i$

$E \rightarrow E+E$

$E \rightarrow E \times E$

$E \rightarrow (E)$

六、句型的分析

- 判断给定的符号串是否某文法的句子
- 例如：给定C语言的文法规则，对于一个源程序，要求编译系统判断该源程序是否合法。

六、句型的分析

❶ 句型分析：是识别一个符号串是否为某文法的句型，是整个推导的构造过程。（为一个符号串构造一个语法树/推导树）

即：识别输入符号串是否为语法上正确的程序的过程。

❷ 分析程序（识别程序）：是完成句型分析的程序 。

分析算法分为

- 自上而下分析法：从文法开始符号出发，反复使用规则，寻找匹配符号串（推导）
- 自下而上分析法：从输入符号开始，逐步进行“规约”，直至文法的开始符号（归约）

1、自上而下的分析法（用语法树分析）：

例： 文法G[S]：

$S \rightarrow cAd$

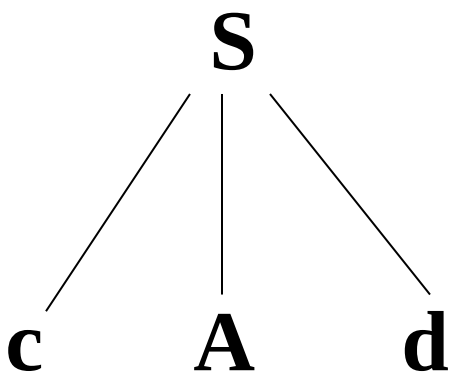
$A \rightarrow ab$

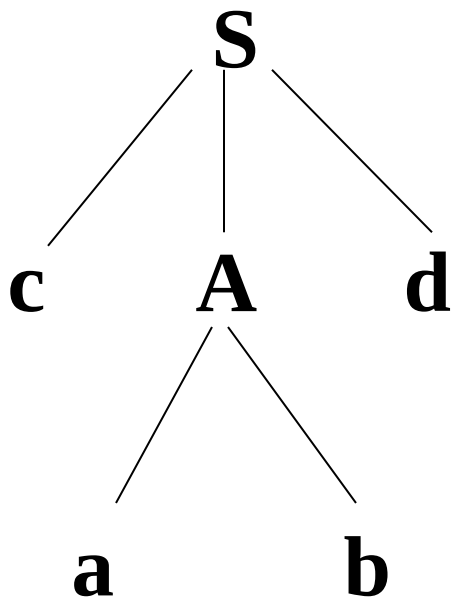
$A \rightarrow a$

求输入串W=cabd是否为该文法的句子。

语法树构造过程如下：

S

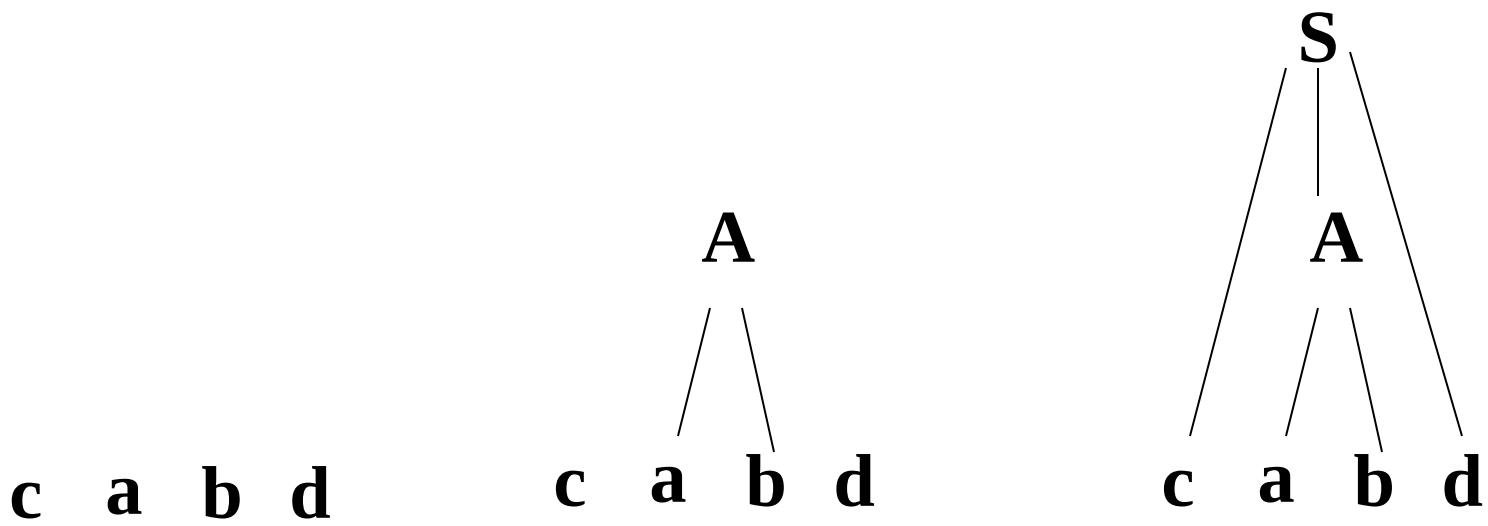




$S \Rightarrow cAd \Rightarrow cabd$

2、自下而上分析法：

从输入符号串cabd开始。


c a b d
A
c a b d
S
A
c a b d

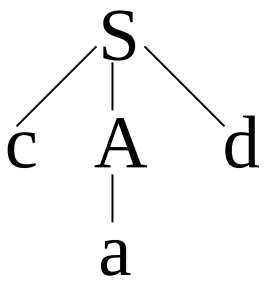
(1)S → cAd (2) A → ab (3)A → a

识别输入串w=cabd是否为该文法的句子

自上而下的语法分析

若$S \Rightarrow cAd$ 后选择(3)扩展A，$S \Rightarrow cAd \Rightarrow cad$

那将会？



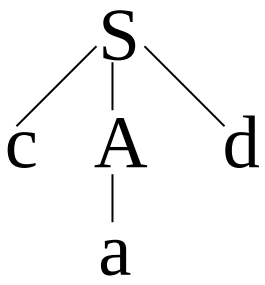
(1)S → cAd (2) A → ab (3)A → a

识别输入串w=cabd是否为该文法的句子

自上而下的语法分析

若S ⇒ cAd 后选择(3)扩展A，

S ⇒ cAd ⇒ cad



那将会？

w的第二个符号可以与叶子结点a得以匹配，但第三个符号却不能与下一叶子结点d匹配

？宣告分析失败（其意味着，识别程序不能为串w构造语法树，即w不是句子）

-显然是错误的结论。

导致失败的原因是在分析中对A的选择不是正确的。

(1)S → cAd (2) A → ab (3)A → a

识别输入串w=cabd是否为该文法的句子

自下而上的语法分析

对串cabd的分析中，如果不是选择ab用产生式(2),而是选择a用产生式(3)将a归约到了A，那么……?

c a b d

c A b d

a

(1)S → cAd (2) A → ab (3)A → a

识别输入串w=cabd是否为该文法的句子

自下而上的语法分析

对串cabd的分析中，如果不是选择ab用产生式(2),而是选择a用产生式(3)将a归约到了A，那么最终就达不到归约到S的结果，因而也无从知道cabd是一个句子

c a b d

c A b d

a

3、句型分析的2个问题 (讨论)：

◈ 在自上而下分析中，被代换的最左部非终结符号在右部有多个终结符时，如何确定？

假定要被代换的最左非终结符号是B，且有n条规则：B→A1|A2|…|An，那么如何确定用哪个右部去替代B？

方法：采取回溯法（在第5章介绍）

◈ 自下而上的分析中，如何确定可归约串呢？

方法：寻找句柄。

4、句柄的定义：（句柄是可归约串的称呼）

令G是一文法，S是文法的开始符号，αβδ是文法G的一个句型。（为αβδ 确定可归约串）如果有$S \overset{*}{\Rightarrow} \alpha A \delta$ 且 $A \overset{+}{\Rightarrow} \beta$，则称β是句型αβδ相对于非终结符A的短语。

若有$A \Rightarrow \beta$，则称β是句型 αβδ 相对$A \rightarrow \beta$的直接短语。一个句型的最左直接短语称为该句型的句柄。

句柄是自底向上句法分析中当前时刻需要规约的符号串。如果能够自动计算出当前的句柄，则可执行自动句法分析。

例：文法G[E]：

$$E \rightarrow T \mid E+T$$

$$T \rightarrow F \mid T \times F$$

$$F \rightarrow (E) \mid i$$

求句型i×i＋i的短语，直接短语和句柄。

记$i\times i+i$ 为$i_1\times i_2+i_3$

推导如下：

① $E\overset{*}{\Rightarrow}F\times i_2+i_3$ 且$F \Rightarrow i_1$，则称i_1是句型$i_1\times i_2+i_3$的相对非终结符F的短语，是相对规则$F\rightarrow i$的直接短语。

②$E\overset{*}{\Rightarrow}i_1\times F+i_3$ 且$F\Rightarrow i_2$，则称i_2是句型$i_1\times i_2+i_3$的相对非终结符F的短语，是相对规则$F\rightarrow i$的直接短语。

③$E \overset{*}{\Rightarrow} i_1 \times i_2 + F$且$F \Rightarrow i_3$，则称$i_3$是句型$i_1 \times i_2 + i_3$的相对非终结符F的短语，是相对规则$F \rightarrow i$的直接短语。

④$E \overset{*}{\Rightarrow} T \times i_2 + i_3$ 且$T \overset{+}{\Rightarrow} i_1$，则$i_1$是句型$i_1 \times i_2 + i_3$的相对于T的短语。

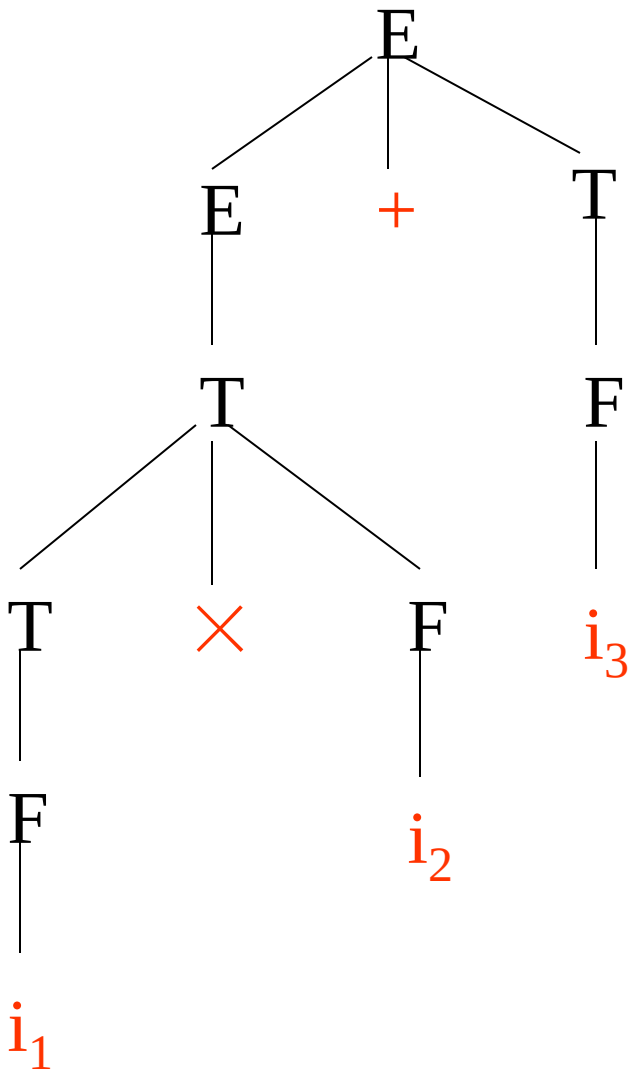
⑤$E \overset{*}{\Rightarrow} i_1 \times i_2 + T$ 且$T \overset{+}{\Rightarrow} i_3$，则$i_3$是句型$i_1 \times i_2 + i_3$的相对于T的短语。

⑥$E \overset{*}{\Rightarrow} T+i_3$ 且$T \overset{+}{\Rightarrow} i_1 \times i_2$，则$i_1 \times i_2$是句型$i_1 \times i_2+i_3$相对于T的短语。

⑦$E \overset{*}{\Rightarrow} E+i_3$ 且 $E \overset{+}{\Rightarrow} i_1 \times i_2$，则$i_1 \times i_2$是句型$i_1 \times i_2+i_3$相对于E的短语。

⑧$E \overset{*}{\Rightarrow} E$ 且 $E \overset{+}{\Rightarrow} i_1 \times i_2 + i_3$，则$i_1 \times i_2 + i_3$是句型$i_1 \times i_2 + i_3$相对于E的短语。

i_1，i_2，i_3，$i_1 \times i_2$和$i_1 \times i_2 + i_3$都是句型$i_1 \times i_2 + i_3$的短语，且i_1，i_2，i_3均为直接短语，其中i_1是最左直接短语（句柄）



从语法树中可以看出，所以树叶的组合就是其相对应的父亲的短语。

文法实用中的一些说明
----化简文法

文法中不含有*有害规则*和*多余规则*

有害规则：形如U→U的产生式。会引起文法的二义性

多余规则：指文法中任何句子的推导都不会用到的规则

多余规则也可表示为：文法中不含有不可到达和不可终止的非终结符，分以下两种情况：

1）文法中某些非终结符不在任何规则的右部出现，该非终结符称为不可到达。

2）文法中某些非终结符，由它不能推出终结符号串，该非终结符称为不可终止。

对于文法G[S]，为了保证任一非终结符A在句子推导中出现，必须满足如下两个条件：

1. A必须在某句型中出现

即有$S \Rightarrow^* \alpha A \beta$，其中α，β属于$V^*$

可到达

2. 必须能够从A推出终结符号串t来

即$A \Rightarrow^* t$，其中$t \in V_T^*$

可终止

例： 文法G[S]:

$S \rightarrow Be$

$B \rightarrow Ce \mid Af$

$A \rightarrow Ae \mid e$

$C \rightarrow Cf$

$D \rightarrow f$

1）文法中某些非终结符不在任何规则的右部出现；

2）文法中某些非终结符，由它不能推出终结符号串。

请判断其中是否有多余规则

化简文法:

化简文法

- 例：G[S]：　1) S→Be

2) B→Ce　　D为不可到达

3) B→Af　　C为不可终止

4) A→Ae

5) A→e

6) C→Cf

7) D→f

产生式 2），6），7）为多余规则应去掉。

典型例题及解答

- 1.证明文法G=({E,O},{(,),+,*,v,d},P,E)是二义的
- E →EOE | (E) | v | d
- O →+ | *

典型例题及解答

- 1.证明文法G=({E,O},{(,),+,*,v,d},P,E)是二义的
- E →EOE | (E) | v | d
- O →+ | *
- 只要存在一个句型，其语法树不只一棵，则可证明文法的二义性
- v*v+v

- 2. 考虑以下两个语言，给出其文法，并证明它们都是上下文无关的。
- $L1=\{a^n b^{2n} c^m | n,m>=0\}$
- $L2=\{a^n b^m c^{2m} |\ n,m>=0\}$

- 2. 考虑以下的两个语言，给出其文法，并证明它们都是上下文无关的。
- $L1=\{a^n b^{2n} c^m | n,m>=0\}$
- $L2=\{a^n b^m c^{2m} | n,m>=0\}$

- L1:
- S →AB
- A → ε |aAbb
- B → ε |cB

- L2:
- S →AB
- A → ε |aA
- B → ε |bBcc

本章目的

为语言的语法描述寻求工具,以便：

- 对源程序给出精确无二义的语法描述。（严谨、简洁、易读）
- 根据语言文法的特点来进行语法分析
- 从描述语言的文法可以自动构造出可用的分析程序
- 制导语义翻译

[本章小结]

1. 概念较多,应重点理解文法、语言、推导、句型、句子、短语、句柄的定义等概念.

2.文法作为程序语言的语法的描述工具,它用规则只能陈述的是:语言的所有句子以什麽样的符号串能出现.请记住文法和语言的形式定义中的 “形式”的含义—只涉及语言的语法不涉及语言的语义.

3.本章内容是形式语言理论的一部分.形式语言理论是对符号串集合的表示法、结构及其特性的研究。是程序设计语言语法分析研究的基础。

课后作业

P47-48练习:

1、2、6、7