编译原理

第6章：自底向上分析方法

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可归约串时，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，称为归约。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。

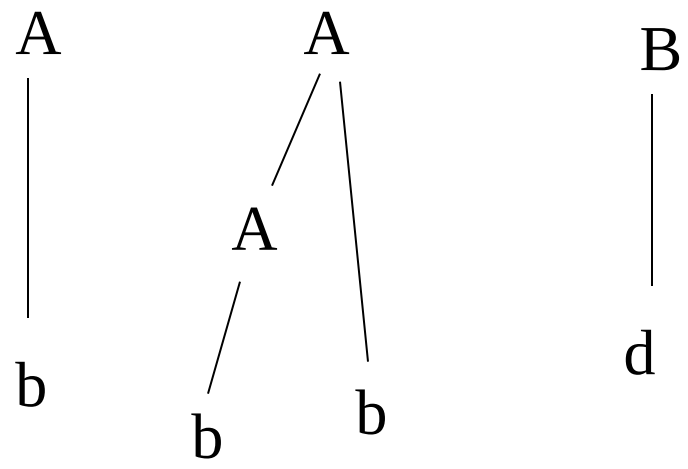
关键问题

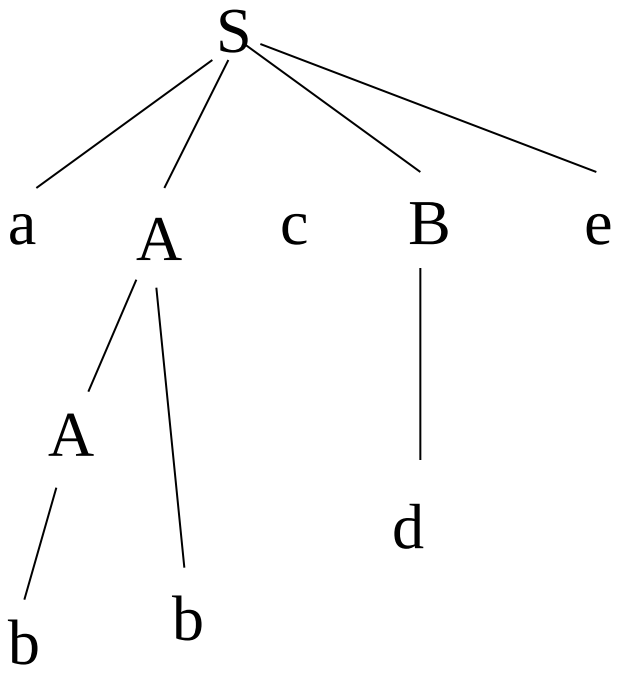
例1：文法

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

自底向上的移进-规约过程（规范规约）是自顶向下最右推导（规范推导）的逆过程

输入串abbcde#分析

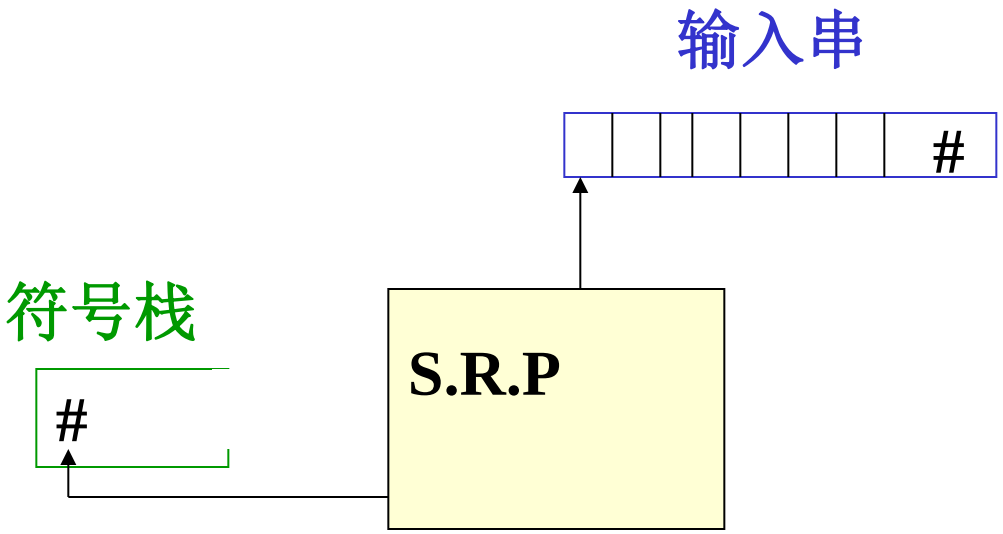


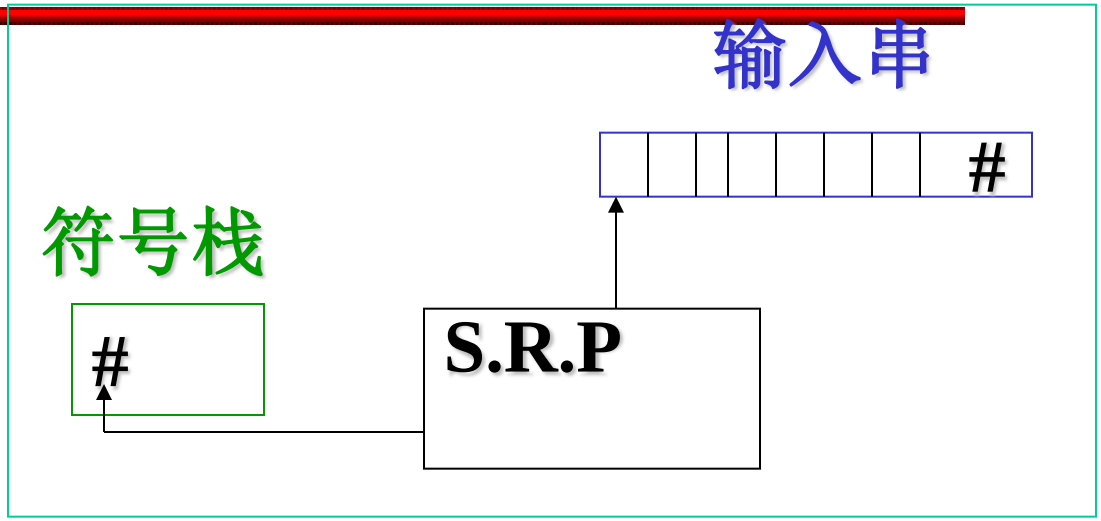


$S \Rightarrow aAcBe \Rightarrow aAcde \Rightarrow aAbcde \Rightarrow abbcde$

移进—规约分析（Shift-reduce parsing)

要点：建立符号栈，用来纪录分析的历史和现状，并根据所面临的状态，确定下一步动作是移进还是规约。





分析过程：

把输入符号串从左向右一一地移进符号栈，

检查栈中符号，当在栈顶的若干符号形成当前句型的可归约串时，就根据规则进行规约，将句柄从符号栈中弹出，将相应的非终结符号压入栈内（即规则的左部），再检查栈内符号串是否形成新的可归约串，若有就再进行规约，否则移进符号。分析一直进行到读到输入串的右界符为止。若栈中仅含有左界符号和识别符号，则表示分析成功，否则失败。

例1：文法

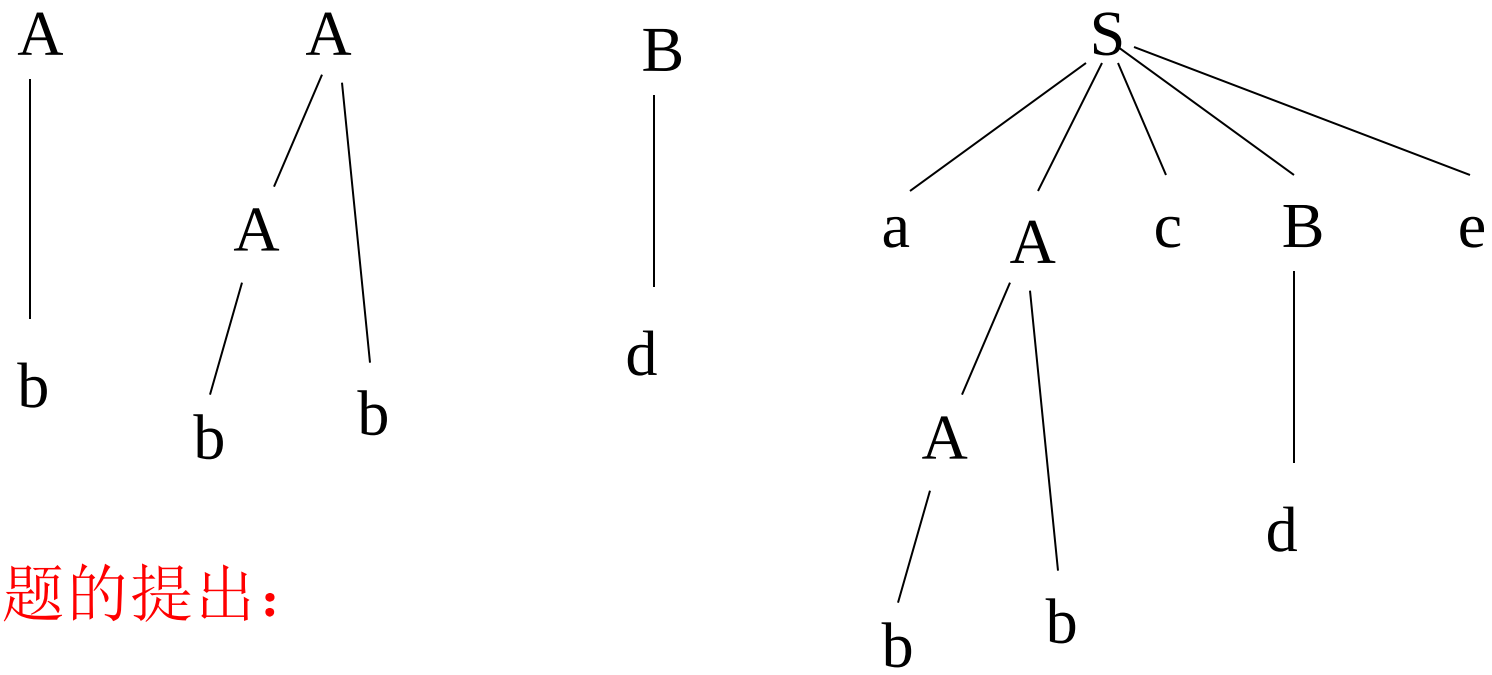
$$\begin{cases} S \rightarrow aAcBe \\ A \rightarrow b \\ A \rightarrow Ab \\ B \rightarrow d \end{cases}$$

输入串abbcde#分析

归约分析过程（移进归约）：

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	#	规约S→aAcBe
11	#S	#	接受

上述的每一步规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，何时规约？

当可归约串出现在栈顶符号串中就可以规约。

②如何知道在栈顶符号串中已经形成可归约串？

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(最左直接短语)
- 算符优先分析法：寻找最左素短语

一、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。

①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——从左向右的规约）；

②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）

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$A \overset{+}{\Rightarrow} \beta$，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 →β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。
- **P 44**

二、简单优先分析法

1、优先关系的表示

x =. y 表示x与y的优先关系相等

x ·> y 表示x的优先性大于y

x <· 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 =. Y 当且仅当G中存在产生式A →…XY…（在语法树的同一层）

（2） X <. 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 > . 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=.关系:

由S→bAb ， A→(B， B→Aa)

b =. A， A =. b， (=. B， A =. a， a =.)

(2) 求<.关系：

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

$A \overset{+}{\Rightarrow}$ a 可得 b <. a

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

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

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

(3) 求> .关系:

$A \Rightarrow (B \Rightarrow (Aa) \Rightarrow \cdots)$

$A \Rightarrow (B \Rightarrow \cdots B$

$A \Rightarrow a$

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

$A \overset{+}{\Rightarrow} \ldots B$ 可得 B > . b

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

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

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

$A \overset{+}{\Rightarrow} \ldots B$ 可得 B > . a

优先关系矩阵表示法：

所有符号>.#

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

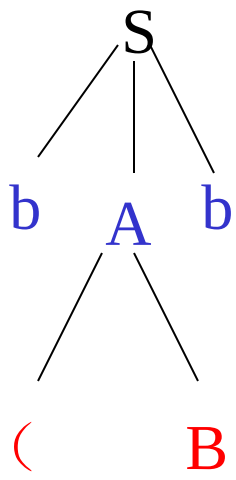
寻找句柄

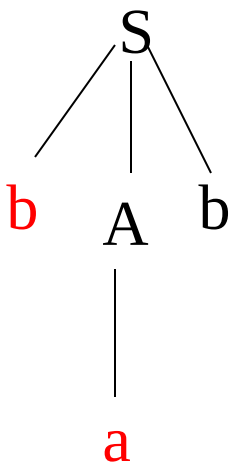
简单优先文法的定义：

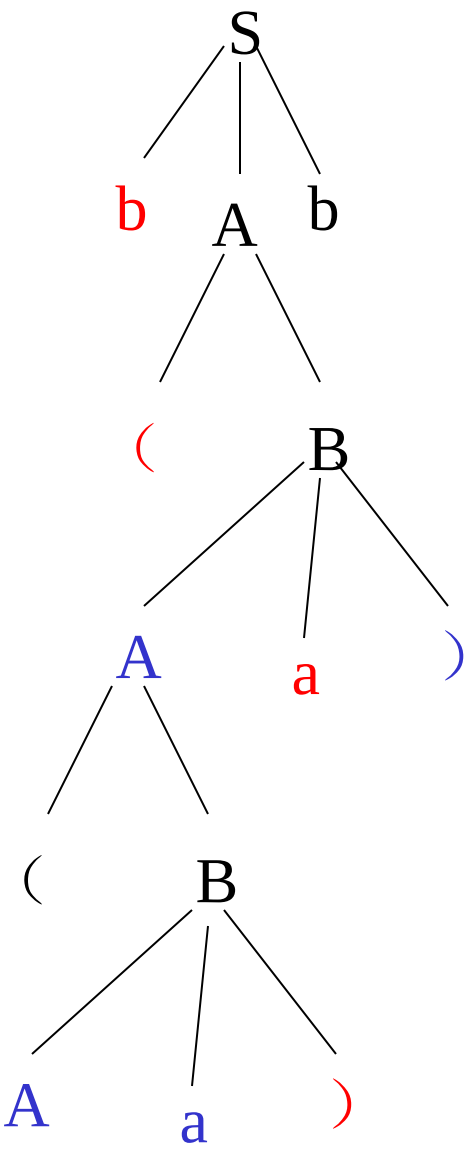
（1）在文法符号集中，任意两个符号之间最多只有一种优先关系；

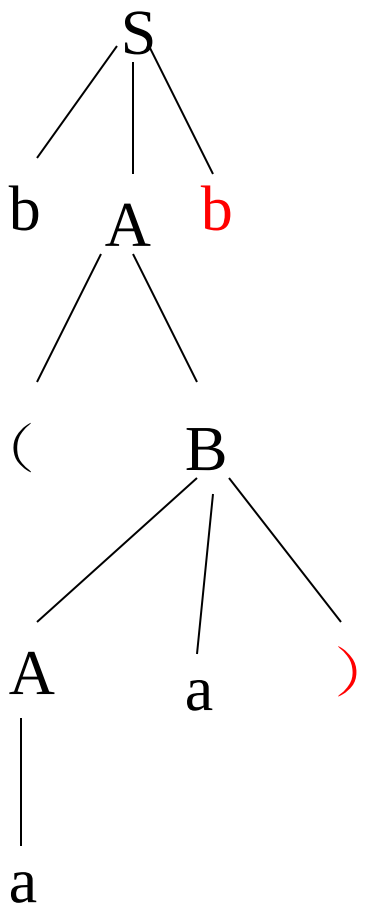
（2）在文法中任意两个产生式没有相同的右部。

语法树结构如下：









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必须先规约。因此b <.(,b <.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(<.右边的(，左边(<. a，左边(<. b。

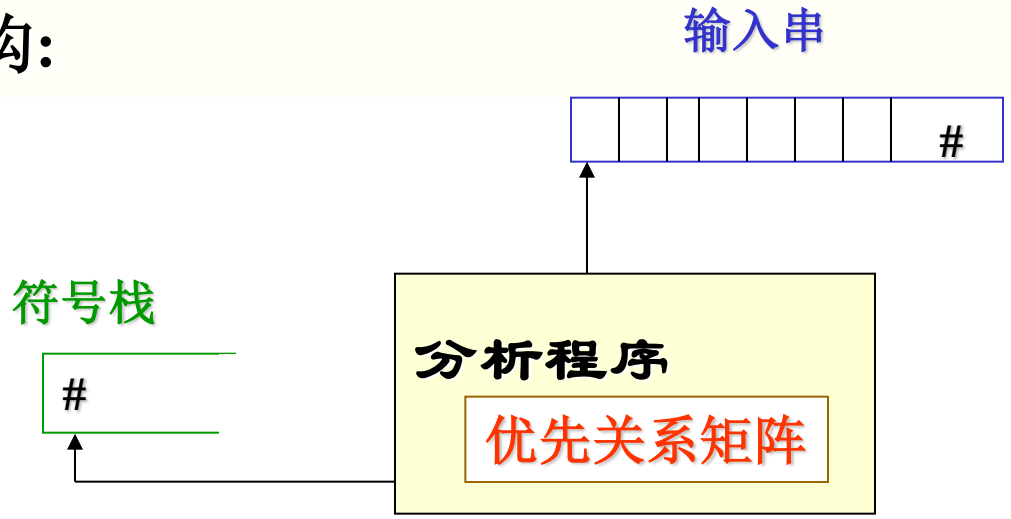
(3) 当ab、aa出现在某一句型中，左边a在句柄中，则右边的a、b不可能在句柄中。因此有左边a >. b，左边a >. a。

- 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。

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：P.105

分析器结构：



三、算符优先分析

1）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。

①简单优先分析法：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——从左向右的规约）；

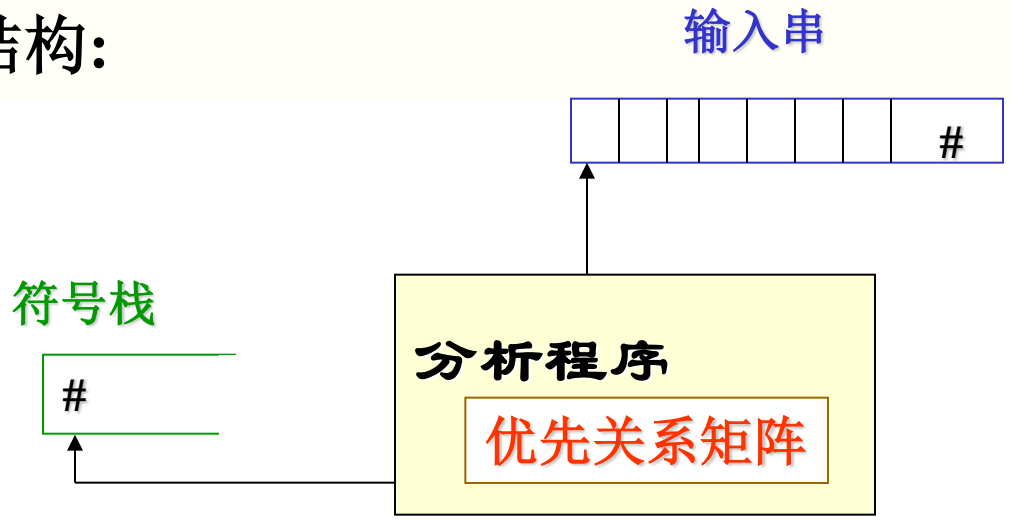
②算符优先分析法：只考虑算符（终结符）之间的优先关系，不考虑非终结符之间的优先关系。按这种关系求出最左素短语。（不规范归约）

算符优先关系的定义

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 ，并进行规约。

4)分析器结构：



- P106 ：表6.3
- 第6步，先乘再加（运算规则），规约

- 运算次序只与运算符有关，与运算对象无关
- 规定算符的优先顺序（优先级别），以及同一个级别中的结合性质 如表6.4

- P106

表 6.3　对输入串 $i_1+i_2*i_3$ 的归约过程

步骤	栈　S	当前输入符	输入串剩余部分	动　作
(1)	#	i_1	$+i_2*i_3$#	移进
(2)	#i_1	+	i_2*i_3#	归约(3)
(3)	#E	+	i_2*i_3#	移进
(4)	#E+	i_2	$*i_3$#	移进
(5)	#$E+i_2$	*	i_3#	归约(3)
(6)	#$E+E$	*	i_3#	移进
(7)	#$E+E*$	i_3	#	移进
(8)	#$E+E*i_3$	#		归约(3)
(9)	#$E+E*E$	#		归约(2)
(10)	#$E+E$	#		归约(1)
(11)	#E	#		接受

*优先于+

表 6.4　算符优先关系表

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

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）

P.108 证明：反证法，如果b不和A一起规约。

【算符优先关系的定义 】

设G是一个算符文法，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$B \overset{+}{\Rightarrow} b\ldots$或$B \overset{+}{\Rightarrow} Cb\ldots$;

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

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

【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，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。

【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，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[illegible]OPG文法)。

即a =. b，a <.b，a >. b只有一种成[illegible]b >. a同时存在。

A →aB ， a＝+， B＝E，
B ⇒Cb…， b＝*

例：E→E+E | E*E | (E) | i　　证明不是OPG文法。

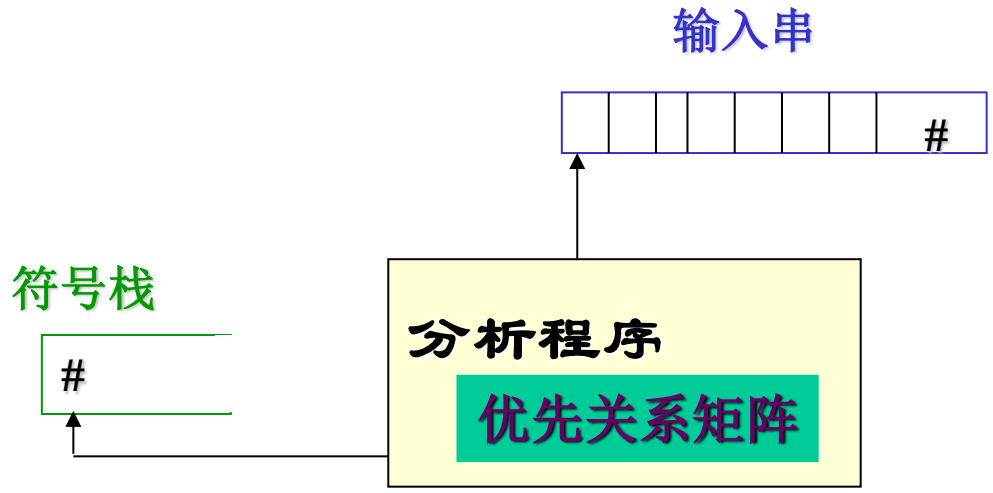
因为：E→E+E ，　$E \overset{+}{\Rightarrow} E*E$　则有 + <. *

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

所以不是算符优先文法。

A →Bb ， B＝E ，
b＝*，
B ⇒ … aC，a＝+

算符优先分析法的实现：



当栈内终结符的优先级＜＝栈外的终结符的优先级时，移进；栈内终结符的优先级＞栈外的终结符的优先级时，归约。表明找到了素短语的尾，再往前找其头，并进行规约。

(3) 算符优先关系表

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|$B\overset{+}{\Rightarrow}b\ldots$ 或 $B\overset{+}{\Rightarrow}Cb\ldots$}

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

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]

构造优先关系矩阵的算法

FOR 每条规则$U \to X_1X_2\ldots X_n$ 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 $i \le n-2$，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

构造FIRSTVT(U)的算法

1)若有规则$U \to b\ldots$或$U \to Vb\ldots$(存在$U \overset{+}{\Rightarrow} b\ldots$或$U \overset{+}{\Rightarrow} Vb\ldots$)
则$b \in FIRSTVT(U)$

2)若有规则$U \to V\ldots$且$b \in FIRSTVT(V)$, 则$b \in FIRSTVT(U)$

说明:因为$V \overset{+}{\Rightarrow} b\ldots$或$V \overset{+}{\Rightarrow} Wb\ldots$,所以有$U \Rightarrow V\ldots \overset{+}{\Rightarrow} b\ldots$或
$U \Rightarrow V\ldots \overset{+}{\Rightarrow} Wb\ldots$

具体方法如下:

设一个栈S和一个二维布尔数组F

F[U,b]=TRUE iff b∈FIRSTVT(U)

```
PROCEDURE  INSERT(U,b)
        IF  NOT  F[U,b]  THEN
            BEGIN
              F[U,b]:=TRUE
              把(U,b)推进S栈  /* b∈FIRSTVT(U) */
              END
BEGIN  {main}
    FOR  每个非终结符号U和终结符b  DO
            F[U,b]:=FALSE    /*赋初值*/
    FOR 每个形如U→b…或U→Vb… 的规则  DO
          INSERT(U,b)
```

```
WHILE   S栈非空      DO
        BEIGN
            把S栈的顶项弹出,记为 (V,b)/* b∈FIRSTVT(V)*/
            FOR    每条形如U→V…的规则   DO
                        INSTER(U,b); /* b∈FIRSTVT(U)*/
        END OF WHILE
END
```

上述算法的工作结果是得到一个二维的布尔数组F,从F可以得到任何非终结符号U的FIRSTVT

$$FIRSTVT(U)=\{b|F[U,b]=TRUE\}$$

构造LASTVT(U)的算法

1.若有规则U→…a或U→=…aV,则a∈LASTVT(U)

2.若有规则U→…V,且a∈LASTVT(V)则a∈LASTVT(U)

```
设一个栈ST,和一个布尔数组B
PROCEDURE    INSERT(U,a)
        IF  NOT  B[U,a]    THEN
            BEGIN
                B[U,a]→TRUE;把(U,a)推进ST栈;
            END;
```

```
BEGIN
    FOR 每个非终结符号U和终结符号a    DO
                B[U,a]:=FALSE;
    FOR  每个形如U→…a或U→…aV的规则  DO
                INSERT (U,a);
    WHILE  ST栈非空   DO
        BEGIN
            把ST栈的栈顶弹出,记为(V,a);
            FOR 每条形如U→…V的规则    DO
                INSERT(U,a);
        END  OF    WHILE;
END;
```

解: (1)求firstVT和lastVT集

firstVT(E')={#}

firstVT(E)={+,*,↑, (, i)

firstVT(T)={*,↑, (, i)

firstVT(F)={↑, (, i)

firstVT(P)={ (, i)

lastVT(E')={#}

lastVT(E)={+,*,↑,) , i }

lastVT(T)={*,↑,) , i }

(0) $E' \rightarrow \# E \#$
(1) $E \rightarrow E + T$
(2) $E \rightarrow T$
(3) $T \rightarrow T * F$
(4) $T \rightarrow F$
(5) $F \rightarrow P \uparrow F \mid P$
(6) $P \rightarrow (E)$
(7) $P \rightarrow i$

firstVT(B)={b|B⇒b…
或 B⇒Cb…}

E'= #E#
E' ⇒#…

E⇒E+T
E⇒T⇒T*F
E⇒T⇒F⇒P↑**F**
E⇒T⇒P⇒(E)
⇒i

lastVT(B)={a|B⇒…a
或 B⇒…aC}

lastVT(F)={) ,↑, i }

lastVT(P)={ i ,) }

(2) 求 =.关系

E’→#E#　　# =.　#

P→(E)　　(=.)

(3) 求 <. 关系

E’ → #E#　则 # <.firstVT(E)

所以　# <.+，　# <.*，　# <.↑，　# <. (，　# <.i

E’ →E+T　则 + <.firstVT(T)

所以 + <.*, + <.↑, + <. (, + <.i

T→T*F　则 * <. firstVT(F)

所以 * <. ↑, * <. (, * <. i

F→P↑F　则↑ <. firstVT(F)

所以 ↑ <. ↑, ↑ <. (, ↑ <. i

P→（E ）　则（ <. firstVT(E)

所以 (<. +, (<. *, (<. ↑, (<.(, (<. i

(4) 求 >.关系

E’ →#E#　则 lastVT(E) >. #

所以 + >. #, * >. # , ↑ >. # ,) >. #, i >. #

E→E+T 则lastVT(E) >. +

所以 + >. +, * >. +, ↑ >. + ,) >. +, i >. +

T→T*F 则lastVT(T) >. *

所以 * >. *, ↑ >. *, i >. *,) >. *

F→P↑F 则lastVT(P) >. ↑ 所以 i >. ↑,) >. ↑

P→(E) 则lastVT(E) >.)

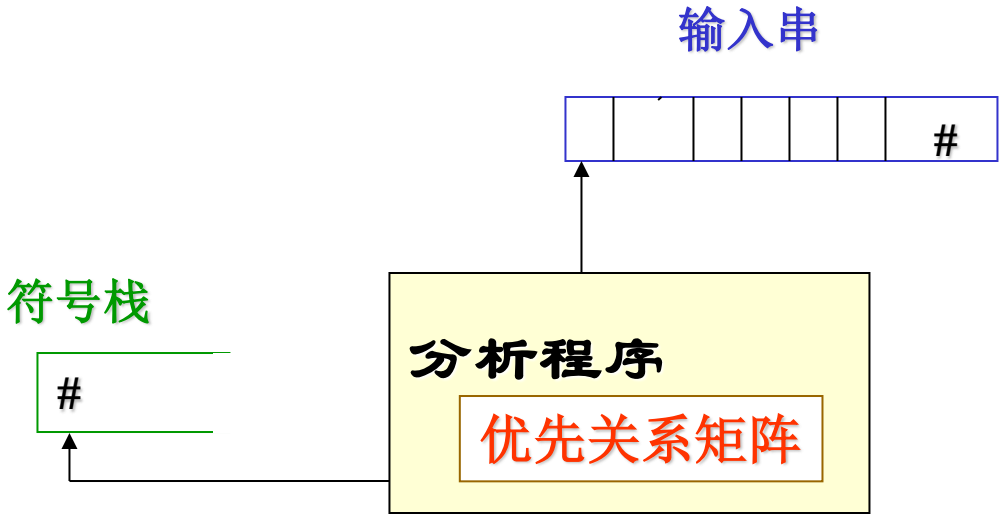
所以 + >.) , * >.) , ↑ >.) , i >.) ,) >.)

算符优先关系表

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

自底向上优先分析

自底向上分析也称移进归约分析（是推导的逆过程）



简单优先文法：确定所有符号优先关系的规则

算符优先文法：确定算符优先关系的规则

firstVT()，lastVT()

作业

- 计算 例6.1 (P102) 文法的简单优先关系矩阵
- P122 练习 1、2