编译原理及实现课后习题解答

2.1　设字母表 A={a}，符号串 x=aaa，写出下列符号串及其长度：x^0,xx,x^5 以及 A^+和 A^*.

$x_0=(aaa)_0=\varepsilon$　　$|x_0|=0$

xx=aaaaaa　　|xx|=6

x_5=aaaaaaaaaaaaaaa　　$|x_5|=15$

$A^+=A_1\cup A_2\cup \cdots . \cup A_n \cup \ldots=\{a,aa,aaa,aaaa,aaaaa\ldots\}$

$A^*=A_0\cup A_1\cup A_2\cup \cdots . \cup A_n \cup \cdots=\{\varepsilon,a,aa,aaa,aaaa,aaaaa\ldots\}$

2.2　令∑={a，b，c}，又令 x=abc，y=b，z=aab，写出如下符号串及它们的长度：xy，xyz，$(xy)^3$

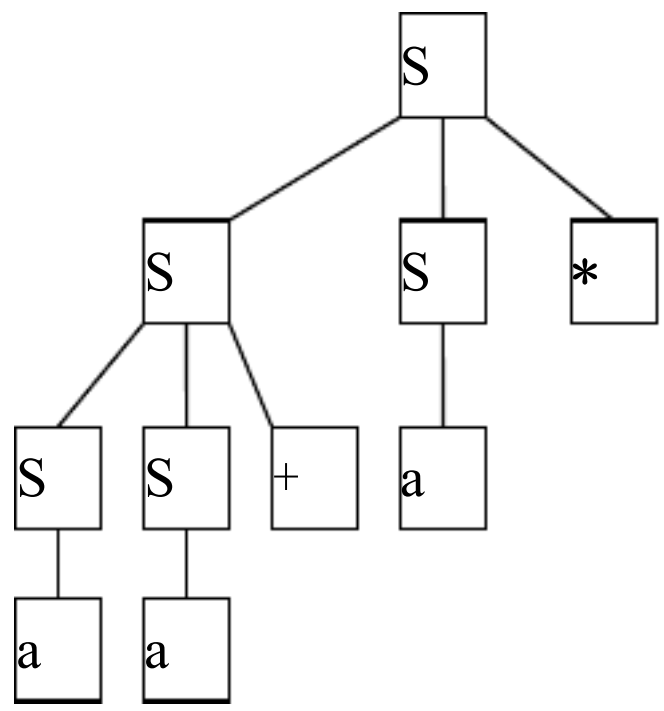
xy=abcb　　|xy|=4

xyz=abcbaab　　|xyz|=7

$(xy)_3=(abcb)_3$=abcbabcbabcb　　$|(xy)_3|=12$

2.3　设有文法 G[S]：S∷=SS*|SS+|a，写出符号串 aa+a* 规范推导，并构造语法树。

S => SS* => Sa* => SS+a* => Sa+a* => aa+a*



2.4　已知文法 G[Z]：Z::=U0｜V1 、 U::=Z1｜1 、 V::=Z0｜0 请写出全部由此文法描述的只含有四个符号的句子。

Z=>U0=>Z10=>U010=>1010

Z=>U0=>Z10=>V110=>0110

Z=>V1=>Z01=>U001=>1001

Z=>V1=>Z01=>V101=>0101

2.5　已知文法 G[S]：S::=AB　A::=aA｜ε B::=bBc| bc，写出该文法描述的语言。

A::=aA｜ε 描述的语言：　$\{a_n|n>=0\}$

B::=bBc｜bc 描述的语言：$\{b_nc_n|n>=1\}$

$L(G[S])=\{a_nb_mc_m|n>=0,m>=1\}$

2.6　已知文法 E::=T｜E+T｜E-T 、 T::=F｜T*F｜T/F 、 F::=(E)｜i，写出该文法的开始符号、终结符号集合V_T、非终结符号集合V_N。

开始符号：E

$V_t=\{+,\ -\ ,\ *\ (\ ,\ /\ ),\ ,\ i\}$

$V_n=\{E\ ,F\ ,T\}$

2.7　对 2.6题的文法，写出句型 T+T*F+i 的短语、简单短语以及句柄。

短语：T+T*F+i　　T+T*F

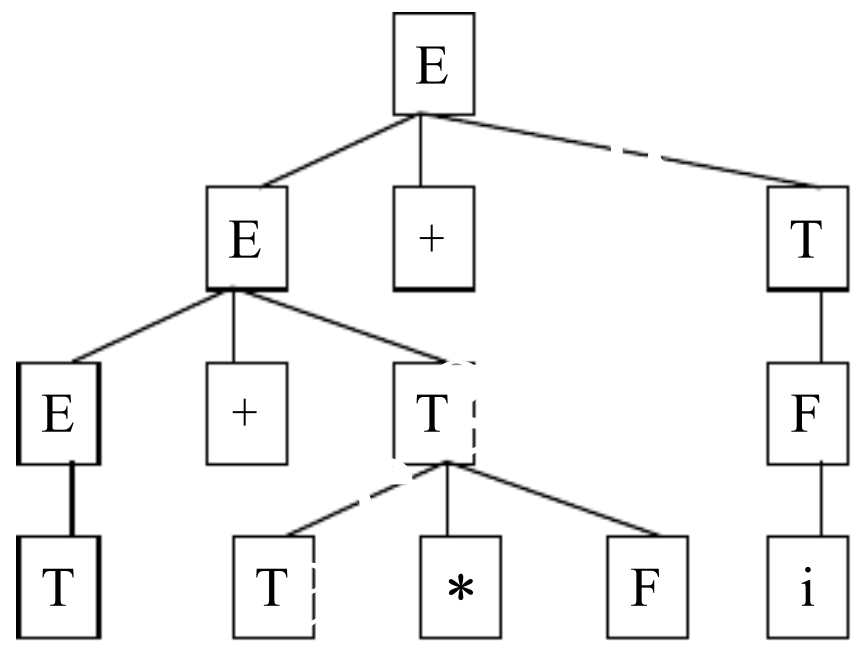
　　i　　i

　　T　　T*F

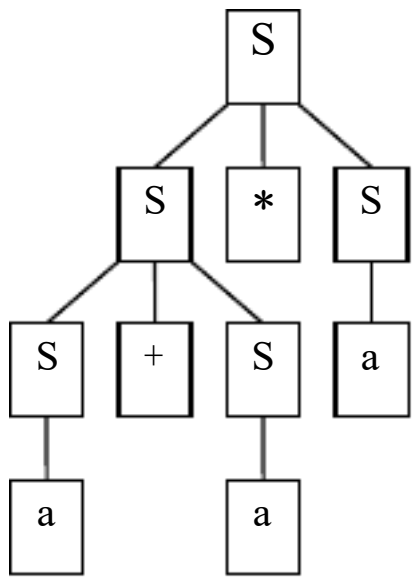
简单短语：i　　T*F

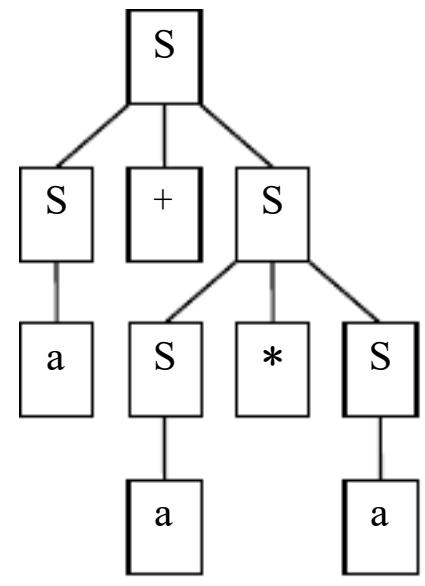
　　T

句柄：T



2.8　设有文法 G[S]：S∷=S*S|S+S|(S)|a，该文法是二义性文法吗？





根据所给文法推导出句子 a+a*a，画出了两棵不同的语法树，所以该文法是二义性文法。

2.9　写一文法，使其语言是奇正整数集合。

A::=1|3|5|7|9|NA

N::=0|1|2|3|4|5|6|7|8|9

2.10给出语言{$a^n b^m$|n,m≥1}的文法。

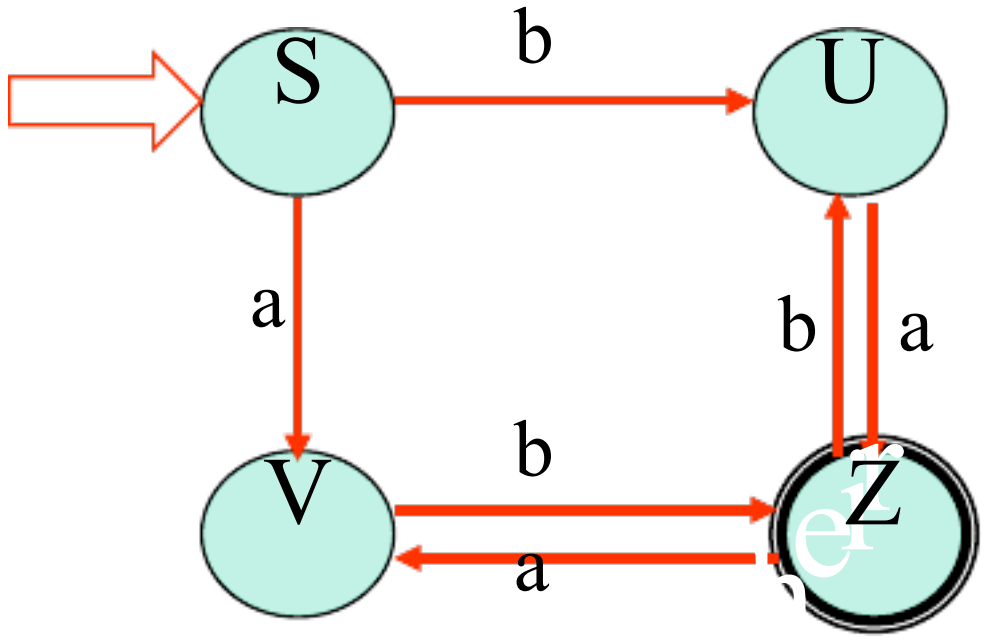
G[S]: S::=AB

　　A::=aA|a

B::=bB|b

3.1 有正则文法 G[Z]：Z::=Ua|Vb，U::=Zb|b，V::=Za|a ，画出该文法的状态图，并检查句子 abba 是否合法。

解：该文法的状态图如下：



句子 abba 合法。

3.2 状态图如图 3.35所示，S 为开始状态，Z 为终止状态。写出相应的正则文法以及 V，V_n 和 V_t。

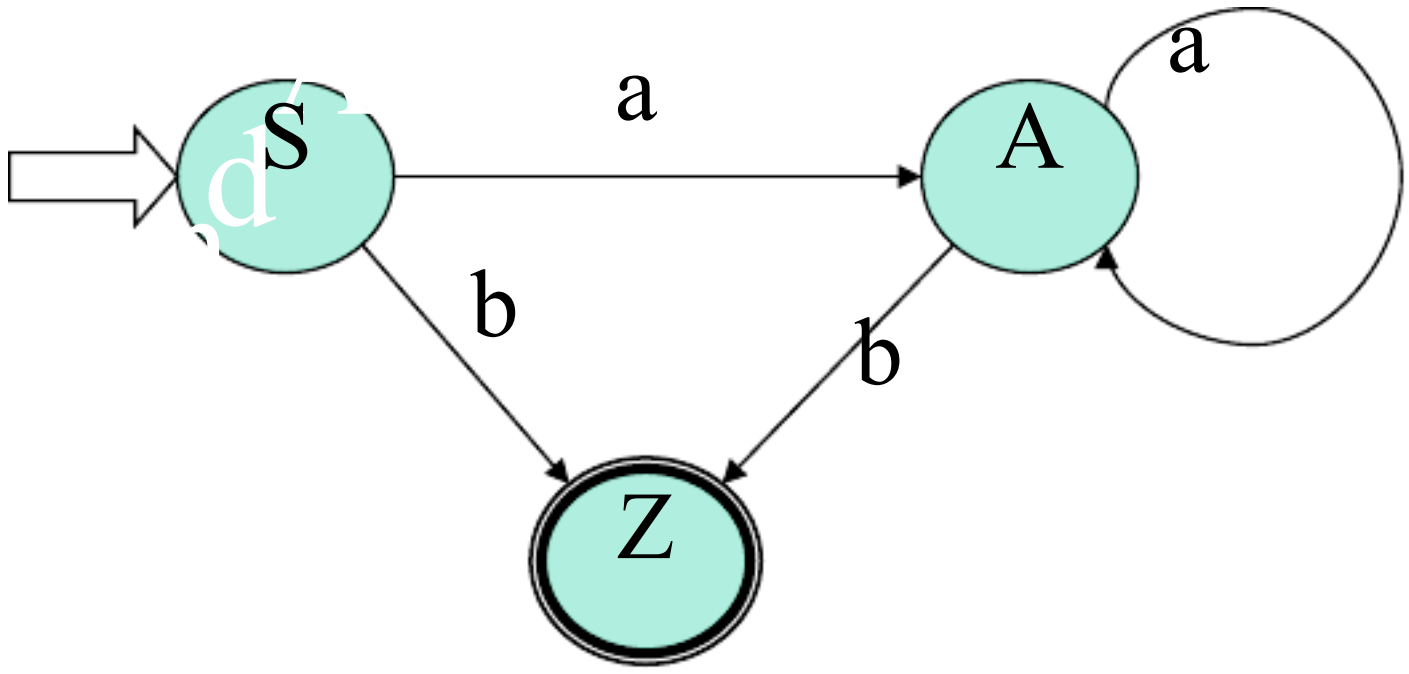


图3-35状态图

解：　左线性文法 G[Z]:　　右线性文法 G’[S]:

Z::=Ab|b　　S::=aA|b

A::=Aa|a　　A::=aA|b

V={Z,A,a,b}　　V={S,A,a,b}

V_n={Z,A}　　V_n={S,A}

V_t={a,b}　　V_t={a,b}

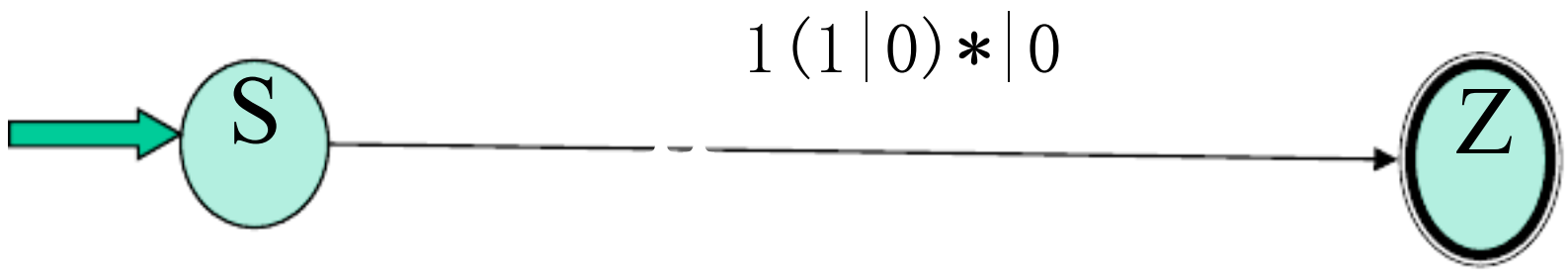
3.3　构造下列正则表达式相应的 NFA：

1(1|0)*|0

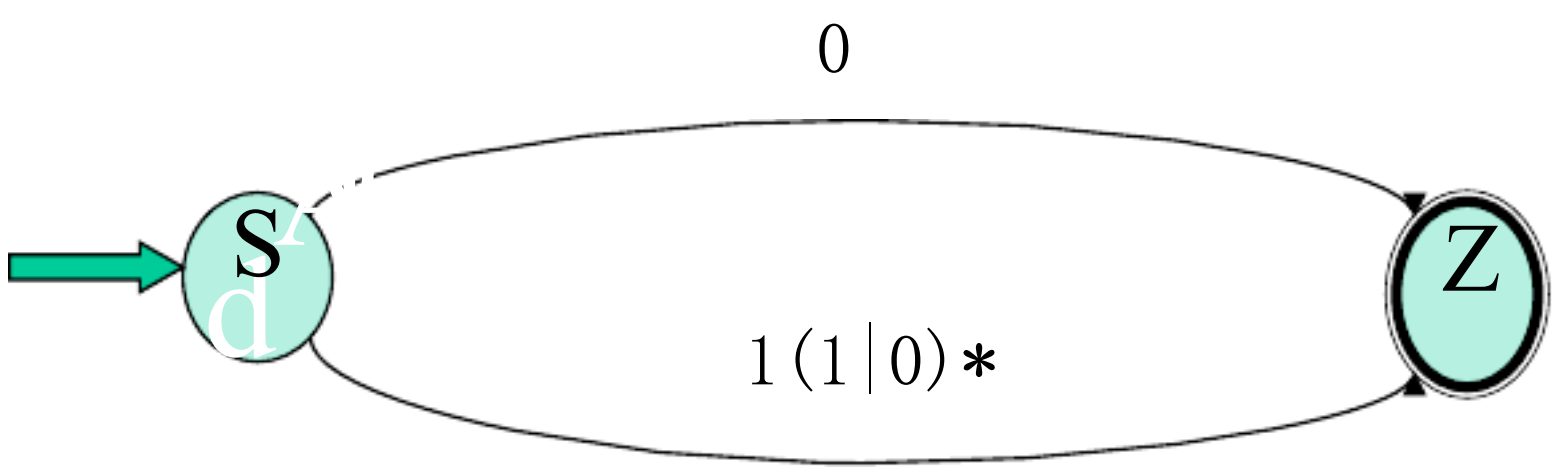
1(1010*|1(010)*1)*0

解：正则表达式：1(1|0)*|0

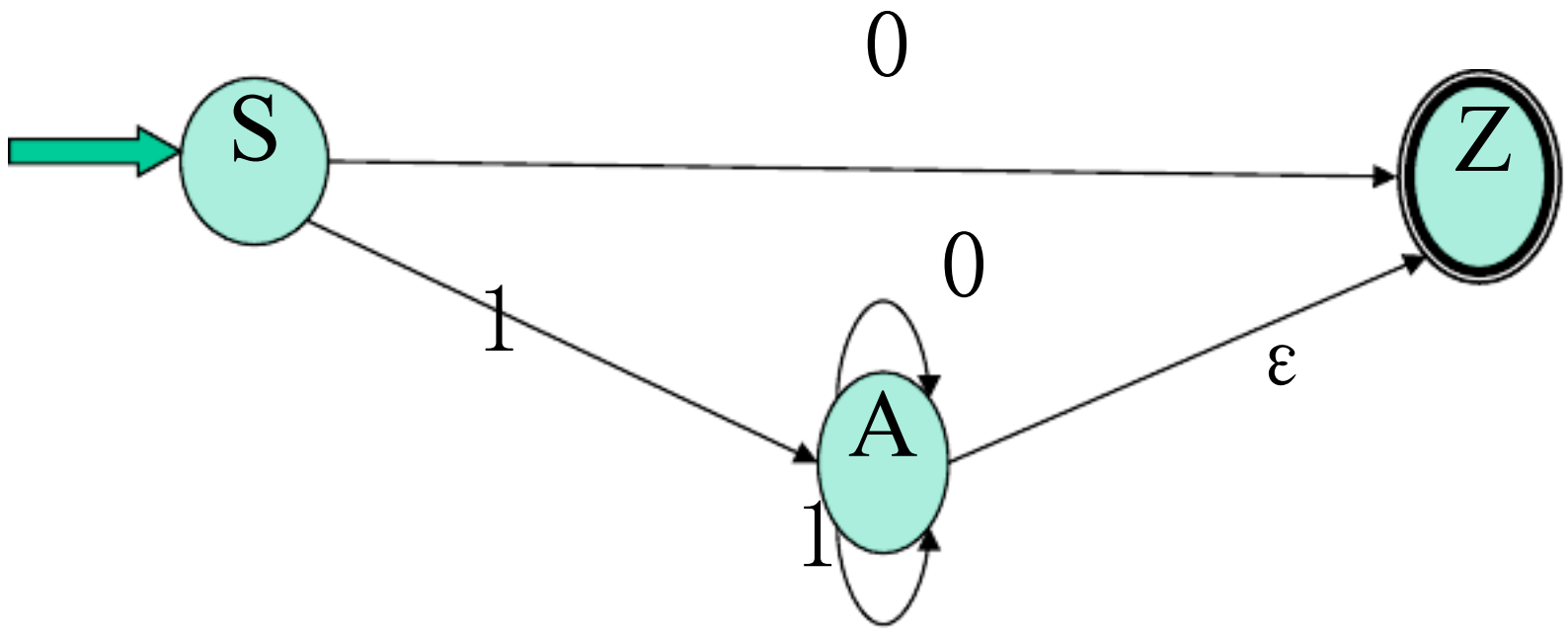
1、



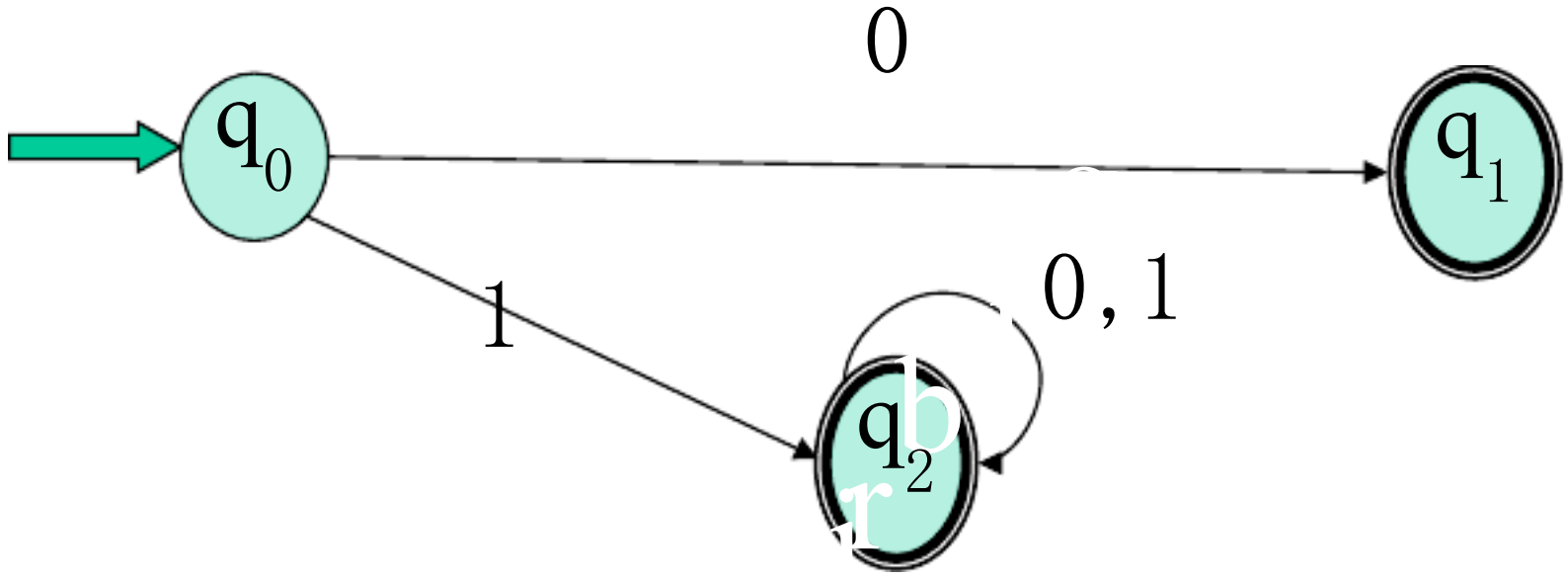
2、



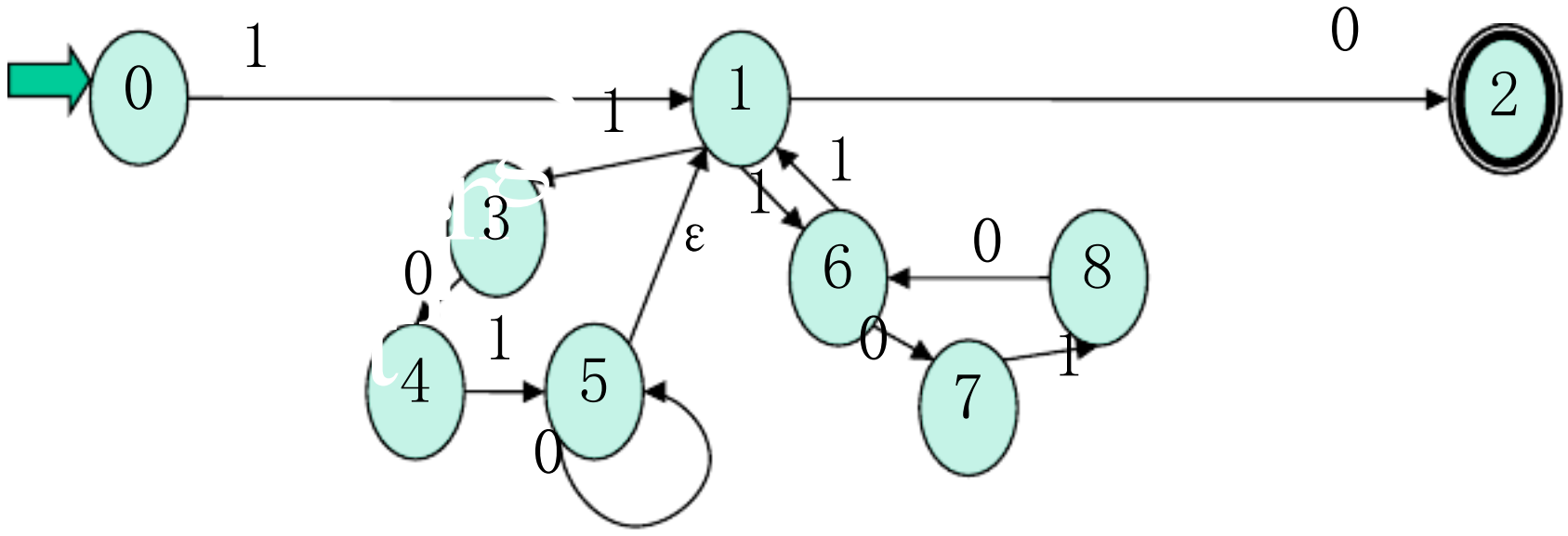
3、



4、



正则表达式：1(1010*|1(010)*1)*0



3.4 将图 3.36的 NFA M 确定化

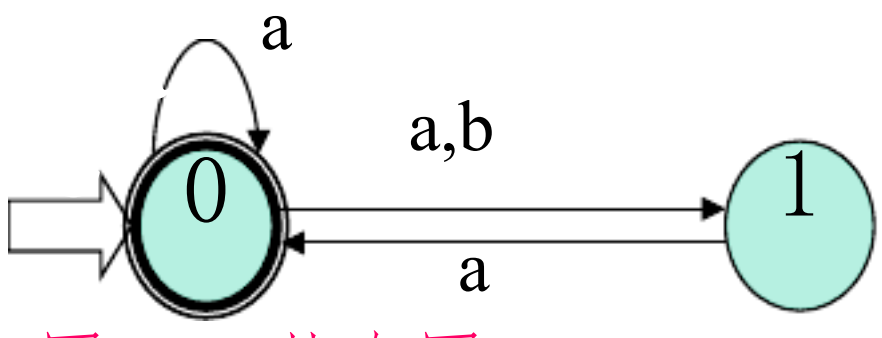
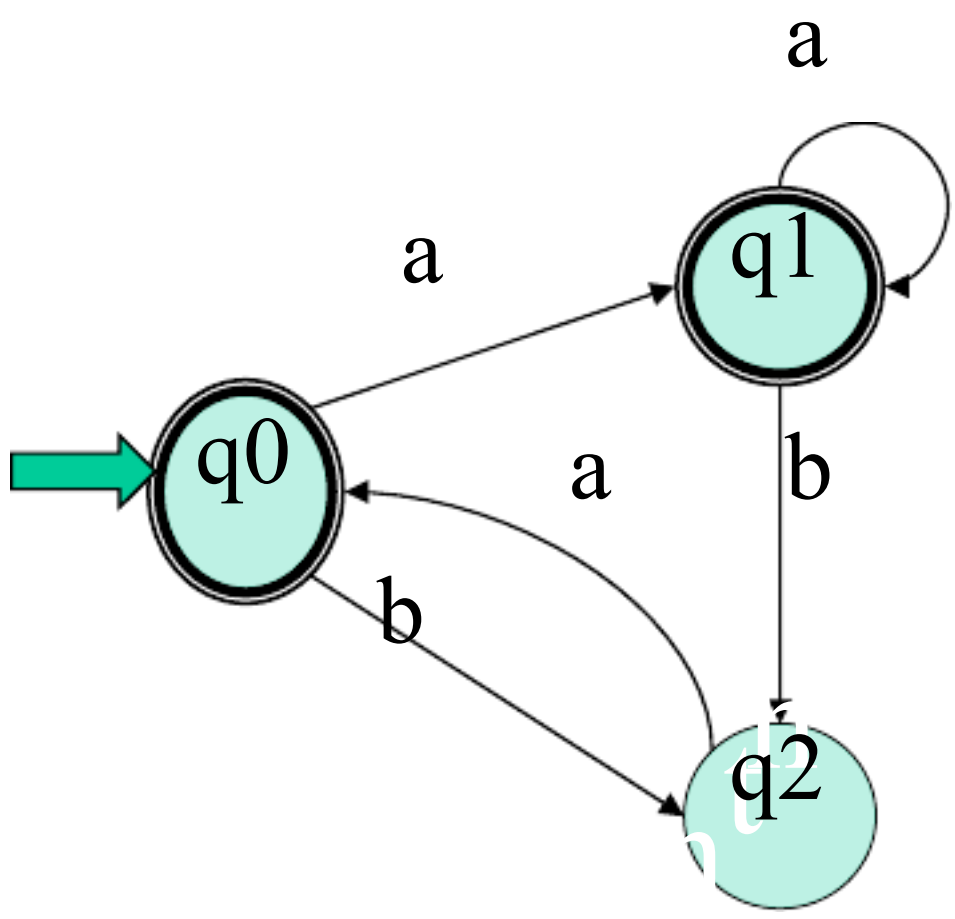


图3.36 状态图

解：

	a	b
q0={0}	{0, 1}	{1}
q1={0,1}	{0, 1}	{1}
q2={1}	{0}	Φ

DFA:



3.5 将图 3.37的 DFA 化简。

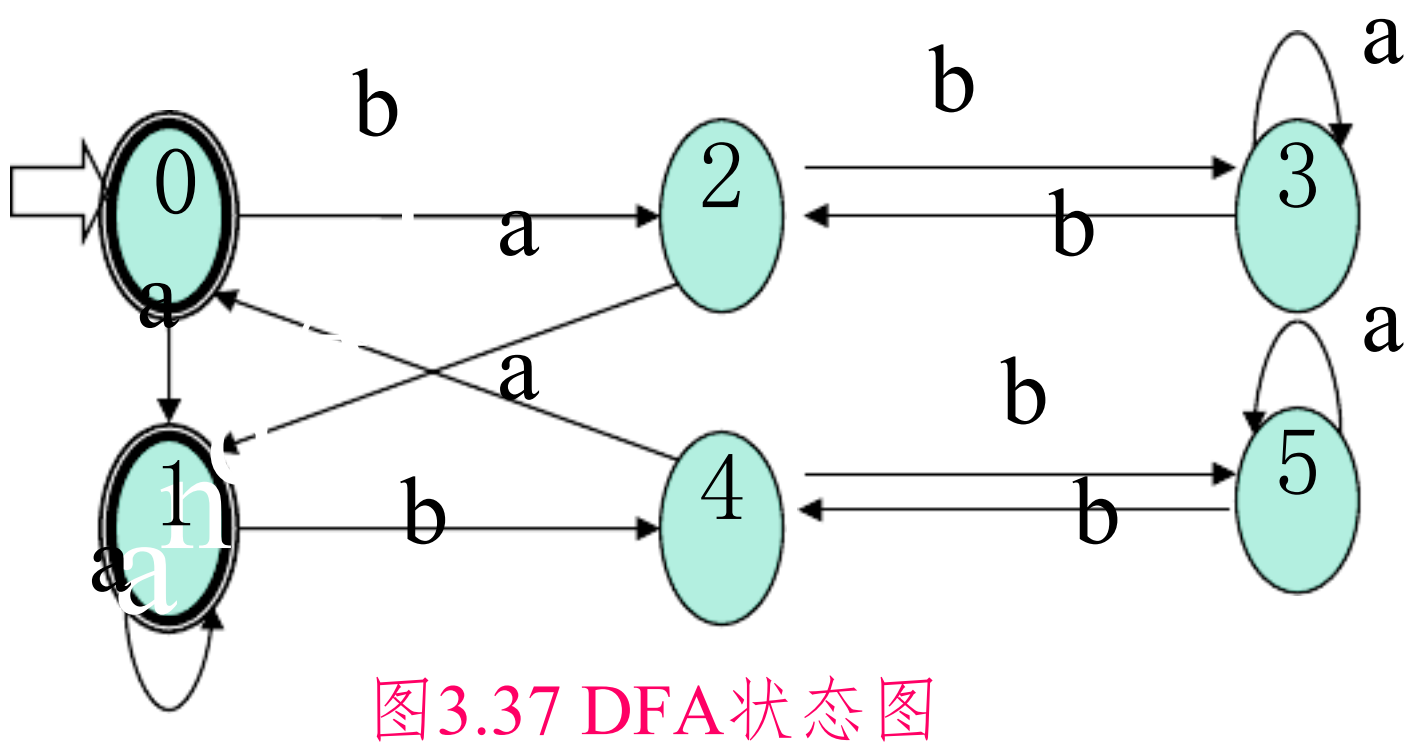


图3.37 DFA状态图

解：

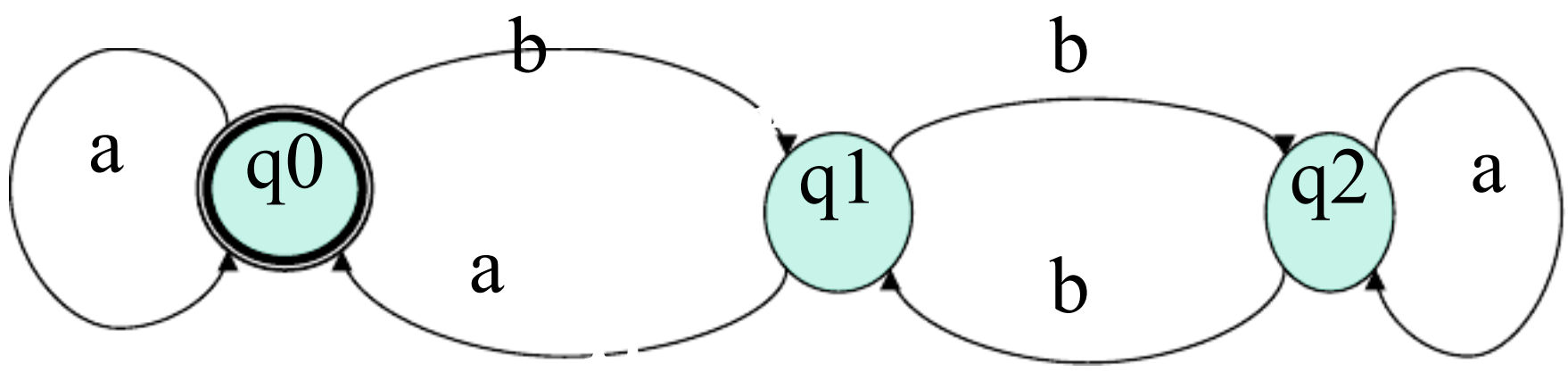
划分	a	b

{0,1}	{1}	{2,4}
{2,3,4,5}	{1,30,5}	{3,5,2,4}

划分	a	b
{0,1}	{1}	{2,4}
{2,4}	{0,1}	{3,5}
{3,5}	{3,5}	{2,4}

q0={0,1}　　q1={2,4}　　q2={3,5}

化简后的 DFA：

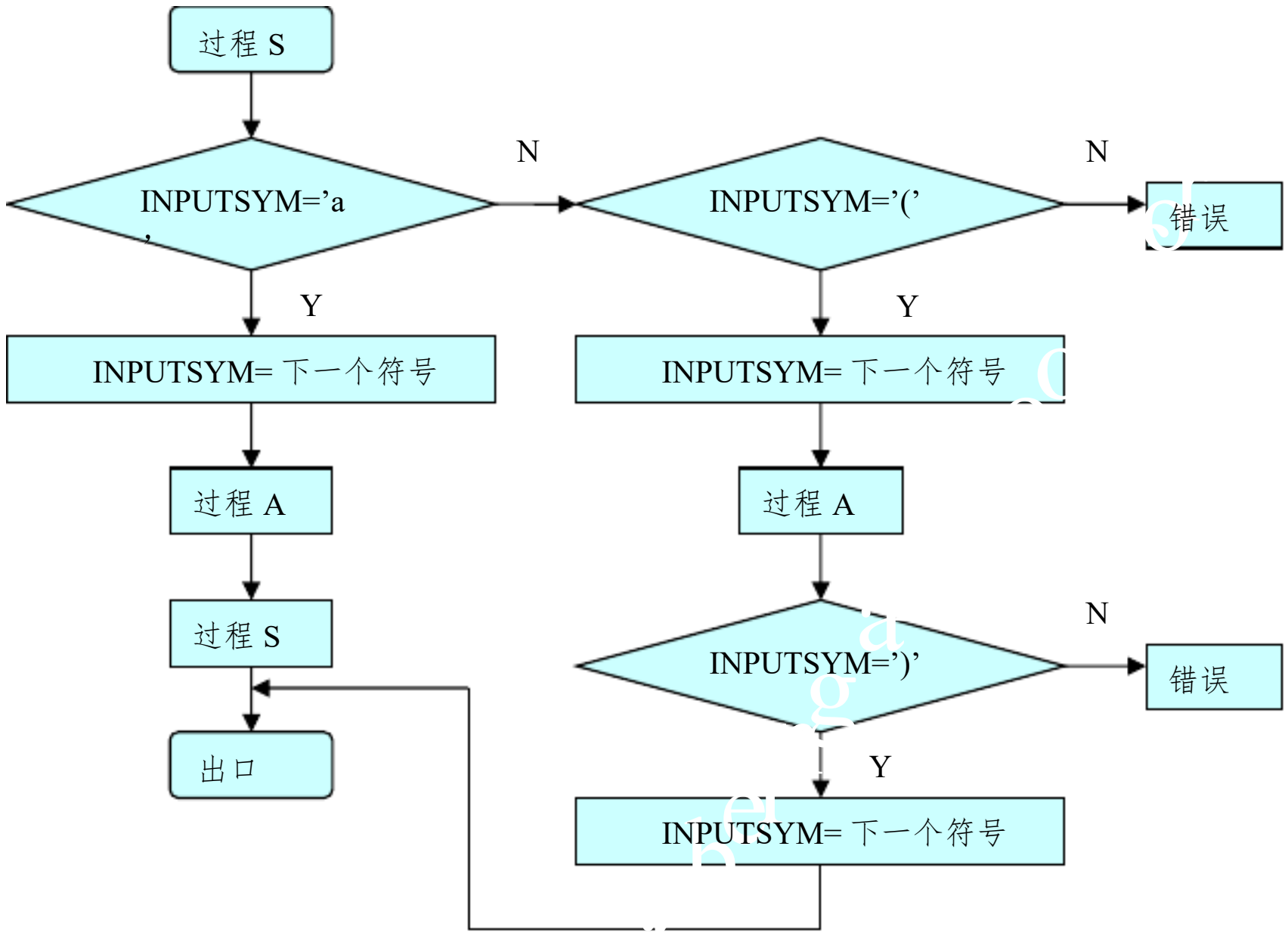


4.1 对下面文法，设计递归下降分析程序。

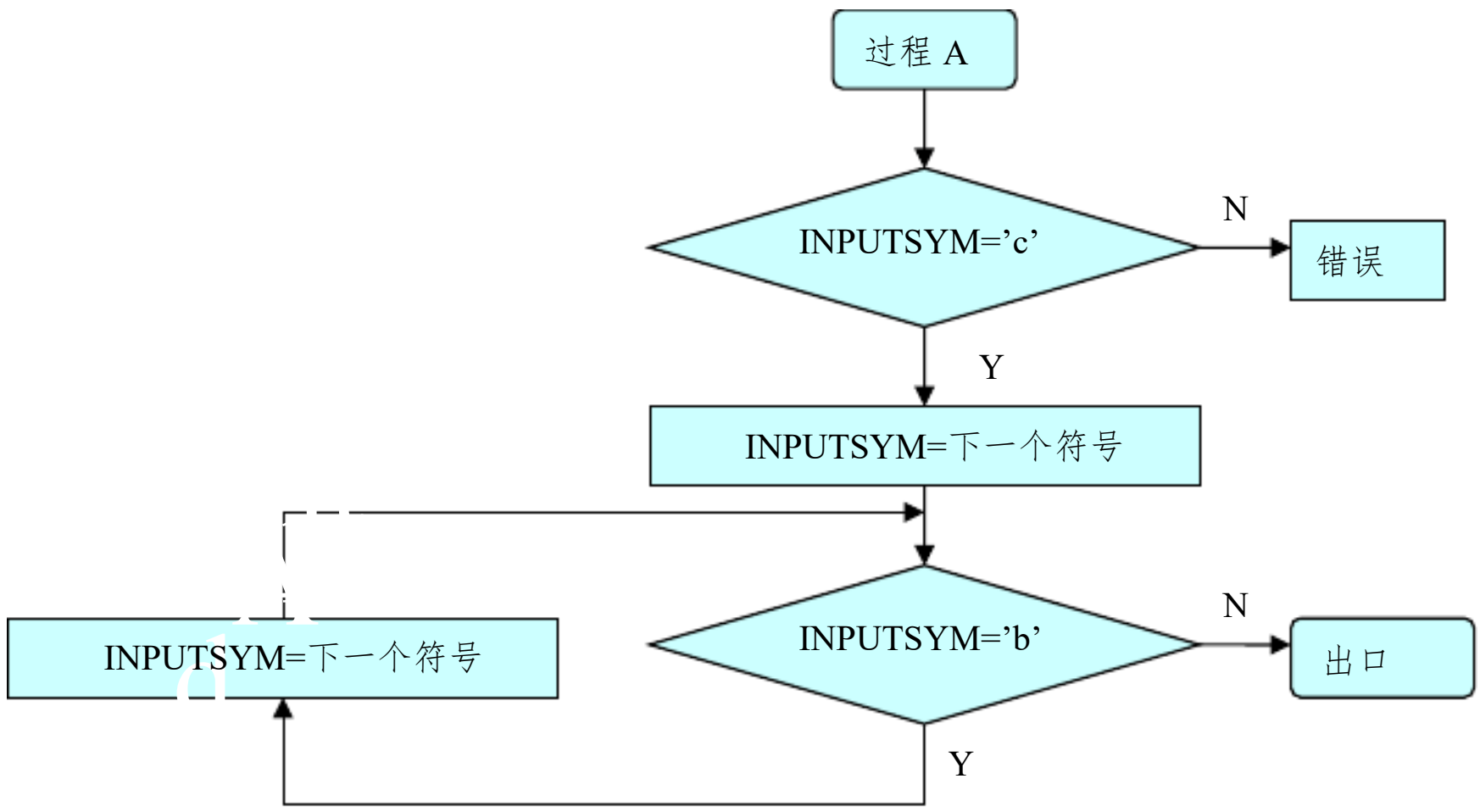
S→aAS|(A)，　A→Ab|c

解：首先将左递归去掉，将规则 A→Ab|c 改成 A→c{b}

非终结符号 S 的分析程序如下：



非终结符号 A 的分析程序如下：



4.2 设有文法 G[Z]：

Z∷=(A) ， A∷=a|Bb ， B∷=Aab

若采用递归下降分析方法，对此文法来说，在分析过程中，能否避免回溯？为什么？

解：若采用递归下降分析方法，对此文法来说，在分析过程中不能避免回朔。因为规则

=a|Bb 和规则 B∷=Aab 构成了间接左递归，不满足实现没有回溯的递归下降分析方法的条件（1）（书 P67），且规则 A: := a|Bb，FIRST(a)={a}，FIRST(Bb)={a}，即此规则候选式的首符号集有相交，不满足实现没有回溯的递归下降分析方法的条件（2）（书 P67），在分析过程中，将造成回溯。

改写文法可避免回溯：

将规则 B∷=Aab 代入规则 A∷=a|Bb 得：A∷=a|Aabb，再转换成：A∷=a{abb}，可避免回溯。

4.3 若有文法如下，设计递归下降分析程序。

〈语句〉→〈语句〉〈赋值语句〉|ε

〈赋值语句〉→ID=〈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〉→ID|NUM|(〈表达式〉)

解：首先，去掉左递归

〈语句〉→〈语句〉〈赋值语句〉|ε 改为：〈语句〉→{〈赋值语句〉}

〈表达式〉→〈项〉 | 表达式〉 + 〈项〉 | 表达式〉 - 项〉 改为：

〈表达式〉→〈项〉{（+ |）-〈项〉}

〈项〉→〈因子〉 | 项〉 * 〈因子〉 | 项〉 / 因子〉 改为：

〈项〉→〈因子〉{（* |）/〈因子〉}

则文法变为：

〈语句〉→{〈赋值语句〉}

〈赋值语句〉→ID=〈表达式〉

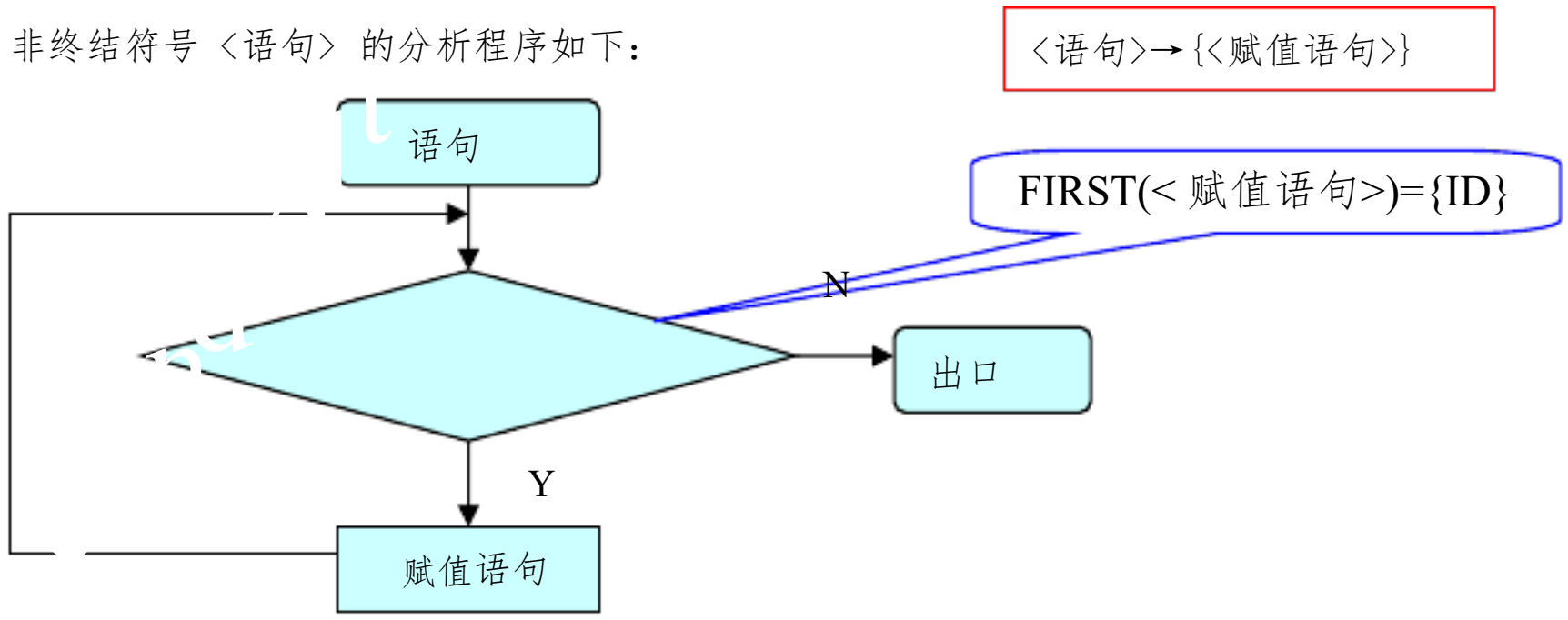
〈表达式〉→〈项〉{（+ |）-〈项〉}

〈项〉→〈因子〉{（* |）/〈因子〉}

〈因子〉→ID|NUM|(〈表达式〉)

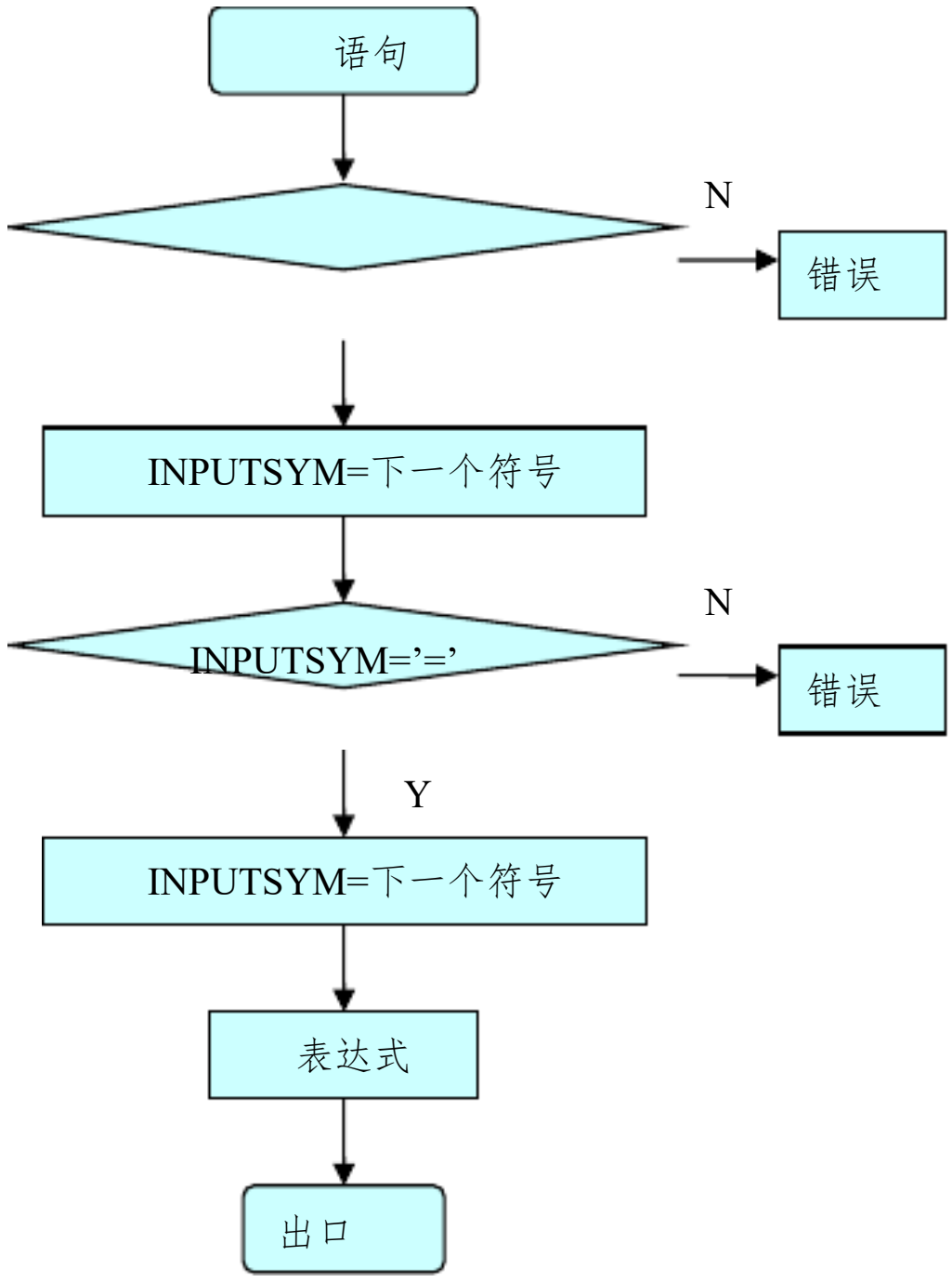
非终结符号〈语句〉的分析程序如下：

〈语句〉→{〈赋值语句〉}



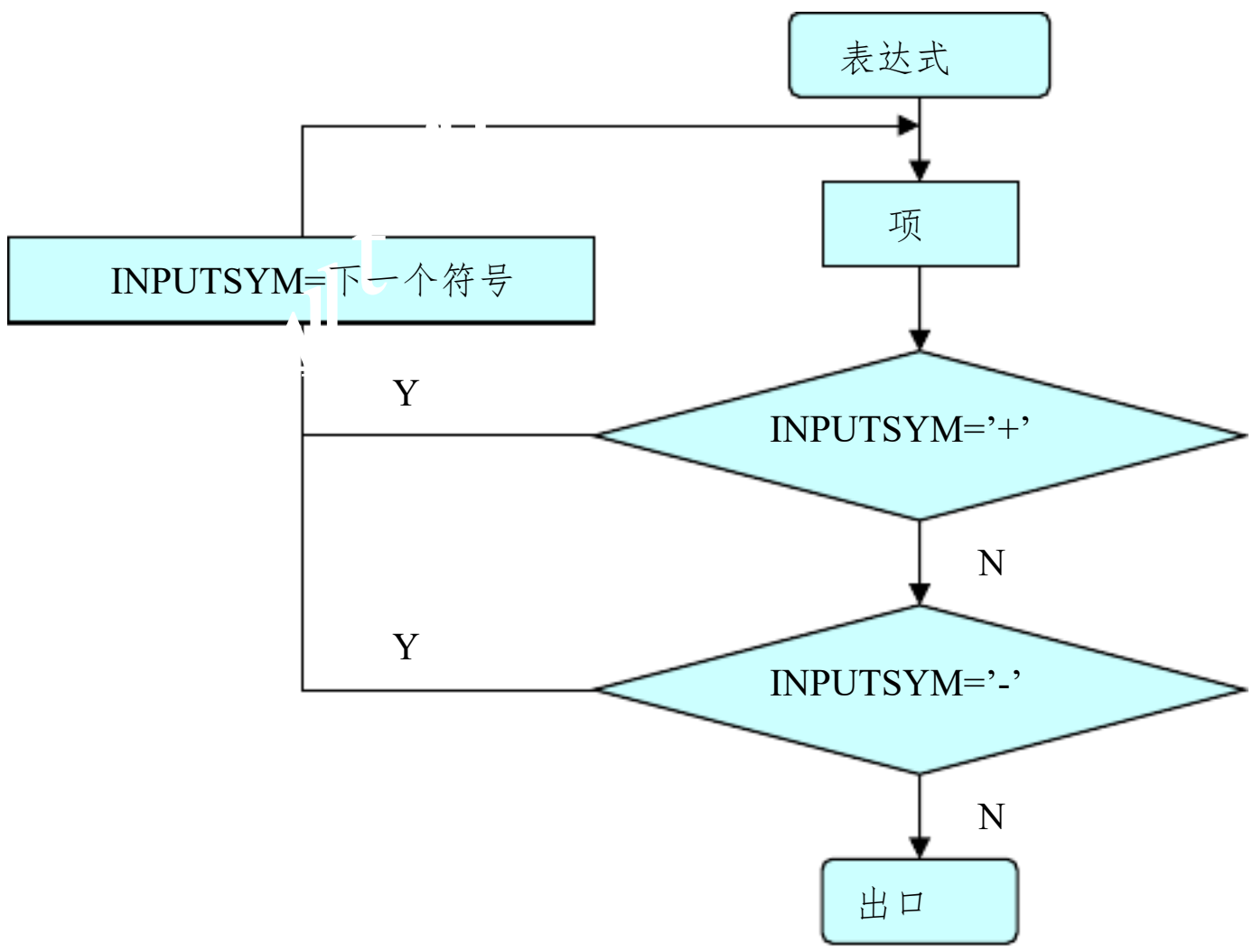
非终结符号〈赋值语句〉的分析程序如下：

〈赋值语句〉→ID=〈表达式〉



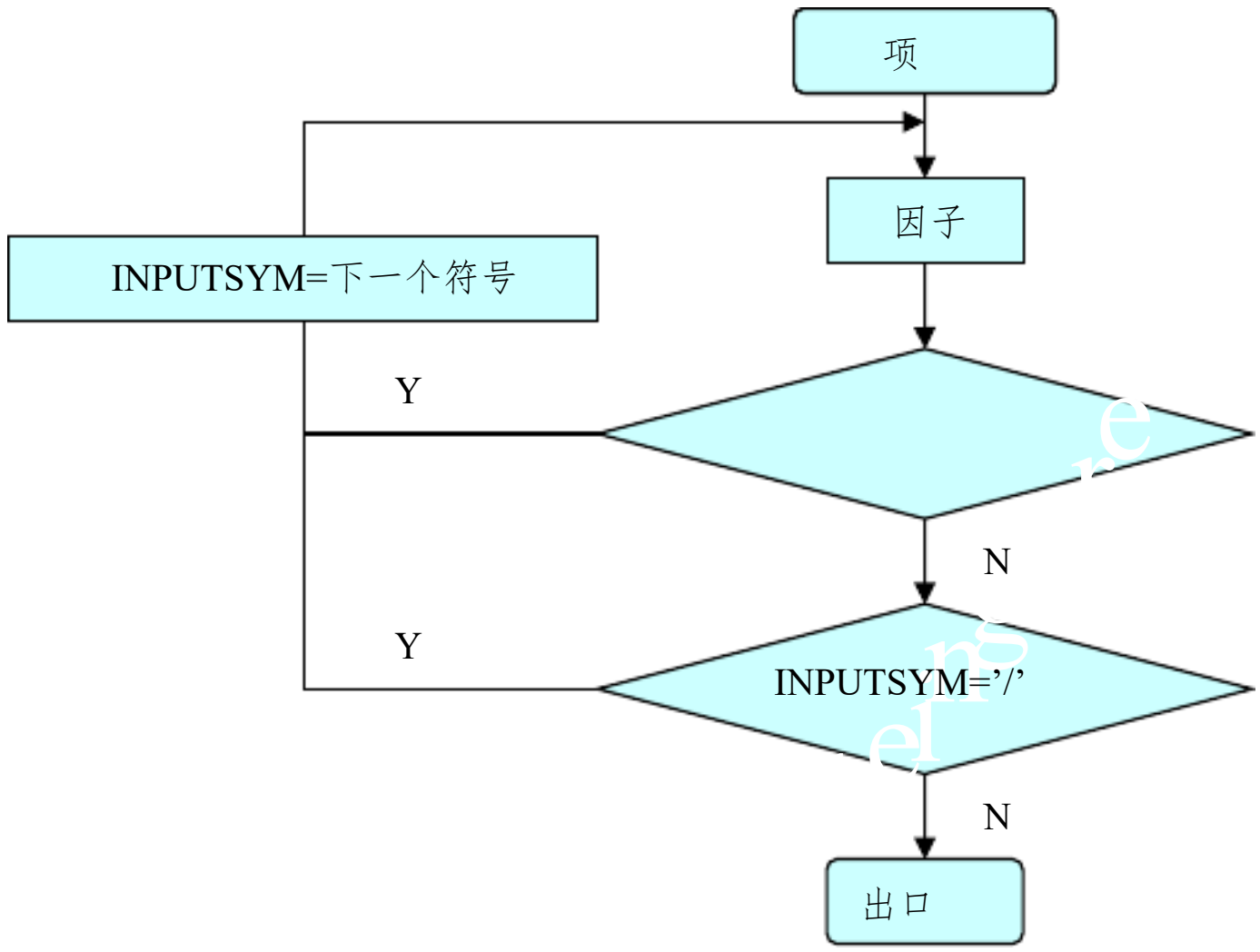
非终结符号〈表达式〉的分析程序如下：

〈表达式〉→〈项〉{（+ | -）〈项〉}



〈项〉的分析程序如下：

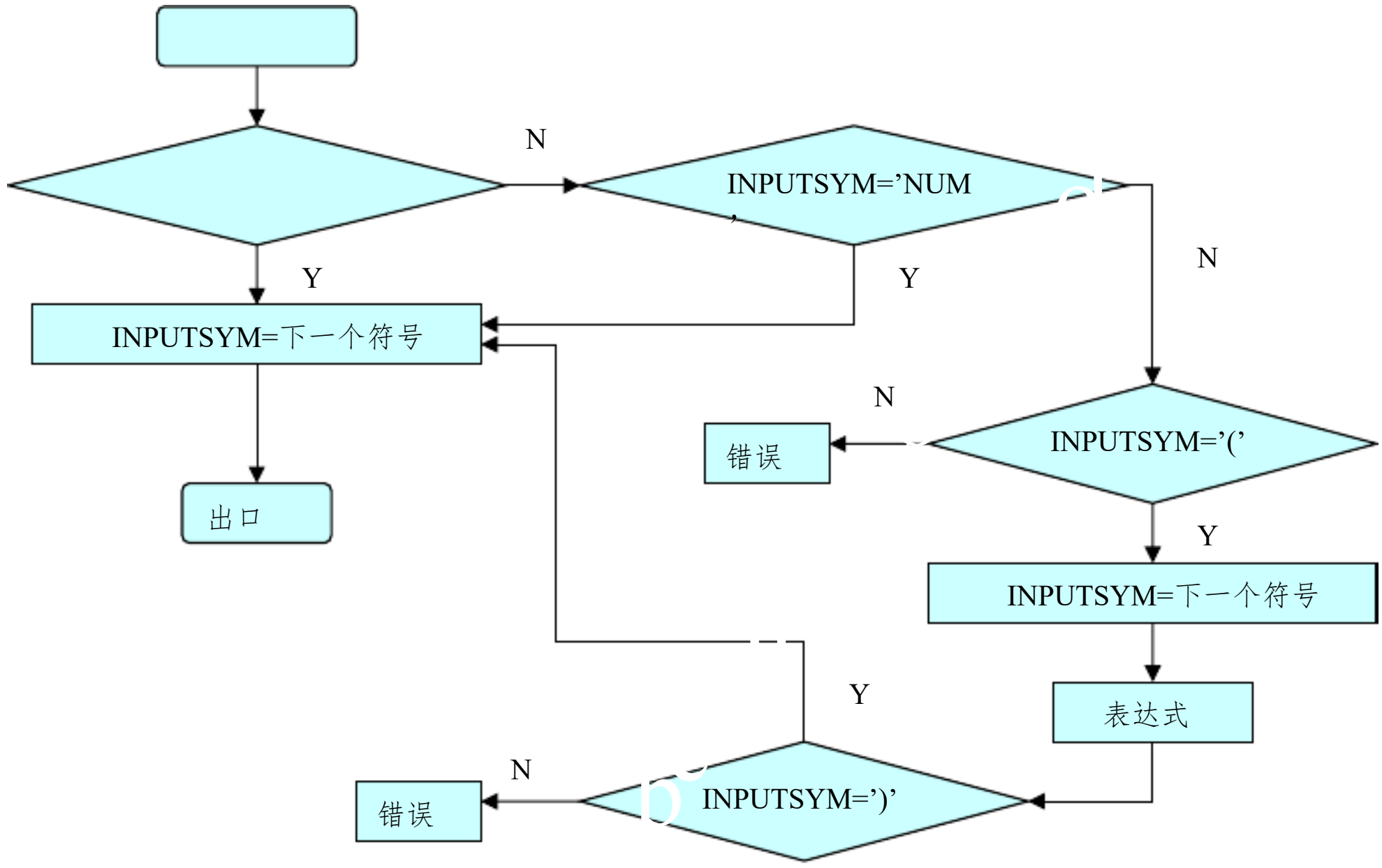
〈项〉→〈因子〉{（* |)/〈因子〉}



非终结符号〈因子〉的分析程序如下：

〈因子〉→ID|NUM|(<表达式〉)

复值语句的分析程序



有文法 G[A]：A::=aABe| ε，B::=Bb|b

（1）求每个非终结符号的 FOLLOW 集。

（2）该文法是 LL(1)文法吗？

（3）构造 LL(1)分析表。

解：

（1） FOLLOW(A)=First(B) ∪ {#}={b,#}

FOLLOW(B)={e,b}

（2） 该文法中的规则 B::=Bb|b 为左递归，因此该文法不是 LL(1)文法

（3） 先消除文法的左递归（转成右递归），文法变为：A::=aABe| ε，B::=bB’，B’::=bB’ |ε，该文法的 LL(1)分析表为：

	a	e	b	#
A	POP ，PUSH(eBAa)		POP	POP
B			POP ，PUSH(B’b)	
B’		POP	POP ，PUSH(B’b)	
a	POP, NEXTSYM			
e		POP, NEXTSYM		

			POP, NEXTSYM	
#				ACCEPT

LL(1)分析表：

	a	e	b	#
A	A→aABe		A→ε	A→ε
B			B→bB’	
B’		B’→ ε	B’→bB’	

4.5 若有文法 A→(A)A| ε

（1）为非终结符 A 构造 FIRST 集合和 FOLLOW 集合。

（2）说明该文法是 LL(1)的文法。

解：

（1）FIRST(A) = {(, ε }

FOLLOW(A) = {), #}

（2）

该文法不含左递归；

FIRST（(A)A）={(}，FIRST（ε）={ε}，FIRST（(A)A）∩FIRST（ε）=Φ，

且 FOLLOW(A) = {), #}，FIRST（(A)A）∩ FOLLOW(A) =Φ，

因此，该文法满足 LL(1)文法的条件，是 LL(1)文法。

4.6 利用分析表 4-1，识别以下算术表达式，请写出分析过程。

（1）i+i*i+i

（2）i*(i+i+i)

解：

（1）i+i*i+i

步骤	分析栈	余留输入串	分析表元素	所用产生式
1	#E	i+i*i+i#	POP,PUSH(E’T)	E→TE’
2	#E’T	i+i*i+i#	POP,PUSH(T’F)	T→FT’
3	#E’T F’	i+i*i+i#	POP,PUSH(i)	F→i
4	#E’T’i	i+i*i+i#	POP,NEXTSYM	
5	#E’T’	+i*i+i#	POP	T’→ ε
6	#E’	+i*i+i#	POP,PUSH(E’T+)	E’→+TE’
7	#E’T+	+i*i+i#	POP,NEXTSYM	
8	#E’T	i*i+i#	POP,PUSH(T’F)	T→FT’
9	#E’T’F	i*i+i#	POP,PUSH(i)	F→i
10	#E’T’i	i*i+i#	POP,NEXTSYM	
11	#E’T’	*i+i#	POP,PUSH(T’F*)	T’→*FT’
12	#E’T’F*	*i+i#	POP,NEXTSYM	
13	#E’T’F	i+i#	POP,PUSH(i)	F→i
14	#E’T’i	i+i#	POP,NEXTSYM	
15	#E’T’	+i#	POP	T’→ ε

	#E’	+i#	POP,PUSH(E’T+)	E’ +TE’
17	#E’T+	+i#	POP,NEXTSYM	
18	#E’T	i#	POP,PUSH(T’F)	T→FT’
19	#E’T’F	i#	POP,PUSH(i)	F→i
20	#E’T’i	i#	POP,NEXTSYM	
21	#E’T’	#	POP	T’→ ε
22	#E’	#	POP	E’→ ε
23	#	#	ACCEPT	

（2）i*(i+i+i)

步骤	分析栈	余留输入串	分析表元素	所用产生式
1	#E	i*(i+i+i)#	POP,PUSH(E’T)	E→TE’
2	#E’T	i*(i+i+i)#	POP,PUSH(T’F)	T→FT’
3	#E’T’F	i*(i+i+i)#	POP,PUSH(i)	F→i
4	#E’T’i	i*(i+i+i)#	POP,NEXTSYM	
5	#E’T’	*(i+i+i)#	POP,PUSH(T’F*)	T’→*FT’
6	#E’T’F*	*(i+i+i)#	POP,NEXTSYM	
7	#E’T’F	(i+i+i)#	POP, PUSH()E()	F→(E)
8	#E’T’)E(	(i+i+i)#	POP,NEXTSYM	
9	#E’T’)E	i+i+i)#	POP,PUSH(E’T)	E→TE’
10	#E’T’) E’T	i+i+i)#	POP,PUSH(T’F)	T→FT’
11	#E’T’) E’ T’F	i+i+i)#	POP,PUSH(i)	F→i
12	#E’T’) E’ T’i	i+i+i)#	POP,NEXTSYM	
13	#E’T’) E’ T’	+i+i)#	POP	T’→ ε
14	#E’T’) E’	+i+i)#	POP,PUSH(E’T+)	E’→+TE’
15	#E’T’) E’T+	+i+i)#	POP,NEXTSYM	
16	#E’T’) E’T	i+i)#	POP,PUSH(T’F)	T→FT’
17	#E’T’) E’T’F	i+i)#	POP,PUSH(i)	F→i
18	#E’T’) E’T’i	i+i)#	POP,NEXTSYM	
19	#E’T’) E’T’	+i)#	POP	T’→ ε
20	#E’T’) E’	+i)#	POP,PUSH(E’T+)	E’→+TE’
21	#E’T’) E’ T+	+i)#	POP,NEXTSYM	
22	#E’T’) E’ T	i)#	POP,PUSH(T’F)	T→FT’
23	#E’T’) E’ T’F	i)#	POP,PUSH(i)	F→i
24	#E’T’) E’ T’i	i)#	POP,NEXTSYM	
25	#E’T’) E’ T’	)#	POP	T’→ ε
26	#E’T’) E’	)#	POP	E’→ ε
27	#E’T’)	)#	POP,NEXTSYM	
28	#E’T’	#	POP	T’→ ε
29	#E’	#	POP	E’→ ε
30	#	#	ACCEPT	

4.7 考虑下面简化了的 C 声明文法：

〈声明语句〉→〈类型〉〈变量表〉;

〈类型〉→int|float|char

〈变量表〉→ID,<变量表>|ID

（1） 在该文法中提取左因子。

（2） 为所得的文法的非终结符构造 FIRST 集合和 FOLLOW 集合。

（3） 说明所得的文法是 LL(1)文法。

（4） 为所得的文法构造 LL(1)分析表。

（5） 假设有输入串为“char x，y，z；”，写出相对应的 LL(1)分析过程。

解：

（1） 规则〈变量表〉→ID,<变量表>|ID 提取公因子如下：〈变量表〉→ID（,〈变量表〉| ε ）

增加新的非终结符〈变量表 1〉，规则变为：

〈变量表〉→ID<变量表 1〉

〈变量表 1〉→，〈变量表〉| ε

C 声明文法改变为：

〈声明语句〉→〈类型〉〈变量表〉;

〈类型〉→int|float|char

〈变量表〉→ID<变量表 1〉

〈变量表 1〉→，〈变量表〉| ε

（2） FIRST(< 声明语句>)＝FIRST(< 类型>)＝｛int，float，char ｝

FIRST(< 变量表>)＝｛ID｝

FIRST(< 变量表 1>)＝｛,， ε ｝

FOLLOW(< 声明语句>)＝｛#｝

FOLLOW(< 类型>)＝FIRST(< 变量表>)＝｛ID｝

FOLLOW(< 变量表>)＝FOLLOW(< 变量表 1>)＝｛；｝

（3） 所得文法无左递归，且

FIRST （int）∩FIRST （float）∩FIRST （char ）＝Φ

FIRST （,〈变量表〉）∩FIRST （ ε ）＝Φ

FIRST （,〈变量表〉）∩FOLLOW(< 变量表 1>)＝Φ

因此，所得文法为 LL(1)文法。

（4） 所得的文法构造 LL(1)分析表如下所示：

	;	int	float	char	ID	,	#
〈声明语句〉		POP ，PUSH(；〈变量表〉〈类型〉)	POP ，PUSH(；〈变量表〉〈类型〉)	POP ，PUSH(；〈变量表〉〈类型〉)			
〈类型〉		POP ，PUSH(int)	POP ，PUSH(float)	POP ，PUSH(char)			
〈变量					POP ，		

表>					PUSH(<变量表1>ID)		
<变量表 1>	POP					POP ，PUSH(<变量表>，)	
;	POP, NEXTSYM						
int		POP, NEXTSYM					
float			POP, NEXTSYM				
char				POP, NEXTSYM			
ID					POP, NEXTSYM		
,						POP, NEXTSYM	
#							ACCEPT

更常用且简单的 LL(1)分析表：

	;	int	float	char	ID	,	#
<声明语句>		<声明语句>→<类型><变量表>;	<声明语句>→<类型><变量表>;	<声明语句>→<类型><变量表>;			
<类型>		<类型>→int	<类型>→float	<类型>→char			
<变量表>					<变量表>→ID<变量表 1>		
<变量表 1>	<变量表 1>→ ε					<变量表 1>→,<变量表>	

（5） 输入串“char x，y，z；”相对应的 LL(1)分析过程如下：

步骤	分析栈	余留输入串	分析表元素	所用产生式
1	#<声明语句>	char x，y，z；#	POP ，PUSH(；<变量表><类型>)	<声明语句>→<类型><变量表>;