第5章 自底向上优先分析法

退出

5.1 自底向上分析方法的基本思想

自底向上分析方法，也称移进—归约分析法，粗略地说它的实现思想是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时，（该句柄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，这称为一步归约。重复这一过程直到归约到栈中只剩文法的开始符号时则为分析成功，也就确认输入串是文法的句子。可以看出，移进—归约过程是自顶向下最右推导的逆过程。最右推导称为规范推导。自左向右的归约过程称为规范归约。

5.1 自底向上分析方法的基本思想

例：设文法G[S]为：

（1）S→aAcBe

（2）A→b

（3）A→Ab

（4）B→d

对输入串abbced#进行分析，检查该符号串是否是G[S]的句子。

（1） S→aAcBe
（2）A→b
（3）A→Ab
（4）B→d

用移进—归约对输入串abbcde#的分析过程

	符 号 栈	输入符号串	动 作
	#	abbcde#	移 进
	#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

如何知道何时在栈顶符号串中已形成某句型的句柄，这是自底向上分析的关键。在自底向上分析方法中，本章主要介绍常用的算符优先分析法和LR类分析法。

5.2 算符优先分析法

算符优先分析法只考虑算符（广义为终结符）之间优先关系，根据算符之间优先关系确定何时移进，何时归约。

确定算符之间优先关系的方法:

(1) 对一个给定的文法，人为地规定其算符的优先顺序，称为直观算符优先分析法。

(2) 根据文法确定算符之间的优先关系。

5.2.1 算符优先文法的定义

我们首先给出算符文法的定义

定义5.1 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（Operater Grammar）也称OG文法。

例如: 表达式文法 E → E+E | E*E | (E) | i

其中任何一个产生式中都不包含两个非终结符相邻的情况，因此该文法是算符文法。

算符文法有如下两个性质:

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。 证明思路：归纳法

$$S \Rightarrow W1 \Rightarrow \cdots\cdots \Rightarrow Wn\text{-}1 \Rightarrow Wn$$

W1中不含相继非终结符

归纳假设Wn-1中不含相继非终结符，设 $Wn\text{-}1=\alpha A\delta$

由A→β得 $Wn=\alpha\beta\delta$ 中也一定不含相继非结符

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。

证明思路：反证法

若有句型…Ab……

短语,归约到B

则出现句型…AB…

与性质1矛盾。

引入优先关系符号≒，<· ，·> （以后分别以＝， ＜ ，＞代替）若句型形如：……ab……或……aAb……

a=b 表示a与b的优先关系相等

a<b 表示a的优先性比b的优先性小

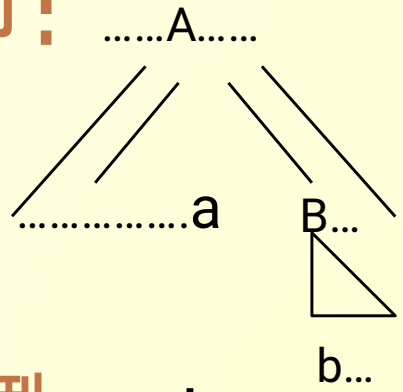
a>b 表示a的优先性比b的优先性大

定义5.2 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=，<，>定义如下：

（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$B \Rightarrow^{+} b$…或$B \Rightarrow^{+} Cb$…

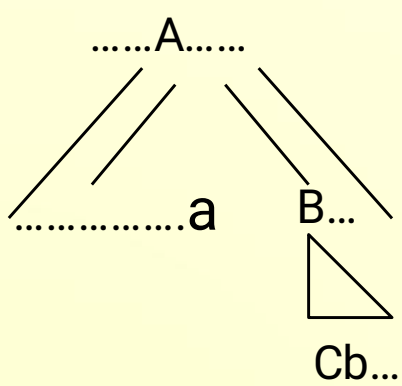
即：



对应句型 ……a b……

应先归约

或

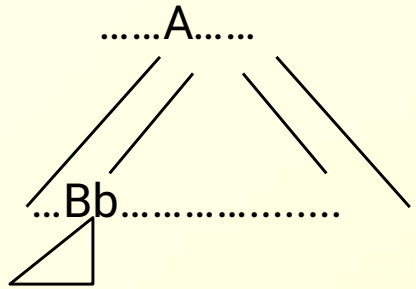


或 句型 ……a Cb……

应先归约

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

即：

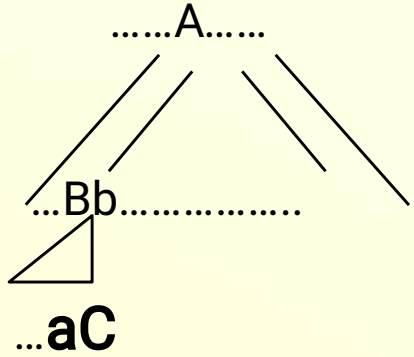


…a

对应句型 ……ab……

应先归约

或



…aC

或 句型 ……aCb……

应先归约

下面给出算符优先文法的定义。

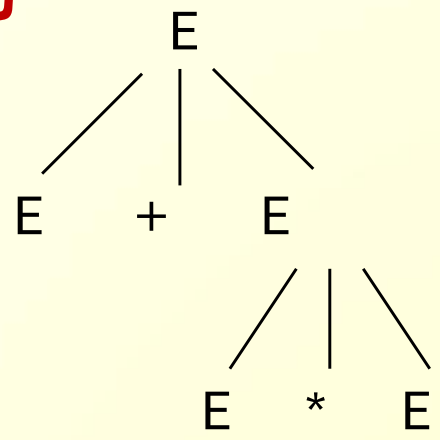
定义5.3 设有一不含ε产生式的算符文法G，如果对任意两个终结符对a, b之间至多只有<、>和=三种关系的一种成立，则称G是一个算符优先文法。

（Operator Precedence Grammar）即OPG文法。

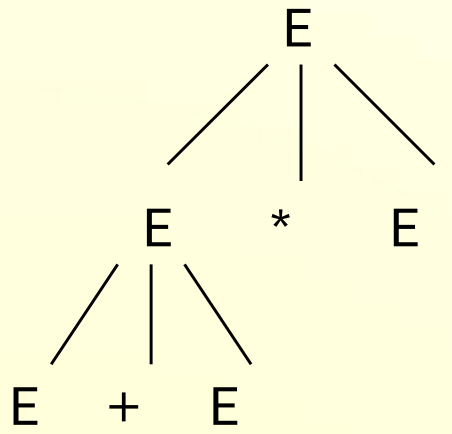
如：E→E+E|E*E|(E)|i

是算符文法但不是算符优先文法。

因为



+< *



+ > *

- 这样+、*的优先关系不唯一。
- 所以该文法不是算符优先文法。

注意：两个终结符之间的优先关系是有序的。

左边的a>右边的b，不一定b<a

a=b　　不一定b=a等。

5.2.2 算符优先关系表的构造

（1）对文法G的每个非终结符B构造两个集合

$FIRSTVT(B)=\{b \mid B \Rightarrow^{+} b\ldots$ 或 $B \Rightarrow^{+} Cb\ldots\}$

构造规则：

（1）若$B \to b\ldots$或$B \to Qb\ldots$，则$b \in FIRSTVT(B)$

（2）若$B \to Q\ldots$，则$FIRSTVT(Q) \subseteq FIRSTVT(B)$

$LASTVT(B)=\{a \mid B \Rightarrow^{+} \ldots a$ 或 $B \Rightarrow^{+} \ldots aC\}$

构造规则

（1）若$B \to \ldots a$或$B \to \ldots aQ$，则$a \in LASTVT(B)$

（2）若$B \to \ldots Q$，则$LASTVT(Q) \subseteq LASTVT(B)$

（2）三种优先关系的计算

（a）=关系： 对形如产生式A→…ab…，

A→…aBb…

则有a=b成立

（b）<关系：对形如产生式A→…aB…中

对每一b∈FIRSTVT(B) 有a<b

（c）>关系：对形如产生式A→…Bb…

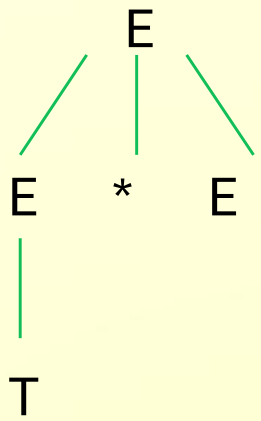
对每一a∈LASTVT(B) 有a>b成立

将文法中终结符之间的优先关系用表格形式列出来，则得文法的优先关系表。(看例题)

5.2.3 最左素短语

· 因为算符优先关系仅定义在终结符号之间，对于某句型的句柄是单个非终结符时（即文法含有规则右部为单个非终结符），则不能用优先关系找句柄。

例如下面语法树，用算符优先关系无法找到句柄T



为此，引进最左素短语概念

1、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，并且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。

例如，若表达式文法G[S]为：

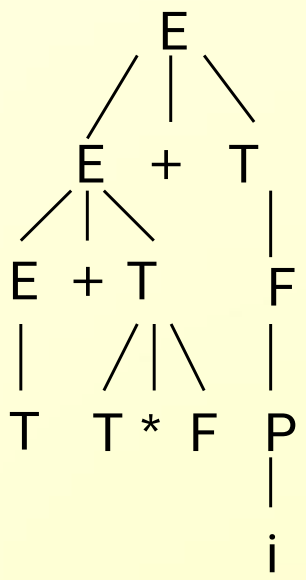
E→E+T|T

T→T*F|F

F→P↑F|P

P→（E）|i

有句型#T+T*F+i#，它的语法树如下图：



由此语法树可发现此句型有如下短语：

T+T*F+i

T+T*F

T

T*F

i

其中素短语有：　i，T*F

最左素短语为：　T*F

2、根据算符优先关系判断最左素短语

- 根据算符文法性质1, 算符文法的句型应为如下形式:

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

  其中N_i为非终结符或空, a_i为终结符。

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r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。

- 根据算符文法性质2, 句型中的短语形式为：

$N_ka_k\ldots N_ma_mN_{m+1}$

（若有N_k，N_{m+1}，则必在此句柄中）

2、根据算符优先关系判断最左素短语

- 根据算符优先关系的定义，句型的最左素短语是满足条件

ai-1 <ai=ai+1=…=aj-1=a j> aj+1 的最左子串:

Niai…NjajNj+1

其中Ni, Nj+1 为非终结符或空。

- 如上面文法G[E]的句型#T+T*F+i#中终结符之间的优先关系是: #<+<*>+<i>#

最左素短语就是： T*F

退出

栈	优先关系	输入串	最左素短语	动 作
#	<	i+i*i#		移进I
# i	>	+i*i#	i	用F→i归约

初始时栈底存#，输入指针指向输入串的首字符。

· 控制程序根据栈顶终结符a（若栈顶是非终结符，则次栈顶的终结符称为栈顶终结符）和输入指针所指的输入符b，查优先关系表M，可能有四种情况：

（1）M[a, b]为<或=时移进b，即将b进栈，输入指针指向下一输入符。

（2）M[a, b]为>时，则将栈顶含a的素短语按对应的产生式归约，素短语与产生式右部需终结符对应相同，非终结符位置应相同名称可不同。顶出栈中素短语，非终结符入栈。

（3）M[a, b]为空白，语法错，调用相应出错处理程序。

（4）a=b=# 时分析结束。（看例题）

5.2.5 优先函数

1、优先函数的概念

每个终结符a与两个优先函数f(a), g(a)相对应。

f(a)对应终结符对中左边的a；

g(a)对应终结符对中右边的a

若a=b 则令f(a)=g(b)

若a<b 则令f(a)<g(b)

若a>b 则令f(a)>g(b)

2、构造优先函数

用关系法构造。 构造步骤：

（a）对每一终结符a（包括#），用fa，ga为结点名。

（b）若ai>aj或ai=aj，则从fai到gaj画一条箭弧。

若ai<aj或ai=aj，则从gaj到fai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，就是函数f(ai)的值，赋给g (aj)的数，就是函数g(aj)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，则在关系图中有回路说明不存在优先函数。

退出

现举例如下：

例1 若已知优先关系矩阵为下表

优先关系矩阵

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

构造优先关系图为下图。

构造

+ < i

+ < *

+ > +

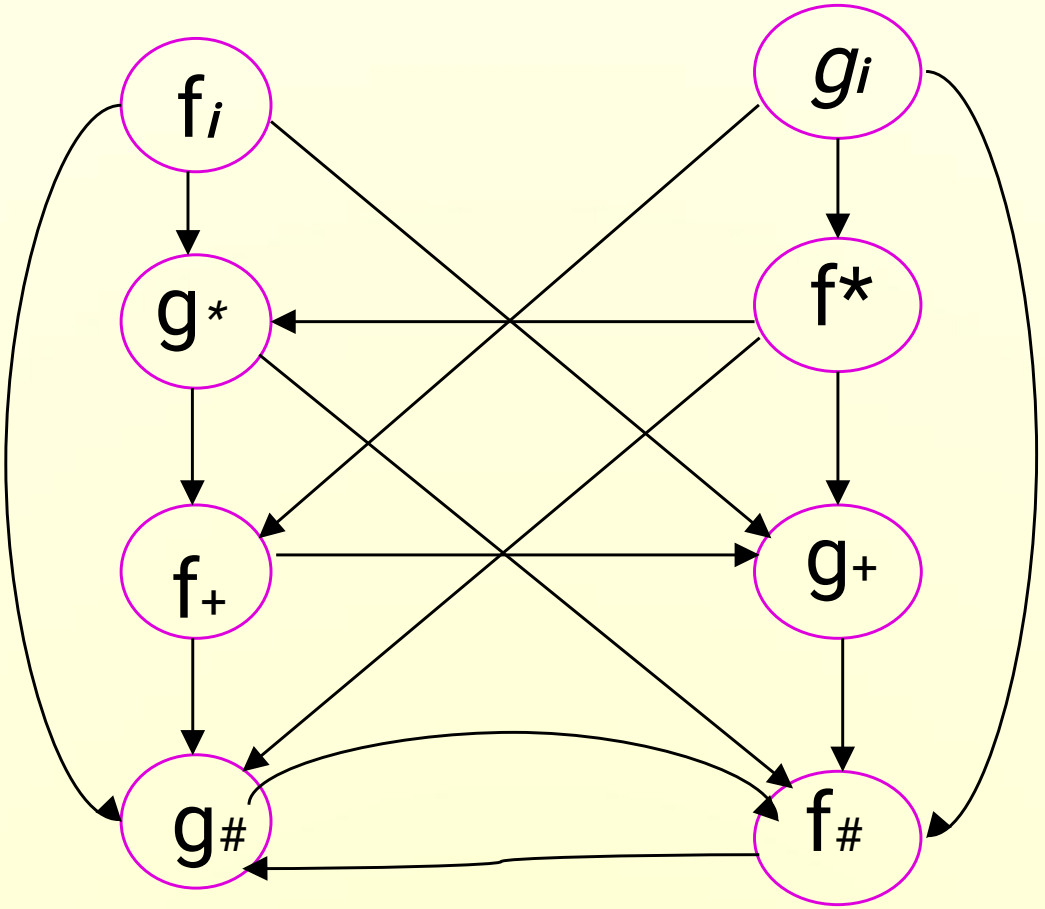
+ > #

< i

< *

< +

=



i> *

i>#

i>+

* < i

* > *

* > +

* > #

给 fi 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， 即 fi, g*,f+, g#, f#, g+，因此 赋给结点fi的数是6。同理可得其它结点的值。

由上图求得的优先函数结果为下表。

表优先函数关系表

	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。

3、优先函数的优缺点

优点：（1）节省存储空间；

（2）执行整数比较运算比查优先关系表方便。

缺点：（1）有些优先关系表不存在优先函数。

（2）原先不存在优先关系的两个终结符变成可比较其函数值大小了，需加以克服。

本章要点：

移进—归约方法

算符优先文法

求非终结符的FIRSTVT和LASTVT

构造算符优先关系表

判断最左素短语

算符优先分析

例：若表达式文法为：

（0）E′→#E#

（1）E→E+T

（2）E→T

（3）T→T*F

（4）T→F

（5）F→（E）

（6）F→i

按任一键继续

返回 退出 下一页

计算优先关系：

产 生 式	优 先 关 系
(0) E′→#E# FIRSTVT（E）={+, *, (, i} LASTVT（E）={+, *,I,)}	#=# #<+, #<*, #<(, #<i +>#,*>#, i>#,)># (1) E→E+T
LASTVT（E）={+, *, i,)} FIRSTVT（T）={*, (, i}	+>+, *>+, i>+,)>+ +<*, +<(, +<I
(3) T→T*F LASTVT（T）={*,), i} FIRSTVT（F）={(, i}	*>*,)>*, i >* *<(, *<i
(5) F→(E) FIRSTVT（E）={+, *, (, i} LASTVT（E）={+, *, I,)}	(=) (<+, (<*, (<(, (<i +>), *>), i>),)>)

(0) E′→#E#
(1) E→E+T
(2) E→T
(3) T→T*F
(4) T→F
(5) F→（E）
(6) F→i

按任一键继续

列出优先关系表为：

左边＼右边	+	*	(	)	i	#
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
#	<	<	<		<	=

返回

例：表达式文法为：

（0）E′→#E#

（1）E→E+T

（2）E→T

（3）T→T*F

（4）T→F

（5）F→（E）

（6）F→i

按任一键继续

返回 退出 下一页

（0）E′→#E#　（1）E→E+T　（2）E→T　（3）T→T*F
（4）T→F　（5）F→（E）　（6）F→i

栈	优先关系	输入串	最左素短语	下步 动 作
#	<	i+i*i#		移进i
#i	>	+i*i#	i	用F→i归约
#F	<	+i*i#		移进 +
#F+	<	i*i#		移进i
#F+i	>	*i#	i	用F→i归约
#F+F	<	*i#		移进*
#F+F*	<	i #		移进i
#F+F*i	>	#	i	用F→i归约
#F+F*F	>	#	F*F	用T→T*F归约
#F+T	>	#	F+T	用E→E+T归约
#E	=	#		结束

返回

退出　上一页