第五章　优先分析法

廖力
xobjects@seu.edu.cn
3793235

引言

一、语法分析

- 推导：
  - 自上而下的语法分析过程
  - 预测分析程序，递归下降分析法（最左推导）
  - 注：要求文法是LL(1)文法
- 归约：
  - 自下而上的语法分析过程
  - 简单优先分析法，算符优先分析法，LR分析法

引言

二、自下而上的语法分析过程

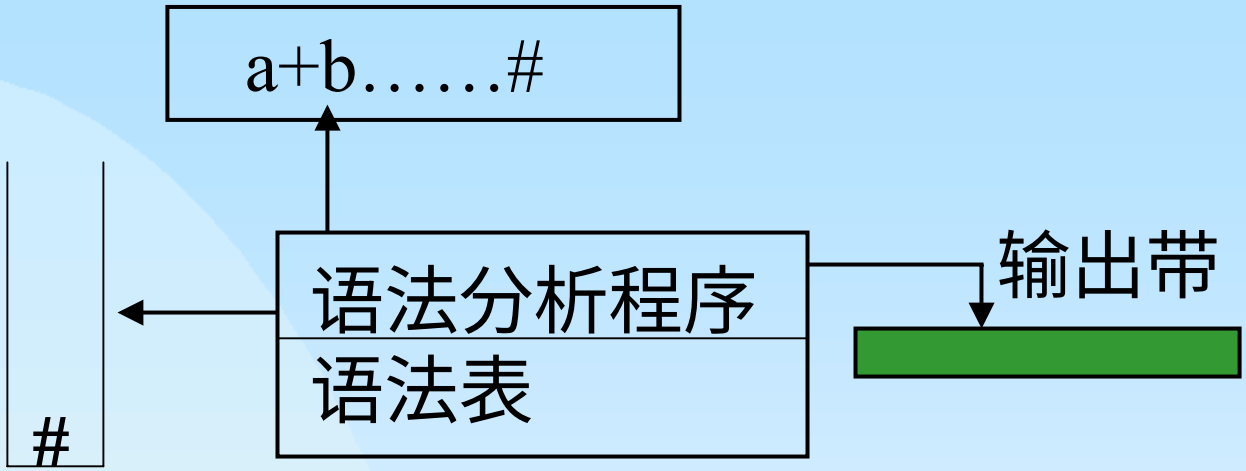
1、自下而上的语法分析过程思想

- 从输入串开始，朝着文法的开始符号进行最左归约，直到到达文法的开始符号为止。主要是进行移进或归约操作，采用最左归约。

2、基本过程

- 从输入串开始，朝着文法的开始符号进行归约，直到到达文法的开始符号为止的过程。
- 注意：输入串在这里是指从词法分析器送来的单词符号组成的二元式的有限序列。

3、自下而上分析的PDA



- 工作方式：“移进—归约”方式
- 即：自左至右把输入串的符号一个一个移进栈，在移进过程中不断查看栈顶符号串，一旦形成某个句型的句柄时，就将此句柄用相应的产生式左部替换（归约），若再形成句柄，就继续替换，直到栈顶不再形成句柄为止。然后继续移进符号，重复上面的过程直到栈顶只剩下文法的开始符号，输入串读完为止，这样就认为识别了一个句子。

引言
二、自下而上的语法分析过程

注：1）初态时栈内仅有栈底符“#”，读头指在最左边的单词符号上。

2）语法分析程序执行的动作：

a）移进 读入一个单词并压入栈内，读头后移；

b）归约 检查栈顶若干个符号能否进行归约，若能，就以产生式左部替代该符号串，同时输出产生式编号；

c）识别成功 移进－归约的结局是栈内只剩下栈底符号和文法开始符号，读头也指向语句的结束符；

d）识别失败

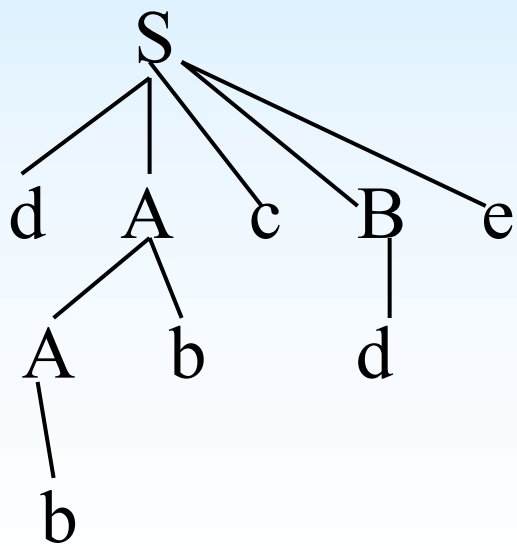
引言

二、自下而上的语法分析过程

- 例如：有文法如下
- （1）S →aAcBe
- （2）A →b
- （3）A →Ab
- （4）B →d

问：语句abbcde是不是该文法的合法语句？

步骤	栈	输入串	输出带	动作
0	#	abbcde#		
1	#a	bbcde#		移进
2	#ab	bcde#		移进
3	#aA	bcde#	2	归约
4	#aAb	cde#		移进
5	#aA	cde#	2,3	归约
6	#aAc	de#		移进
7	#aAcd	e#		移进
8	#aAcB	e#	2,3,4	归约
9	#aAcBe	#		移进
10	#S	#	2,3,4,1	归约
11	识别成功			



引言

三、常用的自下而上语法分析

1、优先分析（Precedence Parser)

- 简单优先分析法
- 算符优先分析法
  - 特别适合于表达式的分析，基本思想是按算符的优先关系和结合规则进行语法分析。

2、LR分析

5.1 简单优先分析法

一、基本思想

1、对句型中相邻的文法符号规定优先关系，以寻找句型中的句柄。

2、规定句柄内各相邻符号之间具有相同的优先级。

3、规定句柄两端符号优先级要比位于句柄之外而又和句柄相邻的符号的优先级高，以先归约句柄。

4、对于文法中所有符号，只要它们可能在某个句型中相邻，就要为它们规定相应的优先关系，若某两个符号永远不可能相邻，则它们之间就无关系。

5.1 简单优先分析法

二、简单优先文法

1、定义：一个文法G，如果它不含ε产生式，也不含任何右部相同的不同产生式，并且它的任何符号对(X,Y)，或者没有关系，或者存在优先级相同或低于、高于等关系之一，则这是一个简单优先文法。

2、优先级别定义：

- $X \doteq Y$　当且仅当G中含有形如$P \rightarrow \ldots XY \ldots$
- $X \lessdot Y$　当且仅当G中含有形如$P \rightarrow \ldots XQ \ldots$的产生式，其中$Q \overset{+}{\Rightarrow} Y \ldots$
- $X \gtrdot Y$　当且仅当Y为文法G的终结符，G中有形如$P \rightarrow \ldots QR \ldots$，且$Q \overset{+}{\Rightarrow} \ldots X$，$Y \in first(R)$
- 对任何X，若文法开始符号$S \overset{+}{\Rightarrow} X \ldots$,则$\# \lessdot X$;若有$S \overset{+}{\Rightarrow} \ldots X$，则$X \gtrdot \#$。

5.1 简单优先分析法

三、简单优先分析的基本思想

1、简单优先矩阵：根据优先关系的定义，将简单优先文法中各文法符号之间的这种关系用一个矩阵表示，称作简单优先矩阵。

注：简单优先矩阵的构造可通过Warshall算法实现。

2、简单优先分析法思想：

- PDA读入一个单词后，比较栈顶符号和该单词的优先级，若栈顶符号优先级低于该单词，继续读入；若栈顶符号优先级高于或等于读入符号，则找句柄进行归约，找不到句柄就继续读入。直到最后栈内只剩下开始符号，输入串读到“#”为止。此时识别正确。

5.1 简单优先分析法

四、简单优先分析法的优缺点

优点：技术简单

缺点：适用范围小，分析表尺寸太大。

5.2 算符优先分析法

一、基本思想

1、自下而上归约

2、规定算符(更一般地说，指终结符)的优先级及结合规则，以使得分析过程唯一。

3、比较相邻两个算符而决定动作。

注：1)这里的关键是对所有算符定义某种优先关系。

2)算符优先分析法是仿效四则运算的计算过程而构造的一种语法分析方法。

3)算符优先分析法的特点：简单直观，特别方便于表达式分析，易于手工实现，但未必按照句柄归约。

5.2 算符优先分析法

4、实例：表达式文法：

$$E \rightarrow E+E|E-E|E*E|E/E|(E)|i$$

对这个二义文法可能会有好几个规范推导和归约，真正运算时也有几种不同结果，但若按算符优先顺序和结合规则的规定进行归约，句子的归约过程就是唯一的，运算结果也唯一。

如： i+i-i*(i+i)归约过程如下：

1) i+i-i*(i+i)　　设算数级别最高，最先归约；

2) E +i-i*(i+i)

3) E+E-i*(i+i)　　＋，－同级，先归约左边“＋”

4) E-i*(i+i)

5) E-E*(i+i)　　－，×不同级，先归约右边“×”

6) E-E*(E+i)

7) E-E*(E+E)　　先算括号内，再算括号外

8) E-E*(E)

9) E-E*E　　先归约“×”，再归约“－”

10)E-E

11)E

5.2 算符优先分析法

二、确定运算符的优先级

2、对于任何两个可能相继出现的终结符a,b，若具有以下形式：“…ab…”， “…aQb…”,其中Q是某非终结符，则a、b之间的关系为：

1) $a \lessdot b$　　a的优先级低于b

2) $a \doteq b$　　a的优先级等于b

3) $a \gtrdot b$　　a的优先级高于b

4) 若ab在任何情况下都不可能相继出现，则ab无关系。

5.2　算符优先分析法

二、确定运算符的优先级

注：1)算符优先分析法的关键是比较两个相继出现的终结符的优先级而决定应采取的动作。

2)要完成运算符间优先级的比较，可以先定义各种可能相继出现的运算符的优先级，并表示为矩阵的形式，使得能够在分析中通过查询矩阵元素而得到算符之间的优先关系。

5.2　算符优先分析法

左符＼右符	+	*	(	)	i	#
+	$\cdot>$	$\cdot<$	$\cdot<$	$\cdot>$	$\cdot<$	$\cdot>$
*	$\cdot>$	$\cdot>$	$\cdot<$	$\cdot>$	$\cdot<$	$\cdot>$
(	$\cdot<$	$\cdot<$	$\cdot<$	$\doteq$	$\cdot<$	
)	$\cdot>$	$\cdot>$		$\cdot>$		$\cdot>$
i	$\cdot>$	$\cdot>$		$\cdot>$		$\cdot>$
#	$\cdot<$	$\cdot<$	$\cdot<$		$\cdot<$	

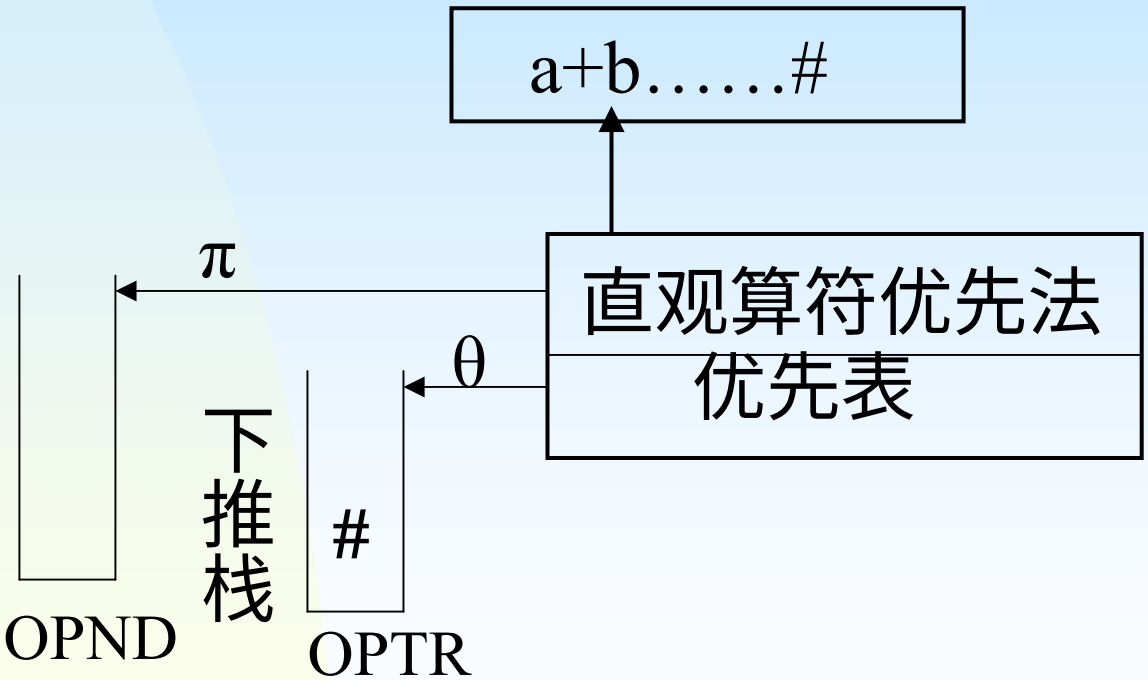
5.2 算符优先分析法

- 注：1）在此表中，＋包括－，*包括/，“#”是一个特殊符号，用于语句的开始符号和结束符号。
- 2）这张表满足通常数学上的习惯约定。值得注意的是：
  - a、运算量i的优先级高于算符；
  - b、语句开始和结束符号“#”与终结符a相继出现时，应该有# ⋖a和a⋗ #,以此来保证语句内先归约。
- 3）(=)表示括号是成对归约的。
- 4）优先关系和代数中的大于小于关系不同，a ⋖ b并不意味着b⋗ a，终结符所处的左右位置很重要。

5.2　算符优先分析法

三、直观算符分析法

1、PDA



5.2　算符优先分析法

三、直观算符分析法

1、PDA

注：1）直观算符分析法使用两个工作栈：一个算符栈（OPTR）存放运算符和括号；一个算量栈（OPND）用于存放操作数和运算结果；OPTR栈的栈顶符号用θ表示,OPND栈的栈顶符号用π表示

2）初态时，OPND为空，OPTR中只有“#”。

2、直观算符分析法算法

```
OPND=“”;
OPTR=“#”;
flag=true;
advance;      /*读入一个单词*/
while flag {
      if  θ=“#” and  SYM=“#”   then flag=false    /*成功*/
        else if θ=“(“ and SYM=“)”  then     /*匹配括号对*/
           {OPTR栈顶出栈；advance;}
        else if SYM是算量  then    /*移进算量*/
           {SYM入OPND栈；advance;}
        else if θ＜SYM  then    /*移进算符*/
           {SYM入OPTR栈；advance;}
```

else if θ ∈SYM then /*归约*/

{OPND栈顶两个元素π_1、 π_2弹出；

弹出OPTR栈顶元素θ；

将$\pi_1\theta\pi_2$ 的结果入OPND栈；

}

else ERROR;

}}

5.2 算符优先分析法

三、直观算符分析法

3、算符优先分析法的优缺点

- 优点：1）简单明了，易于手工实现，适于分析各种算术表达式。
  - 2）使用算符优先分析法可以很方便的把表达式译成目标指令，只要在归约时把计算$\pi_1\theta\pi_2$值改为生成相应指令（θ，π_1，π_2，T）即可。
- 缺点：1）算法采用两个栈，有时会把错误句子当成合法句子；而且，它也无法指出输入串出错位置；
  - 2）对于含有单目正负号的算数表达式不好处理，因为负号的优先级高于加减法，低于乘除法，但负号的形式与减号相同，不容易识别。

5.2　算符优先分析法

四、算符优先文法及优先表的构造

1、算符文法的定义：

- 给定上下文无关文法G，若G中所有产生式右部都不包含两个相继的非终结符，则G为算符文法。
- 注：算符文法保证了两个运算符之间只有一个操作数。

2、算符优先文法定义：

- 设G是一个不包含空串产生式的算符文法，并设a,b $\in V_T$; P,Q,R $\in V_N$,定义关系：

5.2　算符优先分析法
四、算符优先文法及优先表的构造

2、算符优先文法定义：

- $a \doteq b$　当且仅当G中含有形如$P \to \ldots ab \ldots$产生式，或者$P \to \ldots aQb \ldots$产生式；
- $a \lessdot b$　当且仅当G中含有形如$P \to \ldots aR \ldots$的产生式，其中$R \overset{+}{\to} b \ldots$, 或$R \overset{+}{\to} Qb \ldots$;
- $a \gtrdot b$　当且仅当G中有形如$P \to \ldots Rb \ldots$产生式，其中$R \overset{+}{\to} \ldots a$，或$R \overset{+}{\to} \ldots aQ$.
- 若G中任何终结符序偶（a,b）至多满足上述关系之一，则称G为算符优先文法（OPG）。

5.2　算符优先分析法

四、算符优先文法及优先表的构造

3、算符文法和算符优先文法定义的意义

- 这两个定义相当于对文法的句型和可归约的短语做了如下规定：
  - 设$A_1A_2\ldots A_{i-1}A_iA_{i+1}\ldots A_n$是文法G的一个句型，
  - 1、若$A_i\in V_N$，则A_{i-1}、$A_{i+1}\in V_T$，即不许出现相继两个非终结符；
  - 2、若$B_1B_2\ldots B_m$是当前可归约短语，并可归约为A_i，则：

a) $B_1B_2\ldots B_m$中不能有相继两个非终结符且相邻的终结符优先级全相等；

b)对于$B_1B_2\ldots B_m$中首终结符b有$A_{i-1} \lessdot b$

c)对于$B_1B_2\ldots B_m$中尾终结符b有$b \gtrdot A_{i+1}$.

注：实际上，可归约短语是某产生式右部符号串，所以通过检查G的每个产生式的每个候选式，可以查找出任意优先级相同的终结符序偶。要找出所有满足关系 $\lessdot$ 和 $\gtrdot$ 的终结符序偶，就要找出各非终结符P的首终结符集和尾终结符集。

5.2 算符优先分析法

四、算符优先文法及优先表的构造

4、求各非终结符P的首终结符集和尾终结符集

1）定义：

- 首终结符集FIRSTVT（P）＝
  - $\{a \mid P \overset{+}{\rightarrow} a\ldots$或$P \overset{+}{\rightarrow} Qa\ldots，\ a \in V_T;\ P,Q \in V_N\}$
- 尾终结符集LASTVT（P）＝
  - $\{a \mid P \overset{+}{\rightarrow} \ldots a$或$P \overset{+}{\rightarrow} \ldots aQ，\ a \in V_T;\ P,Q \in V_N\}$
- 注：有了这两个集合后，就可以通过检查每个产生式的每个候选式，确定满足关系 $\lessdot$ 和 $\gtrdot$ 的所有终结符序偶。

5.2 算符优先分析法

2）优先关系确定方法：

- 假定产生式右部有形如：…aP…的串，则
  - 对于任何b ∈FIRSTVT(P),有a⋖ b;
  - 假定产生式右部有形如：…Pb…的串，则
  - 对于任何a ∈LASTVT(P),有a⋗ b;
- 例:设文法G的产生式为：

  S →aAcBe　　A →Ab|b　　B →d

  计算每个非终结符的FIRSTVT与LASTVT及所有终结符之间的关系。

- 解：FIRSTVT(S)={a}　LASTVT(S)={e}
- 　　FIRSTVT(A)={b}　LASTVT(A)={b}
- 　　FIRSTVT(B)={d}　LASTVT(B)={d}

左＼右	a	b	c	d	e
a		$\lessdot$	$\doteq$		
b		$\gtrdot$	$\gtrdot$		
c				$\lessdot$	$\doteq$
d					$\gtrdot$
e					

5.2　算符优先分析法

四、算符优先文法及优先表的构造

5、构造算法

1）构造集合FIRSTVT(P)的算法

a)方法1：

- 根据FIRSTVT(P)的定义，按下面的规则来构造：
- (1)若有产生式P →a…或P →Qa…，则a ∈FIRSTVT(P)
- (2)若a ∈FIRSTVT(Q)，且有产生式P →Q…，则 a ∈FIRSTVT(P)。

5.2　算符优先分析法

四、算符优先文法及优先表的构造

5、构造算法

1）构造集合FIRSTVT(P)的算法

a)方法1：

- 注：（1）规则1是求P FIRSTONE a的关系，即当且仅当有产生式P →a…或P →Qa… ；

（2）规则2是求P FIRST* Q，P、Q∈V， 这是求解自反传递闭包问题，用Warshall算法很容易求得。因此，FIRSTVT(P)=(FIRST*)(FIRSTONE)。

5.2 算符优先分析法

四、算符优先文法及优先表的构造

5、构造算法

1）构造集合FIRSTVT(P)的算法

b)方法2：

- 基本思想：在此算法中使用了两个数据结构
  - 一个是二维布尔矩阵F，行标为非终结符P，列标为终结符a；F[P,a]＝true，当且仅当a ∈FIRSTVT(P)；
  - 另一个是堆栈STACK，栈中动态存放曾在F[P,a]中为真的序偶(P,a)。
- 算法如下：

（1）初始化：将布尔矩阵F中各元素置为假；栈清空；

（2）按规则1，查看产生式，对于形如 P →a…或P →Qa..的产生式，置相应的F[P,a]为真，并将序偶(P,a)入栈；

（3）按规则2，对栈施加如下操作：

- 弹出栈顶序偶并记为(Q,a)，查看所有产生式，看有无形如 P →Q…的产生式，若有，且a ∉FIRSTVT(P)（即F[P,a]为假），则将F[P,a]为置为真，并把(P,a)入栈；

（4）重复步骤3，直到栈空为止。

那么，在F[P,a]中，凡是“真”的元素即属于P的首终结符集。

5.2 算符优先分析法

四、算符优先文法及优先表的构造

5、构造算法

2）构造算符优先表算法

- FOR 每条产生式$P \rightarrow X_1X_2\ldots X_n$ DO
- {FOR (i=1,i<=n-1,i++)
- {if X_i和X_{i+1}均为终结符 then 置$X_i \doteq X_{i+1}$
- if i<=n-2 and X_i和X_{i+2}均为终结符 and X_{i+1}为非终结符 then 置$X_i \doteq X_{i+1}$
- if X_i为终结符而X_{i+1}为非终结符 then

- for FIRSTVT(X_{i+1})中的每个a DO
- {置 $X_i \lessdot a$}
- if X_i为非终结符而X_{i+1}为终结符 then
- for LASTVT(X_i)中的每个a DO
- {置 $a \gtrdot X_i$}
- }

注：如果文法G按此算法构造出的优先表没有重定义项，则文法G是个算符优先文法。

例：构造下面文法的算符优先表。

$S \rightarrow$ if E_b then E else E　　$E \rightarrow E+T|T$

$T \rightarrow T*F|F$　　$F \rightarrow i$　　$E_b \rightarrow b$

解：1）求各非终结符的首终结符集和尾终结符集。为了考虑语句的开始和结束符号“#”，对文法拓广，加一个产生式$S' \rightarrow \#S\#$

- FIRSTVT(S)={if}　　LASTVT(S)={else,+,*,i}
- FIRSTVT(E)={+,*,i}　LASTVT(E)={+,*,i}
- FIRSTVT(T)={*,i}　　LASTVT(T)={*,i}
- FIRSTVT(F)={i}　　LASTVT(F)={i}
- FIRSTVT(E_b)={b}　　LASTVT(E_b)={b}

2)填写算符优先表

左＼右	if	then	else	+	*	i	b	#
if		$\doteq$					$\lessdot$	
then			$\doteq$	$\lessdot$	$\lessdot$	$\lessdot$		
else				$\lessdot$	$\lessdot$	$\lessdot$		$\gtrdot$
+			$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$		$\gtrdot$
*			$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$		$\gtrdot$
i			$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$
b		$\gtrdot$						
#	$\lessdot$							

5.2　算符优先分析法

五、算符优先分析的若干问题

1、优先表构造算法的讨论

- 构造优先表的算法仅仅反映文法符号间关系，并不反映附加条件，对二义性文法构造算符优先表，可能会出现多重定义项。

2、非规范分析

- 1）规范归约
  - 严格按照句柄进行归约，终结符和非终结符一起考虑，只要栈顶形成句柄，不管句柄内是否包含终结符都要进行归约。

5.2 算符优先分析法

例如：考虑非二义的表达式文法G(E)：

– E →E+T|T　　T →T*F|F　　F →(E)| i

– 识别语句 i+i*i的过程

- （1）规范归约

– i+i*i#

– 1.F+i*i#　　6. E+T*F#

– 2.T+i*i#　　7. E+T#

– 3.E+i*i#　　8. E#

– 4.E+F*i#

– 5.E+T*i#

5.2　算符优先分析法

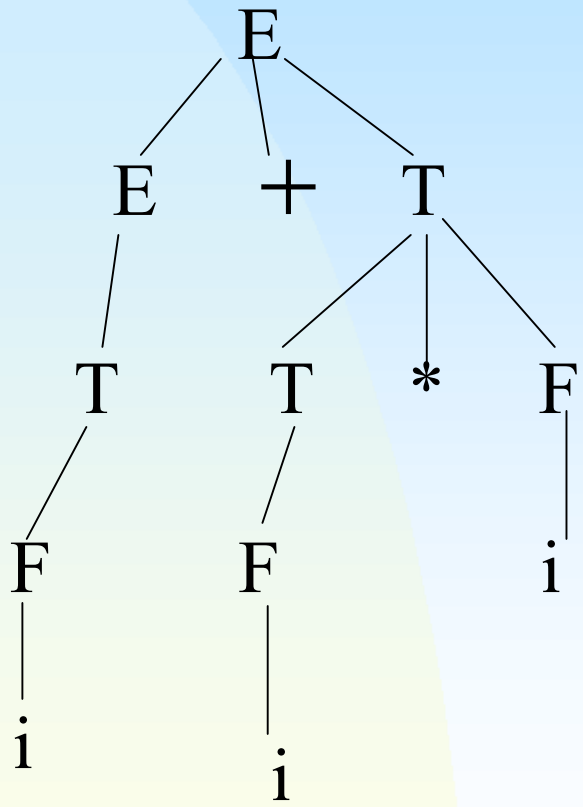
五、算符优先分析的若干问题

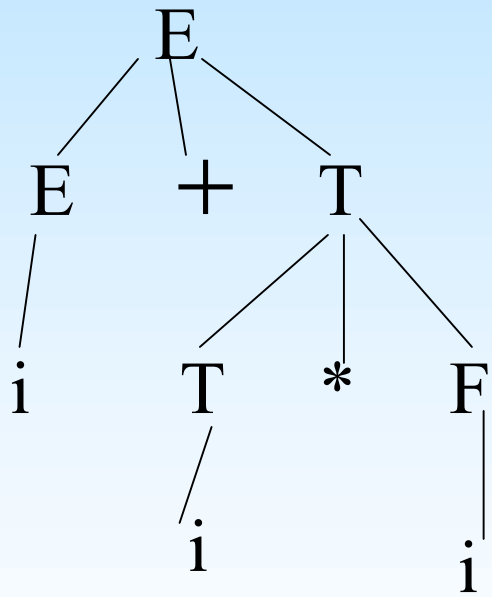
2、非规范分析

2)算符优先分析

- 在算符优先分析中，仅研究终结符之间的优先关系，而不考虑非终结符之间的优先关系，但句柄是由终结符和非终结符一起构成的，所以算符优先分析相对来说是非规范的分析。
- 对上题分析过程：
  - i+i*i#
  - 1.E+i*i#　　4. E+T#
  - 2.E+T*i#　　5. E#
  - 3.E+T*F#

- 两种分析的语法树对比





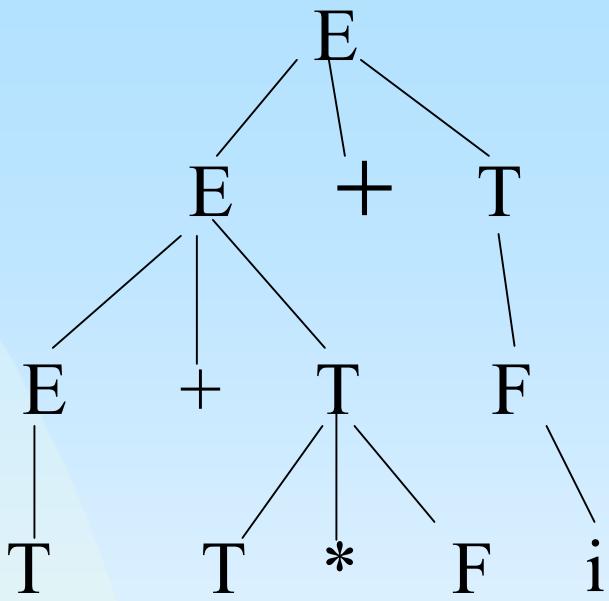
5.2　算符优先分析法

五、算符优先分析的若干问题

2、非规范分析

2)算符优先分析

- 注:在算符优先分析中，可归约的短语不再称为句柄，而称为最左素短语。素短语是指这样一个短语，至少含有一个终结符，且除它自身外不再包含其他素短语，最左边的素短语称为最左素短语。
- 例如：考虑非二义的表达式文法G(E)：
  - $E \rightarrow E+T|T \qquad T \rightarrow T*F|F \qquad F \rightarrow (E)|\ i$
  - 句型#T+T*F+i#的素短语是什么？



- 由定义可知，素短语有：T*F,i
- 最左素短语：T*F

5.2　算符优先分析法

五、算符优先分析的若干问题

3、通用算符优先分析

1)最左素短语的判断

- 假定文法的句型的一般形式为:#$N_1a_1\ N_2a_2\ldots\ N_na_n\ N_{n+1}$#

  其中a_i是终结符，N_i是可有可无的非终结符,设最左素短语为$a_j\ldots\ N_ia_i\ N_{i+1}$,则必有：

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

  则，$a_jN_i\ldots\ N_ia_i\ N_{i+1}$一定能归约为某非终结符。

- 注意：在程序设计语言中会经常看到这种素短语。当栈顶形成某素短语时就可以进行归约。算符优先分析中也要考虑到这种归约。

5.2　算符优先分析法

五、算符优先分析的若干问题

3、通用算符优先分析

2)通用算符优先分析算法

{k=1;s[k]='#'; /* S为下推栈，这里称符号栈*/

repeat

将一个输入符号读入SYM中；

IF S[k]$\in V_T$ THEN j=k ELSE j=k-1;

While s[j] $\doteq$ SYM /*素短语归约可能做若干次*/

{ repeat q=s[j]; /*找素短语的头*/

if s[j-1]$\in V_T$ then j=j-1 else j=j-2

until s[j] $\lessdot$ Q;

```
    将s[j+1]…s[k]归约为某个N；/*若找不到相应的产生式归约，则出错*/
      k=j+1;
      s[k]=N }   /* end of while*/
    if s[j]<· SYM or s[j] ≐ SYM then
        {k=k+1;s[k]=SYM} /*移进*/
    else ERROR /*调出错处理程序*/
until SYM='#' /*识别成功*/
}
```

5.2　算符优先分析法

五、算符优先分析的若干问题

3、通用算符优先分析

2)通用算符优先分析算法

注：（1）算法结束时，若栈内只有“#”和某非终结符，读头下为“#”，则表示分析成功，否则输入串有错。

（2）在进行最左素短语归约时，只要能找出产生式，其右部的终结符与栈顶的若干终结符有一一对应的关系，当名称相同，位置也相同时即可进行归约，由于最左素短语不考虑非终结符，所以归约成什么符号无关紧要。

（3）通用算符优先分析不考虑非终结符，终结符和非终结符放在同一个栈中。

5.2 算符优先分析法

例如：根据下面文法及其算符优先表，按通用算符优先分析的算法分析语句#if b then i else i#。

$S \rightarrow$ if E_b then E else E　　$E \rightarrow E+T|T$

$T \rightarrow T*F|F$　　$F \rightarrow i$　　$E_b \rightarrow b$

左＼右	if	then	else	+	*	i	b	#
if		$\doteq$		$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	
then			$\doteq$	$\lessdot$	$\lessdot$	$\lessdot$		
else				$\lessdot$	$\lessdot$	$\lessdot$		$\gtrdot$
+			$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$		$\gtrdot$
*			$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$		$\gtrdot$
i			$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$
b		$\gtrdot$						
#	$\lessdot$							

步骤	下推栈	关系*	输入串	动作
0	#	⋖	if b then i else i #	
1	#if	⋖	b then i else i #	移进
2	# if b	⋗	then i else i #	移进
3	#if N	≐	then i else i #	对b归约
4	#if N then	⋖	i else i #	移进
5	#if N then i	⋗	else i #	移进
6	#if N then N	≐	else i #	对i归约
7	#if N then N else	⋖	i #	移进
8	#if N then N else i	⋗	#	移进
9	#if N then N else N	⋗	#	对i归约
10	#N		#	归约
	成功			

5.2 算符优先分析法

五、算符优先分析的若干问题

4、算符优先分析的优缺点

- A、算符优先分析比规范归约要快得多，因为它跳过了许多单非终结符的归约。但是，由于忽略了非终结符在归约中的作用，它可能会把错误的输入串误认为是句子。
- B、算符优先文法适用范围比简单优先文法大得多，许多程序设计语言都可以用它来分析。算符优先分析优先表构造简单，甚至可以用手工构造。
- C、缺点在于有些文法不满足算符优先文法，必须先改写，有些甚至无法改写。若终结符数目多，优先表可能会占有太多空间。

5.3 优先函数

一、优先函数的作用：减小优先表的空间占有量。

二、实现方法

- 把每个终结符θ与一对整数f(θ),g(θ)联系在一起。 f(θ)是θ在栈内时的优先数， g(θ)是θ还未进栈时的优先数，叫比较优先数。
- f(θ),g(θ)的值应该满足以下关系：
  - 若 $\theta_1 \lessdot \theta_2$ 则$f(\theta_1)<g(\theta_2)$
  - 若 $\theta_1 \doteq \theta_2$ 则$f(\theta_1)=g(\theta_2)$
  - 若 $\theta_1 \gtrdot \theta_2$ 则$f(\theta_1)>g(\theta_2)$
- 这样可以把优先表转换为优先函数表，所需的存储空间也由n*n个单元减少到2*n个单元。同时把比较运算转化成了数学的比较大小，方便了语法分析过程。

5.3 优先函数

三、优先表向优先函数的转化

1、算法1：逐次加1法

- 1）对所有终结符a（包括#），令f(a)＝g(a)=c，c为一任意常数。
- 2）对所有终结符：
  - 若a ⋗b 而f(a)<=g(b),则取f(a)=g(b)＋1；
  - 若a ⋖b 而f(a)>=g(b),则取g(b) = f(a)+1；
  - 若a ⋍b 而f(a)<>g(b),则取f(a)=g(b)=max(f(a),g(b))；
- 3）重复步骤2）直到f(a)，g(b)不再改变为止。若存在f(a)或g(b)值>=2n+c而步骤2）还未结束，则优先函数不存在。

例：优先表如下，构造优先函数表

左符＼右符	+	*	(	)	i	#
+	$\gtrdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$
*	$\gtrdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	$\lessdot$	
)	$\gtrdot$	$\gtrdot$		$\gtrdot$		$\gtrdot$
i	$\gtrdot$	$\gtrdot$		$\gtrdot$		$\gtrdot$
#	$\lessdot$	$\lessdot$	$\lessdot$		$\lessdot$	

- 步骤1：置初值，设c=1

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

- 步骤2：对第一步结果进行迭代，执行算法第二步

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

- 步骤3：对第2步结果进行迭代，执行算法第二步

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

- 步骤4：对第3步结果进行迭代，执行算法第二步

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

- 步骤4与步骤3的结果相同，迭代收敛，步骤5的结果即为优先函数。

5.3 优先函数

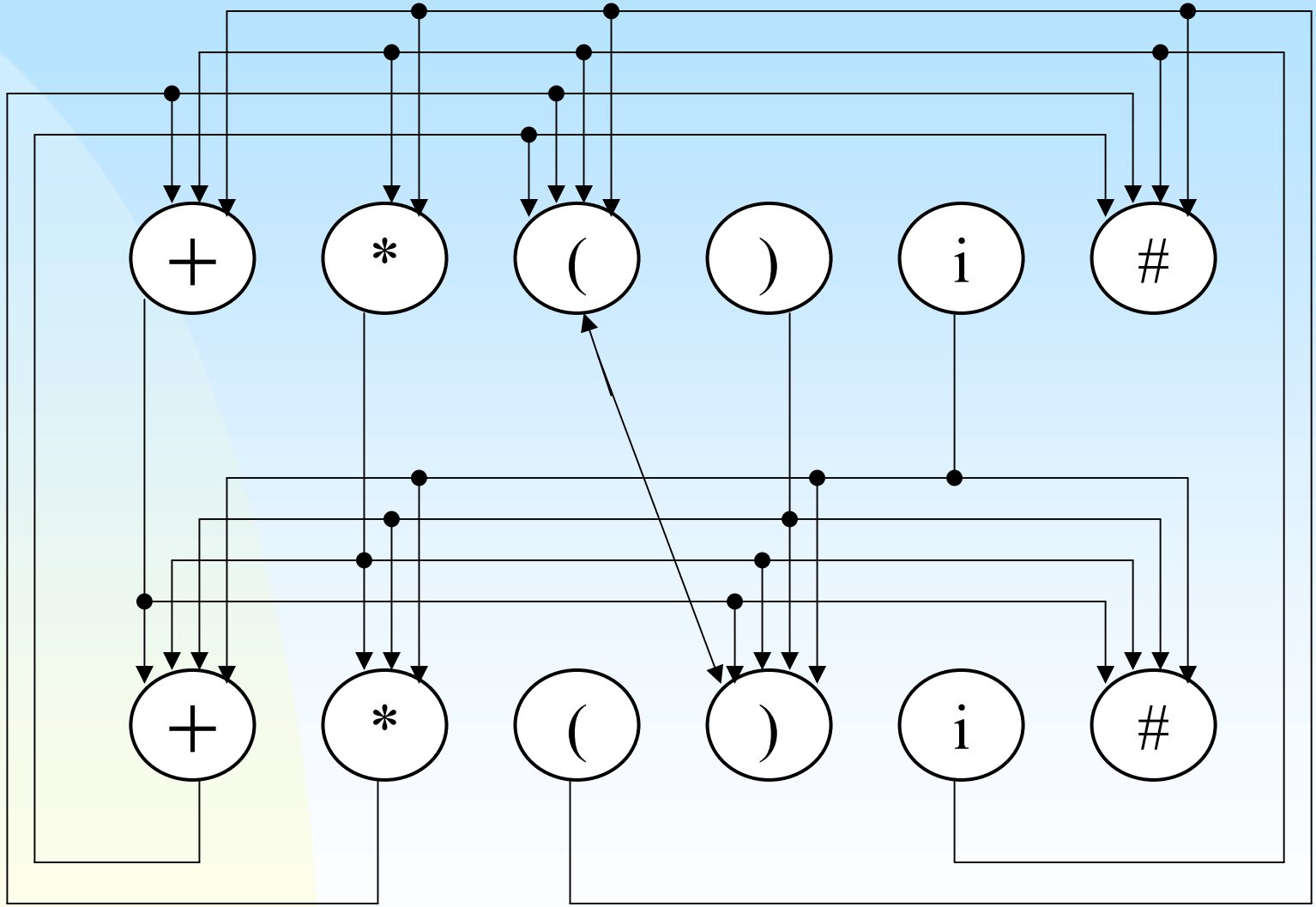
三、优先表向优先函数的转化

2、算法2：Bell有向图

- 1）对每个终结符a(包括#)，令其对应两个结点fa和ga，画一张一所有fa和ga为结点的有向图，如果，a ⋗ b或 a ≐ b,就从fa画一条弧指向gb,反之，若a ⋖ b或 a ≐ b,就从gb 画一条弧指向fa；
- 2）令f(a)等于结点fa可达的结点数（包括到达自己），令g(a)等于结点ga可达的结点数（包括到达自己）；
- 3）检查构造出来的f(a)和g(a),若与优先表符合则优先函数存在，否则不存在。

例：优先表如下，构造优先函数

左符＼右符	+	*	(	)	i	#
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
#	<	<	<		<	


+
*
(
)
i
#
+
*
(
)
i
#

	+	*	(	)	i	#
f	6	8	2	8	8	1
g	4	7	9	2	9	1

- 注：用两种算法得到的优先函数表不同，说明了如果存在一对优先函数，就存在无穷多对优先函数。

5.3 优先函数

四、优先表与优先函数的关系

- 1、优先函数并不等价于优先表，在优先表中没有关系的终结符对也存在优先函数。优先表能发现的错误,优先函数不能发现。优先函数的能力弱于优先表。
- 2、有些优先表不存在优先函数，例如

	a	b
a	≐	⋗
b	≐	≐

- 因为：f(a)=g(a) , f(a)>g(b), f(b)=g(a), f(b)=g(b)
- 导致结果f(a)>g(b)＝f(b)=g(a)=f(a),出现矛盾

5.3 优先函数

四、优先表与优先函数的关系

- 3、如果存在一对优先函数，就存在无穷多对优先函数。
- 注：由于优先函数是在优先表的基础上才能构造出来，能力又弱于优先表，唯一优点是节省空间，所以近来优先函数用作语法分析已不多见了。

小结

- 1、简单优先分析的思想
- 2、算符优先文法及优先表的构造（重点）
  - FIRSTVT、LASTVT
- 3、通用算符优先分析（重点）