§4.2.3算符优先分析法

1、算符优先文法

1)算符文法(OG文法) (Operator Grammar)

设一文法G,若G中不含U→…VW…规则

$V, W \in V_n$

原因(1)两个运算符之间操作数个数不定

(2)每个运算符的操作数不定

OG文法：不会含有两个非终结符号相邻的句型

优先关系：两个终结符号间

2)算符优先文法(OPG)　对象：终结符

Operator Precedence Grammar

设G是一OG文法,$a,b \in V_t$,　$U,V,W \in Vn$

(1)a = b　当且仅当OG有

形如U→….ab….或U→….aVb….的规则

(2)a < b　当且仅当OG有(终结符，先前后后，后大)

形如U→….aW….的规则

而且$W \Rightarrow^{+} b$….或$W \Rightarrow^{+} Vb$…..

(3)a > b　当且仅当OG有(终结符，先后后前，前大)

形如U→….Wb….的规则

而且$W \Rightarrow^{+}$ …. a或$W \Rightarrow^{+}$….aV

若a,b之间至多存在上述三种优先关系之一,OG为OPG文法.

例G[E]:E→E+T｜T

T→T*F｜F

F→(E)｜i

①G[E]是OG文法　不含U→…VW…规则

②由F→(E)得　(=)

由E→E+T　$T \overset{+}{\Rightarrow} T*F$　得 + < *

$T \overset{+}{\Rightarrow} (E)$　得 + < (

$T \overset{+}{\Rightarrow} i$　得 + < i

以此类推,可得教材134页算符优先关系矩阵

2、算符优先分析算法设计

---自底向上语法分析，“最左归约”

---“句柄”：最左素短语

(1)素短语

是这样一个短语,它至少包含有一个终结符号,并且除它自身之外,不再包含其它任何更小的素短语.

(2) 最左素短语:句型最左边的那个素短语.

例G[E]:E→E+T | T

T→T*F | F

F→(E) | i

句型T+T*F+i

短语 T+T*F，T

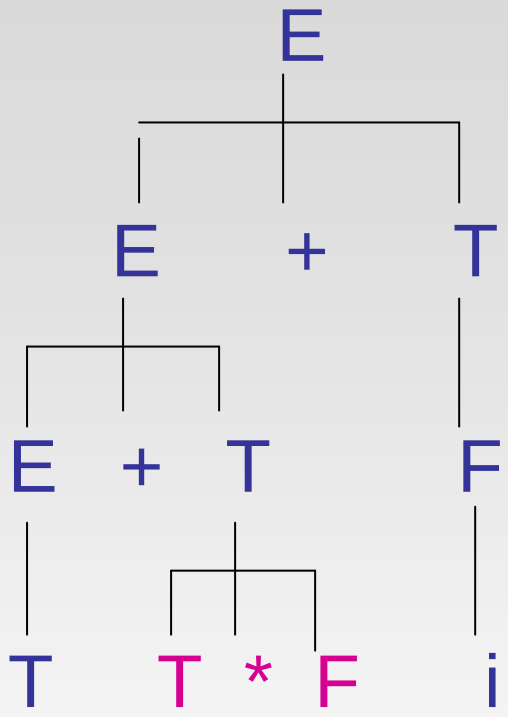
T*F

i

} 素短语

最左素短语:T*F

句柄：T



(3)算符优先文法句型的一般形式

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

$a_i \in V_t$　　$N_i \in V_n$　　可有可无

(4)最左素短语的确定

定理：

一个OPG句型的最左素短语是满足以下条件的最左子串：

$N_ja_j \ldots\ldots N_ia_i N_{i+1}$　　其中：$a_{j-1} < a_j$

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

$a_i > a_{i+1}$

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

3、OPG文法的分析算法

------每次归约均是归约当前句型的最左素短语

- 数据结构:

  符号栈S---存放所有读进的符号(计数i)

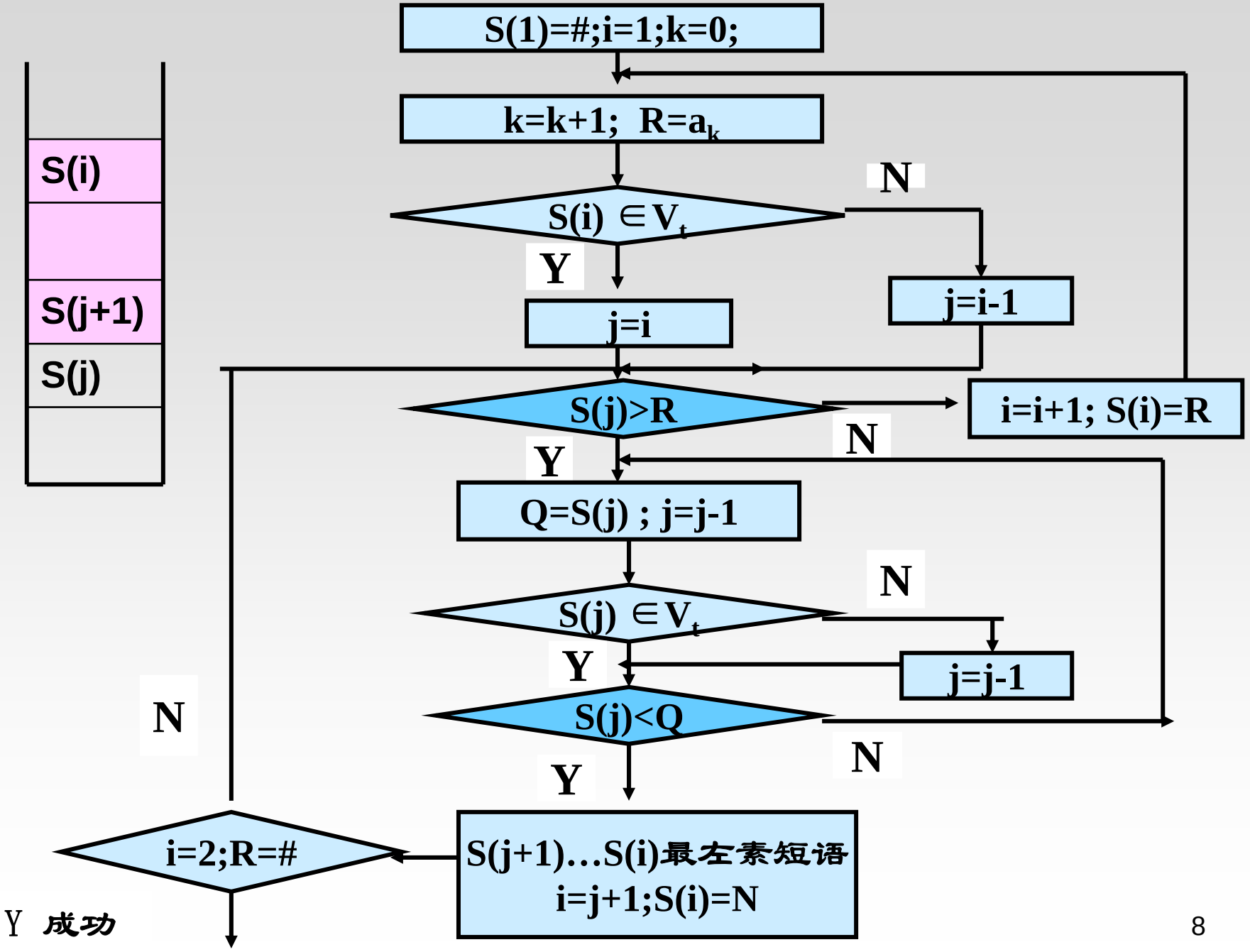
  K----符号栈使用深度

  a-----工作单元

  R,Q-----变量

- 分析算法:先找最左素短语的尾部(>)

  再找最左素短语的头部(<)


S(1)=#;i=1;k=0;
k=k+1; R=a_k
S(i) ∈V_t
N
Y
j=i
j=i-1
S(j)>R
i=i+1; S(i)=R
N
Y
Q=S(j) ; j=j-1
S(j) ∈V_t
N
Y
j=j-1
S(j)<Q
N
N
Y
i=2;R=#
S(j+1)…S(i)最左素短语
i=j+1;S(i)=N
S(i)
S(j+1)
S(j)
Y 成功

- **算符优先关系矩阵**

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

例:输入串 i+i*i 的算符优先分析过程 (查算符优先关系矩阵)

分析栈	优先关系	输入 R
#	#<i	i
# i	#<i>+	+
# N	#<+	
# N +	+<i	i
# N + i	+<i>*	*
# N + N	+<*	

分析栈	优先关系	输入 R
# N + N *	*<i	i
# N + N * i	*<i>#	#
# N + N * N	+<*>#	
# N + N	#<+>#	
# N		

接 受

结论：i+i*i是文法的合法句子

4、算符优先关系矩阵构造

1)算法分析

(1)a ⩵ b　　查规则

(2)a ⩾ b或a ⩽ b　　构造两个集合

$U \in V_n$

①FIRSTVT(U)=

$\{b \mid U \overset{+}{\Rightarrow} b\ldots$, 或$U \overset{+}{\Rightarrow} Vb\ldots,\ b \in V_t, V \in V_n\}$

则形如$W \rightarrow \ldots aU\ldots$的规则　　$a < b$

$b \in$ FIRSTVT(U)

$U \in V_n$

②LASTVT(U)=

$\{a \mid U \overset{+}{\Rightarrow} \ldots a, \text{或} U \overset{+}{\Rightarrow} \ldots aV, a \in V_t, V \in V_n\}$

则形如$W \rightarrow \ldots Ub \ldots$的规则 $a > b$

$a \in$LASTVT(U)

2)构造FIRSTVT(U)和LASTVT(U)的算法

- 两条原则

①若有规则U→b…, 或U→Vb…,

则b ∈ FIRSTVT(U)

②若有规则U →V…,且b ∈ FIRSTVT(V)

则b ∈ FIRSTVT(U)

$U \overset{+}{=}> b\ldots$或

$U \overset{+}{=}> Wb\ldots$

$W \in V_n$

$V \overset{+}{=}> b\ldots$或

$V \overset{+}{=}> Wb\ldots$

$W \in V_n$

- 过程描述

数据结构：

STACK栈

布尔数组F(U,b)　$U \in V_n, b \in V_t$

$$F(U,b)\begin{cases} \text{真} & b \in FIRSTVT(U) \\ \text{假} & b \in FIRSTVT(U) \end{cases}$$

U_b	a	b
E	T	
T		

- 算法分析

初始:F(U,b)初值(根据原则①)

F(U,b)为真的(U,b)对进STACK栈

循环:直至STACK空(根据原则②)

弹出栈顶元素,记(V,b)

对每一个形如U→V...的规则

若F(U,b) 为假,变为真, 进STACK栈

若F(U,b)为真,再循环

结果:FIRSTVT(U)={b｜F(U,b)=TRUE}

同理可得求LASTVT(U)的算法

3)构造OPG优先矩阵的算法

FOR 每一条规则$U \to X_1X_2\ldots X_n$ Do

FOR i:=1 To n-1 Do

…ab… BEGIN IF X_i和X_{i+1}均$\in V_t$ THEN 置$X_i = X_{i+1}$

…aVb… IF i<=n-2 且X_i和X_{i+2}均$\in V_t$ 但$X_{i+1} \in V_n$

THEN 置$X_i = X_{i+1}$

…aU… IF $X_i \in V_t$ 但$X_{i+1} \in V_n$ THEN

FOR b $\in$FIRSTVT(X_{i+1}) Do 置X_i < b

…Ub… IF $X_i \in V_n$ 但$X_{i+1} \in V_t$ THEN

FOR a $\in$LASTVT(X_i) Do 置 a > X_{i+1}

END

4、优先函数

-----用优先函数代替优先关系矩阵

-----节省内存,便于执行比较运算

1)优先函数 $\theta \in V_t$

f(θ):栈内(入栈)优先函数

g(θ):栈外（比较）优先函数

(θ_1, θ_2)

若 $\theta_1 < \theta_2$ 则 $f(\theta_1) < g(\theta_2)$

若 $\theta_1 = \theta_2$ 则 $f(\theta_1) = g(\theta_2)$

若 $\theta_1 > \theta_2$ 则 $f(\theta_1) > g(\theta_2)$

} 建立数字

优先函数的特点：

1. 非全面性　2. 非唯一性　3. 每一对符号都有于对优先数

2)优先函数的构造

(1)有向图法（Bell方法）

①对每$a \in V_t$(包括#),使之对应两个结点f_a,g_a

②对于优先关系矩阵的关系

$a \gtrdot b$　画一条从f_a到g_b的有向弧

$a \lessdot b$　画一条从g_b到f_a的有向弧

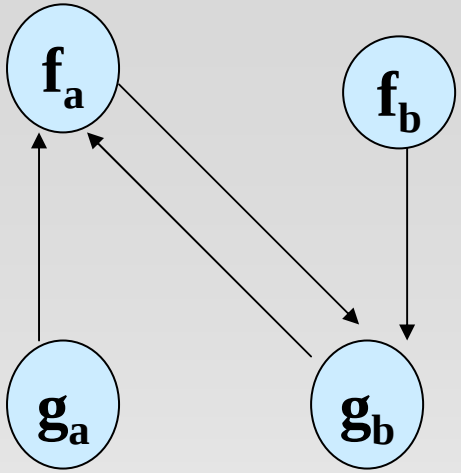
③对每一结点赋一个数,其值为从该结点出发,所能到达的结点个数(包含自身),

赋给f_a的数为f(a), 赋给g_b的数为g(b)

④检查优先函数有无矛盾,如有,则优先函数不存在.

例

	a	b
a	≺	≍
b		≻



优先函数

	a	b
f	2	3
g	3	2

缺点：

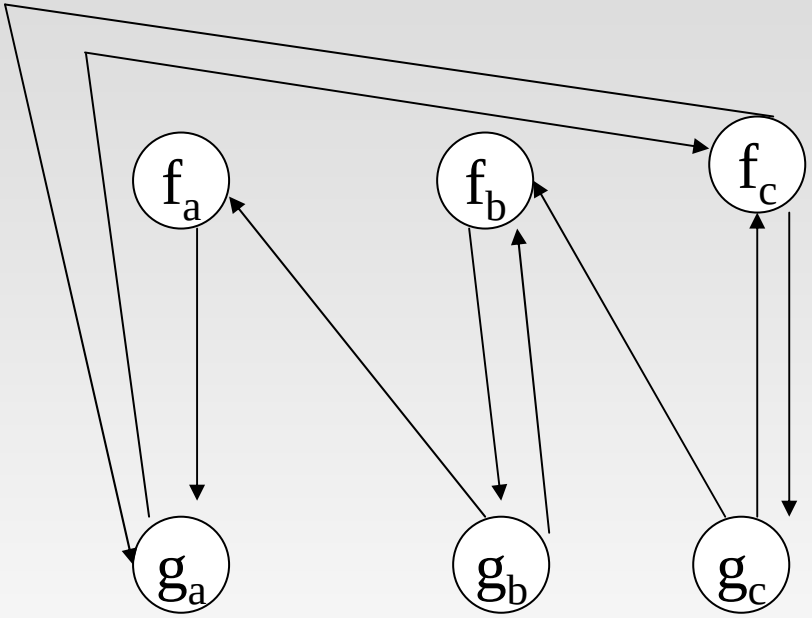
① 原来无优先关系现 也存在优先关系

如b= a

②不是所有的优先矩阵都有优先函数

例：以下优先关系矩阵

f \ g	a	b	c
a	>	<	
b		=	<
c	=		=



优先函数

	a	b	c
f	6	6	6
g	6	6	6

2)Floyd方法 P143

①对每一个f和g赋初值,置$f(\theta)=g(\theta)=1$

②迭代,即每一(θ_1, θ_2)

若 $\theta_1 > \theta_2$ 但$f(\theta_1)<=g(\theta_2)$,

则执行$f(\theta_1)=g(\theta_2)+1$

若 $\theta_1 < \theta_2$ 但$f(\theta_1)>=g(\theta_2)$,

则执行$g(\theta_2)=f(\theta_1)+1$

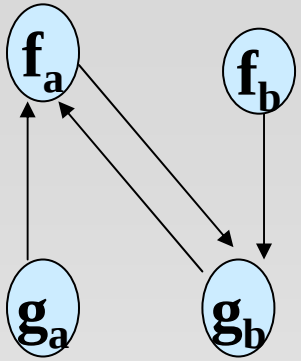
若 $\theta_1 \neq \theta_2$ 但$f(\theta_1) \neq g(\theta_2)$,

则令$f(\theta_1)$和$g(\theta_2)$中的最小者等于最大者

③重复,直至过程收敛为止.

例

f \ g	a	b
a	<	=
b		>



优先函数 BELL方法

	a	b
f	2	3
g	3	2

优先函数 floyd方法

f(a)=f(b)=g(a)=g(b)=1

a<a 且f(a)=g(a) g(a)=f(a)+1=2

a=b 且f(a)=g(b) f(a)=g(b)

b>b 且f(b)=g(b) f(b)=g(b)+1=2

	a	b
f	1	2
g	2	1

收敛

例G[E]:E→E+T｜T

T→T*F｜F

F→(E)｜i

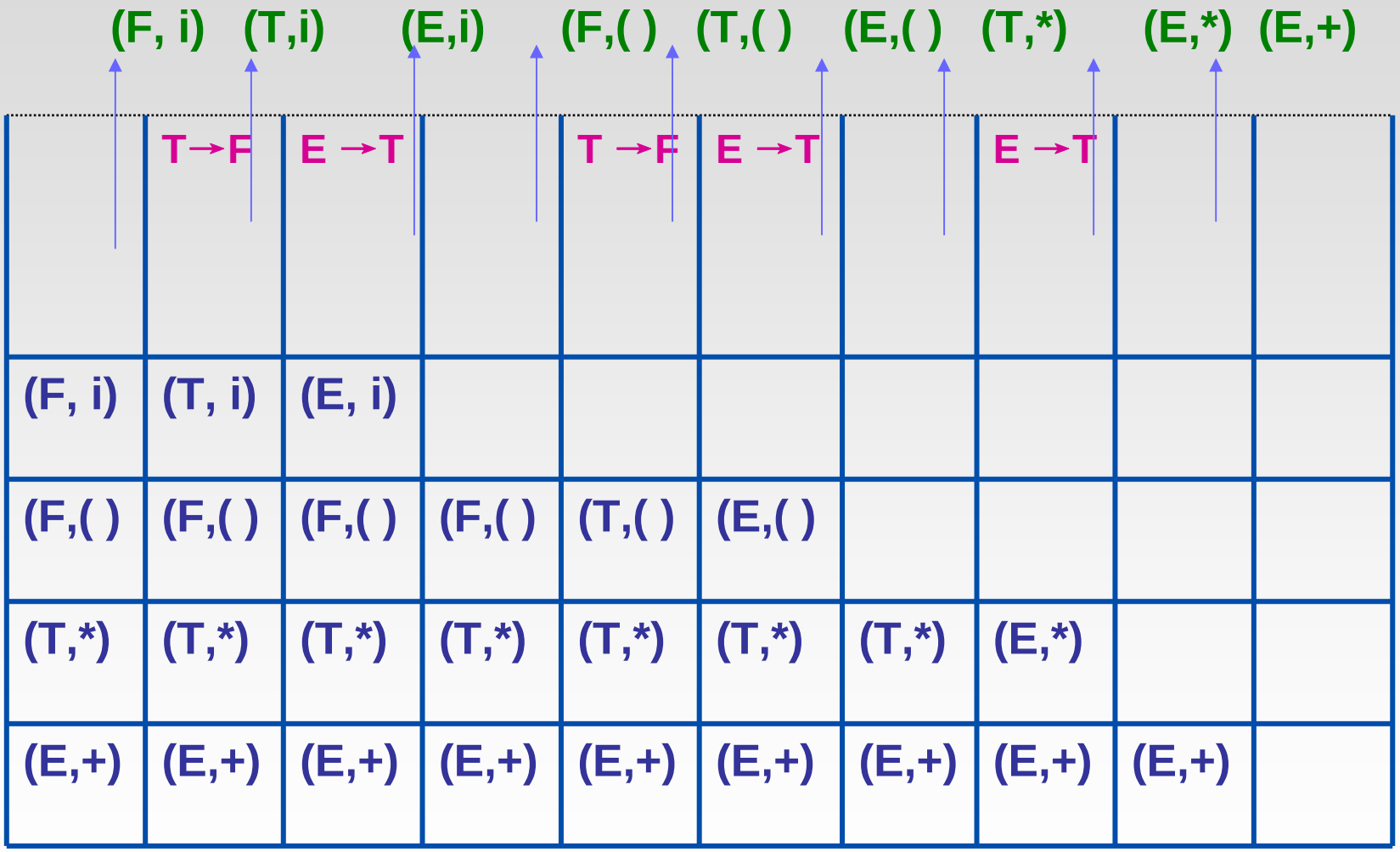
构造FIRSTVT(U)

由原则①初始 FIRSTVT(E)={+, }

FIRSTVT(T)={*, }

FIRSTVT(T)={(,i }

由原则②循环U→V……


(F, i)
(T,i)
(E,i)
(F,( )
(T,( )
(E,( )
(T,*)
(E,*)
(E,+)
T→F
E →T
T →F
E →T
E →T
(F, i)
(T, i)
(E, i)
(F,( )
(F,( )
(F,( )
(F,( )
(T,( )
(E,( )
(T,*)
(T,*)
(T,*)
(T,*)
(T,*)
(T,*)
(T,*)
(E,*)
(E,+)
(E,+)
(E,+)
(E,+)
(E,+)
(E,+)
(E,+)
(E,+)
(E,+)

FIRSTVT(U)

U \ b	+	*	(	)	i
E	T	T	T		T
T		T	T		T
F			T		T

FIRSTVT(E)={+,*,(,i}

FIRSTVT(T)={*,(,i}

FIRSTVT(E)={(,i}

例G[E]:E→E+T｜T

T→T*F｜F

F→(E)｜i

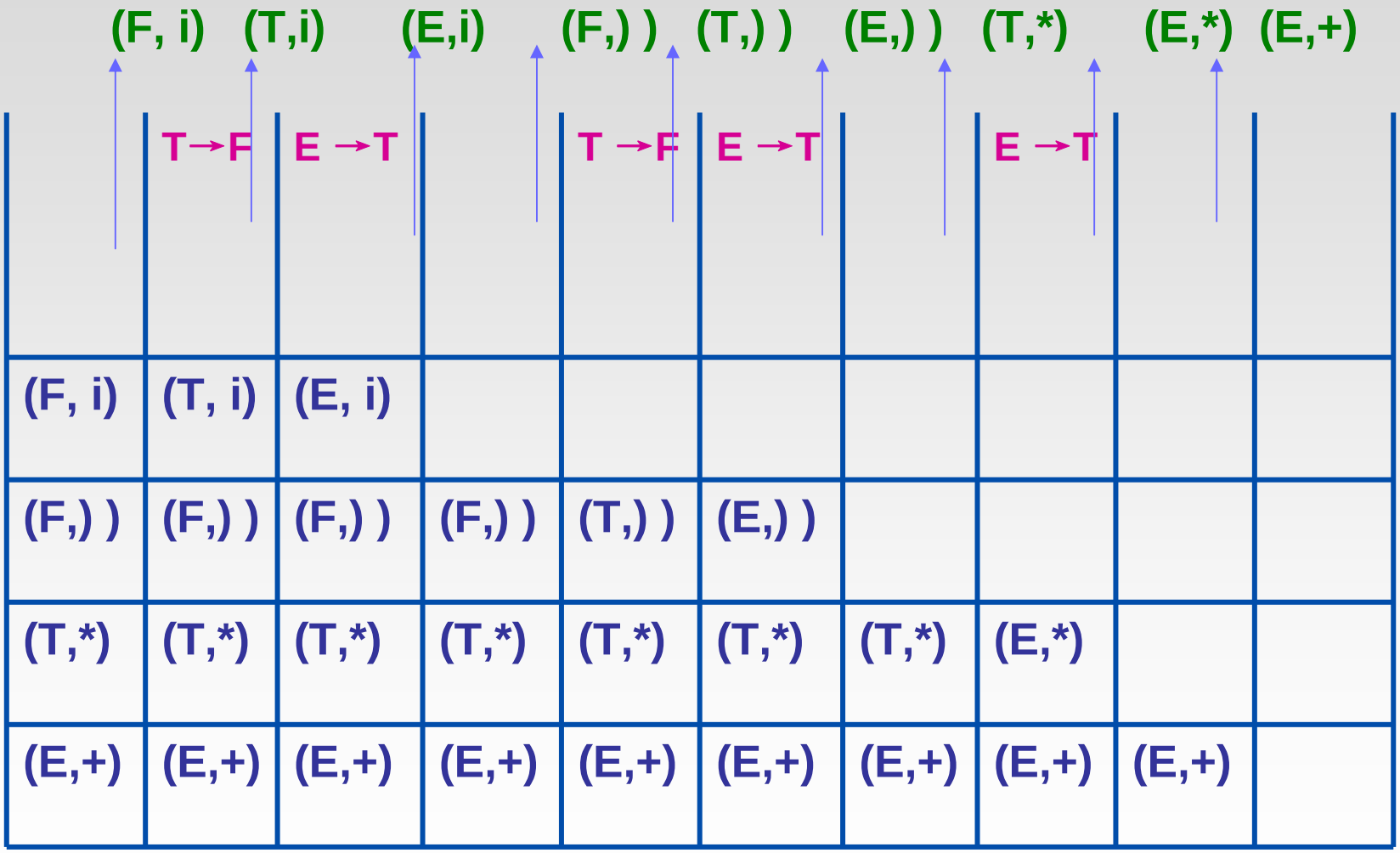
构造LASTVT(U)

由原则①初始 LASTVT(E)={+, }

LASTVT(T)={*, }

LASTVT(T)={),i }

由原则②循环U→……V


(F, i)
(T,i)
(E,i)
(F,))
(T,))
(E,))
(T,*)
(E,*)
(E,+)
T→F
E →T
T →F
E →T
E →T
(F, i)
(T, i)
(E, i)
(F,))
(F,))
(F,))
(F,))
(T,))
(E,))
(T,*)
(T,*)
(T,*)
(T,*)
(T,*)
(T,*)
(T,*)
(E,*)
(E,+)
(E,+)
(E,+)
(E,+)
(E,+)
(E,+)
(E,+)
(E,+)
(E,+)

LASTVT(U)

U \ b	+	*	(	)	i
E	T	T		T	T
T		T		T	T
F				T	T

LASVT(E)={+,*,),i}

LASTVT(T)={*,),i}

LASTVT(E)={),i}

G[E]:E→E+T | T
T→T*F | F
F→(E) | i

FIRSTVT(U)

U \ b	+	*	(	)	i
E	T	T	T		T
T		T	T		T
F			T		T

LASTVT(U)

U \ b	+	*	(	)	i
E	T	T		T	T
T		T		T	T
F				T	T

- 算符优先关系矩阵 及优先函数

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

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

讨论：

(1) 简单优先分析所确定的可归约串是
当前句型的句柄。

(2) 算符优先分析属于自底向上的语法分析，
但不是严格的最左归约。

(3) 算符优先分析所确定的可归约串是
当前句型的最左素短语.

(4) 算符优先分析只考虑终结符号之间的优先关系。

(5) 算符优先分析算法很容易程序实现。

(6) 算符优先分析是比简单优先分析更有效的分析法。