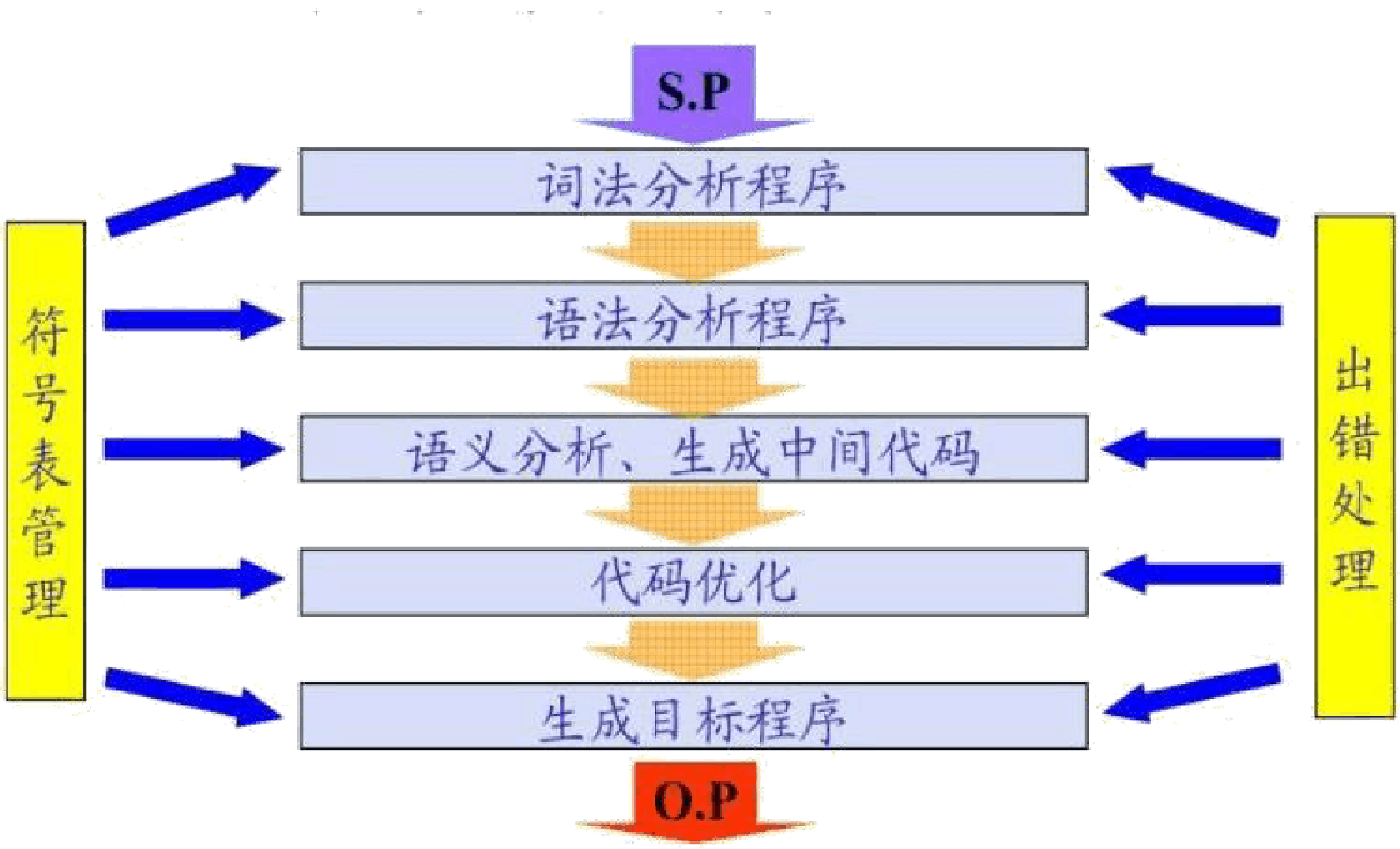
第一章

练习 1

2、典型的编译程序可区分为哪几个主要的逻辑部分？各部分的主要功能是什么？

典型的编译程序拥有 7 个逻辑部分：



第二章

练习 2.2

4. 试证明： $A^+ = AA^* = A^*A$

证：∵ $A^* = A^0 \cup A^+$，$A^+ = A^1 \cup A^2 \cup \cdots \cup A^n \cup \cdots$

得： $A^* = A^0 \cup A^1 \cup A^2 \cup \cdots \cup A^n \cup \cdots$

∴ $AA^* = A(A^0 \cup A^1 \cup A^2 \cup \cdots \cup A^n \cup \cdots)$

$= AA^0 \cup AA^1 \cup AA^2 \cup \cdots \cup AA^n \cup \cdots$

$=A\cup A2\cup A3\cup An+1\cup$

$= A+$

同理可得：

$A*A =(A0\cup A1\cup A2\cup \quad \cup\ An\cup \quad)A$

$=A0\ A\cup A1A\cup A2A\cup \quad \cup\ AnA\cup$

$= A\cup A2\cup A3\cup An+1\cup$

$= A+$

所以： $A+ =AA*=A*A$

练习 2.3

1.设 G[〈表记符〉]的规则是 ：

〈表记符〉 ::=a|b|c|

〈表记符〉 a| 〈表记符〉 c|

〈表记符〉 0| 〈表记符〉 1

试写出 VT 和 VN，

并对以下符号串 a,ab0,a0c01,0a,11,aaa 给出可能的一些推导。

解： VT ={a,b,c,0,1}， VN ={〈表记符〉 }

(1) 不可以推导出 ab0,11,0a

(2)〈表记符〉 =>a

(3)〈表记符〉 =>〈表记符〉 1

=>〈表记符〉 01 => 〈表记

符〉 c01 => 〈表记符〉 0c01

=>   a0c01

(4)〈表记符〉 =>〈表记符〉 a

=>〈表记符〉 aa

=>aaa

2．写一文法，其语言是偶整数的会合

解： G[<偶整数 >]：

<偶整数 >::= <符号 > <偶数字 >| <符号 ><数字串 ><偶数字 >

<符号 > ::= + | — | ε

<数字串 >::= <数字串 ><数字 >|< 数字 >

<数字>::=<偶数字>|1|3|5|7|9

<偶数字>::=0|2|4|6|8

4. 设文法 G 的规则是：
〈A〉 ::=b<A>| cc

试证明： cc, bcc, bbcc, bbbcc ∈L[G]

证：（1）〈A〉 =>cc

（2）〈A〉 =>b〈A〉 =>bcc

（3）〈A〉 =>b〈A〉 =>bb〈A〉 =>bbcc

（4）〈A〉 =>b〈A〉 =>bb〈A〉 =>bbb〈A〉 =>bbbcc

又∵ cc, bcc, bbcc, bbbcc ∈Vt*

∴由语言定义， cc, bcc, bbcc, bbbcc ∈L[G]

5 试对以下语言结构相应文法：

（1）{ a(bn)a | n=0,1,2,3, },此中左右圆括号为终结符。

（2） { （an ）（bn ）| n=1,2,3, }

解：（1）文法 [G〈S〉]：

S::= a(B)a

B::= bB | ε

(2) 文法 [G〈S〉]：--错了，两个 n 不等

S ::= (A)(B)

A::= aA|a

B::= bB|b

7.对文法 G3[〈表达式〉]

〈表达式〉 ::=〈项〉 | 〈表达式〉 +〈项〉 | 〈表达式〉 -〈项〉

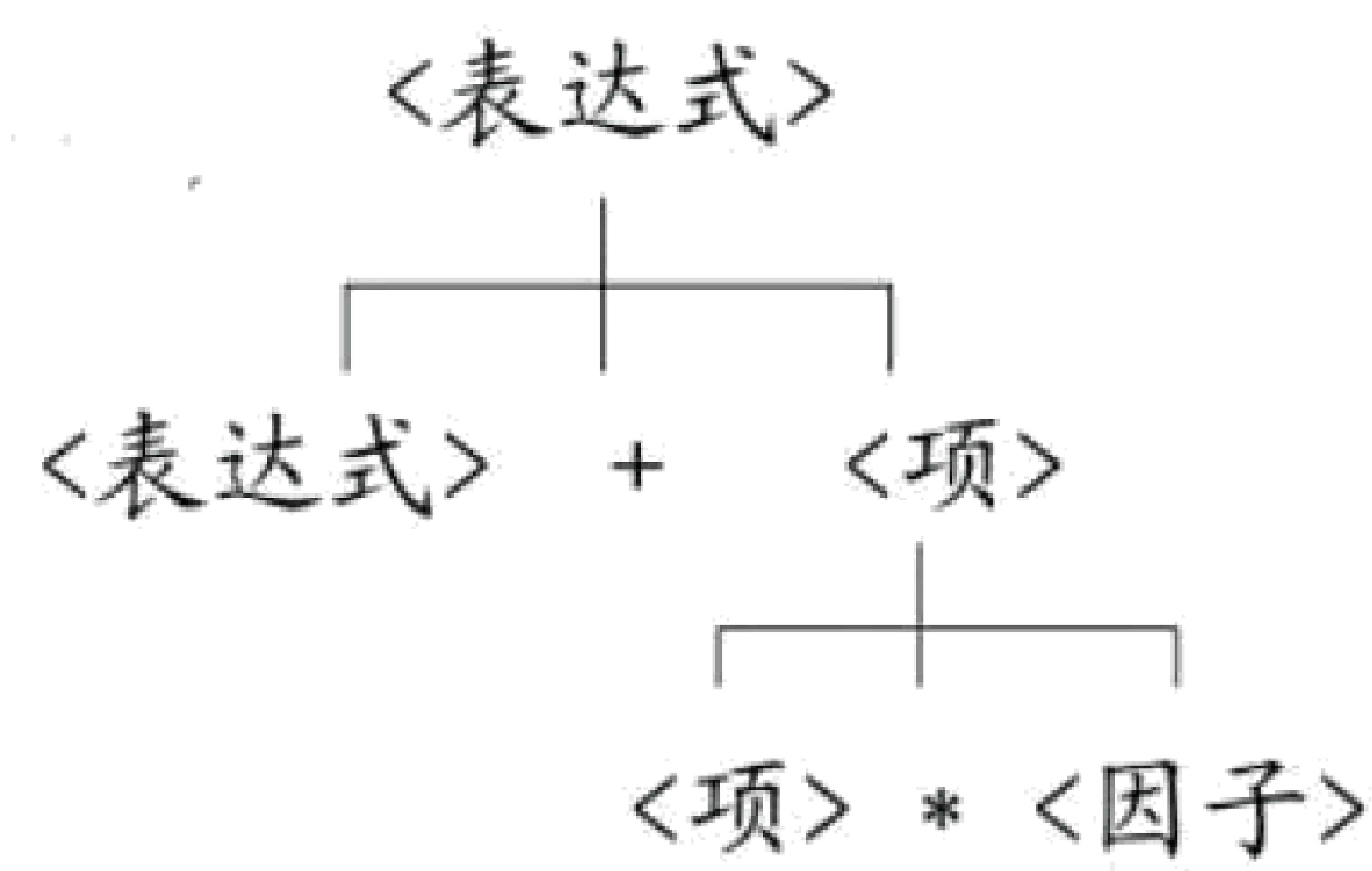
〈项〉 ::=〈因子〉 | 〈项〉 * 〈因子〉 | 〈项〉 / 〈因子〉

〈因子〉 ::=（〈表达式〉）| i

列出句型〈表达式〉 +〈项〉 * 〈因子〉的全部短语和简单短语。

<表达式 > => < 表达式 > + < 项>

=> <表达式 > + < 项> * <因子 >



短语有：

〈表达式〉 +〈项〉 * 〈因子〉和〈项〉 * 〈因子〉

简单短语是：〈项〉 * 〈因子〉

8 文法 V::= aaV | bc 的语言是什么？

?

解： L（ G[V]）= {a2nbc | n=0,1,2, }

V ? aaV ? aaaaV ? .... ? a2nbc (n ≥ 1)

V ? bc (n=0)

练习 2.4

5.已知文法 G[E]：

E::=ET+ | T

T::=TF* |F

F::=FP ↑ |P

P::=（E）| i

有句型 TF*PP ↑+，

问此句型的短语，简单短语，和句柄是什么？

解：此句型的短语有： TF*PP ↑+，TF*， PP ↑， P

简单短语有： TF*，P

句柄是： TF*

8.证明下边的文法 G 是二义的：

S::= iSeS | iS | i

证：由文法可知 iiiei 是该文法的句子，

又由文法可知 iiiei 有两棵不一样的语法树。

所以该文法是二义性文法。

第三章

练习 3.1

1．画出下述文法的状态图

〈Z〉::=〈B〉e

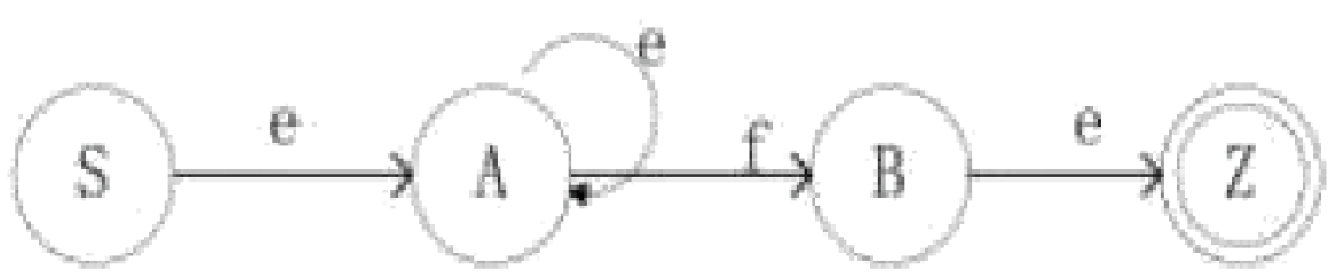
〈B〉::=〈A〉f

〈A〉::= e | 〈A〉e

使用该状态图检查以下句子是不是该文法的合法句子

f，eeff，eefe

解:



f，eeff不是该文法的合法句子，eefe是该文法的合法句子

2、有以下状态图，此中 S 为初态，Z 为终态。

（1） 写出相应的正则文法；

（2） 写出该文法的 V，Vn 和 Vt;

（3） 该文法确立的语言是什么？

解：（1）Z→A1|0　　A→A0|0

（2） V={A,Z,0,1}

Vn={A,Z}

Vt={0,1}

（3） L (G[S])= {0 或 0n1,n ≥1}

L (G[S])= {0|00*1}

练习 3.2

1．令 A，B，C 是随意正则表达式，证明

以下关系建立：

A|A=A

（A*）*= A*

A*= ε | AA*

（AB）*A = A（BA）*

（A|B）* =（A*B*）*=（A*|B*）

证明：

⑴A｜A=｛x｜x∈L(A)或 x∈L(A)｝＝｛

x｜x∈L(A)｝＝A

⑵（A*）*＝（A*）0∪（A*）1∪（A*）2∪ ∪（A*）n

＝ε∪(A0∪A 1∪A2∪ ∪ An) ∪(A1)

＝ε∪A0∪A 1∪A2∪ ∪ An

＝A*

⑶ε｜AA*所表示的语言是 :

{ε}∪LA • LA*

＝LA0∪LA（LA0∪LA1∪LA2∪ ）

＝LA0∪LA1∪LA2∪ ＝LA*

故 ε ｜AA*＝A*

(4)

(LALB)*LA=({ε }∪ L(A)LB∪LALBLALB∪LALBLALBLALB

∪) LA

=LA∪LALBLA∪LALBLALBLA∪LALBLALBL

ALB∪LA

= LA∪({ ε }∪LBLA∪LBLALBLA∪)

= LA(LBLA)

∴(AB)*A=A(BA)*

⑸三个表达式所描绘的语言都是 LALB中

随意组合

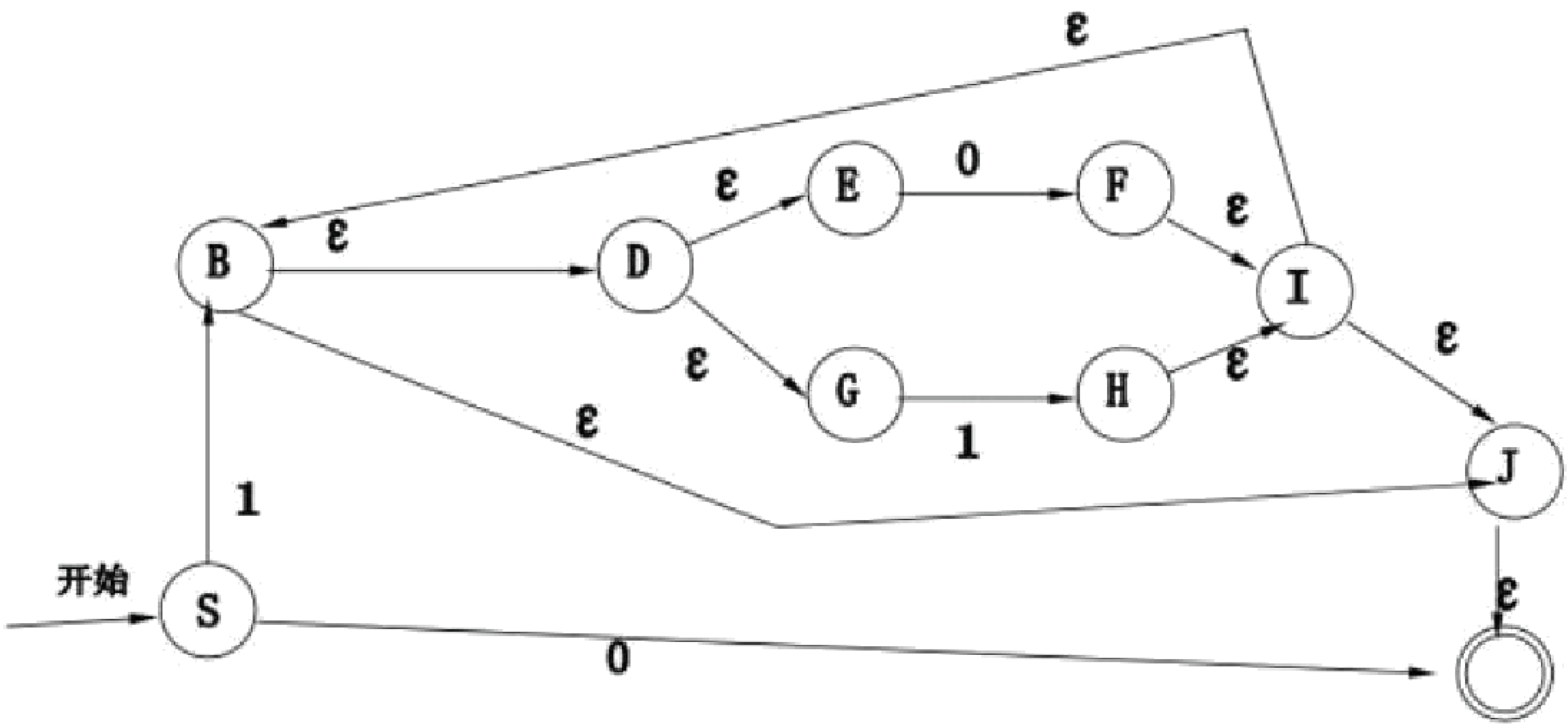
∴（ A|B ）*=(A*B*)=(A*|B*)*

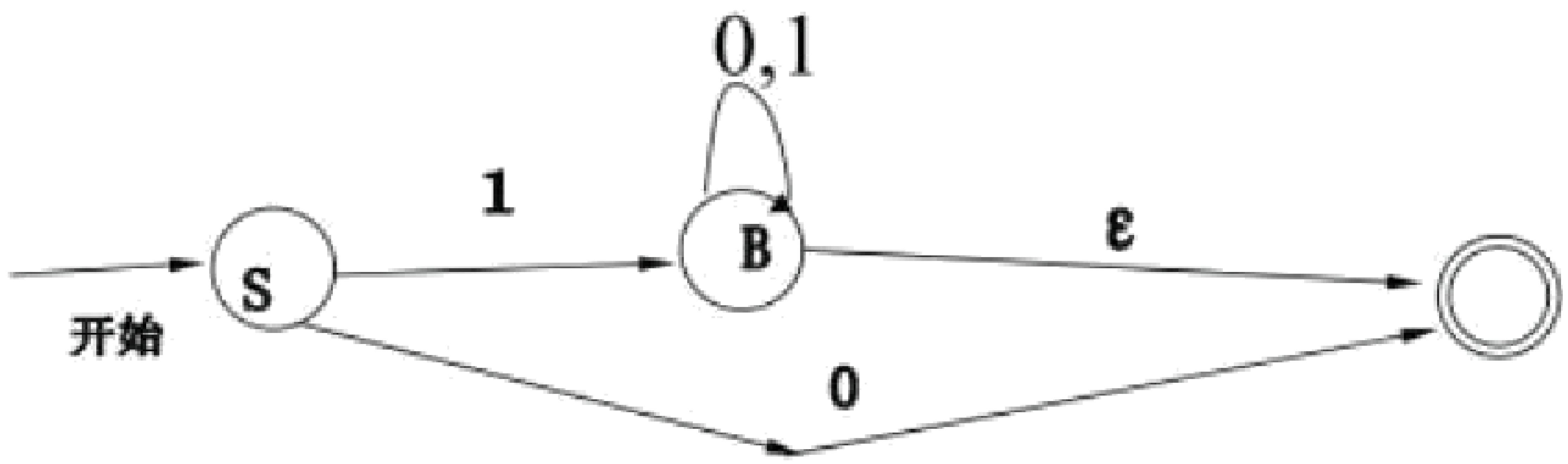
2. 结构以下正则表达式相应的 DFA

（1） 1（0|1 ）*|0

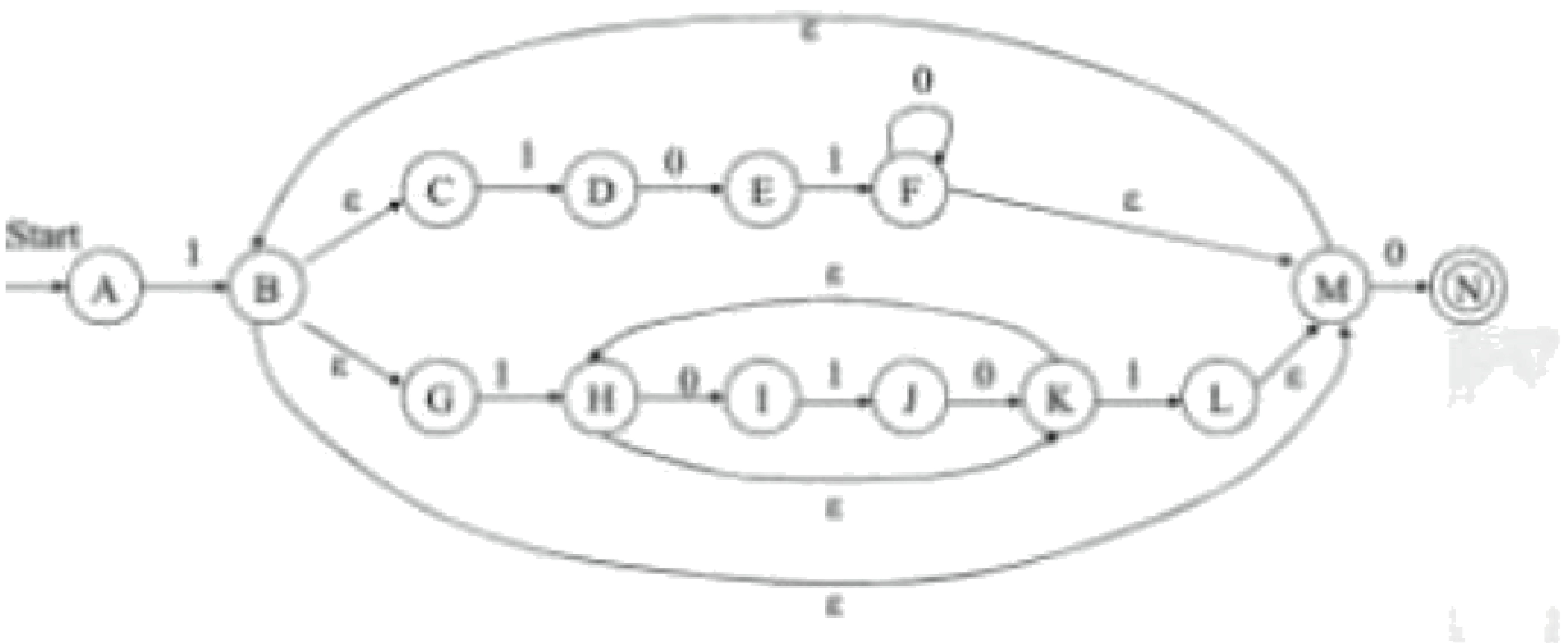
（2） 1（1010*|1 （010 ）*1 ）*0

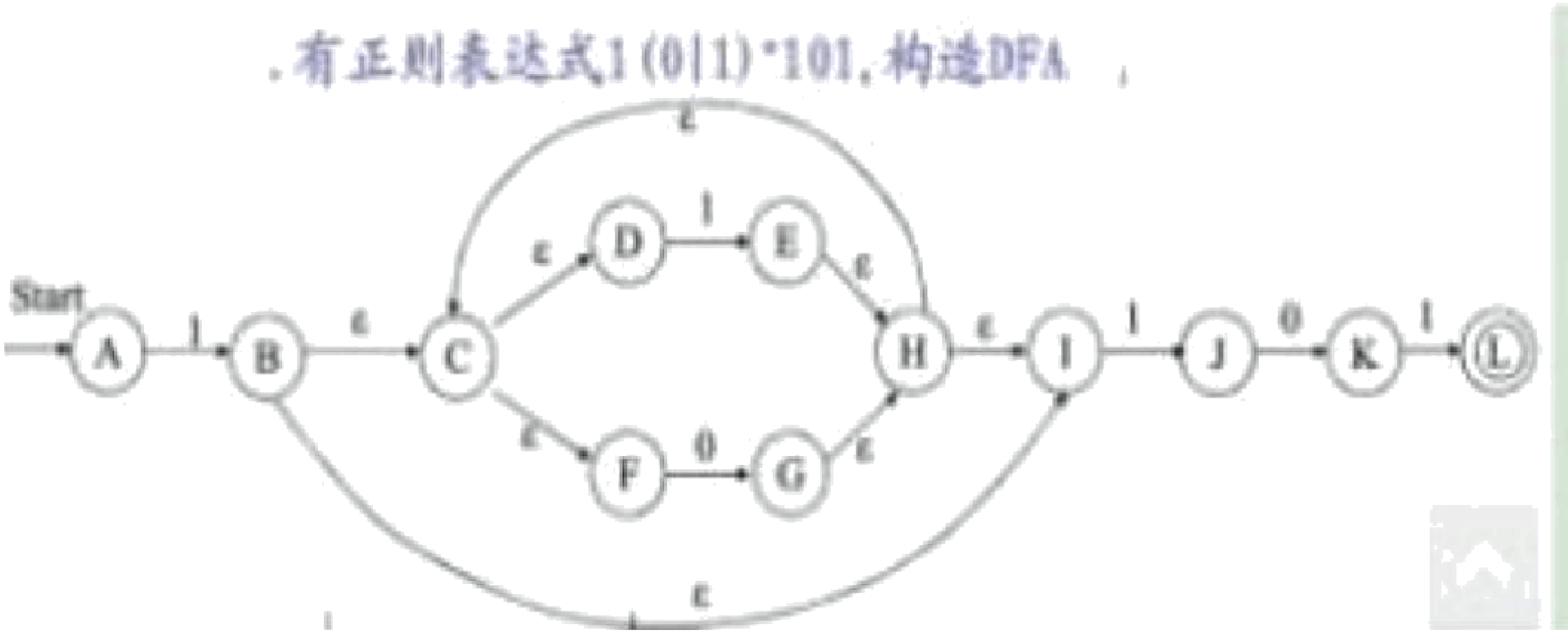
- （1）与1（0|1）*|0对应的NFA为:





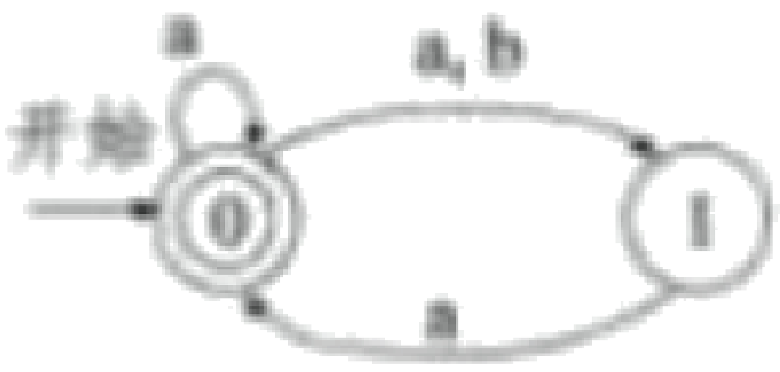
（2）1（1010*|1（010）*1）*0



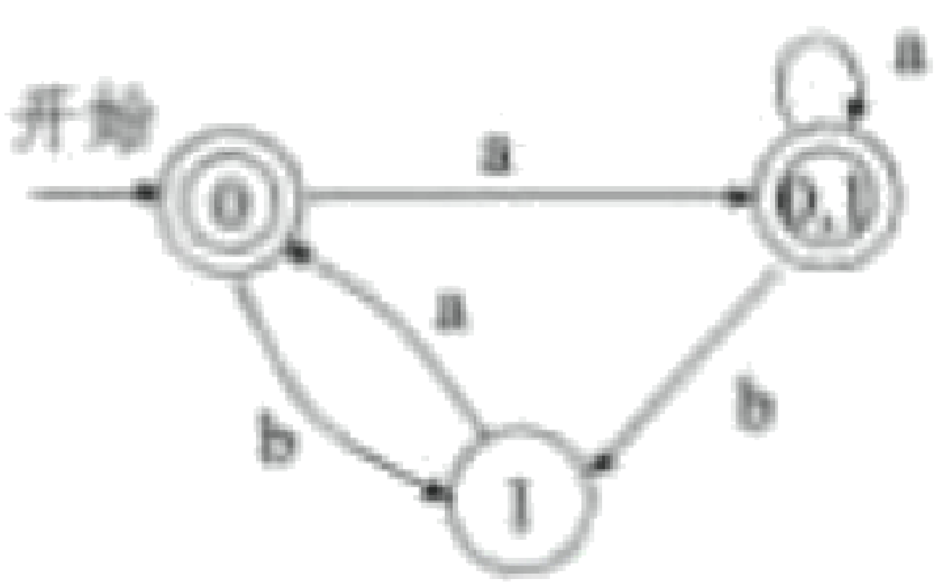


4.

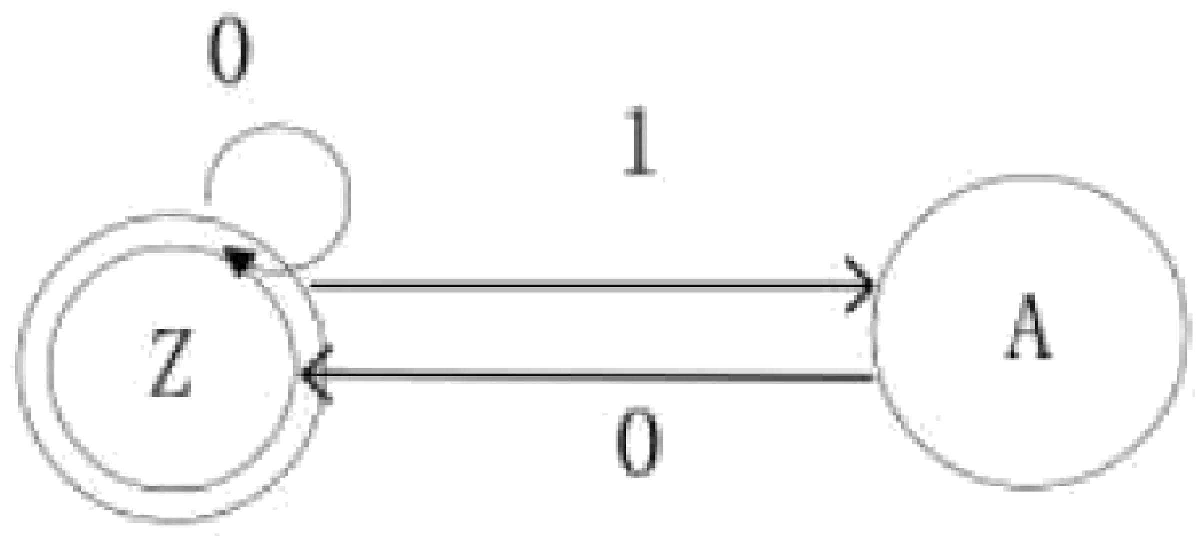
. 把图3.24的（a）和（b）分别确定化



	a	b
{0}	{0,1}	{1}
{0,1}	{0,1}	{1}
{1}	{0}	--



5. 结构一 DFA，它接受 Σ={0,1}上全部知足以下条件的字符串： 每个 1 都有 0 直接跟在右侧。



第四章

练习 4.2

2. 有文法 G[A]:

A::= (B) | dBe

B::= c | Bc

试设计自顶向下的语法剖析程序。

解：除去左递归：

A::= (B) | dBe

B::= c{c}

```
procedure B;

if CLASS = 'c' then

begin

nextsym;

while CLASS = 'c' do nextsym;

end;
```

```
else
error;
program        G;
begin
nextsym;
A;
end;
procedure A;
if CLASS = '(' then
begin
nextsym;
B;
if CLASS = ')' then
nextsym ；
else
error
end ；
else
if CLASS = 'd' then
begin
nextsym;
B;
```

if CLASS = 'e' then

nextsym;

else

error;

end;

else

error;

3. 有文法 G[Z]:　　　　Z::= AcB| Bd

A::=AaB|c

B::= aA|a

(1) 试求各选择（候选式）的 FIRST 会合；

(2) 该文法的自顶向下的语法剖析程序能否要编成递归子程序？为何？

(3) 试用递归降落剖析法设计其语法剖析程序。

解: (1) FIRST(B)={a} FIRST(A)={c} FIRST(Z)={a,c}

FIRST(AcB)={c}　　　　FIRST(Bd) ={a}

FIRST(AaB)={c}　　　　FIRST(c) ={c}

FIRST(aA) ={a}　　　　FIRST(a) ={a}

(2) 要编成递归子程序，由于文法拥有递归性

(3) 改写文法：

Z::= AcB| Bd

A::= c{aB}

B::= a[A]

```
void Z()
{
    if( sym =='c' )
    {
        A();
        if( sym == 'c')
        {
            getsym();
            B();
        }
        else
            error();
    }
    else if( sym == 'a')
    {
        B();
        if( sym != 'd')
            error();
        else
            getsym();
    }
    else
        error();
}
```

Z::= AcB| Bd
A::= c{aB}
B::= a[A]

```
void A()
{
    if( sym == 'c')
    {
        getsym();
        while( sym == 'a')
        {
            getsym();
            B();
        }
    }
    else
        error();
}
```

```
void B()
{
    if( sym!='a' )
        error();
    else
    {
        getsym();
        if( sym == 'c')
            A();
    }
}
```

```
void main()
{
    getsym();
    Z();
}
```

练习 4.3

1． 对下边的文法 G[E]: E → TE ‘

E' → + E | ε

T→ FT'

T' → T | ε

F→PF'

F' → * F' | ε

P →(E)| a | b | ∧

(1) 计算这个文法的每个非终结符号的 FIRST 和 FOLLOW 会合

(2) 证明这个文法是 LL(1)的

(3) 结构它的展望剖析表

解： (1)

FIRST(E) = {(, a, b, ∧ }

FOLLOW(E) = {#,) }

FIRST(E') = { +, ε }

FOLLOW(E') = {#,)}

FIRST(T) = { (, a, b, ∧}

FOLLOW(T) = { #,),+}

FIRST(T') = { (, a, b, ∧, ε }

FOLLOW(T') = {#,),+ }

FIRST(F) = { (, a, b, ∧}

FOLLOW(F) = { (, a, b, ∧,#,),+}

FIRST(F') = { *, ε }

FOLLOW(F') = { (, a, b, ∧,#,),+}

FIRST(P) = {(, a, b, ∧ }

FOLLOW(P) = {*,(, a, b, ∧,#,),+ }

(2) 证明：

FIRST(+E) ∩FIRST(ε) = {+}∩{ ε } = ∮

FIRST(+E) ∩FOLLOW(E') ={+}∩{#,)} = ∮

FIRST(T) ∩FIRST(ε)={ (, a, b, ∧}∩{ ε }= ∮

FIRST(T) ∩FOLLOW(T') ={ (, a, b, ∧}∩{#,),+ }= ∮

FIRST(*F') ∩FIRST(ε)= {*} ∩{ ε } = ∮

FIRST(*F') ∩FOLLOW(F') = {*}∩{ (, a, b, ∧,#,),+} = ∮

FIRST((E)) ∩FIRST(a) ∩ FIRST(b) ∩FIRST(∧)= ∮

所以此文法是 LL(1)文法

（3）分析表

	+	*	(	)	a	b	∧	#
E			E → TE'		E → TE'	E → TE'	E → TE'	
E'	E' → + E			E' →ε				E' → ε
T			T → FT'		T → FT'	T → FT'	T → FT'	
T'	T' →ε		T' → T	T' → ε	T' → T	T' → T	T' → T	T'→ ε
F			F → PF'		F → PF'	F → PF'	F → PF'	
F'	F' →ε	F' →* F'	F' →ε	F' →ε	F' →ε	F' →ε	F' →ε	F' →ε
P			P →(E)		P →a	P →b	P →∧	

2. 关于文法 G[S]：S → aABbcd | ε

A →ASd | ε

B → SAh | eC | ε

C → Sf | Cg | ε

D → aBD | ε

（1) 对每一个非终结符号，结构 FOLLOW 集；

（2) 对每一产生式的各侯选式，结构 FIRST 集；

（3) 指出此文法能否为 LL（1）文法。

解：（1）FIRST(S) = {a, ε }

FIRST(A) = {a,d, ε }

FIRST(B) = {a,d,h,e, ε }

FIRST(C) = {a,f,g, ε}

FIRST(D) = {a, ε}

(2) FIRST(aABbcd) = {a}

FIRST(ε) = { ε}

FIRST(ASd) = {a,d}

FIRST(ε) = { ε}

FIRST(SAh) = {a,d,h}

FIRST(eC) = {e}

FIRST(ε) = { ε}

FIRST(Sf) = {a,f}

FIRST(Cg) = {a,f,g}

FIRST(ε) = { ε}

(3) 不是 LL(1)文法，因

FIRST(Sf) ∩ FIRST(Cg) = {a,f} ∩ {a,f,g}≠∮

或 FOLLOW(S) ∩FIRST(aABbcd) ＝{d,a,f,h#} ∩{a}≠∮

或 FOLLOW(A)∩FIRST(ASd) ＝{ a,h,e,b,d } ∩{ a,d }≠∮

或 FOLLOW(B)∩FIRST(SAh) ＝{ a,b } ∩{ a,d,h }≠∮ 或

FOLLOW(C)∩FIRST(Sf) ＝{ g,a,b } ∩{ a,f }≠∮

或 FOLLOW(C) ∩FIRST(Cg) ＝{ g,a,b } ∩{ a,f ,g}≠ ∮

6. 一个文法 G 是 LL(1)的必需与充足条件是什么？试证明之。

充要条件是：关于 G 的每一个非终结符 A 的任何两条不一样规则 A::= α

| β, 有：

(1) FIRST(α)∩FIRST(β)= ∮

(2) 倘若 β =*=> ε , 则 FIRST(α)∩FOLLOW(A)= ∮

证明：

充足性：条件

（证明：

充足性：条件（ 1）（2）建立

反证：若剖析表中存在多重进口，

即）建立

反证：若剖析表中存在多重进口，即

M[B,a] = { B::= α 1, B::= α 2} ,表示 FIRST(α 1)∩FIRST(α 2) ≠ ∮

或 M[B,a] = { B::= α 1, B::= α 2}，此中 α 2= ε 或 α 2=+=> ε

表示 FIRST(α 1)∩FOLLOW(B)≠ ∮

与条件（ 1）（2）矛

盾。必需性：文法是

）矛盾。

必需性：文法是 LL(1)文法，即剖析表中不含多重进口

若条件

文法，即剖析表中不含多重进口

若条件 (1) 不建立，即存在某非终结符 B 的两条规则 B::= α1| α2，

FIRST(α1)∩FIRST(α2) ≠ ∮

则对随意的 a ∈FIRST(α1)∩FIRST(α2)，有

M[B,a] = { B::= α1, B::= α2},矛盾

若条件（

矛盾

若条件（ 2）不建立，即存在某非终结符 B 的两条规则 B::= α1| α2，

α2=*=> ε

有 FIRST(α1)∩FOLLOW(B)≠∮

则对随意的 a ∈FIRST(α1)∩FOLLOW (B)，有

M[B,a] = { B::= α1, B::= α2},矛盾

练习 4.4

4. 有文法 G[E]: E::= E+T | T
T::= T*F | F
F::= (E) | i
列出下述句型的短语和素短语： E、 T、 i 、 T*F、F*F、i*F 、F* i 、F+F+F

解: 句型	短语	素短语
E		
T	T	
i	i	i
T*F	T*F	T*F
F*F	F，F*F	F*F
i*F	i，i*F	i
F* i	F，i，F* i	i
F+F+F	F,F,F,F+F, F+F+F	F+F

练习 4.5

1. 考虑拥有以下规则的文法

S→E#　E →T|E+T　T →P| P ↑T　P →F|P*F　F →i|(E)

(a)　以下句型的最右推导步骤中，其活前缀的会合是什么？

(1) E+i*i#

(2) E+P ↑(i ＋i) ＃

(b)

为以下输入串结构最右推导的逆：

(1) i+i*i#

(2) i+i ↑(i+i)#

解：(a) （1）句柄为 i，所以活前缀会合为： E，E，E＋i

（2）句柄为 i，所以活前缀会合为： E，E+, E+P, E+P ↑, E+P ↑(, E+P ↑(I

(b)　（1）

i+i*i#　<=

F+i*i#　<=

P + i*i#　<=

T+i*i#　<=

E+i*i#　<=

E+F*i#　<=

E+P*i#　<=

E+P*F#　<=

E+P#　<=

E+T#　<=

E#　　<=

S

练习 4.6

1. 给定拥有以下产生式的文法

S→E#　E →E-T|T　T　→F|F ↑ T F　→i|(E)

试求以下活前缀的有效项目集：

(a) F　↑ (b) E-(　　(c) E-T

解：(a)E　↑

{ T→.F　T　→F ↑ .T　T　→.F　↑ T　F　→.i　F　→.(E) }

(b)E-(

{F→ (.E), E　→.E-T, E　→.T, T　→.F, T　→.F　↑ T, T　→.i, T

→ .(E)}

(c)E - T

{E→E-T. }

练习 4.7

1. 给定以下产生式的文法：

S→E　E →T|E+T T　→P|T*P P　→F|F　↑ P F　→i|(E)

(1)　为该文法结构 SLR（1）剖析表。

状态	S	E	T	P	F	+	*	↑	i	(	)	#
C0		C1	C2	C3	C4				S5	S6		
C1						S7						A
C2						r1	S8				r1	r1
C3						r3	r3				r3	r3
C4						r5	r5	S9			r5	r5
C5						r7	r7	r7			r7	r7
C6		C10	C2	C3	C4				S5	S6		
C7			C11	C3	C4				S5	S6		
C8				C12	C4				S5	S6		
C9				C13	C4				S5	S6		
C10						S7						
C11						r2	S8				r2	r2
C12						r4	r4				r4	r4
C13						r6	r6				r6	r6
C14						r8	r8	r8			r8	r8

第五章

练习 5

3. 以下非分程序结构语言的程序段，画出编译该程序段时将生成的有序符号表。

BLOCK

REAL X,Y,Z1,Z2,Z3;

INTEGER I,J,K,LASTI;

STRING LIST-OF-NAMES;

LOGICAL ENTRY-ON EXIT-OFF;

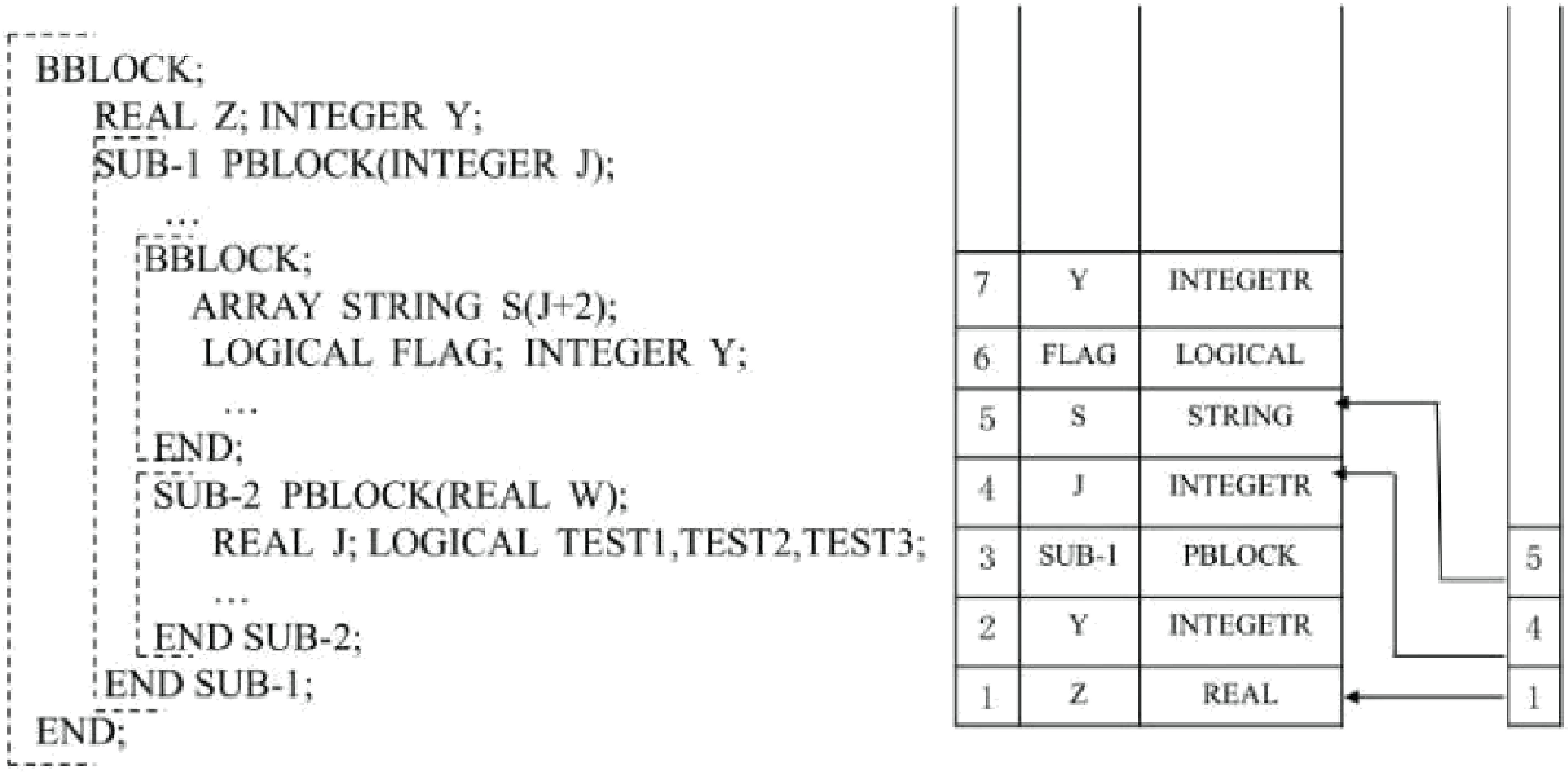
ARRAY REAL VAL(20);

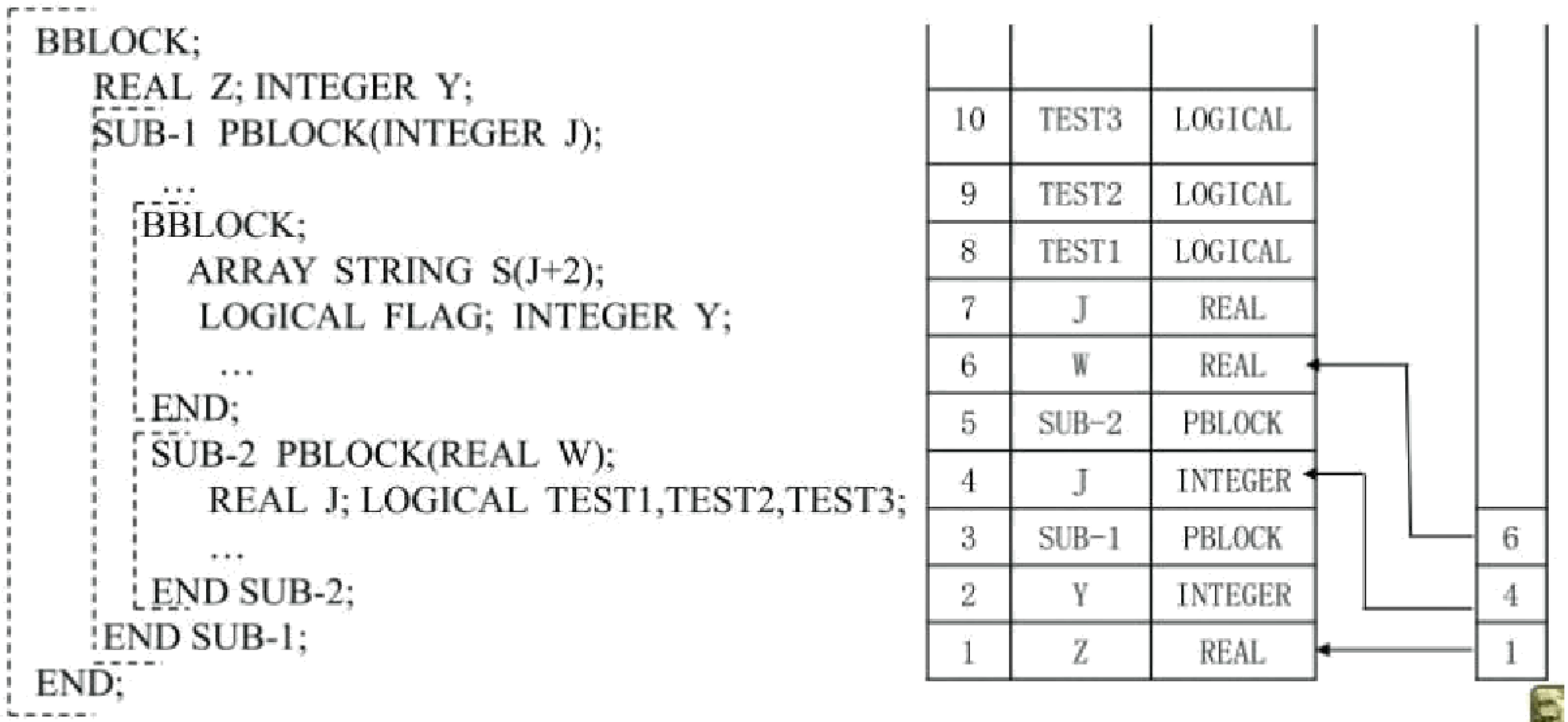
ARRAY INTEGER MIN-VAL-IND(20);

END OF BLOCK;

变量名	类型	维数
ENTRY-ON	LOGICAL	0
EXIT-OFF	LOGICAL	0
I	INTEGER	0
J	INTEGER	0
K	INTEGER	0
LASTI	INTEGER	0
LIST-OF-NAMES	STRING	0
MIN-VAL-IND	INTEGER	1
VAL	REAL	1
X	REAL	0
Y	REAL	0
Z1	REAL	0
Z2	REAL	0
Z3	REAL	0

5. 画出下边的分程序结构的程序段当程序段 3 和 4 的编译马上达成从前的栈式符号表的图形（包含有效部分和无效部分）。

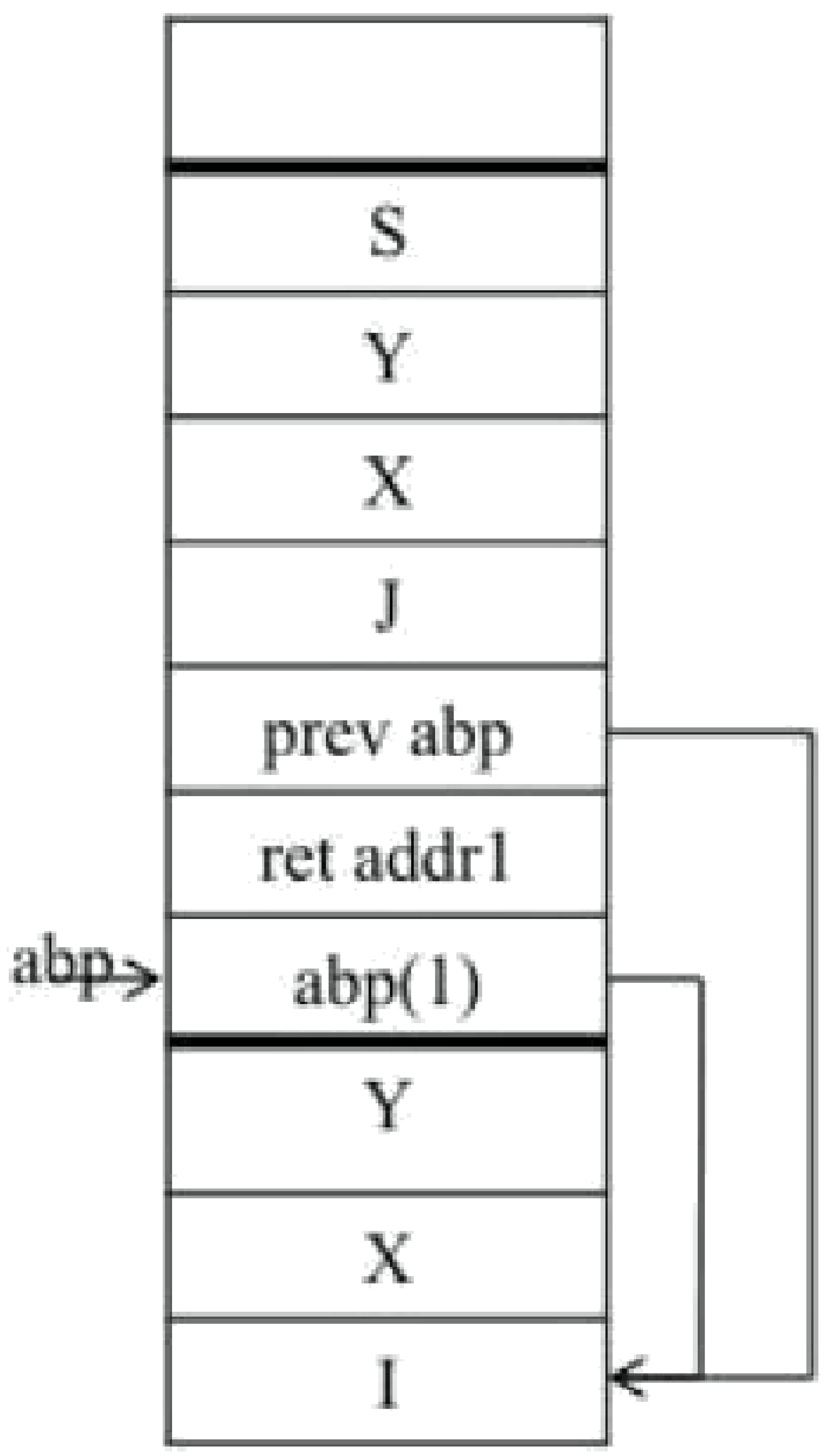


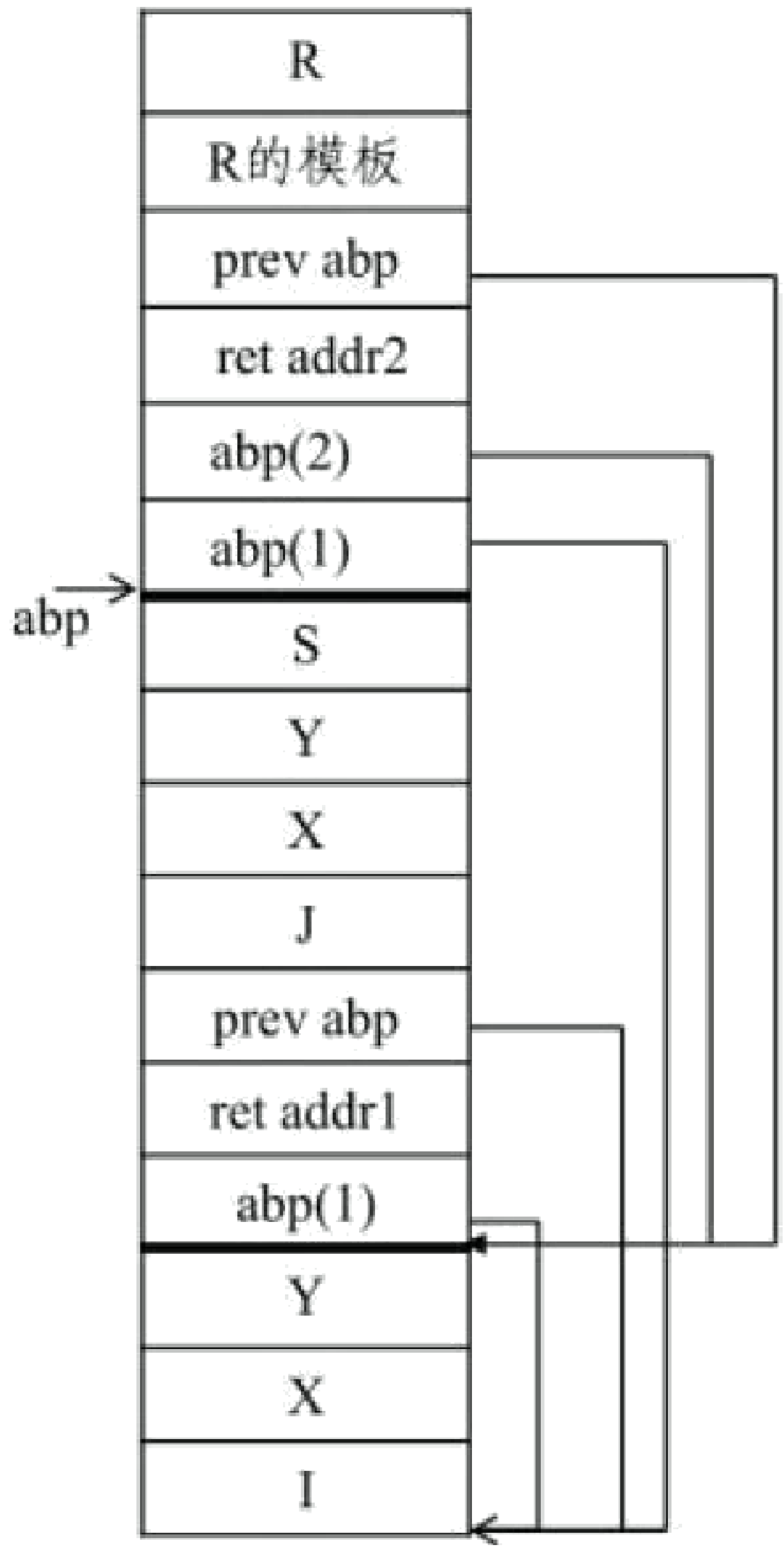


第六章

练习 6.2

2　考虑下边的类 ALGOL 程序，画出当程序履行到①和②时，运转栈内容的图像。


S
Y
X
J
prev abp
ret addr1
abp
abp(1)
Y
X
I



第七章

练习 7

2. 将下边的语句 A:= (B+C) ↑E+(B+C)*F 变换成三元式，间接三元式和四元式序列。

三元式

(1) + B, C
(2) ↑ (1),E
(3) + B, C
(4) * (3),F
(5) + (2),(4)
(6) := A, (5)

四元式

(1) + B, C, T1
(2) ↑ T1, E, T2
(3) + B, C T3
(4) * T3, F, T4
(5) + T2, T4, T5
(6) := A, T5

间接三元式

操作

1、(1)
2、(2)
3、(1)
4、(3)
5、(4)
6、(5)

(1) + B, C
(2) ↑ (1), E
(3) * (1), F
(4) + (2), (3)
(5) := A, (4)

第九章

练习 9.1

1. 试分别结构一个符号串翻译文法， 它将由一般中缀表达式文法所定义的中缀表达式翻译成波兰前缀表达式和波兰后缀表达式 .

解：翻译为波兰前缀表达式

的文法为：

E->@+E+T

E->T

T->@*T*F

T->F

F->(E)

F->@ii

翻译为波兰后缀表达式

的文法为：

E->E+T@+

E->T

T->T*F@*

T->F

F->(E)

F->i@i

2． 结构一符号串翻译文法， 它将接受由 0 和 1 构成的随意输入符号串，并产生下边的输出符号串：

(a) 输入符号串倒置 (c) 输入符号串自己。

（a）S->0S@0 或 S->@0S0

S->1S@1 S->@1S1

S-> ε

S-> ε

（ c ）E->0@0E

E->1@1E E-> ε

3. 以下的符号串翻译文法能做什么？

<s>->@C EN @HI GL @N I @E S @SE H

解：能够辨别终结符号串 ENGLISH，并输出符号串 CHINESE。

4. 有特别的翻译文法产生的两个活动序列是：

@x@yb@z 和@qa@x@yb@z@x@x@yb@z@y

由这个翻译文法删掉诸动作符号获得的输入文法是：

<S>->a<S><S>

<S>->b

这是个什么翻译文法？

解：其文法为：

<S>->@x@yb@z

<S>->@qa<S>@x<S>@y

5. 下边给出带有开始符号 <S> 的翻译文法，试列出属于这个文法所定

义的

语法制导翻译的全部对偶。

的翻译文法，试列出属于这个文法所定义的

语法制导翻译的全部对偶。

<S>-><A>xc<B>@y

<S>->@yd@xc@zb

<A>-><B>a@y

<A>->d

<B>->b@x

解：

(1)dcb @y@x@z

(2)dxcb @x@y

(3)baxcb @x@y@x@y