第3章

词法分析

本章主要内容

- 词法分析程序
- 单词的形式化描述工具
- 有穷自动机
- 有穷自动机和正规表达式
- 有穷自动机和正规文法
- 词法分析程序的自动构造

回顾: 什么是词法分析程序?

习题P64页1(1),4(a),5,8

3.1词法分析程序的设计

3.1.1 词法分析程序和语法分析程序接口方式

- 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

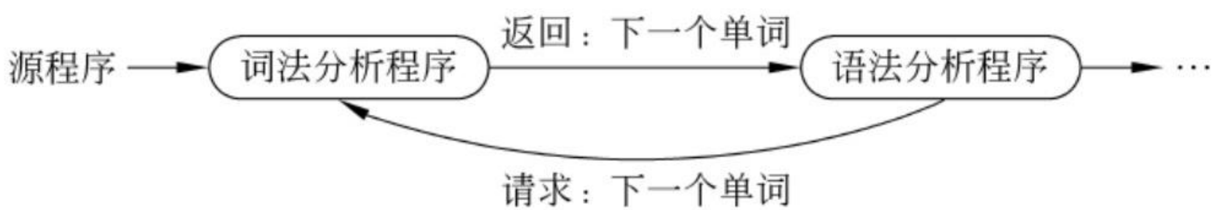


图 3.1　语法分析程序调用词法分析程序

3.1.2 词法分析程序的输出

- 词法分析程序的功能是读入源程序，输出单词符号。
- 词法分析程序所输出的单词符号常常用以下二元式表示：

  （单词种别，单词自身的值）

  - 单词的种别是语法分析需要的信息
  - 单词的值则是编译其它阶段需要的信息

- 程序语言单词的分类：

1. 标识符：用来表示各种名字
2. 字面常数：256，3 .14，true,'abc'
3. 关键字(保留字或基本字)：begin,end
4. 运算符：如，+、－、*、/ 等等
5. 界符：如逗号，分号，冒号等

单词种类用整数编码表示:

- **例**
  if i=5 then x:=y;

保留字 if	(3,'if')
标识符 i	(1,指向 i 的符号表入口)
等号=	(4,'=')
常数 5	(2,'5')
保留字 then	(3,'then')
标识符 x	(1,指向 x 的符号表入口)
负值号 :=	(4':=')
标识符 y	(1,指向 y 的符号表入口)
分号;	(5,';')

PL/0编译程序中词法分析程序

- 文法

<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> ::= (|) | , | ; | .

词法单位
(31个单词种别)

标识符 1 个 : ident
无符号整数 1 个 : number
保留字13个 : plus | minus | …
运算符11个 : beginsym | endsym | …
界符5个 : lparen | rparen | …

PL/0二元组示例

- 输出表示（单词种别，单词自身的值）。
  A=B+2

(ident,指向A的符号表的入口指针)

(becomes,   )

(ident,指向B的符号表的入口指针)

(plus,   )

(number, 2   )

词法分析程序

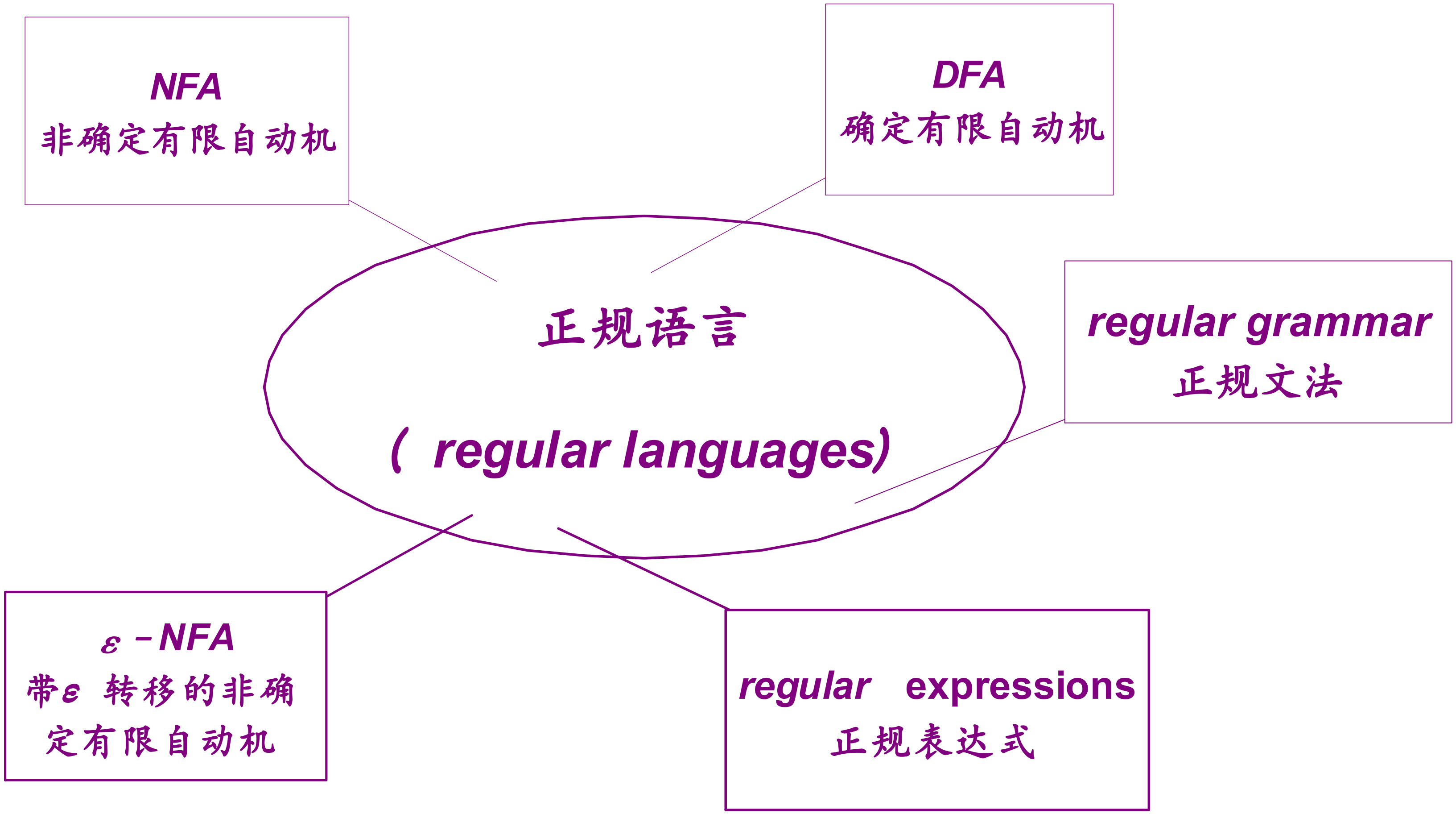
- 主要任务:
  - 读源程序，产生单词符号
- 其他任务:
  - 滤掉空格，跳过注释、换行符
  - 追踪换行标志，复制出错源程序，
  - 宏展开，…

```c
* 词法分析，获取一个符号
*/
int getsym()
{
    int i,j,k;

    /* the original version lacks "\r", thanks to foolevery */
    while (ch==' ' || ch==10 || ch==13 || ch==9)  /* 忽略空格、换行、回车和TAB */
    {
        getchdo;
    }
    if (ch>='a' && ch<='z')
    {           /* 名字或保留字以a..z开头 */
        k = 0;
        do {
            if(k<al)
            {
                a[k] = ch;
                k++;
            }
            getchdo;
        } while (ch>='a' && ch<='z' || ch>='0' && ch<='9');
        a[k] = 0;
        strcpy(id, a);
        i = 0;
        j = norw-1;
        do {    /* 搜索当前符号是否为保留字 */
            k = (i+j)/2;
            if (strcmp(id,word[k]) <= 0)
            {
                j = k - 1;
            }
            if (strcmp(id,word[k]) >= 0)
```

- 保留字
- 运算符
- 标识符
- 无符号整数
- 界符

3.3 单词的形式化描述工具



3.3.1正规文法

对于文法中每一条产生式$\alpha \to \beta$的形式都为$A \to aB$或$A \to a$，其中$A \in V_N$，$B \in V_N$，$a \in V_T$。

例3.1：<标识符>→c<字母数字>

<字母数字>→c|d|c<字母数字>|d<字母数字>

<无符号数>→ d|d<无符号数>

<运算符>→+|-|*|/|=

<界符>→,|;|(|)

其中：c表示字母，d表示数字

3.3.2 正规式

- 定义-正规式和它所表示的正规集

设字母表为Σ，辅助字母表Σ′={Φ，ε，|，.,*，(，)}。

1. ε和Φ都是Σ上的正规式，它们所表示的正规集分别为{ε}和{ }；

2.任何a∈ Σ，a是Σ上的一个正规式，它所表示的正规集为{a}；

3.假定e1和e2都是Σ上的正规式，它们所表示的正规集分别为L(e1)和L(e2)，那么(e1),e1|e2,e1.e2, e1*也都是正规式,它们所表示的正规集分别为L(e1), L(e1)∪L(e2),L(e1)L(e2)和(L(e1))*。

4.仅由有限次使用上述三步骤而定义的表达式才是Σ上的正规式，仅由这些正规式所表示的集合才是Σ上的正规集。

规定算符的优先顺序为“*”、“.”、“|”。“.”一般可省略不写。“*”、“.”和“|”都是左结合的。

$\{a^n b^m \mid (m, n>=1)\}$

- **例3.2** $\{a^n b^n \mid (n>=1)\}$

令Σ={a，b}

正规式	正规集
a	{a}
a|b	{a,b}
ab	{ab}
(a|b)(a|b)	{aa,ab,ba,bb}
a*	{ε ,a,aa,…任意个a的串}
(a|b)*	{ε ,a,b,aa,ab,…所有由a和b组成的串}
(a|b)*(aa|bb)(a|b)*	{Σ*上所有含有两个相继的a或两个相继的b组成的串}

又例3.3

1.令Σ={L，D}，其中L代表字母,D代表数字。则Σ上的正规式

$$r=L(L|D)^*$$

定义的正规集为{L,LL,LD,LDD,…}，是 “字母打头的字母数字串”,也就是Pascal和多数程序设计语言允许的的标识符的词法规则。

2.Σ={d，.，e，+，-},则Σ上的正规式 $d^*(.dd^*|\varepsilon)(e(+|-|\varepsilon)dd^*|\varepsilon)$表示的是无符号数的集合。其中d为0~9的数字。

- 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,写作e1=e2。
- 设r,s,t为正规式，正规式服从的代数规律有

1.r|s=s|r “或”服从交换律

2.r|(s|t)=(r|s)|t “或”的可结合律

3.(rs)t=r(st) “连接”的可结合律

4.r(s|t)=rs|rt

(s|t)r=sr|tr 分配律

5.εr=r,rε=r ε是"连接"的恒等元素

6.r|r=r

r*=ε|r|rr|… “或”的抽取律

例如： e1=a|b，e2 = b|a

又如： b(ab)* = (ba)*b，(a|b)* = (a*|b*)*

课堂练习

设计表示如下语言的正规表达式:

该语言中的每个字符串由交替的 0 和 1 构成。

$(01)^* \mid (10)^* \mid 0(10)^* \mid 1(01)^*$

$(\varepsilon \mid 1)(01)^*(\varepsilon \mid 0)$

$(\varepsilon \mid 0)(10)^*(\varepsilon \mid 1)$

3.3.3 正规文法到正规式

- 一个正规语言可以由正规文法定义， 也可以由正规式定义，对任意一个正规文法，存在一个定义同一个语言的正规式；反之，对每个正规式，存在一个生成同一个语言的正规文法。

- 对Σ上的正规式r ,存在一个正规G=(V_N,V_T,P,S)使L(G)=L(r) 。

首先，$V_T=\Sigma$， $S\in V_N$，生成产生式：

$S\rightarrow r$， 且S为G的开始符。

R.1： 对形如 $A\rightarrow r_1r_2$的产生式，生成产生式：

$A\rightarrow r_1B$ ， $B\rightarrow r_2$ $B\in V_N$

R.2： 对形如$A\rightarrow r^*r_1$的产生式，生成产生式：

$A\rightarrow rB$， $A\rightarrow r_1$， $B\rightarrow rB$， $B\rightarrow r_1$ $B\in V_N$

R.3： 对形如$A\rightarrow r_1 | r_2$的产生式，生成产生式：

$A\rightarrow r_1$ ， $A\rightarrow r_2$

不断应用上述规则进行变换，直到每个产生式右端只含一个V_N 。

例3.4： $\Sigma=\{a,d\}$， $r=a(a|d)^*$

正规式到正规文法

正规式	正规文法
$r_1 r_2$	$A \to r_1 B$， $B \to r_2$
$r^* r_1$	$A \to rB$，$A \to r_1$，$B \to rB$，$B \to r_1$
$r_1 \mid r_2$	$A \to r_1$ ， $A \to r_2$

例： $\Sigma=\{a,d\}$， $r=a(a \mid d)^*$

第1步： $V_T=\{a,d\}$ $S \to a(a \mid d)^*$

第2步： $S \to a(a \mid d)^*$ 变为：

$S \to aA$， $A \to (a \mid d)^*$ 利用R.1

第3步： $A \to (a \mid d)^*$ 变为：

$A \to (a \mid d)B$，$A \to \varepsilon$，$B \to (a \mid d)B$，$B \to \varepsilon$ 利用R.2

第4步： $A \to (a \mid d)B$和$B \to (a \mid d)B$变为：

$A \to aB$，$A \to bB$，$B \to aB$，$B \to bB$ 利用R.3

最后得G[s]： $S \to aA$

$A \to aB \mid dB \mid \varepsilon$

$B \to aB \mid dB \mid \varepsilon$

正规式→正规文法的示例

- 例：Σ={a,d}，　r=a(a|d)*

第1步：V_T={a,d}　　S→a(a|d)*

第2步：S→a(a|d)* 变为：

　　S→aA，　A→(a|d)*　　利用R.1

第3步：　A→(a|d)* 变为：

　　A→(a|d)B，A→ε，B→(a|d)B，B→ε　利用R.2

第4步：A→(a|d)B和B→(a|d)B变为：

　　A→aB，A→bB，B→aB，B→bB　利用R.3

最后得G[s]：　S→aA

　　　　　　　A→aB|dB|ε

　　　　　　　B→aB|dB|ε

将正规文法转换成正规式

	文法产生式	正规式
规则1	$A \to xB \quad B \to y$	$A=xy$
规则2	$A \to xA\|y$	$A=x^{*}y$
规则3	$A \to x \quad A \to y$	$A=x\|y$

例3.5:文法G[S]:

$S \to aA$

$S \to a$

$A \to aA$

$A \to dA$

$A \to a$

$A \to d$

3.4 有穷自动机

- 有穷自动机(也称有限自动机)作为一种识别装置，它能准确地识别正规集，即识别正规文法所定义的语言和正规式所表示的集合，引入有穷自动机这个理论，正是为词法分析程序的自动构造寻找特殊的方法和工具。
- 有穷自动机分为两类:
  - 确定的有穷自动机(**Deterministic Finite Automata**)
  - 不确定的有穷自动机(**Nondeterministic Finite Automata**)

3.4.1 确定的有穷自动机DFA

一个确定有限状态自动机 *DFA* (*deterministic finite automata*) 是一个五元组

$$A = (K, \Sigma, f, S, Z).$$

- 有限状态集
- 有限输入符号集
- 转移函数
- 一个开始状态
- 一个终态集合

$f : K \times \Sigma \rightarrow K$

$S \in K$

$Z \subseteq K$

例3.6

DFA M=（{S， U， V， Q}，{a， b}， f， S，{Q}）,其中f定义为:

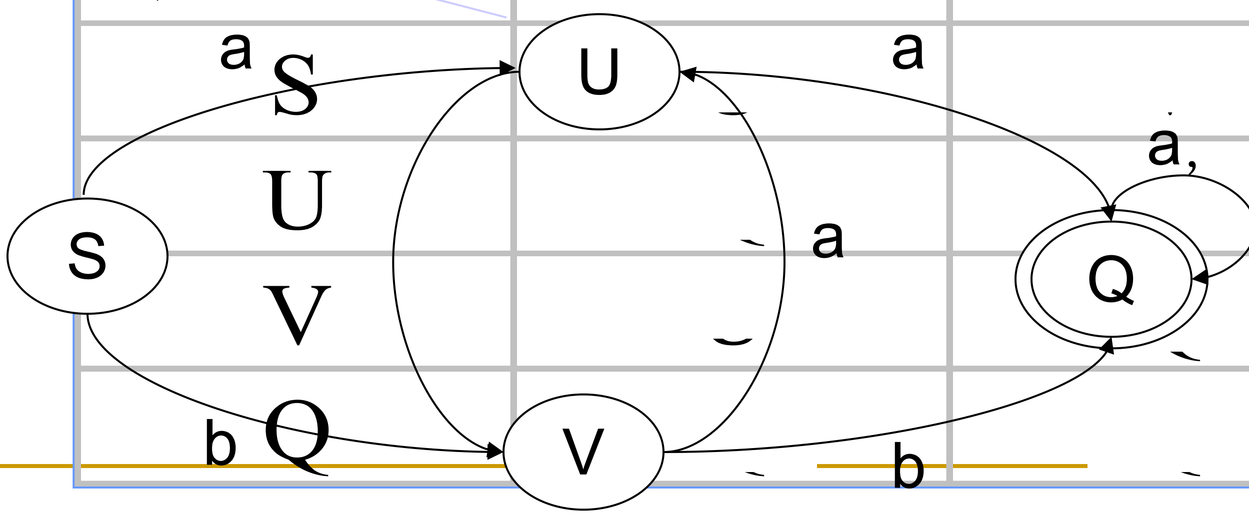
f(S， a)=U　　　　　f(V， a)=U

f(S， b)=V　　　　　f(V， b)=Q

f(U， a)=Q　　　　　f(Q， a)=Q

f(U， b)=V　　　　　f(Q， b)=Q

状态 \ 字符	a	b	
S			0
U			0
V			0
Q			1



符号串t能被M接受

- 一个输入符号串t (将它表示成t_1t_x的形式，其中$t_1 \in \Sigma$，$t_x \in \Sigma^*$)在DFA M上运行的定义为：
  - $f(Q, t_1t_x)=f(f(Q, t_1), t_x)$其中$Q \in K$
  - 扩充转换函数f是$K \times \Sigma^* \rightarrow K$上的映射，且：$f(k_i,\varepsilon)= k_i$
- Σ^*上的符号串t被M接受
  - 若$t\in \Sigma^*$，$f(S, t)=P$，其中S为 M的开始状态，$P \in Z$，Z为终态集。
  - 则称t为DFA M所接受(识别)。

例：证明t=baab被下图的DFA所接受。

f（S，baab）=f（f（S，b），aab）

=f（V，aab）=f（f（V，a），ab）

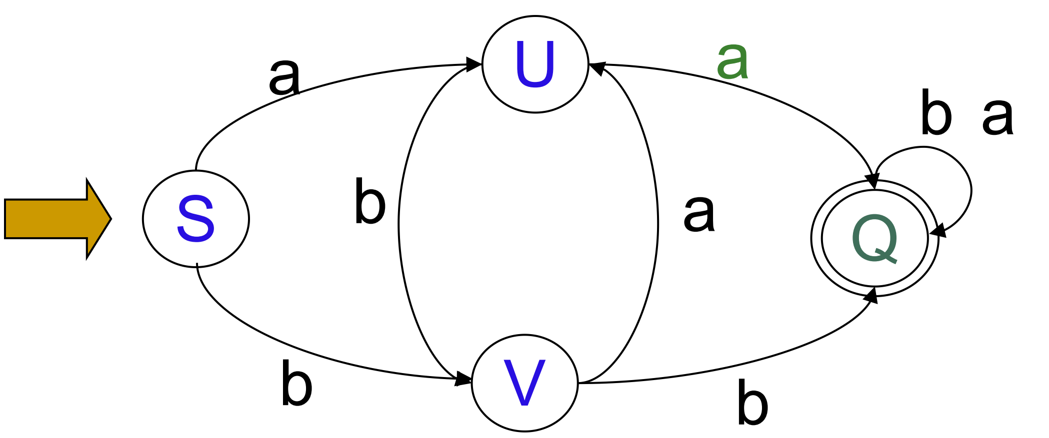
=f（U，ab）=f（f（U，a），b）

=f（Q，b）=Q

Q属于终态。

得证。

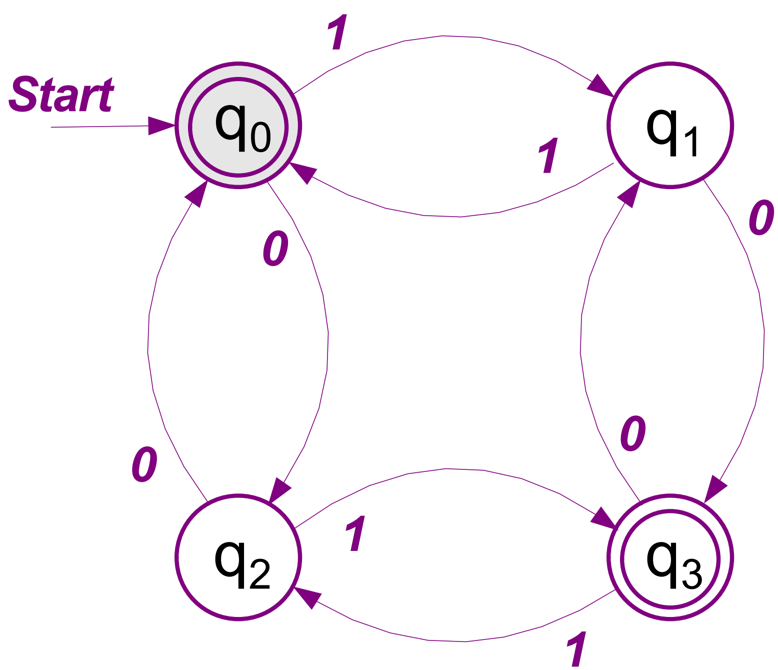
问t=ababab能被下图的DFA所接受?

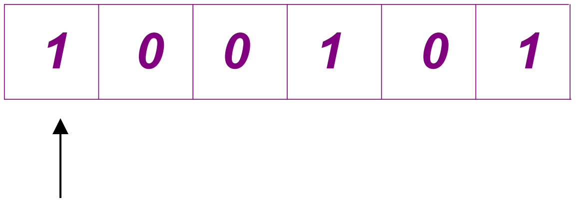


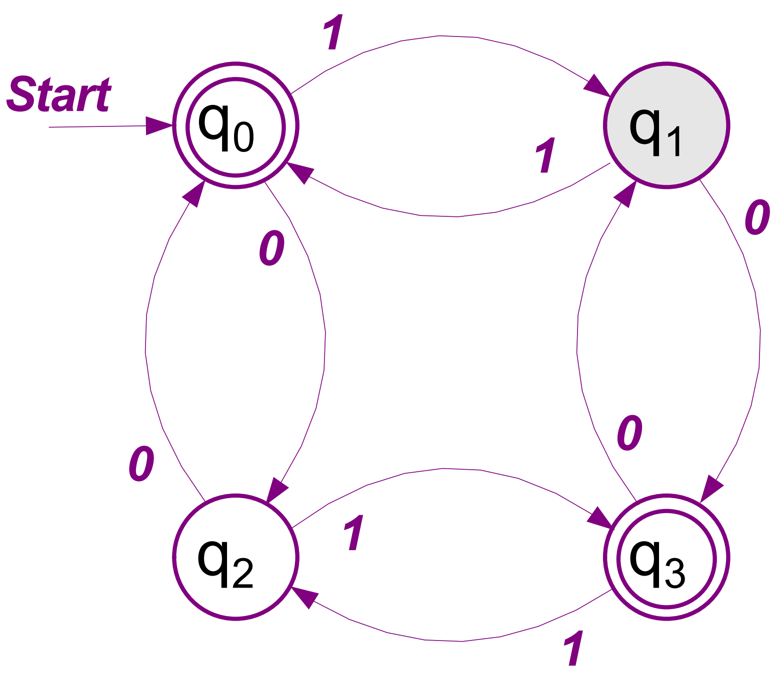
- **DFA M=（K，Σ，f，S，Z）的行为的模拟程序**

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

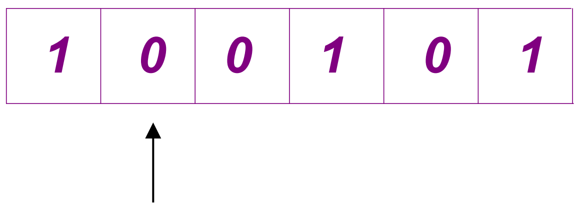
✧ DFA如何接受输入符号串



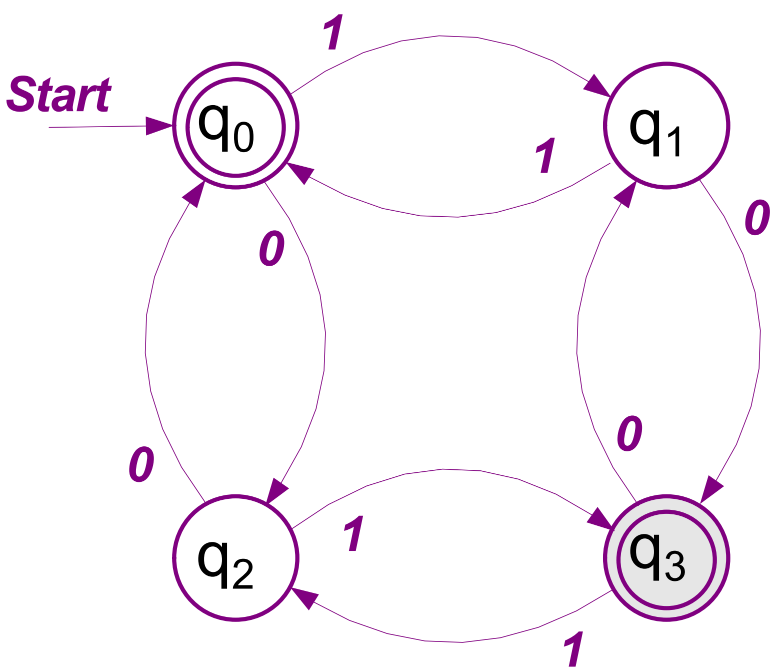


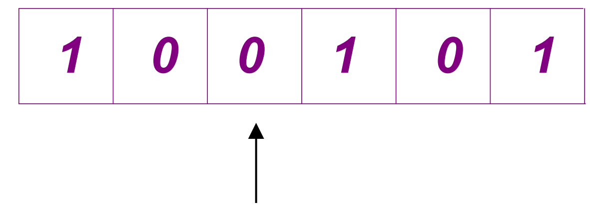


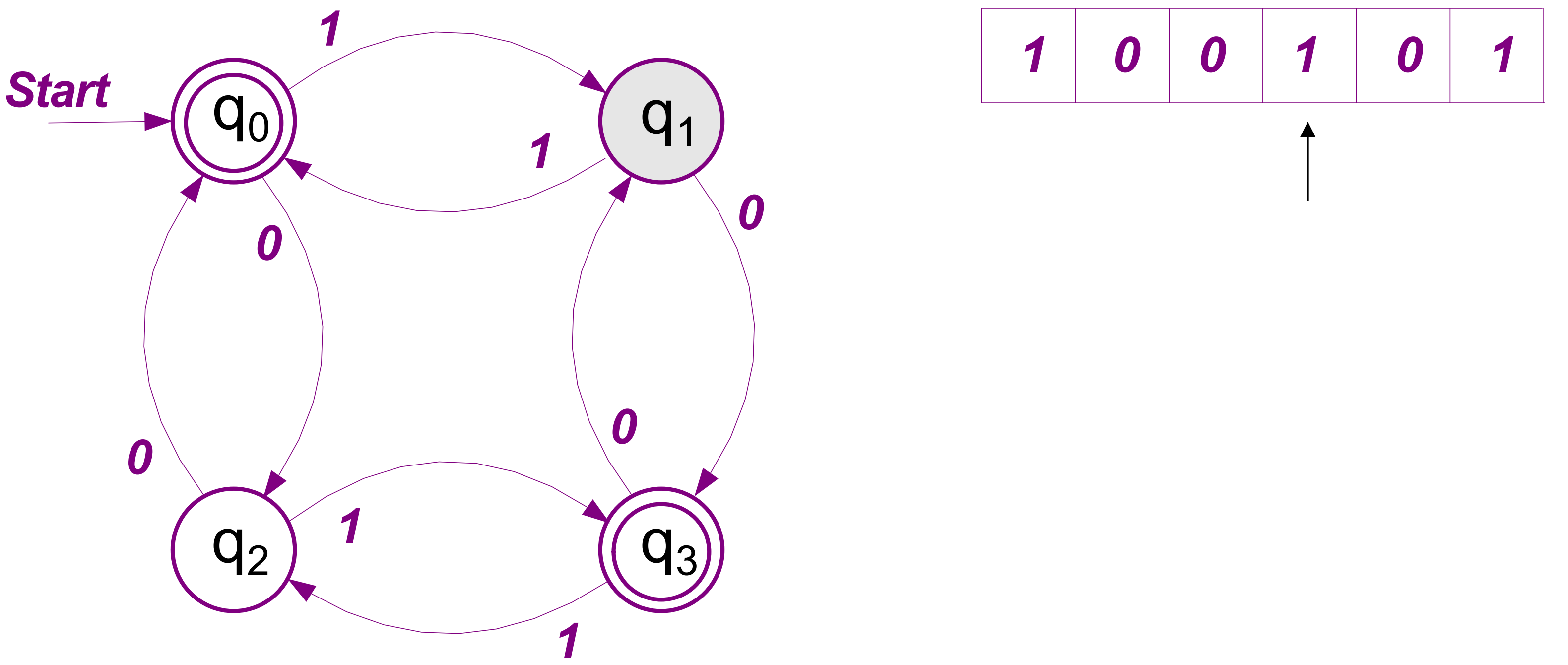
Start
q0
q1
q2
q3
1
1
0
0
0
0
1
1

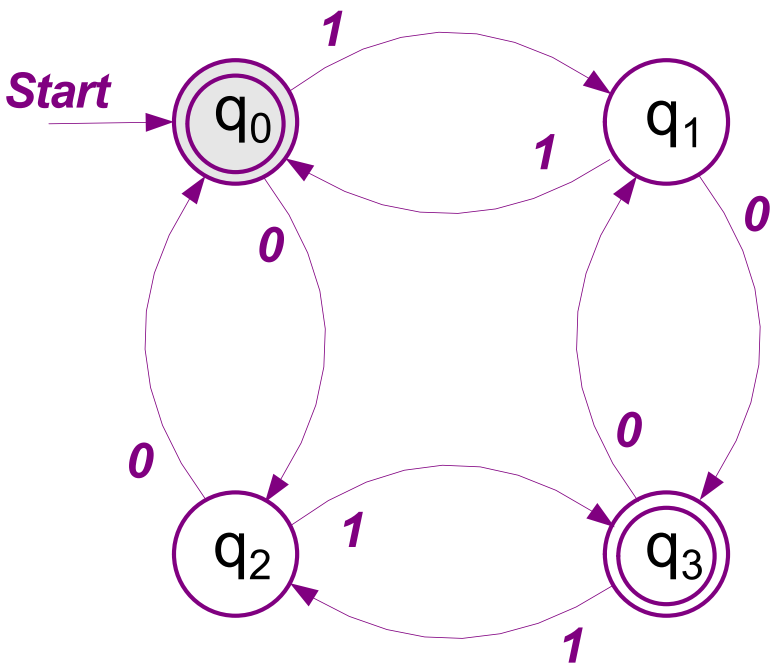


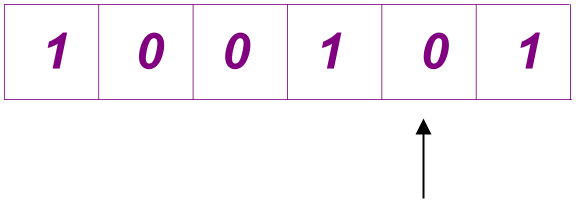
1 0 0 1 0 1

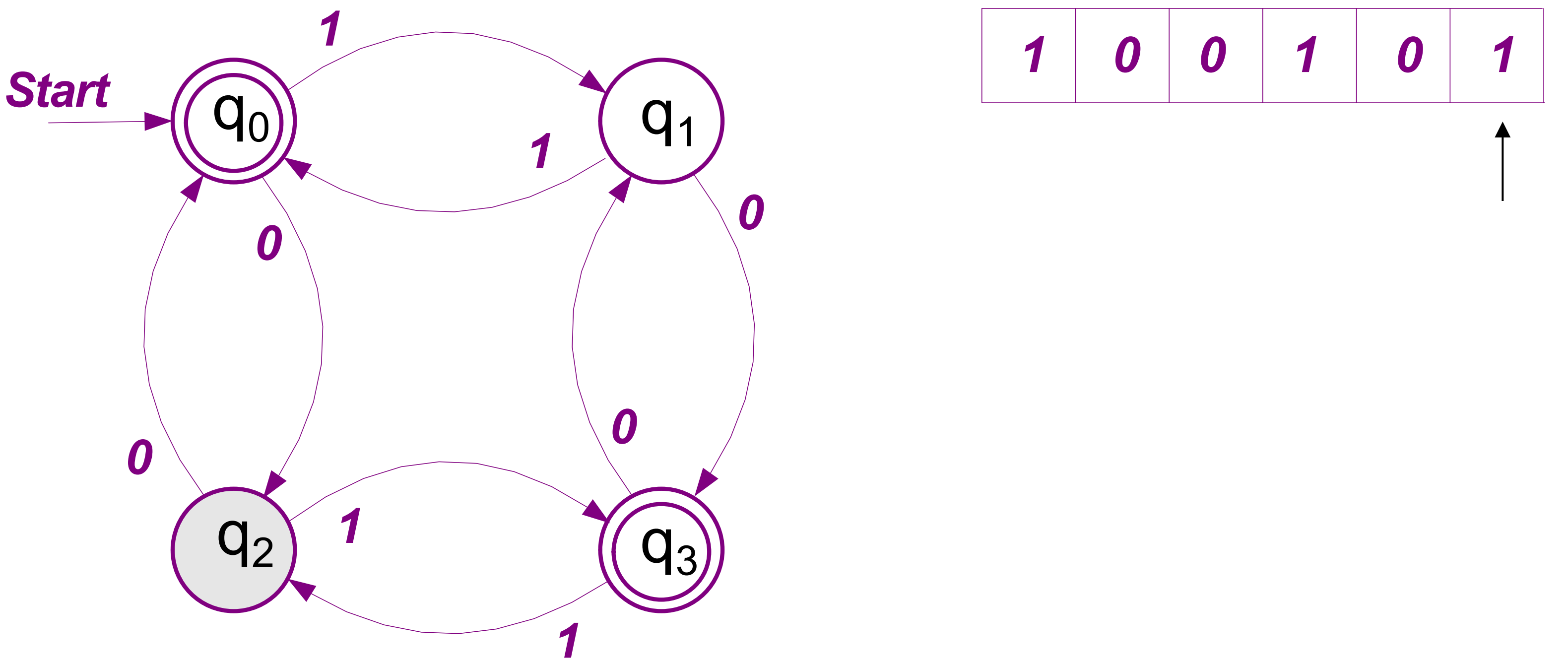

Start
q0
q1
q2
q3
1
1
0
0
0
0
1
1


1 0 0 1 0 1

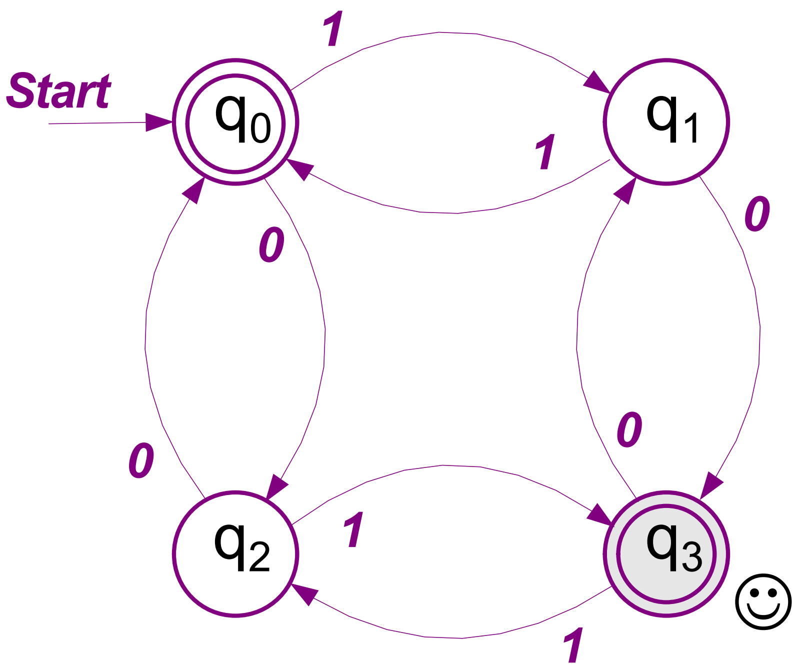

Start
q0
q1
q2
q3
1
1
0
0
0
0
1
1
1 0 0 1 0 1

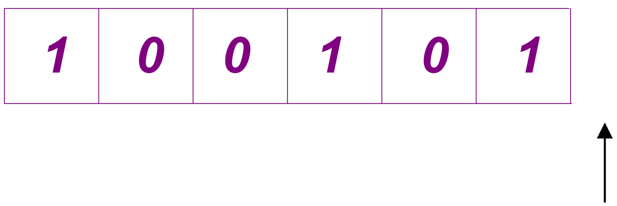

Start
q0
q1
q2
q3
1
1
0
0
0
0
1
1

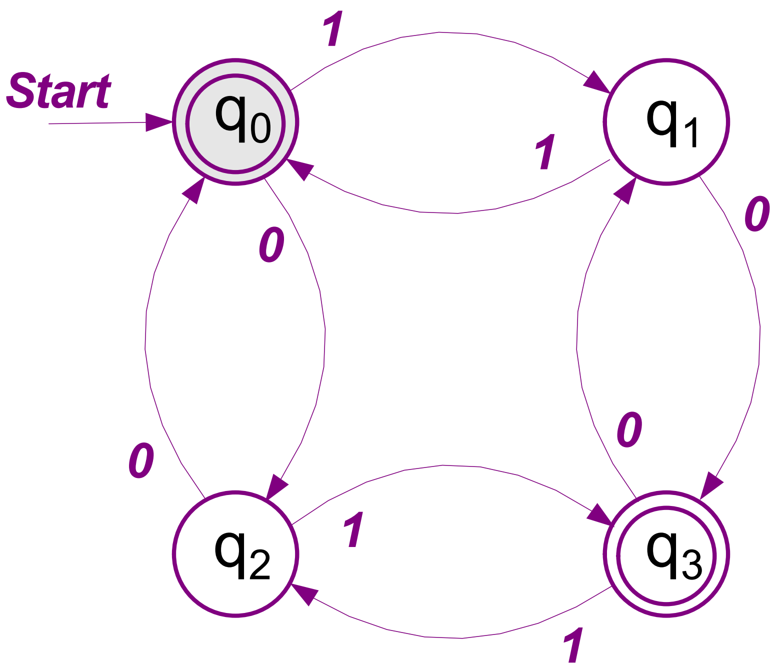

1 0 0 1 0 1



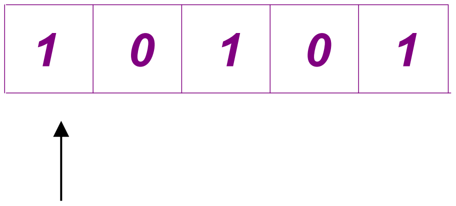
Start
q_0
q_1
q_2
q_3
1
1
0
0
0
0
1
1
1 0 0 1 0 1


Start
q_0
q_1
q_2
q_3
1
1
0
0
0
0
1
1

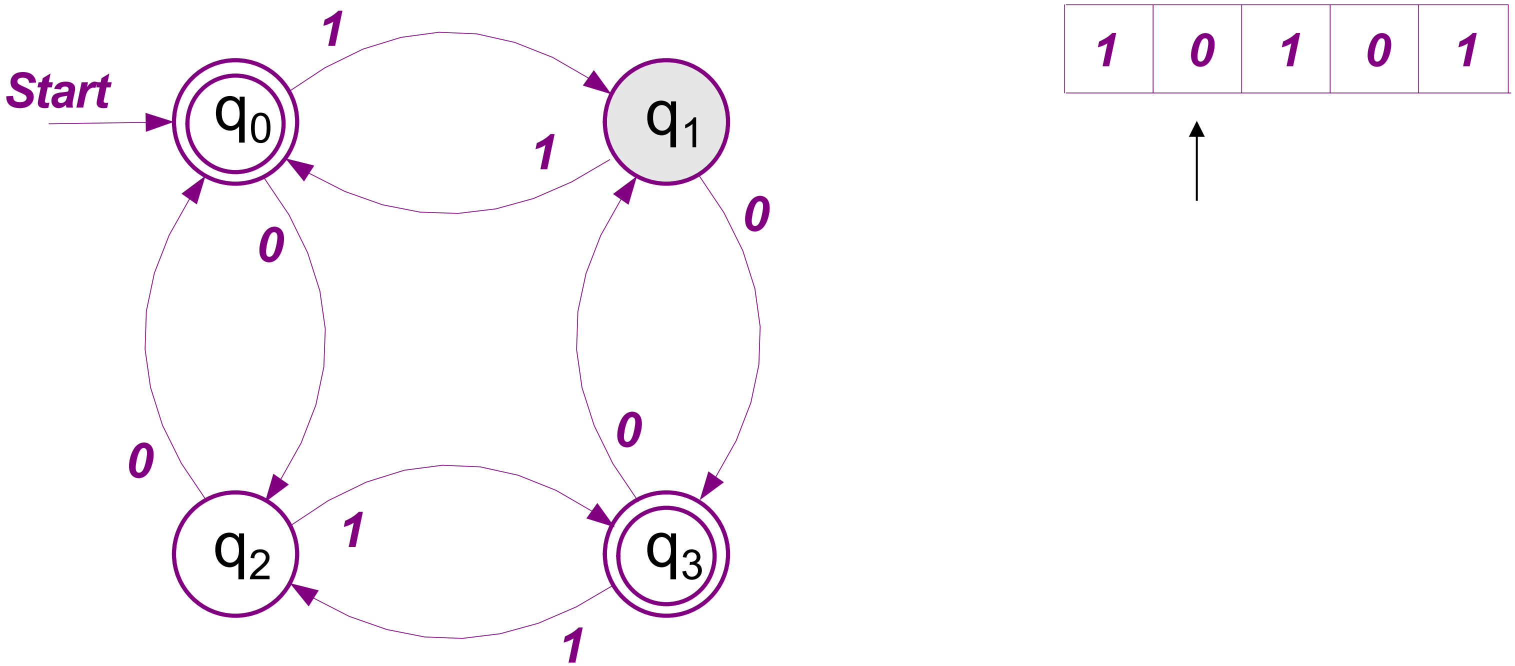

1 0 0 1 0 1

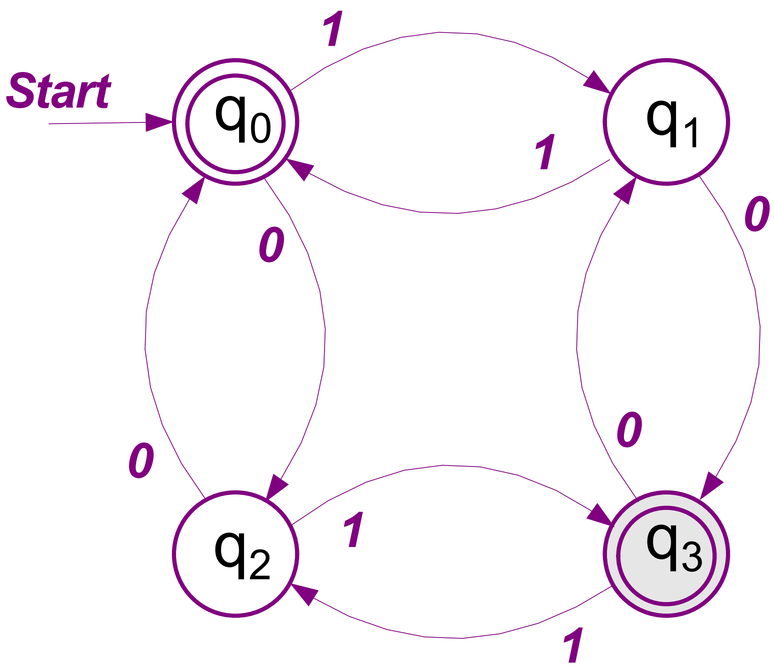


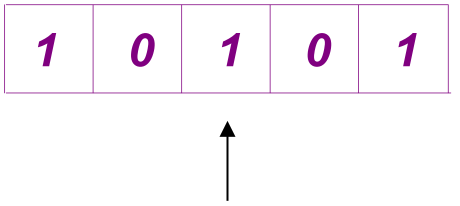
Start
q0
q1
q2
q3
1
1
0
0
0
0
1
1

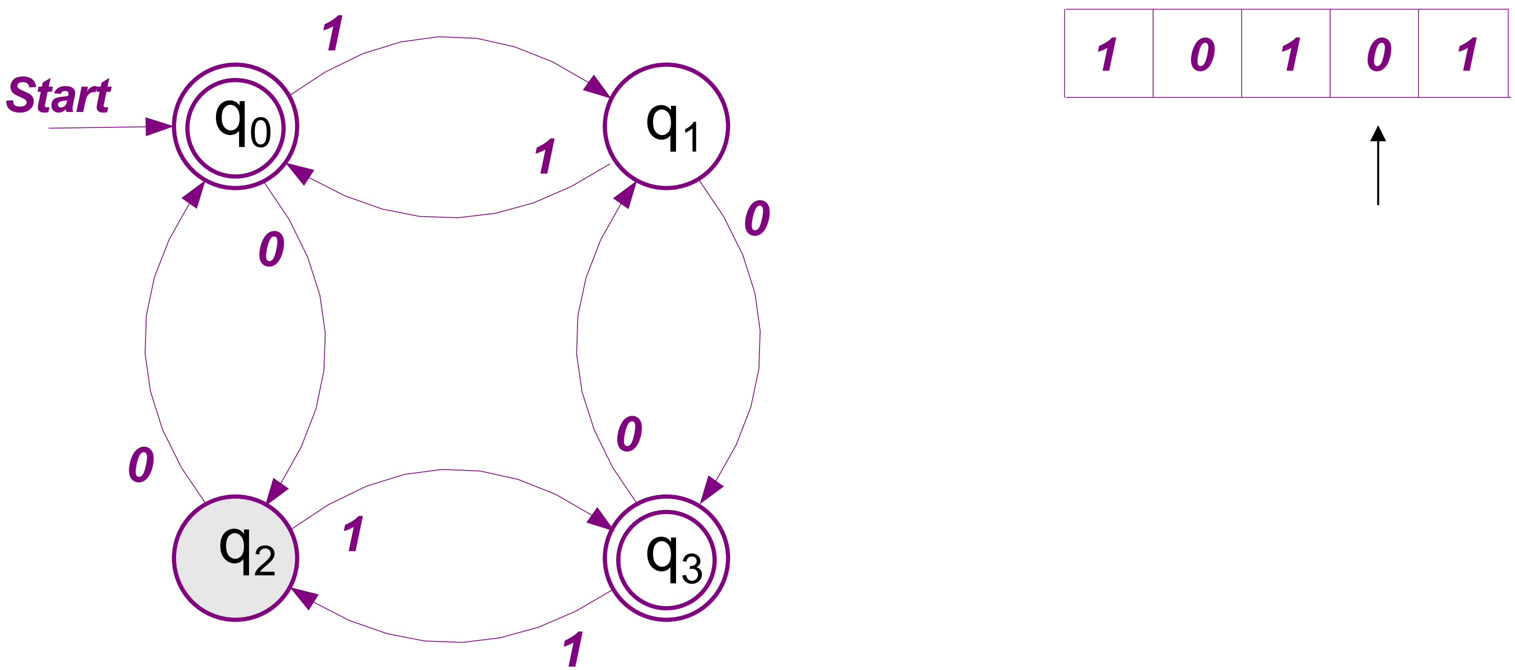


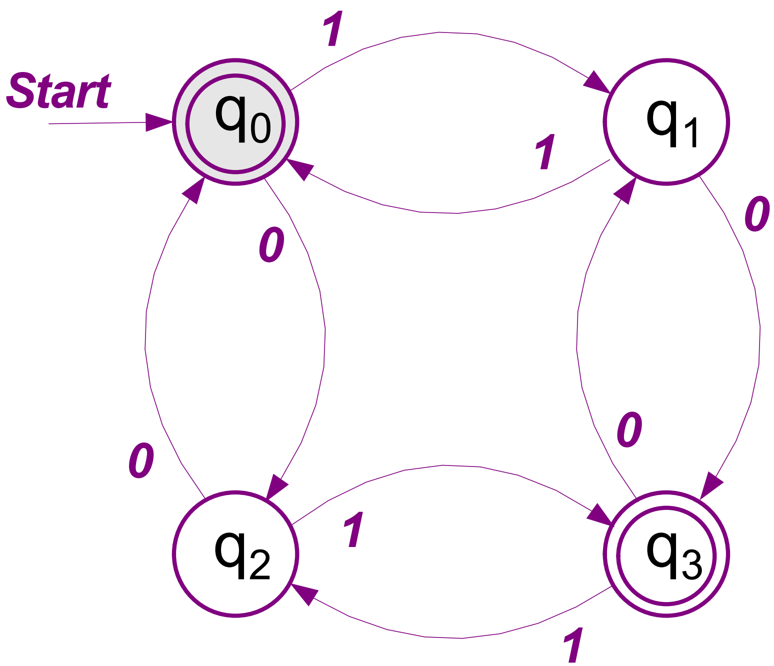
1
0
1
0
1

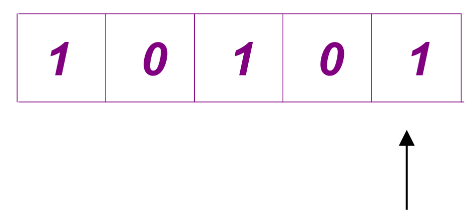

Start
q0
q1
q2
q3
1
1
0
0
0
0
1
1
1 0 1 0 1

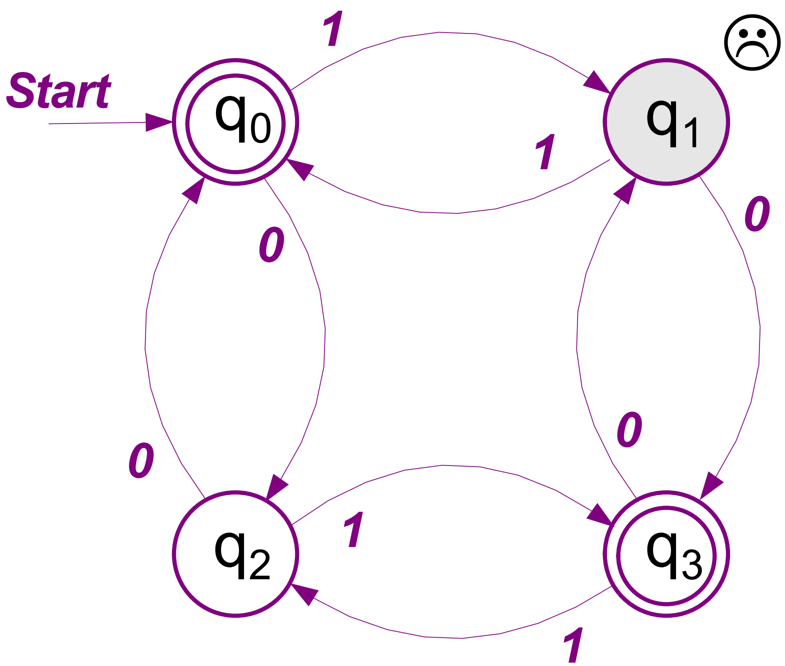

1
Start
q_0
q_1
1
0
0
0
0
q_2
1
q_3
1


1
0
1
0
1

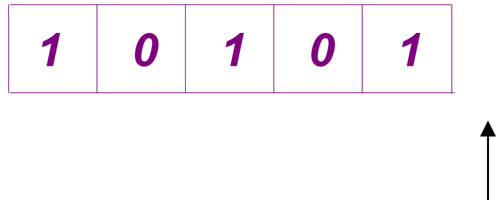

Start
q0
q1
q2
q3
1
1
0
0
0
0
1
1
1
0
1
0
1


Start
q0
q1
q2
q3
1
1
0
0
0
0
1
1


1
0
1
0
1



Start
q0
q1
q2
q3
1
1
0
0
0
0
1
1
☹



1 0 1 0 1

- **DFA M**所能接受的符号串的全体记为**L(M)**。对于任何两个有穷自动机**M**和**M′**，如果**L(M)=L(M ′)**，则称**M**与**M′**是等价的。
- 结论

  ∑上一个符号串集**V**⊂∑*是正规的，当且仅当存在一个∑上的确定有穷自动机**M**，使得**V=L(M)**。

- **DFA的确定性表现在**
  转换函数$f:K \times \Sigma \rightarrow K$是一个单值函数，也就是说，对任何状态$k \in K$和输入符号$a \in \Sigma$，$f(k,a)$唯一地确定了下一个状态。从状态转换图来看，若字母表Σ含有n个输入字符，那么任何一个状态结点最多有n条弧射出，而且每条弧以一个不同的输入字符标记。

3.4.2 不确定的有穷自动机NFA

- 定义

NFA M={K，∑，f，S，Z}，其中

1.K为状态的有穷非空集，

2.∑ 为有穷输入字母表，

3.f为K× ∑* 到K的子集的一种映射，

4.S⊆K是初始状态集，

5.Z ⊆K为终止状态集。

DFA M=（K，Σ，f，S，Z）

1.K是一个有穷非空集；

2.Σ是一个有穷输入字母表

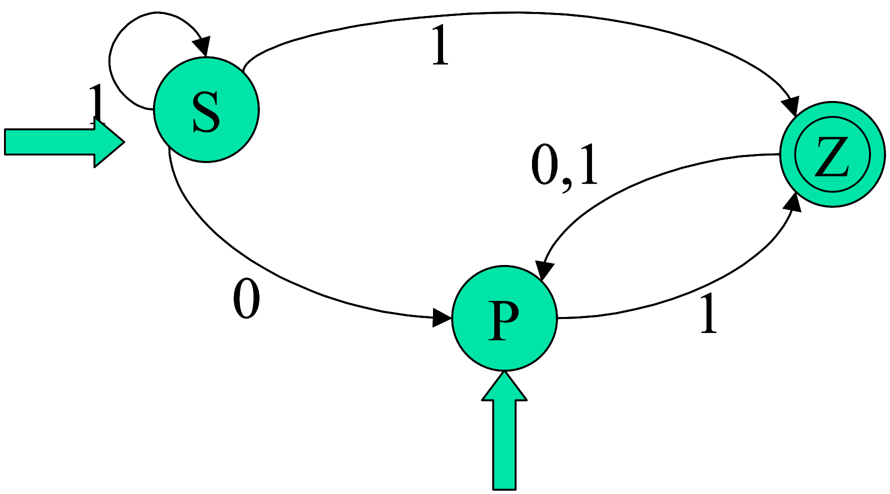
3.f是转换函数，是在K × Σ→K上的映射

4.S ∈ K是唯一的一个初态；

5.Z⊆ K是一个终态集。

- 例子

NFA M=（{S，P，Z}，{0，1}，f，{S，P}，{Z}），其中
f（S，0）={P}，f（Z，0）={P}，
f（P，1）={Z}，f（Z，1）={P}，
f（S，1）={S，Z}



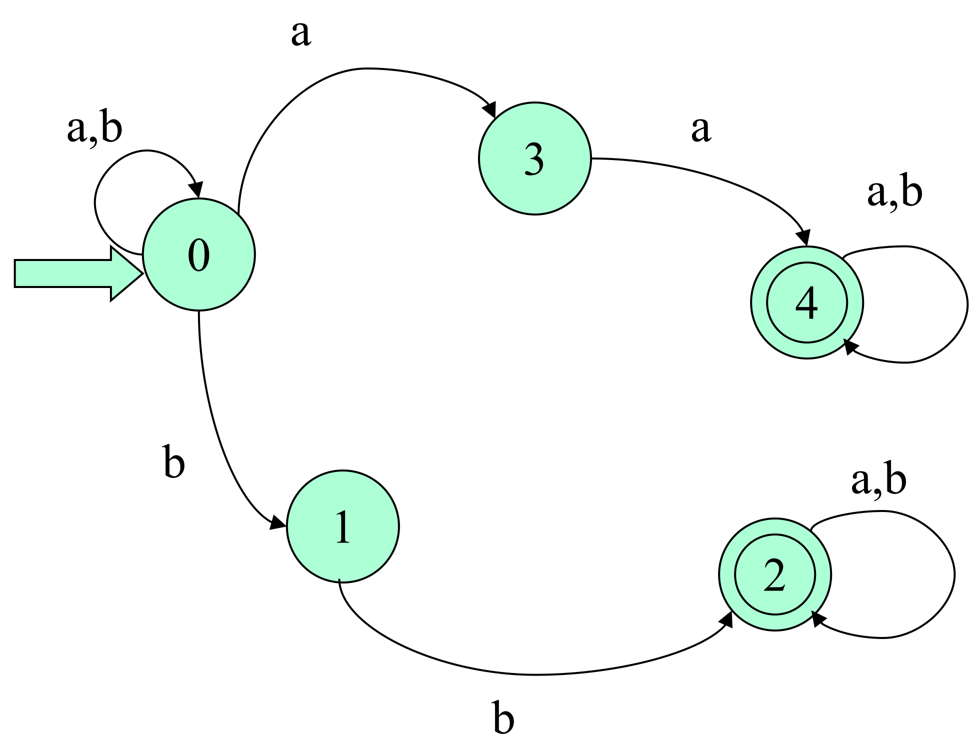
	0	1	
S	{P}	{S,Z}	0
P	{}	{Z}	0
Z	{P}	{P}	1

简化为

	0	1	
S	P	S,Z	0
P	.	Z	0
Z	P	P	1

NFA的示例2

- NFA M=({0,1,2,3,4},{a,b},f,{0},{2,4})。其中
  - 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}



- ∑*上的符号串t被NFA M接受也可以这样理解：对于Σ＊中的任何一个串t，若存在一条从某一初态结到某一终态结的道路，且这条道路上所有弧的标记字依序连接成的串(不理采那些标记为ε的弧)等于t，则称t可为NFA M所识别(读出或接受)。
- 若M的某些结既是初态结又是终态结，或者存在一条从某个初态结到某个终态结的道路,其上所有弧的标记均为ε，那么空字可为M所接受。
- **NFA和DFA的不同点：**
  - NFA可以具有ε转移
  - NFA 中f为 K×∑ *到K的子集（2^K）的一种映射。

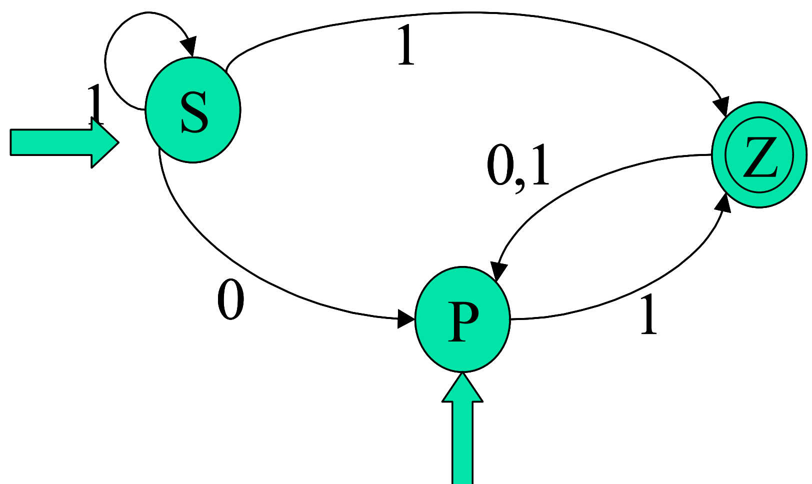
3.4.3 NFA→DFA的转换

- 对每个NFA　N一定存在一个DFA　M，使得L(M)=L(N)。对每个NFA N存在着与之等价的DFA M。

  与某一NFA等价的DFA不唯一.

- 基本思路（子集法）

  DFA的每一个状态对应NFA的一组状态. DFA使用它的状态去记录在NFA读入一个输入符号后可能达到的所有状态.

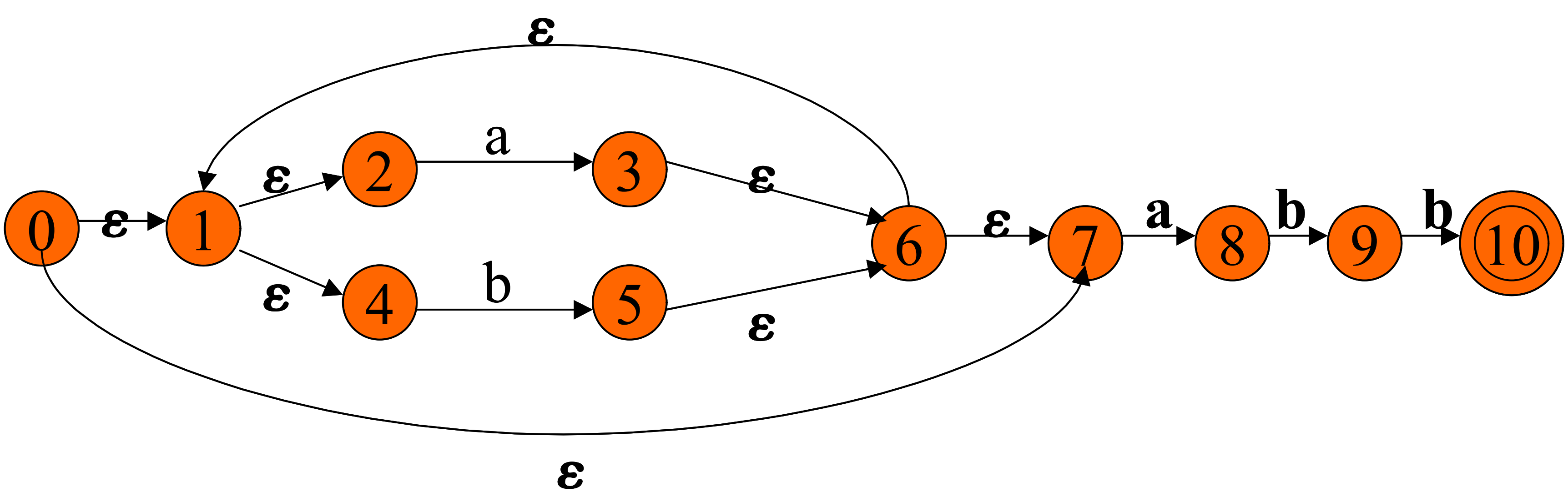


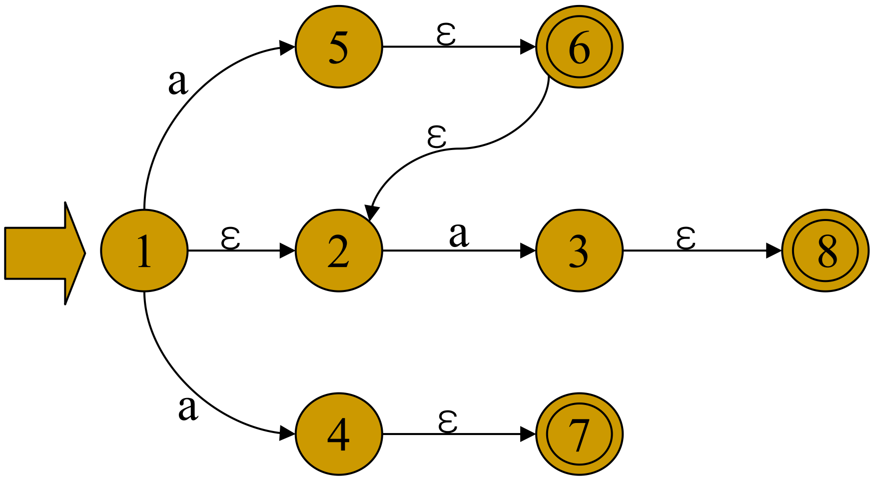
对状态集合I的几个有关运算

- 状态集合I的ε-闭包，表示为ε-closure(I)。它定义为一状态集，是状态集I中的任何状态S经任意条ε弧而能到达的状态的集合。

  状态集合I中的任何状态S都属于ε-closure(I)。
- 状态集合I的a弧转换，表示为move(I,a)。它定义为状态集合J，其中J是所有那些可从I的某一状态经过一条a弧而到达的状态的全体。
- 定义I_a = ε-closure(J)。

- 例
  ε-closure(0)=
- **move({0,1,2,4,7},a)=**
- **ε-closure({3,8})=**





1. I={1}, ε-closure(I)=

2. I={5}, ε-closure(I)=

3. I={1,2} move(I,a)= $\quad\quad$ **I_a=**

NFA确定化算法

假设NFA N=(K, ∑,f,K0,Kt)按如下办法构造一个DFA M=(S, ∑,d,S0,St)，使得L(M)=L(N)：

1. M的状态集S由K的一些子集组成。用[S1, S2,... ,Sj]表示S的元素，其中S1, S2,... , Sj是K的状态。并且约定，状态S1, S2,... ,Sj是按某种规则排列的，即对于子集{S1, S2}={S2, S1,}来说，S的状态就是[S1 S2];

2. M和N的输入字母表是相同的，即是∑;

3. 转换函数是这样定义的：

d([S1, S2,... ,Sj],a)= [R1,R2,... ,Rt],其中

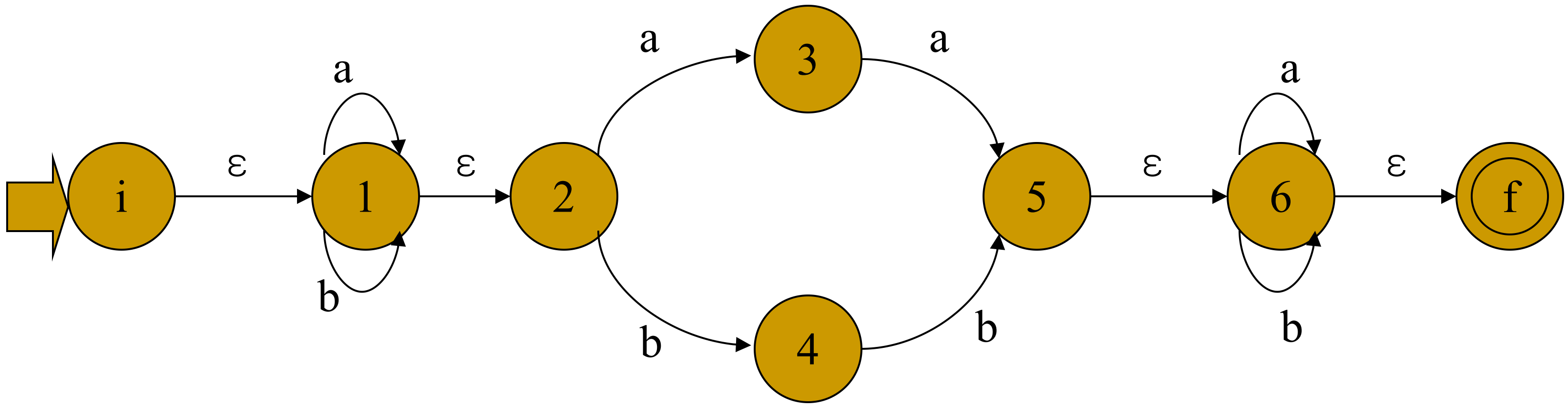
[R1,R2,... , Rt]=ε-closure(move({S1, S2,…, Sj},a))

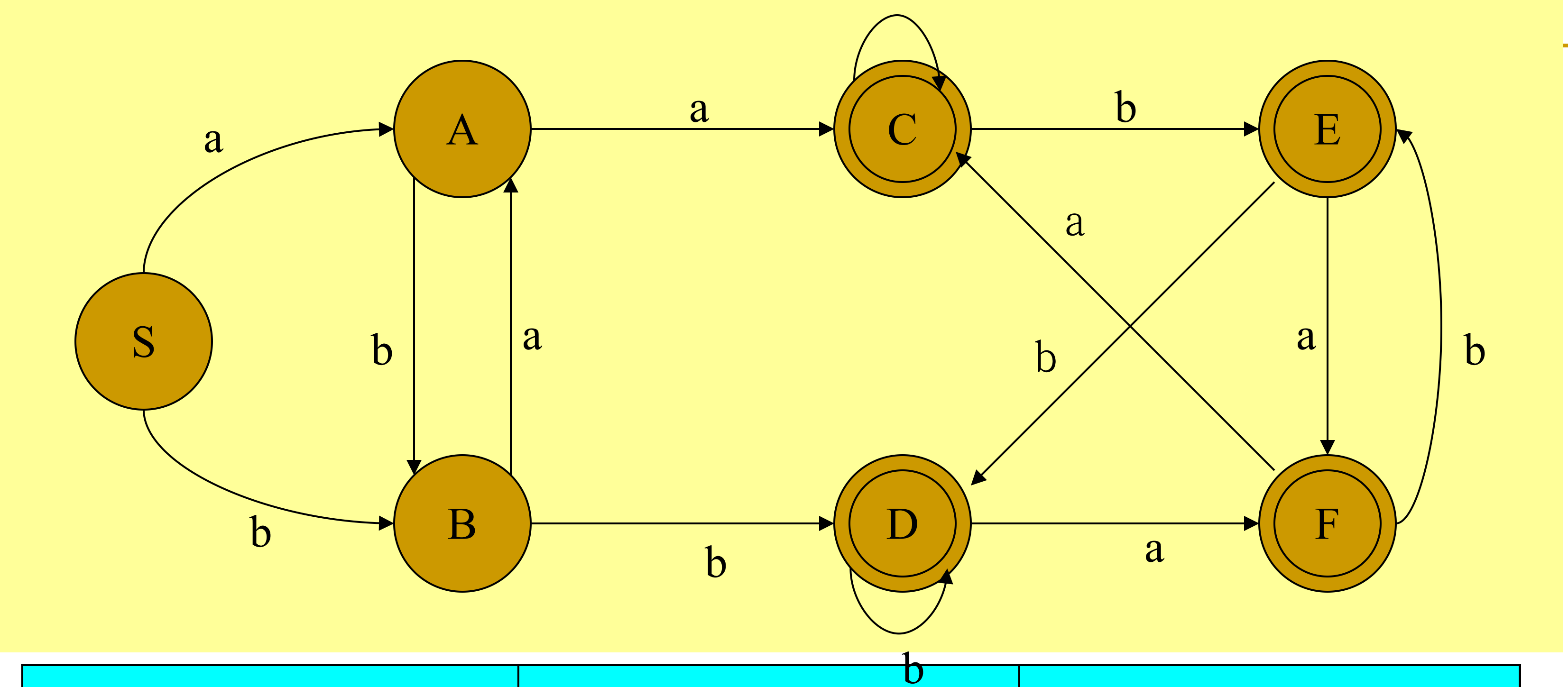
4. S0=ε-closure(K0)为M的开始状态;

5. St={[Si, Sk,…, Se]，其中[Si, Sk,... ,Se]∈S且{Si , Sk,... , Se}∩Kt≠Φ}

NFA确定化算法示例

- 例子





	I_a	I_b
{i,1,2} S	{1,2,3} A	{1,2,4} B
{1,2,3} A	{1,2,3,5,6,f} C	{1,2,4} B
{1,2,4} B	{1,2,3} A	{1,2,4,5,6,f} D
{1,2,3,5,6,f} C	{1,2,3,5,6,f} C	{1,2,4,6,f} E
{1,2,4,5,6,f} D	{1,2,3,6,f} F	{1,2,4,5,6,f} D
{1,2,4,6,f} E	{1,2,3,6,f} F	{1,2,4,5,6,f} D
{1,2,3,6,f} F	{1,2,3,5,6,f} C	{1,2,4,6,f} E

3.4.4 确定有穷自动机的化简

- 对于任一个**DFA**，存在一个唯一的状态最少的等价的**DFA**。
- 一个有穷自动机是化简了的，即是说它没有多余状态并且它的状态中没有两个是互相等价的。
- 一个有穷自动机可以通过消除多余状态和合并等价状态而转换成一个最小的与之等价的有穷自动机。

- 所谓有穷自动机的多余状态是指是指这样的状态：从该自动机的开始状态出发，任何输入串也不能到达那个状态。

例：

	0	1
S_0	S_1	S_5
S_1	S_2	S_7
S_2	S_2	S_5
S_3	S_5	S_7
S_4	S_5	S_6
S_5	S_3	S_1
S_6	S_8	S_0
S_7	S_0	S_1
S_8	S_3	S_6

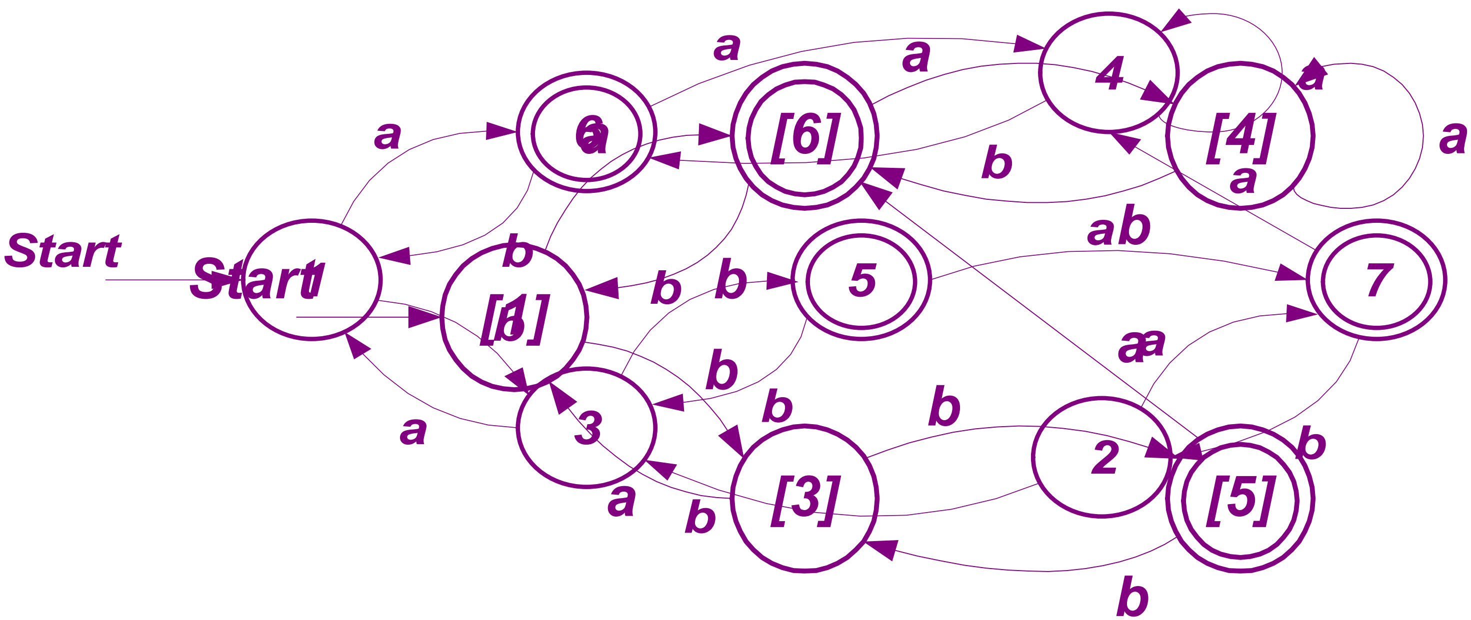
画状态图可以看出 S_4,S_6,S_8 为不可达状态应该消除

	0	1
S_0	S_1	S_5
S_1	S_2	S_1
S_2	S_2	S_5
S_3	S_5	S_1
S_5	S_3	S_1
S_7	S_0	S_1

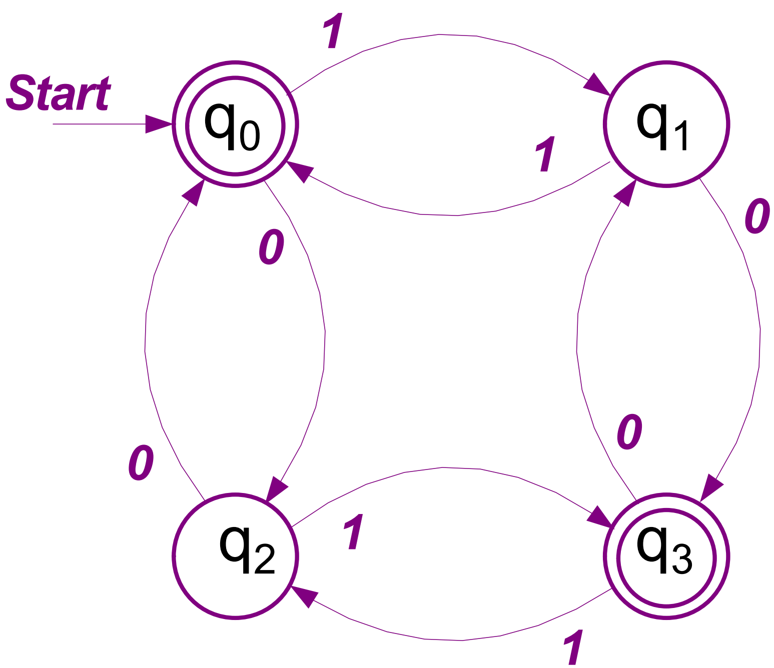
- 状态s和t的等价条件是:
  - 一致性条件：状态s和t必须同时为可接受状态或不接受状态。
  - 蔓延性条件：对于所有输入符号,状态s和t必须转换到等价的状态里。
- 有穷自动机的状态s和t不等价,称这两个状态是可区别的。

- **DFA的最小化算法- “分割法”**

　　把一个DFA的状态分成一些不相交的子集，使得任何不同的两子集的状态都是可区别的，而同一子集中的任何两个状态都是等价的。

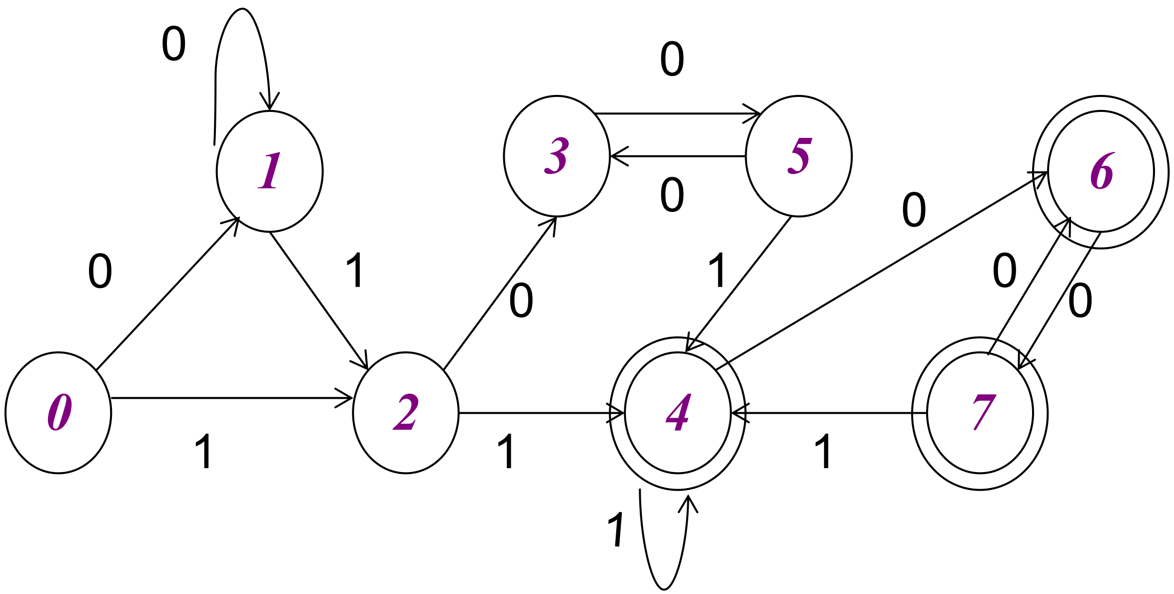


– 最小化下图表示的 DFA



课堂练习

– 最小化下图表示的 DFA



3.5有穷自动机和正规表达式

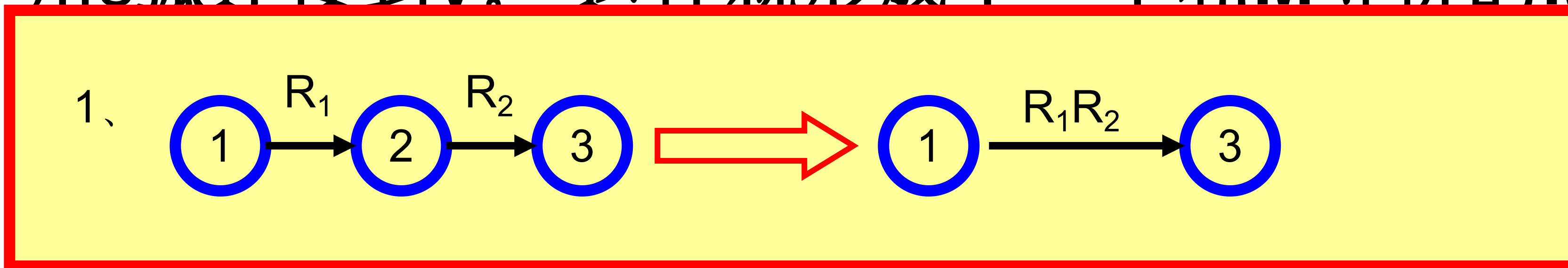
- 有穷自动机和正规表达式的等价性:

  1.对于∑上的一个NFA M，可以构造一个∑上的正规式R,使得L(R)=L(M)。

  2.对于∑上的一个正规式R，可以构造一个∑上的NFA M，使得L(M)=L(R)。

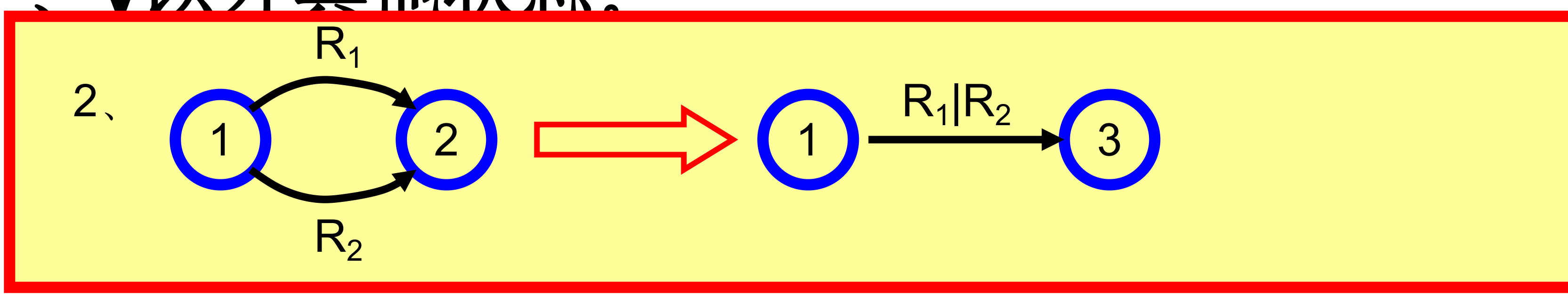
一、NFA转化为正规式

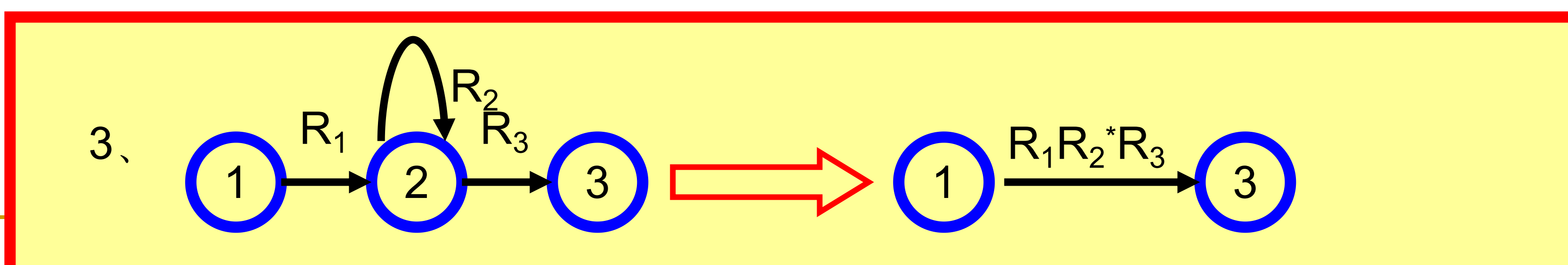
- 第一步：在**NFA M**上加两个新的状态**x**和**y**，从**x**出发用ε弧连接到**M**的所有初态，从**M**的所有终态用ε弧连接到**y**。这样就形成了一个和**M**等价的**M'**



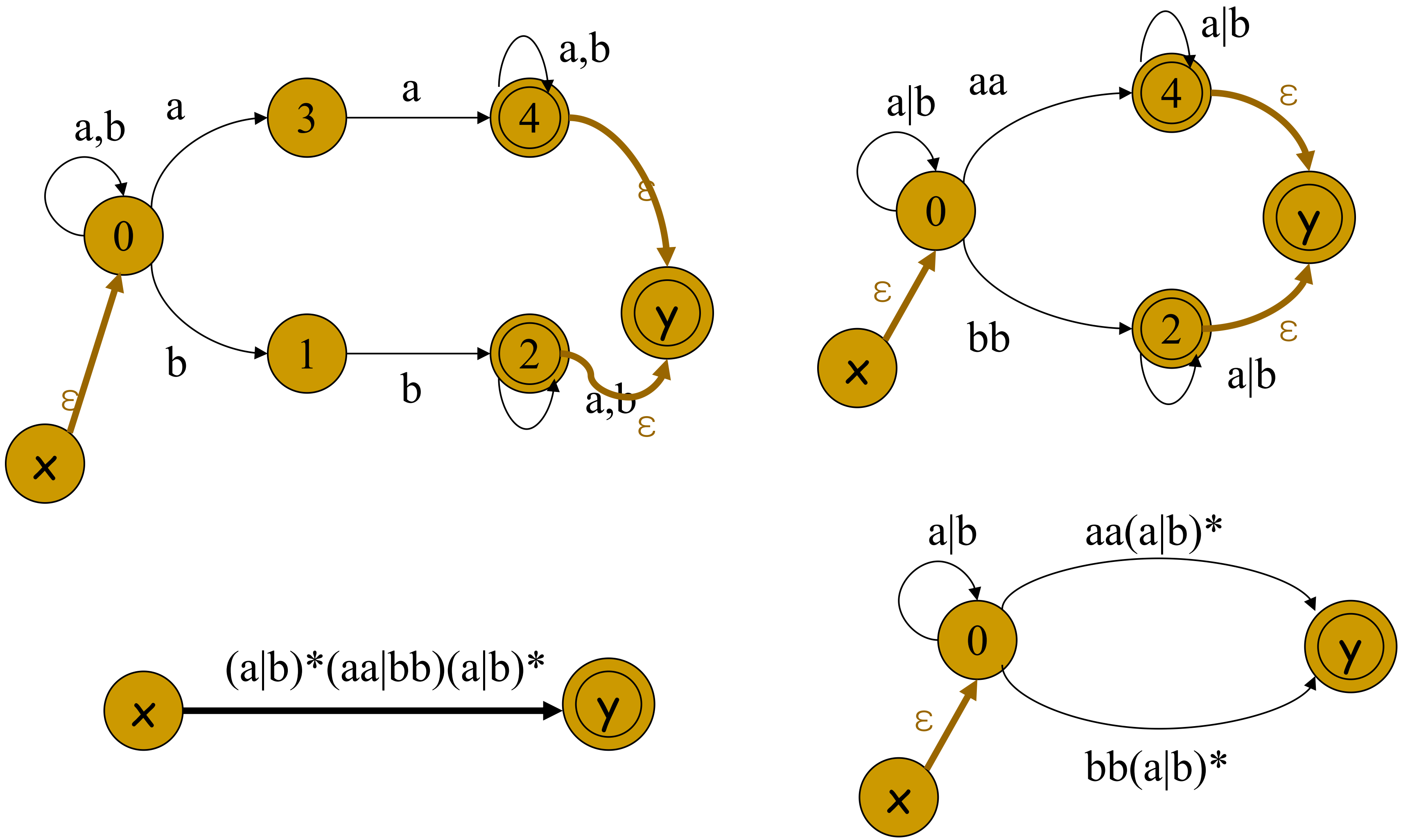
- …x

、y以外其他状态。

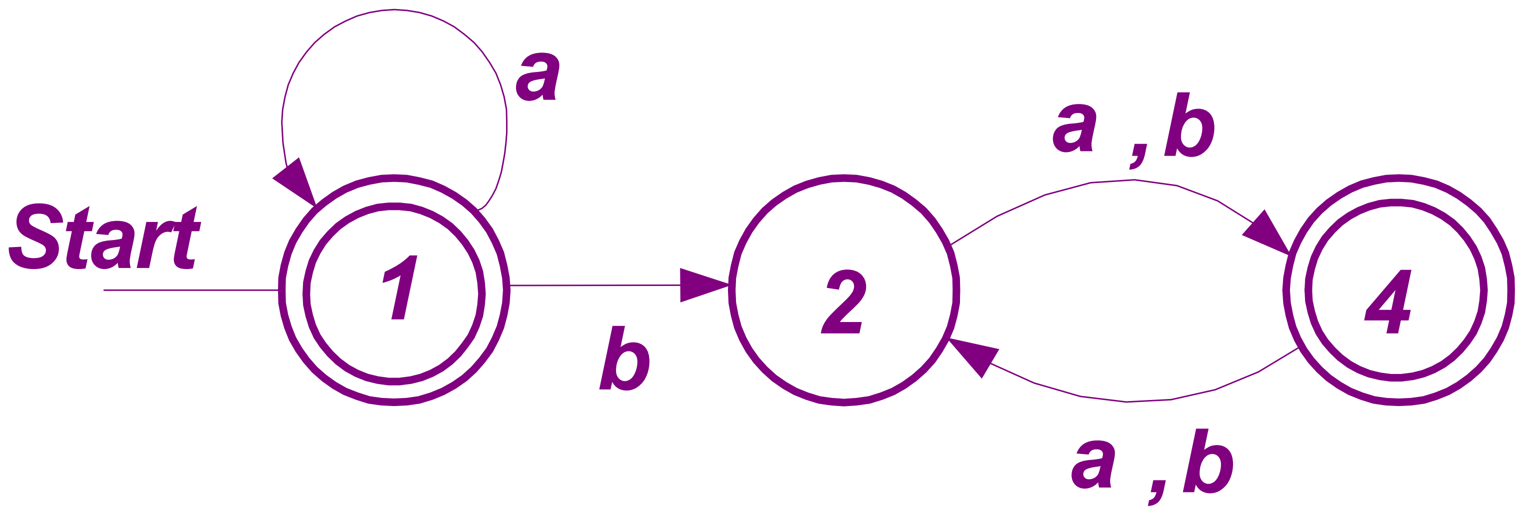




NFA转化为正规式的示例

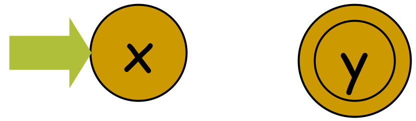


课堂练习

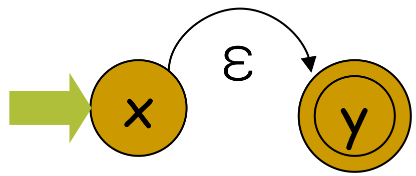


二、正规式转化为NFA

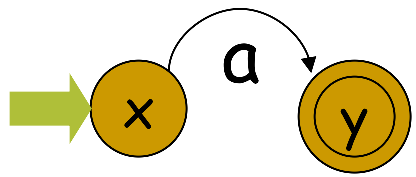
1. R=∅:



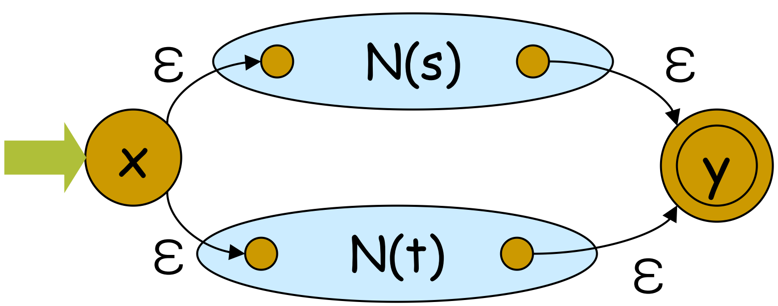
2. R= ε:



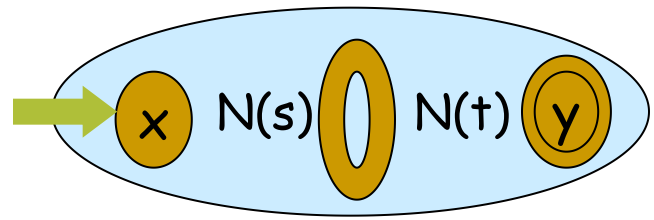
3. R= a:



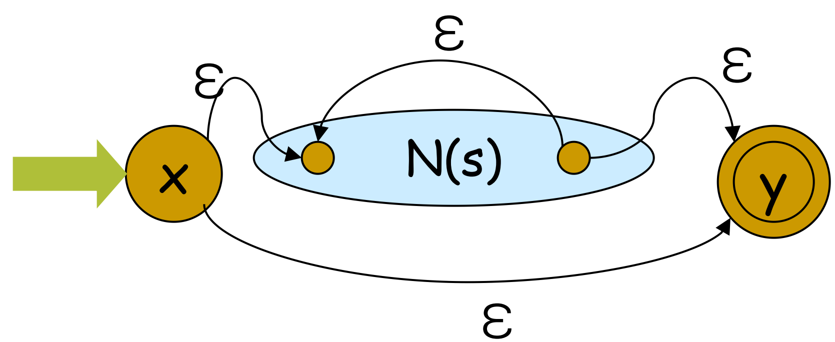
4. R= s|t:



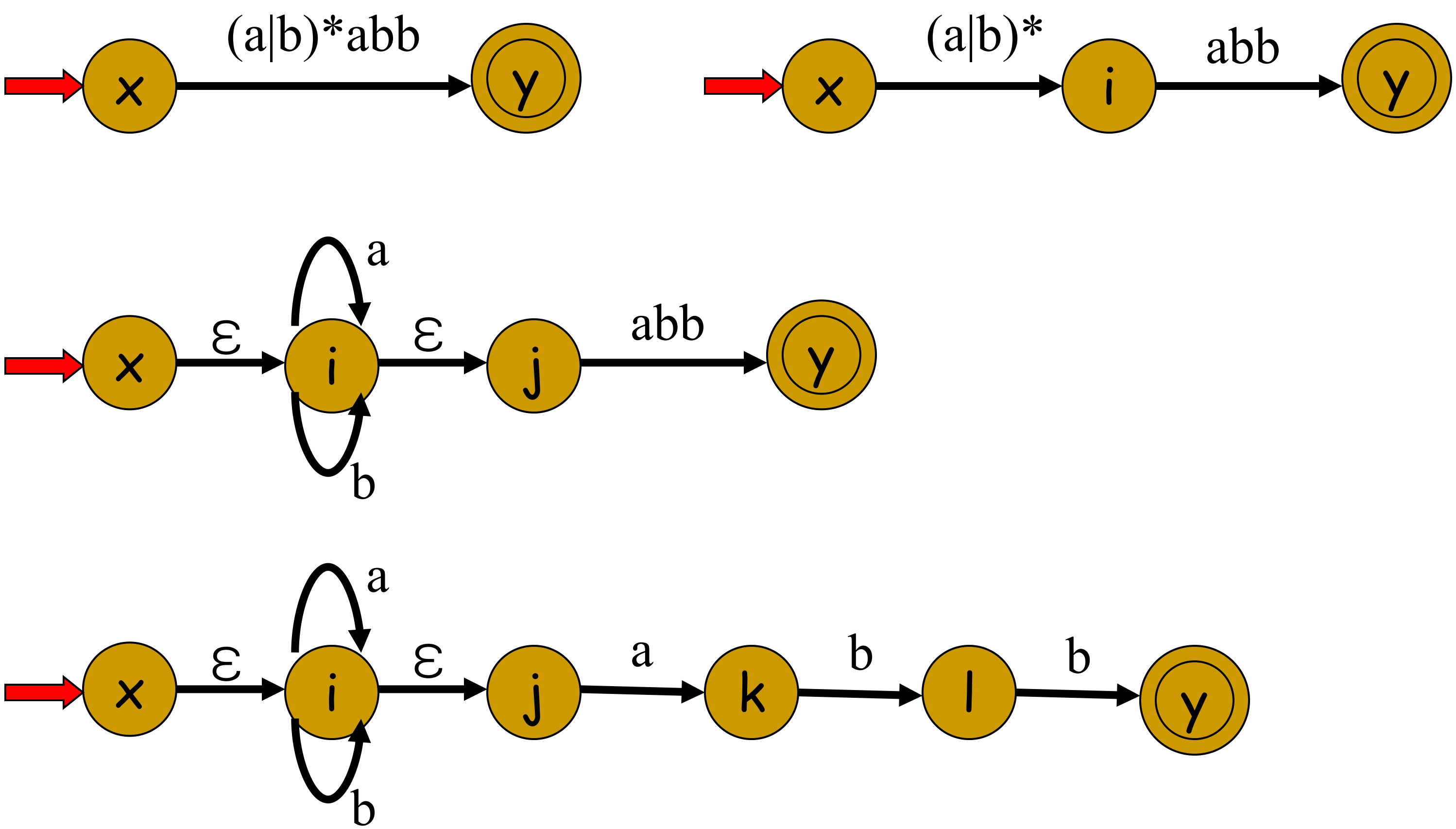
5. R= st:



6. R= s*:



正规式转化为NFA的示例



3.6 正规文法与有穷自动机间的转换

- 正规文法到NFA

1.字母表与G的终结符相同；
2.为G中的每个非终结符生成M的一个状态，G的开始符号S是开始状态S；
3.增加一个新状态Z,作为NFA的终态；
4.对G中的形如A→tB,其中t为终结符或ε，A和B为非终结符的产生式，构造M的一个转换函数f(A,t)=B；
5.对G中的形如A→t的产生式,构造M的一个转换函数 f(A,t)=Z。

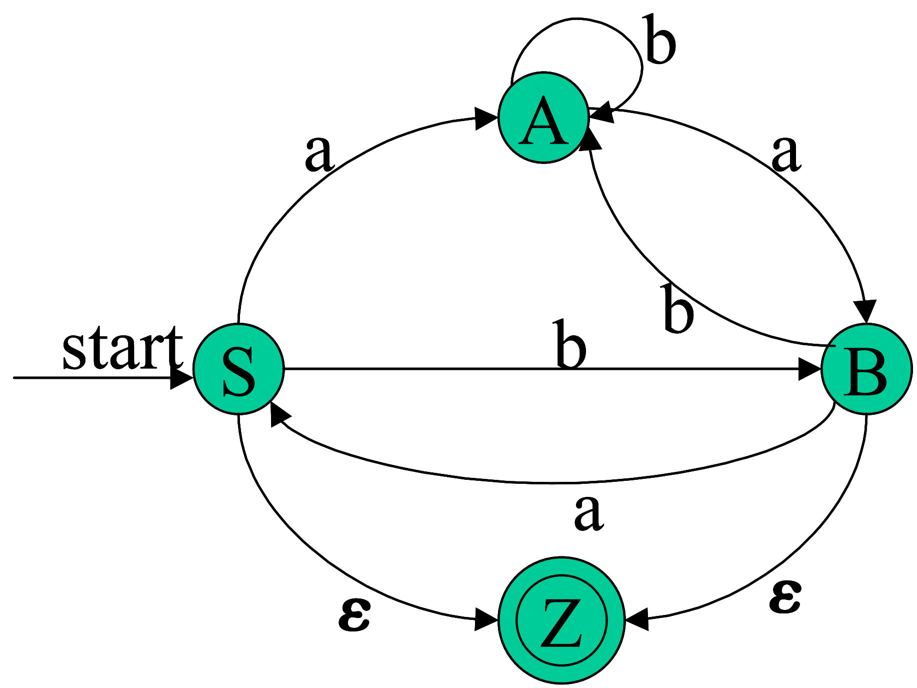
- 例:求与文法G[S]等价的NFA

G[S]:

S→aA|bB|ε

A→aB|bA

B→aS|bA|ε



- **NFA到正规文法**

  1.对转换函数f(A,t)=B，可写成一个产生式：$A \rightarrow tB$

  2.对可接受状态Z，增加一个产生式: $Z \rightarrow \varepsilon$

  3.有穷自动机的初态对应于文法的开始符号，有穷自动机的字母表为文法的终结符号集。

- 例:给出如图NFA等价的正则文法G

G[A]:

A → aB

A → bD

B → bC

C → aA

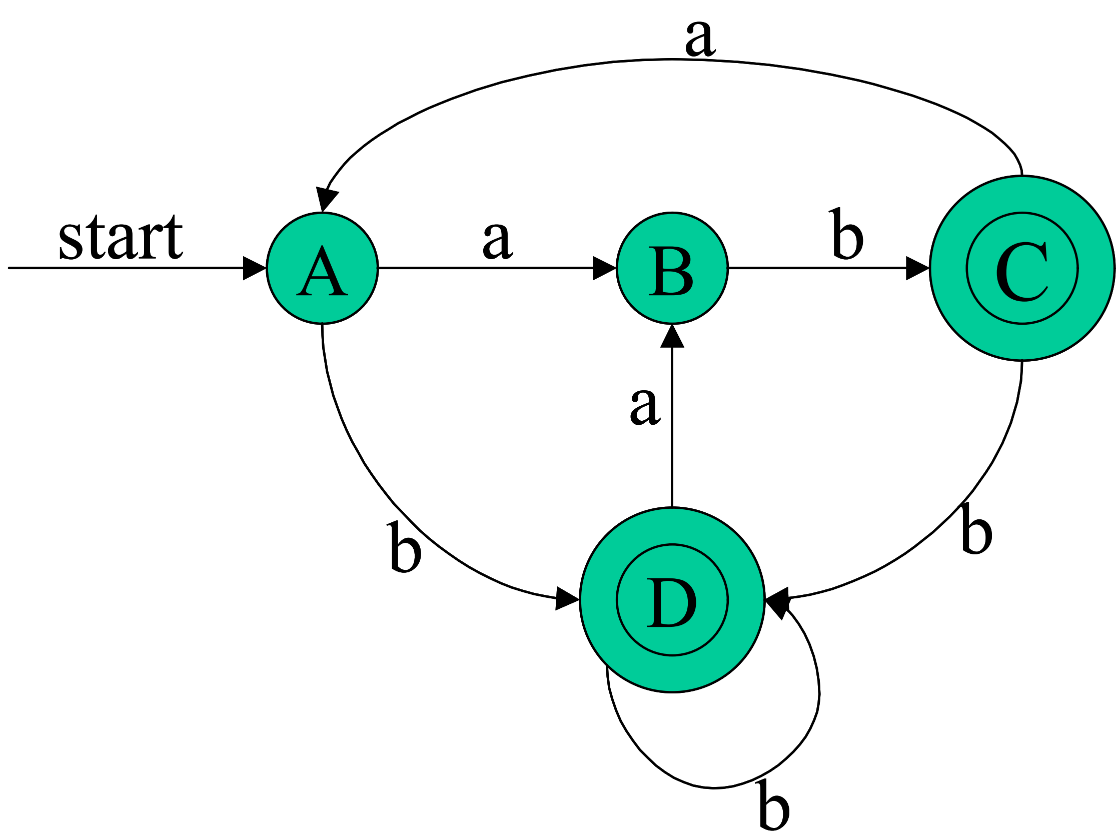
C → bD

C → ε

D → aB

D → bD

D → ε

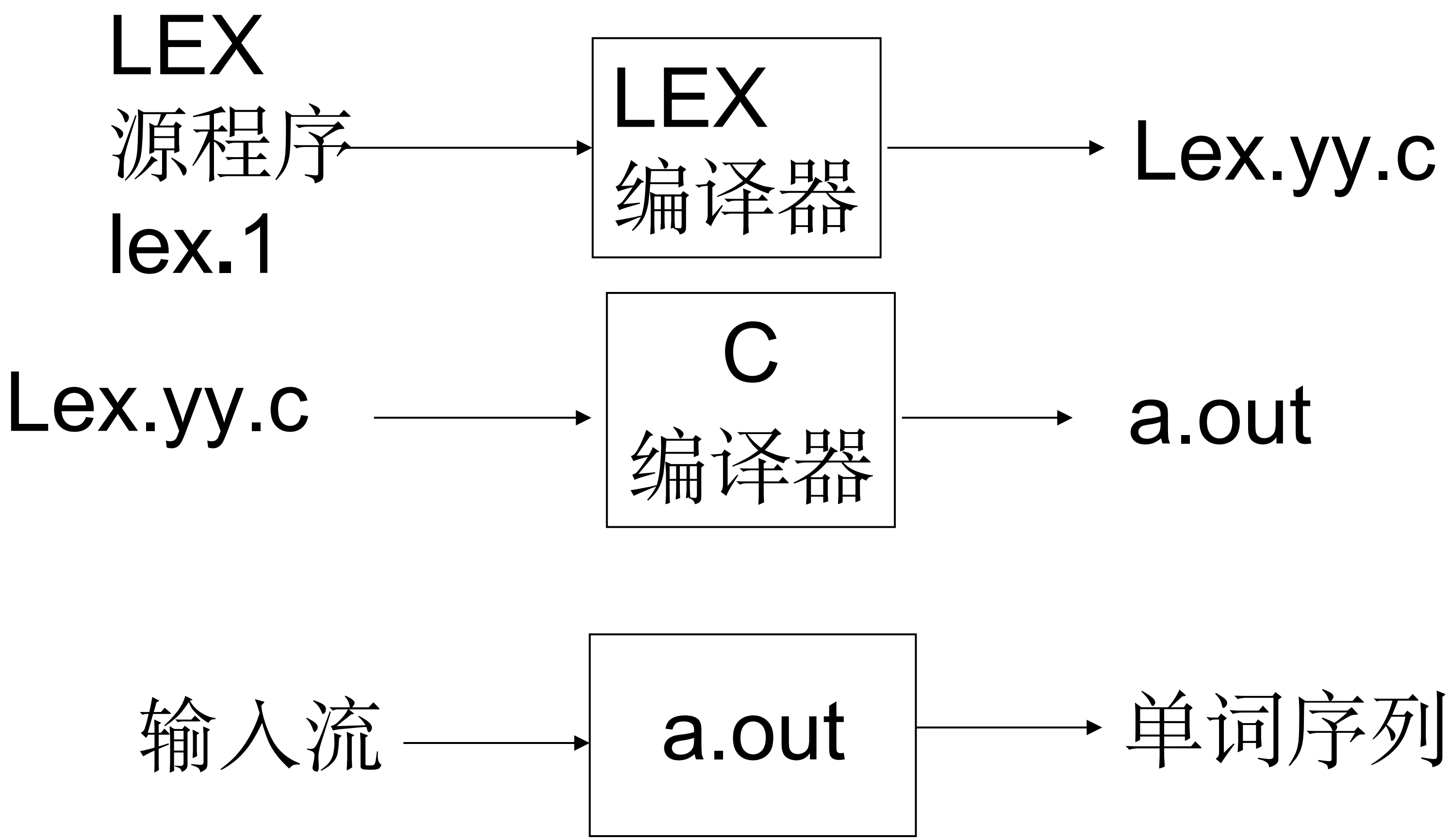


课堂练习

构造正规式1（0|1）* 101相应的DFA。

3.7 词法分析程序的自动构造

- **词法分析程序的自动构造工具LEX简介**



课堂练习

构造如下语言的上下文无关文法：

（1） $\{a^n b^{2n} c^m \mid n，m \geq 0\}$

（2） $\{a^n b^{n+m} c^m \mid n，m \geq 0\}$

给出下列每个正规语言的一个正规表达式

1. $\{ a^n b^m \mid n , m \geq 0$ 且 $n + m$ 为偶数 $\}$
2. $\{ w \in \{a, b\}^* \mid w$ 中既不包含子串 aa ，也不包含子串 bb $\}$