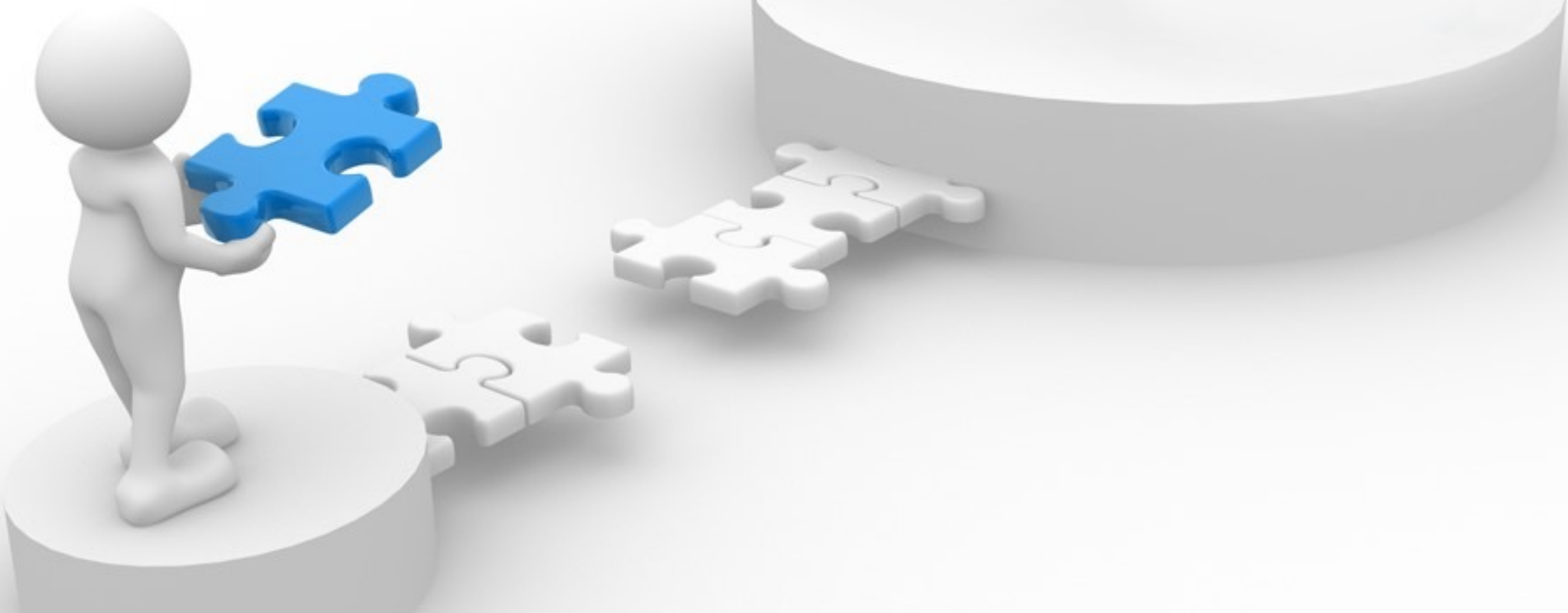
编译原理



目录

CONTENTS

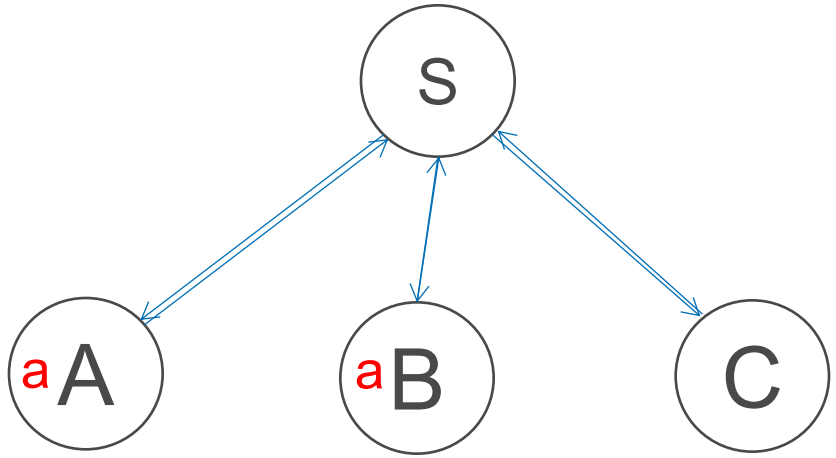
基本思想

✧ 语法分析

– **核心问题：识别**（*recognition*）**与解析**（*parsing*）

– **两种实现途径**

自顶向下（*top-down*）分析
自底向上（*bottom-up*）分析



第4章 自顶向下语法分析方法

- 例4.1
  设文法G1[S]：
  S → pA | qB
  A → cAd | a
  B → dB | b

考虑对输入串w=pccadd的分析。

- 这个文法有以下两个特点：
  - 每个产生式的右部都由终结符号开始。
  - 如果两个产生式有相同的左部，那么它们的右部由不同的终结符开始。

- 例4.2
  设文法G2[S]：考虑对输入串w= ccap的分析。
  S → Ap
  S → Bq
  A → a
  A → cA
  B → b
  B → db

- 这个文法有以下特点：
  - 产生式的右部不全是由终结符开始
  - 如果两个产生式有相同的左部，它们的右部是由不同的终结符或非终结符开始。
  - 文法中无空产生式

- 例4.3
  设文法G[S]：
  S→aA
  S→d
  A→bAS
  A→ε
  考虑对输入串w= abd的分析。

- 结论
  - 当某一非终结符的产生式中含有空产生式时，它的非空产生式右部的首符号集两两不相交，并与在推导过程中紧跟该非终结符右边可能出现的终结符集也不相交则仍可构造确定的自顶向下分析。

自顶向下预测分析

✧ 确定的自顶向下分析　　✧ LL（1）分析

– 非终结符选择和产生式选择都是确定的

无回溯的方法

从左向右扫描，可能向前查看（lookahead）确定数目的单词

分析成功的结果：得到唯一的最左推导

分析条件：对文法需要有一定的限制

自顶向下预测分析

✧ 举例

– 单词序列 a^nb^m（$n \geq 0, m > 0$）的预测分析过程

文法 G（S）：

（1）$S \to AB$

（2）$A \to aA$

（3）$A \to \varepsilon$

（4）$B \to b$

（5）$B \to bB$

S （1）

$\Rightarrow$ AB （2）

$\Rightarrow$aAB （2）

文法 G（S）：$S \to abA \mid abB$

……

$A \to a$

$\Rightarrow a^nAB$ （3）

$B \to b$

$\Rightarrow a^nB$ （5）

$\Rightarrow a^nbB$ （5）

文法 G（S）：$S \to aAb \mid aAc$

……

$A \to a \mid aA$

$\Rightarrow a^nb^{m-1}B$ （4）

$\Rightarrow a^nb^m$ （成功）

只要向前查看 2 个单词，就可预测分析L(G)中所有句子

LL（1） 分析

从左（Left）向右扫描单词

–向前查看一个单词

–产生的是最左（Leftmost）推导

✧ 对文法的限制

– 要求文法是LL（1）的

思考题： 什么是LL（1）文法？

例4.4

文法G1[S] :

S→aSb

S→aS

S→ε

- 定义4.1（开始符号集或首符号集）
  设$G=(V_T,V_N,P，S)$是上下文无关文法

  $FIRST(\alpha)=\{a|\alpha \overset{*}{\Rightarrow} a\beta,a\in V_T, \alpha,\beta\in V^*\}$

  若$\alpha \overset{*}{\Rightarrow} \varepsilon$，则规定$\varepsilon\in FRIST(\alpha)$。

- 设文法G2[S]：
- S → Ap
  S → Bq
  A → a
  A → cA
  B → b
  B → db

- **定义4.2（后跟符号集）**

  设$G=(V_T,V_N,P,S)$是上下文无关文法, 对$A\in V_N$定义

  $FOLLOW(A)=\{a|S\overset{*}{\Rightarrow}\mu A\beta$ 且$a\in V_T$，$a\in FRIST(\beta)$，$\mu\in V^*$，$\beta\in V^+\}$

  若$S\overset{*}{\Rightarrow}uA\beta$，且$\beta\overset{*}{\Rightarrow}\varepsilon$，则规定$\#\in FOLLOW(A)$

- 设文法G[S]：

  S→aA | d

  A→bAS | ε

- **定义4.3**

  给定上下文无关文法的产生式$A\to\alpha$， $A\in V_N$，$\alpha\in V^*$，

  若$\alpha \overset{*}{\not\Rightarrow} \varepsilon$，则SELECT($A\to\alpha$) = FIRST($\alpha$)

  若如果$\alpha \overset{*}{\Rightarrow} \varepsilon$，则SELECT($A\to\alpha$) = (FIRST($\alpha$)-{$\varepsilon$})∪FOLLOW(A)

例如对文法G3[S]:

S→aA|d

A→bAS|ε

SELECT(S→aA)={a}

SELECT(S→d)={d}

SELECT(A→bAS)={b}

SELECT(A→ε)={a,d,#}

- **定义4.4**

  一个上下文无关文法是LL(1)文法的充分必要条件是，对每个非终结符A的两个不同产生式，$A\to\alpha$， $A\to\beta$，满足

  $SELECT(A\to\alpha) \cap SELECT(A\to \beta) = \varnothing$

  其中α、β不能同时$\overset{*}{\Rightarrow}\varepsilon$。

4.2 LL(1)文法的判别

- 求出能推出ε的非终结符
- 计算FIRST集
- 计算FOLLOW集
- 计算SELECT集
- 判别是否是LL(1)文法

- 例：设文法G[S] 为:
  S→AB
  S→bC
  A→ε
  A→b
  B→ε
  B→aD
  C→AD
  C→b
  D→aS
  D→c
  判断它是否是LL(1)文法。

1.求出能推出ε的非终结符

~~S→ε~~

~~S→bC~~

~~A→ε~~

~~A→b~~

~~B→ε~~

~~B→aD~~

~~C→D~~

~~C→b~~

~~D→aS~~

~~D→c~~

非终结符	S	A	B	C	D
初值	未定	未定	未定	未定	未定
第一次扫描					
第二次扫描					

2.计算FIRST集

- 根据定义计算
  1.若$X \in V_T$,则FIRST(X)={X}
  2.若$X \in V_N$,且有产生式$X \to a\ldots$， $a \in V_T$,则$a \in$FIRST(X);若$X \to \varepsilon$也是一条产生式,则$\varepsilon \in$FIRST(X).
  3.若$X \to Y\ldots$是一个产生式且$Y \in V_N$,则把FIRST(Y)中的所有非ε元素都加到FIRST(X)中;若$X \to Y_1Y_2\ldots Y_K$ 是一个产生式,$Y_1, Y_2, \ldots, Y_{i-1}$都是非终结符,而且对于任何j,1≤j ≤i-1,FIRST(Y_j)都含有ε(即 $Y_1..Y_{i-1} \overset{*}{\Rightarrow} \varepsilon$),则把FIRST(Y_i)中的所有非ε元素都加到FIRST(X)中;特别是,若所有的FIRST(Y_j , j=1,2,…,K)均含有ε,则把ε加到FRIST(X)中.

计算FIRST集示例

文法G[S]为：

S→AB|bC

A→b|ε

B→aD|ε

C→AD|b

D→aS|c

非终结符	S	A	B	C	D
初值	FIRST(S)={a,b,ε}	未定	未定	未定	未定
第一次扫描	FIRST(A)={b,ε}	是	是	FIRST(bC)={b}	否
	FIRST(B)={a,ε}			FIRST(b)={b}	
第二次扫描	FIRST(C)={a,b,c}	是	是	否	否
	FIRST(D)={a,c}			FIRST(ε)={ε}	
				FIRST(AD)={a,b,c}	
				FIRST(aD)={a}	
				FIRST(aS)={a}	
				FIRST(c)={c}	

3.计算FOLLOW集

- **根据定义计算**

  1.对于文法的开始符号S,置#于FOLLOW(S) 中;

  2.若 A →αBβ是一个产生式,则把 FIRST(β)-{ε}加至FOLLOW(B)中;

  3.若 A →αB是一个产生式,或 A →αBβ是一个产生式而β $\overset{*}{\Rightarrow}$ ε，则把FOLLOW(A)加至FOLLOW(B)中.

计算FOLLOW集示例

文法G[S]为：

S→AB|bC

A→b|ε

B→aD|ε

C→AD|b

D→aS|c

非终结符	S	A	B	C	D
初值	FOLLOW(S)={#}	未定	未定	未定	未定
第一次扫描	FOLLOW(A)={a,c,#}	是	是	否	否
第二次扫描	FOLLOW(B)={#}	是	是	否	否
	FOLLOW(C)={#}				
	FOLLOW(D)={#}				

4.计算SELECT集

文法G[S]为：

S→AB|bC

A→b|ε

B→aD|ε

C→AD|b

D→aS|c

非终结符	S	A	B	C	D
初值	未定	未定	未定	未定	未定
第一次扫描		是	是		否
第二次扫描	是			否	

4.计算SELECT集

非终结符名	是否$\overset{*}{\Rightarrow}\varepsilon$	FIRST 集	FOLLOW 集
S	是	{b,a,ε}	{#}
A	是	{b,ε}	{a,c,#}
B	是	{a,ε}	{#}
C	否	{a,b,c}	{#}
D	否	{a,c}	{#}

SELECT(S→AB)={a,b,#}

SELECT(S→bC)={b}

SELECT(A→ε)={a,c,#}

SELECT(A→b)={b}

SELECT(B→ε)={#}

SELECT(B→aD)={a}

SELECT(C→AD)={a,b,c}

SELECT(C→b)={b}

SELECT(D→aS)={a}

SELECT(D→c)={c}

5. 判别是否是LL(1)文法

$$SELECT(S\rightarrow AB)\cap SELECT(S\rightarrow bC)=\{b,a,\#\}\cap\{b\}=\{b\}\neq\varnothing$$

$$SELECT(A\rightarrow \varepsilon)\cap SELECT(A\rightarrow b)=\{a,c,\#\}\cap\{b\}=\varnothing$$

$$SELECTt(B\rightarrow \varepsilon)\cap SELECT(B\rightarrow aD)=\{\#\}\cap\{a\}=\varnothing$$

$$SELECTt(C\rightarrow AD)\cap SELECT(C\rightarrow b)=\{b,a,c\}\cap\{b\}=\{b\}\neq\varnothing$$

$$SELECT(D\rightarrow aS)\cap SELECT(D\rightarrow c)=\{a\}\cap\{c\}=\varnothing$$

所以文法G[S] 不是LL(1)文法。

LL（1） 分析

✧ **课堂练习：** 验证如下文法 G（S）不是 LL（1）文法

文法 G（S）：

（1）$S \rightarrow AB$

（2）$A \rightarrow Da \mid \varepsilon$

（3）$B \rightarrow cC$

（4）$C \rightarrow aADC \mid \varepsilon$

（5）$D \rightarrow b \mid \varepsilon$

4.3 某些非LL(1)文法到LL(1)文法的等价变换

- 提取左公共因子
- 消除左递归

1. 提取左公共因子

- 例4.6
  文法G1[S]：
  S→aSb
  S→aS
  S→ε

$A \to \alpha\beta_1|\alpha\beta_2|\ldots|\alpha\beta_n$ 变换为

$A \to \alpha(\beta_1|\beta_2|\ldots|\beta_n)$，即

$A \to \alpha A'$

$A' \to \beta_1|\beta_2|\ldots|\beta_n$

化为：
S→aS(b|ε)
S→ε

进一步化为：
S→aSA
A→b
A→ε
S→ε

结果仍然不是LL(1)文法。因此，文法中不含左公共因子只是LL(1)文法的必要条件。

- 例4.7
  文法G2[A]:
  A→ad
  A→Bc
  B→aA
  B→bB

1.化为：
A→ad
A→aAc
A→bBc
B→aA
B→bB

2.化为：
A→a(d|Ac)
A→bBc
B→aA
B→bB

3.化为：
A→aA'
A→bBc
A'→d
A'→Ac
B→aA
B→bB

- 例4.8

文法G3[S]：

S→aSd

S→Ac

A→aS

A→b

1.化为：

S→aSd

S→aSc

S→bc

A→aS

A→b

2.化为：

S→aS(d|c)

S→bc

A→aS

A→b

3.化为：

S→aSA'

S→bc

A'→d

A'→c

结果中A是不可达到的符号。

- 例4.9
  文法G4[S]：
  S→Ap|Bq
  A→aAp|d
  B→aBq|e

1.化为：
S→aApp|aBqq|dp|eq
A→aAp|d
B→aBq|e

2.化为：
S→a(App|Bqq)
S→dp|eq
A→aAp|d
B→aBq|e

3.化为：
S→aS'
S→dq|eq
S'→App|Bqq
A→aAp|d
B→aBq|e

4.化为：
S→aS'
S→dp|eq
S'→
aAppp|aBqqq|dpp|eqq
A→aAp|d
B→aBq|e

- **结论**

  不一定每个文法的左公共因子都能在有限的步骤内替换成无左公共因子的文法。

  一个文法提取了左公共因子后，只解决了相同左部产生式右部的FIRST集不相交问题，当改写后的文法不含空产生式，且无左递归时，则改写后的文法是LL(1)文法，否则还需用LL(1)文法的判别方式进行判断才能确定是否为LL(1)文法。

2. 消除左递归

- 直接左递归

  $A \rightarrow A\beta \quad A \in VN,\ \beta \in V^*$

- 间接左递归

  $A \rightarrow B\beta$

  $B \rightarrow A\alpha \quad A,B \in VN,\ \alpha,\beta \in V^*$

- 例4.10
  G5[S]:
  S→Sa |Se
  S→b |c |d
  考虑对输入串baaaa#的分析
- 例4.11
  G6[A]:
  A→aB
  A→Bb
  B→Ac
  B→d
  考虑对输入串adbcbcbc#的分析

消除直接左递归

- $A \rightarrow A\alpha_1 | A\alpha_2 | \ldots | A\alpha_m | \beta_1 | \beta_2 | \ldots | \beta_n$
  其中： α_i 不等于ε， β_j不以A开头。
  改为：
  $A \rightarrow \beta_1 A' | \beta_2 A' | \ldots | \beta_n A'$
  $A' \rightarrow \alpha_1 A' | \alpha_2 A' | \ldots | \alpha_m A' | \varepsilon$

例4.10
G5[S]:
S→Sa |Se
S→b |c |d

消除间接左递归

- 将间接左递归变为直接左递归，然后消除直接左递归。

例4.11
G6[A]：
A→aB
A→Bb
B→Ac
B→d

消除文法中一切左递归的算法

1. **以某种顺序将文法非终结符排列$A_1, A_2 \ldots A_n$**

2. for i:=1 to n do

   begin

   for j:=1 to i-1 do

   用A_j-->α_1| α_2 |…|α_k关于A_j的全部产生式替代

   形如A_i--> A_jr的产生式为:

   A_i--> α_1r| α_2r|…|α_kr;

   end;

   消除A_i的直接左递归;

3. **化简由2得到的文法**

⊙ 例

文法G[S]:

S→Qc|c

Q→Rb|b

R→Sa|a

将非终结符排序为S、Q、R

R→ *Qca*|ca|a

R→ *Rbca*|*bca*|*ca*|a

R→ (bca|ca|a)R'

R' → bcaR'|ε

✧ 课堂练习：左递归消除

原文法 G[E]: $E \to E + T \mid T$

$T \to T * F \mid F$

$F \to (E) \mid a$

消除左递归后的文法 G'[E]:

(1) $E \to TE'$

(2) $E' \to + TE'$

(3) $E' \to \varepsilon$ (4) $T \to FT'$

(5) $T' \to * FT'$ (6) $T' \to \varepsilon$

(7) $F \to (E)$ (8) $F \to a$

4.4不确定的自顶向下分析思想（自学）

✧ 举例

– 单词序列 aaab 的一个自顶向下分析过程

文法 G（S）：

（1）$S \rightarrow AB$

（2）$A \rightarrow aA$

（3）$A \rightarrow \varepsilon$

（4）$B \rightarrow b$

（5）$B \rightarrow bB$

S （1）

$\Rightarrow$ AB （2）

$\Rightarrow$ aAB （3）

$\Rightarrow$ aB （4）

$\Rightarrow$ ab （回溯）

……

复杂度很高

失败条件较复杂

4.5 确定的自顶向下分析方法

- 递归子程序法
- 预测分析法

借助于**预测分析表**和**符号栈**。

- **实现思想**
  - 对文法中每个非终结符编写一个递归过程，每个过程的功能是识别由该非终结符推出的串，当某非终结符的产生式有多个候选时能够按LL(1)形式可唯一地确定选择某个候选进行推导。
- **限制**
  - 对文法要求高，必须满足LL(1)文法；
  - 速度慢，占用空间多。

递归下降 LL（1）分析程序

✧ 非终结符对应的递归下降子程序

– 一般结构

设 A 的产生式:

$$A \to u_1 \mid u_2 \mid \ldots \mid u_n,$$

相对于非终结符 A 的递归下降子程序 ParseA 的一般结构

```
void ParseA()
{
    switch (lookahead) {
        case S(A→u1):
            /* code to recognize u1 */
            break;
        case S(A→u2):
            /* code to recognize u2 */
            break;
        ...
    }
}
```

递归下降 LL（1）分析程序

G(S): $S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

Select($S \rightarrow AaS$) = {*a*}

Select($S \rightarrow BbS$) = {*c,b*}

Select($S \rightarrow d$) = {*d*}

```
void ParseS( )
{
    switch (lookahead)  {
        case a:
            ParseA( );
            MatchToken(a);
            ParseS( );
            break;
        case b,c:
            ParseB( );
            MatchToken(b);
            ParseS( );
            break;
        case d:
            MatchToken(d);
            break;
        default:
            printf("syntax error \n")
            exit(0);
    }
}
```

递归下降 LL（1）分析程序

G(S): $S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

$Select(A \rightarrow a) = \{a\}$

$Select(B \rightarrow \varepsilon) = \{b\}$

$Select(B \rightarrow c) = \{c\}$

```
void ParseB( )
{
    if (lookahead==c)  {
        MatchToken(c);
    }
    else if (lookahead==b) {
    }
    else {
        printf("syntax error \n”);
        exit(0);
    }
}
```

```
void ParseA( )
{
    if (lookahead==a)  {
        MatchToken(a);
    }
    else {
        printf("syntax error \n”);
        exit(0);
    }
}
```

PL/0 编译程序的语法分析

递归子程序的设计实例

– <表达式> ::= [+ | -] <项> { （ + | - ） <项> }

```
int expression(…)
{
    if (sym==plus || sym==minus)    {/* 此时表达式被看作正的或负的项 */
            getsym();
            term (…);                /* 处理<项> */
    }
    else  /* 此时表达式被看作项的加减 */
    {
            term (…);                /* 处理<项> */
    }
    while (sym==plus || sym==minus)
    {
            getsym;
            term (…);      /* 处理<项> */
    }
    return 0;
}
```

PL/0 编译程序的语法分析

递归子程序的设计实例

– <项> ::= <因子> { （* | / ） <因子> }

```
int term(…)
{
    factor（…）;            /*处理<因子>*/
      while (sym==times || sym==slash)
      {
          getsym();
           factor (…);          /* 处理<因子> */
    }
    return 0;
}
```

PL/0 编译程序的语法分析

递归子程序的设计实例

– <因子> ::= <标识符> | <无符号整数> | ‘（’ <表达式> ‘）’

```
int factor (…)
{
    if (sym==ident)          {/* <因子>为常量或变量 */
        getsym();
    else  if (sym==number)       /*<因子>为立即数*/
            getsym();
        else if  (sym==lparen);        /* <因子>为立即数*/
            {
                expression(…);
            if (sym==rparen)
                getsym();
            else
                error(22);   /*提示22号出错信息：缺少右括
    号*/
            }
    return 0;
}
```

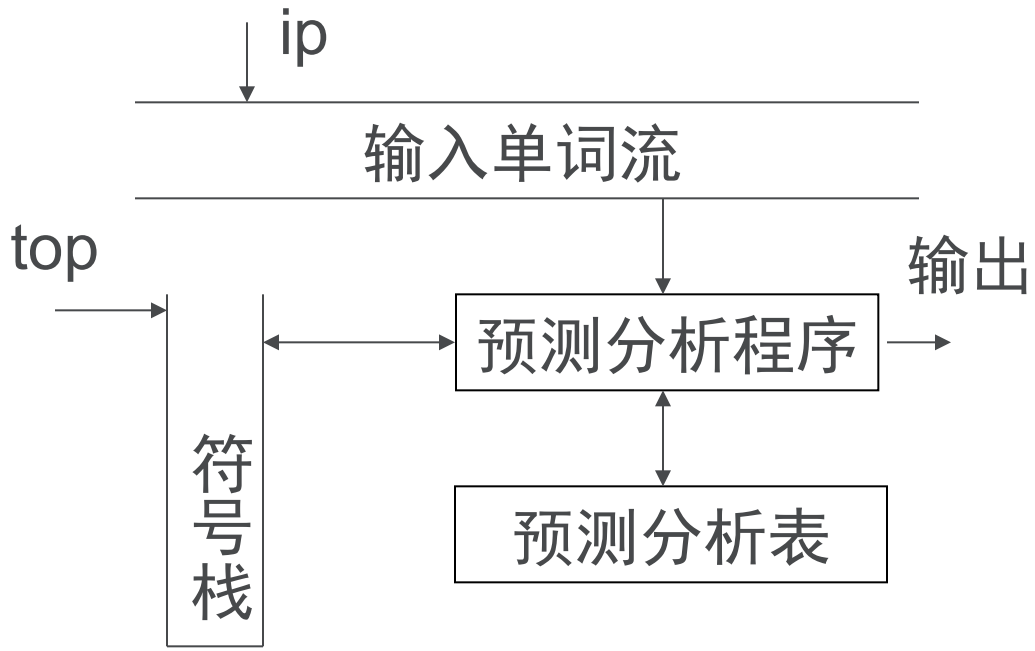
预测分析方法

- **一个预测分析器是由三个部分组成**

  预测分析程序

  符号栈

  预测分析表



预测分析表

- 预测分析表可用一个矩阵M表示。矩阵的元素M[A，a]中的内容为一条关于A的产生式，表明当用非终结符A向下推导且面临输入符a时，所应采取的候选产生式，当元素内容无产生式时，则出错。
- 一个文法的预测分析表不含有多重入口，当且仅当该文法是LL(1)的。
- 预测分析表的构造算法

若$a \in SELECT(A \rightarrow \alpha)$，则把$A \rightarrow \alpha$放入M[A, a]中。

把所有无定义的M[A，a]标上出错标记。

- 例

文法G[E]：

E →TE’

E’ →+TE’| ε

T →FT’

T’ →*FT’| ε

F → i | (E)

	i	+	*	(	)	#
E						
E'						
T						
T'						
F						

构造过程：

1.判断文法是否为LL(1)文法

2.构造预测分析表

加法表达式文法:

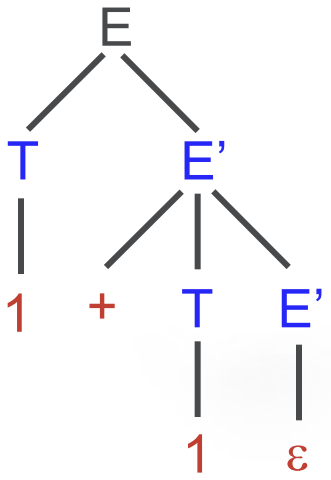
E→TE'

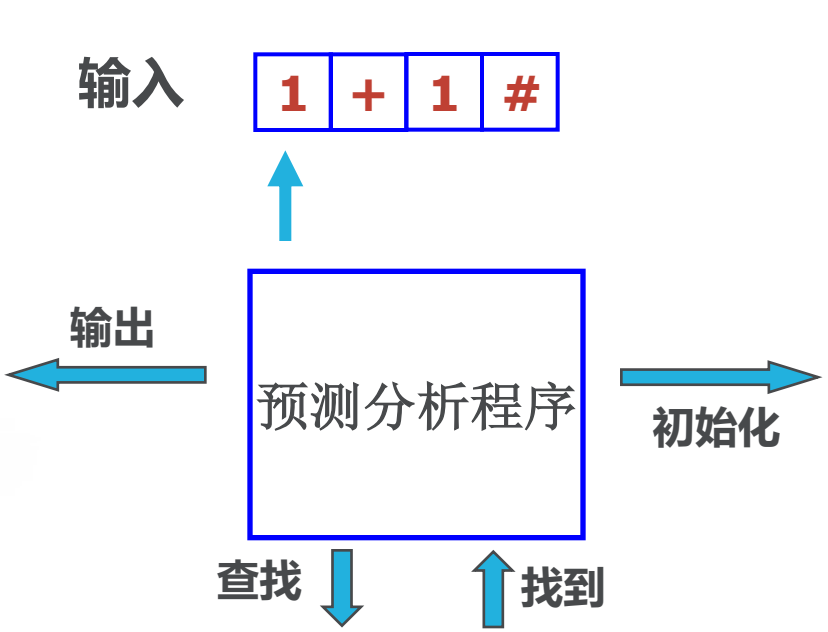
E'→+TE' |ε

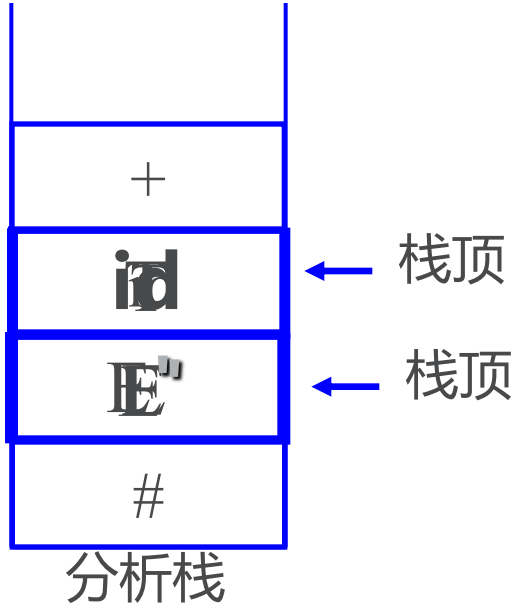
T→id

成功？

失败？







	id	+	#
E	→TE'	不应为+	无效句子
E '	不应为终结符	→+TE'	→ε
T	→id	不应为+	无效句子

例

对输入串 i+i*i的分析

	i	+	*	(	)	#
E	→TE'			→TE'		
E'		→+TE'			→ ε	→ ε
T	→FT'			→FT'		
T'		→ ε	→*FT'		→ ε	→ ε
F	→i			→(E)		

分析栈	剩余串	产生式
#E	i+i*i#	E→TE'
#E'T	i+i*i#	T→FT'
#E'T'F	i+i*i#	F → i
#E'T'i	i+i*i#	i 匹配
#E'T'	+i*i#	T' → ε
#E'	+i*i#	E' → +TE'
#E'T+	+i*i#	+ 匹配
#E'T	i*i#	T →FT'
#E'T'F	i*i#	F → i
#E'T'i	i*i#	i 匹配
#E'T'	*i#	T → *FT'
#E'T'F*	*i#	* 匹配
#E'T'F	i#	F → i
#E'T'i	i#	i 匹配
#E'T'	#	T' → ε
#E'	#	E' → ε
#	#	接受

表驱动 LL（1）分析程序

✧ 表驱动预测分析过程举例

– 对于下列文法*G*(*S*)：

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

剩余的输入串

aabd#

S
#

分析输入串 *aabd* 的过程：

表驱动 LL（1）分析程序

✧ 表驱动预测分析过程举例

– 对于下列文法*G*(*S*)：

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

剩余的输入串

aabd#

A
a
S
#

分析输入串 ***aabd*** 的过程：

	a	b	c	d	#
S	*S→AaS*	*S→BbS*	*S→BbS*	*S→d*	
A	*A→a*				
B		*B→ε*	*B→c*		

表驱动 LL（1）分析程序

✧ 表驱动预测分析过程举例

– 对于下列文法*G*(*S*)：

$S \to AaS \mid BbS \mid d$

$A \to a$

$B \to \varepsilon \mid c$

剩余的输入串

aabd#

a
a
S
#

分析输入串 ***aabd*** 的过程：

	a	b	c	d	#
S	*S→AaS*	*S→BbS*	*S→BbS*	*S→d*	
A	*A→a*				
B		*B→ε*	*B→c*		

表驱动 LL（1）分析程序

✧ 表驱动预测分析过程举例

– 对于下列文法*G*(*S*)：

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

剩余的输入串

abd#

a
S
#

分析输入串 *aabd* 的过程：

	a	b	c	d	#
S	*S*→*AaS*	*S*→*BbS*	*S*→*BbS*	*S*→*d*	
A	*A*→*a*				
B		*B*→ε	*B*→*c*		

表驱动 LL（1）分析程序

✧ 表驱动预测分析过程举例

– 对于下列文法*G*(*S*)：

$S \to AaS \mid BbS \mid d$
$A \to a$
$B \to \varepsilon \mid c$

剩余的输入串
bd#

S
#

分析输入串 ***aabd*** 的过程：

	a	b	c	d	#
S	*S*→*AaS*	*S*→*BbS*	*S*→*BbS*	*S*→*d*	
A	*A*→*a*				
B		*B*→ε	*B*→*c*		

表驱动 LL（1）分析程序

✧ 表驱动预测分析过程举例

– 对于下列文法*G*(*S*)：

$S \to AaS \mid BbS \mid d$
$A \to a$
$B \to \varepsilon \mid c$

剩余的输入串
bd#

B
b
S
#

分析输入串 ***aabd*** 的过程：

	a	b	c	d	#
S	*S→AaS*	*S→BbS*	*S→BbS*	*S→d*	
A	*A→a*				
B		*B→ε*	*B→c*		

表驱动 LL（1）分析程序

✧ 表驱动预测分析过程举例

– 对于下列文法*G*(*S*)：

$S \to AaS \mid BbS \mid d$

$A \to a$

$B \to \varepsilon \mid c$

剩余的输入串

bd#

b
S
#

分析输入串 ***aabd*** 的过程：

	a	b	c	d	#
S	*S→AaS*	*S→BbS*	*S→BbS*	*S→d*	
A	*A→a*				
B		*B→ε*	*B→c*		

表驱动 LL（1）分析程序

✧ 表驱动预测分析过程举例

– 对于下列文法*G*(*S*)：

$S \rightarrow AaS \mid BbS \mid d$

$A \rightarrow a$

$B \rightarrow \varepsilon \mid c$

剩余的输入串

d#

S
#

分析输入串 ***aabd*** 的过程：

	a	b	c	d	#
S	*S→AaS*	*S→BbS*	*S→BbS*	*S→d*	
A	*A→a*				
B		*B→ε*	*B→c*		

表驱动 LL（1）分析程序

✧ 表驱动预测分析过程举例

– 对于下列文法*G*(*S*)：

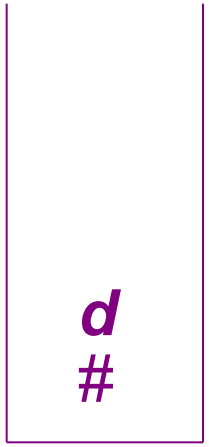
$S \to AaS \mid BbS \mid d$

$A \to a$

$B \to \varepsilon \mid c$

剩余的输入串

d#



分析输入串 ***aabd*** 的过程：

	a	b	c	d	#
S	*S→AaS*	*S→BbS*	*S→BbS*	*S→d*	
A	*A→a*				
B		*B→ε*	*B→c*		

表驱动 LL（1）分析程序

✧ 表驱动预测分析过程举例

– 对于下列文法*G*(*S*)：

$S \to AaS \mid BbS \mid d$

$A \to a$

$B \to \varepsilon \mid c$

剩余的输入串

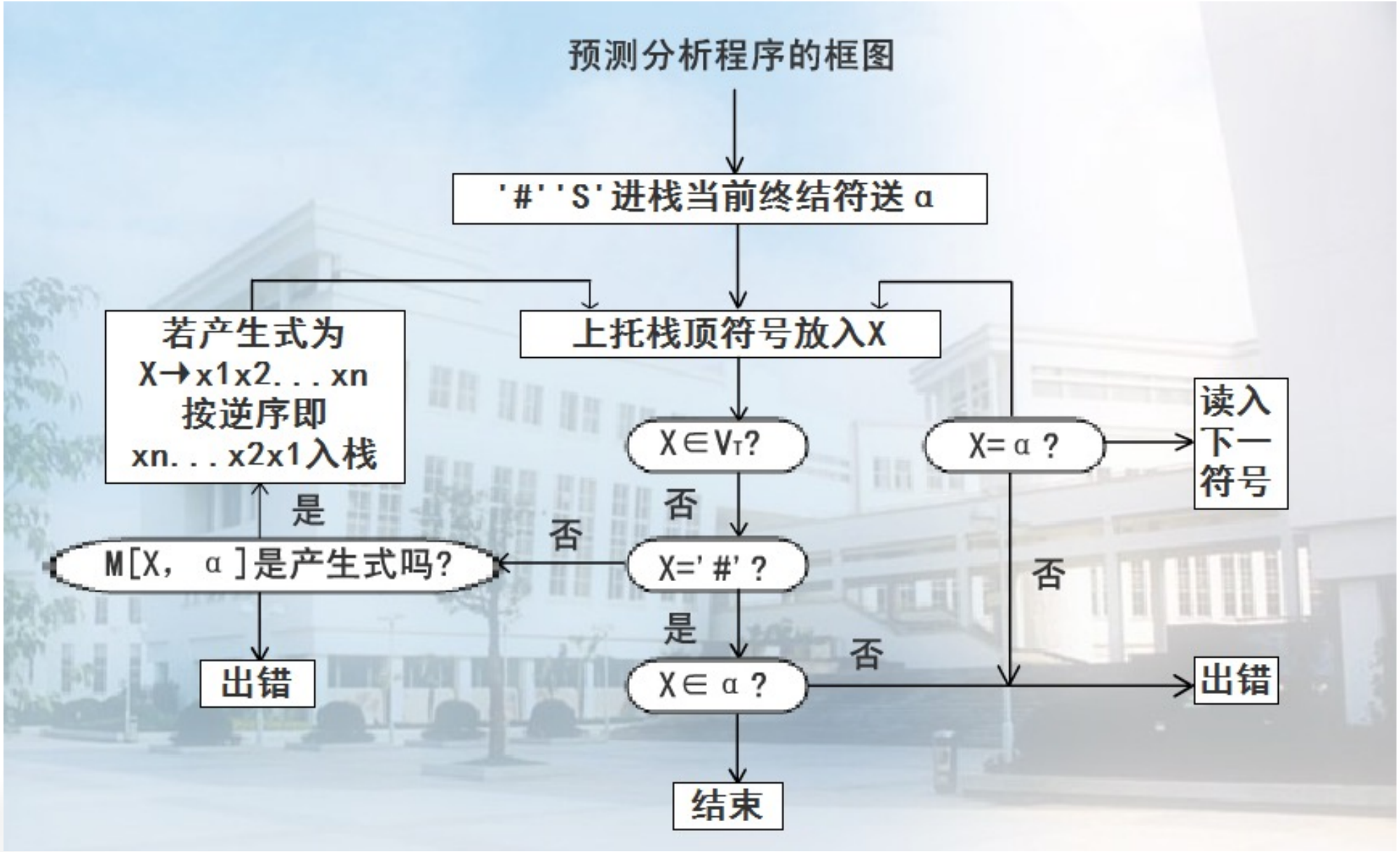
#

#

分析输入串 ***aabd*** 的过程：

	a	b	c	d	#
S	*S*→*AaS*	*S*→*BbS*	*S*→*BbS*	*S*→*d*	
A	*A*→*a*				
B		*B*→ε	*B*→*c*		

预测分析程序算法


预测分析程序的框图
'#''S'进栈当前终结符送α
上托栈顶符号放入X
若产生式为
X→x1x2...xn
按逆序即
xn...x2x1入栈
X∈VT?
X=α？
读入
下一
符号
是
否
否
M[X, α]是产生式吗?
X='#'？
否
是
出错
X∈α？
否
出错
结束

✧思考题：

1 如何对一个给定的文法判断是否是LL(1)文法；
2 如何构造预测分析表；
3 如何用预测分析方法判断给定的输入符号串是否是该文法的句子；
4 是不是对所有的非LL(1)文法做等价变换后可能变成LL(1)文法，如果是怎么变换？如果不是，那需要什么条件？

LL(*K*)文法- 推广LL(1)文法

课堂练习

已知文法G【S】：**S→aH**

H→aMd|d

M→Ab| ε

A→aM| e

1.判断文法是否为LL(1)文法,若是，请构造预测分析表。

2.请给出输入串aaabd#的预测分析过程，并说明该输入串是否是G[S]的句子。

正规式考题：

- 1．设有语言 L = { a | a {0,1} + ，且a不以 0 开头，但以 00 结尾 }。
- （1）试写出描述 L 的正规表达式。（2分）
- （2）构造识别 L 的 最小的DFA（要求先画出构造过程中的NFA，再转化为DFA。如果转化得到的DFA不是最小的DFA，则将其最小化)。（8分）
- （3）根据（2）构造的DFA，写出相应的正规文法。（4分）