讲稿（教学内容、步骤）

第 3 章 词法分析（1）

1. 词法分析（lexical analysis）

逐个读入源程序字符，输出“单词符号”，供语法分析使用。

- 主要任务：读源程序，产生**单词符号**
- 其他任务：① 滤掉空格，跳过注释、换行符；②追踪换行标志，复制出错源程序；③宏展开，……

(1) 单词符号：一般可分为下列五种

标识符：各种名称，如常量名、变量名、过程名

常数：25, 3.1415, TRUE, “ABC”等

关键字(保留字)：begin, end, if, while

运算符：如 + - * / < <=等

界符：逗号，分号，括号等

(2) 词法分析的输出形式

二元式（单词种类，单词自身的值）

【举例】 if i<5 then x=y;

单词	二元式
关键字 if	(3, ‘if’)
标识符 i	(1, 指向i的符号表入口)
小于号 <	(4, ‘<’)
常数 5	(2, ‘5’)
关键字 then	(3, ‘then’)
标识符 x	(1, 指向x的符号表入口)
赋值号 =	(4, ‘=’)
标识符 y	(1, 指向y的符号表入口)
分号 ;	(5, ‘;’)

(3) 词法分析程序与语法分析程序的接口方式

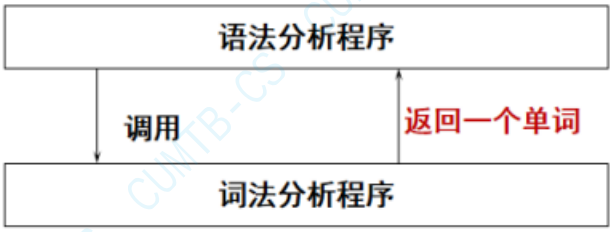
- 方式一（常用）：

优点：

（1）整个编译结构简洁、清晰、条理化

（2）可移植性好

- 方式二（PL/0 采用）：

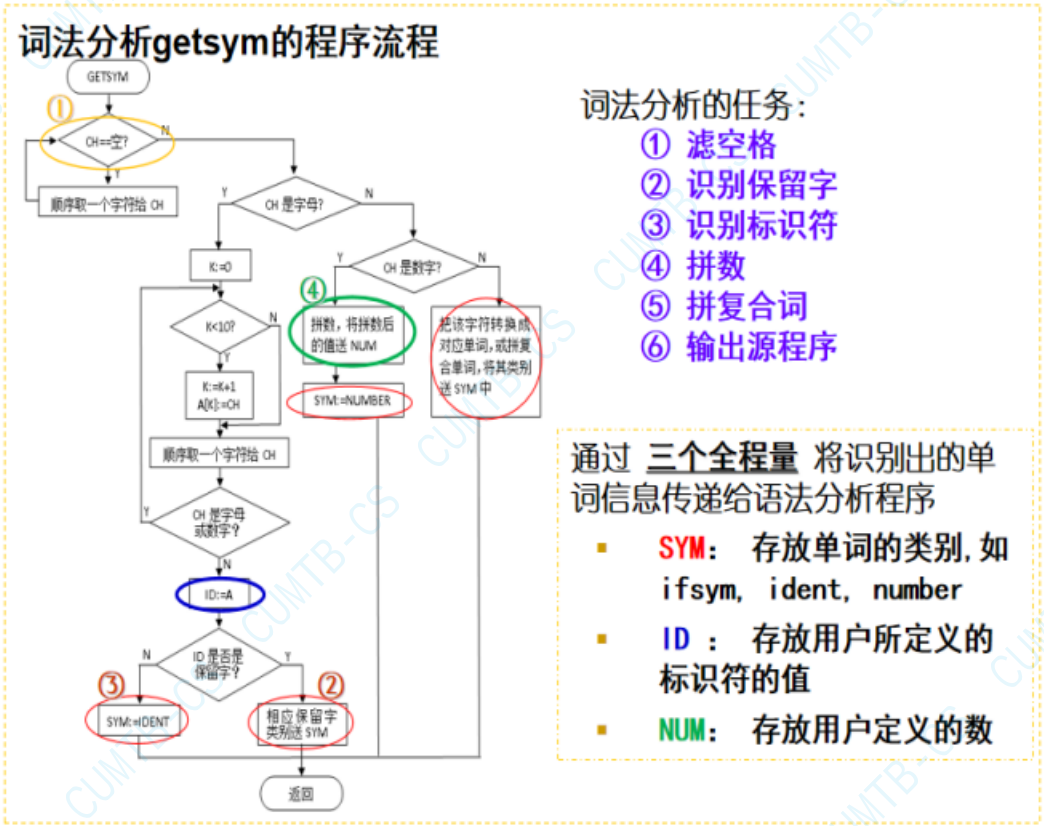


2. PL/0 编译器的词法分析

PL/0 的词法分析是一个独立的函数，调用一次，就按序返回一个单词的信息。

- 输出信息存放在三个全局变量中：sym，ident，num
- 词法分析函数：

int getsym()　　参见 P42-43 和第 1 章流程图

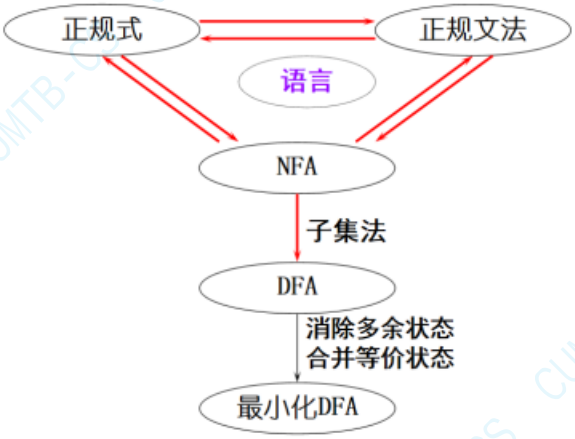


- 验证方法：

```
void    main( )
{    打开源程序文件
     while(…)
     {          getsym( );
                printf(…, sym, ident, num);
     }
     关闭源程序文件;
}
```

3. 单词的形式化描述工具和识别工具：
   (1) 正规文法（正则文法、3 型文法）
   (2) 正规式（正则式）
   (3) 有穷自动机

   三者之间可以相互转换



4. 正规文法（3 型文法，正则文法）

文法中每个产生式的形式：

- 右线性（A→aB 或 A→a）

或

- 左线性（A→Ba 或 A→a）

其中 A，B∈V_N ，a∈V_T^*

【举例】

- 标识符的正规文法：（若 i 表示任一字符，d 表示任一数字）
  〈标识符〉→i〈字母数字〉
  〈字母数字〉→ ε |i〈字母数字〉| d〈字母数字〉

- 无符号整数的正规文法：
  〈无符号整数〉→ d | d〈无符号整数〉

- 运算符的正规文法：
  〈运算符〉→ +|-|*|/|=|<〈等号〉| >〈等号〉……
  〈等号〉→ =

- 界符的正规文法：
  〈界符〉→ , | ; | (|) |……

5. **正规式（正则表达式）**：也是一种描述单词符号串规则的工具，即表示正规集的工具。

   设 字母表为Σ

   辅助字母表Σ′ = { Φ，ε，|，•，*，(，) }

   - * 表示“闭包”，即任意有限次的自重复连接

- • 表示“连接”，有时可以省略
- | 表示“或”

优先顺序为 ()、*、•、|

*、•、|都是左结合的

则

(1) Φ和ε都是Σ上的正规式

(2) 任何 a∈Σ，a 是Σ上的正规式

(3) (e1)、e1|e2、e1e2、e1* 都是Σ上的正规式（e1 和 e2 表示Σ上的正规式）

(4) 仅由有限次使用上述 3 步定义的表达式才是Σ上的正规式

【举例】

例 1：令 Σ={0，1}， Σ上正规式和相应正规集的例子有：

正规式	正规集
0	{0}
0|1	{0,1}
01	{01}
(0|1)(0|1)	{00,01,10,11}
0*	{ε,0,0, ……任意个 0 的串}
(0|1)	{ε,0,1,00,01 ……所有由 0 和 1 组成的串}
(0|1)*(00|11)(0|1)*	{Σ上所有含有两个相继的 0 或两个相继的 1 组成的串}

6. 两个正规式等价

若两个正规式 e1 和 e2 所表示的正规集相同，

则说 e1 和 e2 等价。

记作 e1 = e2

【举例】0|1 = 1|0

1(01)*= (10)*1

(0|1)*= (0*|1*)*

7. 正规式的代数运算

设 r, s, t 为正规式，则有：

r|s=s|r		“或”的交换律
r|(s|t)=(r|s)|t		“或”的结合律
(rs)t=r(st)		“连接”的可结合律
r(s|t)=rs|rt	(s|t)r=sr|tr	分配律
εr=r	rε=r	ε是“连接”的恒等元素

【举例】

程序中的单词符号都能用正规式表示：

e = <字母>(<字母>|<数字>)*

8. 正规式转换为等价的正规文法

① 任何正规式 r：定义S为开始符号　S→r

② 转换规则

正规式	正规文法
A→xy	A→xB B→y
A→x*y	A→xA A→y
A→x|y	A→x A→y

③ 不断用上述规则进行变换，直到每个产生式的右部只含一个终结符为止。

【举例】

例：正规式 r = 0(0|1)*　，写出对应的正规文法。

S→0 (0|1)*

S→0A
A→(0|1)*

S→0A
A→(0|1)A
A→ε

S→0A
A→0A|1A
A→ε

S→0A
A→0A
A→1A
A→ε

【练习】 r = (01|10)*(0|1)，写出对应的正规文法。

练习：r = (01|10)*(0|1)

S→01S
S→10S
S→0
S→1

S→0A
A→1S
S→1B
B→0S
S→0
S→1

S→0A
S→1B
S→0
S→1
A→1S
B→0S

9. 正规文法转换为等价的正规式

① 等价转换的规则

正规文法	正规式
A→xB B→y	A = xy
A→xA A→y	A = x*y
A→x A→y	A = x | y

② 不断用上述规则进行变换，直到最后只剩一个开始符号为止。

【举例】

例：正规文法 G[S]:　S⟶ aA ， 写出等价的正规式。

S⟶a

A⟶aA

A⟶dA

A⟶a

A⟶d

S→aA S→a A→aA A→dA A→a A→d	S = aA|a A = aA|dA A = a|d	S = a(A|ε) A = (a|d)A A = a|d	S = a(A|ε) A = (a|d)*(a|d) S = a((a|d)*(a|d)|ε) S = a(a|d)*

【练习】课后习题 8

G[S]:　S⟶0A

S⟶1B

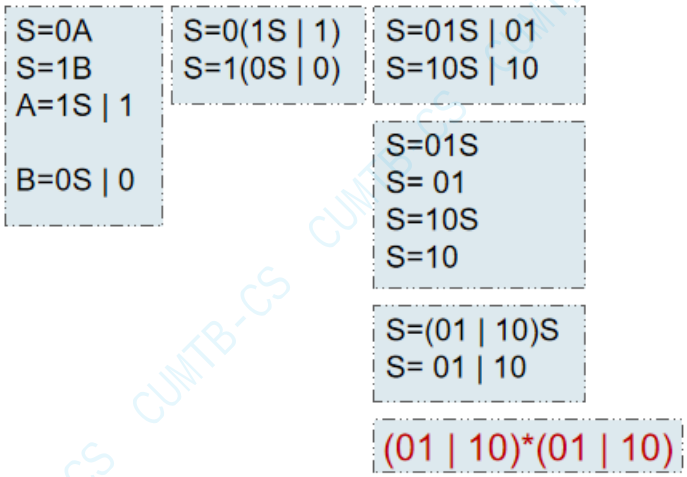
A⟶1S

A⟶1

B⟶0S

B⟶0

给出对应的正规式。



参考答案：(01 | 10)*(01 | 10)
(01 | 10)(01 | 10)*

10. 有穷自动机(FA，Finite Automata)

- 有穷自动机：是一个识别装置，用于识别“所有句子”。
- 引入 FA 的目的：
  为词法分析程序的自动构造寻找特殊的方法和工具
- 类型：
  - ✓ 确定的有穷自动机 DFA (Deterministic Finite Automata)
  - ✓ 不确定的有穷自动机 NFA (Nondeterministic Finite Automata)

11. DFA

(1) 定义：一个 DFA 是一个五元组 $M=(K,\Sigma,f,S,Z)$

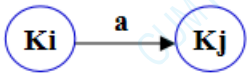
K：有穷的状态集
Σ：有穷的字母表（即输入符号的集合）
f：转换函数 $K\times\Sigma\rightarrow K$ 上的映像
S：初态（初态唯一）
Z：终态集（终态不唯一）

【举例】DFA M =({S,U,V,Q} , {a,b} , f , S , {Q})

f：
f(S,a)=U　　f(S,b)=V
f(U,a)=Q　　f(U,b)=V
f(V,a)=U　　f(V,b)=Q
f(Q,a)=Q　　f(Q,b)=Q

(1) DFA 的“直观”表示

- 状态图（状态转换图）
  - ✓ 每个状态用结点表示

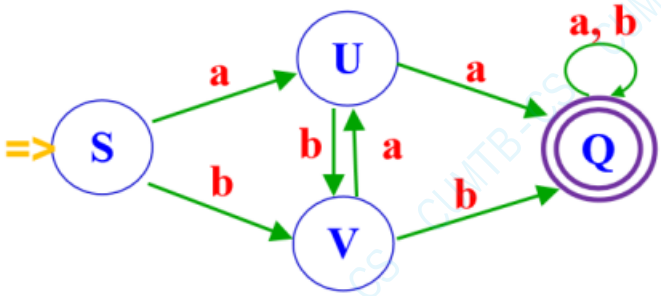
- ✓ 若 f(Ki , a) = Kj，则 

- ✓ 初态用“=>” 或 “-” 标出
  终态用 双圈 或 “+” 标出

- 矩阵
  - ✓ 列标题：输出符号　　行标题：状态
  - ✓ 若 f(Ki , a) = Kj，则 Ki 和 a 的交汇处是 Kj
  - ✓ 初态用“=>” 标出 或 默认第一行（表格左端）
    终态用“1”标出（表格右端）
    非终态用“0”标出（表格右端）

【举例】

上例对应的状态图（动画演示）：



矩阵（动画演示）：

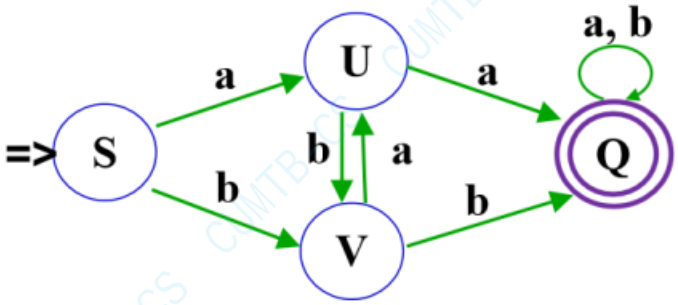
	a	b	
=> S	U	V	0
U	Q	V	0
V	U	Q	0
Q	Q	Q	1

(2) DFA 可以接受的句子（符号串）：

- 若 t∈Σ*，且存在 f(S,t)= … = Q，Q∈终态集，则 t 为该 DFA 可以接受的句子。

即：从初态 S 到某终态结点 Q 的道路上，所有弧上的标记符连接而成字符串 t，t 为该 DFA 可以接受的句子。

【举例】



例：判断下列句子能否被该DFA接受：

abba	baab	abb	aa	a
接受	接受	接受	接受	不

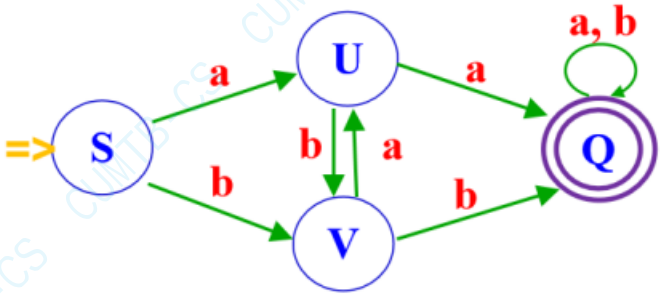
- DFA M 能够接受的句子的全体记为 L(M)

(3) DFA 的确定性

f: K×Σ→K 是一个单值函数

对任何状态 K，当输入一个字符时，下一状态唯一。

【举例】DFA：



(4) DFA M=（K， Σ，f，S，Z）的行为模拟程序：

```
K:=S;
c:=getchar;
while (c<>eof)
{       K:=f(K,c);
        c:=getchar;
}
if (K in Z)
{       return (‘yes’);  }
else
{       return (‘no’);       }
```

12. NFA

(1) 定义：一个 NFA 是一个五元组 M=(K ,Σ,f ,S ,Z)

K：有穷的状态集

Σ：有穷的字母表（即输入字符的集合）

f：转换函数 K×Σ*→K* 上的映像（K* 表示 K 的子集）

S：初态集（初态不唯一）

Z：终态集

【举例】NFA M'=({0,1,2,3,4} , {a,b} , f ,{0} ,{2,4})

f： f(0,a)={0,3} f(0,b)={0,1}

f(1,b)={2}

f(2,a)={2} f(2,b)={2}

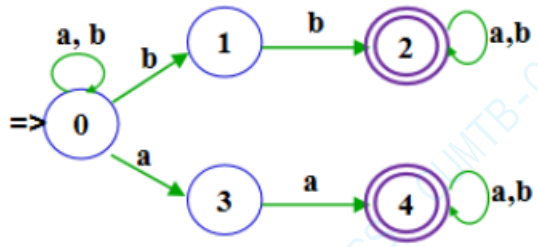
f(3,a)={4}

f(4,a)={4} f(4,b)={4}

(2) NFA 的"直观"表示

【举例】

状态图（状态转换图）



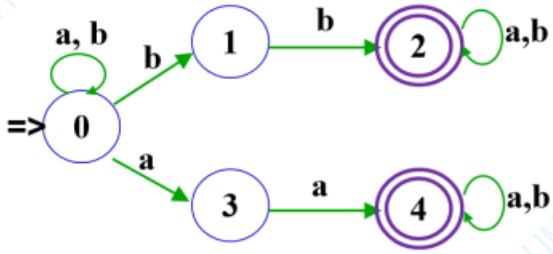
矩阵

	a	b	
=> 0	0, 3	0, 1	0
1		2	0
2	2	2	1
3	4		0
4	4	4	1

(3) NFA 可以接受的句子（符号串）

- 若 t∈Σ*，且存在 f(S,t)= … = Q，Q∈终态集，则 t 为该 NFA 可以接受的句子。

【举例】



判断下列句子能否被上面的NFA接受：

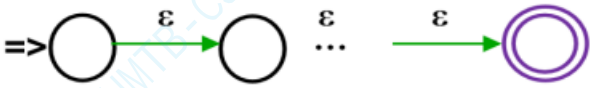
aaa	baab	abb	a	aba	bab
接受	接受	接受	不接受	不接受	不接受

- NFA M' 能够接受的句子的全体记为 L(M')

(4) ε可以被 NFA 能够接受的两种情况

- 情况 1：某结点既是初态，又是终态

- 情况 2：存在一条从初态到终态的ε道路



【讨论】如下哪个 NFA 可以接受ε？为什么？

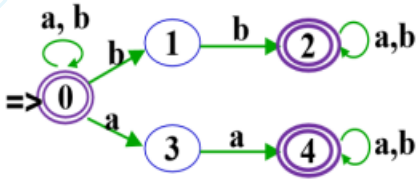


图 1

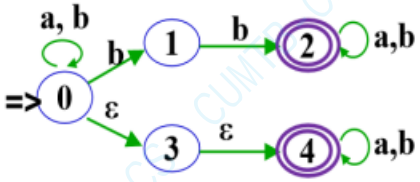
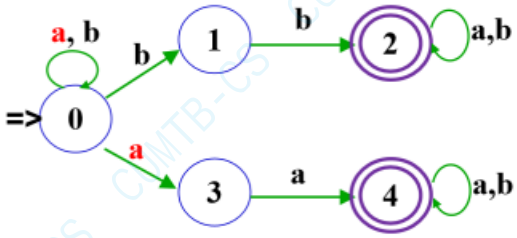


图 2

(5) NFA 的不确定性

对于状态K，当输入同一字符时，下一状态不一定唯一。



(6) NFA 的确定化

- 对于每个 NFA M’ 存在一个 DFA M，使得 L(M’)= L(M)
  注：与某一 NFA 等价的 DFA 不唯一。
- 找到与 NFA 等价的 DFA 的方法，是“子集法”。