第3章 词法分析及词法分析程序

- 主讲： 王慧娇
- 办公室： 金鸡岭3301-1
- 电话：13978321977
- QQ：248886622
- Email：whj7667@qq.com

- 答疑地点： 5507
- 辅导时间： 周三1、2节

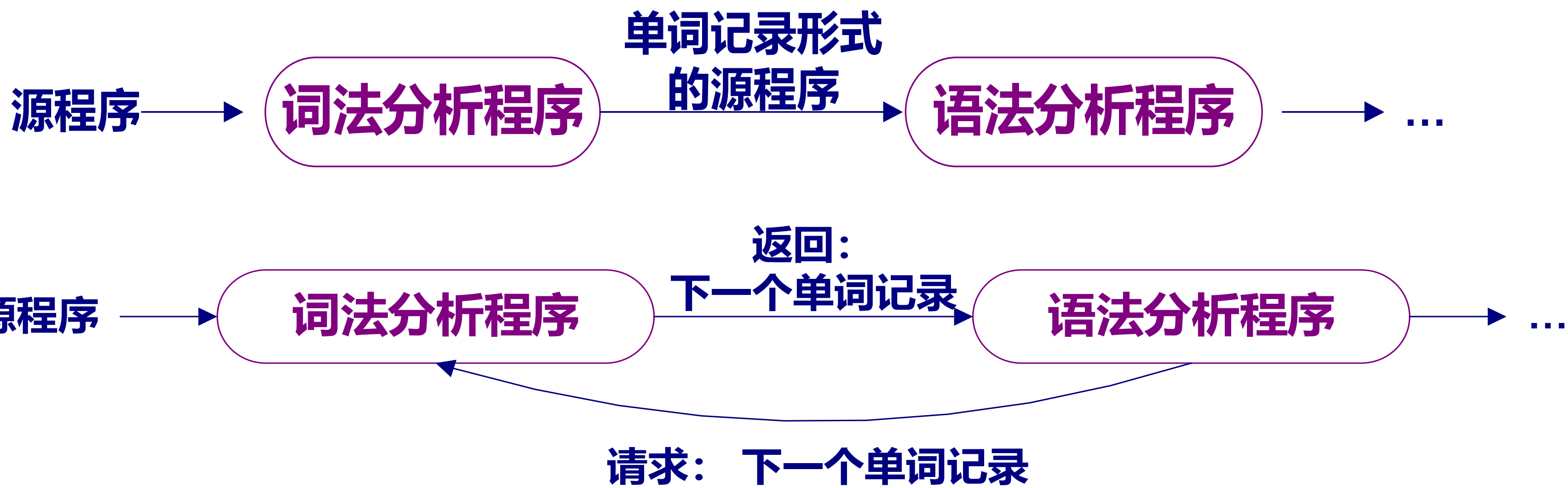
主要内容

- 词法分析器(Lexical Analyzer, Scanner)的功能
- 正规表达式
- 正规文法
- 有限状态自动机FA——状态图
- 词法分析器的设计与实现

3.1 词法分析概述
3.1.1词法分析与语法分析的接口

编译程序主体中如何组织词法分析程序

- 可以作为单独的一遍
- 较常用的方式是由语法分析程序调用
- 基本任务都是识别单词



3.1.2 词法分析（扫描）器功能

- 词法分析程序（Lexical Analyzer）或词法扫描程序（Scanner）的作用
  - 从左至右扫描构成源程序的字符流
  - 识别出有词法意义的单词（Lexemes）
  - 返回单词记录（由单词记号（Token）和单词的属性值组成），或词法错误信息
  - 除以上主要任务外，常伴有如下任务

    滤掉空格，跳过注释、换行符，追踪换行标志，复制出错源程序，宏展开，……

    也可能包含访问符号表的操作

3.1.2 词法分析（扫描）器输出

- 单词符号的形式
  - 按照最小的语义单位设计
  - 通常表示为二元组：

（单词符号种别，属性值）

- 例：Pascal 程序文本

position := initial + rate * 60;

经词法分析程序处理后，转换为下列单词序列

3.1.2 词法分析（扫描）器输出

词法单元（Token）	单词属值
标识符	position
赋值算符(:=)	
标识符	initial
加算符(+)	
标识符	rate
乘算符(*)	
整数常量	60
分号(；)	

单词符号的表示

- 常用单词符号类别——分类
- 各关键字（保留字、基本字），各种运算符，各种分界符——各用一个类别码标识
  - 其它标识符——用一个类别码标识
  - 常数——用一个类别码标识
- 属性（值）——单词符号的值
  - 常数的值，标识符的名字等
  - 保留字、运算符、分界符的属性值可以省略

单词符号编码举例

单词符号	种别编码	内部值	助记符
BEGIN	1		$BEGIN
END	2		$END
IF	3		$IF
THEN	4		$THEN
ELSE	5		$ELSE
标识符	6	内部符号串	$IDN
整数	7	标准二进制	$ INT
=	8		$ASG
+	9		$PLUS
*	10		$STAR
>	11		$GT
<	12		$LT
（	13		$SLP
）	14		$SRP

3.1.3 词法分析作为单独一个阶段

- 好处：
  - 使整个编译程序的结构更加简洁、清晰和调理化
  - 编译程序的效率会改进
  - 增强编译程序的移植性

3.1.4 单词的描述与识别

- 如何描述单词的结构、如何识别单词
- 1、正规文法（正规式）
  表示单词的构词规则
- 2、有限自动机
  实现词法分析器，识别单词
- 3、词法分析器自动生成器
  使用单词的正规式表示作为输入

3.2 词法分析程序的设计与实现

实例：某语言词法分析程序的设计

– 单词类别（种别）的 BNF 描述

<无符号整数> ::= <数字> {<数字>}

<标识符> ::= <字母> {<字母> | <数字>}

<字母> ::= *a* | *b* | … | *X* | *Y* | *Z*

<数字> ::= 0 | 1 | 2 | … | 8 | 9

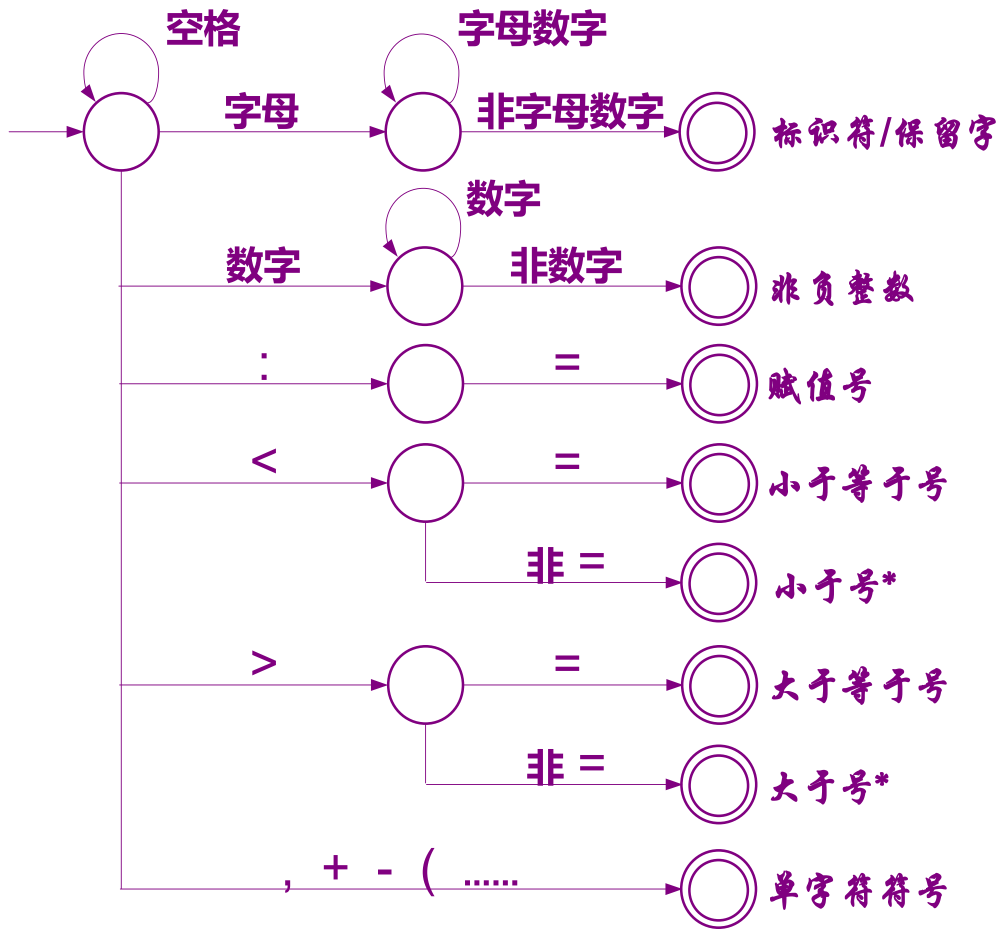
<保留字> ::= const | var | procedur | begin | end | odd
| if | then | call | while | do | read | write

<运算符> ::= + | - | * | / | = | # | < | <= | > | >= | :=

<界符> ::= (|) | , | ; | .

3.2 词法分析程序的设计与实现

– 词法规则的状态转换图



3.3 单词的形式化描述工具

- 3.3.1 正规文法(RG)
  - 文法中的产生式都具有形如以下的产生式
    $A \to aB$ 或 $A \to a$　A、B∈V_N，a∈$V_T \cup \{\varepsilon\}$
    为右线性(Right Linear)文法。
  - 或 $A \to Ba$ 或 $A \to a$　A、B∈V_N，a∈$V_T \cup \{\varepsilon\}$
    为左线性(Right Linear)文法。
  - 左线性和右线性文法都是3型文法(正规文法 Regular Grammar -RG)
- 例：标识符的正规文法
  - ＜标识符＞→letter| letter ＜标识符＞|digit＜标识符＞

3.3.2 符号的描述——正规(表达)式

- 正规式(Regular Expression——RE)：正规语言的另一种描述方法
- 例：标识符的正规表达式
  - letter (letter | digit)*
  - | 表示“或”运算
  - * 表示Kleene闭包运算
  - •表示连接运算，可以省略•
- 正规集：用 r 表示正规式,对应的语言的正规集记为 L(r)

3.3.2 符号的描述——正规(表达)式

1.正规式与正规集的定义

设∑是一个字母表，（归纳基础）

- (1) Φ是∑上的正规式，则正规集L(Φ)=Φ；
- (2) ε是∑上的正规式，则正规集L(ε)={ε}；
- (3) 对于∀a∈∑，a是正规式，则正规集L(a)={a}；

  （以下为归纳步骤）

- (4)如果r和s是正规式，L(r)=R，L(s)=S，则：

  (r|s)是正规式，L(r|s)=R∪S；

  (rs)是正规式，L(rs)=RS；

  (r^*)是正规式，L(r^*)=R^*。

- (5) 只有满足(1)、(2)、(3)、(4)的才是正规式。

3.3.2 符号的描述—正规(表达)式

2.运算优先级和结合性:

- 一元运算符*优先级最高，左结合
- “连接”·高于|，左结合，具有结合律、和对|的分配律
- | 运算优先级最低，左结合，具有交换律、结合律
- ( ) 指定优先关系
- 意义清楚时，括号可以省略

例: (a) |((b)*(c)) 改写为a|b*c

3.3.2 符号的描述—正规(表达)式

3.正规式与正规集举例

正规式	正规集
a*	$\bigcup_{i=0}^{\infty}\{a^i\}=\{\varepsilon,a,aa,aaa,\cdots\}$
aa*	{a,aa,aaa,...}
a|b	{a,b}
a|ba*	{a,b,ba,baa,baaa,⋯}
(a|b)*abb	任何以abb为结尾的a,b符号串
(aa|ab|ba|bb)*	空串和任何长度为偶数的a,b符号串
(a|b)(a|b)(a|b)*	任何长度大于等于2的a,b符号串

3.3.2 符号的描述—正规(表达)式

4.正规表达式公理

若两个正规式所表示的正规集相同，则认为二者等价

A1. $r|s=s|r$

A2. $r|r=r$

A3. $r|\phi=r$

A4. $(r|s)|t=r|(s|t)$

A5. $(rs)t=r(st)$

A6. $r(s|t)=rs|rt$

A7. $(s|t)r=sr|st$

A8. $r\phi=\phi r=\phi$

A9. $r\varepsilon=\varepsilon r=r$

A10. $r^*=(\varepsilon|r)^*=\varepsilon|rr^*$

- 正规文法与正规式等价
  - 对任何正规文法，存在定义同一语言的正规式
  - 对任何正规式，存在生成同一语言的正规文法

3.3.2 符号的描述——正规(表达)式

- 5.正规式和正规文法示例
- 描述下列正规式定义的语言（正规集）
  - (1) a(a|b)*a
  - (2) a*ba*ba*ba*
  - (3) (a|b)*a(a|b) (a|b)
- 试写出下列语言的正规式定义
  - (1)标识符
  - (2)所有的二进制奇数的集合

3.3.3 由正规文法构造相应的正规式

- 1.转换方法
- 正规文法与正规式等价
  - 对任何正规文法，存在定义同一语言的正规式
  - 对任何正规式，存在生成同一语言的正规文法
- 正规文法转换为相应的正规式
  - 方法：将G视为定义所含非终结符为变量的联立方程组,通过解方程组求得相应的正规式.

3.3.3 由正规文法构造相应的正规式

- 例，对于文法G[S]，求其对应的正规式

$$S \to aS \mid bA \mid b$$

$$A \to aS$$

设非终结符S，A对应的正规集为L_S, L_A

则 $L_S = \{a\} L_S \cup \{b\} L_A \cup \{b\}$

$$L_A = \{a\} L_S$$

由定义可知,$L(G) = L_S$，记‘|’为‘+’，

有 $S = aS + bA + b$ (1)

$A = aS$ (2)

将(2)代入(1),

得 $S = aS + baS + b = (a+ba)S + b$ (3)

3.3.3 由正规文法构造相应的正规式

2.论断3.1 方程$X= rX +t$有形如$X= r^*t$的解(不唯一)

证明：$X= rX +t$ 对应产生式： $X\rightarrow rX$ $X\rightarrow t$

则 $L_x=\{t,rt,rrt,rrrt,\cdots\}$

正规式r^*t对应的正规集为

$L=\{t,rt,rrt,rrrt,\cdots\}$

得证。

$S=(a+ba)S+b$的解为：$S= (a|ba)^*b$

3.3.3 由正规文法构造相应的正规式

- 3.转换举例
- 对于文法G[S]：$S \rightarrow aA$

  $A \rightarrow bA \mid aB \mid b$

  $B \rightarrow aA$

构造对应的正规式。

转换为相应的方程组为

$$\begin{cases} S = aA & (1) \\ A = bA + aB + b & (2) \\ B = aA & (3) \end{cases}$$

解方程组，过程为：

将(3)代入(2)中 $A = (b+aa)A+b$

根据论断3.1得 $A = (b+aa)^*b$　　(4)

将(4)代入(1)中 $S = a(b+aa)^*b = a(b \mid aa)^*b$

3.3.3 由正规文法构造相应的正规式

- 例，文法G[S]

  S→bS|aA
  A→aA|bB
  B→aA|bC|b
  C→bS|aA

求解正规式。

转换为相应的方程组

$$
\begin{cases}
S=bS+aA & (1)\\
A=aA+bB & (2)\\
B=aA+bC+b & (3)\\
C=bS+aA & (4)
\end{cases}
$$

由(1),(4)得,　　C=S　(5)
将(5)代入(3)中, B=S+b　(6)
将(6)代入(2)中, A=S+bb (7)
将(7)代入(1)中, S=(a+b)S+abb
根据论断3.1,　　S=(a+b)abb*
　　　　　　　　　=(a|b)abb*

3.3.3 由正规文法构造相应的正规式

- 4.对于左线性文法的情形
- 论断3.2：方程$X=Xr+t$有形如$X=tr^*$的解。

证明：$X=Xr+t$ 对应产生式：$X \rightarrow Xr$　$X \rightarrow t$

则 $L_X=\{t, tr, trr, trrr, \cdots\}$

正规式 tr^*对应的正规集为

$L=\{t, tr, trr, trrr, \cdots\}$

得证。

3.3.3 由正规文法构造相应的正规式

- 例，文法G[S]
- $S \to Sa|Ab$

  $A \to Ab|Ba$

  $B \to Ab|Ca|a$

  $C \to Sa|Ab$

求解正规式。

转换为相应的方程组

$$\begin{cases} S=Sa+Ab & (1) \\ A=Ab+Ba & (2) \\ B=Ab+Ca+a & (3) \\ C=Sa+Ab & (4) \end{cases}$$

由(1),(4)得， $C=S$ (5)

将(5)代入(3)中, $B=S+a$ (6)

将(6)代入(2)中, $A=S+aa$ (7)

将(7)代入(1)中, $S=Sa+(S+aa)b=S(a+b)+abb$

根据论断3.2, $S=abb\,(a|b)^*$

3.3.4 由正规式构造相应的正规文法

- 对于$A \rightarrow x^{*}y$ 重写为：
- **A→xB**
  A→y
  B→xB
  B→y
- 对于$A \rightarrow xy$ 重写为：
- **A→xB**
  B→y
- 对于$A \rightarrow x|y$ 重写为：
- **A→x**
  A→y

课堂举例：

1. 文法G[S]：$S \rightarrow 0B$

$$B \rightarrow 0B \mid 1S \mid 0$$

求S的正规式R。

2. 设$\Sigma=\{a,b\}$，请写出不以a开头，但以aa结尾的字符串集合的正规表达式。

3.4 有限自动机

- 一、确定有穷状态自动机（DFA）
- 二、非确定有穷状态自动机（NFA）
- 三、NFA和DFA的转换
- 四、具有 ε -动作的NFA
- 五、 ε -动作的NFA的确定化
- 六、DFA的化简

3.4.1确定的有限自动机

DFA:(Deterministic Finite Automata)

1.确定的有限自动机的定义

状态转换图的形式化描述（定义为一个五元组）

DFA M=(K, Σ, f, S_0, Z)

- K：有限个状态的集合；
- Σ：有限个输入符集合；
- f：转换函数，是在K×Σ→K上的映像

  $f(k_i,a)=k_j$ (k_i, $k_j \in K$)：当前状态为k_i，输入符为a时，将转换为下一个状态k_j
- S_0：$S_0 \in K$，初态；
- Z：$Z \subseteq K$，若干个终态之集。

3.4.1确定的有限自动机

DFA:(Deterministic Finite Automata)

- DFA M=({0, 1, 2, 3}, {a, b}, f, 0, {3}),

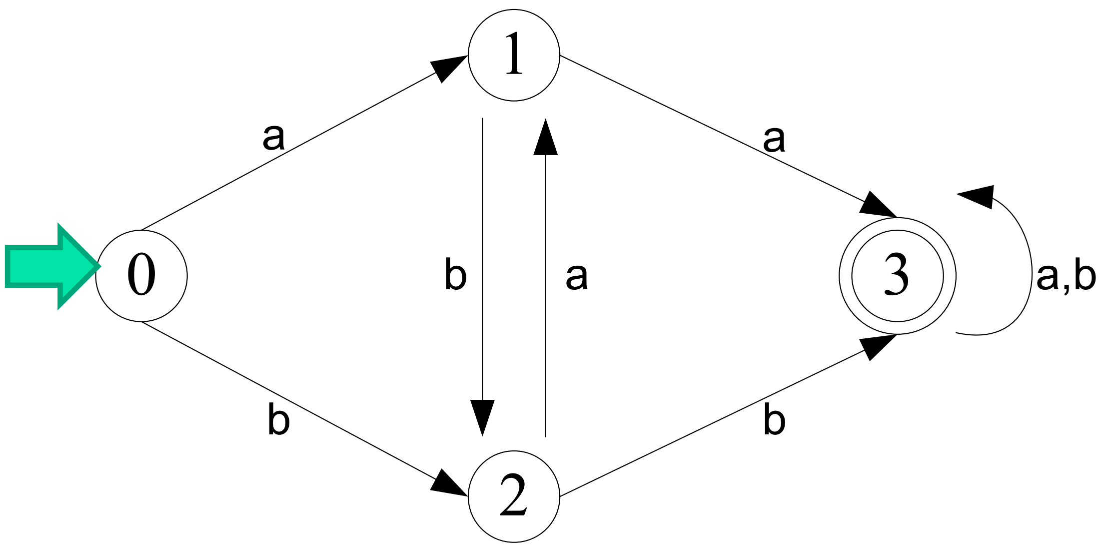
  其中，f:

  f(0,a)=1,f(0,b)=2,

  f(1,a)=3,f(1,b)=2,

  f(2,a)=1,f(2,b)=3,

  f(3,a)=3,f(3,b)=3



3.4.1确定的有限自动机

DFA:(Deterministic Finite Automata)

2.将DFA的转换函数f的定义域拓广到 $\hat{f}$: $K\times\Sigma^*$ 。

(1) $\hat{f}(S,\varepsilon)=S$, $S\in K$;

(2) $\hat{f}(S,aw)=\hat{f}(f(S,a),w)$, $S\in K$, $a\in\Sigma$, $w\in\Sigma^*$;

- 对于$x\in\Sigma^*$,$\hat{f}(S,x)=t$ 的含义是,当自动机M从状态S出发,依次扫描完x的各个符号后将进入状态t.

- 在DFA中，可以由f的定义求出$\hat{f}$，所以并不区分$\hat{f}$与f。

3.4.1确定的有限自动机

DFA:(Deterministic Finite Automata)

- 3. DFA的接受集
- 识别单词的过程：从初态S_0出发，经一恰好标有$x \in \Sigma^*$的路径后可达到某终态$F \in Z$；
- DFA M的接受集： DFA M所接受的符号串的全体称为M的接受集，记为L(M)，即

$$L(M)=\{x \mid f(S_0,x) \in Z, x \in \Sigma^*\}$$

举例
DFA的接受集

- 例：DFA M=（{S，Z，A，B}，{a，b}，f，S，{Z}）

  f（S，a）=A　　f（S，b）=B

  f（A，a）=Z　　f（A，b）=B

  f（B，a）=A　　f（B，b）=Z

  f（Z，a）=Z

  则：f(S, ababaa)=f(f(S,a), babaa) =f(A,babaa)= f(f(A,b), abaa)= f(B,abaa)=f(A,baa)= f(B,aa)=f(A,a)= Z

3.4.1确定的有限自动机

DFA:(Deterministic Finite Automata)

- **DFA识别符号串算法：**

```
s=s0;
c=getnextchar();
while(c!=eof){
   s=move(s,c); //状态转换，计算下一个状态
   c=getnextchar();// 读入下一个符号
   }
 if(s在F中) return “yes”;
   else  return”No”;
```

3.4.1确定的有限自动机

DFA:(Deterministic Finite Automata)

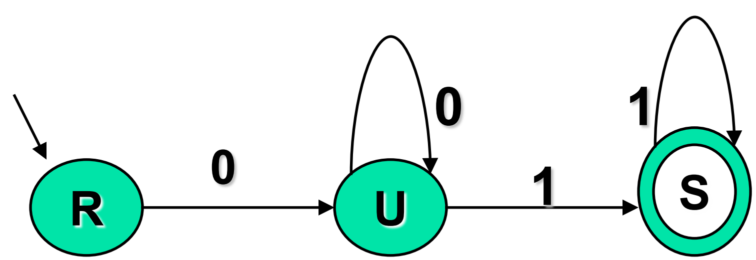
- 4.“确定的”的含义
- 在状态转换的每一步，根据FA当前的状态及扫描的输入字符，便能**唯一地**确定FA的下一状态。
- M=({R,U,S},{0,1},f,R,{S}) 其中，f的定义如下：

  f(R,0)=U

  f(U,0)=U

  f(U,1)= S

  f(S,1)= S



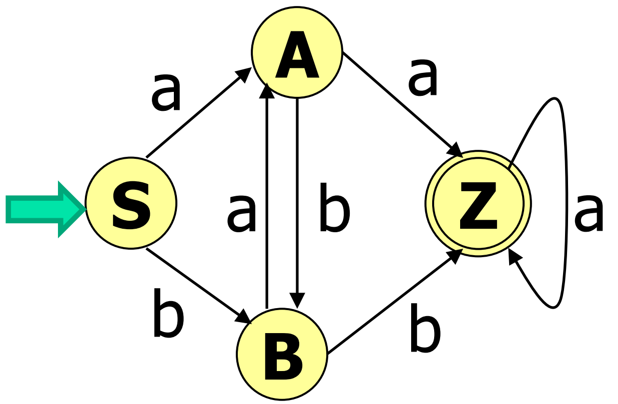
- ∀正规文法G，∃DFA M，使L(M)=L(G)，反之亦然。

3.3.1确定的有限自动机

DFA:(Deterministic Finite Automata)

DFA 的矩阵表示法(状态转换表)

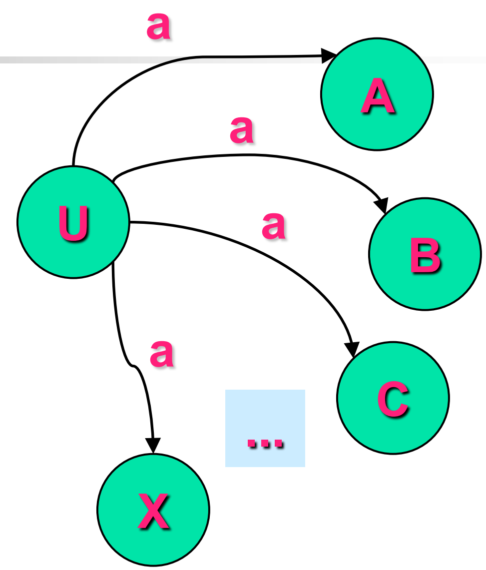
状态 \ 字符	a	b
S	A	B
A	Z	B
B	A	Z
Z	Z	Ø



行代表状态，列代表输入字符，
矩阵元素是映像得到的新状态。

3.4.2 非确定的有限自动机

NFA:(Nondeterministic Finite Automata)



- 1.定义
- 若在一左线性文法中含有多个右部相同的产生式，如 A→Ua B→Ua C→Ua … X→Ua,
- 或在一右线性文法中同时含有形如 U→aA U→aB U→aC … U→aX 的产生式，

由上图可知，在U状态下，输入符号为a时，FA的下一状态不唯一，而是在状态集{A,B,C,…,X}中任选其一。具有这种性质的FA称为非确定的FA（NFA: Nondeterministic FA）

3.4.2 非确定的有限自动机(NFA)

- 形式化定义
- NFA M=（K，Σ，f，S_0，Z），其中
  - K，Σ，S_0，Z 的定义与DFA相同；
  - f：转换函数，K×Σ到K的子集所组成集合的映像，$f(q, a)=\{Q_1,Q_2,\cdots.Q_n\}$，记为：K×Σ到$2^K$的映射。

	DFA	NFA
开始状态	唯一	一个
映像	单个状态	状态集合

3.4.2 非确定的有限自动机(NFA)

- 举例
- NFA M=({S,A,B,Z},{a,b},f,{S},{Z})

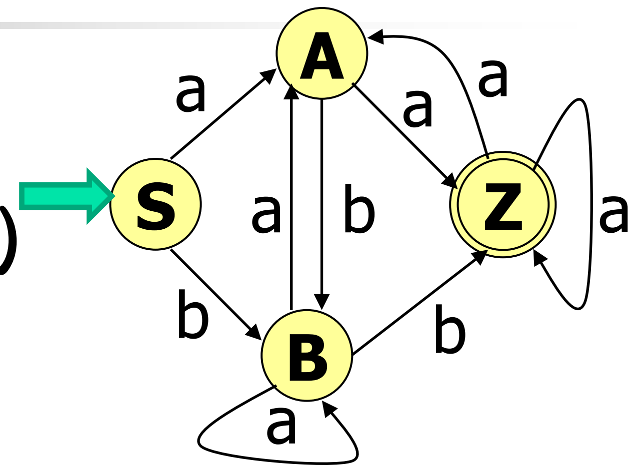
  其中f:

f(S,a)={A}　f(S,b)={B}

f(A,a)={Z}　f(A,b)={B}

f(B,a)={A,B}　f(B,b)={Z}

f(Z,a)={A,Z}



	a	b
S	{A}	{B}
A	{Z}	{B}
B	{A,B}	{Z}
C	{A,Z}	∅

3.4.2 非确定的有限自动机(NFA)

- 2.扩充映像 $\hat{f}: K\times\Sigma^*$
- NFA M 转换函数 f 的定义为 $f: K\times\Sigma\rightarrow 2^k$，即将 (S_i, a_j) 映射到 K 的一个子集 $\{S_{k1},\cdots,S_{km}\}$
- 把 f 的定义域拓广到 $K\times\Sigma^*\rightarrow 2^k$：

  (1) $\hat{f}(S,\varepsilon)=\{S\}$;

  (2) $\hat{f}(S,aw)=\hat{f}(f(S,a),w)$　　$a\in\Sigma, w\in\Sigma^*$.

  再设 $f(S,a)=\{S_{k1},\cdots,S_{km}\}$，且定义

$$\hat{f}(\{S_{k_1},S_{k_2},\cdots,S_{k_m}\},w)=\bigcup_{i=1}^{m}\hat{f}(S_{k_i},w)$$

则有：

$$\hat{f}(S,aw)=\hat{f}(f(S,a),w)=\bigcup_{i=1}^{m}\hat{f}(S_{k_i},w)$$

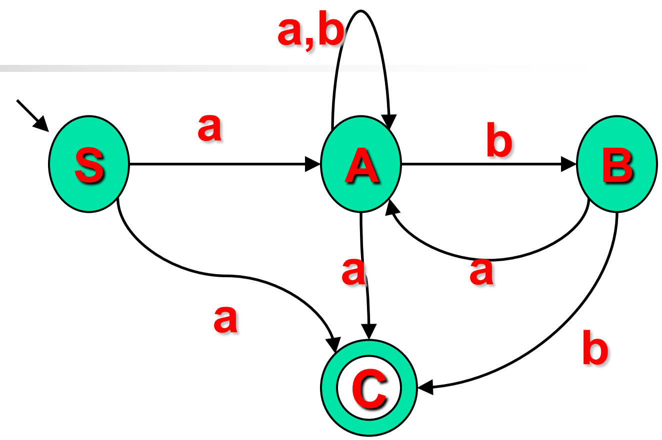
3.4.2 非确定的有限自动机(NFA)

- 3.NFA的接受集
- 对于$x \in \Sigma^*$,若集合$f(S_0,x)$中含有Z中的元素（终态），则说明，至少存在一条从初态S_0到某一终态的路径，此路径上的符号之连接恰为x，此时，我们称x为M所接受.
- 所有为M所接受的符号串之集称为NFA M的接受集（或识别集），记作 L(M).即:

$$L(M)=\{x \mid f(S_0, x) \cap Z \neq \varnothing, x \in \Sigma^*\}$$

3.4.2 非确定的有限自动机(NFA)

- 例：给定M= ({S,A,B,C}, {a,b}, f , S ,{C}), 其状态转换图见右。由图可知M是一NFA。
- M识别符号串ababb的路径为 S(a)→A(b)→B(a)→A(b)→B(b)→C(接受)



步骤	当前状态	输入的其余部分	可能的后继	选择
1	S	ababb	A,C	A
2	A	babb	A,B	B
3	B	abb	A	A
4	A	bb	A,B	B
5	B	b	C	接受

注意，NFA识别输入符号串时有一个试探过程，为了能走到终态，往往要走许多弯路（带回溯），这影响了FA的效率。

能否提高其工作效率就是我们下一小节讨论的课题。

3.4.3 NFA与DFA的转换
NFA的确定化

- NFA状态转换的下一状态不唯一，如何解决？
- 1、确定化的概念

1).确定化：对任给的NFA，都能对应地构造一DFA，它们有相同的接受集

2).确定化原理：令构造出的“新”DFA的状态与“旧”NFA的某一状态子集对应，并使“新”DFA对“旧”NFA的状态转移保持跟踪。

$f(q,a)=\{Q_1,Q_2,\cdots,Q_n\}$ 变为 $f(q,a)=[Q_1,Q_2,\cdots,Q_n]$（单状态）

3.4.3 NFA与DFA的转换
NFA的确定化

- 2、确定化定理
- 对于字母表Σ上的任一NFA $M=(K,\Sigma,f,S_0,Z)$，构造与M等价的DFA $M'=(K',\Sigma,f',S_0',Z')$
- 1).$K'=2^k$.即，由M的全部状态子集构成，特别地，令$S_0'=[S_0]$.
- 2).映射f'的定义:

  如果 $f(\{S1,S2,\cdots,Si\},a)=\{R1,R2,\cdots,Rj\}$

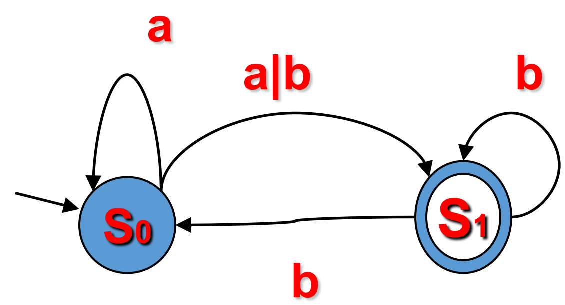
  那么 $f'([S1,S2,\cdots,Si],a)=[R1,R2,\cdots,Rj]$
- 3).M'的终态集 $Z=\{[Sp,Sq,\cdots,Sr] \mid \{Sp,Sq,\cdots,Sr\}\cap Z\neq\varnothing\}$

3.4.3 NFA与DFA的转换
NFA的确定化

NFA确定化的例子

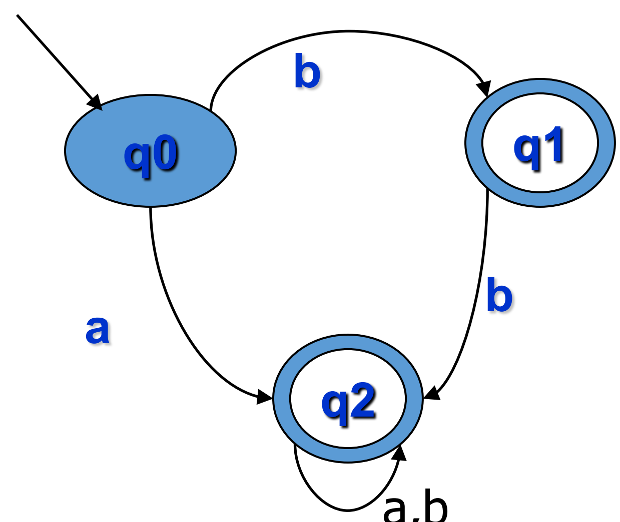
例3.2 $M=(\{S_0,S_1\},\{a,b\},f,S_0,\{s1\})$

	a	b
S0	{S0,S1}	{S1}
S1	∅	{S0,S1}



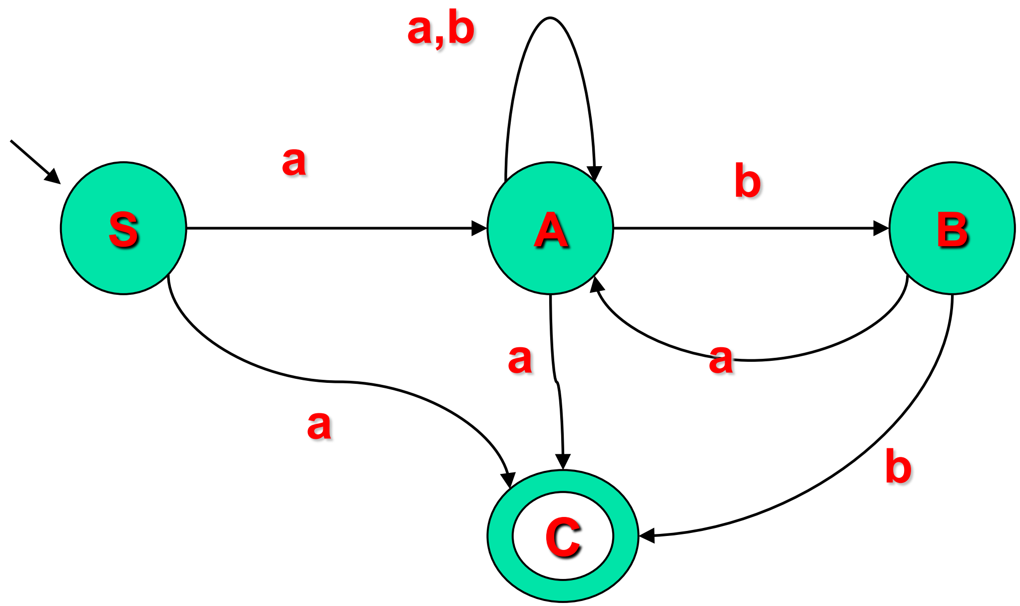
$M'=(K',\{a,b\},f',[S_0],\{[S_1],[S_0,S_1]\})$

重新命名状态		a	b
	~~[∅]~~	~~[∅]~~	~~[∅]~~
q_0 ⟺	[S0]	[S0,S1]	[S1]
q_1 ⟺	[S1]	[∅]	[S0,S1]
q_2 ⟺	[S0,S1]	[S0,S1]	[S0,S1]



3.4.3 NFA与DFA的转换
NFA的确定化

- 课堂举例



	a	b
[S]	[A,C]	ø
[A]	[A,C]	[A,B]
[B]	[A]	[C]
[C]	ø	ø
[S,A]	[A,C]	[A,B]
[S,B]	[A,C]	[C]
[S,C]	[A,C]	ø
[A,B]	[A,C]	[A,B,C]
[A,C]	[A,C]	[A,B]
[B,C]	[A]	[C]
[S,A,B]	[A,C]	[A,B,C]
[S,A,C]	[A,C]	[A,B]
[S,B,C]	[A,C]	[C]
[A,B,C]	[A,C]	[A,B,C]
[S,A,B,C]	[A,C]	[A,B,C]

3.4.3 NFA的确定化--有效子集法

3.有效子集法

步骤：

1). 令$S_0'=[S_0]$，$k'=\{[S_0]\}$

2). 对k' 中任一尚未标记的状态$q_i=[S_{i1},S_{i2},\cdots,S_{im}]$,$S_{ik}\in k$,做：

(1) 标记q_i

(2) 对每个$a\in\Sigma$，置$q_j=f([S_{i1},S_{i2},\cdots,S_{im}],a)=[R_{j1},R_{j2},\cdots,R_{jn}]$

(3) 若q_j不在k' 中，将q_j作为一个未被标记的状态加入k'

3). 重复步骤2，直到k' 中不再有未标记的状态为止。

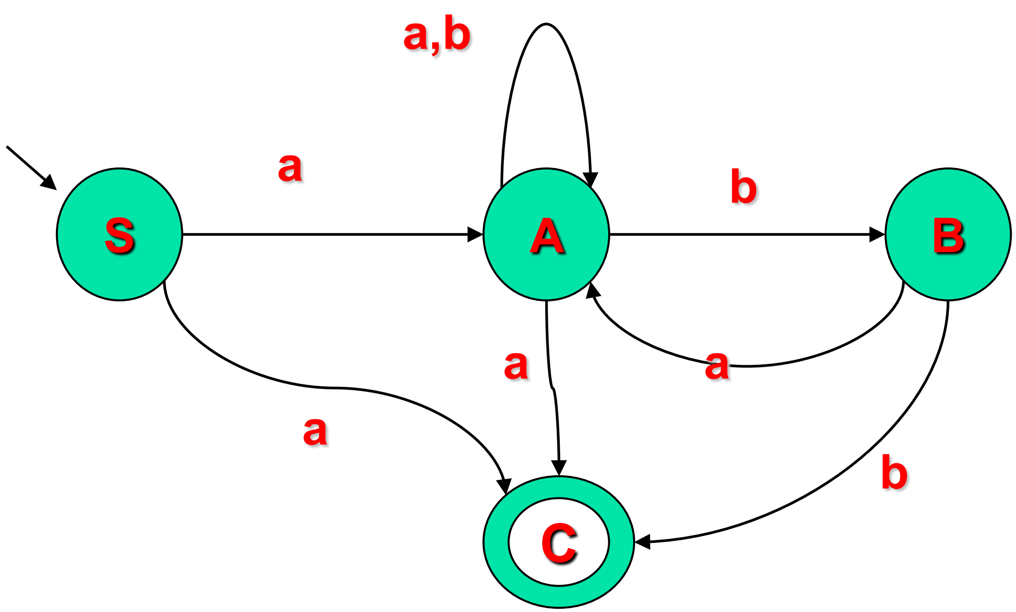
4) 确定初态和终态

初态：S_0'

终态：$Z=[S_{i1},S_{i2},\cdots,S_{im}]$，满足$\{S_{i1},S_{i2},\cdots,S_{im}\}\cap Z\neq\varnothing$

3.4.3 NFA的确定化--有效子集法

- 将下图所示的NFA采用有效子集法确定化



	a	b
S	{A,C}	ø
A	{A,C}	{A,B}
B	{A}	{C}
C	Ø	Ø

NFA的确定化-有效子集法

	a	b
[S]	[A,C]	ø
[A,C]	[A,C]	[A,B]
[A,B]	[A,C]	[A,B,C]
[A,B,C]	[A,C]	[A,B,C]

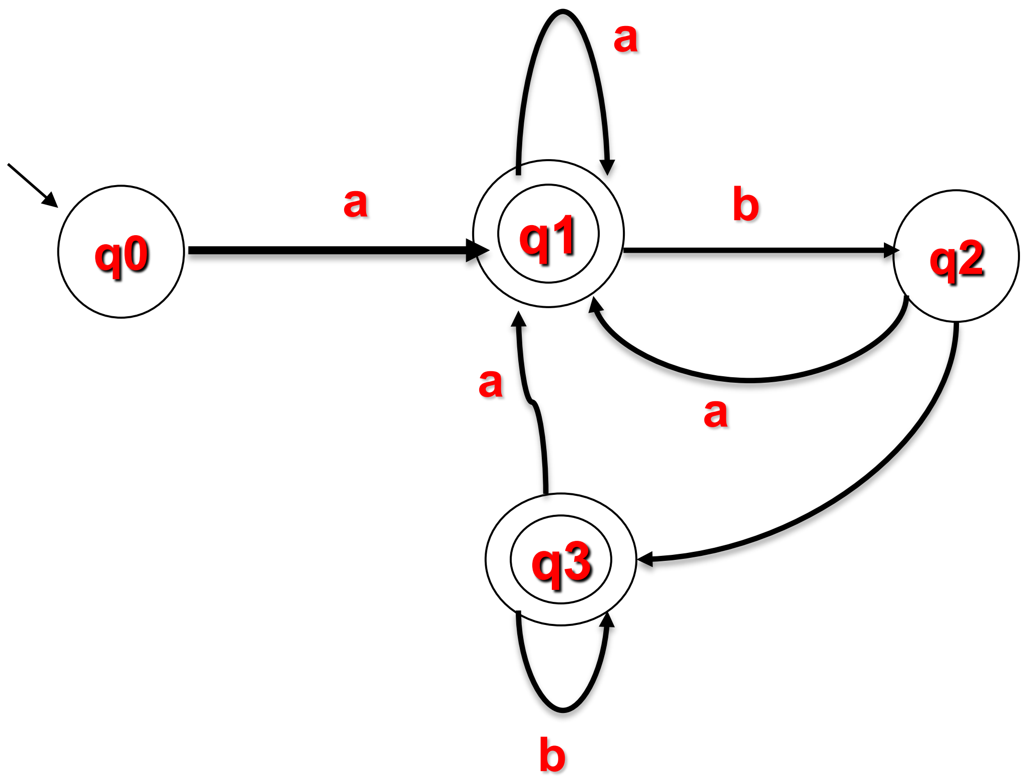
	a	b
q0	q1	ø
q1	q1	q2
q2	q1	q3
q3	q1	q3

q0=[S] q1=[A,C] q2=[A,B] q3=[A,B,C]

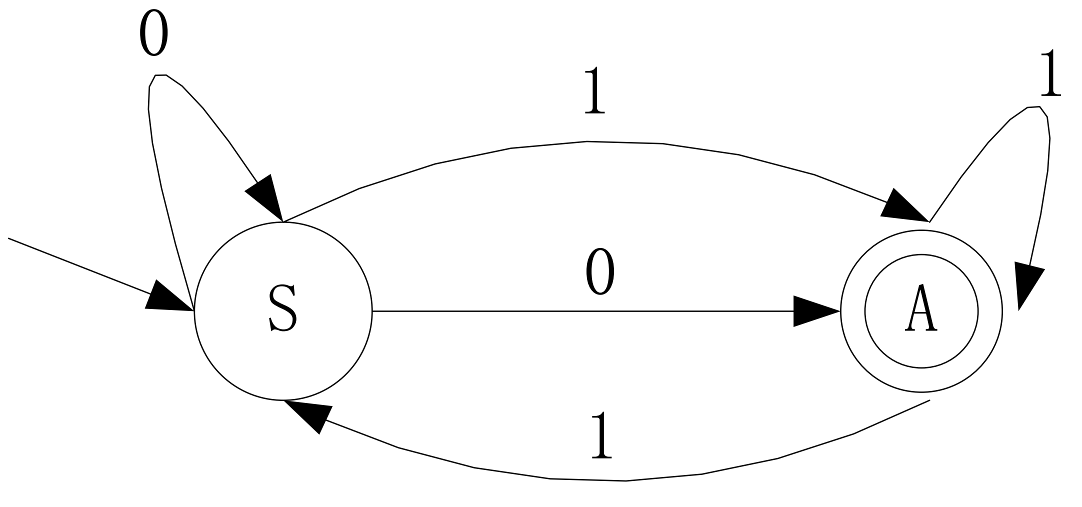
- 初态：q_0 终态：q_1, q_3

3.4.3 NFA的确定化--有效子集法

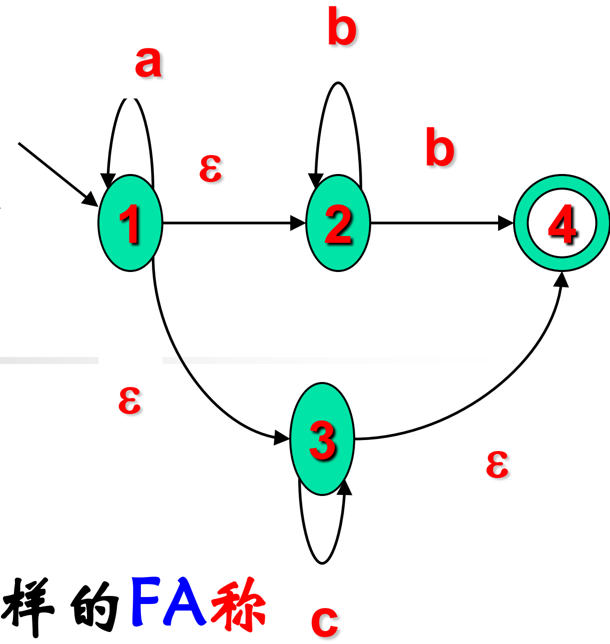
- 确定化之后的DFA：



NFA确定化、有效子集法——练习



3.4.4 具有ε动作的NFA

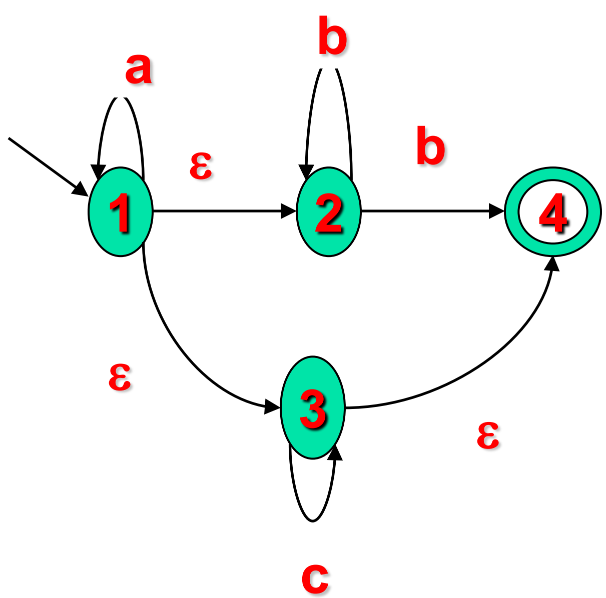


1.ε_NFA的概念

- 若在一FA中,允许对ε也作状态转移,则这样的FA称为具有ε动作的FA(NFA).此时,有的矢线上标记为ε
- 标记为ε 的矢线对识别符号串无影响,但却改变了当前的状态.
- 例如,右图中的FA中,从状态1到状态4存在路径:1(a) →1(a)→1(ε) →3(c) →3(c) →3(ε) →4,即M识别了aaεccε=aacc.

3.4.4 具有ε动作的NFA

- ε_NFA $M=(K,\Sigma,f,S_0,Z)$,其中,K,Σ,S_0,Z的定义与NFA相同,f的定义为 $f:K\times(\Sigma\cup\{\varepsilon\})\to 2^k$.
- ε可以视为一个输入符号,在矩阵表示中,也有相应的列.



状态转换表(f的定义)

	a	b	c	ε
1	{1}	ø	ø	{2,3}
2	ø	{2,4}	ø	ø
3	ø	ø	{3}	{4}
4	ø	ø	ø	ø

3.4.4 具有ε动作的FA

- f也可以拓广到 $\hat{f}: K\times\Sigma^* \rightarrow 2^k$. $\hat{f}(S,x)$是由这样的状态Q组成,存在从S到Q的路径,该路径上的连线标记组成的符号串恰好为x,其中,允许有有限个标记为ε

3.4.4 具有ε动作的FA

- 2. ε-闭包：*ε-CLOSURE(S)*

- ε-CLOSURE(S)：从NFA的状态S出发只经过标记为ε的边所能达到的状态集合
- ε-CLOSURE(Q)：设Q是K的某一子集，$Q=\{S_{i1},S_{i2},\cdots,S_{im}\}$，则

$$\boxed{\varepsilon-CLOSURE(Q)=\bigcup_{S\in Q}\varepsilon-CLOSURE(S)}$$

3.4.4 具有ε动作的FA

- 例，在上页的NFA中，
- ε-CLOSURE(1)={1,2,3,4}
- ε-CLOSURE(2)={2}
- ε-CLOSURE(3)={3,4}
- ε-CLOSURE(4)={4}

3.4.4 具有ε动作的FA

- 3.状态转换函数f的拓广
- 利用ε-CLOSURE(S)，可定义$\hat{f}$
- 对于S∈K, a∈Σ及w∈Σ*

(1) $\hat{f}$(S, ε)= ε-CLOSURE(S)

(2) $\hat{f}$(S,wa)=ε-CLOSURE(f($\hat{f}$(S,w),a))

(3)f(R,a)= $\bigcup_{q\in R} \mathrm{f}(q,a)$

(4) $\hat{f}$(R,a)= $\bigcup_{q\in R} \hat{f}(q,w)$

3.4.4 具有ε动作的FA

- f与$\hat{f}$的区别
- $\hat{f}$(S,a)是从S出发经过路径a到达的状态集（可走若干步）；
- f(S,a)是从S出发经过矢线a所达状态之集（只走一步）

$$f(S,a) \subseteq \varepsilon\text{-CLOSURE}(f(S,a)) \subseteq \hat{f}(S,a)$$

只走一步a矢线　　多步，第一步必须走a矢线　　路径a：第一步可以是ε矢线

- 例如，在前面的NFA中，

$\hat{f}(1,\varepsilon)=\varepsilon\text{-CLOSURE}(1)=\{1,2,3,4\}$　　$f(1,\varepsilon)=\{2,3\}$

$\hat{f}(1,a)=\{1,2,3,4\}$　　$f(1,a)=\{1\}$

3.3.4 具有ε动作的FA

- 4.ε-NFA的接受集：
- $L(M)=\{w \mid \hat{f}(S_0,w)\cap Z\neq\varnothing\}$
- 上例中ε-NFA的接受串aac的识别过程：

$$\hat{f}(1,\varepsilon)=\varepsilon\text{-CLOSURE}(1)=\{1,2,3,4\}$$

$$\hat{f}(1,a)=\varepsilon\text{-CLOSURE}(f(\hat{f}(1,\varepsilon),a)=\{1,2,3,4\}$$

$$\hat{f}(1,aa)=\varepsilon\text{-CLOSURE}(f(\hat{f}(1,a),a)=\{1,2,3,4\}$$

$$\hat{f}(1,aac)=\varepsilon\text{-CLOSURE}(f(\hat{f}(1,aa),c)=\{3,4\}$$

3.4.4具有ε动作的FA

- 5、ε-NFA的用途：构造更复杂的FA
- 有了ε-NFA，就可把识别各种不同单词的FA用ε矢线（并连）连接起来，组成一个单一的NFA，经确定化后可得识别所有单词的DFA，由此可设计编译程序的词法分析器。
- 例如，某语言的单词有：

1.BEGIN 2.END 3.IF 4.THEN 5.ELSE 6.标识符 7.无符号整数 8. < 9. <= 10. = 11.<> 12. > 13.>=

分别构造出识别各类单词的FA,然后将其合并为一个大FA,如书中P66图3-11所示(由于较难描绘,这里略去,请自行参阅教材)

3.4.5 ε-NFA的确定化：有效子集法

- 1. ε-NFA有效子集法确定化
- 设已给具有ε动作的NFA $M=(K,\Sigma,f,S_0,Z)$,构造相应的DFA $M'=(K',\Sigma,f',q_0,Z')$
- 初始状态$q_0=[\varepsilon\text{-CLOSURE}(S_0)]$, 然后对每个输入符号$a\in\Sigma$，反复计算新的状态转换函数，若产生新的子集则作为新状态，如此反复，直到无新状态产生。

操作	描述
ε-CLOSURE(S)	从NFA的状态S出发只经过标记为ε的边所能达到的状态集合
ε-CLOSURE(T)	从状态集合T中的状态出发，只经过标记为ε的边所能达到的状态集合

3.4.5 ε-NFA的确定化：有效子集法

2.有效子集法的步骤

1. 令 $K'=\{[\varepsilon\text{-CLOSURE}(S_0)]\}$; $f'=\varnothing$;

2. 对K'中尚未被标记的状态$q_i=[S_{i1},S_{i2},\cdots,S_{im}]$:

   (1)标记q_i;

   (2)对于每个$a\in\Sigma$,令$Ta=f(\{S_{i1},S_{i2},\cdots,S_{im}\},a)$; $q_j=[\varepsilon\text{-CLOSURE}(Ta)]$;

   (3)若$q_j\notin K'$,则令$K'=K'\cup\{q_j\}$;

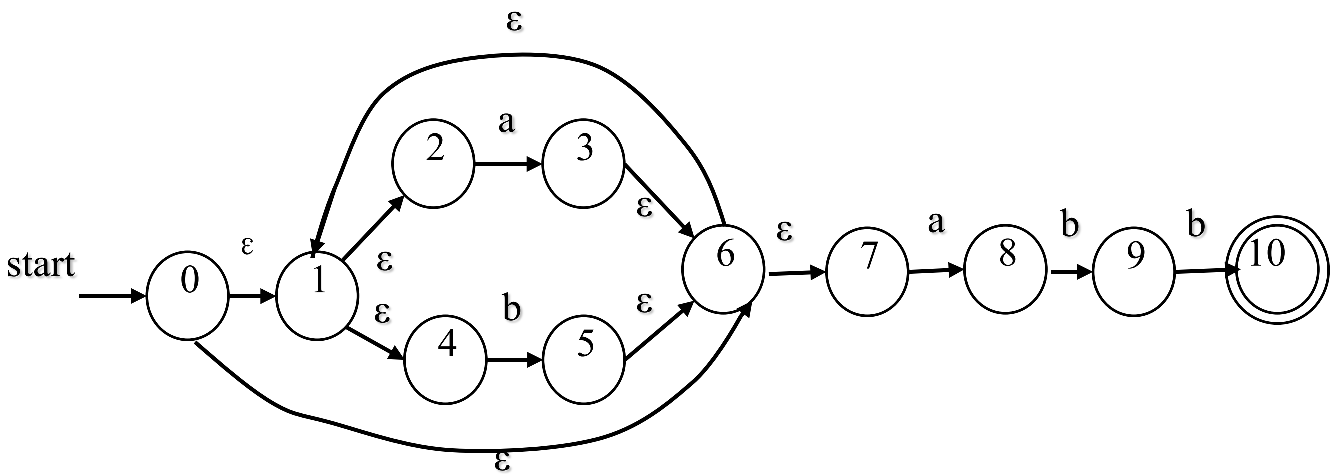
   (4)令$f'=f'\cup\{f'(q_i,a)=q_j\}$;

3. 重复2,直到K'中无未标记的状态;

4. 令$Z'=\{q_j \mid q_j \cap Z\neq\varnothing\}$ (这里把q_j视为集合)

3.4.5 ε-NFA的确定化：有效子集法

- 3.示例：将下图所示的NFA确定化



3.4.5 ε-NFA的确定化：有效子集法

- 步骤1：求每个状态S的ε-CLOSURE(S)

状态	ε-CLOSURE
0	{ 0,1,2,4,6,7 }
1	{ 1,2,4 }
2	{ 2 }
3	{ 3,6,7,1,2,4 }
4	{ 4 }
5	{ 5,6,7,1,2,4}
6	{ 6,7,1,2,4 }
7	{ 7 }
8	{ 8 }
9	{ 9 }
10	{ 10 }

3.4.5 ε-NFA的确定化：有效子集法

- 步骤2：计算新的状态转换Dtran，状态转换计算方法如下例所示，状态转换表如下表所示。
- Dtran[A,a]=ε-CLOSURE(f(A,a))=

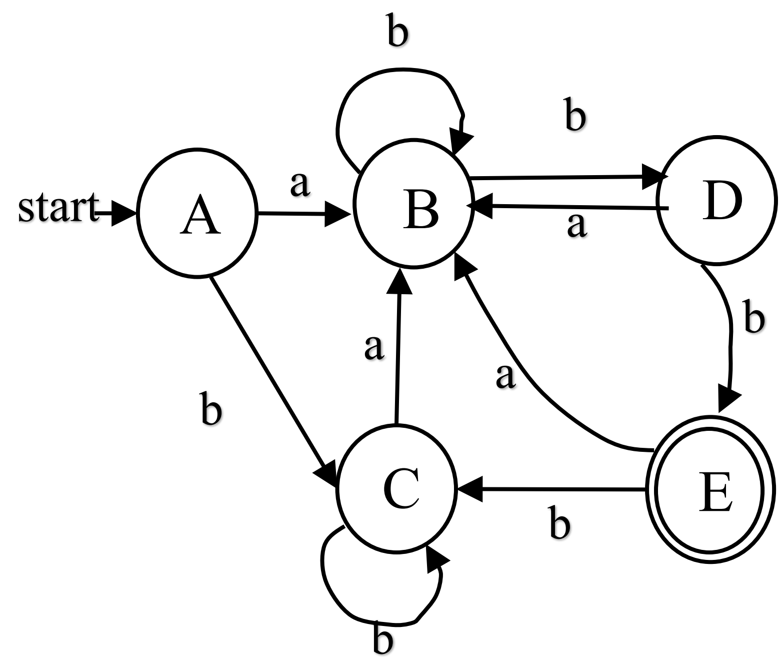
  ε-CLOSURE({3,8})= {1,2,3,4,6,7,8}

	a	b
{0,1,2,4,7}	{1,2,3,4,6,7,8}	{1,2,4,5,6,7}
{1,2,3,4,6,7,8}	{1,2,3,4,6,7,8}	{1,2,4,5,6,7,9}
{1,2,4,5,6,7}	{1,2,3,4,6,7,8}	{1,2,4,5,6,7}
{1,2,4,5,6,7,9}	{1,2,4,5,6,7,9}	{1,2,4,5,6,7,10}
{1,2,4,5,6,7,10}	{1,2,3,4,6,7,8}	{1,2,4,5,6,7}

3.4.5 ε-NFA的确定化：有效子集法

- 步骤3：重命名，确定初态及终态
- 初态：A　　终态：E

DFA 新状态		a	b
{0,1,2,4,7}	A	B	C
{1,2,3,4,6,7,8}	B	B	D
{1,2,4,5,6,7}	C	B	C
{1,2,4,5,6,7,9}	D	B	E
{1,2,4,5,6,7,10}	E	B	C



3.4.5 ε-NFA的确定化：有效子集法

DFA M’ f’的定义：

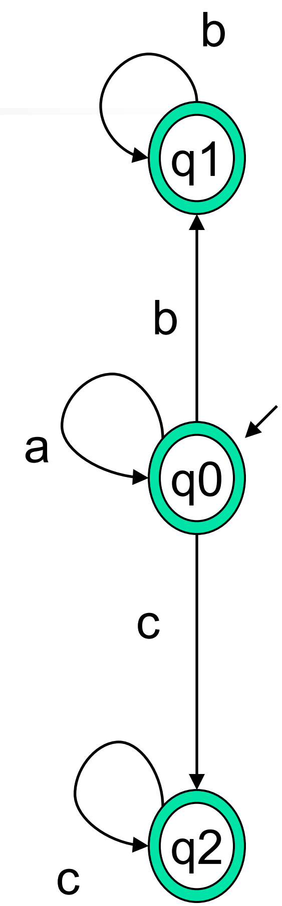
	a	b	c
[1,2,3,4]	[1,2,3,4]	[2,4]	[3,4]
[2,4]	ø	[2,4]	ø
[3,4]	ø	ø	[3,4]

重命名：

	a	b	c
q0	q0	q1	q2
q1	ø	q1	ø
q2	ø	ø	q2

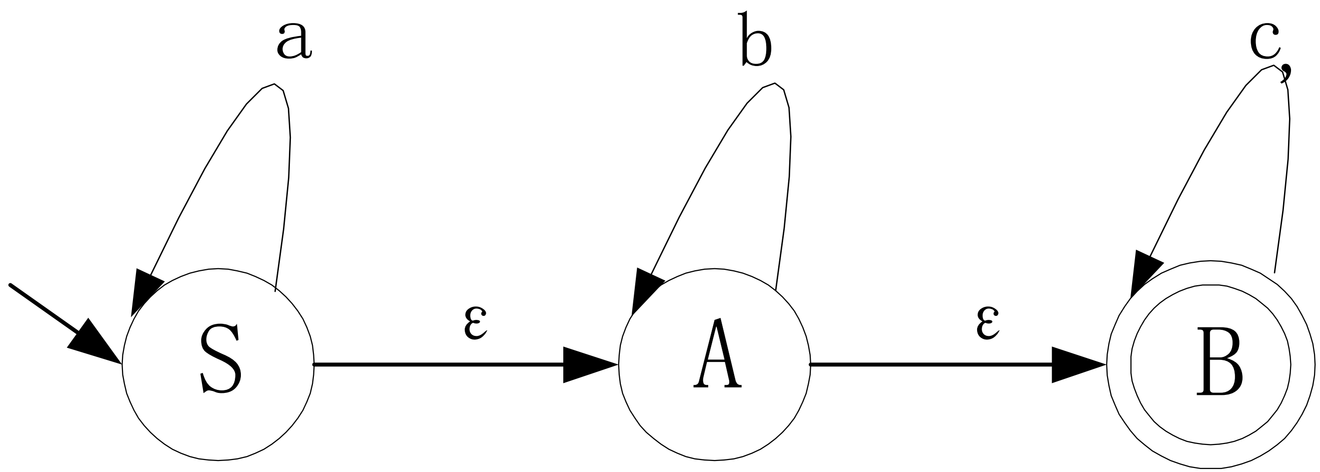
初态：q0

终态：q0,q1,q2



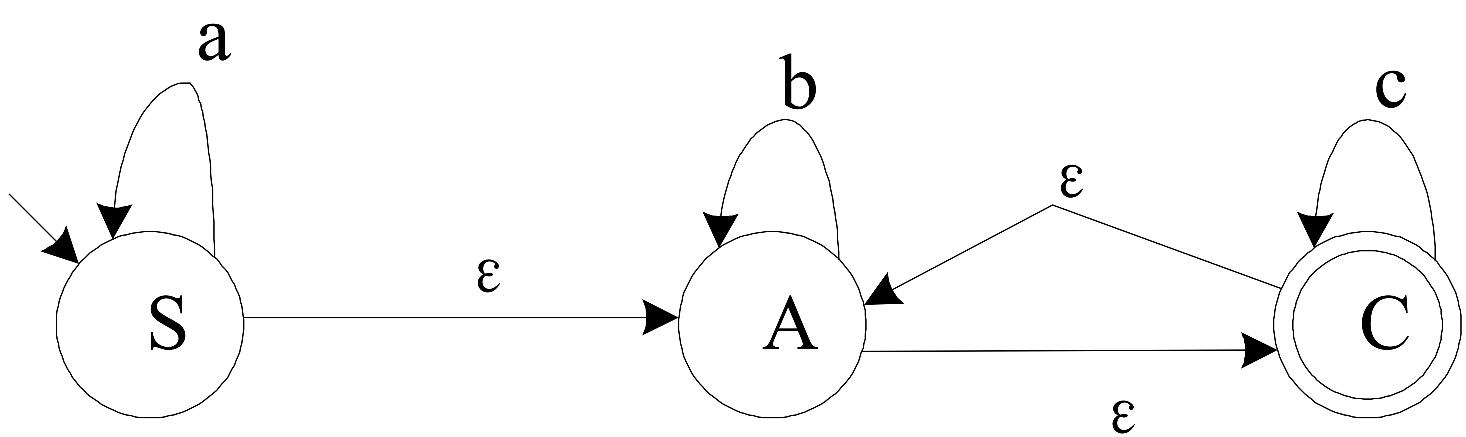
3.4.5 ε-NFA的确定化：有效子集法

- 示例：将如图所示的NFA确定化



3.4.5 ε-NFA的确定化：有效子集法

- 作业：将如下图所示的ε-NFA，采用有效子集法确定化



3.4.6 DFA状态数的最小化

1.相关概念

- DFA M的最小化: 构造等价的DFA M’ 其状态数达到最小。
- **可区分状态:** 设P,Q∈K,字符串w把状态P和状态Q区别开， iff

  $(f(P,w)\in Z \wedge f(Q,w)\notin Z) \vee (f(P,w)\notin Z \wedge f(Q,w)\in Z)$
- **等价状态:** 若$\forall w\in\Sigma^*$, $f(P,w)\in Z \Leftrightarrow f(Q,w)\in Z$,则称 P与 Q等价(不可区分)

3.4.6 DFA的化简

- 2.DFA最小化算法
- **基本思想**:把状态集划分成互不相交的子集，使子集中的状态是等价的
- 化简DFA的算法步骤:
  - 划分状态集
  - **合并状态**:取每组状态中的代表状态，删去其他等价状态
  - 若有**死状态**或**不可达状态**，则把它们删去。
- 死状态：无法达到终止状态的非终止状态
- 不可达状态：不能从开始状态达到它的那些状态

3.4.6 DFA状态数的最小化

3.DFA最小化步骤

1）初始划分：将状态集K按照是否属于终态集划分为：$\pi=\{Z, K\text{-}Z\}$.

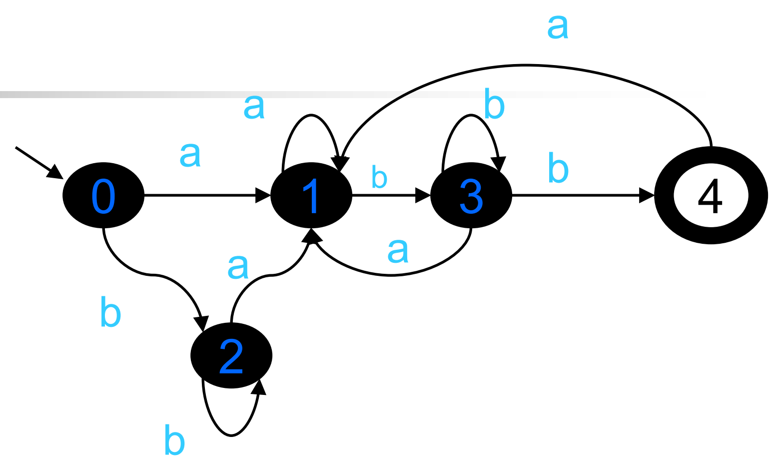
2）设当前的划分:$\pi=\{I_1,I_2,\cdots,I_m\}$, 考察子集$I_i$：若存在$a\in\Sigma$,使得$S_u=f(S_{ip},a)\in I_j$，$S_v=f(S_{iq},a)\in I_k$,则 S_{ip} 和S_{iq} 可区分得到 新划分π_{new}

3）若π_{new}不等于π,则令$\pi=\pi_{new}$. 转2.

4）对于最终的π，从每个划分块中任选一状态为代表，构成K'，S_0的代表为初态。若$I_i\cap Z\neq\varnothing$，则I_i的代表$\in Z'$；将引入(出)非代表的矢线引向代表.

3.4.6 DFA状态数的最小化

4.DFA最小化的例子



1. π={{0,1,2,3},{4}}

$\{0,1,2,3\}_a$={1}未区分;

$\{0,1,2\}_b$={2,3},

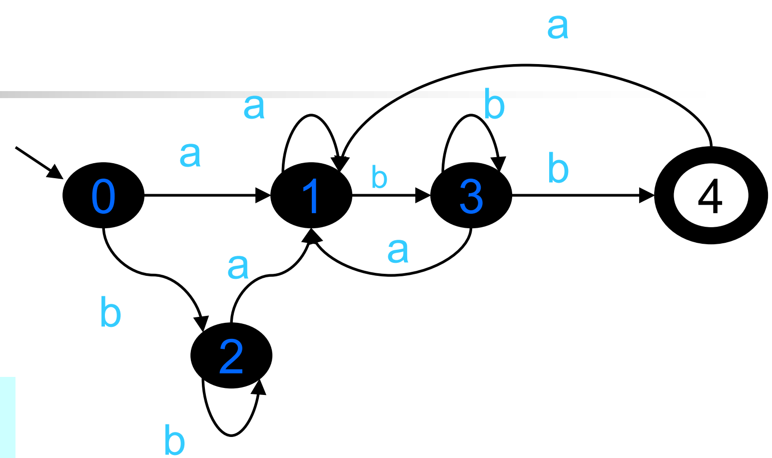
$\{3\}_b$={4}, 所以3 与 0,1,2 可区分;

π={{0,1,2}, {3},{4}}

	a	b
0	1	2
1	1	3
2	1	2
3	1	4
4	1	2

3.4.6 DFA状态数的最小化

4.DFA最小化的例子



	a	b
0	1	2
1	1	3
2	1	2
3	1	4
4	1	2

2.$\{0,1,2\}_a=\{1\}$, 未区分;

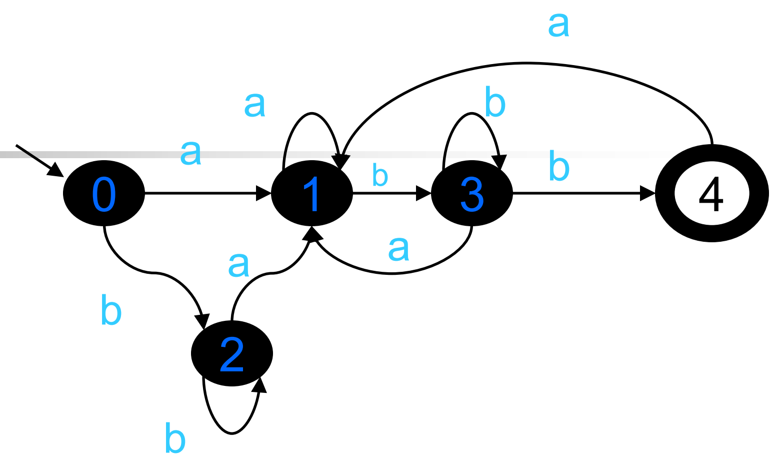
$\{0,2\}_b=\{2\}$,$\{1\}_b=\{3\}$, 1 与0,2可区分; $\pi=\{\{0,2\},\{1\},\{3\},\{4\}\}$;

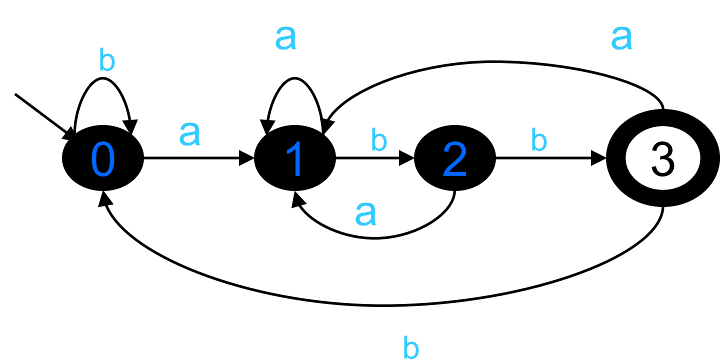
3. $\{0,2\}_a=\{1\}$,$\{0,2\}_b=\{2\}$不可区分, $\pi_{new}=\pi$.结束.

3.4.6 DFA的化简

4.DFA最小化的例子

	a	b
0	1	2
1	1	3
2	1	2
3	1	4
4	1	2

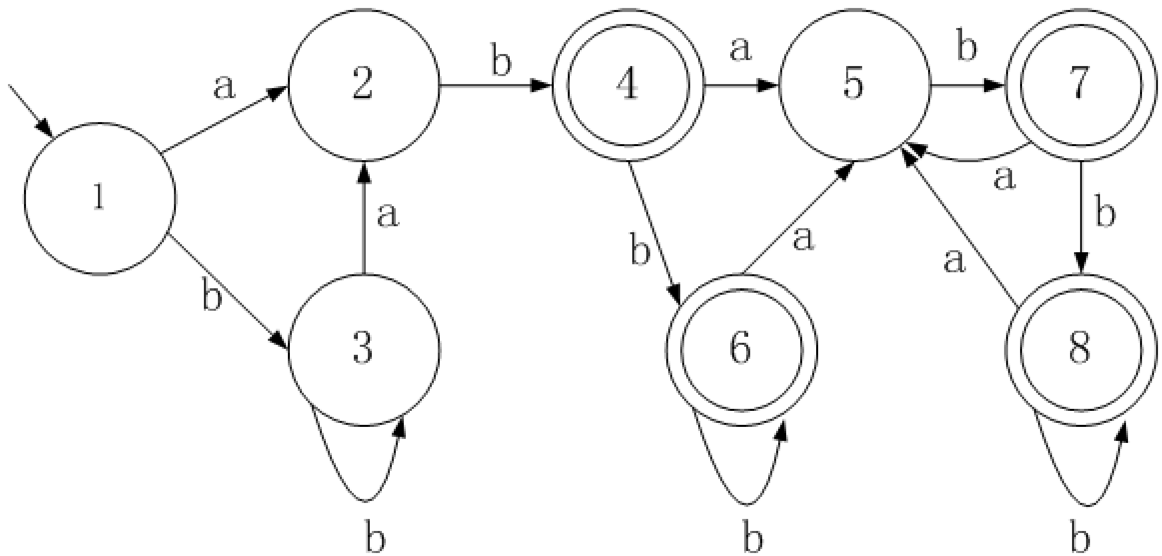




定理3.2 最小状态数的DFA在同构意义下是唯一的.

3.4.6 DFA状态数的最小化

举例：对如下图所示的DFA最小化



- 划分状态集为π：π1={4,6,7,8}和π2={1,2,3,5}
- 对于$\{4,6,7,8\}_a$={5},未区分;
  $\{4,6,7,8\}_b$={6,8},未区分;

3.4.6 DFA状态数的最小化

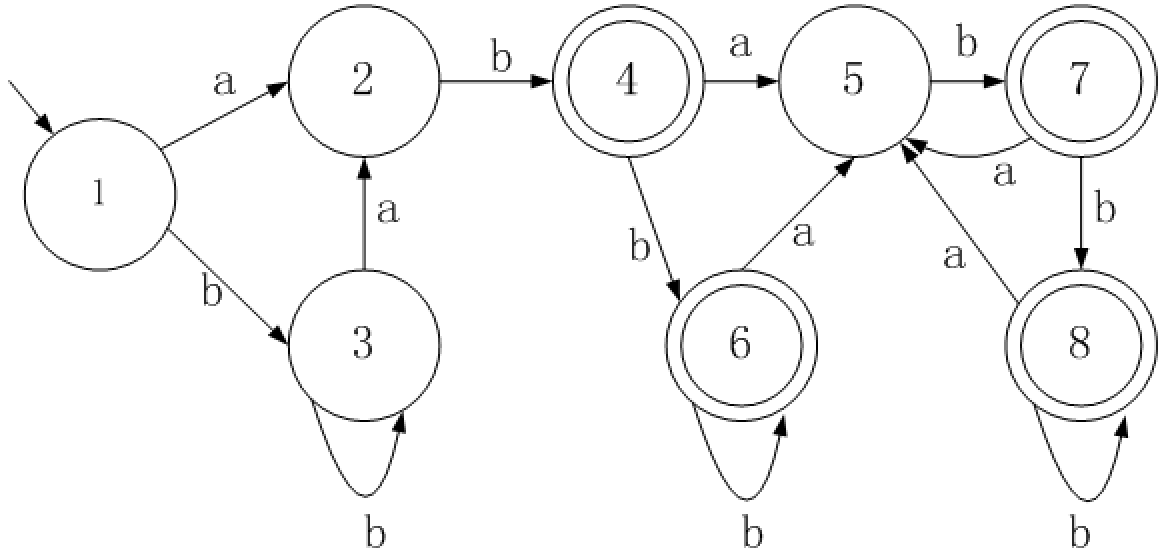
举例（续1）

- 对于{1,2,3,5}

{1,3}a={2}, {2,5}a=ø,{}

{1,3}b={3}, {2,5}a={4,7}

π= {π1={4,6,7,8},π2={1,3}, π3={2,5}}}

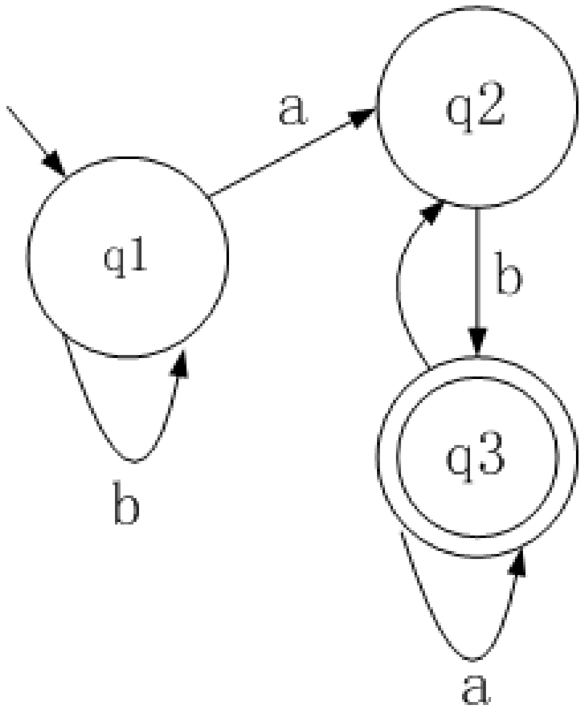


3.4.6 DFA状态数的最小化

举例（续2）

重新命名：q1={1,3}, q2={2,5} q3={4,6,7,8}

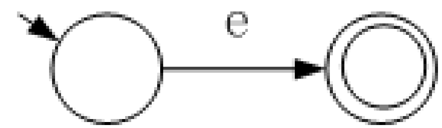
初态：q1，终态q3



	a	b
q1	q2	q1
q2	ø	q3
q3	q2	q3

3.5 正规式构造FA

- 对于∀正规式r,∃FA M,使 L(M)=Lr。可以将任何正规式转变为接受相同语言的NFA。
- 方法：



(1)首先构造一个广义自动机只有初态和终态，边上标记为相应正规式

(2)按照*,| ,•对e进行分解

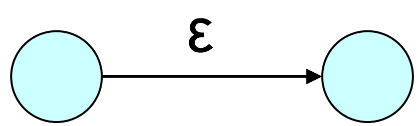
(3)直到图中每条边上标记为a∈Σ或ε为止。

3.5正规式构造FA

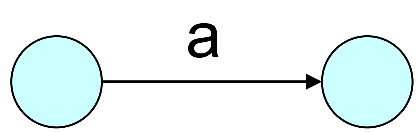
- 正规式与其对应的FA

空集Φ

ε

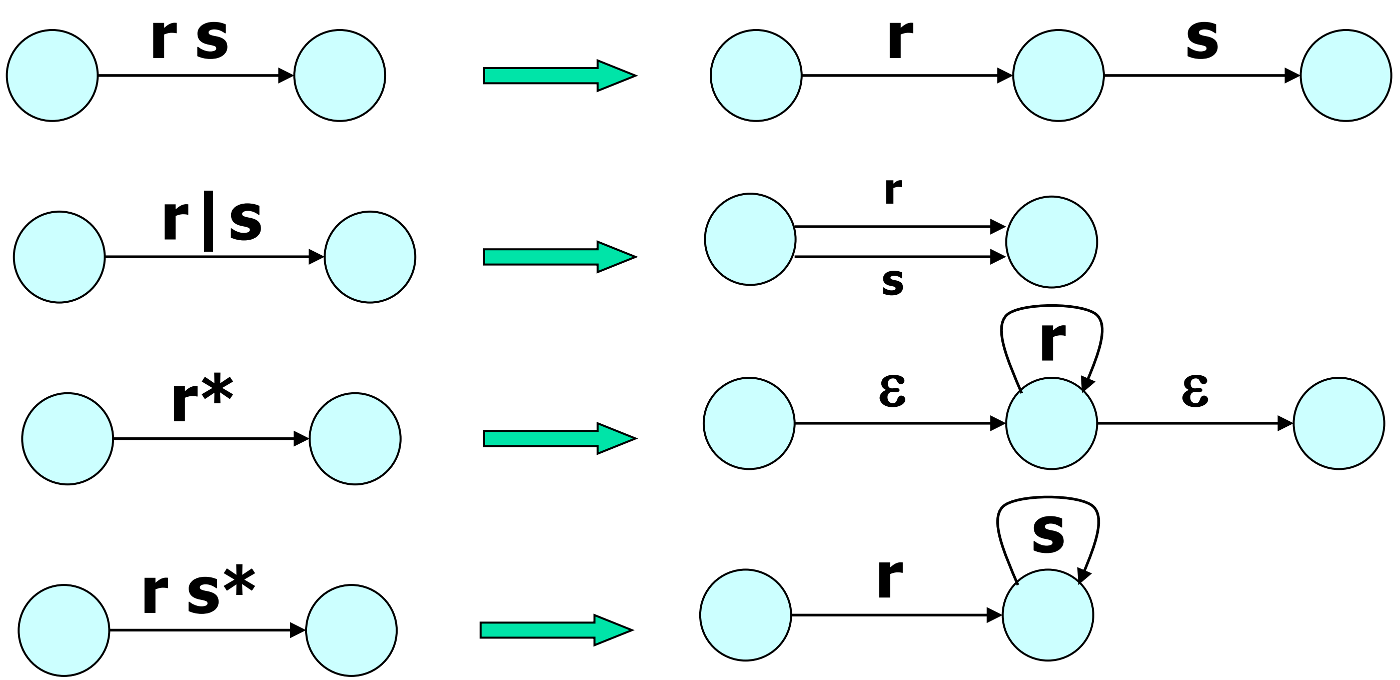


a



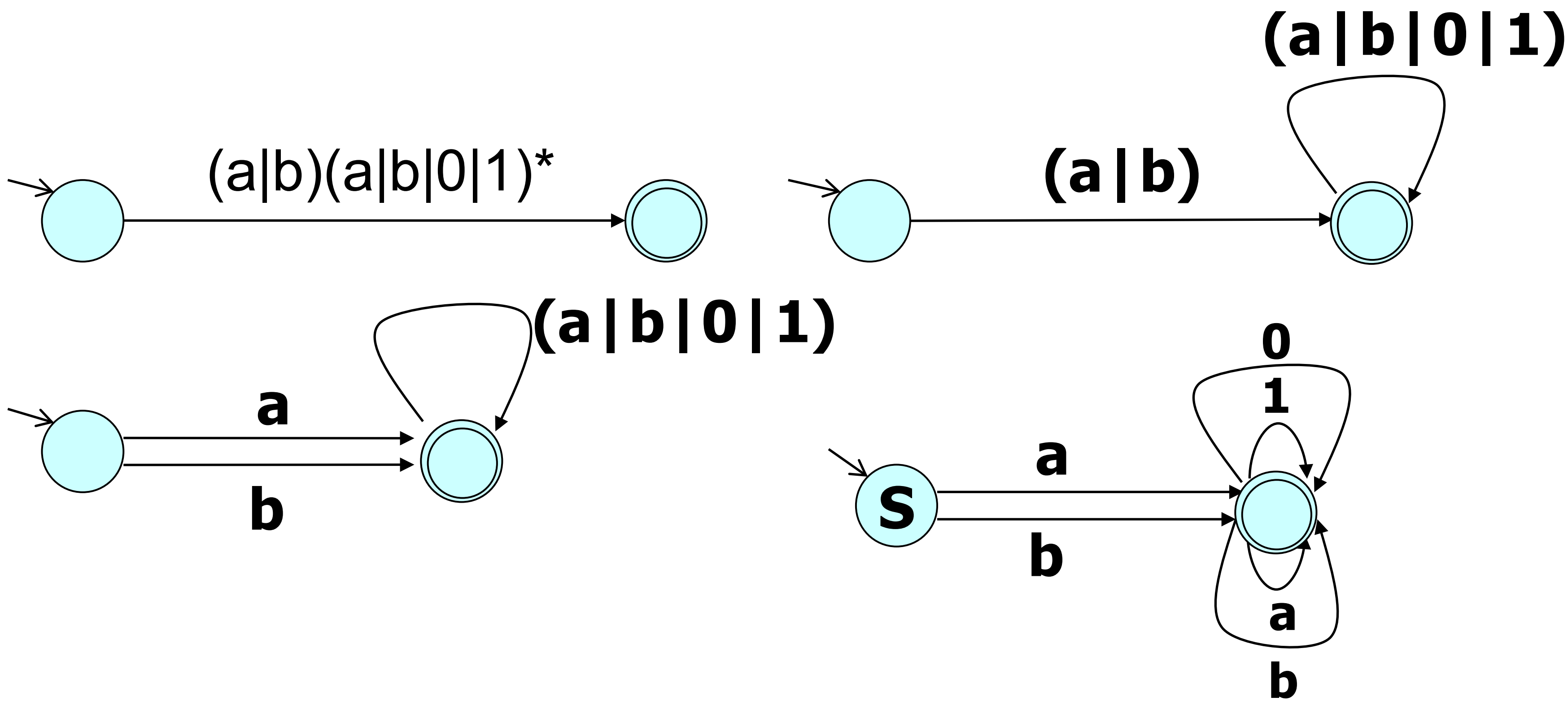
3.5正规式构造FA

- 正规式与其对应的FA（分解规则）



举例

- 例：正规表达式(a|b)(a|b|0|1)*转换NFA的构造过程如下：

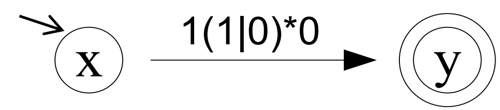


3.5 正规式构造FA

- 练习：为以下正规式构造NFA
- a(b|aa)*b
- (a|b)*abb
- ε|(0|1)01*|0*
- a*ba*ba*ba*
- (a|b)*a(a|b) (a|b)

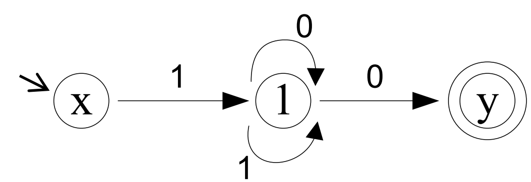
有限自动机举例

- 例：设计一个DFA M，它识别二进制偶数（不以0开头的无符号数）
- 解：
- 1、 写出正规式 1 (1|0) *0



- 2、 画出NFA M'

  细化为：



- 3. 确定化为DFA M

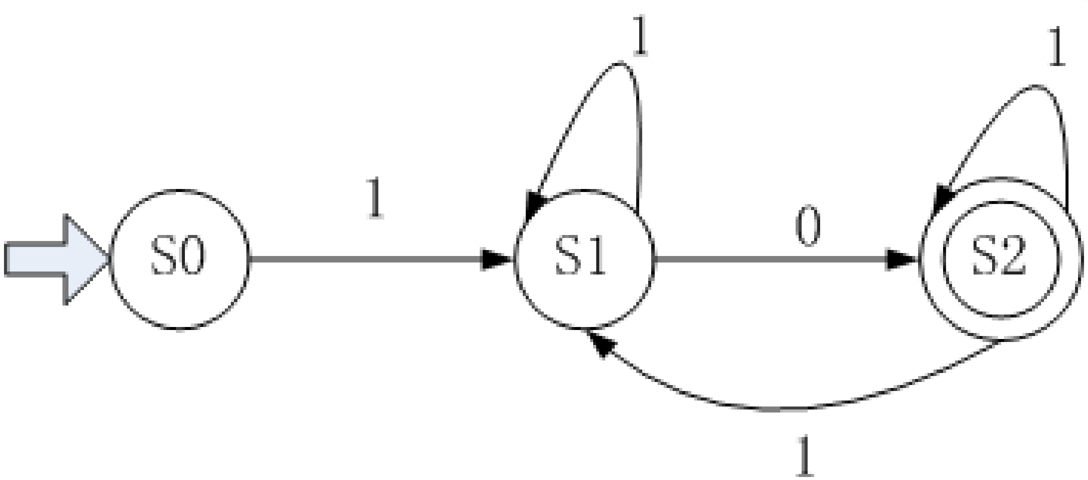
确定化

	0	1
[x]	ø	[1]
[1]	[1,y]	[1]
[1,y]	[1,y]	[1]

重新命名

	0	1
S0	ø	S1
S1	S2	S1
S2	S2	S1

DFA为：



习题

- 1.构造正规式对应的DFA，并化简。

  $R=(a^*|b^*)b(ba)^*$

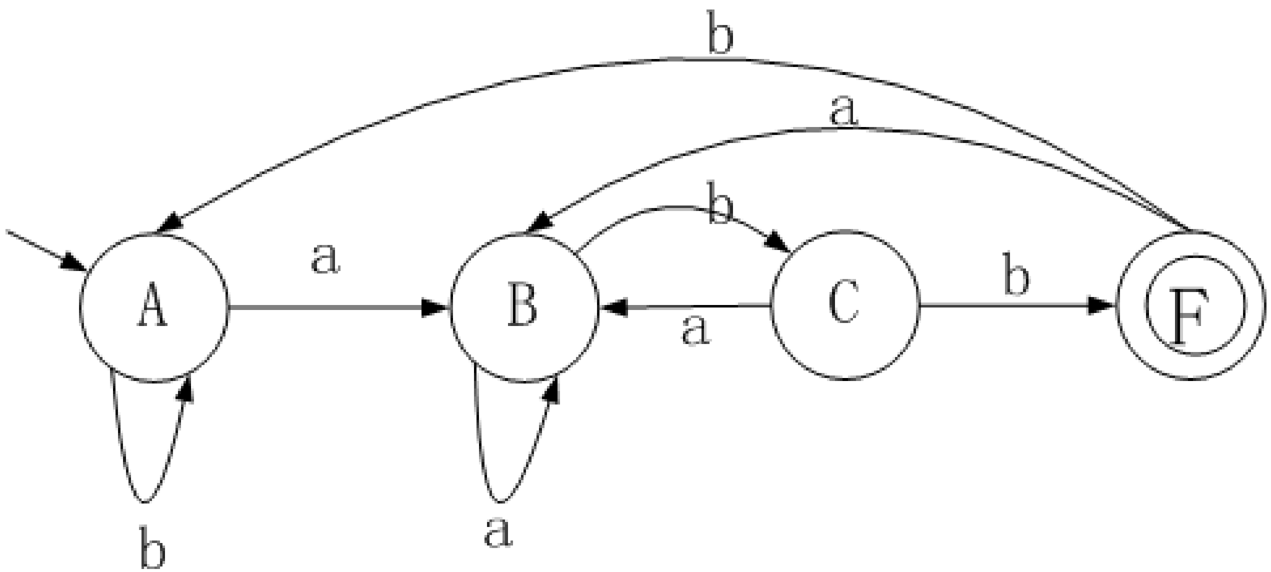
- 2.求文法对应的正规式

  $S \rightarrow 0B$

  $B \rightarrow 0B|1S|0$

习题

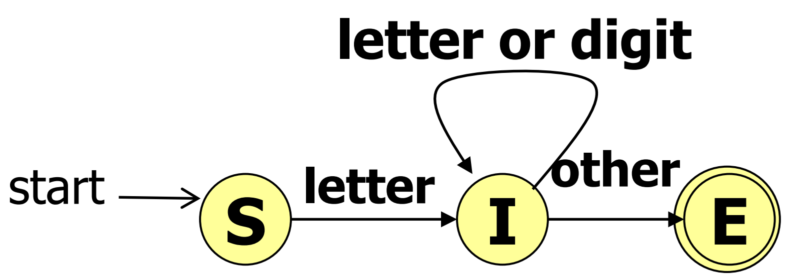
- 3.写出下面DFA的右线性文法和正规式



3.6 正规文法和有穷自动机的等价性转换

有穷自动机的作用：构造词法分析器的一个中间步骤

- 正规式⇨有穷自动机；
- 正规文法⇨有穷自动机；



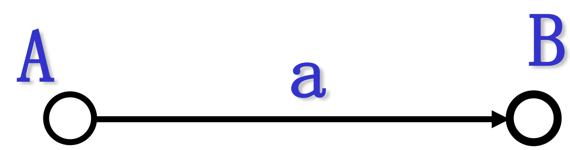
3.6.1右线性文法构造状态转换图

1.构造规则

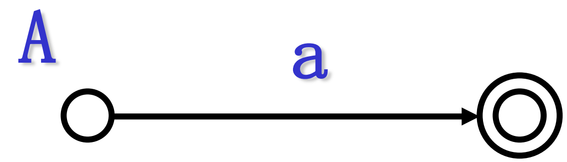
1) 以每个非终结符为状态结点，开始符号对应初态 S

2) 引入一个终态 T ($\notin V_N$)。

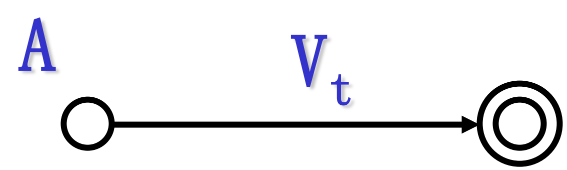
3) 对于规则 $A \to aB$，画从状态 A 到 B 的弧，标为 a



4) 对于规则 $A \to a$，画从状态 A 到终态 T 的弧，标为 a



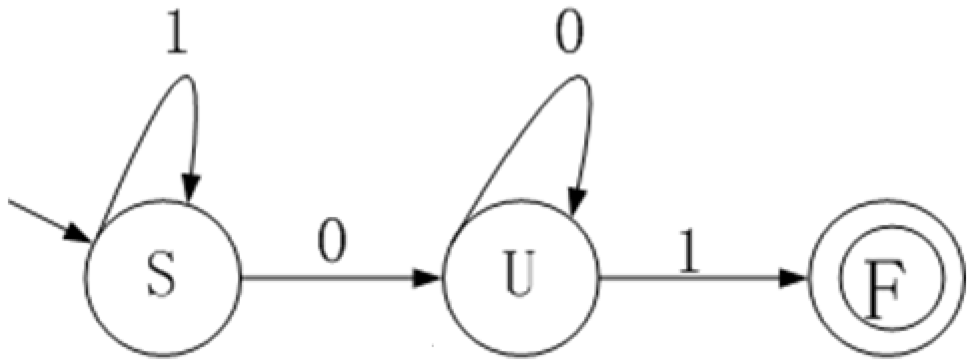
对于G中产生式$A \to \varepsilon$(若有的话),V_t由未出现在节点A射出弧的节点构成，或者直接标记ε。



3.6.1右线性文法构造状态转换图

- 右线性文法⇨状态转换图示例

- 文法G[S]： S→1S S→0U U→0U U→1

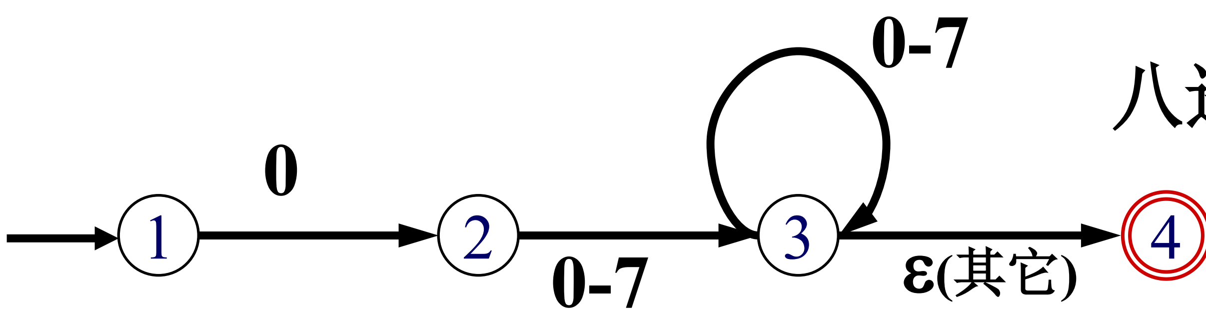


3.6.1右线性文法构造状态转换图

举例：C语言不同进制数的状态转换图

S→0A　　A →(0|1|2|3|4|5|6|7)B

B →(0|1|2|3|4|5|6|7)B| ε



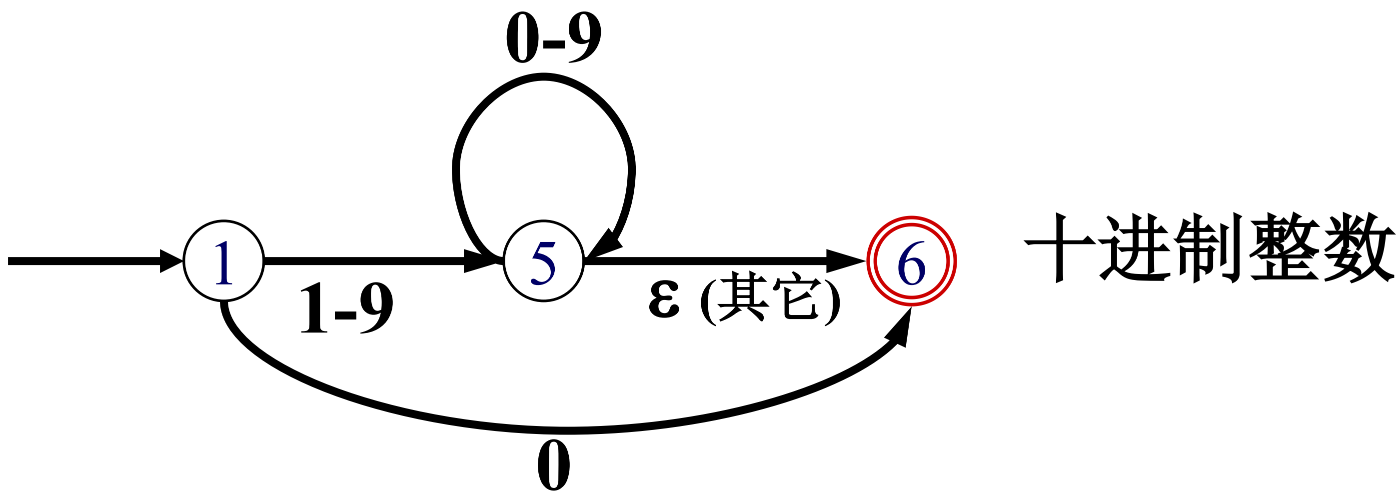
八进制整数

3.6.1右线性文法构造状态转换图

- 举例：C语言不同进制数的状态转换图

S→0 | (1|2|3|4|5|6|7|8|9)A

A → (0|1|2|3|4|5|6|7|8|9)A | ε



3.6.1右线性文法构造状态转换图

2. 利用状态转换图识别单词符号

1). 从初态出发

2). 读入一字符

3). 按当前字符转入下一状态

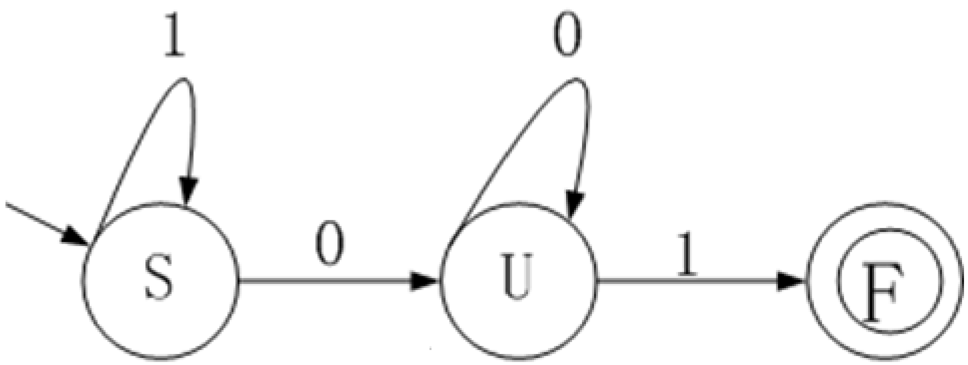
4). 重复2,3 直到无法继续转移

- 若当前状态是终止状态，说明读入的字符组成一单词；否则，说明输入不符合词法规则。

3.6.1右线性文法构造状态转换图

2. 利用状态转换图识别单词符号

示例：利用状态转换图识别符号串：10001



3.2.1右线性文法构造状态转换图

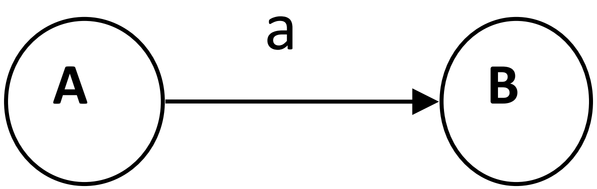
3.状态转换图与文法推导

用状态转换图识别符号串w的过程，就是为w建立一个推导$S \overset{*}{\Rightarrow} w$的过程。

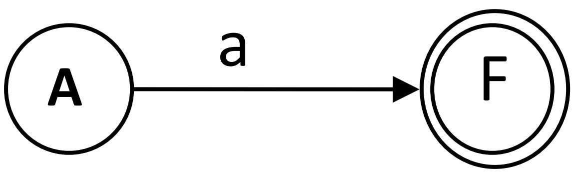
$$S \Rightarrow a_1A_1 \Rightarrow a_1a_2A_2 \Rightarrow \ldots\ldots \Rightarrow a_1a_2\ldots a_{n-1}A_{n-1} \Rightarrow a_1a_2\ldots a_n$$

3.6.1右线性文法构造状态转换图

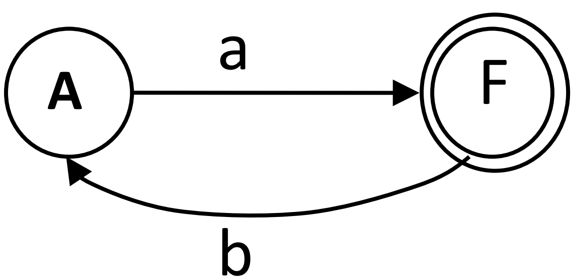
4、状态转换图⇨右线性文法



A→aB



A→a



A→a　　A→aF

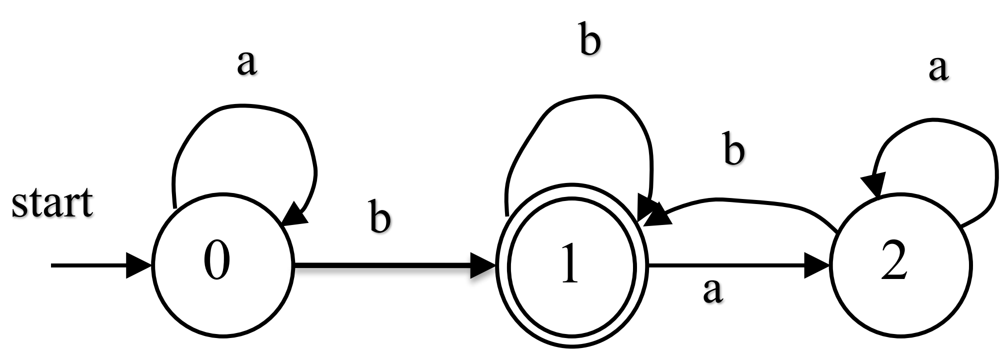
F →bA

3.6.1右线性文法构造状态转换图

- 课堂提问举例：
- 对于如下文法，构造状态转换图

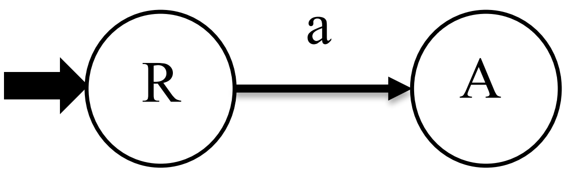
G[S]: S →0A　　A→0A | 0B　　B →1A | 1

- 写出下面状态转换图对应的右线性文法

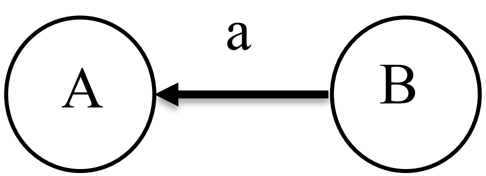


3.6.2 由左线性文法构造状态转换图

- 1.构造规则
  - 1)开始符S对应**终态结点**,再引入一个**新结点**$R(\notin V_N)$**作为初态**.
  - 2)对于G中形如$A\rightarrow a$的产生式,引矢线:$R\rightarrow A$,且标记为a;



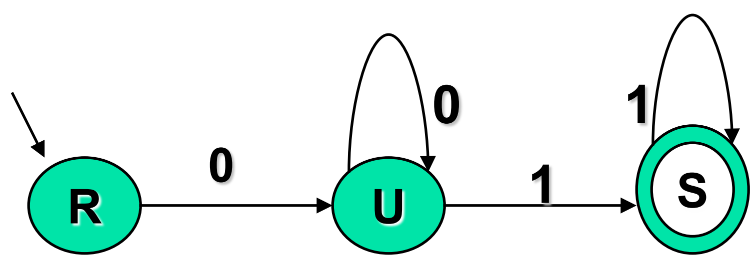
  - 3)对于G中形如$A\rightarrow Ba$的产生式,引矢线$B\rightarrow A$,且标记为a.



3.6.2 由左线性文法构造状态转换图

- 已给文法**G=({S,U},{0,1}, {S→S1 | U1, U→U0 | 0},S)**

S →S1　S →U1　U →U0　U→0

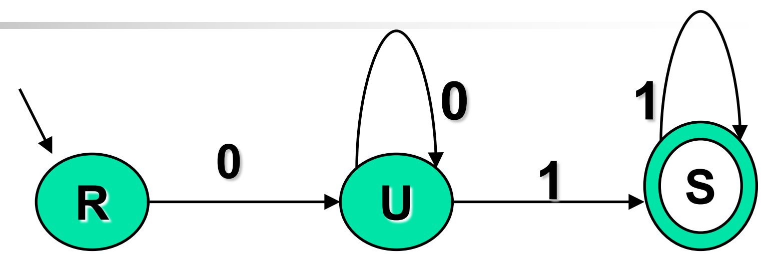


3.6.2由左线性文法构造状态转换图

2.用左线性文法构造出的状态转换图来识别文法的句子

3.就识别的方法而言,它却属于“↑”分析（归约）.

以句子00011为例,给出其识别的的步骤，见右表.



步骤	当前状态	余留的符号串
1	R	00011
2	U	0011
3	U	011
4	U	11
5	S	1
6	S	(识别结束)

小结

- 1.状态转换图
- 2.由正规文法构造状态转换图
- 3.根据状态转换图识别单词

3.7 词法分析程序的实现

构造词法分析程序的方法

- 方法1：用**手工方式**，即根据识别语言单词的状态转换图，使用某种高级语言，例如，C语言直接编写词法分析程序。
- 方法2：利用**自动生成工具**LEX自动生成词法分析程序。
- 方法3：采用状态转换矩阵，根据单词的状态转换图，驱动状态转换矩阵

3.7 词法分析程序的实现

词法分析程序实现中要考虑的问题

- 确定实现词法分析程序的执行方式
- 确定属性字的结构
- 缓冲区预处理，超前搜索，
- 关键字的处理，符号表的实现
- 查找效率，算法的优化实现
- 词法错误处理

3.7 词法分析程序的实现

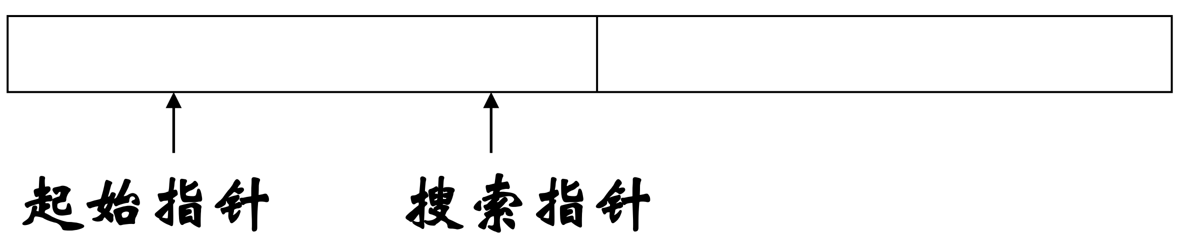
属性字

- 词法分析程序对说明部分不做语义处理。
- 词法分析程序输出属性字一般采用下面的形式：（符号类，符号值）
- 属性字是符号的**机内表示**，有**统一固定的长度**

3.7 词法分析程序的实现

源程序的输入

- 在内存开辟缓冲区，将程序文本放进该缓冲区
- 预处理：删除无用字符等
- 词法分析程序对缓冲区扫描时，设置两个指示器，一个指向当前正在识别的单词的开始位置，称为起始指针；另一个用于向前搜索，以寻找单词的终点，称为扫描指针。



3.7 词法分析程序的实现

超前搜索

- 词法分析程序在读取单词时，为了判断是否已读入整个单词的全部字符，常采取向前多读取字符并通过读取的字符来判别，即所谓超前搜索技术。

3.7 词法分析程序的实现

关键字的识别与查表算法

- 对于关键字，先把它们当成标识符，然后去查关键字表。若在表中查到，则为关键字，获取相应的类别码；否则，认为是标识符。
- 查找算法：
  - 线性查找
  - 折半查找
  - Hash函数

3.7 词法分析程序的实现

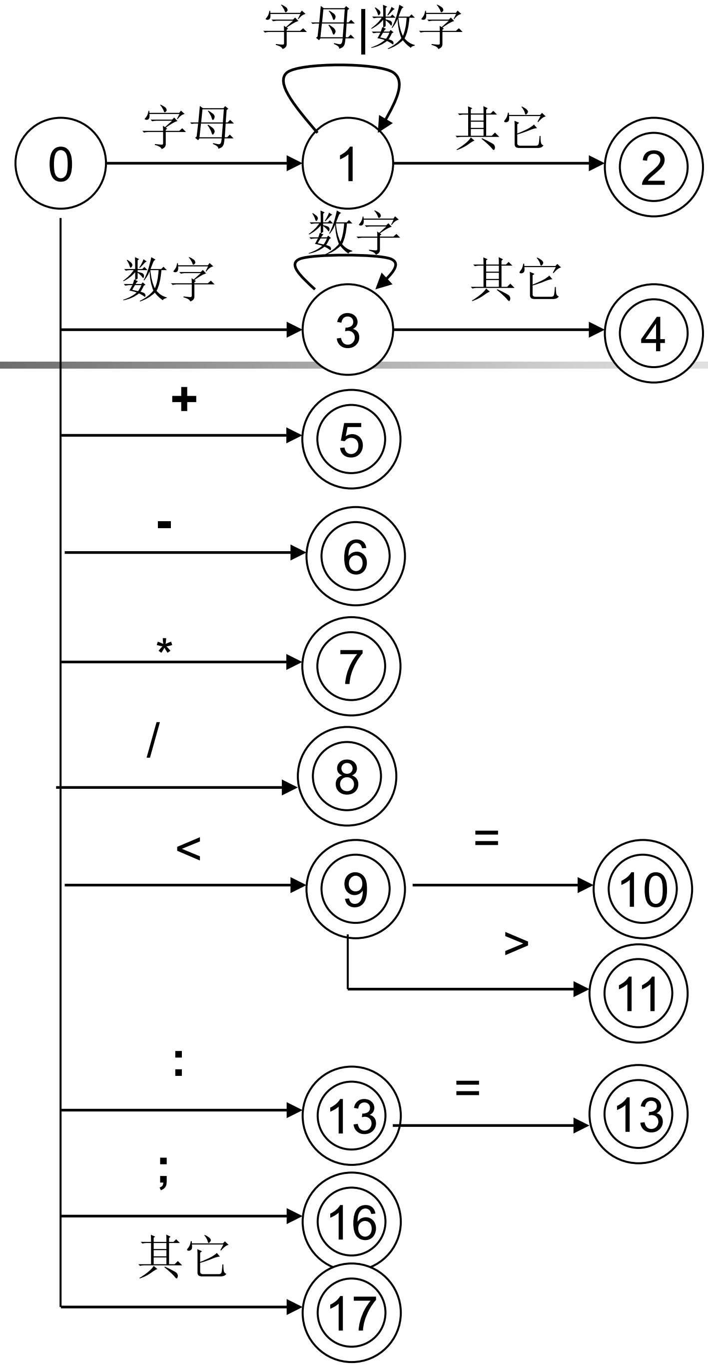
出错处理

- 对定义外的（如，对首字符不是字母的，不是数字的，不是运算符和分界符的）单词进行出错处理。

3.7.1词法分析程序的编写

- 多数语言的词法规则可用正则文法和正则表达式来描述。正则文法或正则表达式定义的语言都可以被状态图识别。
- 使用状态图设计词法分析程序的步骤如下：
  - 对程序设计语言的单词按类构造相应的状态图。（这里把关键字与标识符作为一类）
  - 合并各类单词的状态图，增加一个出错处理终态，构成一个识别该语言所有单词的状态转换图
  - 对状态图的每一个终点编一段相应的子程序。

举例



手工方式编写词法分析程序

- 根据状态转换图：

1.保存上一个读入字符，从输入串读下一个字符；

2.判别读入的字符由此状态出发的哪条边上的标记相匹配，转至相应的状态；

3.均不匹配时，报告出错。

通过case语句多路转换完成DFA的处理流程

扫描器算法见书上81-83

3.7.2 词法分析器的自动生成技术

- 本节讨论：

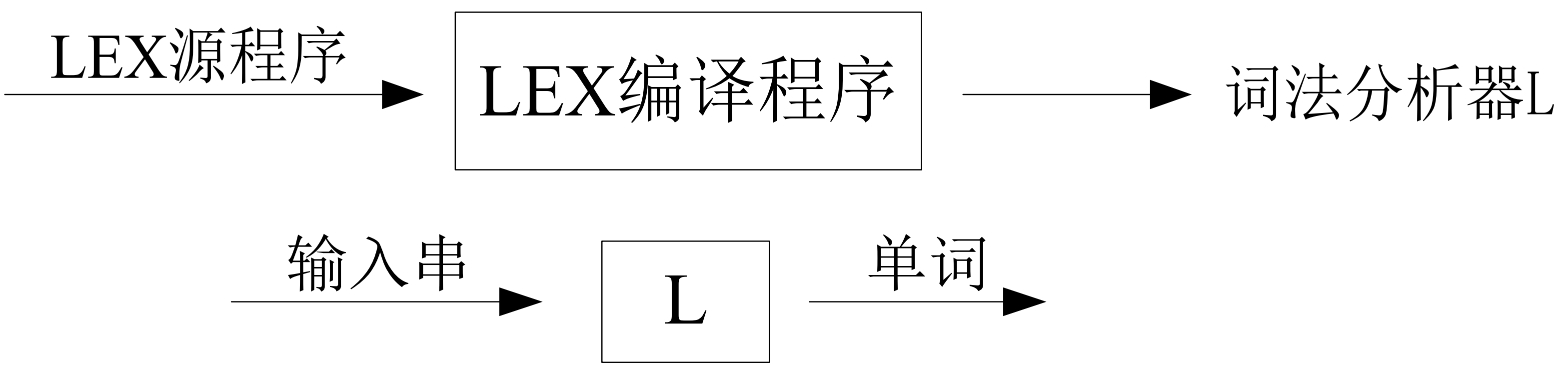
  用正规式描述单词符号，并研究如何从正规式产生识别这些单词符号的词法分析程序。

LEX：基于正规式的专门表示构造词法分析器

工具：LEX编译器 bison和flex

一、LEX语言

- 一个LEX语言程序经过编译后得到的结果程序，其作用相当于一个DFA，可用来识别和产生单词也就是说其功能即为一个词法分析器。



3.7.2 词法分析器的自动生成技术

- 一、LEX语言
- 一个LEX源程序包括三部分：定义部分、识别规则、辅助函数
- 1、定义部分

  头文件定义、常数定义、宏定义、全局变量、外部变量…
- 2、识别规则部分

  单词正规式定义
- 3、辅助函数

  识别规则部分用到的各个局部函数

3.7.2 词法分析器的自动生成技术

- 二、LEX的实现
- LEX编译程序的目的是把一个LEX程序改造为一个词法分析器L，这个词法分析器L将像自动机一样工作。
- 1．词法分析器L的工作方法

  L逐个地扫描输入串的每个字符，寻找一个最大的子串匹配某个Pi（即：当输入串已匹配某个词形时，并不立即返回，而是沿着此道路继续前进，直至不能前进为止，逐个字符地逆向搜索，找到第一个逆向搜索到的匹配词形），把这个子串放入TOKEN中，然后L调用动作子程序Ai，当Ai工作完后，L就把所得的单词符号（种别，内部码值）返回语法分析程序。下次调用L，接着往下分析。

3.7.2 词法分析器的自动生成技术

- **二、LEX的实现**
- 2.LEX程序的编译过程

（1）对每条识别规则Pi构造一个NFA Mi;

（2）引入一个新的初态X，从X画ε弧到每一个NFA Mi的初态，构造出一个NFA M;

（3）把NFA M改造为DFA M'，这个DFA M'就是能识别所有形如Pi词形的词法分析器。

这样就可用程序实现之，即编制出词法分析程序L。

实例1

- 编写程序，实现下述LEX源程序的功能

辅助定义：(1)digit→0|1|…|9

(2)letter→A|B|…|Z

识别规则:

(1)digit(digit)* {Return(4,val)}

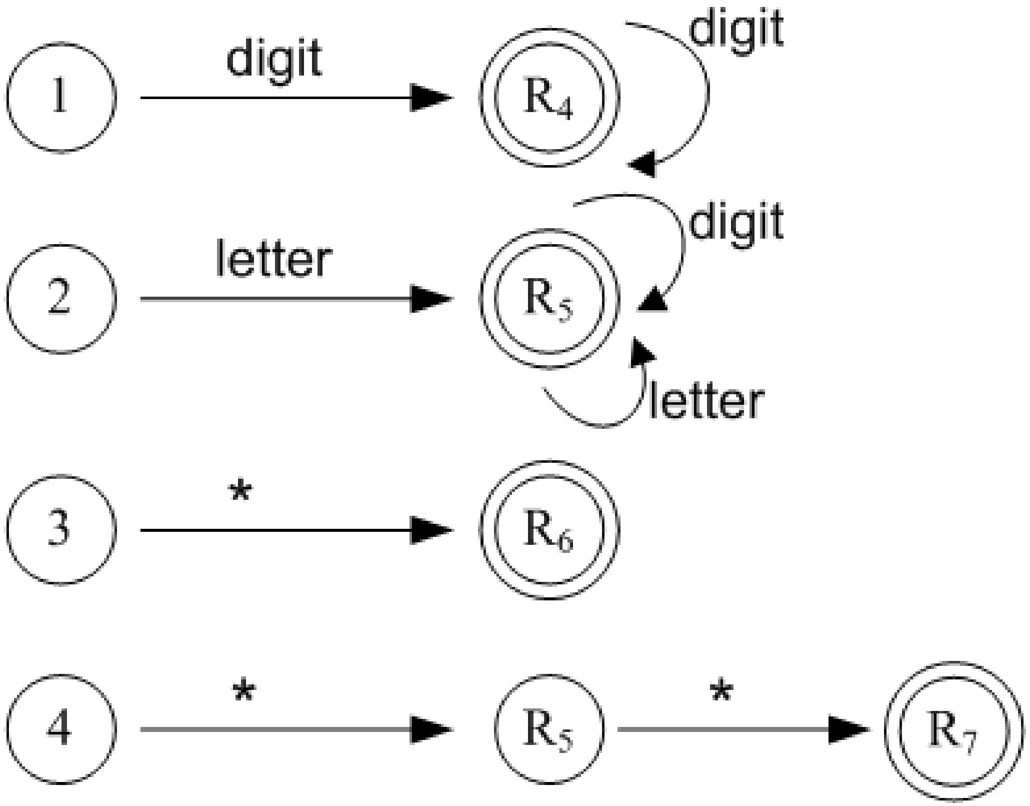
(2)letter(letter|digit)*

{Return(5.Token)}

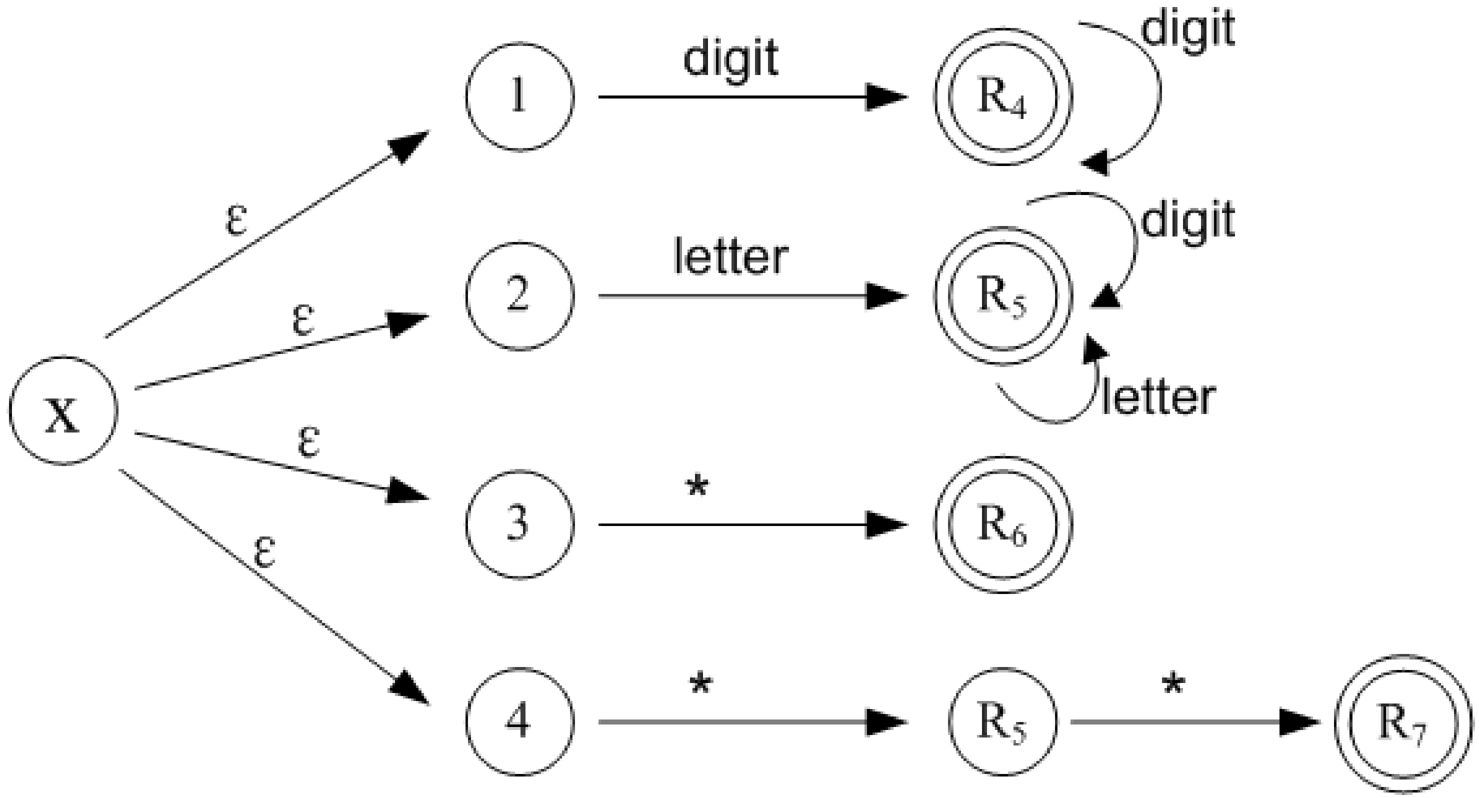
(3) * {Return(6,_)}

(4) ** {Return(7,_)}

- 解：
- 1、各识别规则的NFA为：

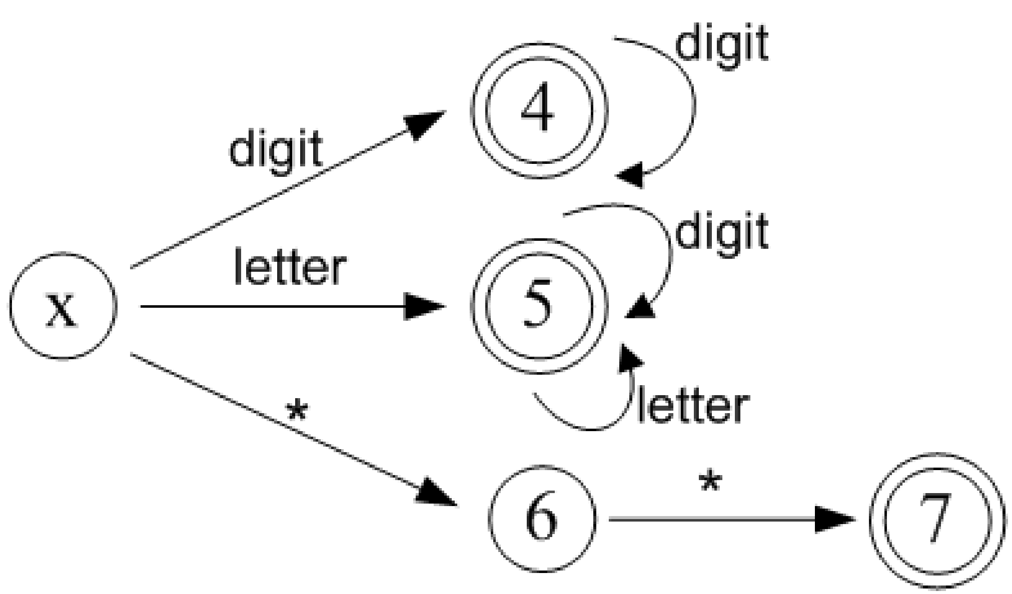


2、加入新初态X，构成NFA M整体



■ 3、确定化为DFA M′

I	I_{digit}	I_{letter}	I_{*}
$\{x, 1, 2, 3, 4\}$	$\{R_4\}$	$\{R_5\}$	$\{R_{6, 5}\}$
$\{R_4\}$	$\{R_4\}$		
$\{R_5\}$	$\{R_5\}$	$\{R_5\}$	
$\{R_{6, 5}\}$			$\{R_7\}$
$\{R_7\}$			



4、写出实现其功能的程序

```
PROGRAM LEX(input,output)
BEGIN   TOKEN:='';  getchar;
        CASE char OF
       '0'..'9':[while char IN['0'..'9']DO
          BEGIN TOKEN:=TOKEN+char; getchar
          END;  retract;
          IF VAL(token,value)  THEN return(4,value)];
       'A'..'Z':[while char IN['A'..'Z','0'..'9']DO
          BEGIN TOKEN:=TOKEN+char; getchar
          END;  retract;
          Return(5,token)];
       '*':[getchar;
          IF char='*'  THEN Return(7,-)
          ELSE  [retract;Return(6,-)]
       ELSE:ERROR;
       END{of case};
END;
```

本章小结

- 词法分析概况
- 正规文法
- 正则表达式
- 确定型有穷状态自动机
- 非确定型有穷状态自动机
- 正规文法、正规式、有限自动机之间的等价转换
- 词法分析程序的实现
- 词法分析器的自动生成