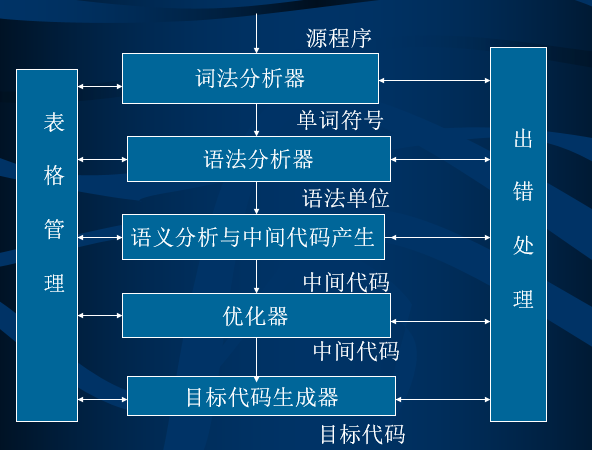
编译技术

(complete reference https://blog.csdn.net/BlueCloudMatrix/article/details/41745701 )

# **概念**





## **有害规则**

形如U→U的产生式。会引起文法的二义性

## **多余规则**

指文法中任何句子的推导都不会用到的规则

文法中某些非终结符（除了开始符）不在任何规则的右部出现，该非终结符称为**不可到达**

文法中某些非终结符，由它不能推出终结符号串，该非终结符称为**不可终止**

## **推导**

最右推导被称为规范推导

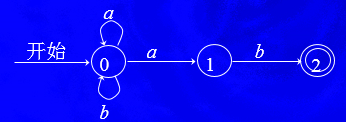
由规范推导所得的句型称为规范句型

## **归约**

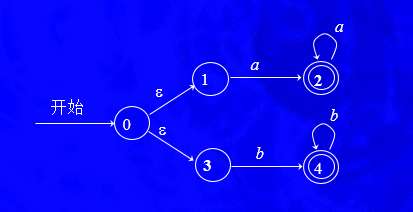
最左归约为规范归约

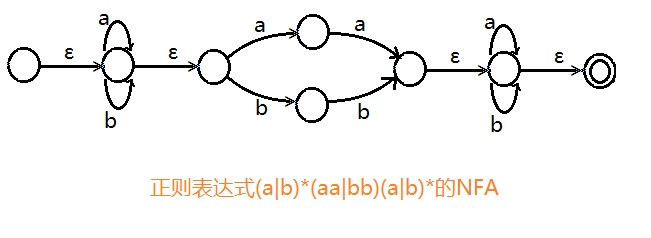
# **NFA**

1. (a|b)\*ab的NFA



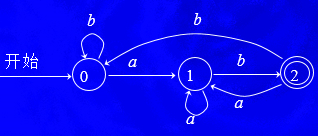
1. aa\*|bb\*的NFA





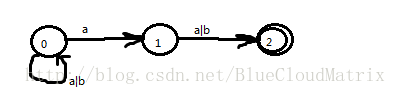
# **DFA**

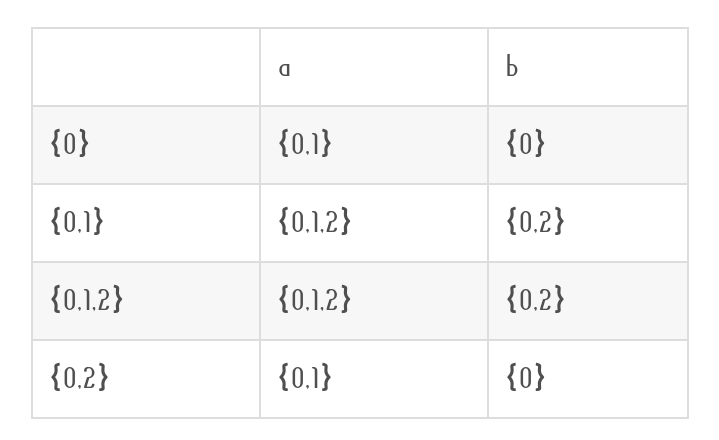
## **(a|b)\*ab的DFA**



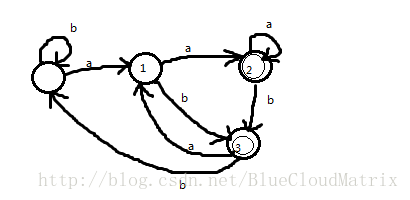
(a|b)\*a(a|b)

## **NFA:**





初始为{0}，因为0和1之间就需要a了。{0}和a结合能到达0和1，所以在<{0}, a>处填{0, 1}，同理，对于{0, 1, 2}和a结合（0,1,2分别和a结合）能到达0,1,2，所以在<{0,1,2}, a>处填{0,1,2}，而{0, 1, 2}和b结合，0与b结合到达0，1与b结合到达2，2与b结合到达空，所以在<{0,1,2}, b>处填{0,2}



## **DFA最小化**

初始划分：{0,1}（非终态） {2,3}（终态）

因为{0,1}a = {1,2}不属于上面任意一个子集，所以分开：{0}，{1}

因为{2,3}a = {1,2}不属于上面任意一个子集，所以分开：{2}，{3}

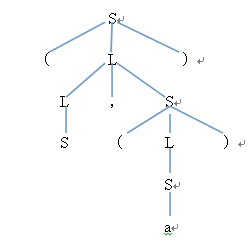
所以已经是最简

# **语法树，短语，简单短语，句柄，素短语**

## **文法**

S->(L)|aS|a

L->L,S|S



## **短语**：

所有大小树的叶子节点(S,(a)) | a | (a) | S | S,(a)

## **简单短语**：

不能再推出子树的节点a S

## **句柄**：

最左简单短语

## **素短语：**

# **FIRST, FOLLOW, SELECT, FIRSTVT, FOLLOWVT, LL(1)**

Vt: 终结符

Vn: 非终结符

## **FIRST & FOLLOW**

例：

S→AB

S→bC

A→ε

A→b

B→ε

B→aD

C→AD

C→b

D→aS

D→c

FIRST(S)={a,b,ε}

FIRST(A)={b,ε}

FIRST(B)={a,ε}

FIRST(C)={a,b,c}

FIRST(D)={a,c}

FOLLOW(S)={#}

FOLLOW(A)={a,#,c}

FOLLOW(B)={#}

FOLLOW(C)={#}

FOLLOW(D)={#}

例：

E →TE’

E’→+TE’

E’→ε

T →FT’

T’→\*FT’

T’→ε

F→(E)|i

FIRST(E)=FIRST(T)=FIRST(F)={ ( ，i }

FIRST(E’)= { + ，ε}

FIRST(T’)={ \* ，ε}

FOLLOW(E)= {)，#}

FOLLOW(E’)= FOLLOW(E)={ ) ，# }

FOLLOW(T)={FIRST(E’)-{e}}∪FOLLOW(E)∪FOLLOW(E’) = { + ， ) ， # }

FOLLOW(T’)= FOLLOW(T)= { + ， ) ， # }

FOLLOW(F)={FIRST(T’)-{e}})∪FOLLOW(T)∪FOLLOW(T’) = {+，\*，) ，# }

## **SELECT:**

文法G [S]为：

　　S→AB

　　S→bC

　　A→ε

　　A→b

　　B→ε

B→aD

C→AD

C→b

D→aS

D→c

求每个产生式的SELECT集，并判断文法G是否为LL(1)文法？

解：SELECT(S→AB)=（FIRST(AB)-{ε})∪FOLLOW(S)={b,a,#}

SELECT(S→bC)=FIRST(bC)={b}

SELECT(A→ε)=(FIRST(ε) -{ε})∪FOLLOW(A)={a,c,#}

SELECT(A→b)=FIRST(b)={b}

SELECT(B→ε)=(FIRST(ε) -{ε})∪FOLLOW(B)={#}

SELECT(B→aD)=FIRST(aD)={a}

SELECT(C→AD)=FIRST(AD)={a,b,c}

SELECT(C→b)=FIRST(b)={b}

SELECT(D→aS)=FIRST(aS)={a}

SELECT(D→c)=FIRST(c)={c}

SELECT(S→AB)∩SELECT(S→bC)={b,a,#}∩{b}={b}≠ф

SELECT(A→ε)∩SELECT(A→b)={a,c,#}∩{b}=ф

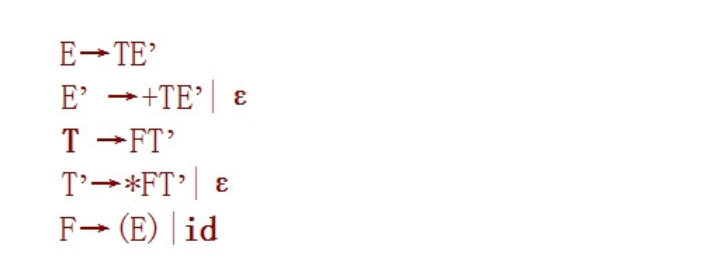
SELECT(B→ε)∩SELECT(B→aD)={#}∩{a}=ф

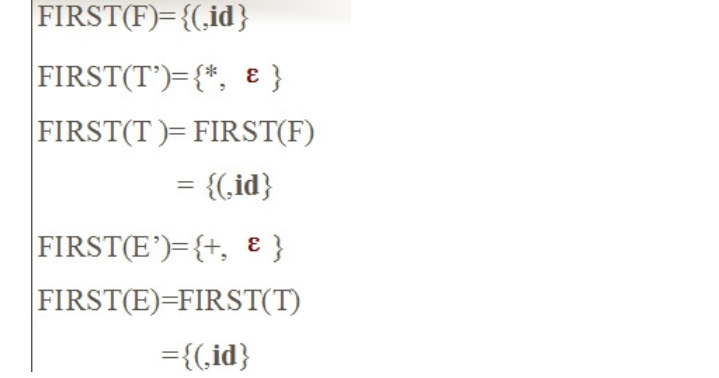
SELECT(C→AD)∩SELECT(C→b)＝{b,a,c}∩{b}＝{b}≠ф

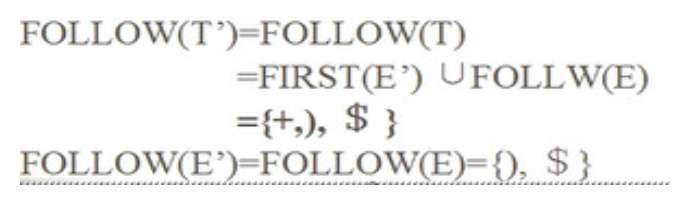
SELECT(D→aS)∩SELECT(D→c)={a}∩{c}＝ф

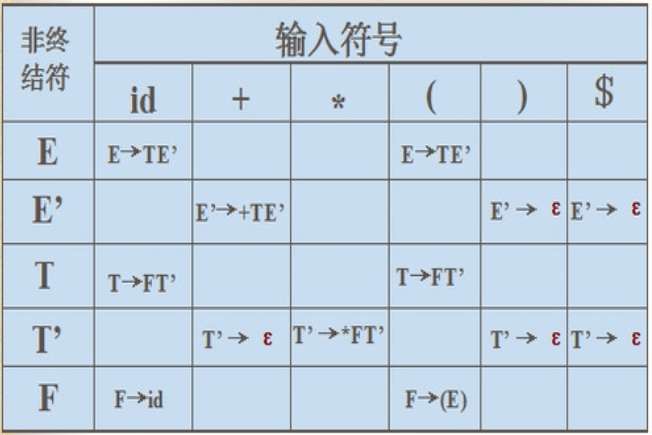
由LL(1)文法定义知该文法不是LL(1)文法，因为具有相同左部其产生式的SELECT集的交集不为空。

## **LL(1):**











## **FIRSTVT规则：**

A->a.......，即以终结符开头，该终结符入Firstvt

A->B.......，即以非终结符开头，该非终结符的Firstvt入A的Firstvt

A->Ba.....，即先以非终结符开头，紧跟终结符，则终结符入Firstvt

## **LASTVT规则：**

A->.......a，即以终结符结尾，该终结符入Lastvt

A->.......B，即以非终结符结尾，该非终结符的Lastvt入A的Lastvt

A->.....aB，即先以非终结符结尾，前面是终结符，则终结符入Firstvt

# **三元式，四元式，逆波兰**

## **三元式：**

a:=b\*c+b\*d

1. (\* b,c)
2. (\* b,d)
3. (+ (1),(2))
4. (:= (3),a)

## **四元式：**

a:=b\*c+b\*d

1. (\*,b,c,t1)
2. (\*,b,d,t2)
3. (+,t1,t2,t3)
4. (:=,t3,-,a)

## **逆波兰：**

a+b\*c->abc\*+

(a+b)\*c->ab+c\*

a:=b\*c+b\*d->abc\*bd\*+:=

# **右线性，左线性**

## **右线性：**

形如：

A → aB

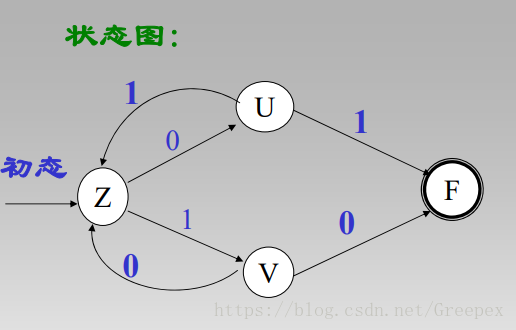
A → a

G[Z]:

Z→0U∣1V

U →1Z∣1

V →0Z∣0



## **左线性：**

形如：

A → Ba

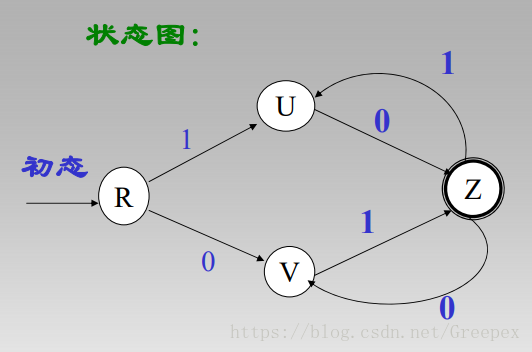
A → a

G[Z]:

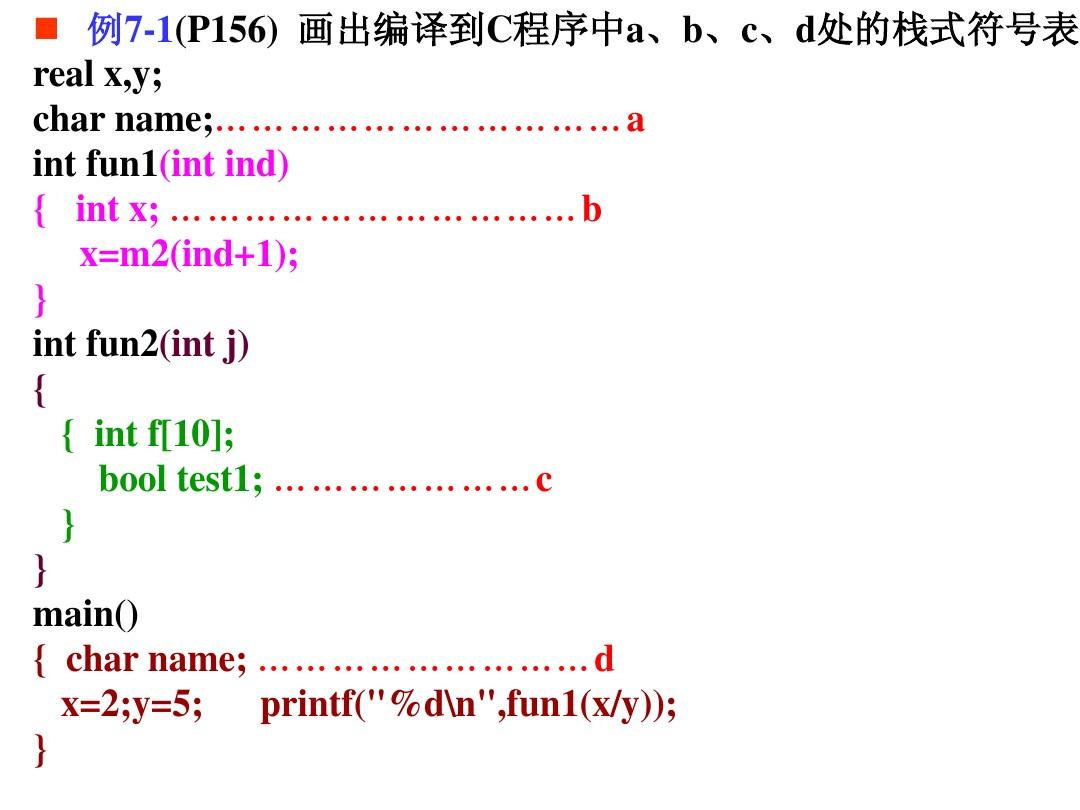
Z→U0∣V1

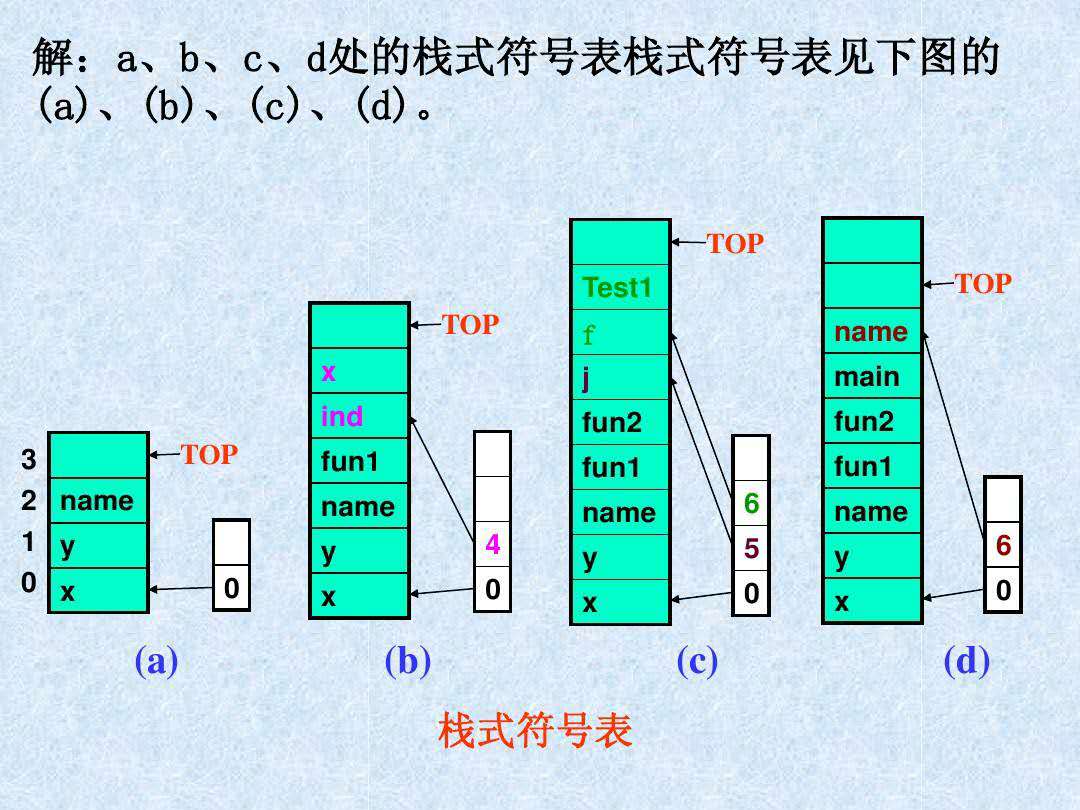
U →Z1∣1

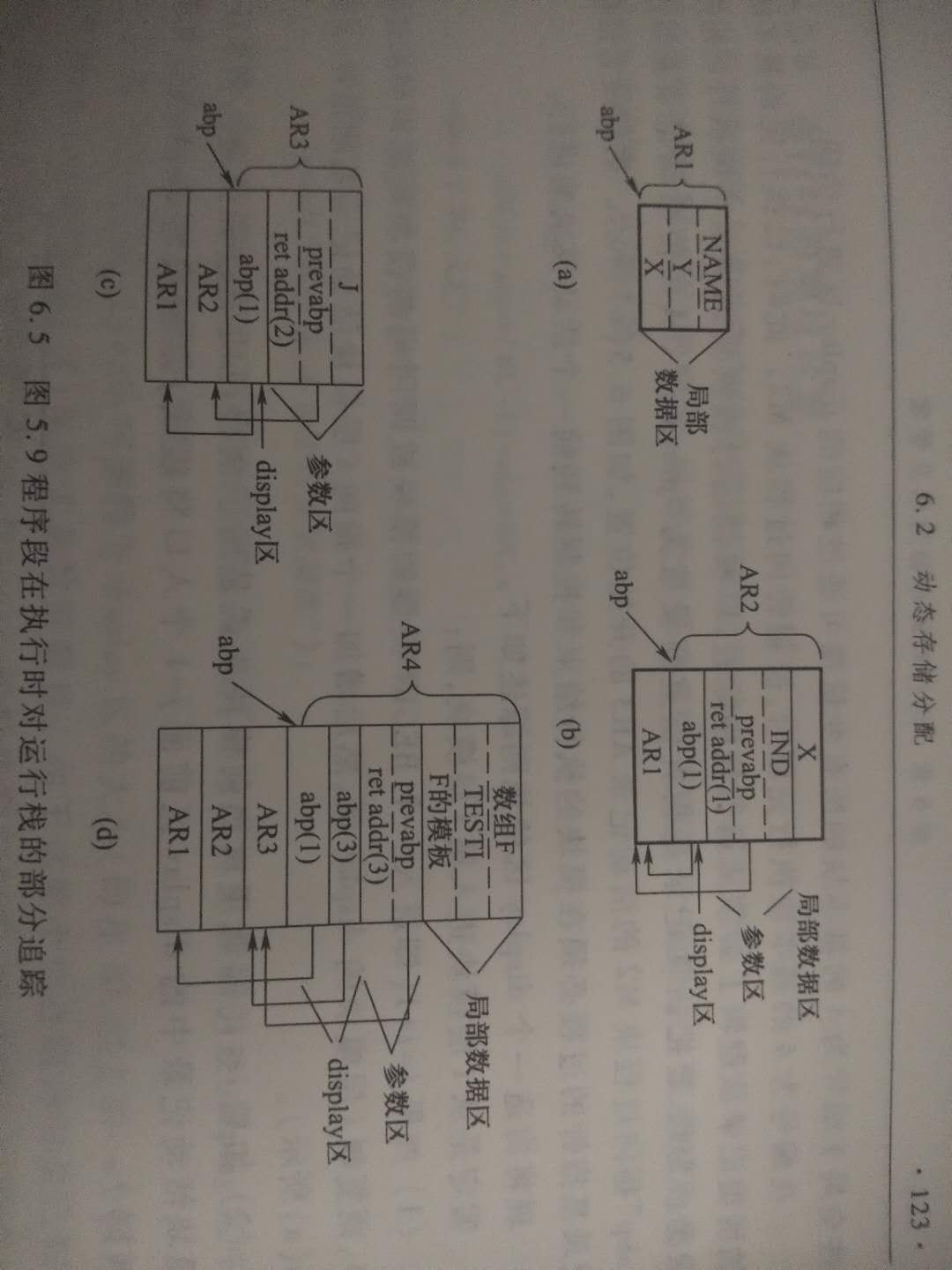
V →Z0∣0



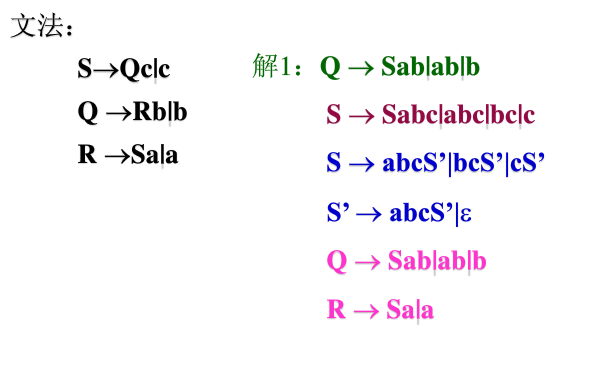
# **栈式符号表**

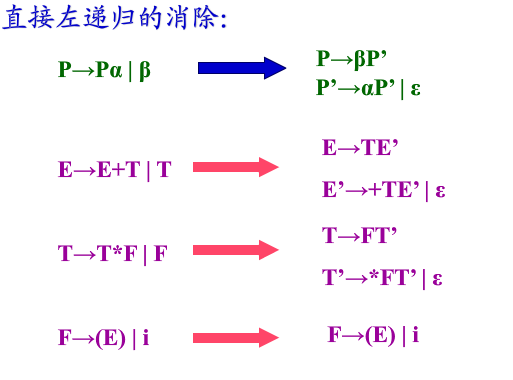




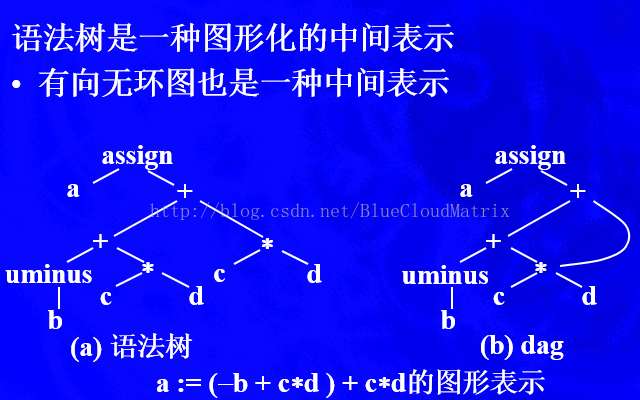


# **消除左递归**

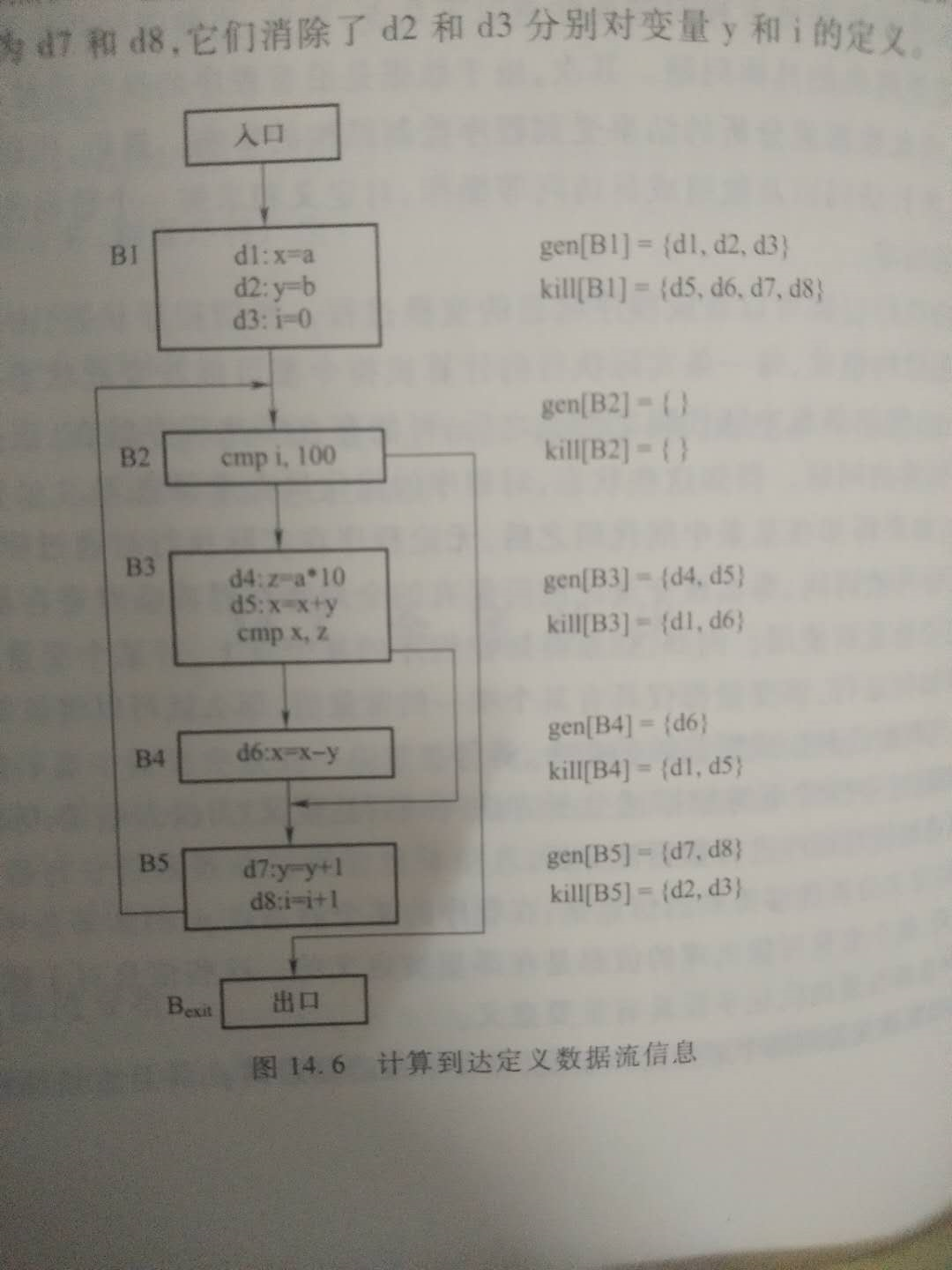




# **DAG图**

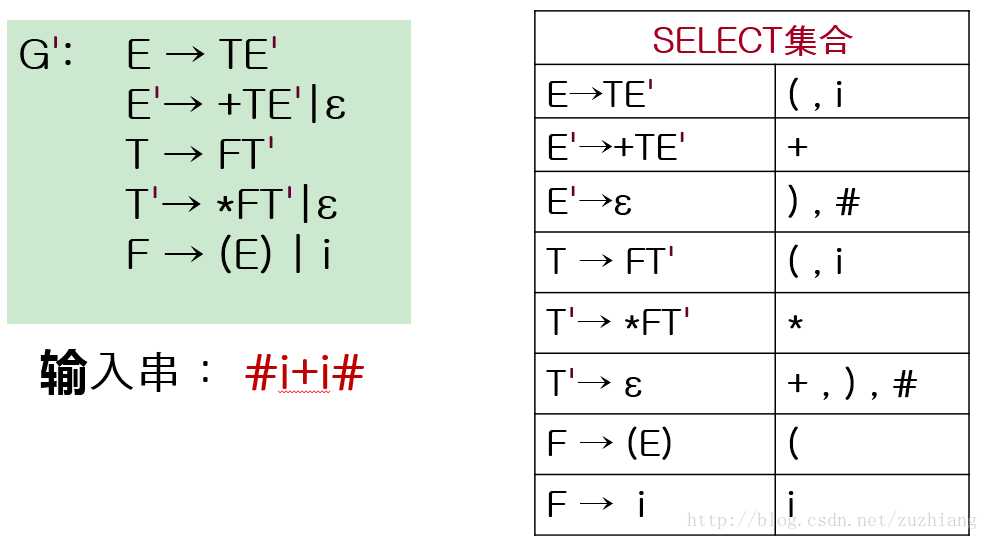


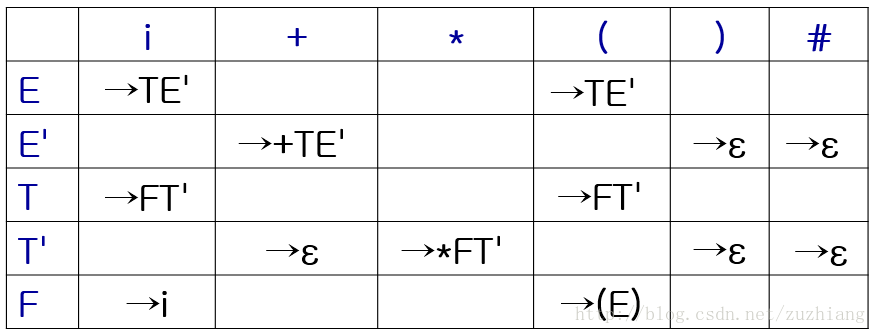
# **代码优化**



# **LL(1) 、LR(0)、SLR、LR(1)、LALR**

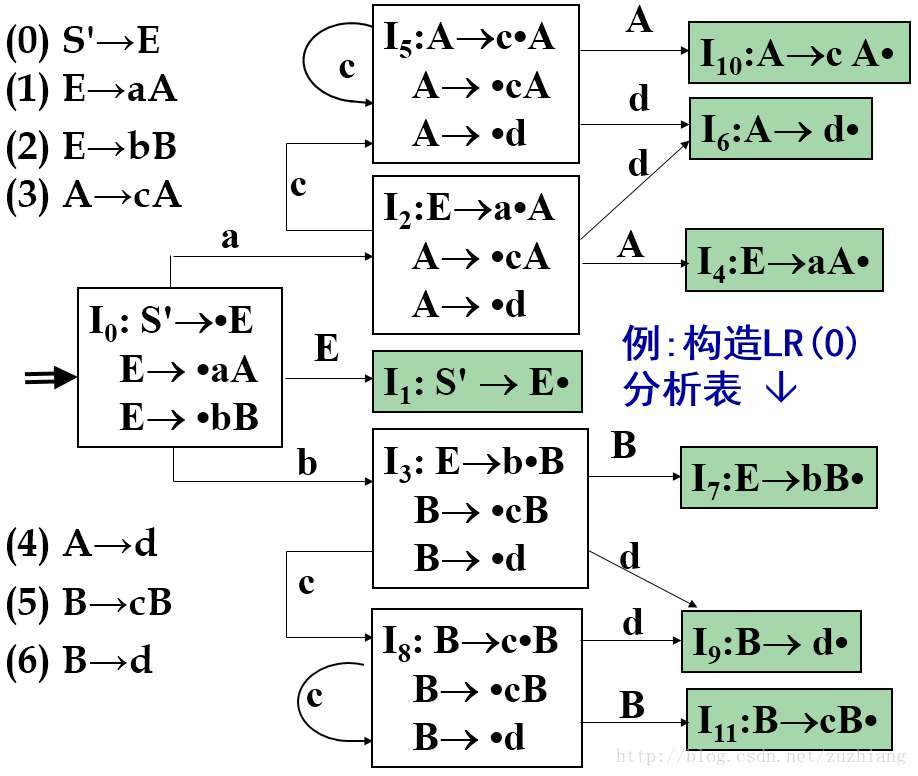
## **LL(1):**

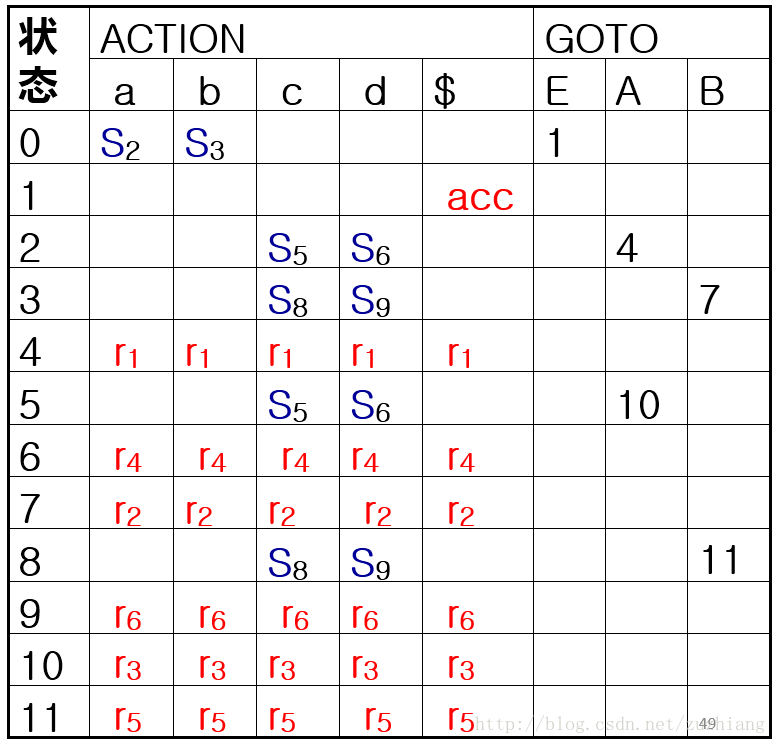




## **LR(0):**

若点之后是非终结符则需要把那非终结符的文法写下





## **LALR:**

