第5章 自底向上优先分析

自底向上优先分析

自底向上分析方法，也称移进-归约分析法

实现思想（是推导的逆过程）：

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后进先出栈中，边移入边分析，一旦栈顶符号串形成某个句型的句柄或可归约串时（该句柄或可归约串对应某产生式的右部），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，称为一步归约。重复这一过程，直到归约到栈中只剩下文法的开始符号时，则分析成功。

- 自底向上的移进-归约过程是自顶向下的最右推导的逆过程，而最右推导称为规范推导，自左向右的归约过程也称规范归约。

例1：文法

$$\begin{cases} S \rightarrow aAcBe \\ A \rightarrow b \\ A \rightarrow Ab \\ B \rightarrow d \end{cases}$$

输入串abbcde#分析

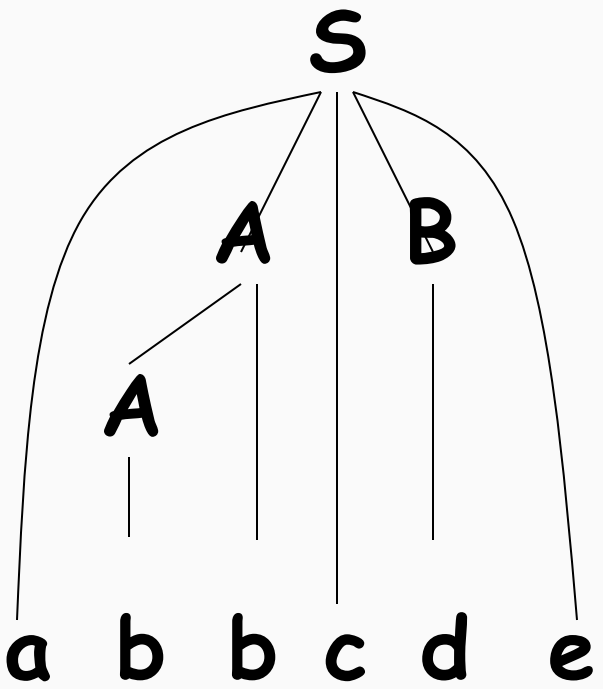
文法G[S]:

(1) S → aAcBe

(2) A → b

(3) A → Ab

(4) B → d



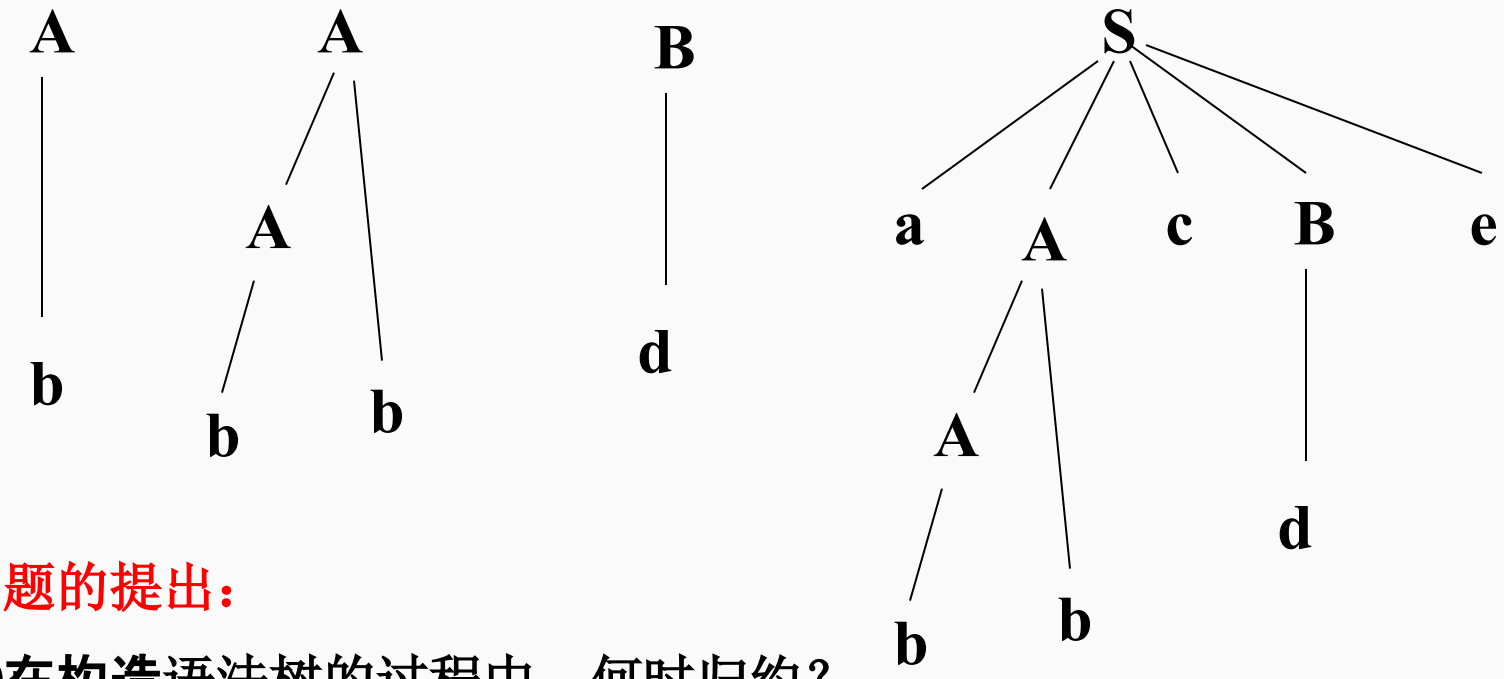
步骤	符号栈	输入符号串	动作
1）	#	abbcde#	移进
2）	#a	bbcde#	移进
3）	#ab	bcde#	归约(A→b)
4）	#aA	bcde#	移进
5）	#aAb	cde#	归约(A→Ab)
6）	#aA	cde#	移进
7）	#aAc	de#	移进
8）	# aAcd	e#	归约(B→d)
9）	#aAcB	e#	移进
10）	#aAcBe	#	归约
(11)	#S	#	接受

对输入串abbcde#的移进-归约分析过程

分析符号串abbcde是否为G[S]的句子？

$S \Rightarrow aAcBe \Rightarrow aAcde \Rightarrow aAbcde \Rightarrow abbcde$

上述的每一步归约可以构造一颗语法树：



问题的提出：

①**在构造**语法树的过程中，何时归约？

当可归约串或句柄出现在栈顶符号串中就可以归约。

②**如何知道在**栈顶符号串中已经形成可归约串或句柄？

通过自底向上分析算法中的优先关系来计算

自下而上分析的关键问题：
如何确定可归约串？

- 简单优先分析法：寻找句柄
- 算符优先分析法：寻找最左素短语

5.1 自底向上优先分析法概述

- 优先分析法又可分简单优先分析法和算符优先分析法
  - 简单优先分析法：按一定原则求出文法中所有符号（终结符和非终结符）的优先关系，按这种关系求出句柄。（规范归约——从左向右的归约）
  - 算符优先分析法：只考虑算符（终结符）之间的优先关系，不考虑非终结符之间的优先关系。按这种关系求出最左素短语。（不规范归约）
- 两种分析法比较
  - 简单优先分析法：准确规范，但分析效率低，实际使用价值不大；
  - 算符优先分析法：不规范，但分析速度快，适于实际使用。

句柄的定义：

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 $A \overset{+}{\Rightarrow} \beta$，**口称β是句型αβδ相对于非终结符A的短语。**

若有$A \Rightarrow \beta$，口称β是句型 αβδ 相对$A \rightarrow \beta$的直接短口。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归约的符号串。如果能够自动计算出当前的句柄，则可执行自动句法分析。

5.2 简单优先分析法

1、优先关系的表示

x $\doteq$ y 表示X与Y的优先关系相等

x $\gtrdot$ y 表示X的优先性大于Y

x $\lessdot$ y 表示X的优先性小于Y

对任意两个文法符号X、Y按其在句型中可能会出现的相邻关系来确定优先关系：

确定优先关系的规则

（1）X $\doteq$ Y 当且仅当G中存在产生式A →…XY…（在语法树的同一层）

（2） X $\lessdot$ Y 当且仅当G中存在产生式A →…XB…，且B $\overset{+}{\Rightarrow}$Y…（ Y 在 X 的下一层）

（3）X $\gtrdot$ Y 当且仅当G中存在产生式A →…BD…，且B $\overset{+}{\Rightarrow}$…X和D $\overset{*}{\Rightarrow}$Y…（ X在 Y 的下一层或X比 Y先归约——规范归约/最左归约 ）

例：有文法G[S]： S→bAb

A→(B | a

B→Aa)

解：文法符号优先关系推导如下：

(1) 求$\doteq$关系:

由S→bAb ， A→(B， B→Aa)

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

G[S]: S→bAb
A→(B | a
B→Aa)

(2) 求$\lessdot$关系：

由S→bAb 且A $\overset{+}{\Rightarrow}$ (B 可得 b $\lessdot$ (

A $\overset{+}{\Rightarrow}$ a 可得 b $\lessdot$ a

由A→(B 且B $\overset{+}{\Rightarrow}$ (B… 可得 ($\lessdot$ (

B $\overset{+}{\Rightarrow}$ aa… 可得 ($\lessdot$ a

B $\overset{+}{\Rightarrow}$ Aa) 可得 ($\lessdot$ A

G[S]: S→bAb
A→(B | a
B→Aa)

(3) 求$\dot{>}$ 关系:

由S→bAb，且A$\overset{+}{\Rightarrow}$…)　可得) $\dot{>}$ b

A$\overset{+}{\Rightarrow}$…B　可得 B $\dot{>}$ b

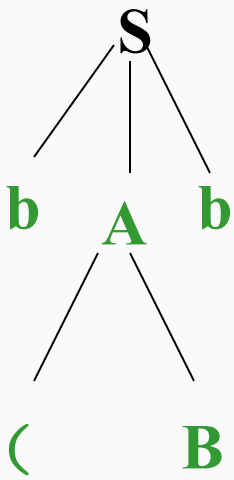
A$\overset{+}{\Rightarrow}$ a　可得 a $\dot{>}$ b

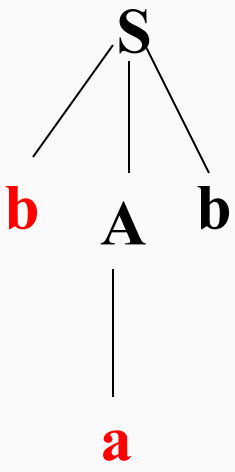
由B→Aa)，且A$\overset{+}{\Rightarrow}$…)　可得) $\dot{>}$ a

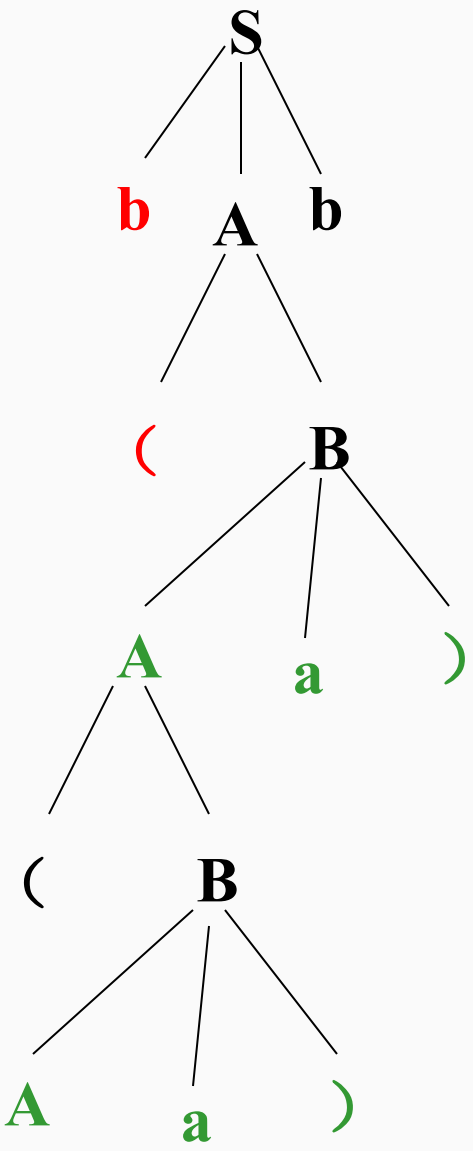
A$\overset{+}{\Rightarrow}$a　可得 a $\dot{>}$ a

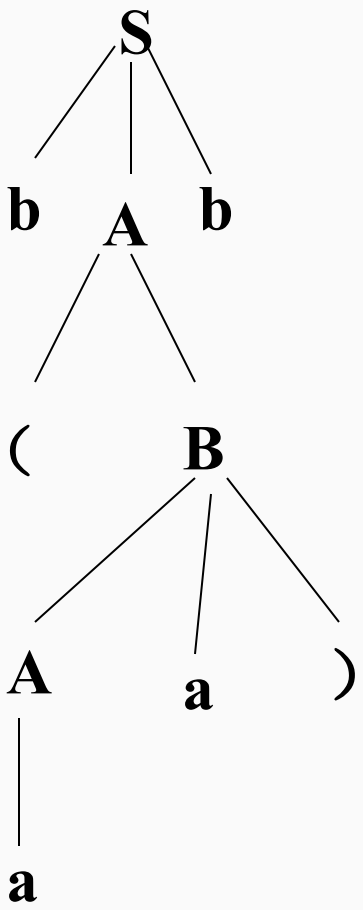
A$\overset{+}{\Rightarrow}$…B　可得 B $\dot{>}$ a

语法树结构如下：









G[S]: S→bAb
A→(B | a
B→Aa)

从上图可以看出：

(1) 当b(、ba出现在某一句型中，则（和a在句柄中，b不在句柄中，则（和a必须先归约。因此b $\lessdot$ (,b $\lessdot$ a。

(2) 同样可以看出，当((，(a ，(A出现在某一句型中，右边的(，a，A出现在句柄中，左边的(不在句柄中。因此，左边($\lessdot$ 右边的(，左边($\lessdot$ a，左边($\lessdot$ b。

(3) 当ab、aa出现在某一句型中，左边a在句柄中，则右边的a、b不可能在句柄中。因此有左边a $\gtrdot$ b，左边a $\gtrdot$ a。

- 因此，可以根据文法符号之间的优先关系确定句柄。
- 在规范归约（最左归约）过程中，出现在栈顶的优先级相同的连续符号串就是句柄。

优先关系矩阵表示法：

所有符号>#

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

如何确定优先关系？

文法G[S]:
(1) S → bAb
(2) A → (B|a
(3) B → Aa)

1.求=关系：
由(1)：b=A　A=b
由(2)：(=B
由(3)：A=a　a=)

2.求<关系：
由(1)(2)：b<(　b<a
由(2)(3)：(<A　(<(　(<a

3.求>关系：
由(1)：B>b　a>b　)>b
由(3)：B>a　a>a　)>a

4.#<所有符号，所有符号>#

简单优先文法的定义：

（1）在文法符号集中，任意两个符号之间最多只有一种优先关系；

（2）在文法中任意两个产生式没有相同的右部。

简单优先分析法的操作步骤

首先根据已知优先文法构造优先关系矩阵，设置符号栈S，算法步骤如下：

(1) 将输入符号$a_1a_2\ldots a_n$#依次逐个存入符号栈S中，直到遇到栈顶符号a_i的优先性大于待输入符号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$a_{k-1}<a_k$为止。

(3) 由句柄$a_k\ldots a_i$在文法中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(1), (2), (3)步骤直到归约完输入符号串，栈中只剩文法的开始符号为止。

文法G[S]：

(1) S → bAb

(2) A → (B|a

(3) B → Aa)

步骤	符号栈	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

简单优先关系矩阵

对输入串b(aa)#的简单优先分析过程

5.3 算符优先分析

1) 这是一种经典的自底向上分析法，简单直观，并被广泛使用，开始主要是对表达式的分析，现在已不限于此。可以用于一大类上下无关的文法.

2) 称为算符优先分析是因为这种方法是仿效算术式的四则运算而建立起来的，作算术式的四则运算时，为了保证计算结果和过程的唯一性，规定了一个统一的四则运算法则，规定运算符之间的优先关系。

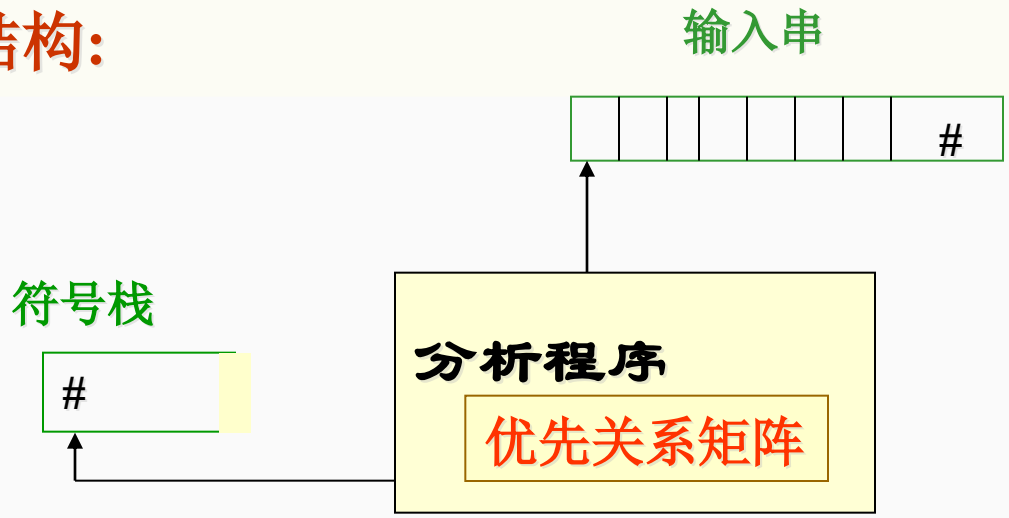
1. 乘除的优先大于加减
2. 同优先级的运算符左大于右
3. 括号内的优先级大于括号外

于是： 4+8-6/2*3 运算过程和结果唯一。

3)算符优先分析的特点:

仿效四则运算过程，预先规定相邻终结符之间的优先关系，然后利用这种优先关系来确定句型的可归约串 ，并进行归约。

4)分析器结构:



(1)直观算符优先分析法

二义性文法G[E]：E→E+E|E-E|E*E|E/E|E↑E|(E)|i

1) i的优先级最高
2) ↑□先□次于i，右结合
3) *和/优先级次之，左结合
4) +和-优先级最低，左结合
5) 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
6) #的优先性低于与其相邻的算符

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

算符优先关系表

- 所给表达式文法虽然是二义性的，但我们人为直观地给出运算符之间的优先关系且这种优先关系是唯一的。
- 有了算符优先关系表，对一般表达式的输入串的计算结果就是唯一的。
- 类似地，通过定义终结符之间的优先关系，对一个输入串的归约过程就是唯一的。

(2) 算符优先文法的定义

【算符文法定义】

设有一文法G，如果G中没有形如A→…BC…的产生式，其中B和C为非终结符，则称G为算符文法(或称OG文法)。

〖要点〗任何一个产生式中都不包含两个非终结符相邻的情况，就是是算符文法。 或：两个非终结符之间一定通过1个或多个终结符相连。

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：如果Ab或（bA）出现在算符文法的句型γ中，其中$A \in V_N$ $b \in V_T$，则γ中任何含b的短语必含有A。

（含b的短语必含A，含A的短语不一定含b）

【算符优先关系的定义】

设G是一个不含 ε 产生式的算符文法，a和b是任意两个终结符，A，B，C是非终结符，算符优先关系如下：

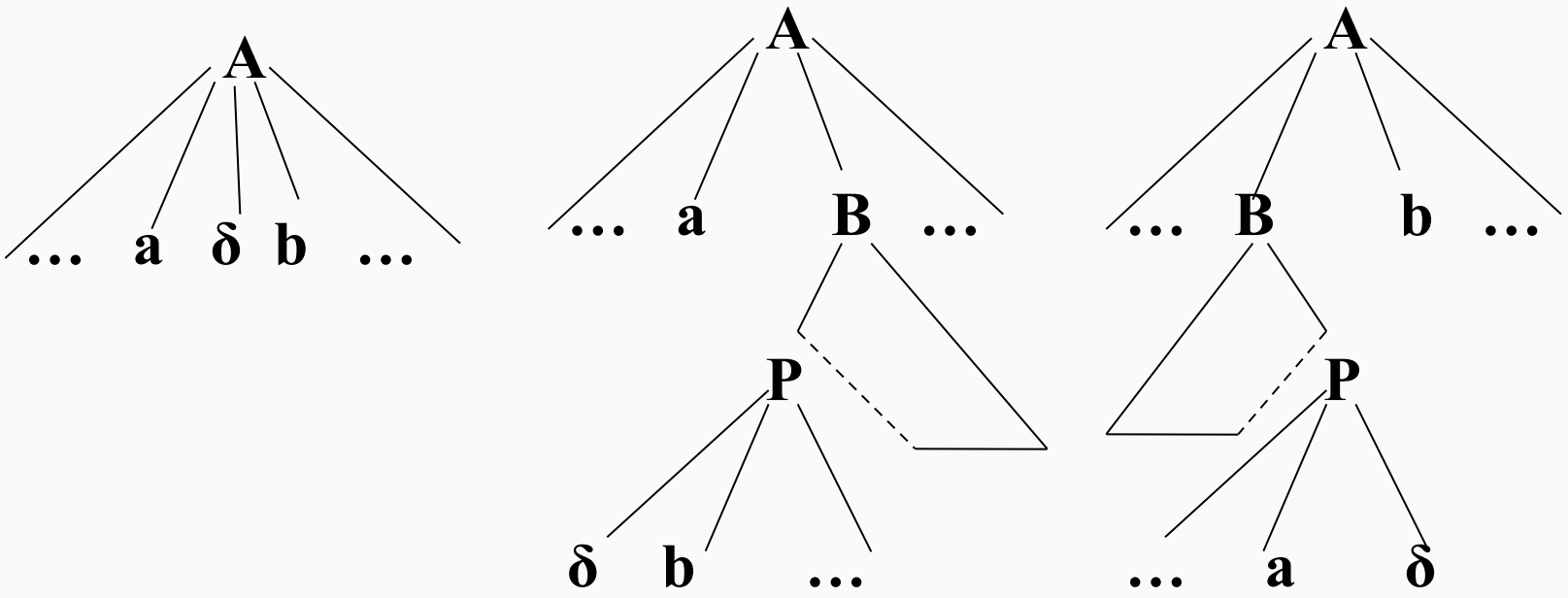
(1)a=b当且仅当G中含有形如A→…ab…或A→…aBb…的产生式;

(2)a<b当且仅当G中含有形如A→…aB…的产生式，且$B \Rightarrow^{+} b\ldots$或$B \Rightarrow^{+} Cb\ldots$;

(3)a>b当且仅当G中含有形如A→…Bb…的产生式，且$B \Rightarrow^{+} \ldots a$或$B \Rightarrow^{+} \ldots aC$ 。

与简单优先关系区别：区分终结符与非终结符，非终结符忽略不计

由语法树结构决定优先性：



a=b
δ为ε或B

a<b
δ为ε或C

a>b
δ为ε或C

【算符优先文法的定义】

设有一不含ε□生式的算符文法G，如果对任意两个终结符a，b之间至多只有= ，<和>三种关系的一种成立，则称G是一个算符优先文法(也称OPG文法)。

即a = b，a <b，a > b只有一种成立，但允许a <b，b < a同时存在。

例：E→E+E | E*E | (E) | i　　证明不是OPG文法。

因为：E→E+E，　$E \overset{+}{\Rightarrow} E*E$　则有 + < *

又因为：E→E*E, $E \overset{+}{\Rightarrow} E+E$　则有 + > *

所以不是算符优先文法。

(3) 算符优先关系表的构造

用表格形式来表示各终结符号的优先关系，这种表称为优先关系表。

构造优先关系表的方法：①按照定义来构造

②按关系图来构造

✓构造步骤：（根据算符优先关系的定义）

- 定义两个集合：firstVT集合lastVT集合。

firstVT(B)={b|B$\overset{+}{\Rightarrow}$b… 或 B$\overset{+}{\Rightarrow}$Cb…}

lastVT(B)={a|B$\overset{+}{\Rightarrow}$…a 或 B$\overset{+}{\Rightarrow}$…aC}

三种优先关系的计算：

a) = 关系：

A→…ab…

A→…aBb…

} 则 a = b

b) <关系：

对于每个非终结符B的firstVT(B)有形如A→…aB…中，对每一个 b∈firstVT(B)。

} 则 a < b

c) >关系：

每个非终结符B的lastVT(B)有形如A→…Bb…中，对每一个a∈lastVT(B)

则a > b

例：为文法

E'→#E#　　T→F

E→E+T　　F→P↑F|P

E→T　　P→（E ）

T→T*F　　P→i

构造算符优先关系表

根据定义构造FIRSTVT(A)的算法

1)若有产生式A → a…或A → Ba…，则a∈FIRSTVT(A),
其中A为非终结符，a为终结符。

2)若有产生式A → B…，则FIRSTVT(B) ⊂FIRSTVT(A),
其中，A、B为非终结符，a为终结符。

根据定义构造LASTVT(A)的算法

1)若有产生式A → …a或A → …aB ， 则a∈LASTVT(A),
其中A为非终结符，a为终结符。

2)若有产生式A → …B，则LASTVT(B)⊂LASTVT(A),
其中，A、B为非终结符，a为终结符。

关系图法构造FIRSTVT(A)的方法

1) 图中结点为每个非终结符的FIRSTVT集或终结符。
2) 若有产生式A → a…或A → Ba… ，其中A为非终结符，a为终结符。则构造由FIRSTVT(A)结点到终结符结点a用箭弧连接的图形。
3) 若有产生式A → B…，其中，A、B为非终结符，a为终结符。则构造由FIRSTVT(A)结点到FIRSTVT(B)结点用箭弧连接。
4) 若某一非终结符A的FIRSTVT(A)结点经箭弧有路径能够到达某终结符结点a，则a∈FIRSTVT(A)。

关系图法构造LASTVT(A)的方法

1) 图中结点为每个非终结符的LASTVT集或终结符。
2) 若有产生式A → …a或A → …aB，其中A为非终结符，a为终结符。则构造由LASTVT(A)结点到终结符结点a用箭弧连接的图形。
3) 若有产生式A → …B，其中，A、B为非终结符，a为终结符。则构造由LASTVT(A)结点到LASTVT(B)结点用箭弧连接。
4) 若某一非终结符A的LASTVT(A)结点经箭弧有路径能够到达某终结符结点a，则a∈LASTVT(A)。

构造优先关系矩阵的算法

FOR 每条规则$U \rightarrow X_1X_2\ldots X_n$ DO

FOR i:=1 TO n-1 DO

BEGIN

IF x_i和x_{i+1}均为终结符,THEN 置 $x_i=x_{i+1}$

IF $i \leq n-2$，且x_i和x_{i+2}都为终结符号但x_{i+1}为非终结符号 THEN 置 $x_i=x_{i+2}$

IF x_i为终结符号x_{i+1}为非终结符号 THEN

FOR FIRSTVT(x_{i+1})中的每个b DO

置$x_i<b$

IF x_i为非终结符号x_{i+1}为终结符号 THEN

FOR LASTVT(x_i)中的每个a DO

置$a>x_{i+1}$

END

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|P
(6) P→(E)
(7) P→i

FIRSTVT(E')={#}
FIRSTVT(E)={+,*,↑,(,i}
FIRSTVT(T)={*,↑,(,i}
FIRSTVT(F)={↑,(,i}
FIRSTVT(P)={(,i}
LASTVT(E')={#}
LASTVT(E)={+,*,↑,),i}
LASTVT(T)={*,↑,),i}
LASTVT(F)={↑,),i}
LASTVT(P)={),i}

	+	*	↑	(	)	i	#
+	>	<	<	<	>	<	>
*	>	>	<	<	>	<	>
↑	>	>	<	<	>	<	>
(	<	<	<	<	=	<	
)	>	>	>		>		>
i	>	>	>		>		>
#	<	<	<	<		<	=

1)'='关系
由产生式(0)和(6),得
#=#, （=）
2）'<'关系
找形如：A→…aB…的产生式
#E：则#<FIRSTVT(E)
+T：则+<FIRSTVT(T)
F：则<FIRSTVT(F)
↑F:则 ↑<FIRSTVT(F)
(E：则 (<FIRSTVT(E)
3)'>'关系
找形如：A→…Bb…的产生式
E# ,则 LASTVT(E)>#
E+ ,则 LASTVT(E)>+
T* ,则 LASTVT(T)>*
P↑ ,则 LASTVT(P)>↑
E) ,则 LASTVT(E)>)

(4) 算符优先分析算法

有了算符优先分析表，就可以对任意给定的符号串进行归约分析，进而判断输入串是否为该文法的句子。

算符文法的任何一个句型为如下形式：

$\#N_1a_1N_2a_2\ldots\ldots N_na_nN_{n+1}\#$

其中$N_i(1\leq i\leq n+1)$为非终结符或空，$a_i(1\leq i\leq n)$为终结符。

若有句型…$N_i a_i \ldots N_j a_j N_{j+1}$…，当$a_i \ldots N_j a_j$属于句柄，则$N_i$和$N_{j+1}$也在句柄中。该句柄中终结符之间的关系为：

$a_{i-1} < a_i$

$a_i = a_{i+1} = \ldots = a_{j-1} = a_j$

$a_j > a_{j+1}$

算符优先文法在归约过程中只考虑终结符号之间的优先关系确定句柄，而与非终结符无关，只需知道当前句柄归约为某一非终结符，不必知道该非终结符的名字是什么，这样也就去掉了单非终结符的归约。

文法G[E]：E→E+E|E-E|E*E|E/E|E↑E|(E)|i

步骤	符号栈	输入符号串	动作
1）	#	i+i*i#	#<i,移进
2）	#i	+i*i#	#<i>+,规约
3）	#E	+i*i#	#<+,移进
4）	#E+	i*i#	+<i,移进
5）	#E+i	*i#	+<i>*,规约
6）	#E+E	*i#	+<*,移进
7）	#E+E*	i#	*<i,移进
8）	#E+E*i	#	*<i>#,规约
9）	#E+E*E	#	+<*>#,规约
10）	#E+E	#	#<+>#,规约
11）	#E	#	接受

对输入串i+i*i的算符优先分析过程

算符优先关系表

实例比较

- 算符优先归约 （表5.8）
- 规范归约（表5.7）
- 算符优先规约去掉单非终结符的归约，并且在归约时不考虑非终结符的名字，因此得到的不是真正的语法树，而是语法树的框架。（比较图5.7与图5.6 ）
- 算符优先分析法的可归约串不是句柄而是最左素短语。
- “算符优先分析的关键是如何找最左素短语”

最左素短语

定义：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，并除自身外不包含其它素短语，最左边的素短语称最左素短语。

与句柄的区别：至少包含一个终结符。（从而去掉了单非终结符的归约）

例：文法G[E]: E→E+T|T

T→T*F|F

F→P↑F|P

P→（E）|i

句型#T+T*F+i#的语法树如下：

根据语法树可知：

句型#T+T*F+i#的短语有：

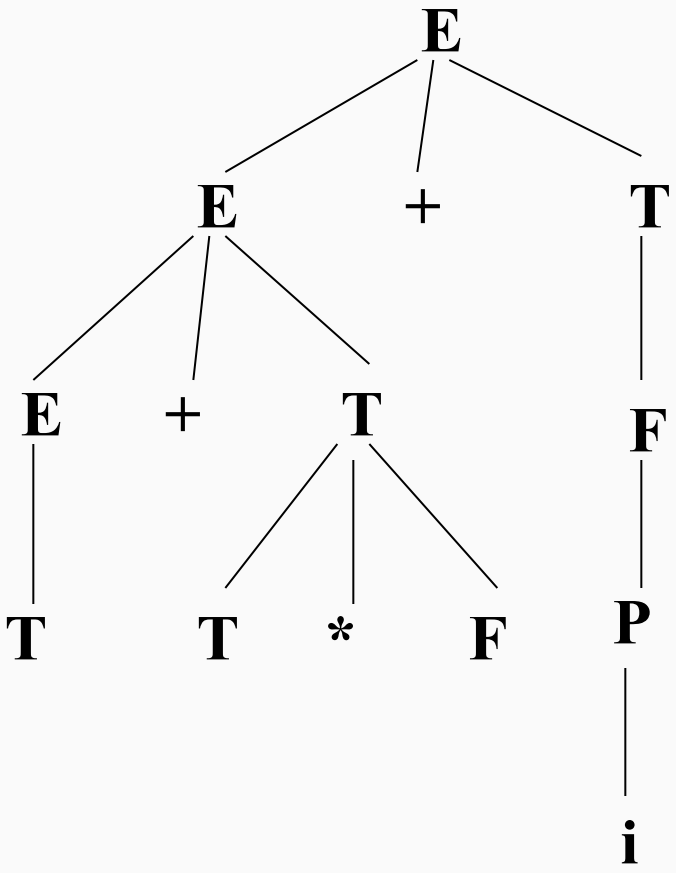
T — 相对非终结符E的短语

T*F — 相对非终结符T的短语

T+T*F — 相对非终结符E的短语

i — 相对非终结符P、F、T的短语

T+T*F+i —相对非终结符E的短语



根据素短语的定义可知：

i和T*F为素短语。

其中：T+T*F (含其他T*F素短语)和

T+T*F+i 不是素短语。

T*F为最左素短语。

T为句柄——最左直接短语

算符优先分析法的关键：

如何确定当前句型的最左素短语？

算符文法的任一句型形式为#$N_1a_1N_2a_2\ldots N_na_nN_{n+1}$#

($N_i \in V_N$ 或$N_i = \varepsilon$, $a_i \in V_T$)

定理：一个OPG句型的最左素短语是满足下列条件的最左子串：$a_{j-1}N_ja_j\ldots N_ia_iN_{i+1}a_{i+1}$

其中 $a_{j-1} < a_j$

$a_j = a_{j+1}$, $a_{j+1} = a_{j+2}$,…, $a_{i-2} = a_{i-1}$, $a_{i-1} = a_i$

$a_i > a_{i+1}$

根据该定理,要找句型的最左素短语就是要找满足上述条件的最左子串.

算符优先分析法的实现：

详见P118 图5.8
基本部分是找句型的最左子串（最左素短语）
并进行归约。

(5) 优先函数：

算符优先关系表示法 { 矩阵表示法(空间大)
优先函数法

1、优先函数的定义：

当a = b，令f(a)=g(b)
当a<b，令f(a)<g(b)
当a>b，令f(a)>g(b)

} f(a)、g(b)为优先函数，函数值用整数表示

2、由定义构造优先函数

构造规则

a) 对终结符$a \in V_T$(包括#号)令f(a)=g(a)=1(初始化)

b)如果a > b，而f(a)≤g(b)，则令f(a)=g(b)+1

c)如果a <b，而f(a)≥g(b)，则令g(b)= f(a)+1

d)如果a=b，而f(a)≠g(b)则令
min{f(a),g(b)}=max{f(a),g(b)}

重复b)~d)过程，直到收敛。若重复过程中有一个值>2n(n为终结符个数)，则该文法不存在算符优先函数。

例1：有优先表如下，构造优先函数。

	+	*	↑
+	>	<	<
*	>	>	<
↑	>	>	<

【解】赋初值

	+	*	↑
f	1	1	1
g	1	1	1

最终结果为：

	+	*	↑
f	2	4	4
g	1	3	5

例2：已知优先关系表，构造优先函数。

	+	*	↑	i	(	)	#
+	>	<	<	<	<	>	>
*	>	>	<	<	<	>	>
↑	>	>	<	<	<	>	>
i	>	>	>			>	>
(	<	<	<	<	<	=	
)	>	>	>			>	>
#	<	<	<	<	<		=

构造过程

(1) 初始化

	+	*	↑	i	(	)	#
f	1	1	1	1	1	1	1
g	1	1	1	1	1	1	1

(2)第一次迭代

	+	*	↑	i	(	)	#
f	2	4	4	6	1	6	1
g	2	3	5	5	5	1	1

(3)第二次迭代

	+	*	↑	i	(	)	#
f	3	5	5	7	1	7	1
g	2	4	6	6	6	1	1

(4)第三次迭代同第二次，上图即为最终结果。

3、用关系图法构造优先函数

构造规则：

a) 对所有终结符$a \in V_T$(包括#号)用有下脚标的f_a，g_a为结点名，画出2n个结点

b)如果$a_i > a_j$或$a_i = a_j$，则从f_{ai}到g_{aj}画一条箭弧

如果$a_i < a_j$或$a_i = a_j$，则从g_{aj}到f_{ai}画一条箭弧

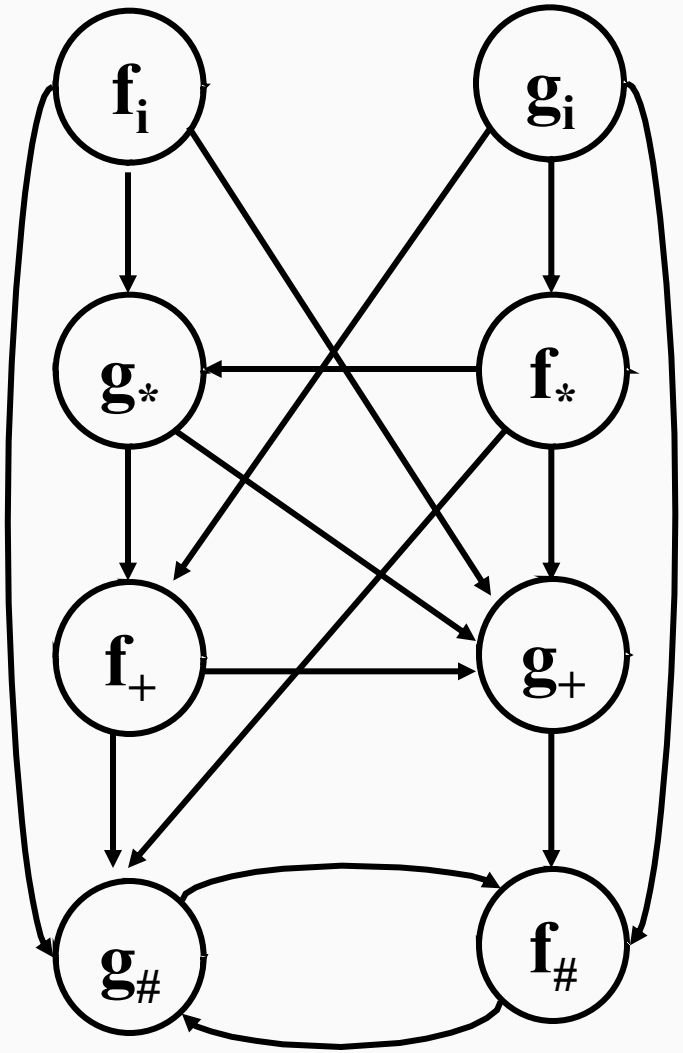
c)给每个结点赋一个数，此数等于从该结点出发所能到达的结点（包括该结点自身在内）的个数。赋给结点f_{ai}的数，就是$f(a_i)$的值，赋给g_{aj}的数，就是$g(a_j)$的值。

d)对构造出的优先函数，按优先关系矩阵检查一遍是否满足优先关系的条件，若不满足时，则说明在关系图中存在3个或3个以上结点的回路，不存在优先函数。

例3：已知优先关系表，构造优先函数。

	i	*	+	#
i		>	>	>
*	<	>	>	>
+	<	<	>	>
#	<	<	<	=

构造优先关系图如下：



	i	*	+	#
i		>	>	>
*	<	>	>	>
+	<	<	>	>
#	<	<	<	=

根据优先关系图求得的优先函数结果如下：

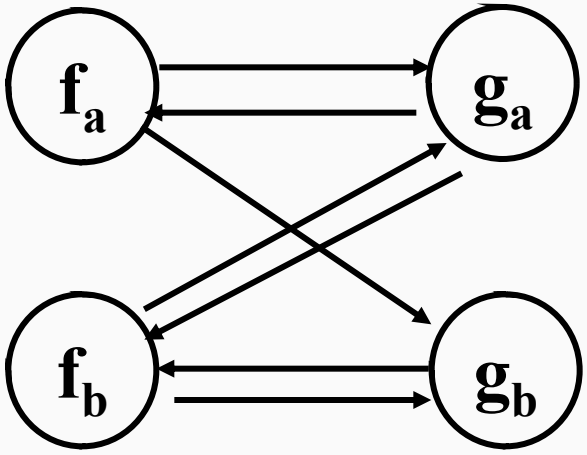
	i	*	+	#
f	6	6	4	2
g	7	5	3	2

其优先函数的优先关系与优先矩阵的优先关系是一致的。

例4：已知优先关系表，构造优先函数。

	a	b
a	=	>
b	=	=

构造优先关系图如下：



根据优先关系图求得的优先函数结果如下：

	a	b
a	4	4
b	4	4

其优先函数的优先关系与优先矩阵的优先关系相矛盾，所以不存在优先函数。

优先函数的缺点

- 在利用优先函数矩阵进行优先分析时，当两个终结符对无优先关系的情况时优先关系矩阵的相应元素为出错信息，而用优先函数进行优先分析时，对两个终结符对没有优先关系的情况不能区分，因而出错时不能准确地指出错误位置。

(6) 算符优先分析法的局限性

由于算符优先分析法去掉了单非终结符之间的归约，尽管在分析过程中，当决定是否为句柄时采取一些检查措施，但仍难完全避免把错误的句子得到正确的归约。

通常一个使用语言的文法很难满足算符优先文法的条件，因而致使算符优先分析法仅适用于表达式的语法分析。

作业

- P121 练习1