## 讲稿（教学内容、步骤）

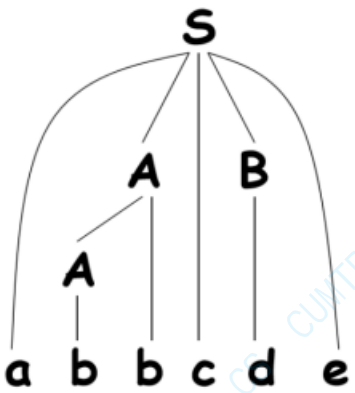
### 第 5 章 自底向上优先分析

1. 预备知识

- 自底向上分析方法，也称移进-归约分析法。
- 实现思想：
  - ✓ 对输入符号串自左向右进行扫描，并将输入符逐个移入一个栈中，边移入边分析，一旦栈顶符号串形成某个句型的句柄时，就用该产生式的左部非终结符代替相应右部的文法符号串，这称为归约。
  - ✓ 重复这一过程，直到栈中只剩文法的开始符号时，则分析成功，也就确认输入串是文法的句子。

【举例】

例1 文法G[S]：
(1) S → aAcBe
(2) A → b
(3) A → Ab
(4) B → d
分析 abbcde




对输入串 #abbcde# 的移进-归约分析过程：

| 步骤 | 符号栈 | 输入符号串 | 动作 |
|---|---|---|---|
| 1） | # | abbcde# | 移进 |
| 2） | #a | bbcde# | 移进 |
| 3） | #ab | bcde# | 归约(A→b) |
| 4） | #aA | bcde# | 移进 |
| 5） | #aAb | cde# | 归约(A→Ab) |
| 6） | #aA | cde# | 移进 |
| 7） | #aAc | de# | 移进 |
| 8） | #aAcd | e# | 归约(B→d) |
| 9） | #aAcB | e# | 移进 |
| 10） | #aAcBe | # | 归约(S→aAcBe) |
| 11） | #S | # | 接受 |

【讨论】什么是移进，什么是归约？

- ✓ 移进就是将一个终结符推进符号栈。
- ✓ 归约就是将 0 个或多个符号从栈中弹出，根据产生式将一个非终结符压入符号栈。

2. 自底向上分析法

(1) 自底向上分析的策略：移进-归约分析。
(2) 移进-归约过程是规范推导（最右推导）的逆过程，所以它是规范归约。
(3) 自底向上分析的关键：在分析过程中如何确定“句柄”。
(4) 方法：

① 简单优先分析法

- 先按照一定原则，求出文法所有符号（VT 和 VN）之间的**优先关系**；再按照这种关系确定归约过程中的句柄。
- 优点：规范归约，分析准确、规范
- 缺点：分析效率低，实用价值不大

② 算符优先分析法

- 先按照一定原则，求出文法所有 VT 之间的**优先关系**；归约时，只要遇到句柄就归约。
- 优点：分析速度快，特别适于表达式的分析
- 缺点：不是规范归约

(5) 优先关系

- X≐Y ：X与Y优先关系相等
- X⋖Y ：X的优先性比Y小
- X⋗Y ：X的优先性比Y大

**① 简单优先关系**

- X≐Y ：X与Y优先关系相等　　不等价于 Y≐X

  文法G中存在产生式A→...XY...
- X⋖Y ：X的优先性比Y小　　不等价于 Y⋗X

  文法G中存在产生式A→...XB...，且B⇒Y...
- X⋗Y ：X的优先性比Y大　　不等价于 Y⋖X

  文法G中存在产生式A→...BD...，且B⇒...X，D⇒Y...

**② 算符优先关系**

设 G[S]是一个不含 ε 产生式的算符文法 G 中

- a≐b：当且仅当文法中含有形如

  A→…ab…或A →…aBb…的产生式

【举例】

文法 G[E]：

(0) E’ →#E#
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→P^F
(6) F→P
(7) P→(E)
(8) P→i

存在 #E#，故 #≐#
存在 (E) ，故 (≐)

- a $\lessdot$ b ：当且仅当文法中含有形如

  $A \to \cdots aB \cdots$ 的产生式，且 $B \overset{+}{\Rightarrow} b\cdots$ 或 $B \overset{+}{\Rightarrow} Cb\cdots$

【举例】

文法 G[E]：

(0) E’ →#E#
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→P^F
(6) F→P
(7) P→(E)
(8) P→i

$E \overset{+}{\Rightarrow} (\cdots$ $E \overset{+}{\Rightarrow} i\cdots$ $E \overset{+}{\Rightarrow} E+\cdots$ $E \overset{+}{\Rightarrow} T*\cdots$ $E \overset{+}{\Rightarrow} P\hat{}\cdots$
#E，故 #$\lessdot$( #$\lessdot$i #$\lessdot$+ #$\lessdot$* #$\lessdot$^

- a $\gtrdot$ b：当且仅当文法中含有形如

  $A \to \cdots Bb \cdots$ 的产生式，且 $B \overset{+}{\Rightarrow} \cdots a$ 或 $B \overset{+}{\Rightarrow} \cdots aC$

【举例】

文法 G[E]：

(0) E’ →#E#
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→P^F
(6) F→P
(7) P→(E)
(8) P→i

$E \overset{+}{\Rightarrow} \cdots)$ $E \overset{+}{\Rightarrow} \cdots i$ $E \overset{+}{\Rightarrow} \cdots +T$ $E \overset{+}{\Rightarrow} \cdots *F$ $E \overset{+}{\Rightarrow} \cdots \hat{}F$
E#，故 )$\gtrdot$# i$\gtrdot$# +$\gtrdot$# *$\gtrdot$# ^$\gtrdot$#

3. 简单优先分析法

(1) 主要思想：

先按照一定原则，求出文法所有符号（VT 和 VN）之间的优先关系；再按照优先关系确定归约过程中的句柄。

(2) 实现步骤：

① 拓广文法 S’→#S#
② 构造优先关系表
③ 判断是否为简单优先文法
④ 根据优先关系表分析句子

(3) 构造优先关系表的方法：

**1）求各种优先关系**

①求 $\doteq$ 关系

在产生式右部找相邻的符号$V_1V_2$，则 $V_1 \doteq V_2$

②求 $\lessdot$ 关系

在产生式右部找$V_TV_N$形式，则 $V_T \lessdot a$，其中$V_N \overset{+}{\Rightarrow} a...$

在产生式右部找$V_{N1}V_{N2}$形式，则 $V_{N1} \lessdot a$，其中$V_{N2} \overset{+}{\Rightarrow} a...$

③求 $\gtrdot$ 关系

在产生式右部找$V_NV_T$形式，则 $b \gtrdot V_T$，其中$V_N \overset{+}{\Rightarrow} ...b$

在产生式右部找$V_{N1}V_{N2}$形式，则 $b \gtrdot V_{N2}$，其中$V_{N1} \overset{+}{\Rightarrow} ...b$

【举例】

例 2 拓广后的文法 G：

(0) S' → #S#
(1) S → bAb
(2) A → (B
(3) A → a
(4) B → Aa)

① 求 $\doteq$ 关系

# $\doteq$ #，b $\doteq$ A，A $\doteq$ b，( $\doteq$ B，A $\doteq$ a，a $\doteq$ )

② 求 $\lessdot$ 关系

# $\lessdot$ S，# $\lessdot$ b，b $\lessdot$ a，b $\lessdot$ (，( $\lessdot$ A，( $\lessdot$ a，( $\lessdot$ (

③ 求 $\gtrdot$ 关系

S $\gtrdot$ #，b $\gtrdot$ #，a $\gtrdot$ b，B $\gtrdot$ b，) $\gtrdot$ b，a $\gtrdot$ a，B $\gtrdot$ a，) $\gtrdot$ a

**2）根据优先关系，构造优先关系矩阵**

【举例】对例 2 文法构造优先关系矩阵

| | S | A | B | ( | ) | a | b | # |
|---|---|---|---|---|---|---|---|---|
| S | | | | | | | | $\gtrdot$ |
| A | | | | | | $\doteq$ | $\doteq$ | |
| B | | | | | | $\gtrdot$ | $\gtrdot$ | |
| ( | | $\lessdot$ | $\doteq$ | $\lessdot$ | | $\lessdot$ | | |
| ) | | | | | | $\gtrdot$ | $\gtrdot$ | |
| a | | | | | $\doteq$ | $\gtrdot$ | $\gtrdot$ | |
| b | | $\doteq$ | | $\lessdot$ | | $\lessdot$ | | $\gtrdot$ |
| # | $\lessdot$ | | | | | | $\lessdot$ | $\doteq$ |

(4) 判断是否为简单优先文法的方法：

简单优先文法的定义

满足以下所有条件的文法是简单优先文法

- 在文法符号集 V 中，任意两个符号之间最多只有一种优先关系。
- 在文法中，任意两个产生式没有相同的右部。
- 不含空产生式。

采用简单优先分析时，必须是简单优先文法。

【举例】判断例 2 是否为简单优先文法

∵ ① 由上述优先关系表可见，任意两个符号之间最多只有一种优先关系

② 由文法可见，任意两个产生式没有相同的右部

③ 由文法可见，不含空产生式

∴ 例 2 是简单优先文法

(5) 根据优先关系表分析句子

构造相应优先关系矩阵，并将文法的产生式保存，设置符号栈S，算法步骤如下：

1. 将输入符号串$a_1a_2a_3\ldots a_n$# 依次逐个存入符号栈S中，直到遇到栈顶符号$a_i$ $\gtrdot$ 下一个待输入符号$a_j$ 时为止。
2. 栈顶当前符号$a_i$为句柄尾，由此向左在栈中找句柄的头符号$a_k$，即找到$a_{k-1}$ $\lessdot$ $a_k$为止。
3. 找到句柄$a_k\cdots a_i$，在文法的产生式中查找右部为$a_k\ldots a_i$的产生式，若找到则用相应左部代替句柄，若找不到则为出错，这时可断定输入串不是该文法的句子。
4. 重复上述三步，直到归约完所有输入符号串为止。（此时栈中只剩文法的开始符号）

【举例】

例2 文法G[S]：
(1) S → bAb
(2) A → (B
(3) A → a
(3) B → Aa)
分析输入串 #b(aa)b#

| | S | A | B | ( | ) | a | b | # |
|---|---|---|---|---|---|---|---|---|
| S | | | | | | | | = |
| A | | | | | | = | = | |
| B | | | | | | > | > | |
| ( | | < | = | < | | < | | |
| ) | | | | | | > | > | |
| a | | | | | = | > | > | |
| b | | = | | < | | < | | = |
| # | < | | | < | | | = | |

| 步骤 | 符号栈S | 待输入符号串 | 动作 |
|---|---|---|---|
| 1） | # | b(aa)b# | #<b,移进 |
| 2） | #b | (aa)b# | b<(,移进 |
| 3） | #b( | aa)b# | (<a,移进 |
| 4） | #b(a | a)b# | a>a,归约A→a |
| 5） | #b(A | a)b# | A=a,移进 |
| 6） | #b(Aa | )b# | a=),移进 |
| 7） | #b(Aa) | b# | )>b,归约B→Aa) |
| 8） | #b(B | b# | B>b,归约A→(B |
| 9） | #bA | b# | A=b,移进 |
| 10） | #bAb | # | b>#,归约S→bAb |
| 11） | #S | # | 接受 |

4. 算符优先分析法（OPG，Operator Precedence Grammar）

(1) 算符文法的定义

- **算符文法**：上下文无关文法G中**没有**形如

  **A→…BC…的产生式，其中B,C∈$V_N$，则称G为算符文法（OG，Operator Grammer）。**

  - **性质1：在算符文法中任何句型都不包含两个相邻的非终结符。**
  - **性质2：如 Ab 或 bA 出现在算符文法的句型 γ 中，其中 A∈$V_N$, b∈$V_T$, 则 γ 中任何含 b 的短语必含有 A 。（但含A的短语未必含b。）**

(1) 直观法确定算符优先关系

【举例】

例3 文法G[E]：

E→E+E | E-E

E→E*E | E/E

E→E↑E

E→(E)

E→i

算符优先关系表：

| | + | - | * | / | ↑ | ( | ) | i | # |
|---|---|---|---|---|---|---|---|---|---|
| + | > | > | < | < | < | < | > | < | > |
| - | > | > | < | < | < | < | > | < | > |
| * | > | > | > | > | < | < | > | < | > |
| / | > | > | > | > | < | < | > | < | > |
| ↑ | > | > | > | > | < | < | > | < | > |
| ( | < | < | < | < | < | < | = | < | |
| ) | > | > | > | > | > | | > | | > |
| i | > | > | > | > | > | | > | | > |
| # | < | < | < | < | < | < | | < | = |

- i的优先级最高
- ↑优先级次于i，右结合
- *和/优先级次之，左结合
- +和-优先级最低，左结合
- 括号的优先级大于括号外的运算符，小于括号内的运算符
- 内括号的优先性大于外括号
- #的优先性低于与其相邻的算符

复杂文法的优先关系表构造困难。

(2) 算符优先分析法的主要思想

对文法按照一定规则，求出终结符之间的优先关系；再按照这种优先关系来确定句柄。

(3) 算符优先分析法的实现步骤

① 拓广文法 S'→#S#

② 构造算符优先关系表

③ 判断是否为算符优先文法（OPG 文法）

④ 根据优先关系表分析句子

(4) 拓广文法

【举例】

例 4 文法 G[E]

(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→P^F
(6) F→P
(7) P→(E)
(8) P→i

拓广文法 G[E']:

(0) E'→#E#
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→P^F
(6) F→P
(7) P→(E)
(8) P→i

(5) 构造算符优先关系表（定义法【讲授】、关系图法【自学】）

定义法构造算符优先关系表的步骤：

### （1）定义 FirstVT 和 LastVT

$\text{FirstV}_T(B)=\{b \mid B \overset{+}{\Rightarrow} b\ldots \text{ 或 } B \overset{*}{\Rightarrow} Cb\ldots\}$

$\text{LastV}_T(B)=\{a \mid B \overset{+}{\Rightarrow} \ldots a \text{ 或 } B \overset{*}{\Rightarrow} \ldots aC\}$

### （2）求优先关系

| | |
|---|---|
| A→…ab…或A →…aBb…时， | 则 a = b |
| A→…aB… | 时，则 a < FirstVT(B) |
| A→…Bb… | 时，则 LastVT(B) > b |

### （3）构造优先关系表

【举例】

例 4 文法 G[E]

(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→P^F
(6) F→P
(7) P→(E)
(8) P→i

1）计算每个非终结符的 $\text{FirstV}_T$ 集和 $\text{LastV}_T$ 集

$\text{FirstV}_T(E')=\{\#\}$
$\text{FirstV}_T(E)=\{+, *, ^, (, i\}$
$\text{FirstV}_T(T)=\{*, ^, (, i\}$
$\text{FirstV}_T(F)=\{^, (, i\}$
$\text{FirstV}_T(P)=\{(, i\}$

$\text{LastV}_T(E')=\{\#\}$
$\text{LastV}_T(E)=\{+, *, ^, i, )\}$
$\text{LastV}_T(T)=\{*, ^, i, )\}$
$\text{LastV}_T(F)=\{^, i, )\}$
$\text{LastV}_T(P)=\{i, )\}$

2）求优先关系

- 求=关系：右部寻找…aBc…或…ac…，a=c
- 求<关系：右部寻找…aB…，a<FirstVT(B)
- 求>关系：右部寻找…Bc…，LastVT(B)>c

求=关系：# = #　( = )
求<关系[逐条扫描产生式，右部寻找A→…aB…的形式]
由于 #E 故 # < FirstVT (E)　#<{+，*，^，(，i}
由于 +T 故 + < FirstVT (T)　+<{*，^，(，i}
由于 *F 故 * < FirstVT (F)　*<{^，(，i}
由于 ^F 故 ^< FirstVT (F)　^<{^，(，i}
由于 (E 故 ( < FirstVT (E)　(<{+，*，^，(，i}
求>关系[逐条扫描产生式，右部寻找A→…Bb…的形式]
由于 E# 故 LastVT (E) > #　{+，*，^，i，)}>#
由于 E+ 故 LastVT (E) > +　{+，*，^，i，)}>+
由于 T* 故 LastVT (T) > *　{*，^，i，)}>*
由于 P^ 故 LastVT (P) > ^　{i，)}>^
由于 E) 故 LastVT (E) > )　{+，*，^，i，)}>)

3）构造优先关系表

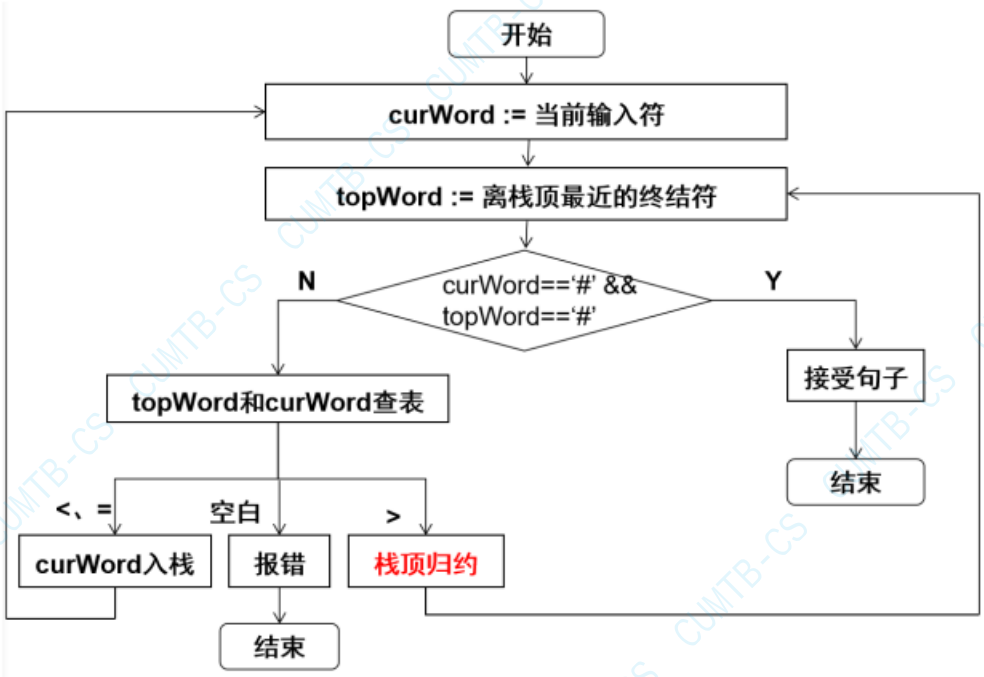
| | + | * | ^ | ( | ) | i | # |
|---|---|---|---|---|---|---|---|
| + | > | < | < | < | > | < | > |
| * | > | > | < | < | > | < | > |
| ^ | > | > | < | < | > | < | > |
| ( | < | < | < | < | = | < | |
| ) | > | > | > | | > | | > |
| i | > | > | > | | > | | > |
| # | < | < | < | < | | < | = |

(6) 判断是否为算符优先文法（OPG 文法）

算符优先文法的定义

- 设有一不含ε产生式的算符文法G，如果对任意两个终结符a和b之间至多只有 $=$ 、$<$ 、$>$ 三种关系的一种成立，则称 G是一个算符优先文法（OPG , Operator Precedence Grammar）。
  - ① 不含空产生式
  - ② 任何产生式右部不包含两个相邻的非终结符
  - ③ 任何两个终结符之间优先关系唯一
- 算符优先文法是无二义的。

(7) 根据优先关系分析句子




【举例】利用例 4 的算符优先分析表分析句子 #i+i#

| 步骤 | 分析栈 | 待输入串 | 动作 |
|---|---|---|---|
| 1 | # | i + i # | #<i 移进 |
| 2 | #i | + i # | i>+ 归约 i(F→i) |
| 3 | #F | + i # | #<+ 移进 |
| 4 | #F+ | i # | +<i 移进 |
| 5 | #F+i | # | i># 归约 i (F→i) |
| 6 | #F+F | # | +># 归约 F+F (E→E+T) |
| 7 | #F | # | 接受 |

(8) 【讨论】算符优先分析是否为规范归约

- 归约过程中，只考虑终结符之间的优先关系来确定句柄，而与非终结符无关。这样去掉了单个非终结符的归约，所以它**不是**规范归约。
- 为寻找算符优先分析过程中的可归约串，引进**最左素短语**的概念

(1) 定义

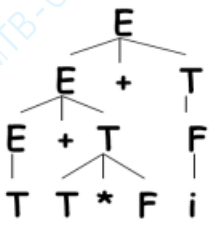
- 素短语：设有文法 G[S]，其句型的素短语是一个短语，它**至少包含一个终结符**，且除自身外**不再包含其他素短语**。

- 最左素短语：句型最左边的素短语。

【举例】例 4 文法 G[E]：

(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→P F
(6) F→P
(6) P→(E)
(7) P→i

求句型 #T+T*F+i# 的素短语。




短语：
T+T*F+i
T+T*F
T
T*F
i

素短语：
~~T+T*F+i~~
~~T+T*F~~
~~T~~
T*F
i

(2) 算符优先分析的局限性

- 简单优先分析：是规范归约，关键是寻找当前句型的句柄，符号栈顶一旦形成句柄就归约。

- 算符优先分析：不是规范归约，关键是寻找当前句型的最左素短语，符号栈顶一旦形成最左素短语就归约。
  - ✓ 算符优先分析可能出现“错误的句子得到正确的归约”；并且一般语言的文法很难满足算符优先文法的条件。
  - ✓ 结论：算符优先分析法只适用于表达式的语法分析。

5. 优先函数：为节约分析表的存储空间，提高查表效率，用**优先函数**代替**优先关系表**

- 优点：优先函数比优先矩阵节省空间
  - ✓ 优先矩阵占用内存空间：$(n+1)^2$
  - ✓ 优先函数占用内存空间：$2(n+1)$
- 缺点：当发生错误时不能准确指出出错位置
- 构造方法【自学】

| 迭代次数 | | + | * | ↑ | ( | ) | i | # |
|---|---|---|---|---|---|---|---|---|
| 0（初值） | f | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | g | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | f | 2 | 4 | 4 | 1 | 6 | 6 | 1 |
| | g | 2 | 3 | 5 | 5 | 1 | 5 | 1 |
| 2 | f | 3 | 5 | 5 | 1 | 7 | 7 | 1 |
| | g | 2 | 4 | 6 | 6 | 1 | 6 | 1 |
| 3 | f | 同第 2 次迭代结果 | | | | | | |
| | g | | | | | | | |

- 利用优先函数分析句子【自学】

【课堂练习】4 (1)　(2)　(3) 分析 a;(a+a)#　(4)　(5)

4. 文法　S→S;G

S→G

G→G(T)

G→H

H→a

H→(S)

T→T+S

T→S

(1) 构造算符优先关系表，并判别该文法是否为算符优先文法。

(2) 给出 a(T+S);H;(S)的短语、句柄、素短语和最左素短语。

(3) 给出 a;(a+a)的算符优先分析过程。

(4) 给出 a;(a+a)的最右推导。

(5) 说明算符优先分析的哪些缺点。

**(1) 拓广文法**

**S'→#S#**

**S→S;G**

**S→G**

**G→G(T)**

**G→H**

**H→a**

**H→(S)**

**T→T+S**

**T→S**

**计算FirstVT和LastVT**

FirstVT(S) = { ; ( a }

FirstVT(G) = { ( a }

FirstVT(H) = { ( a }

FirstVT(T) = { + ; ( a }

LastVT(S) = { ; ) a }

LastVT(G) = { ) a }

LastVT(H) = { ) a }

LastVT(T) = { + ; ) a }

(1) 算符优先关系表，及判别

| | ; | ( | ) | a | + | # |
|---|---|---|---|---|---|---|
| ; | > | < | > | < | > | > |
| ( | < | < | = | < | < | |
| ) | > | > | > | | > | > |
| a | > | > | > | | > | > |
| + | < | < | > | < | > | |
| # | < | < | | < | | = |

**判别：**

**该文法**

**(1) 不含空产生式；**

**(2) 任何产生式右部不包含两个相邻的非终结符；**

**(3) 任何两个终结符之间优先关系唯一。**

**故该文法是算符优先文法。**

(2) 给出a(T+S);H;(S)的短语、句柄、素短语和最左素短语。

短语：
- a(T+S);H;(S)
- a(T+S);H
- (S)
- a(T+S)
- H
- a
- T+S

直接短语：
- a
- T+S
- H
- (S)

句柄：a

素短语：
- (S)
- a
- T+S

最左素短语：a

(3) 给出a;(a+a)的算符优先分析过程。

| 步骤 | 栈 | 待输入串 | 动作 |
|---|---|---|---|
| 1 | # | a;(a+a)# | < 移进 |
| 2 | #a | ;(a+a)# | > 归约 |
| 3 | #H | ;(a+a)# | < 移进 |
| 4 | #H; | (a+a)# | < 移进 |
| 5 | #H;( | a+a)# | < 移进 |
| 6 | #H;(a | +a)# | > 归约 |
| 7 | #H;(H | +a)# | < 移进 |
| 8 | #H;(H+ | a)# | < 移进 |
| 9 | #H;(H+a | )# | > 归约 |
| 10 | #H;(H+H | )# | > 归约 |
| 11 | #H;(T | )# | = 移进 |
| 12 | #H;(T) | # | > 归约 |
| 13 | #H;H | # | > 归约 |
| 14 | #S | # | = 接受 |

**(4)** 给出a;(a+a)的最右推导。

S => S;G
=> S;G(T)
=> S;G(T+S)
=> S;G(T+G)
=> S;G(T+H)
=> S;G(T+a)
=> S;G(S+a)
=> S;G(G+a)
=> S;G(H+a)
=> S;G(a+a)
=> S;H(a+a)
=> 无法继续

**(5)** 说明算符优先分析的哪些缺点。

算符优先分析可能将错误的句子得到正确的归约。

【课后习题】1 (1) (2) (4) 分析(a,a)#

1. 文法 S⟶a | ^ | (T)
   T⟶T,S | S

(1) 计算 FirstVT 和 LastVT。

| | |
|---|---|
| S'→#S# | FirstVT(S')={ # } |
| S→a | FirstVT(S) = { a ^ ( } |
| S→^ | FirstVT(T) = { a ^ ( , } |
| S→(T) | LastVT(S')={ # } |
| T→T,S | LastVT(S) = { a ^ ) } |
| T→S | LastVT(T) = { a ^ ) , } |

(2) 构造算符优先关系表，并说明该文法是否为算符优先文法。

| | | |
|---|---|---|
| S'→#S# | #=# | (=) |
| S→a | #<FirstVT(S) | # < { a ^ ( } |
| S→^ | (<FirstVT(T) | ( < { a ^ ( , } |
| S→(T) | ,<FirstVT(S) | , < { a ^ ( } |
| T→T,S | LastVT(S)>#  | {a ^ ) } > # |
| T→S | LastVT(T)>) | {a ^ ) , } > ) |
| | LastVT(T)>, | {a ^ ) , } > , |

| | a | ^ | ( | ) | , | # |
|---|---|---|---|---|---|---|
| a | | | | > | > | > |
| ^ | | | | > | > | > |
| ( | < | < | < | = | < | |
| ) | | | | > | > | > |
| , | < | < | < | > | > | |
| # | < | < | < | | | = |

判别：该文法(1)不含空产生式；

(2)任何产生式右部不包含两个相邻的非终结符；

(3)任何两个终结符之间优先关系唯一。

故该文法是算符优先文法。

(4) 给出输入串(a,a)#的算符优先分析过程。

| | 栈 | 待输入串 | 动作 | |
|---|---|---|---|---|
| 1 | # | (a,a)# | #<( | 移进 |
| 2 | #( | a,a)# | (<a | 移进 |
| 3 | #(a | ,a)# | a>, | 归约 S→a |
| 4 | #(S | ,a)# | (<, | 移进 |
| 5 | #(S, | a)# | ,<a | 移进 |
| 6 | #(S,a | )# | a>) | 归约 S→a |
| 7 | #(S,S | )# | ,>) | 归约 T→T,S |
| 8 | #(T | )# | (=) | 移进 |
| 9 | #(T) | # | )># | 归约 S→T |
| 10 | #S | # | | 接受 |

【本章小结】

关键问题：

- 在规范归约的过程中，关键问题是 如何确定“句柄”。
- 在算符优先分析归约中，关键问题是 如何确定“最左素短语”。

(1) 简单优先分析 （规范归约）

① 优点：准确、规范

缺点：分析效率低，实际使用价值低

② 基本思想：按照一定原则，求出文法所有符号之间的优先关系，按照这种关系，确定归约过程中的句柄，之后进行归约。

③ 优先关系的表示：a<b　　a=b　　a>b

④ 优先关系矩阵的构造：

⑤ 对输入串的分析过程：< =移进　>归约

(2) 算符优先分析

① 优点：分析速度快，特别适用于表达式的分析

缺点：不规范，可能“错误的句子得到正确的归约”

② 基本思想：按照一定原则，求出文法所有终结符之间的优先关系，按照这种关系，确定归约过程中的最左素短语，之后进行归约。

③ 优先关系的表示：a<b　　a=b　　a>b

④ 优先关系矩阵的构造：拓广文法 S'→#S#；求 FirstVT LastVT，寻找优先关系，构造优先关系表；分析句子。

⑤ 对输入串的分析过程：< = 移进　　> 归约

(3) 优先函数

① 为什么要引入优先函数?

为节约存储。

- 优先矩阵占用内存空间：$(n+1)^2$　　// n:终结符个数；　1:“#”
- 优先函数占用内存空间：2(n+1)

② 优先函数的定义：函数的值用整数表示，数值大小表示了优先关系的大小。

③ 构造方法（自学）

④ 利用优先函数分析句子（自学）

⑤ 缺点：出错时，不能准确指出出错的位置。