第5章 自底向上优先分析

自顶向下(Top Down)的分析

推导(Derivation)

自底向上(Bottom Up)的分析

归约(Reduce)

复习内容

- 如果 $S \overset{*}{\Rightarrow} \alpha A \beta$ and $A \overset{+}{\Rightarrow} \gamma$，则称 γ 是句型 $\alpha \gamma \beta$ 的相对于变量A的**短语**(Phrase)
- 如果 $S \overset{*}{\Rightarrow} \alpha A \beta$ and $A \Rightarrow \gamma$，则称 γ 是句型 $\alpha \gamma \beta$ 的相对于变量A的**直接（简单）**短语
- 最左直接短语叫做**句柄**(Handle)
- **规范归约**

5.1 自底向上的语法分析

- 自底向上的语法分析
  - 从叶子节点(输入串)出发，反复利用产生式进行**归约**，最后到达根节点。
- 移进-归约的分析法
- 核心
  - 寻找句型中的当前归约对象——“句柄”
  - 将句柄归约为何符号？

自底向上的分析过程

- 例 设文法为：

  S→aAcBe

  A→Ab|b

  B→d

句子分析：

abbcde

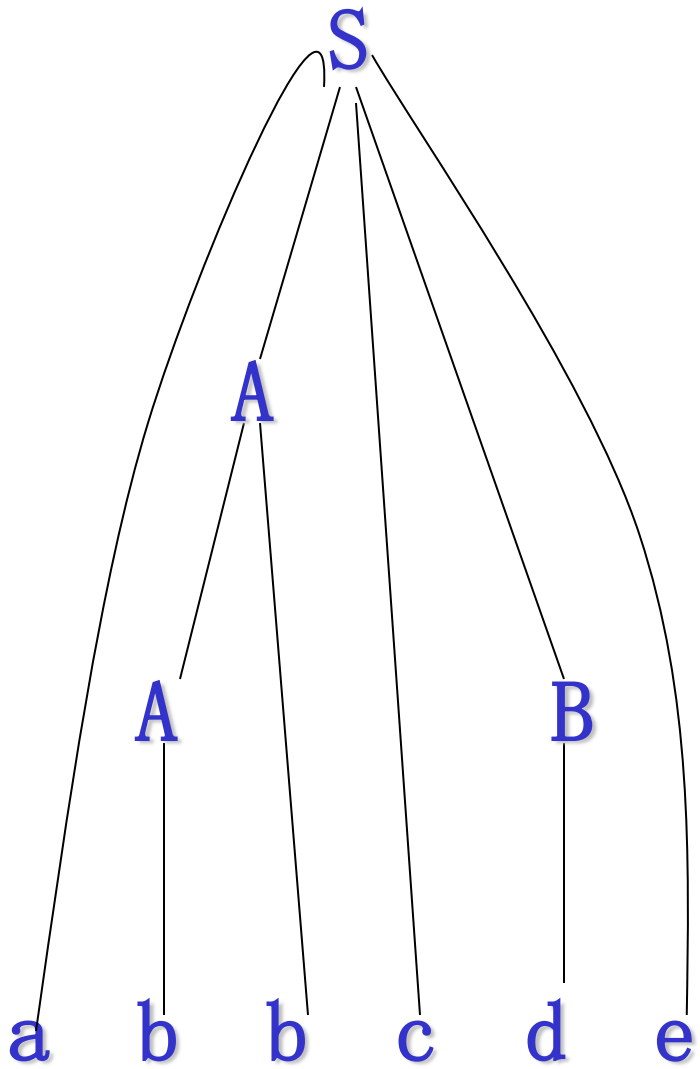
⇐aAbcde

⇐aAcde

⇐aAcBe

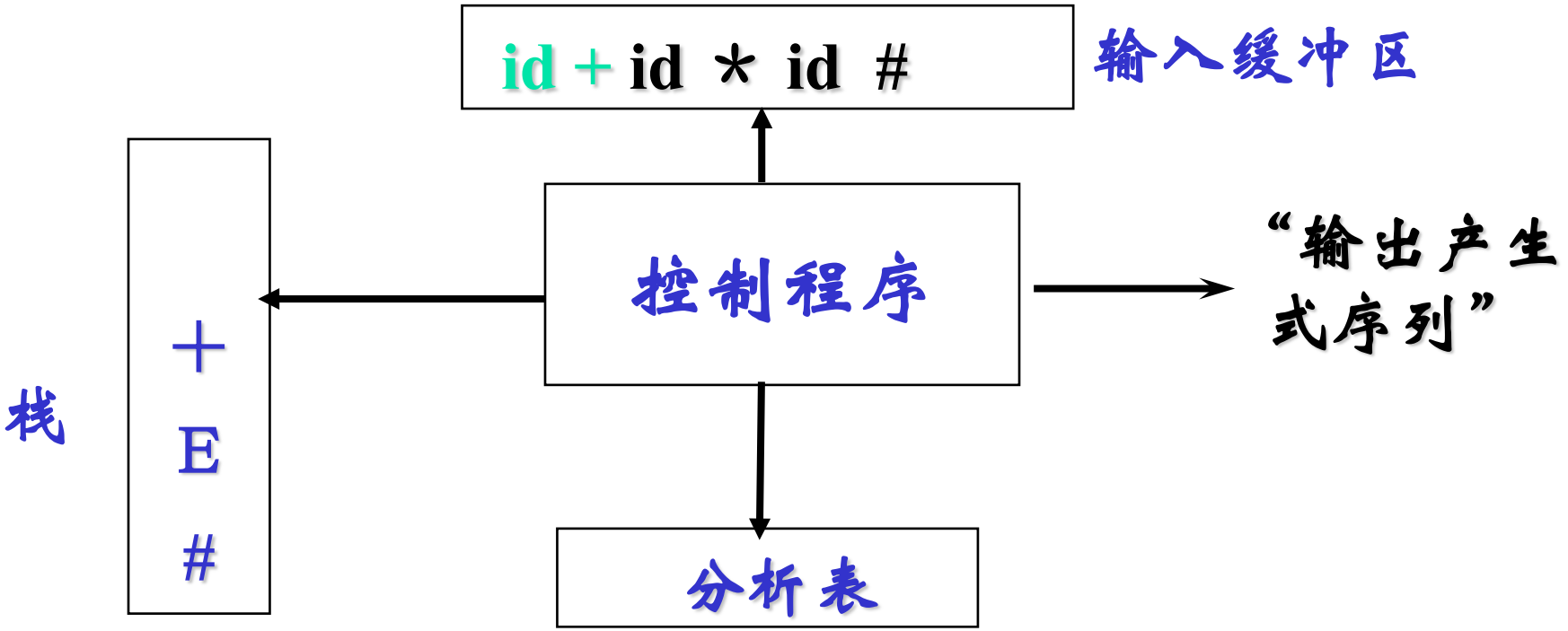
⇐ S

语法树的形成过程



5.1自底向上的分析

1、分析器框架



栈内容+输入缓冲区内容 = # 当前“句型” #

5.1自底向上的分析

- 1、系统框架
  - 控制程序
    - 控制分析过程，输出分析结果——产生式序列
  - 输入缓冲区
    - 保存输入符号串
  - 栈
    - 保存语法符号——已经得到的部分结果
  - 分析表
    - 保存当前分析应当采取的动作及状态转换
  - 栈+输入缓冲区剩余内容=“句型”

5.1自底向上的分析

- 2、系统运行
  - 开始格局
    - 栈：#；输入缓冲区：w#
  - 栈
    - 存放已经分析出来的结果和读入的符号，一旦句柄在栈顶形成，就将其弹出进行归约，并将归约后的符号压入栈
  - 正常结束：栈中为#S，输入缓冲区只有#
  - ？？ 系统如何发现句柄在栈顶形成

5.1自底向上的分析概述

- 3、如何识别句柄？
  - 如何保证找到的直接短语是最左的?利用栈
  - 如何确定句柄的开始处与结束处？
    - 两种分析方法

5.1自底向上的分析概述

- (1)如何识别句柄？--优先法
  - 句柄内相邻符号同时归约，是同优先级的
  - 句柄两个端点符号的优先级要高于句柄外与之相邻的符号的优先级，句柄内相邻符号具有相同的优先级

$$a_1 \ldots a_{i-1} a_i \lessdot a_{i+1} \doteq \ldots \doteq a_{j-1} \doteq a_j \gtrdot a_{j+1} \ldots a_n$$

5.1自底向上的分析概述

- (2)如何识别句柄？---状态法
  - 用状态来描述不同时刻句柄是否形成
  - 因为句柄是产生式的右部，可用产生式来表示句柄的不同识别状态
- 例如：S→bBB可分解为如下识别状态
  - S→.bBB　移进b
  - S→b.BB　等待归约出B
  - S→bB.B　等待归约出B
  - S→bBB.　归约

5.2 算符优先分析法

- 算术表达式分析的启示
- 算符优先关系的直观意义
  - $+ \lessdot *$　　+ 的优先级低于 *
  - $(\doteq)$　　(的优先级等于)
  - $+ \gtrdot +$　　+ 的优先级高于 +
- 方法
  - 将句型中的终结符号当作“算符”，借助于符号之间的优先关系确定句柄

5.2 算符优先分析法

1、算符文法

如果文法G=（V_N,V_T,P,S）中不存在形如

$$A \rightarrow \ldots BC \ldots \ (B,C \in V_N)$$

的产生式，则称之为算符文法(OG—Operator Grammar)

即：如果文法 G 中不存在具有相邻非终结符的产生式，则称为算符文法

5.2 算符优先分析法

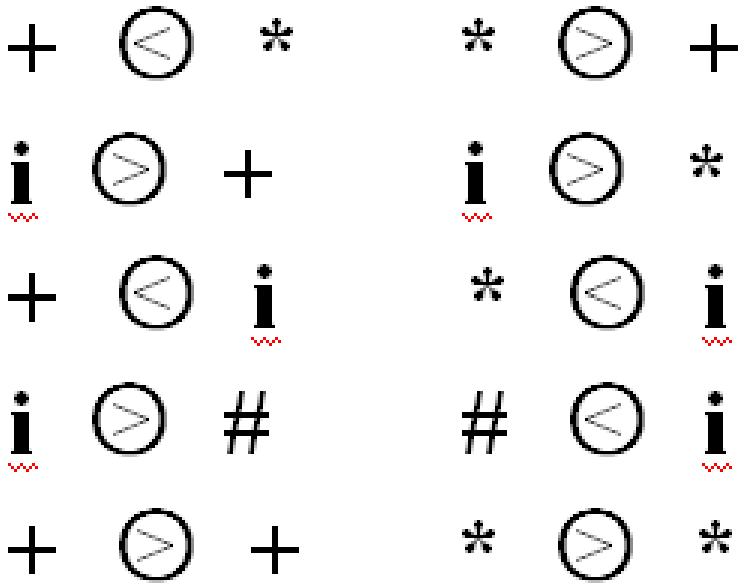
- 2、算符(终结符)间的优先关系
- 设G=（V_N，V_T，P，S）为算符文法，A,B,C∈ V_N，a,b∈ V_T
  - $a \doteq b$ ⇔ A→…ab…或者 A→…aBb…
  - $a \lessdot b$ ⇔ A→…aB…且($B \overset{+}{\Rightarrow} b$…或者$B \overset{+}{\Rightarrow} Cb$…)
  - $a \gtrdot b$ ⇔ A→…Bb…且($B \overset{+}{\Rightarrow}$…a或者$B \overset{+}{\Rightarrow}$…aC)
  - 注意：终结符号之间的优先关系

5.2 算符优先分析法

- 3、算符优先文法
- 设G=（V_N，V_T，P，S）为算符文法，对于 $\forall a,b \in V_T$，$a \doteq b$, $a \lessdot b$, $a \gtrdot b$，至多只有一种优先关系成立，则称之为算符优先文法 (OPG —Operator Precedence Grammar)

5.2 算符优先分析法

表达式文法的算符优先关系



……

5.2 算符优先分析法

- 表达式文法是算符优先文法，算符优先矩阵如下表所示。

	(	i	*	+	)	#
(	⋖	⋖	⋖	⋖	=	
i			⋗	⋗	⋗	⋗
*	⋖	⋖	⋗	⋗	⋗	⋗
+	⋖	⋖	⋗	⋗	⋗	⋗
)			⋗	⋗	⋗	⋗
#	⋖	⋖	⋖	⋖		

5.2 算符优先分析法

- 4、算符优先关系矩阵的建立
  - 根据优先关系的定义
    - $a \lessdot b \Leftrightarrow A \to \ldots aB \ldots \in P$且$(B \overset{+}{\Rightarrow} b \ldots$或者$B \overset{+}{\Rightarrow} Cb \ldots)$
    - 非终结符B派生出的第一个终结符构成的集合
    - $a \doteq b \Leftrightarrow A \to \ldots Bb \ldots \in P$且$(B \overset{+}{\Rightarrow} \ldots a$或者$B \overset{+}{\Rightarrow} \ldots aC)$
    - 非终结符B派生出的最后一个终结符构成的集合
- 引入FirstVT (A)集、LastVT (A)集
- $A \to \ldots aB \ldots$　　$a \lessdot$ FirstVT (B)
- $A \to \ldots Bb \ldots$　　LastVT (B) $\gtrdot b$

5.2 算符优先分析法

- （1）FirstVT (A)集、LastVT (A)集
- 设G=（V_N,V_T,P,S）为算符优先文法，则定义
  - FirstVT(A)：非终结符A派生出的第一个终结符构成的集合
  - FirstVT(A)={b|$A \overset{+}{\Rightarrow} b\ldots$或者$A \overset{+}{\Rightarrow} Bb\ldots$}
  - LastVT(A)：非终结符A派生出的最后一个终结符构成的集合
  - LastVT(A)={b|$A \overset{+}{\Rightarrow} \ldots b$或者$A \overset{+}{\Rightarrow} \ldots bB$}
  - $b \in V_T$， A、$B \in V_N$

求FirstVT和LastVT

- 初值（扫描所有产生式）
  - if A→a… 或 A→Ba… then a∈FirstVT (A)
  - if A→…a 或 A→ … aB then a∈LastVT (A)
- 迭代（扫描所有产生式）
  - if A→B… then将FirstVT (B)放入 FirstVT (A)
  - if A→…B then 将LastVT (B)放入 LastVT (A)
- $a,b \in V_T$ $A,B \in V_N$

求FirstVT和LastVT

E→E+T|E-T|T

T→T*F|T/F|F F→(E)|i

语法变量	FirstVT	LastVT
E	+ - * / i (	+ - * / i)
T	* / (i	* /) i
F	(i	) i

5.2 算符优先分析法

4、算符优先关系表的构造

- (2)算符优先关系的确定
- 对于所有产生式：$A \rightarrow X_1X_2\ldots X_n$

① X_iX_{i+1}的形式为V_TV_T：　$X_i \doteq X_{i+1}$

② $X_iX_{i+1}X_{i+2}$的形式为$V_TV_NV_T$：　$X_i \doteq X_{i+2}$

③ X_iX_{i+1}的形式为V_TV_N：　$X_i \lessdot a$

$\forall a \in FirstVT(X_{i+1})$，

④ X_iX_{i+1}的形式为V_NV_T：　$a \gtrdot X_{i+1}$

$\forall a \in LastVT(X_i)$，

5.2 算符优先分析法
4、算符优先关系表的构造

- （3）构造算符优先矩阵
  - 结构：二维表
  - 行×列＝(终结符∪{#})×(终结符∪{#})
  - 元素：优先关系

    行：优先关系的左侧

    列：优先关系的右侧

  优先关系不满足：对称性、交换性、传递性

算符优先关系表的构造举例

G[E]:E→E+T| T T→T*F| F F→(E)|i

(1) 求FirstVT、LastVT

语法变量	FirstVT	LastVT
E	+, *, (, i	+, *,), i
T	*, (, i	*,), i
F	(, i	), i

(2)算符优先关系：

-  LastVT(E)>+ LastVT(T)>*
  LastVT(E)>) LastVT(E)>#
   +<FirstVT(T) *< FirstVT(F)
  (< FirstVT(E) #< FirstVT(E)

   : (=)

算符优先关系：

- ⋗ lastVT) E+ T* E) E#
- ⋖ (firstVT) +T *F (E #E
- ⋗ (E)

语法变量	FirstVT	LastVT
E	+, *, (, i	+, *,), i
T	*, (, i	*,), i
F	(, i	), i

(3) 填写算符优先矩阵

	(	i	*	+	)	#
(	⋖	⋖	⋖	⋖	=	
i			⋗	⋗	⋗	⋗
*	⋖	⋖	⋗	⋗	⋗	⋗
+	⋖	⋖	⋗	⋗	⋗	⋗
)			⋗	⋗	⋗	⋗
#	⋖	⋖	⋖	⋖		

5、算符优先分析算法

- (1)识别句柄并归约
  - 句柄 $\#a_1 \ldots a_{i-1} a_i \lessdot a_{i+1} \doteq \ldots \doteq a_{j-1} \doteq a_j \gtrdot a_{j+1} \ldots a_n \#$
  - “大于”是句柄尾，“小于”是句柄头
  - 分析栈遇到“小于”或“等于”入栈，遇到“大于”，找到句柄，并弹出句柄，归约为非终结符，非终结符入栈

5、算符优先分析算法

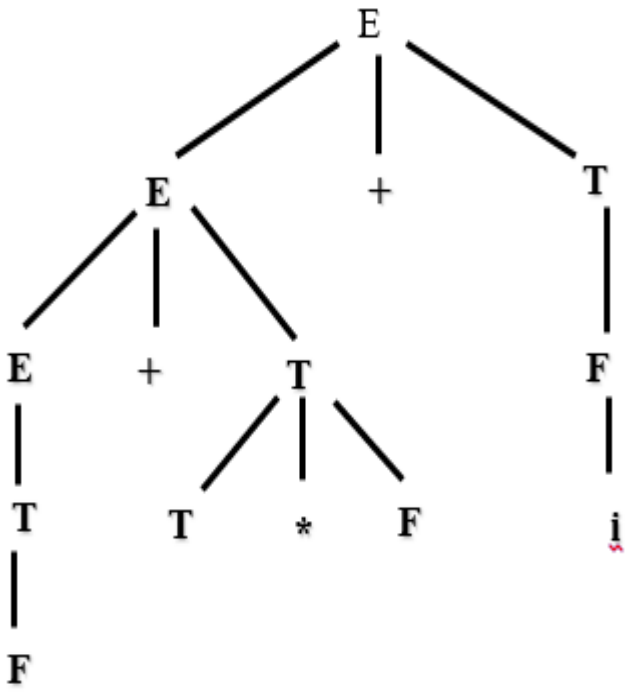
- (2)素短语与最左素短语
- 什么是短语？
- 素短语(Prime Phrase)
  - 句型的至少含一个终结符且不含其它素短语的短语
- 定义算符优先分析过程识别的“句柄”为最左素短语LPP（Leftmost Prime Phase）

5、算符优先分析算法

例 素短语

- G[E]: E→E+T|T
  T→T*F|F
  F→(E)|i

句型: F+T*F+i



短语：F T*F i F+T*F F+T*F+i

素短语：T*F i

句柄：F

T*F为最左素短语，是被归约的对象

5、算符优先分析算法

(3)找最左素短语并归约

设句型的一般形式为

$\#N_1a_1\,N_2a_2\ldots N_na_nN_{n+1}\,\#\ (N_i\in V_N\cup\{\varepsilon\},a_i\in V_T)$

找最左素短语：满足下列条件的最左子串

$$N_ia_i\,N_{i+1}a_{i+1}\ldots N_ja_j\,N_{j+1}$$

其中：

$a_{i-1}\lessdot a_i$

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

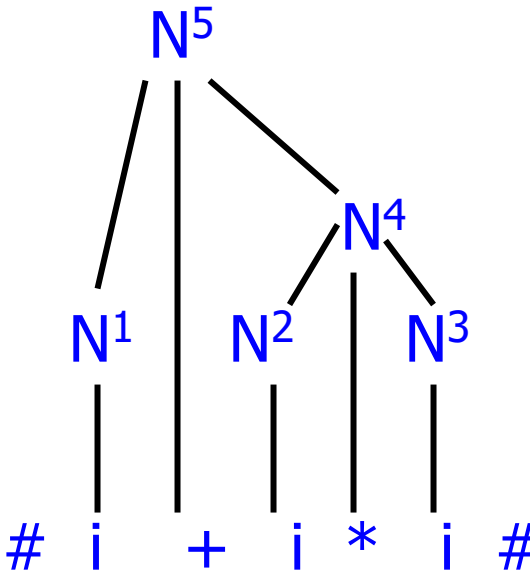
$a_j\gtrdot a_{j+1}$

5、算符优先分析的算法

- 归约最左素短语：$N_i a_i N_{i+1} a_{i+1} \cdots N_j a_j N_{j+1}$
- 最左素短语不一定是句柄（甚至不一定是产生式的右部
- 在P中找形如：$A \rightarrow U_i a_i U_{i+1} a_{i+1} \cdots U_j a_j U_{j+1}$的产生式，其中，$U_i$与$N_i$不一定相同，但每个$a_i$必须相同.若存在这样的产生式，则用其归约，否则分析报错.
- 例 $w=i+i*i$，分析过程见下页

5、算符优先分析的算法

步骤	分析栈	优先关系	输入缓冲区	最左素短语	下一步的动作	输出产生式
0	#	#<i	i+i*i#		移进	
1	#i	i>+	+i*i#	i	归约	**F→i**
2	#F	#<i	+i*i#		移进	
3	#F+	+<i	i*i#		移进	
4	#F+i	i>*	*i#	i	归约	**F→i**
5	#F+F	+<*	*i#		移进	
6	#F+F*	*<i	i#		移进	
7	#F+F*i	i>#	#	i	归约	**F→i**
8	#F+F*F	*>#	#	F*F	归约	**T→T*F**
9	#F+T	+>#	#	F+T	归约	**E→E+T**
10	#E		#		成功	



5.2 算符优先分析法

6、算符优先分析的实现

- 系统组成
  - 移进归约分析器＋优先关系表
- 分析算法
  - 参照输入串、优先关系表，完成一系列归约，生成语法分析树——输出产生式

```
k:=1;
S[k]:='#';
REPEAT
 把下一个输入符号读进a中;
    IF S[k]∈V_T THEN  j:=k  ELSE  j=k-1
          WHILE S[j]> a DO
          BEGIN
            REPEAT
             Q:=S[j];
             IF S[j-1]∈V_T THEN j:=j-1 ELSE
          UNTIL S[j] < Q;
            把S[j+1]...S[k]归约为某个N;
            k:=j+1;
            S[k]:=N
       END OF WHILE;
      IF S[j] < a OR S[j] =  a THEN
         BEGIN k:=k+1;S[k]:=a END
      ELSE ERROR /*调用出错诊察程序*/
  UNTIL a='#'
```

自左至右，终结符对终结符，非终结
对非终结符，而且对应的终结符相同

$$N \rightarrow X_1 \quad X_2 \quad \cdots \quad X_{k-j}$$

$$S[j+1] \quad S[j+2] \quad \cdots \quad S[k$$

5.2算符优先分析

- 算符优先分析法特点：
  - 优点：简单，快速
  - 缺点：可能错误接受非法句子，能力有限.
- 算符优先分析法是一种广为应用、行之有效的方法。
  - 用于分析各类表达式
  - ALGOL 60

5.3 算符优先函数

- 引入函数f和g：f(a)终结符号a的栈内优先数；g(a)终结符号a的栈外优先数，满足：

  如果 $a \lessdot b$，$f(a)<g(b)$

  如果 $a \doteq b$，$f(a)=g(b)$

  如果 $a \gtrdot b$，$f(a)>g(b)$

- 优点：节省存储空间（$n^2 \rightarrow 2n$）便于执行比较运算
- 损失
  - 错误检测能力降低
    - 如：id $\gtrdot$ id不存在，但f(id)>g(id)可比较

有向图法构造优先函数

- 有向图的画法：

  1. $\forall a \in V_T \cup \{\#\}$，建立两个结点 f_a 和 g_a；
  2. 若 $a \doteq b$，则从 f_a 至 g_b 画一条弧，
     从 g_b 至 f_a 再画一条弧；
  3. 若 $a \gtrdot b$，则从 f_a 至 g_b 画一条弧；
  4. 若 $a \lessdot b$，则从 g_b 至 f_a 画一条弧.

- 函数值的计算：对每个结点都赋予一个数，此数等于从该结点出发所能到达的结点数(包括出发点自身)。
- 检查所构造出来的函数f和g是否与原来的关系相容。若没有矛盾，则f和g就是要求的优先函数，若有矛盾，则不存在优先函数。

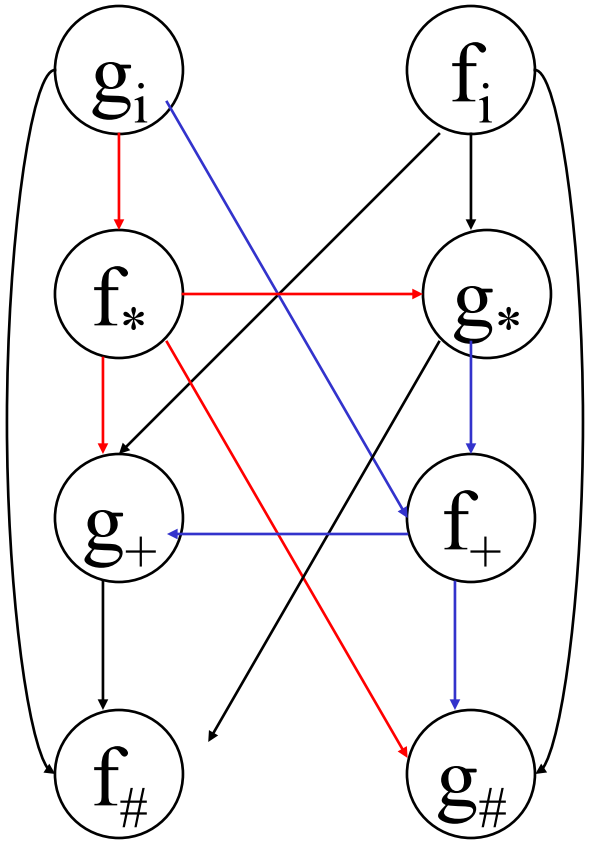
如果 $a \lessdot b$，$f(a)<g(b)$

如果 $a \doteq b$，$f(a)=g(b)$

如果 $a \gtrdot b$，$f(a)>g(b)$

有向图法构造优先函数

	+	*	i	#
+	⋗	⋖	⋖	⋗
*	⋗	⋗	⋖	⋗
i	⋗	⋗		⋗
#	⋖	⋖	⋖	



优先函数:

	+	*	i	#
f	4	6	6	1
g	2	5	7	1

优先函数与原优先矩阵相容，所以是相应矩阵的优先函数

优先函数应用

例： 分析i+i*i

步骤	符号栈	优先函数	余留输入串	最左素短语	所选产生式
1	#	f(#)<g(i)	i+i*i#		
2	#i	f(i)>g(+)	+i*i#	i	F→i
3	#F	f(#)<g(+)	+i*i#		
4	#F+	f(+)<g(i)	i*i#		
5	#F+i	f(i)>g(*)	*i#	i	F→i
6	#F+F	f(+)<g(*)	*i#		
7	#F+F*	f(*)<g(i)	i#		
8	#F+F*i	f(i)>g(#)	#	i	F→i
9	#F+F*F	f(#)>g(#)	#	F*F	T→T*F
10	#F+T	f(#)>g(#)	#	F+T	E→E+T
11	#E		#	accept	

优先分析小结

- 基本框架
  - 移进归约分析
  - 核心：在句型中寻找可规约串进行归约
  - 分析器具有基本相同的逻辑结构
- 控制算法：
  - 基于算符优先关系表

优先分析小结

- 1、自底向上的语法分析基本原理
- 2、算法优先分析的基本原理
- 3、最左素短语
- 4、算法优先矩阵的构造
- 5、算法优先分析的实现