

有限自动机

确定的有限自动机 (DFA)

(Deterministic Finite Automata)

- 一. DFA的定义
- 二. DFA的三种表示
- 三. DFA接受的语言

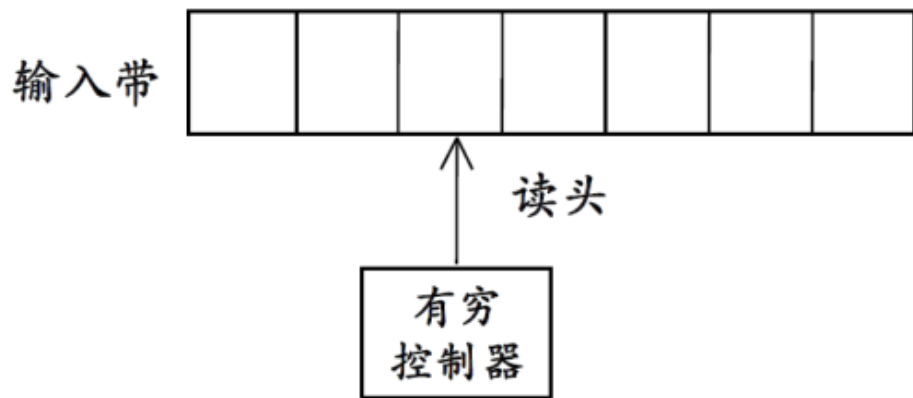
有限自动机

- 有限自动机(*Finite Automata*, *FA*)由两位神经物理学家 *McCulloch*和*Pitts*于1948年首先提出，是对一类处理系统建立的数学模型
- 这类系统具有一系列离散的输入输出信息和有穷数目的内部状态（状态：概括了对过去输入信息处理的状况）
- 系统只需要根据当前所处的状态和当前面临的输入信息就可以决定系统的后继行为。每当系统处理了当前的输入后，系统的内部状态也将发生改变

*FA*的典型例子

- 电梯控制装置
 - 输入：顾客的乘梯需求（所要到达的层号）
 - 状态：电梯所处的层数+运动方向
 - 电梯控制装置并不需要记住先前全部的服务要求，只需要知道电梯当前所处的状态以及还没有满足的所有服务请求

FA的模型



- **输入带**(*input tape*): 用来存放输入符号串
- **读头**(*head*): 从左向右逐个读取输入符号, 不能修改 (只读)、不能往返移动
- **有穷控制器**(*finite control*): 具有有穷个状态数, 根据当前的状态和当前输入符号控制转入下一状态

一. DFA M定义

一个**确定**的**有限**自动机 DFA M是一个五元组
 $M = (\Sigma, S, S_0, Z, f)$

Σ 是一个字母表，它的每个元素称为一个输入符号。

S 是一个有限状态集合。

$S_0 \in S$, S_0 称为初始状态。

Z 是 S 的子集，称为终结状态集合。

f 是一个从 $S \times \Sigma$ 到 S 的单值映射

$$f(q, a) = q' \quad (q, q' \in S, a \in \Sigma)$$

表示当前状态为 q ,输入符号为 a 时，自动机将转换到下一个状态 q' ， q' 称为 q 的后继。

例 设 DFA $M = (\{a, b\}, \{0, 1, 2, 3\}, 0, \{3\}, f)$ 其中

$$f(0, a) = 1, f(1, a) = 3$$

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

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

$$f(2, b) = 3, f(3, b) = 3$$

二 DFA 的三种表示:

- (1) 用转换函数
- (2) 转移矩阵
- (3) 状态转换图

(1) 用转换函数

$$f(0, a) = 1, f(1, a) = 3$$

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

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

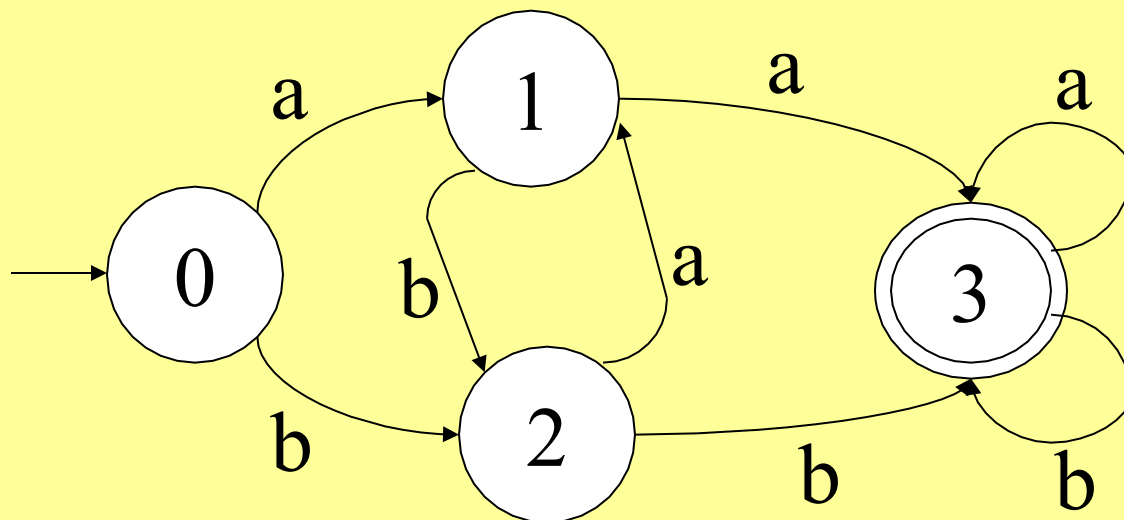
$$f(2, b) = 3, f(3, b) = 3$$

所谓确定的状态机，其确定性表现在状态转移函数是单值函数！

(2) 转移矩阵

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

(3) 状态转换图



输入 字符 状态		
	a	b
0	1	2
1	3	2
2	1	3
3	3	3

结点表示状态，箭弧标记为字母表中的字母

终结状态如何表示？

三. DFA M 接受的语言(字符串集)

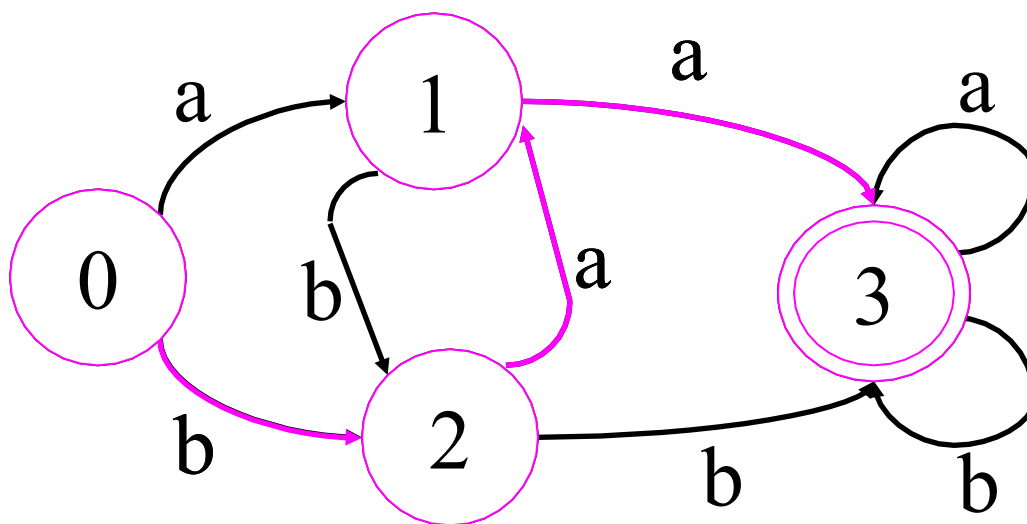
如果对所有 $w \in \Sigma^*$, 以下述方式递归地扩充 f 的定义

$$f(q, \varepsilon) = q$$

$$f(q, wa) = f(f(q, w), a)$$

对于上例中的DFA M 和 $w=baa$,

$$f(0, baa) = f(2, aa) = f(1, a) = 3$$



该DFA M 能够识别字符串baa

从状态转换图看，从初态出发，沿任一条路径到达终结状态，这条路径上的弧上的标记符号连接起来构成的符号串为DFA M 所识别。

DFA M 所能识别的符号串的全体记为 $L(M)$ ，称为DFA M 所识别的语言。

$$L(M) = \{ w \mid w \in \Sigma^*, \text{ 若存在 } q \in Z, \\ \text{使 } f(q_0, w) = q \}$$

非确定的有限自动机 (NFA)

Nondeterministic Finite Automata

- 一. NFA m 的定义
- 二. FA 的等价定理
- 三. 具有 ϵ -转移的NFA构造DFA的算法

一 NFA的形式定义:

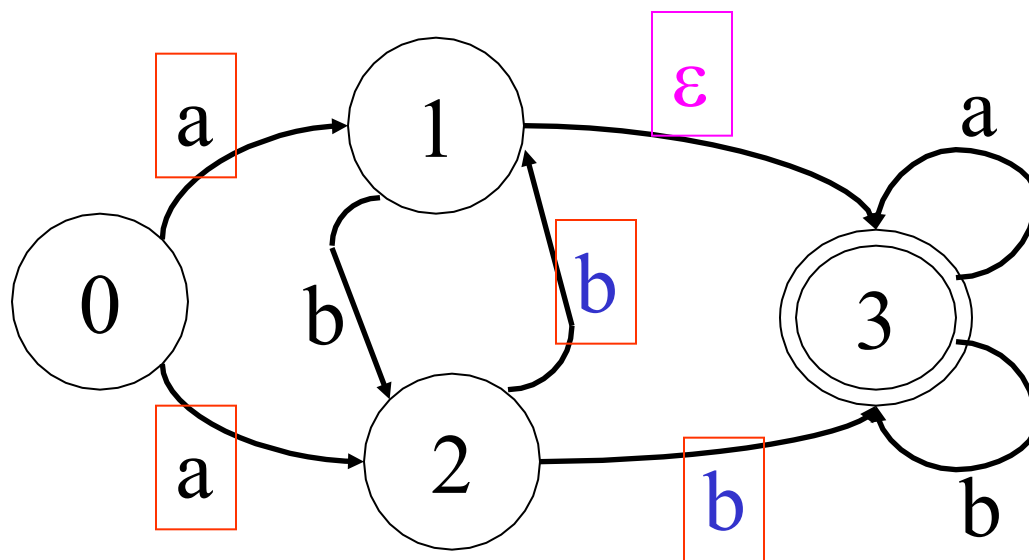
非确定有限自动机M是一个五元组

$$M = (\Sigma, S, S_0, Z, f)$$

其中 Σ , S , Z 的意义和DFA的定义一样, 其中 S_0 表示初始状态集, f 是一个从 $S \times (\Sigma \cup \{\epsilon\})$ 到 S 的子集的映射, 即 $f: S \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^S$, 其中 2^S 是 S 的幂集, 即 **S 中所有子集组成的集合**。

确定的和非确定的有限自动机之间的重要区别是

- 1、状态转换函数是一个多值映射; 反映在状态转换图上即对同一弧标记到达的状态结点不惟一。
- 2、NFA初态集, 而DFA是一个唯一的状态. NFA存在 ϵ 弧标记



类似 DFA, NFA m 可用状态转换图表示,
如果 $f(q, a) = \{q_1, q_2 \dots, q_k\}$, 则从 q 出发
分别向 $q_1, q_2 \dots, q_k$ 各画出一条标记为 a 的箭
弧(非确定的含义)。

同理可定义 NFA m 所识别 (接受) 的语言。
 Σ^* 中所有可能被 NFA m 所识别的符号串的集合记
为 $L(M)$ 。

NFA M' 所识别的语言为:

$$L(M') = \{\alpha | f(q_0, \alpha) = q \quad q \in Z, \text{ 且 } \alpha \in \Sigma^*\}$$

二. FA 的等价定理

定理 对任何一个NFA M , 都存在一个
DFA M' , 使 $L(M')=L(M)$

构造方法: 用 M' 的一个状态对应 M 的多个状态,
用这种方法, 能从一个NFA M 构造一个等价的DFA
 M' , 称作子集构造法。

三、具有 ϵ -转移的NFA构造等价DFA的方法

定义 集合I的 ϵ -闭包:

令I是一个状态集的子集, 定义 ϵ -closure (I) 为:

- 1) 若 $s \in I$, 则 $s \in \epsilon$ -closure (I) ;
- 2) 若 $s \in I$, 则从s出发经过任意条 ϵ 弧能够到达的任何状态都属于 ϵ -closure (I) 。

状态集 ϵ -closure (I) 称为I的 ϵ -闭包

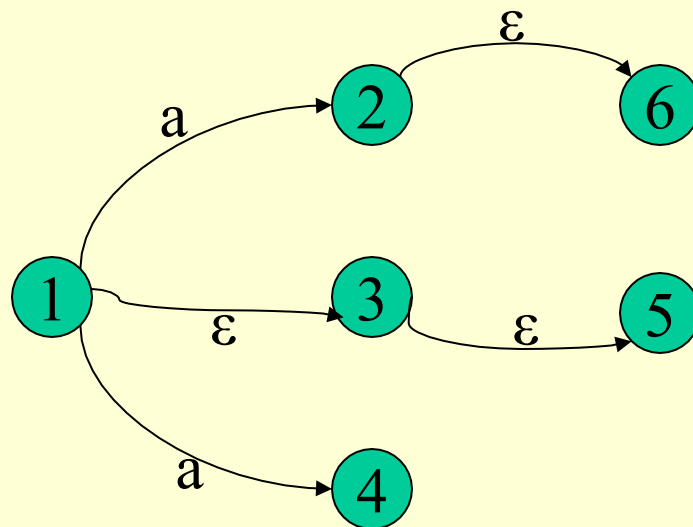
通过例子来说明状态子集的 ϵ -闭包的构造方法

例：

如图所示的状态转换图：

令 $I = \{1\}$ ，

求 ϵ -closure (I) = ?



根据定义：

ϵ -closure (I) = {1, 3, 5}

构造等价DFA算法

- 1) 若 t_1 是NFA的初态, DFA的初态 $A = \varepsilon\text{—closure}(\{t_1\})$ 。
- 2) 对NFA中每一个箭弧标记 m , 计算 $\varepsilon\text{—closure}(f(q,m))$, 其中 q 为已生成的DFA状态。遍历字母表的每个字符为输入

例如字母表为 $\{a,b\}$

$$B = \varepsilon\text{—closure}(f(A,a))$$

$$C = \varepsilon\text{—closure}(f(A,b))$$

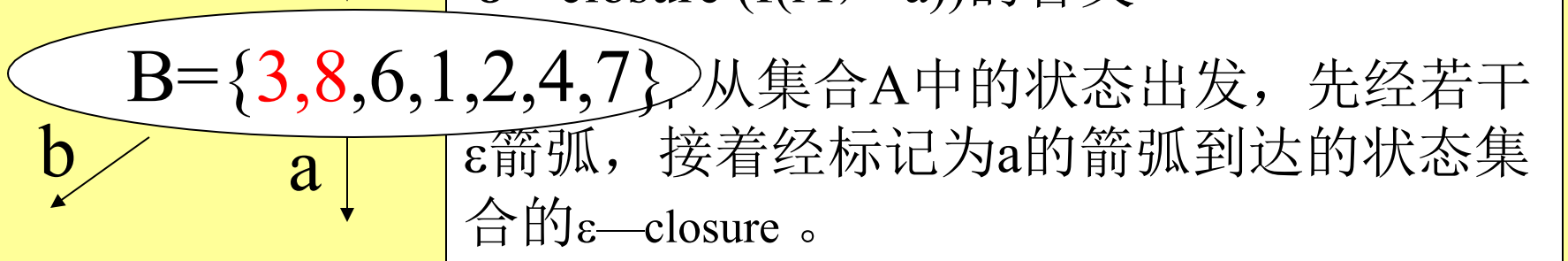
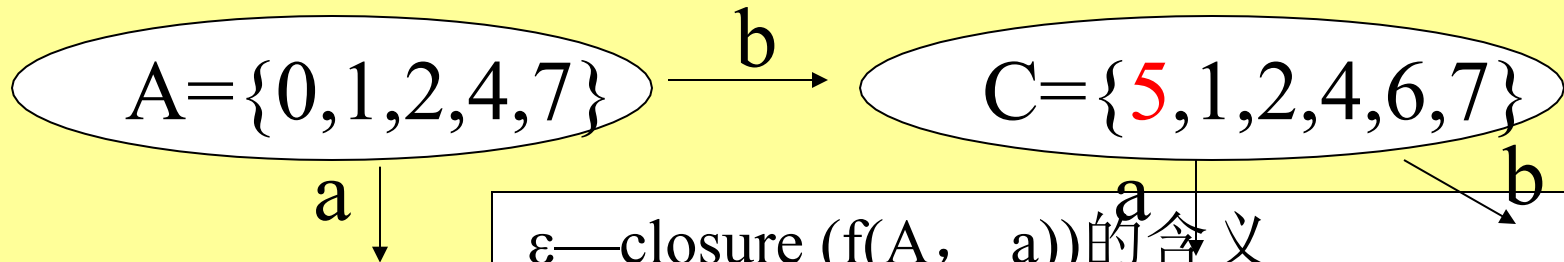
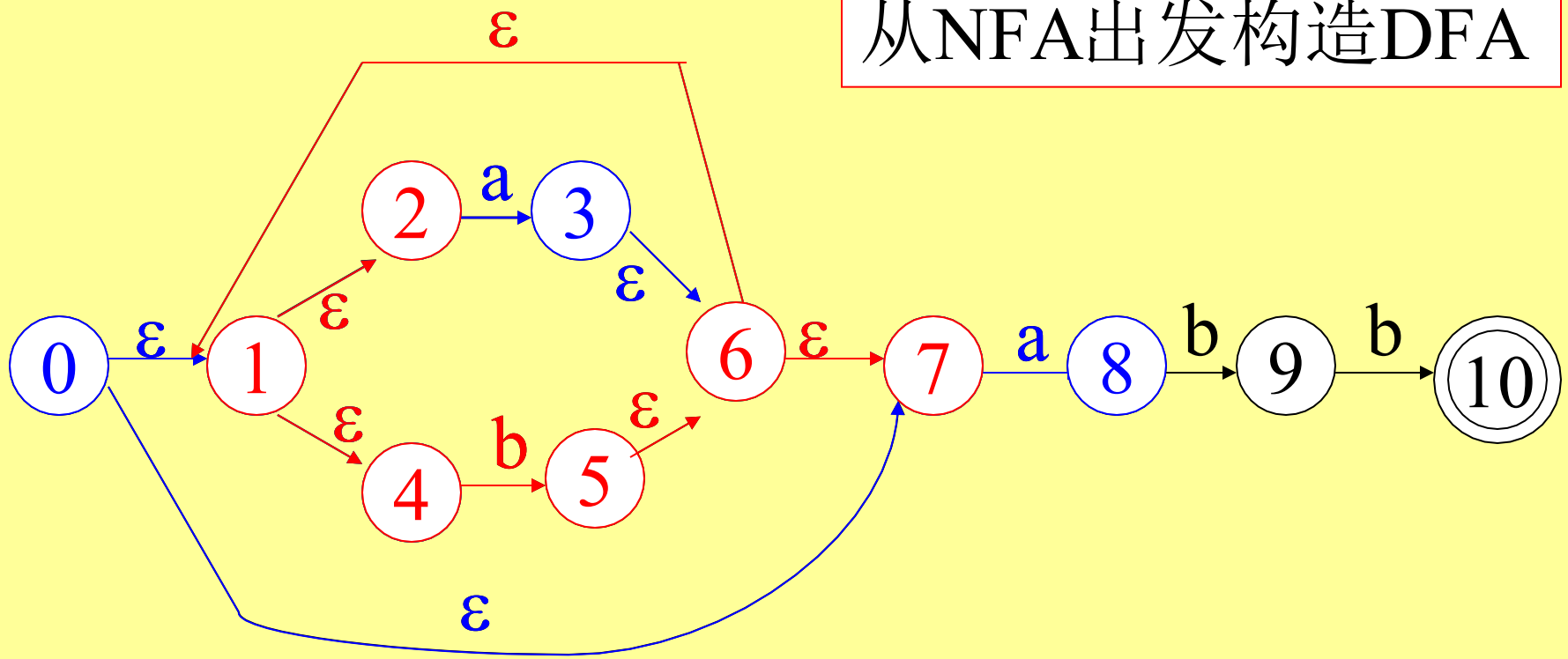
如果 B 和 C 不为空集, 重复这一过程, 直到不在出现新的状态集合)

$$D = \varepsilon\text{—closure}(f(B,a)) \quad E = \varepsilon\text{—closure}(f(B,b))$$

$$F = \varepsilon\text{—closure}(f(C,a)) \quad G = \varepsilon\text{—closure}(f(C,b)) \quad \dots\dots$$

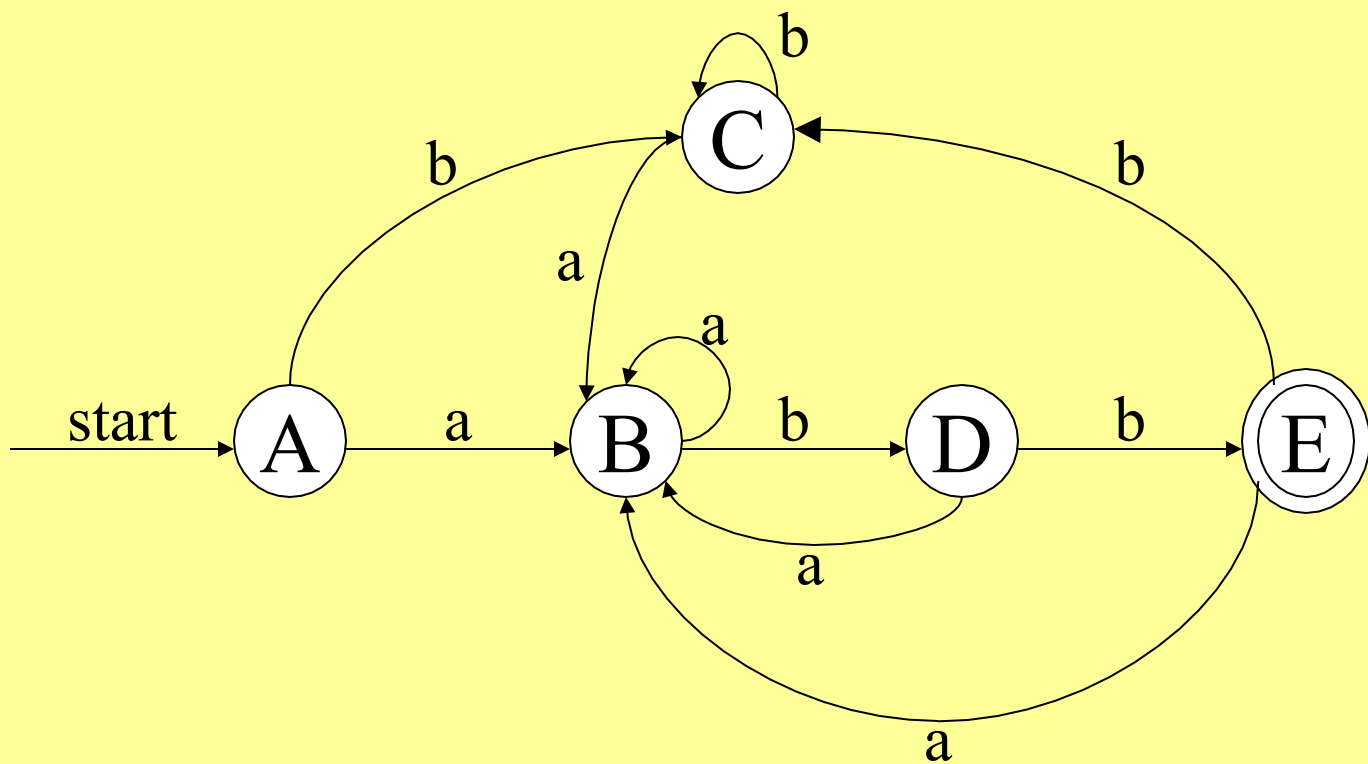
注意: D,E,F,G 中相等的集合合并,空集则舍去.

从NFA出发构造DFA



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

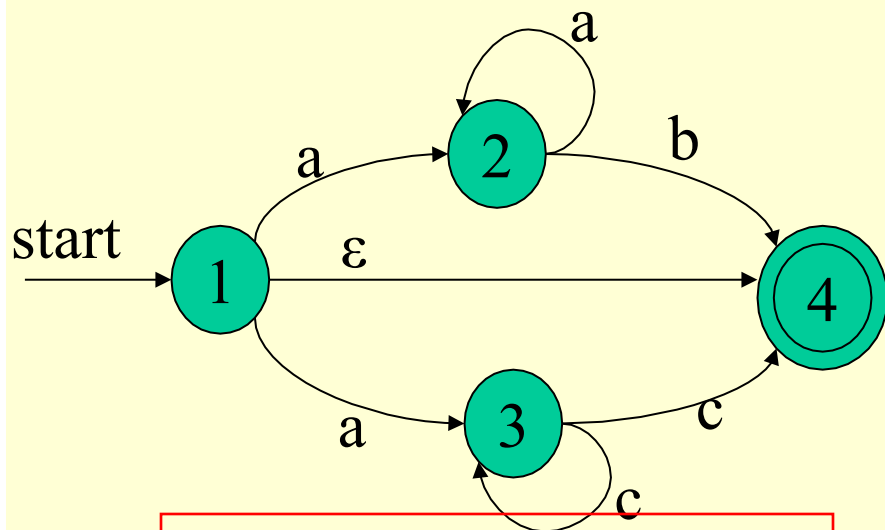
等价DFA的转移矩阵



等价的DFA的状态转换图

☆ 注意：包含原初始状态0的状态子集A为DFA M的初态
包含原终止状态10的状态子集E为DFA M的终态。

例：有NFA M'



$$B = \varepsilon\text{-closure}(f(A, a))$$

$$E = \varepsilon\text{-closure}(f(B, a))$$

$$F = \varepsilon\text{-closure}(f(B, b))$$

$$G = \varepsilon\text{-closure}(f(B, c))$$

.....

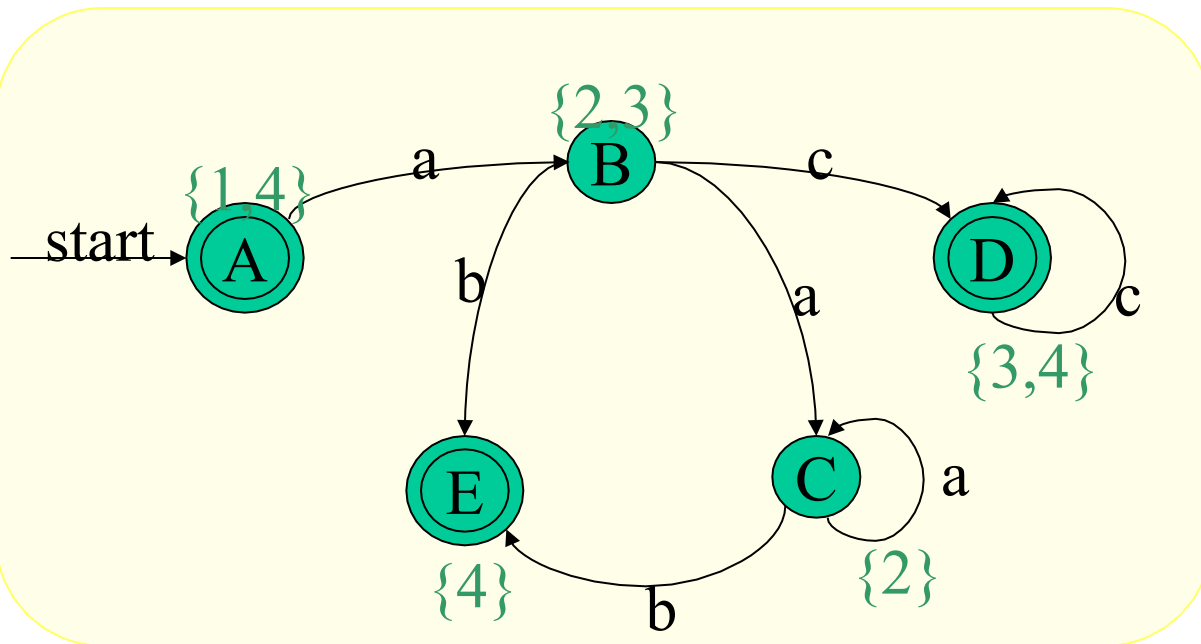
$$A = \varepsilon\text{-closure}(\{1\}) = \{1, 4\}$$

$$\begin{aligned}
 B &= \varepsilon\text{-closure}(f(A, a)) \\
 &= \varepsilon\text{-closure}(f(\{1, 4\}, a)) \\
 &= \varepsilon\text{-closure}(f(1, a) \cup f(4, a)) \\
 &= \varepsilon\text{-closure}(\{2, 3\} \cup \varnothing) \\
 &= \varepsilon\text{-closure}(\{2, 3\}) \\
 &= \{2, 3\}
 \end{aligned}$$

$$\begin{aligned}
 C &= \varepsilon\text{-closure}(f(A, b)) \\
 &= \varepsilon\text{-closure}(f(1, b) \cup f(4, b)) \\
 &= \varepsilon\text{-closure}(\varnothing) \\
 &= \varnothing
 \end{aligned}$$

$$\begin{aligned}
 D &= \varepsilon\text{-closure}(f(A, c)) \\
 &= \varepsilon\text{-closure}(f(1, c) \cup f(4, c)) \\
 &= \varnothing
 \end{aligned}$$

DFA M的状态图:



★ 注意：包含原初始状态1的状态子集为DFA M的初态
包含原终止状态4的状态子集为DFA M的终态。

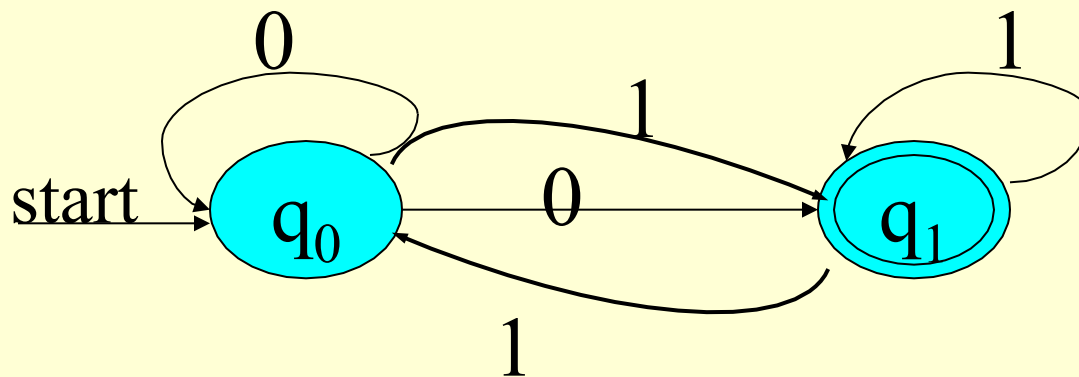
具有 ϵ -转移的NFA构造等价DFA的方法

不具有 ϵ -转移的NFA如何构造等价DFA?

例 NFA $M = (\{0,1\}, \{q_0, q_1\}, q_0, \{q_1\}, f)$, 其中

$$f(q_0, 0) = \{q_0, q_1\}, f(q_0, 1) = \{q_1\}$$

$$f(q_1, 0) = \emptyset \quad f(q_1, 1) = \{q_0, q_1\}$$

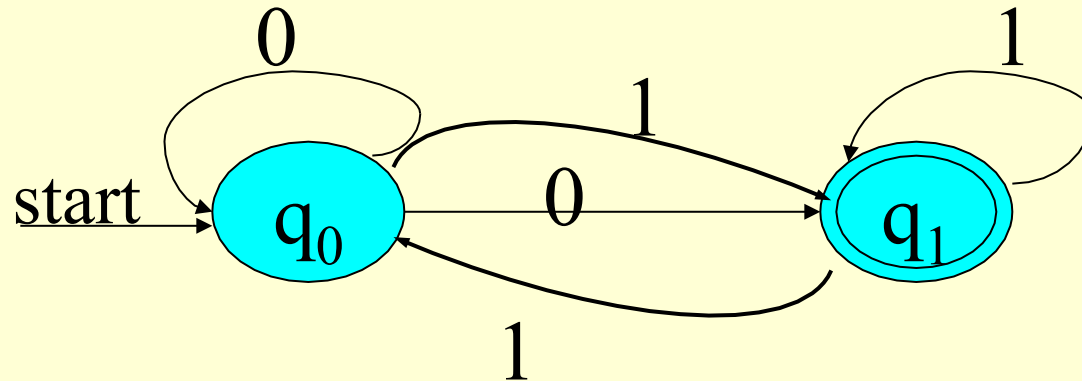


$$f(\{q_0\}, 0) = \{q_0, q_1\}, \quad f(\{q_0\}, 1) = \{q_1\}$$

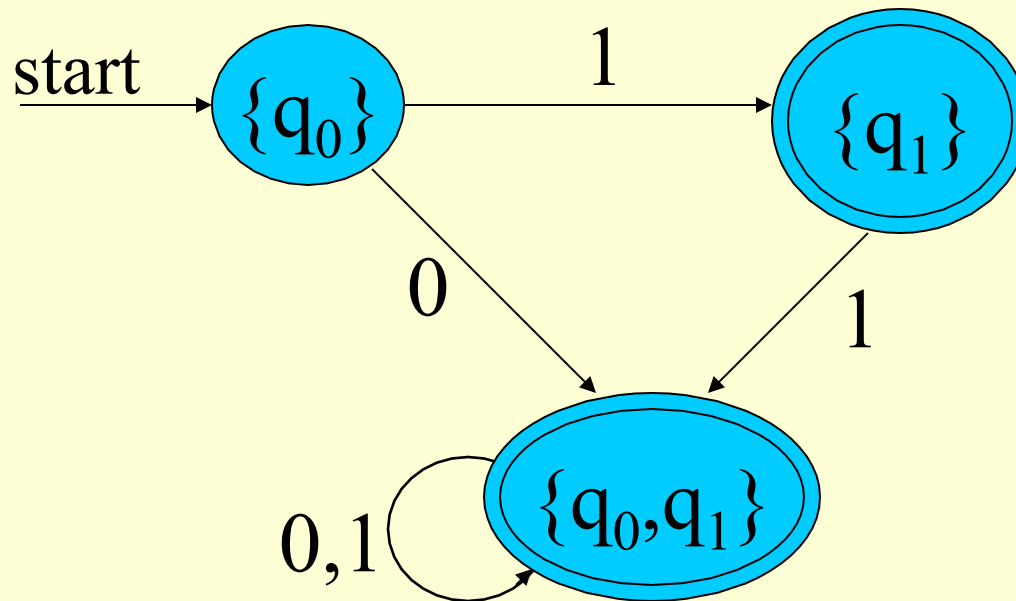
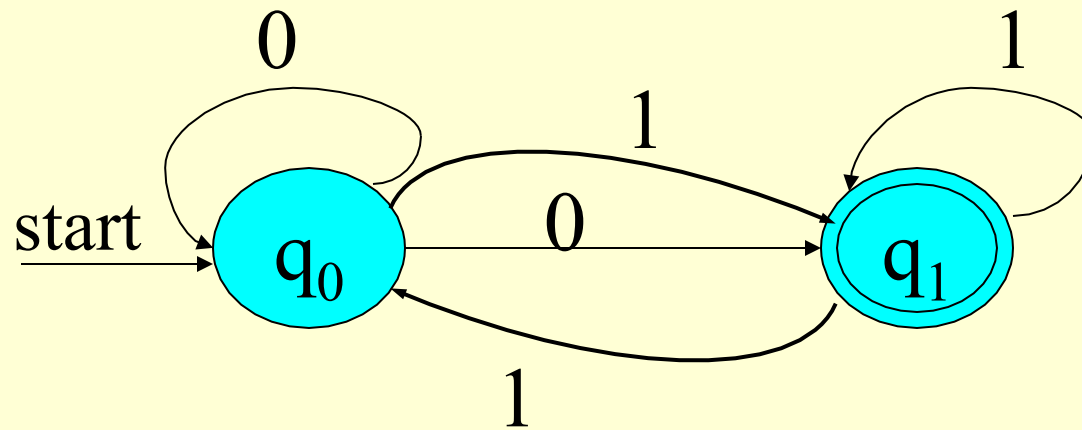
$$f(\{q_1\}, 0) = \emptyset, \quad f(\{q_1\}, 1) = \{q_0, q_1\}$$

$$f(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\}$$

$$f(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0, q_1\}$$



M与M'的状态转换图如下所示:



确定有限自动机的化简

自动机是描述信息处理过程的一种数学模型

对一种语言，它可以用许多文法来描述，同样可以有无限多个FA来描述一种语言；这些FA是等价的，但其构成的复杂程度差别很大

所谓一个DFA $M=(\Sigma, S, S_0, Z, f)$ 的化简是指寻找一个状态数比较少的DFA M' ,使 $L(M)=L(M')$ 。而且可以证明，存在一个最少状态的DFA M' ,使 $L(M)=L(M')$ 。

一个DFA m 是最小化的 \Leftrightarrow 它没有多余状态并且没有互相等价的状态。

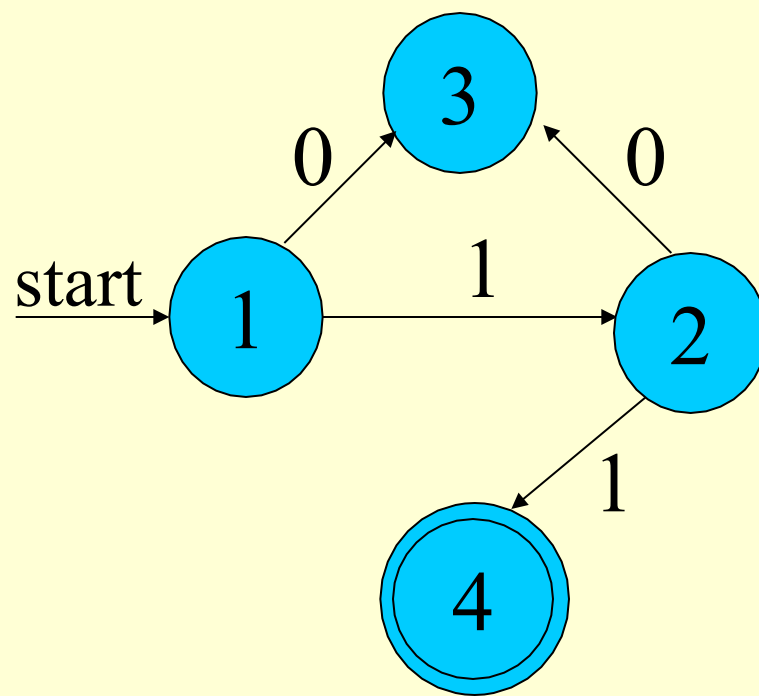
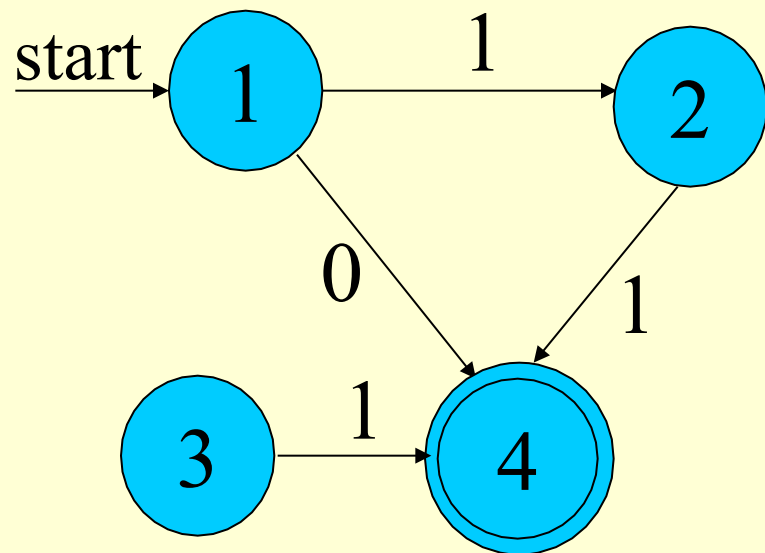
一个DFA m 可以通过消除多余状态和合并等价状态而转换成一个最小的与之等价的DFA m'

一、有限自动机的多余状态（无关状态）

(1) 从该自动机的开始状态出发,任何输入串也不能到达的那个状态

(2) 从该状态出发没有通向终态结的道路

为什么？



这些多余状态不在从初态到终态的路径上，对识别句子无任何作用。

二、等价状态的定义

设 $p, q \in S$, 若对任何 $w \in \Sigma^*$, **$f(p, w)$ 与 $f(q, w)$ 同时到达终止状态或拒绝状态之中** , 则称 p 和 q 是等价的。否则, 称 p 和 q 不等价 (可区别) 。

判定两个状态 p 和 q 不等价, 只要找到一个 $w \in \Sigma^*$, 使 $f(p, w) \in Z$ 且 $f(q, w) \notin Z$, 或者相反。

说明

(a) 终结状态与非终结状态不等价。

(b) 对于 $\forall a \in \Sigma$, $f(p, a) = r$, $f(q, a) = s$,
 r 与 s 均等价, 则 p 与 q 等价;

若存在某个 $a \in \Sigma$, $f(p, a) = r$, $f(q, a) = s$
其中 r 与 s 不等价, 则 p 与 q 不等价。

r 与 s 不等价, 存在 $w \in \Sigma^*$

$f(r, w) \notin Z$ 且 $f(s, w) \in Z$

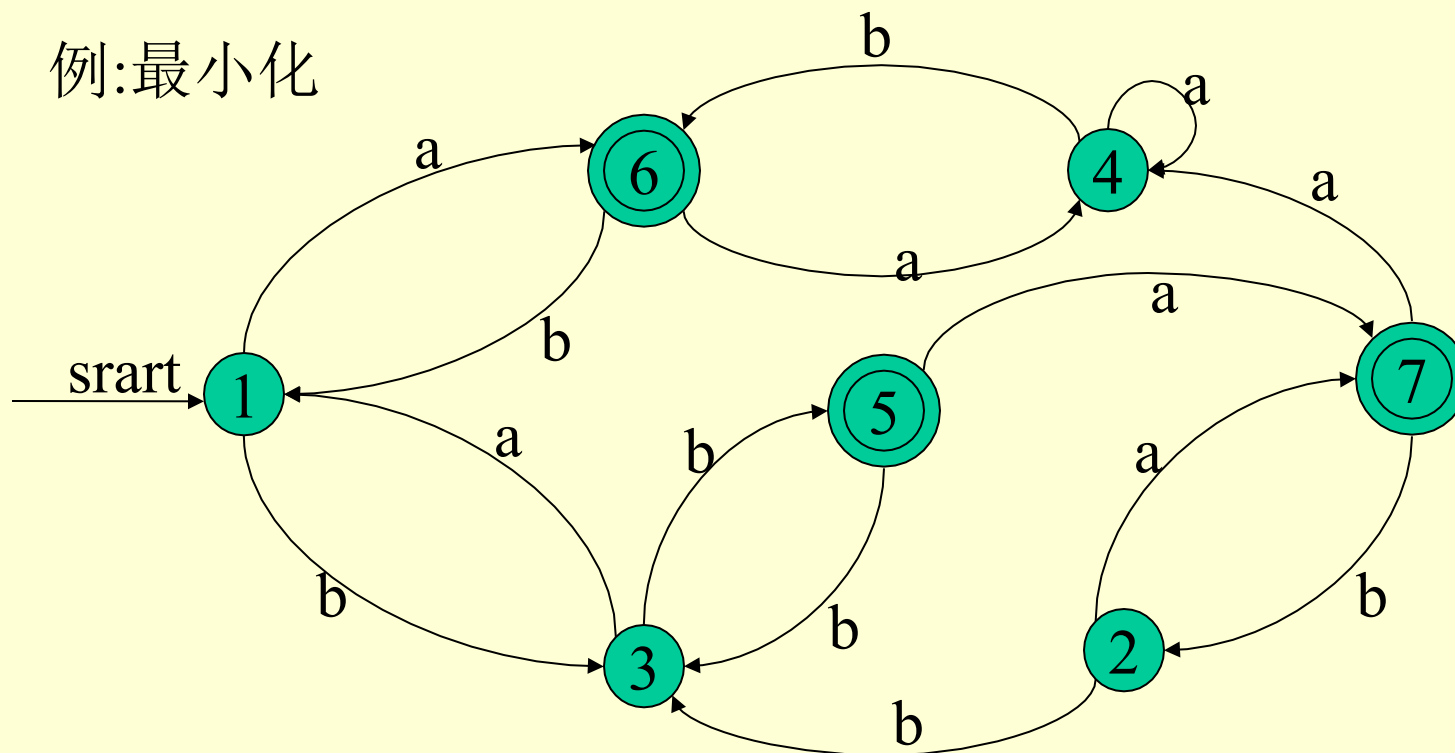
$f(p, aw) \notin Z$ 且 $f(q, aw) \in Z$

一个DFA m 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DFA m'

分割法： 把一个DFA(不含多余状态)的状态分割成一些不相关的子集，使得任何不同的两个子集状态都是可区别的，而同一个子集中的任何状态都是等价的.在各个子集中任取一个状态做代表，删去子集的其余状态。

分割法（划分法）具体实现：

例:最小化



有没有多余状态？

终结状态与非终结状态不等价

	a	b	子集号
1	6	3	
2	7	3	
3	1	5	
4	4	6	
5	7	3	
6	4	1	
7	4	2	

整理

	a	b	子集号
1	6	3	1
2	7	3	
3	1	5	
4	4	6	
5	7	3	2
6	4	1	
7	4	2	



	a	b	子集号
1	6	3	1
2	7	3	
3	1	5	2
4	4	6	
5	7	3	3
6	4	1	
7	4	2	



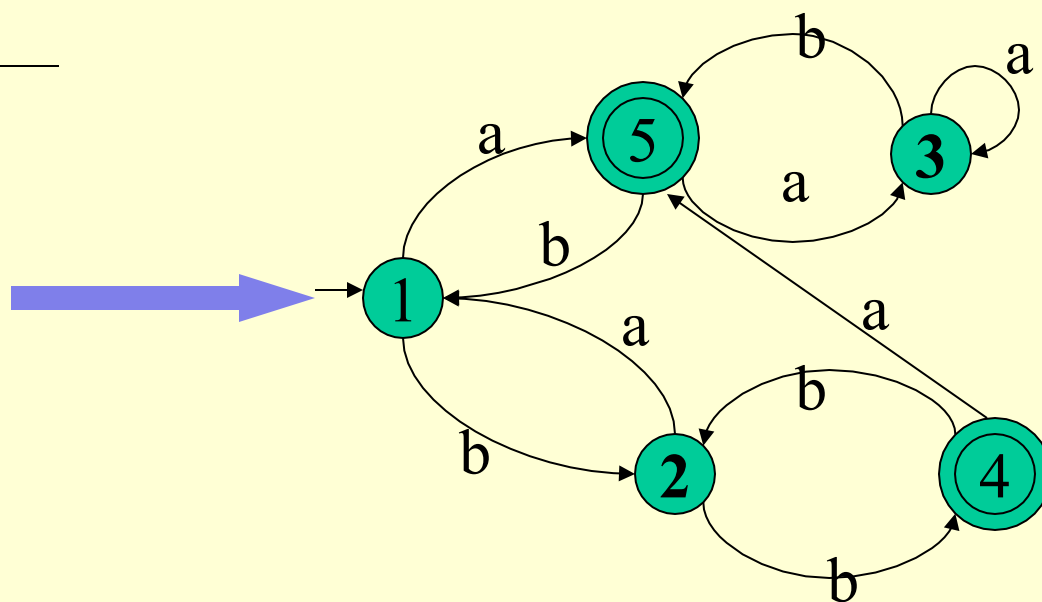
	a	b	子集号
1	6	3	1
2	7	3	
3	1	5	2
4	4	6	
5	7	3	3
6	4	1	
7	4	2	4



	a	b	子集号
1	6	3	1
2	7	3	
3	1	5	2
4	4	6	
5	7	3	4
6	4	1	
7	4	2	5

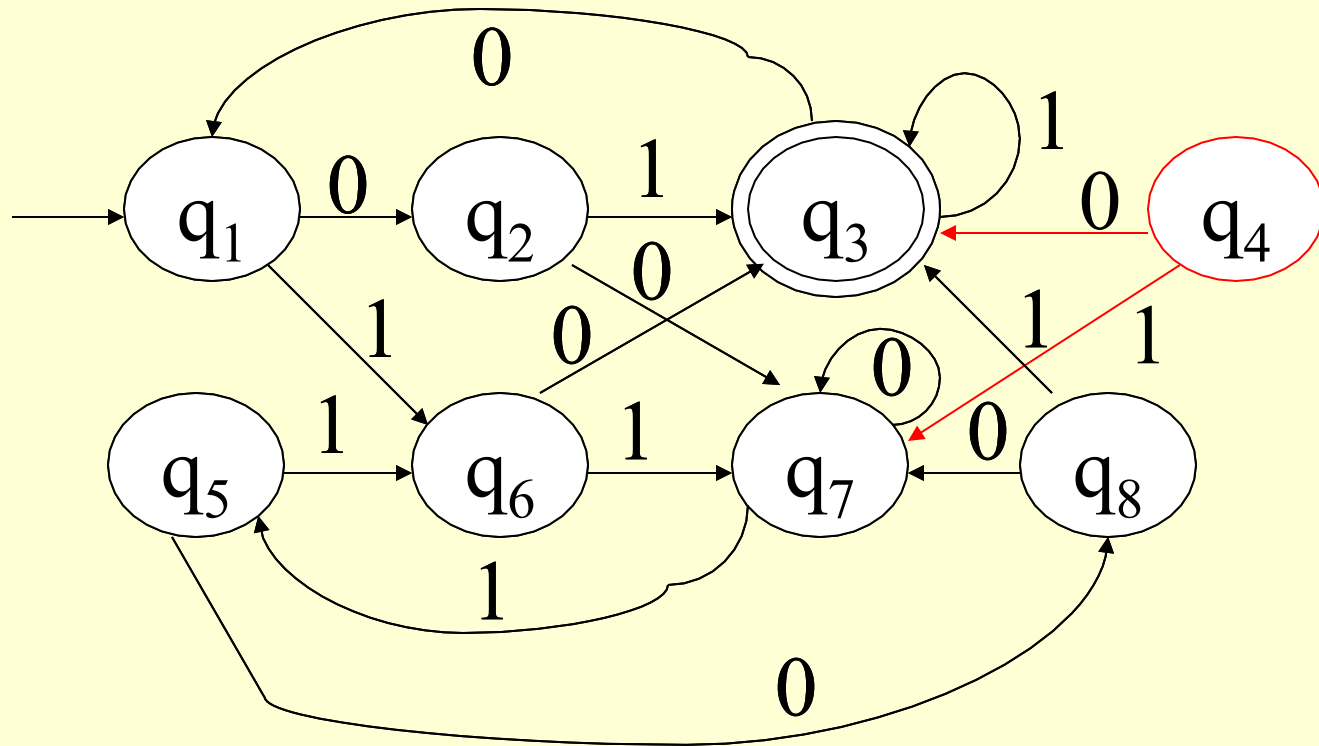
用子集号代替状态号得:

	a	b
1	5	2
2	1	4
3	3	5
4	5	2
5	3	1





化简以下DFA



区分终态与非终态


	0	1	子集号
1	2	6	1
2	7	3	
5	8	6	
6	3	7	
7	7	5	
8	7	3	
3	1	3	2

整理

	0	1	子集号
1	2	6	1
5	8	6	
6	3	7	
7	7	5	
2	7	3	
8	7	3	
3	1	3	2



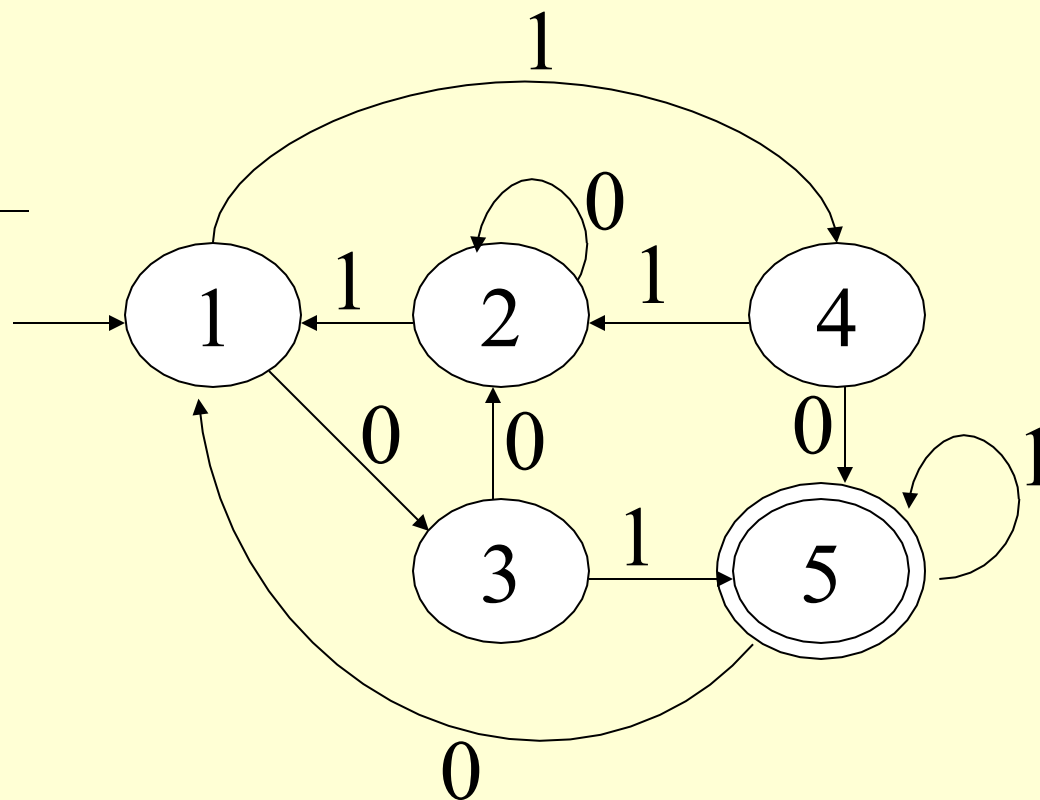
	0	1	子集号
1	2	6	1
5	8	6	
7	7	5	
2	7	3	
8	7	3	
6	3	7	2
3	1	3	3



	0	1	子集号
1	2	6	1
5	8	6	
7	7	5	2
2	7	3	
8	7	3	3
6	3	7	4
3	1	3	5

用子集号代替状态号得:

	0	1
1	3	4
2	2	1
3	2	5
4	5	2
5	1	5



化简过程



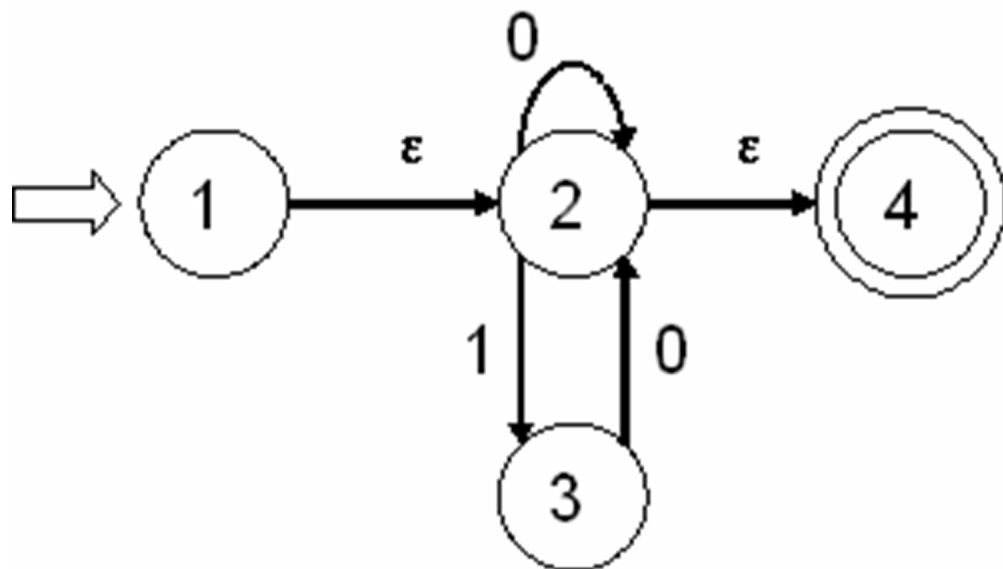
回顾复习

构造一个**DFA**，它接受 $\Sigma=\{0,1\}$
上所有满足如下条件的字符串：
每个**1**都有**0**直接跟在右边。

参考答案

- 解答: $(0|10)^*$

– NFA如下:

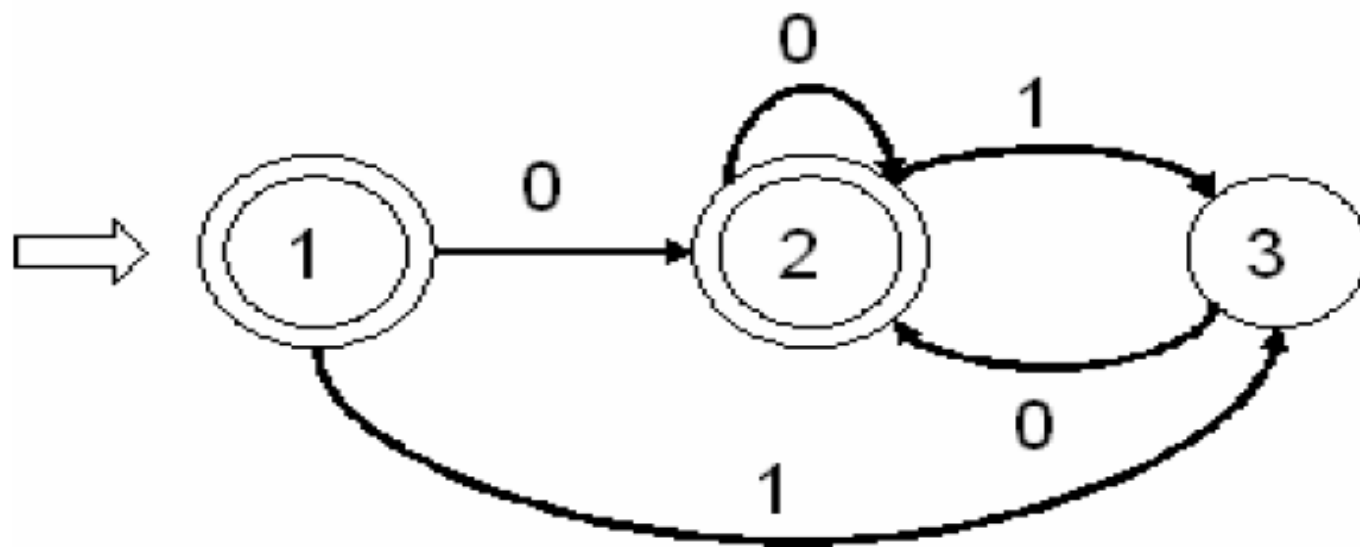


- NFA确定化:

	1	2
{1,2,4}	{2,4}	{3}
{2,4}	{2,4}	{3}
{3}	{2,4}	-

	0	1
1	2	3
2	2	3
3	2	-

DFA 如下：



- DFA化简？

- DFA

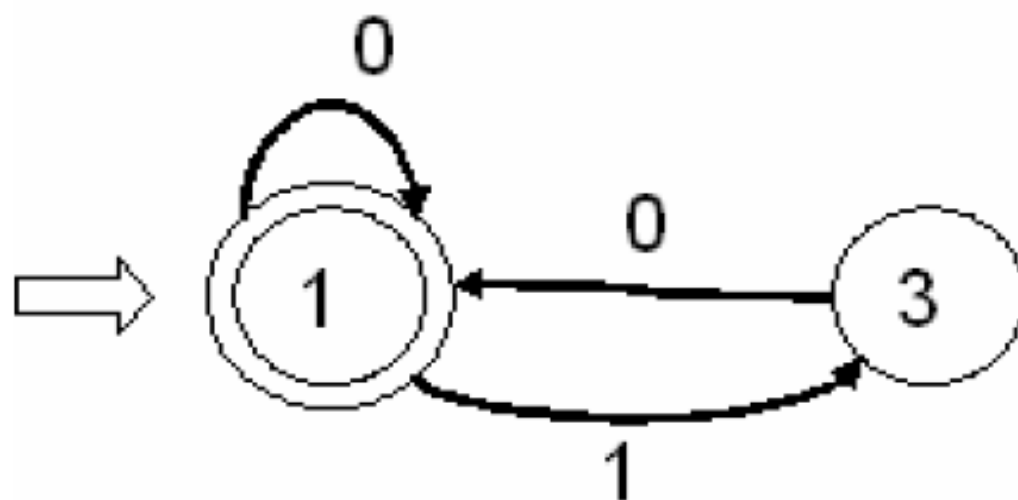
化简:

(1) 基本分划: $\{1,2\}, \{3\}$

(2) $\{1,2\}_0 = \{2,2\} \subset \{1,2\}$

$\{1,2\}_1 = \{3,3\} \subset \{3\}$ 无需再分

(3) 最后分划为: $\{1,2\}, \{3\}$



总结

- 语言

- 自然语言

- 英语、汉语等
 - 语言的产生过程：先有语言，再有语法结构
 - 自然语言的识别，为什么会成为人工智能领域的一个重大的问题

- 形式语言

- C语言、C++、Java等
 - 语言的产生过程：先有正式的语法结构，之后才由这些语法结构来产生语言。

总结

- 集合

- 如何表达一个集合？一般而言，有这么两种方法：一个是列举法，一个是描述法。
- 对于个数有限的集合，可以用列举法一一列出；而对于无穷的集合，则多用描述法来表达。

总结

- 把高级语言看成一个集合，那么每个合法的高级语言源代码就是这个集合里的一个元素，合法的高级语言源代码有无穷多个，那我们要用什么样的描述方法来表达这个无穷集合？

文法（关于如何生成语法的法则）

总结

- $G: \langle V_T, V_N, S, P \rangle$
- $L(G)$: 句型、句子、句子的集合
- 推导
 - 最左推导、最右推导、直接推导、多步推导
- 语法树
- 二义性文法
- 文法分类

总结

- 有限自动机 $\langle \Sigma, S, S_0, Z, f \rangle$
 - DFA
 - 单值映射
 - NFA
 - 多值映射
 - NFA确定化为DFA
 - 构造子集法：集合I的 ϵ -闭包
 - DFA化简
 - 划分法：消除多余状态和等价状态