§4.1.3 无左递归无回溯的自顶向下分析

- 递归下降分析---高级语言或汇编语言实现
- LL(1)分析法---用一个分析表和分析栈实现

1、递归下降分析(递归子程序, 推导过程)

实现方式:

对每一个非终结符号U,编写一个子程序F(U)

F(U):boolean

true:分析过程正常(得以匹配)

false:分析过程出错(无法匹配)

例： G[E]：

E→T|EAT　　改写文法:E→T{AT}

T→F|TMF　　T→F{MF}

T→（E）|i　　F→（E）|i

A→+|-　　A→+|-

M→*|/　　M→*|/

(1)消除左递归：有左递归

改写后的文法

G[E]:

$E \rightarrow TE'$

$E' \rightarrow ATE' \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow MFT' \mid \varepsilon$

$F \rightarrow (E) \mid i$

$A \rightarrow + \mid -$

$M \rightarrow * \mid /$

(2) 消除回溯:无回溯

产生式	FIRST(α)	FOLLOW(A)
E→TE′	{ (, i }	{) , # }
E′→ATE′	{+, — }	{) , # }
E′→ ε	{ε}	
T →FT′	{ (, i }	{+, —,) , # }
T′→MFT′	{ *, / }	{+, —,) , # }
T′→ ε	{ε}	
F →(E)	{ (}	{+, —,*, / ,) , #}
F →i	{ i }	
A →+	{+}	{ (, i }
A → —	{—}	
M →*	{* }	{ (, i }
M →/	{/ }	

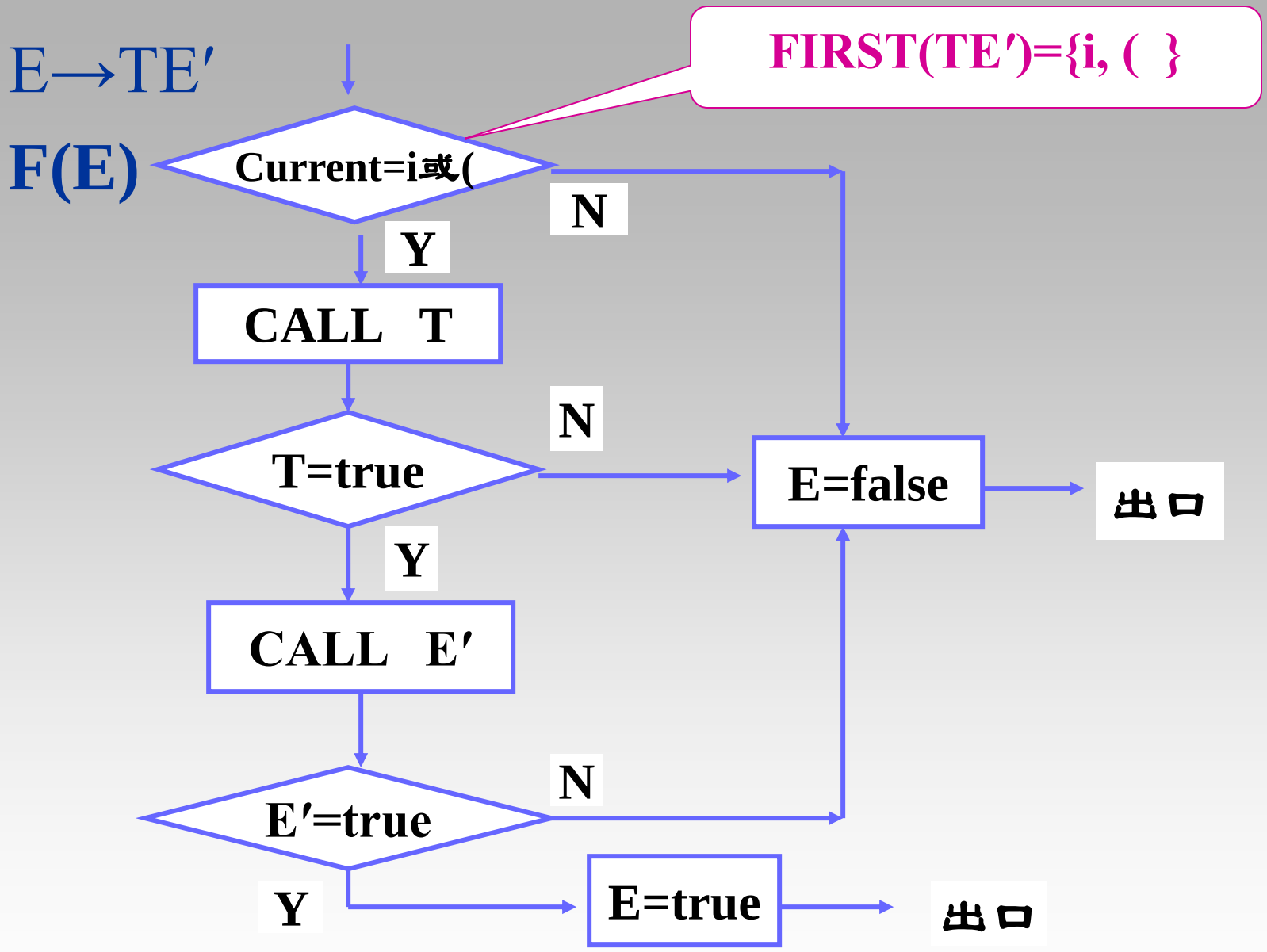
递归子程序的框图

设:

- current 中放置当前正扫描的输入符号.
- advance 表示输入符号指针后移一位.

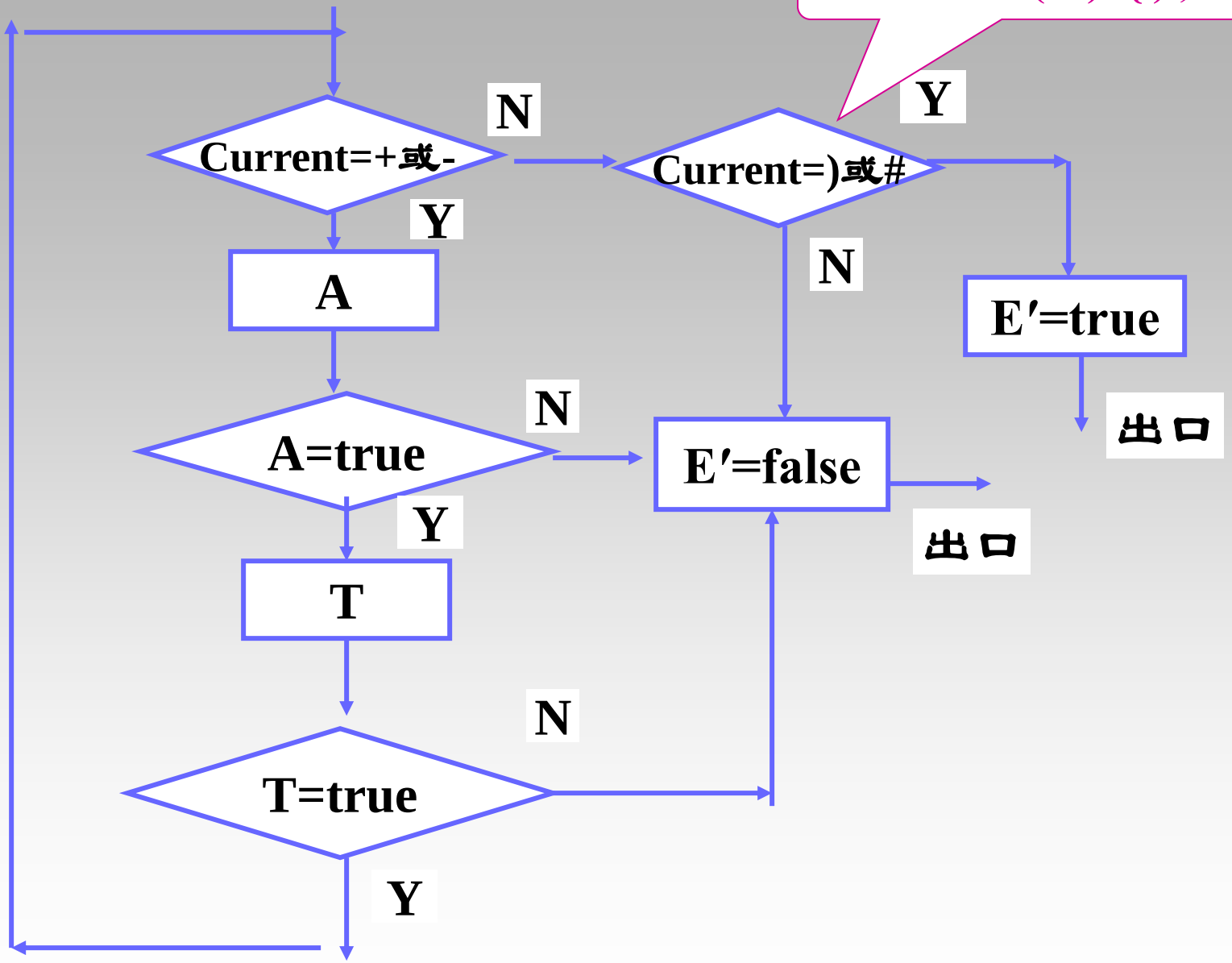
假定:

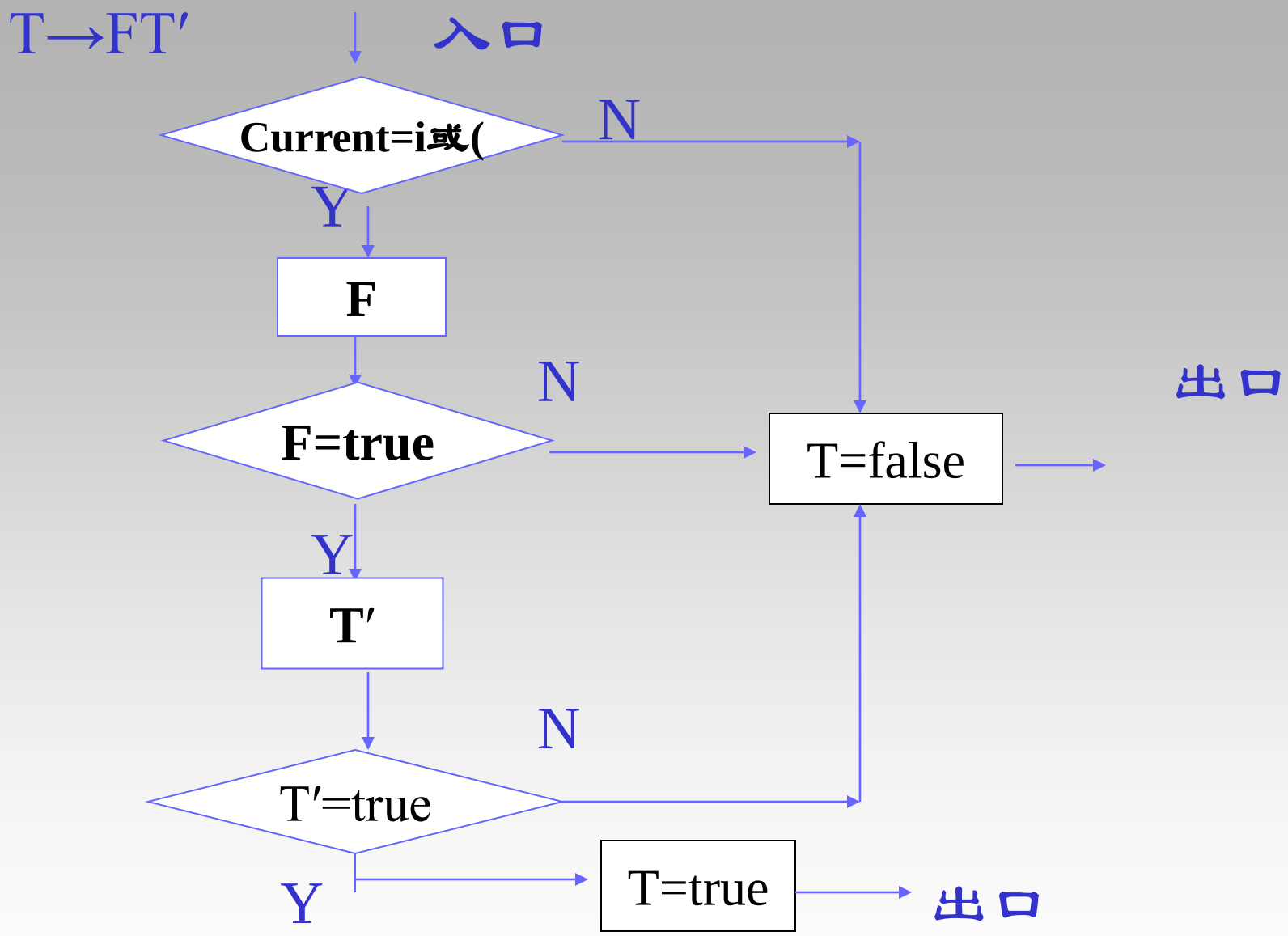
- 当进入某子程序时,要分析的输入符号已经在current中.
- 在从某一子程序退出时,下一个要分析的输入符号放入current中.


E→TE′
F(E)
FIRST(TE′)={i, (}
Current=i或(
N
Y
CALL T
N
T=true
E=false
出口
Y
CALL E′
N
E′=true
Y
E=true
出口

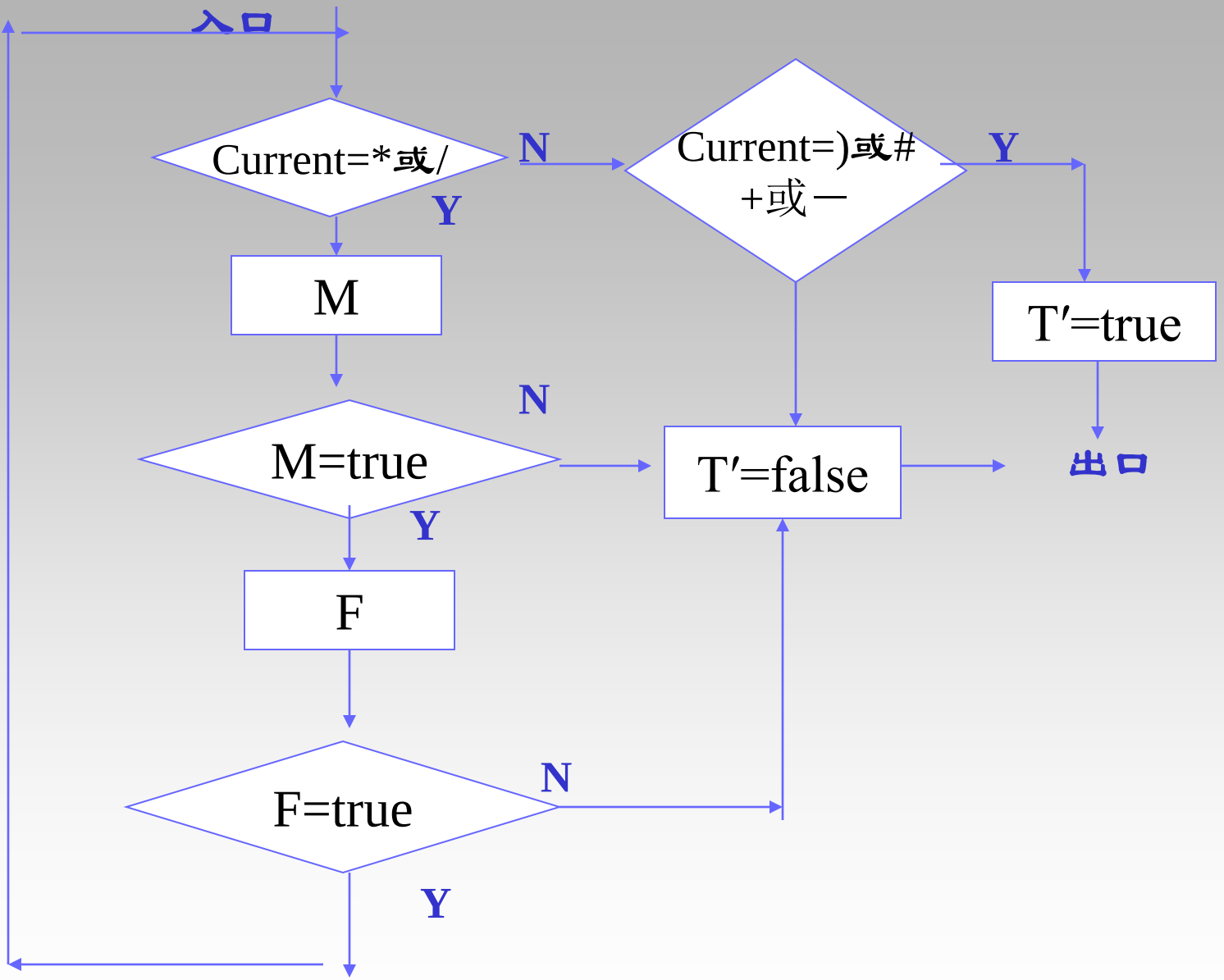
E′→ATE ′| ε

F(E’)


FOLLOW(E’)={) , # }
Current=+或-
N
Y
A
A=true
N
Y
T
T=true
N
Y
Current=)或#
Y
N
E′=true
出口
E′=false
出口

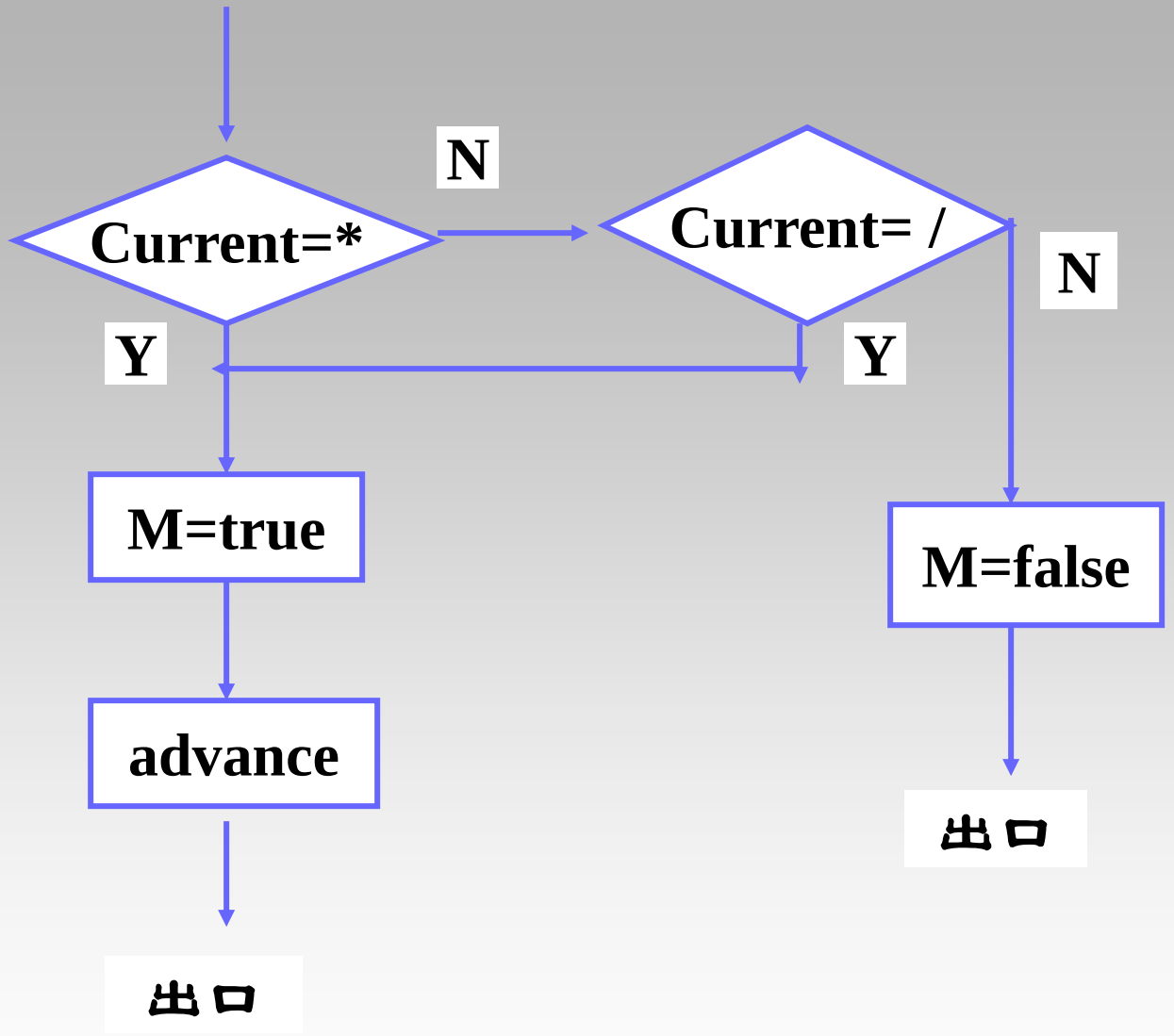

T→FT′
入口
Current=i或(
N
Y
F
N
出口
F=true
T=false
Y
T′
N
T′=true
Y
T=true
出口

T′→MFT ′| ε

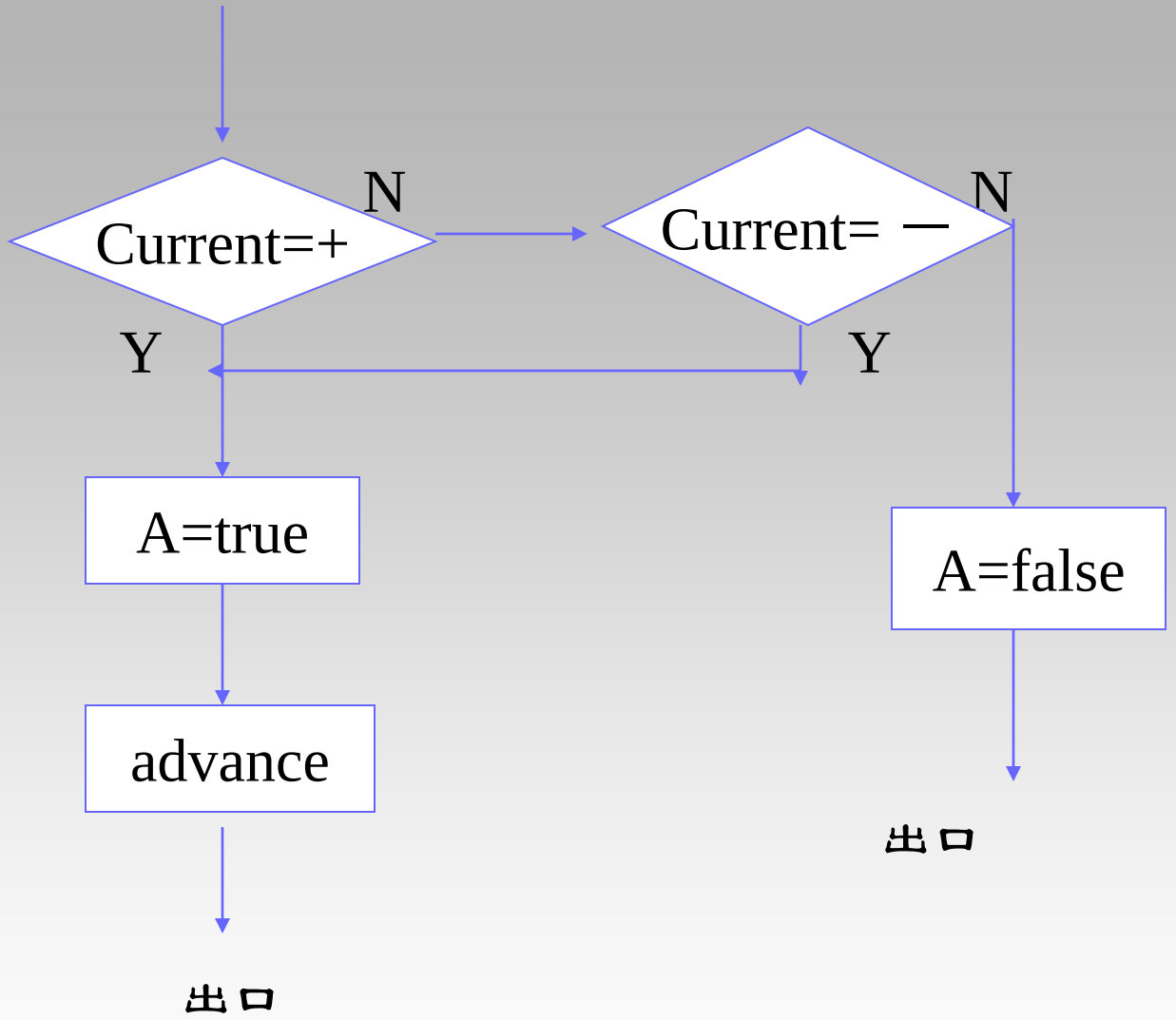

入口
Current=*或/
N
Current=)或#
+或－
Y
Y
M
T′=true
N
M=true
T′=false
出口
Y
F
F=true
N
Y

M→*|/

F(M)

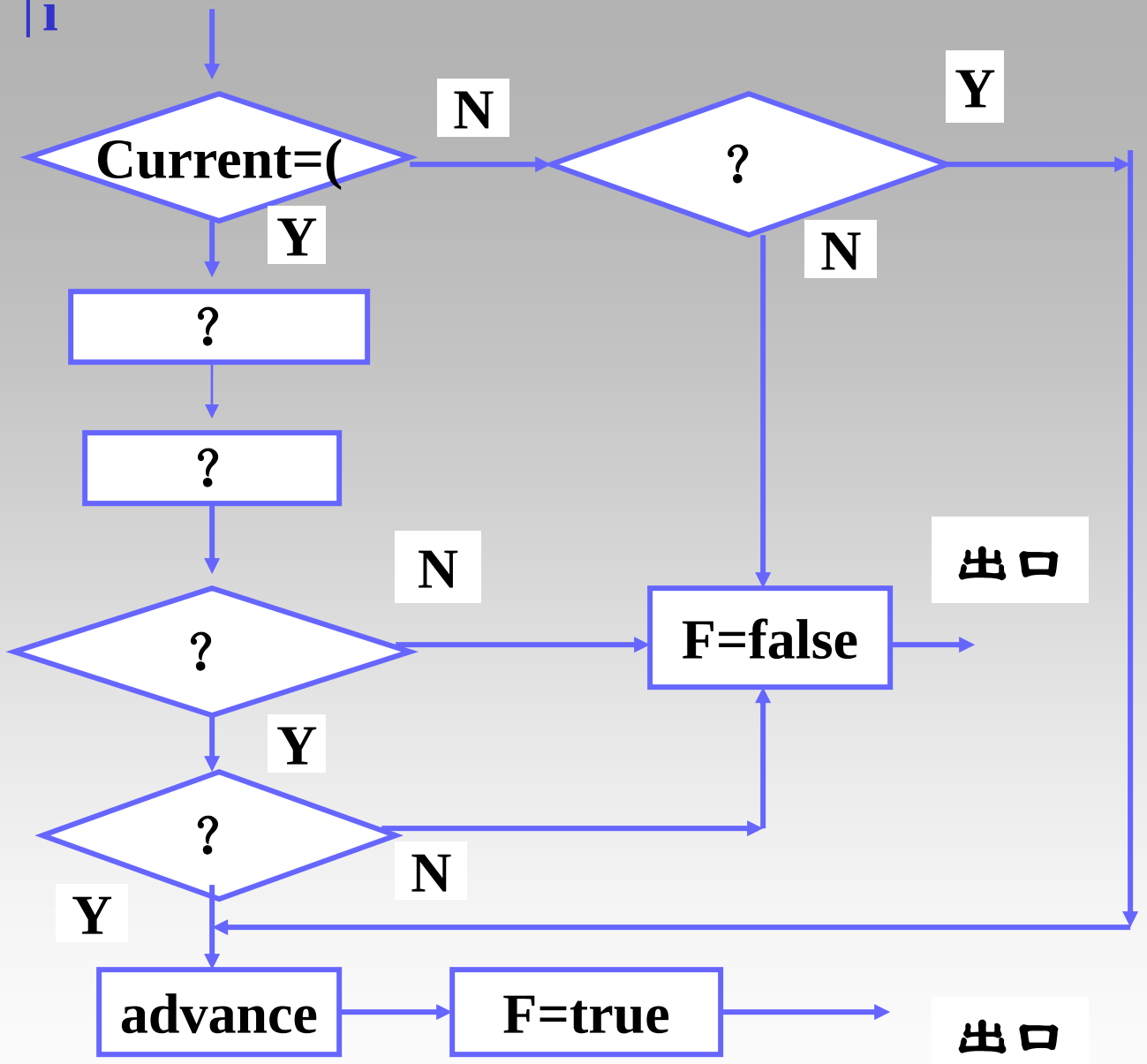


A→+|—



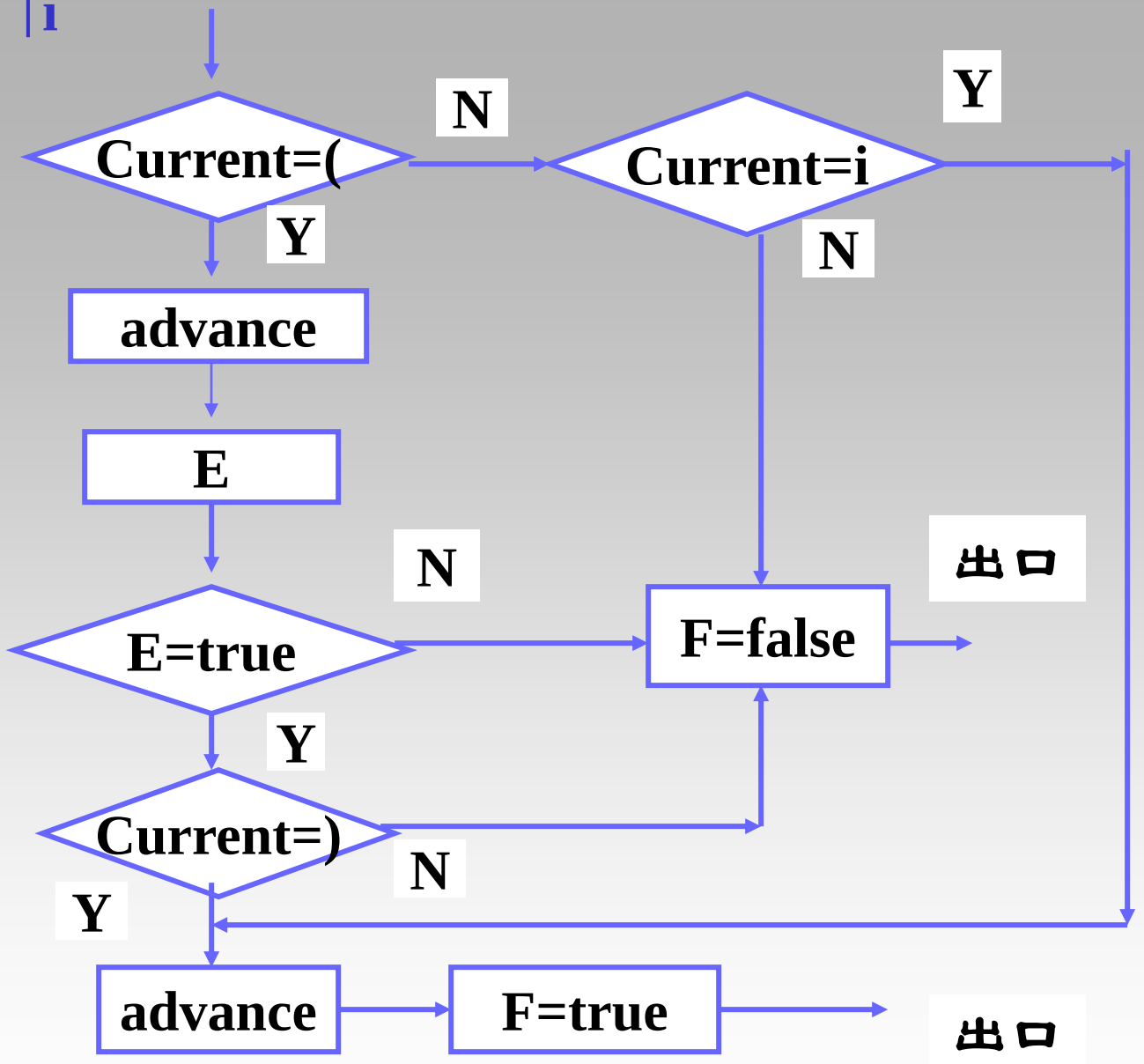
F→(E) |i

F(F)



F→(E) |i

F(F)


Current=(
N
Current=i
Y
Y
N
advance
E
N
出口
E=true
F=false
Y
Current=)
N
Y
advance
F=true
出口

2、LL(1)分析法

1:只向前看一个输入符号便能确定当前应选择的规则

L:产生一个最左推导（leftmost）

L:自左（left）向右扫描源程序

(1) LL(1)分析器的描述

①**逻辑结构:**

一张分析表M：包含文法的全部信息

一分析栈：用于存放分析过程中的文法符号

总控程序：控制分析过程（不同的文法可用一个）

M[A,a]

	a	……
A	M[A,a]	……
…..	……	……

A:处于分析栈中

a: 处于输入串中

M[A,a]:分析栈中面临输入符号a时应采取的动作

a_1	a_2	…….			a_i	…	a_n	#

将#号放在输入串的尾部

②LL(1)分析器

分析栈

#	S	…….	X_{m-1}	X_m	….

输入串　　$a_i\, a_{i+1} \cdots\cdots a_n\#$

$X_m \in V_n$且$M[X_m, a_i]$ 为$X_m \rightarrow UVW$

则分析栈为

#	S	…….	X_{m-1}	W	V	U	….

(2)LL(1)分析过程

①初始格局：#, S依次入栈 ,#置输入串尾

分析栈　　输入:$a_1a_2.....a_n$#

#	S	.....		

②反复执行,任何时候按栈顶X_m和输入a_i依据分析表,执行下述三个动作之一

■若$X_m \in V_n$

若$M[X_m,a_i]$ 对应一产生式 则X_m退栈,

　　产生式右部符号按反序进栈

　　(相当于进行一步推导)右部为ε,不进栈

若$M[X_m,a_i]$ 为error:出错处理

- 若$X_m \in V_T$

①$X_m=a_i$≠# 一步匹配,

则X_m出栈,输入符号指针指下一位置.

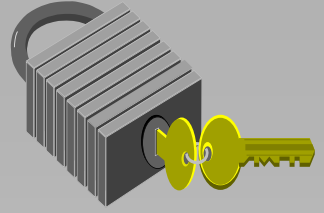
②$X_m \neq a_i$调error

■若X_m=#

①$X_m=a_i$=# 分析成功,结束

②$X_m \neq a_i$,调error

分析算法



BEGIN

把‘#’，文法开始符号依次入栈；把第一个输入符号读进a;

FLAG：=TRUE；

WHILE FLAG DO

BEGIN

把栈顶符号出栈并放在X中；

IF X ∈ V_T THEN IF X=a THEN 把下一个输入符号读进a

ELSE ERROR

ELSE IF X=‘#’ THEN

IF X=a THEN FLAG:=FALSE

ELSE ERROR

ELSE IF M[X,a]={X –> $X_1X_2..X_K$}

THEN 把X_K，X_{K-1},..,X_1依次入栈

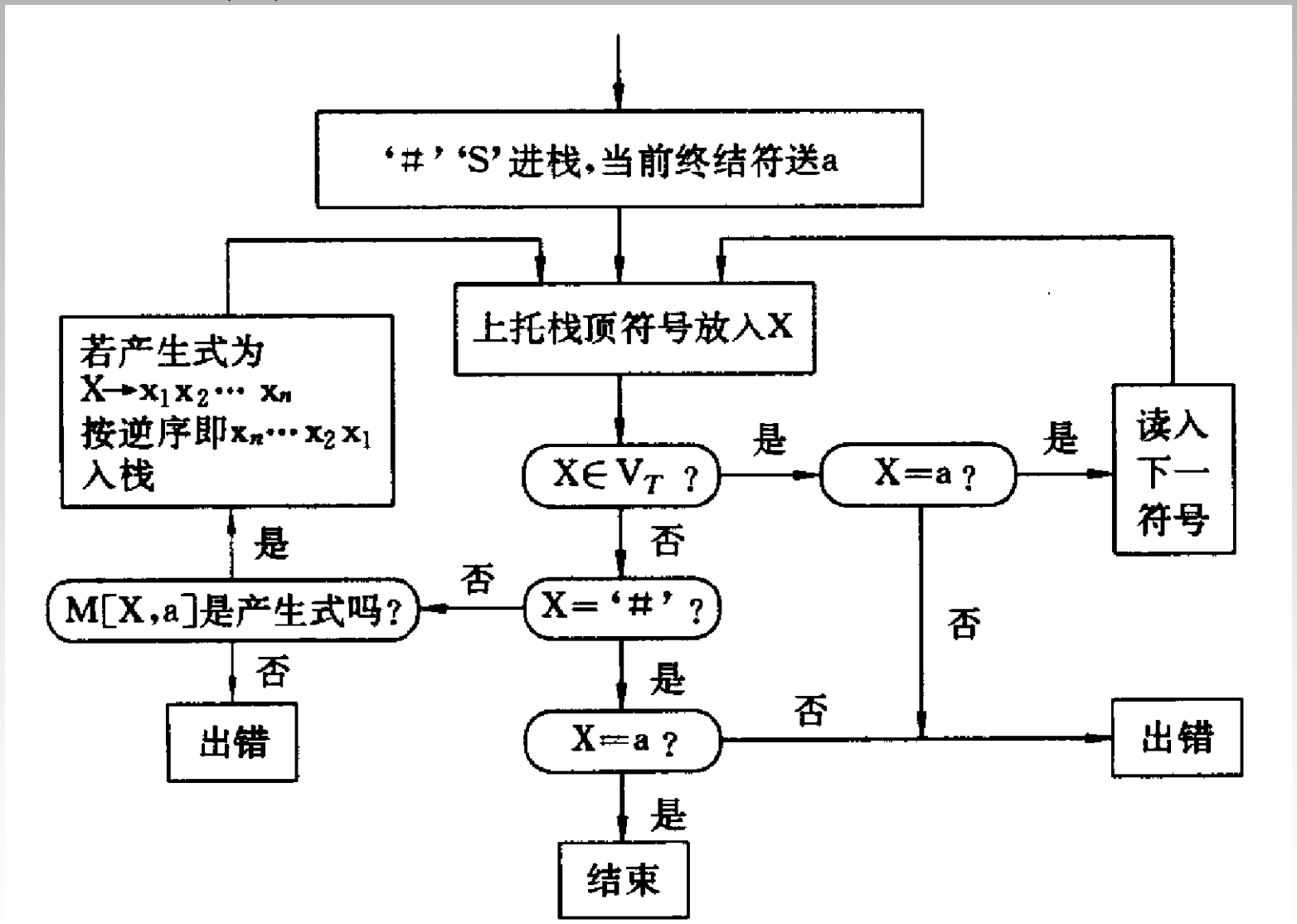
ELSE ERROR

END OF WHILE;

STOP/*分析成功，过程完毕*/

END

LL(1)分析（预测分析）程序框图



G[E]:

E→TE′

E′ →ATE′ ｜ ε

T→FT′

T′ →MFT′ ｜ ε

F→ (E) ｜i

A→+ ｜ -

M→* ｜ /

问题：试用LL（1）分析法分析输入串 i+i* i是否是文法的句子。

LL(1)分析表

	i	+	—	*	/	(	)	#
E	E→TE’					E→TE’		
E’		E’ →ATE’	E’ →ATE’				E’ → ε	E’ → ε
T	T→FT’					T→FT’		
T’		T’ → ε	T’ → ε	T’ →MFT’	T’ → MFT’		T’ → ε	T’ → ε
F	F→i					F →(E)		
A		A→+	A→—					
M				M →*	M →/			

例:输入串 i+i*i 的分析过程 (查LL(1)分析表)

分析栈	余留输入串	分析表中产生式
# E	i+i*i#	M[E,i] E→TE’
# E’ T	i+i*i#	M[T,i] T→FT’
# E’ T’ F	i+i*i#	M[F,i] F→i
# E’ T’ i	i+i*i#	Pop , Nextsym
# E’ T’	+i*i#	M[T’,+] T’ →ε
# E’	+i*i#	M[E’,+] E’ →ATE’

分析栈	余留输入串	分析表中产生式
# E’ T A	+i*i#	A→+
# E’ T +	+i*i#	Pop , Nextsym
# E’ T	i*i#	T→FT’
# E’ T’ F	i*i#	F→i
# E’ T’ i	i*i#	Pop , Nextsym
# E’ T’	*i#	T’ →MFT’
# E’ T’ F M	*i#	M→*

分析栈	余留输入串	分析表中产生式
# E’ T’ F *	*i #	Pop , Nextsym
# E’ T’ F	i#	F→i
# E’ T’ i	i#	Pop , Nextsym
# E’ T’	#	T’ → ε
# E’	#	E’ → ε
#	#	成功

结论：i+i*i是文法的合法句子

(3)LL(1)分析表的构造

两个集合

$FIRST(\alpha_i)=$

$\{a_i \mid \alpha_i \overset{*}{\Rightarrow} a_i\delta, 且 a_i \in V_t, \delta \in V^*\}$

若$\alpha_i \overset{*}{\Rightarrow} \varepsilon$,则 $\varepsilon \in FIRST(\alpha_i)$

$FOLLOW(A)=$

$\{a \mid S \overset{*}{\Rightarrow} \alpha A a\delta, 且 a \in V_t, \alpha, \delta \in V^*\}$

若 $S \overset{*}{\Rightarrow} \alpha A$, 则 $\# \in FOLLOW(A)$

▪构造FIRST的算法

㈠对G[S]，$x \in V_n \cup V_t$，计算FIRST(x)

① 若$x \in V_t$

则FIRST(x)={x}

②若$x \in V_n$

有$x \rightarrow a\alpha$ ，（$a \in V_t$）或/和$x \rightarrow \varepsilon$

则 a或/和$\varepsilon \in$ FIRST(x)

③对$X \to Y_1Y_2\ldots\ldots Y_k$(且$Y_1 \in V_n$), 反复使用以下

直到每一个FIRST(X)不再增大为止.

i 若$Y_1 \in V_n$

则把$FIRST(Y_1)-\{\varepsilon\}$元素加入FIRST(X)中

ii 若Y_1、Y_2、……$Y_{i-1} \in V_n$ $(2 \le i \le k)$

且对于任何j，$\varepsilon \in FIRST(Y_j)$ $(1 \le j \le i-1)$

则把所有$FIRST(Y_i)-\{\varepsilon\}$元素加入FIRST(X)中

iii 若Y_1、Y_2、……$Y_k \in V_n$

且对于任何j，$\varepsilon \in FIRST(Y_j)$ $(1 \le j \le k)$

则把ε元素加入FIRST(X)中

(二) 构造FIRST(α)　α= X_1X_2、……X_n

$X_i \in V$,　$\alpha \in V^*$

①置FIRST(α)={　}

②FIRST(X_1)-{ε}加入FIRST(α)

③若ε ∈ FIRST(X_1),

则FIRST(X_2)-{ε}加入FIRST(α)

若ε ∈ FIRST(X_1)且ε ∈ FIRST(X_2)

则FIRST(X_3)-{ε}加入FIRST(α)

……..以此类推

若ε ∈ FIRST(X_i)　$1 \le i \le n$

则ε ∈ FIRST(α)

■构造FOLLOW(A)的算法　　$A, B \in V_n$

①令# ∈FOLLOW(S)　　S为文法开始符号

②对$A \rightarrow \alpha B\beta$, 且$\beta \neq \varepsilon$

则将 FIRST(β) -{ε}加入FOLLOW(B)中

③反复, 直至每一个FOLLOW(A)不再增大

对$A \rightarrow \alpha B$或$A \rightarrow \alpha B\beta$(且$\varepsilon \in$ FIRST(β))

则FOLLOW(A)中的全部元素加入FOLLOW(B)

P120上部

▪构造分析表的算法

由每一个产生式$A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots\ldots \mid \alpha_n$

确定M[A,a]矩阵　$a \in V_t$

①任何$a \in FIRST(\alpha_i)$，将$A \rightarrow \alpha_i$规则填入M[A, a]

②若$\varepsilon \in FIRST(\alpha_i)$，

则对于任一个$b \in FOLLOW(A)$　$b \in V_t$或#

将$A \rightarrow \varepsilon$规则填入M[A, b]

- 此时 b不属于FIRST(A)

③其它空白为出错

例： G(S): S →MBf

M →BbS | e

B → dMg | ε 是否是LL(1)文法?

LL(1)分析表？

解：计算非终结符的First集和Follow集如下：

First(S) = {e, d, b}　　Follow(S)={#, d, f, g}

First(M) = {e, d, b}　　Follow(M)={d, f, g}

First(B) = {d, ε}　　Follow(B)={f, b}

- 对每个非终结符的产生式都有：

针对M: First(BbS)∩ First(e)= {d, b}∩{e}= Φ

针对B: First(dMg)∩Follow(B)={d} ∩ {f, b}=Φ

所以文法是LL(1)文法

例： G(S): S →MBf

M →BbS | e

B → dMg | ε 是否是LL(1)文法?

LL(1)分析表？

- 对每个非终结符的产生式有：

S: First(MBf) ={e, b, d}

M: First(BbS) = {d, b} First(e) = {e}

B: First(dMg)={d} Follow(B)={f, b}

	f	e	b	d	g	#
S		→MBf	→MBf	→MBf		
M		→e	→BbS	→BbS		
B	→ε		→ε	→dMg		

例：G(S): S→Sab|Sb|Ab

A→aA|a

将文法G改造成LL（1）G’文法,并说明;

然后构造LL(1)分析表？

解：消除左递归和提取左公因子

改造后的文法 G’[S]: S→AbS′

S′ →ab S′ |b S′ |ε

A→aA′

A′ →A|ε

对于S’：First(abS′) ∩ First(bS′)∩Follow(S′)={#}=ϕ

对于A’：First(A)={a}∩Follow(A′)={b}=ϕ

所以文法是LL(1)文法

解：消除左递归和提取左公因子

改造后的文法 G'[S]：S→AbS′

S′ →ab S′ |b S′ |ε

A→aA′

A′ →A|ε

LL(1)分析表：

	a	b	#
S	S→AbS′		
S′	S′ →ab S′	S′ →b S′	S′ →ε
A	A→aA′		
A′	A′ →A	A′ →ε	

例：G(S): S→Aa|b

A→SB

B →ab

将文法G改成（消除左递归和提取左公因子）后是否是LL（1）文法？