

第四讲

✧ 有限状态自动机

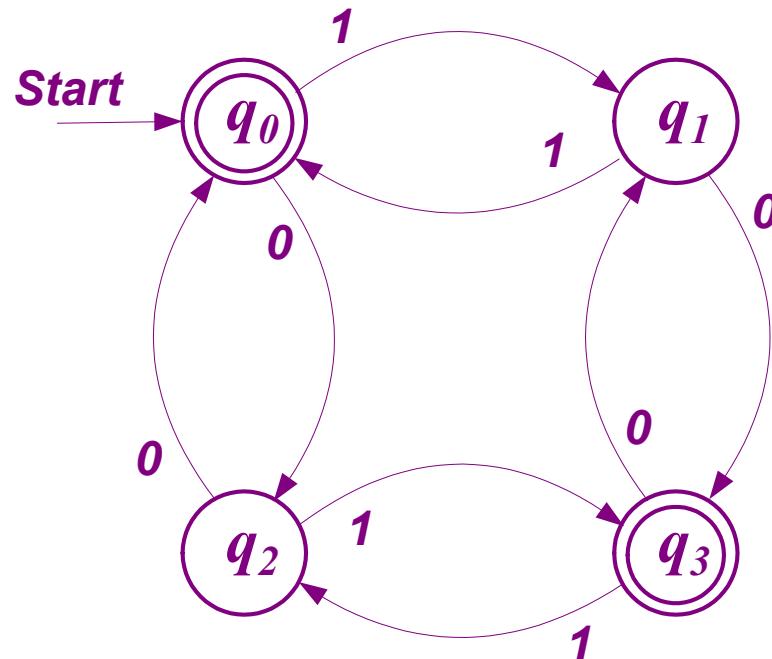
有限状态自动机

- ✧ 确定有限自动机
- ✧ 非确定有限自动机
- ✧ 确定与非确定有限自动机的等价性
- ✧ 有限自动机的一个应用 — 文本搜索
- ✧ 带 ε -转移的非确定有限自动机
- ✧ (确定) 有限自动机的最小化

确定有限自动机

✧ 有限自动机的五要素

- 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

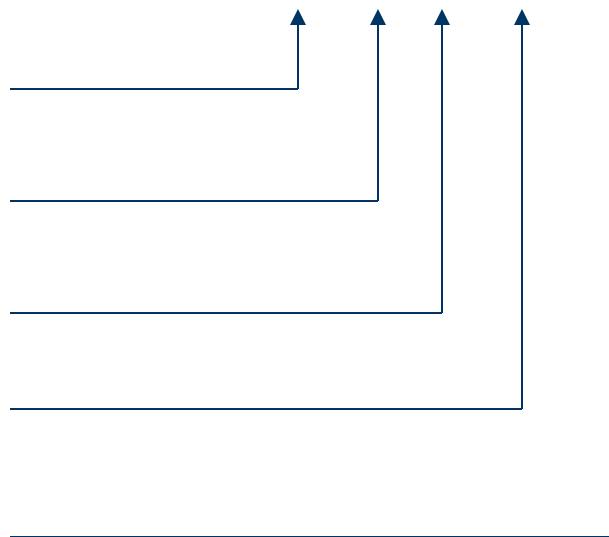


确定有限自动机

◆ 确定有限自动机的形式定义

一个确定有限状态自动机 **DFA** (*deterministic finite automata*) 是一个五元组 $A = (Q, \Sigma, \delta, q_0, F)$.

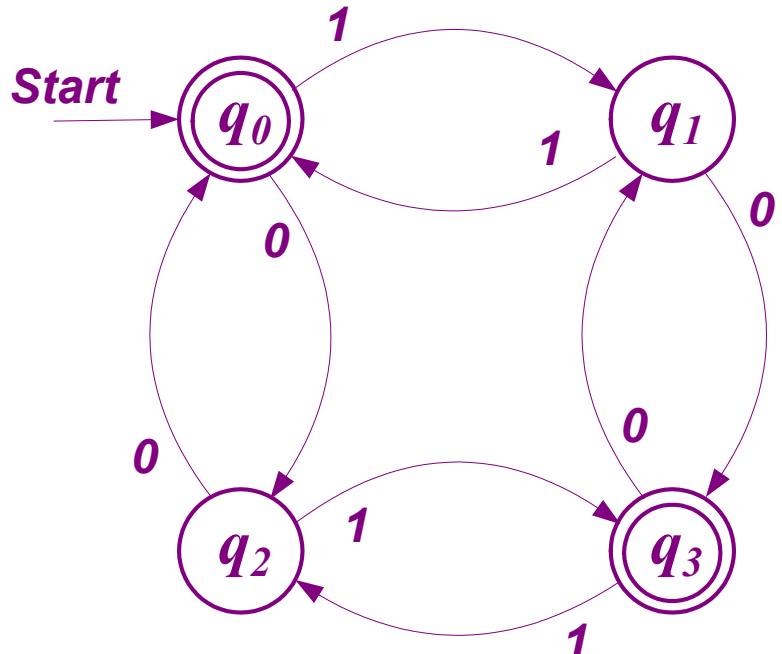
- 有限状态集
- 有限输入符号集
- 转移函数
- 一个开始状态
- 一个终态集合



$$\begin{aligned}\delta &: Q \times \Sigma \rightarrow Q \\ q_0 &\in Q \\ F &\subseteq Q\end{aligned}$$

确定有限自动机

◆ 转移图表示的 DFA



- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- $\delta(q_0, 0) = q_2, \delta(q_0, 1) = q_1$
 $\delta(q_1, 0) = q_3, \delta(q_1, 1) = q_0$
 $\delta(q_2, 0) = q_0, \delta(q_2, 1) = q_3$
 $\delta(q_3, 0) = q_1, \delta(q_3, 1) = q_2$
- q_0
- $F = \{q_0, q_3\}$

确定有限自动机

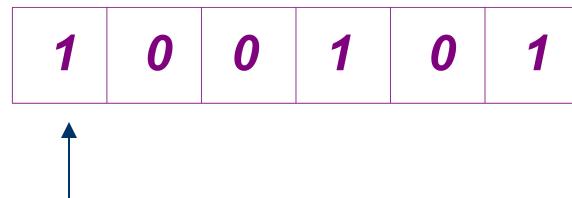
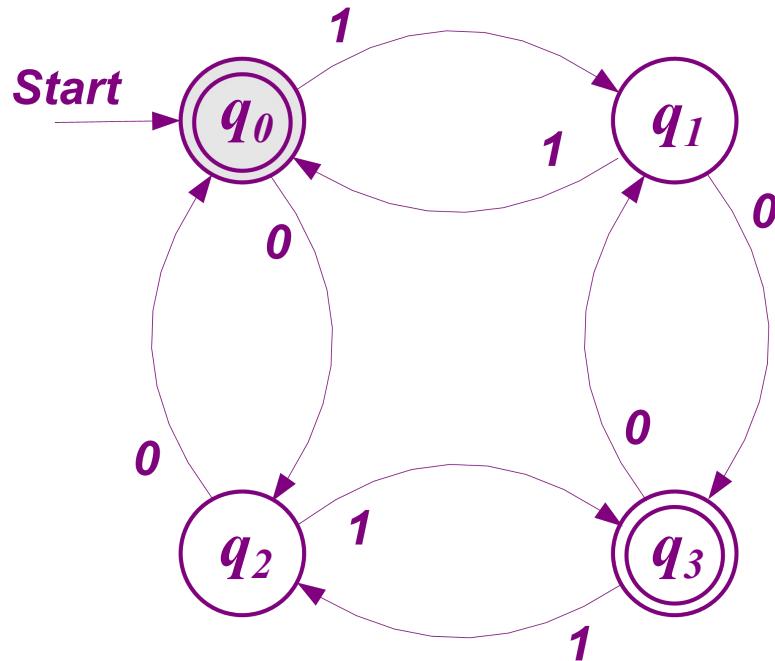
◆ 转移表表示的 DFA

	0	1
→ * q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
* q_3	q_1	q_2

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- $\delta(q_0, 0) = q_2, \delta(q_0, 1) = q_1$
- $\delta(q_1, 0) = q_3, \delta(q_1, 1) = q_0$
- $\delta(q_2, 0) = q_0, \delta(q_2, 1) = q_3$
- $\delta(q_3, 0) = q_1, \delta(q_3, 1) = q_2$
- q_0
- $F = \{q_0, q_3\}$

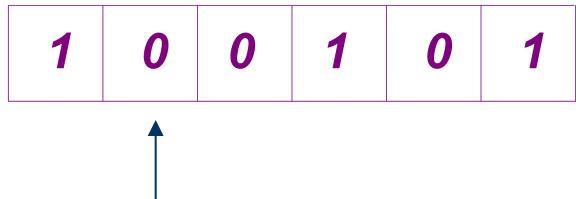
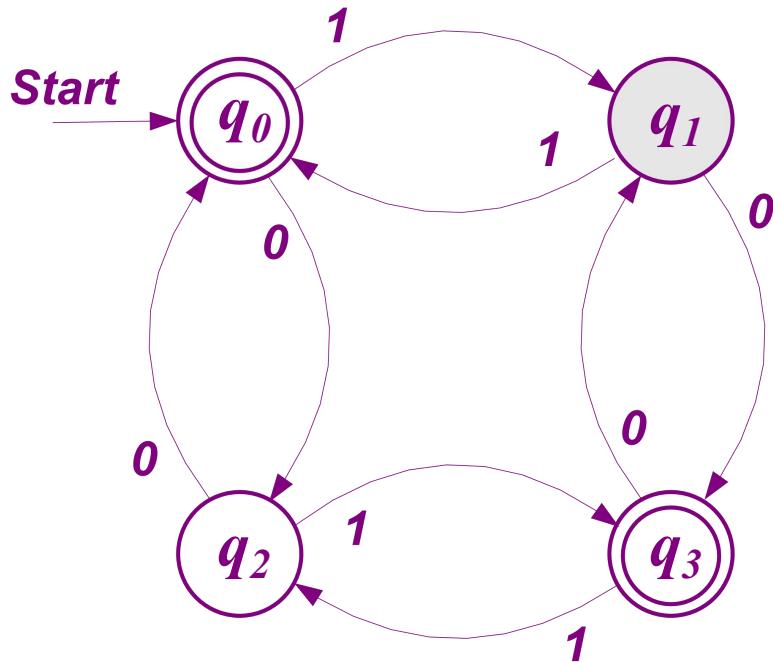
确定有限自动机

◆ DFA如何接受输入符号串



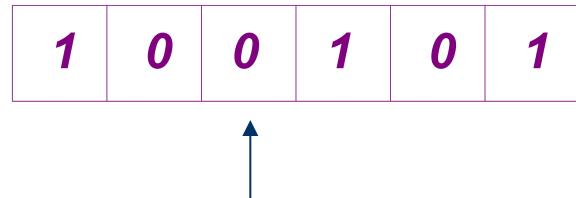
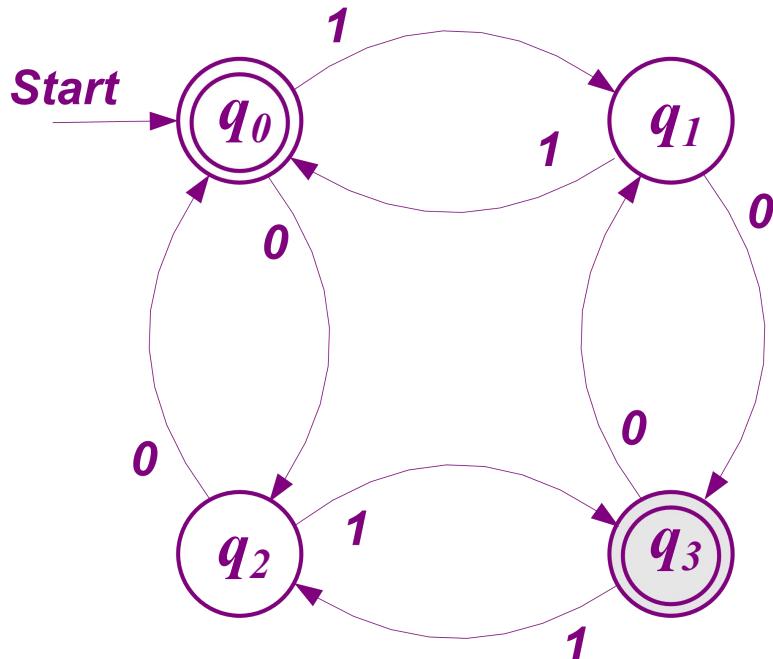
确定有限自动机

◆ DFA如何接受输入符号串



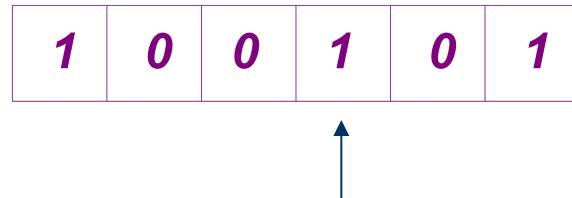
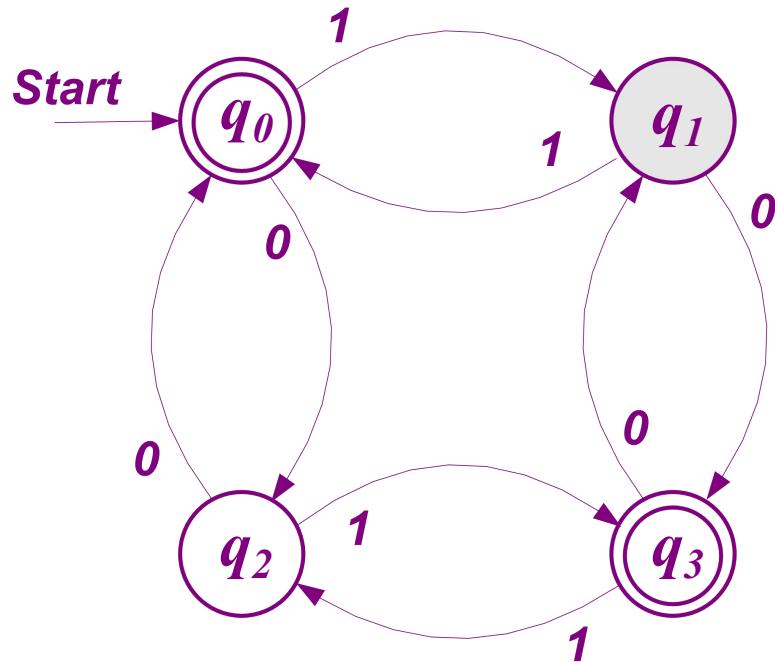
确定有限自动机

◆ DFA如何接受输入符号串



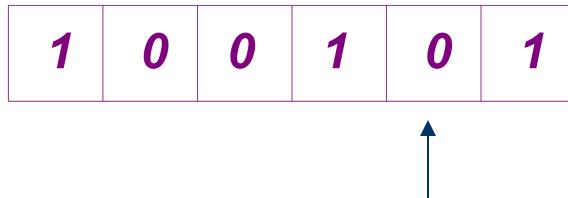
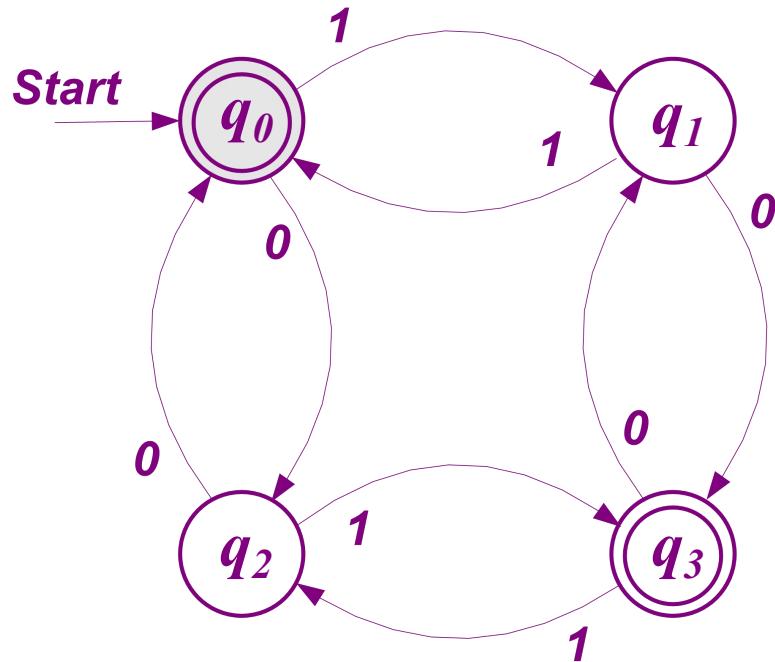
确定有限自动机

◆ DFA如何接受输入符号串



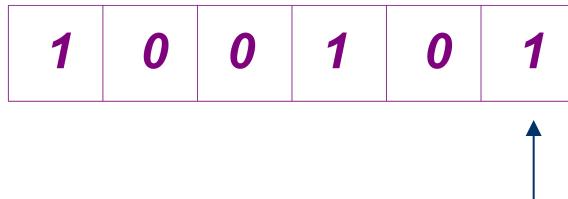
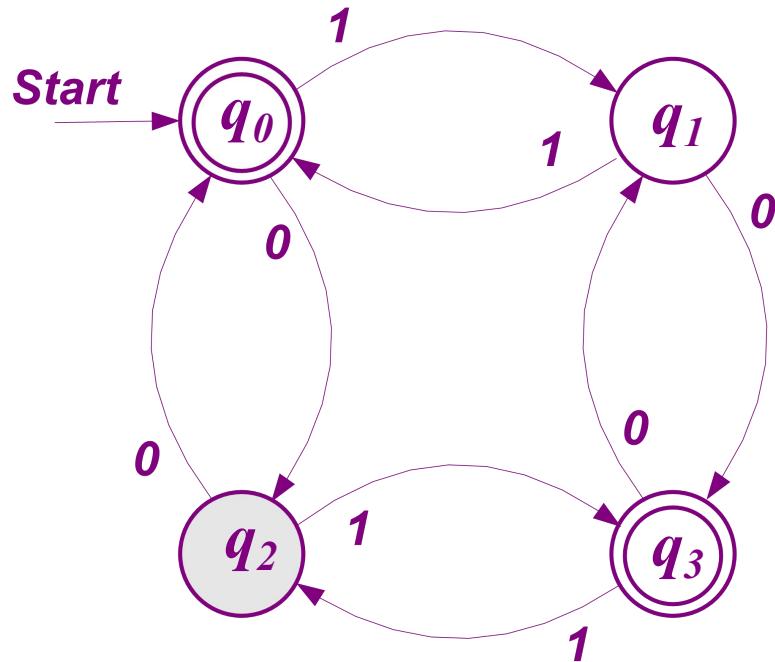
确定有限自动机

◆ DFA如何接受输入符号串



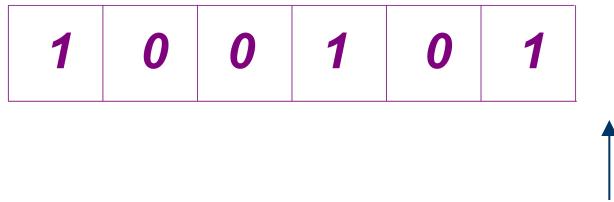
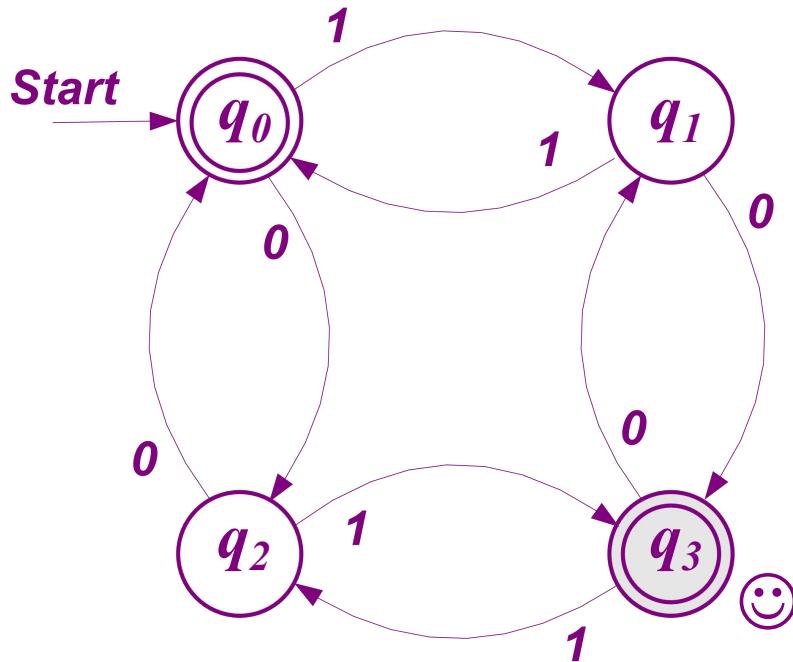
确定有限自动机

◆ DFA如何接受输入符号串



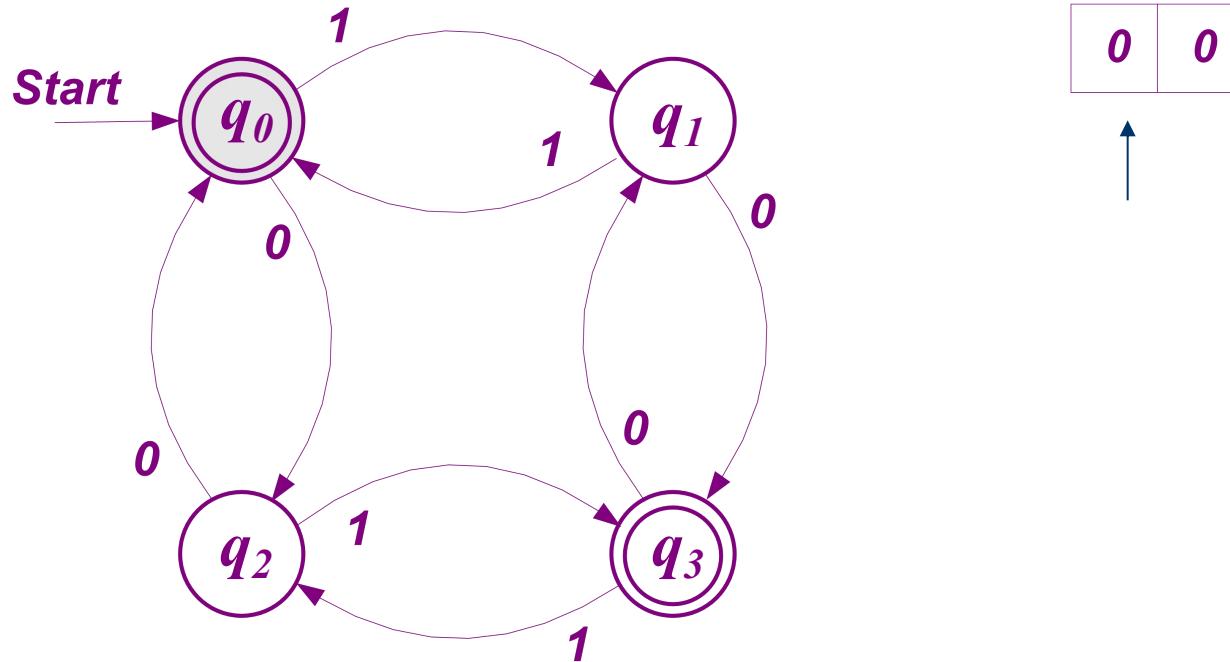
确定有限自动机

◆ DFA如何接受输入符号串



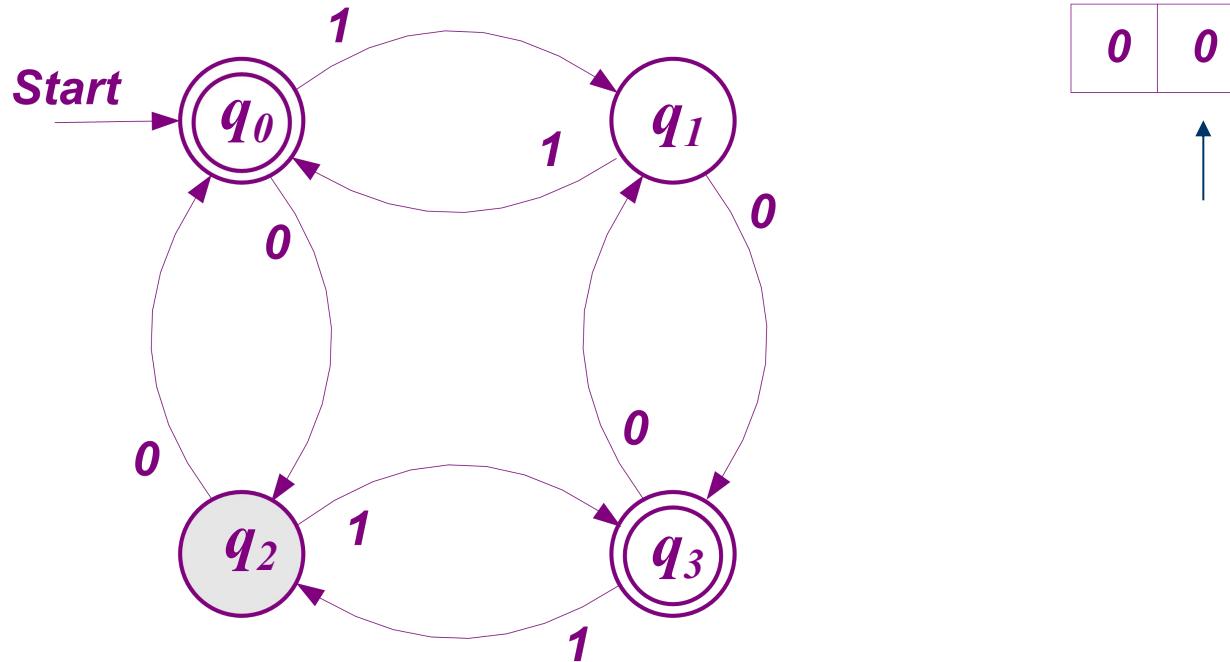
确定有限自动机

◆ DFA如何接受输入符号串



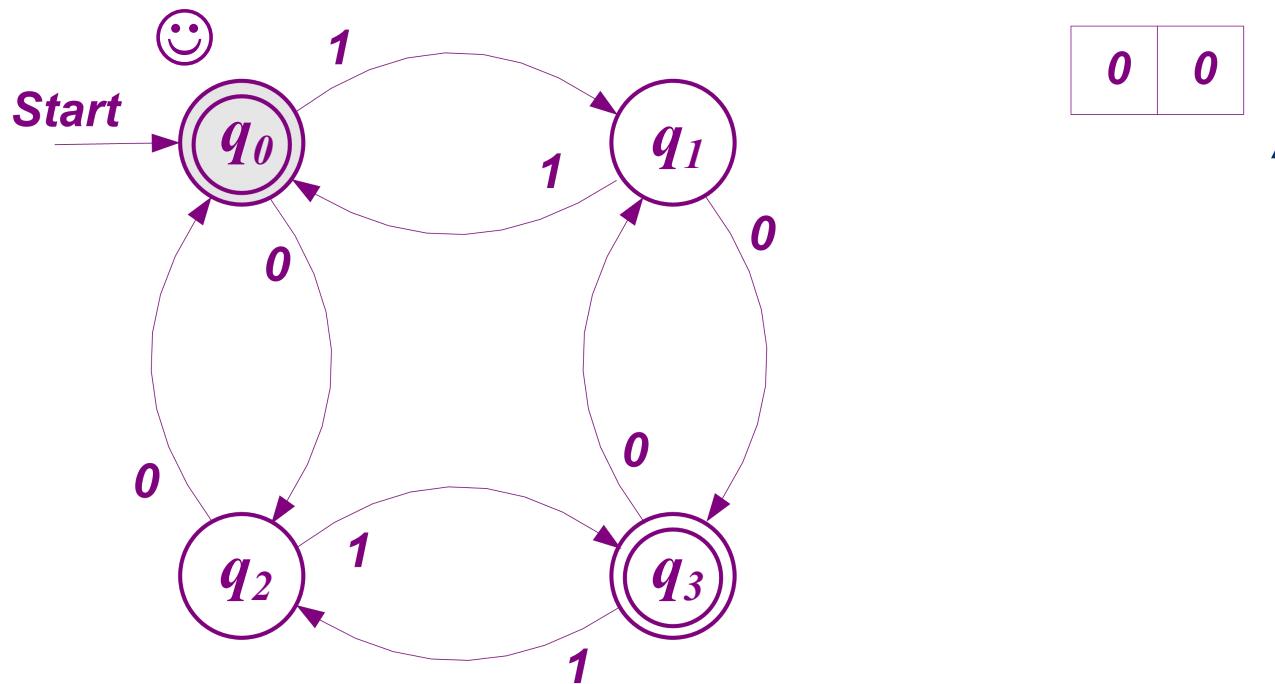
确定有限自动机

◆ DFA如何接受输入符号串



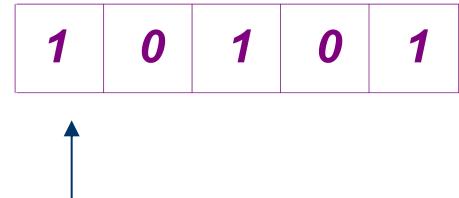
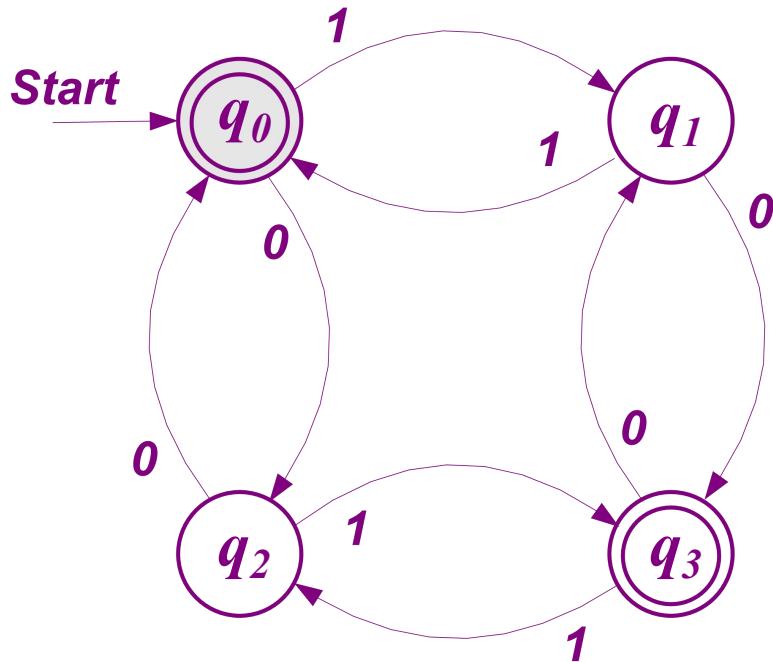
确定有限自动机

◆ DFA如何接受输入符号串

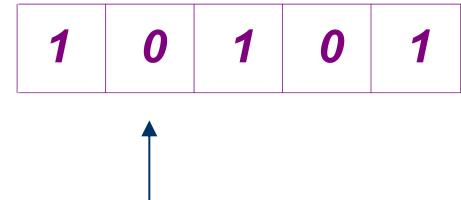
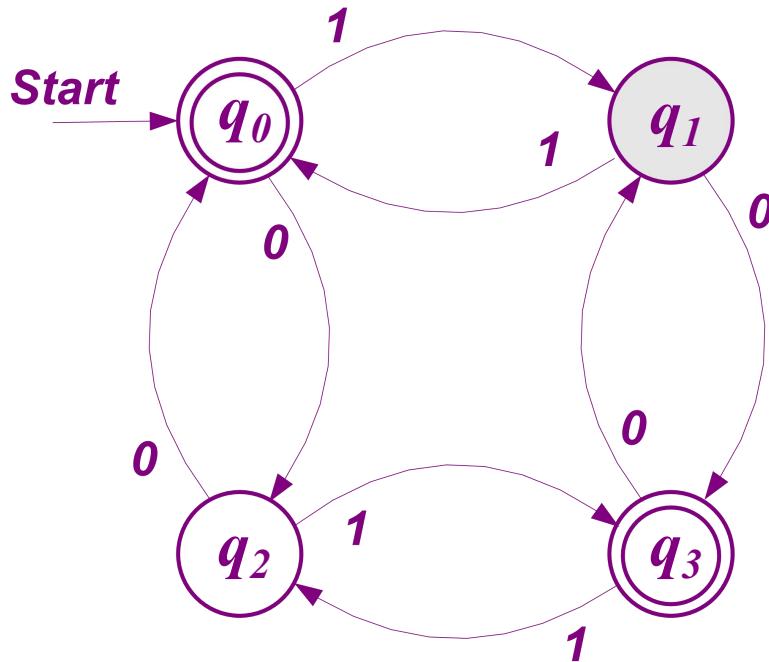


确定有限自动机

◆ DFA如何接受输入符号串

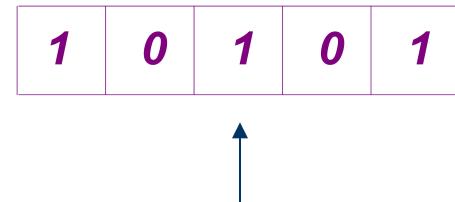
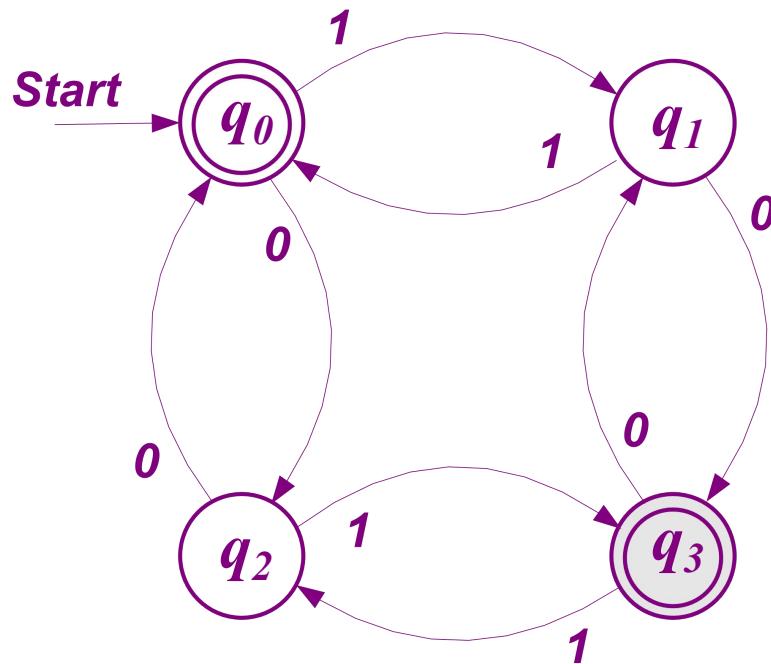


◆ DFA如何接受输入符号串



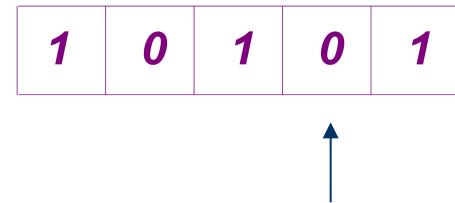
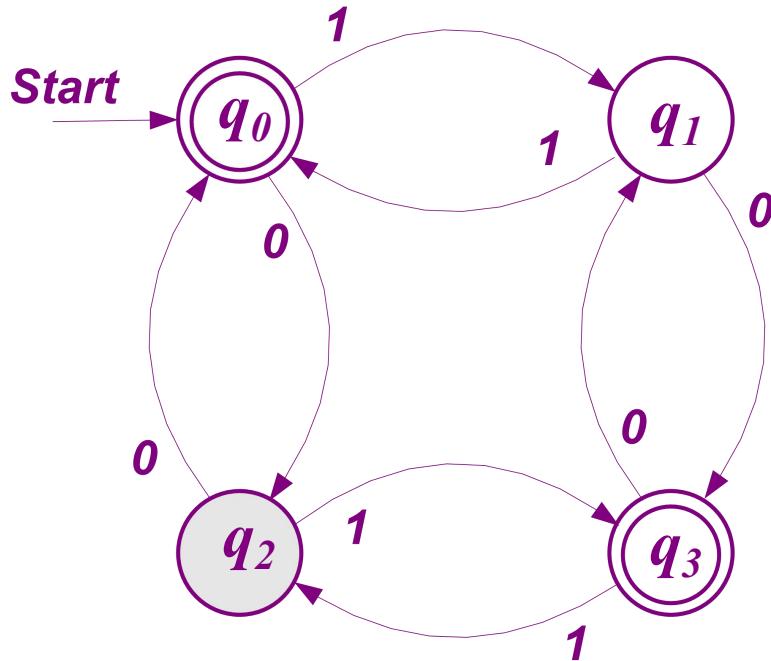
确定有限自动机

◆ DFA如何接受输入符号串



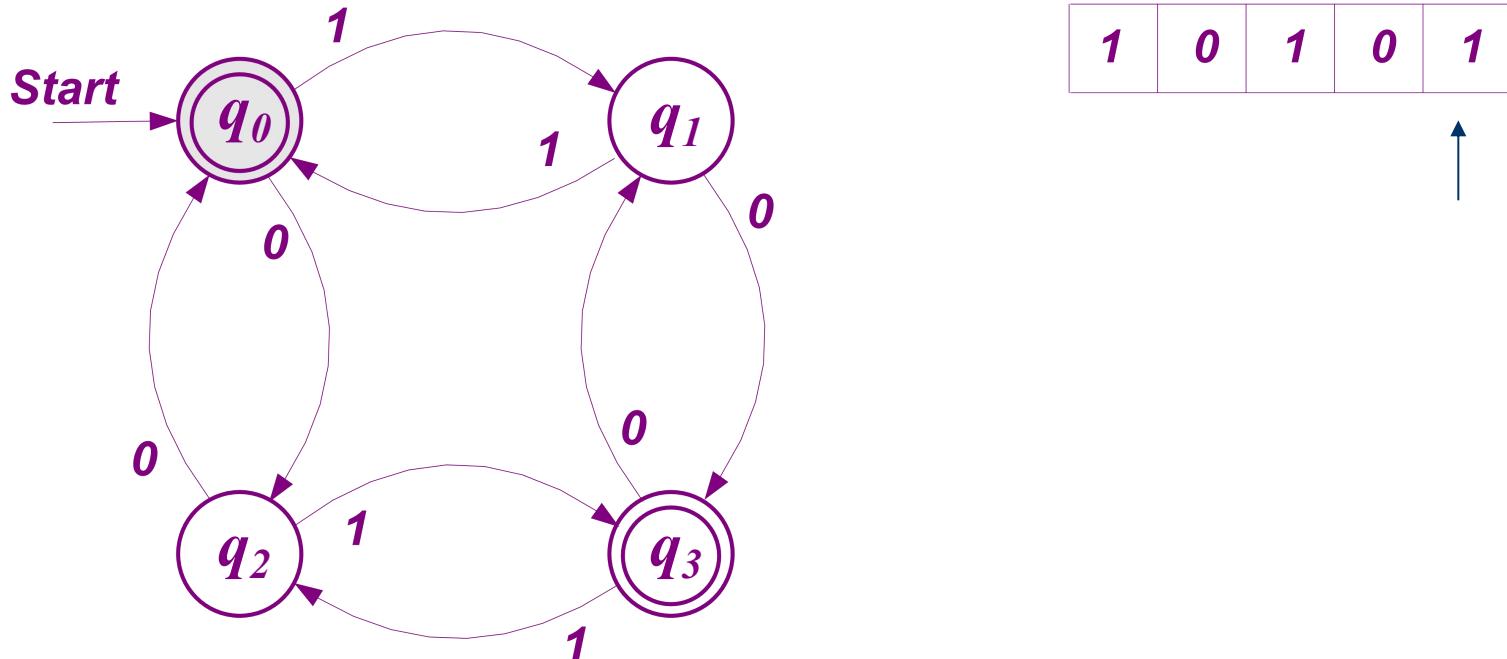
确定有限自动机

◆ DFA如何接受输入符号串



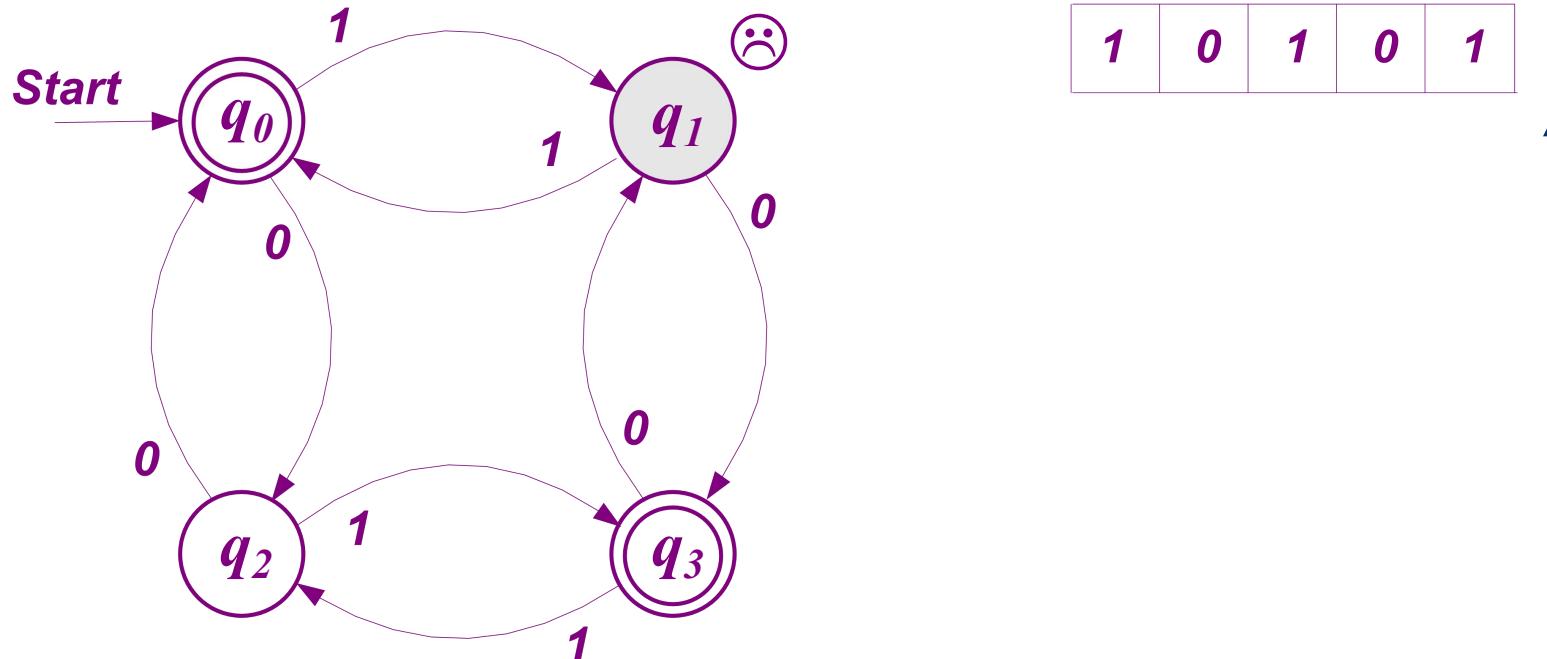
确定有限自动机

◆ DFA如何接受输入符号串



确定有限自动机

◆ DFA如何接受输入符号串



确定有限自动机

◆ DFA如何接受输入符号串

	0	1
$\rightarrow *q_0$	q_1	q_2
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_3	q_0

给出该DFA在读输入串10010001过程中经历的状态序列。

$$\begin{aligned}
 q_0 &\xrightarrow{1} q_2 \xrightarrow{0} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_0 \\
 &\xrightarrow{0} q_1 \xrightarrow{0} q_3 \xrightarrow{0} q_3 \xrightarrow{1} q_0
 \end{aligned}$$

确定有限自动机

◆ 扩展转移函数适合于输入字符串

– 设一个 DFA $A = (Q, \Sigma, \delta, q_0, F)$

$$\delta: Q \times \Sigma \rightarrow Q$$

– 扩充定义 $\delta': Q \times \Sigma^* \rightarrow Q$

对任何 $q \in Q$, 定义:

$$1 \ \delta'(q, \varepsilon) = q$$

2 若 $w = xa$, 其中 $x \in \Sigma^*$, $a \in \Sigma$, 则

$$\delta'(q, w) = \delta(\delta'(q, x), a)$$

确定有限自动机

◆ 扩展转移函数适合于输入字符串

	0	1
$\rightarrow *q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
$*q_3$	q_1	q_2

- 举例

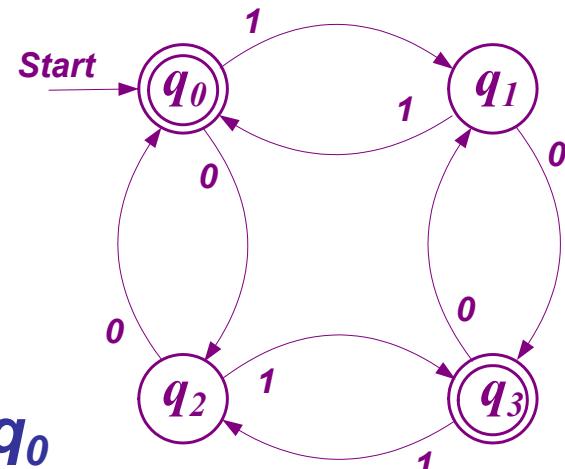
$$\delta'(q_0, \varepsilon) = q_0$$

$$\delta'(q_0, 0) = \delta(q_0, 0) = q_2$$

$$\delta'(q_0, 00) = \delta(q_2, 0) = q_0$$

$$\delta'(q_0, 001) = \delta(q_0, 1) = q_1$$

$$\delta'(q_0, 0010) = \delta(q_1, 0) = q_3$$



确定有限自动机

✧ DFA 的语言

- 设一个 $DFA \ A = (Q, \Sigma, \delta, q_0, F)$
- 定义 A 的语言:

$$L(A) = \{ w \mid w \in \Sigma^* \wedge \delta'(q_0, w) \in F \}$$

- 设 L 是 Σ 上的语言，如果存在一个 DFA $A = (Q, \Sigma, \delta, q_0, F)$ ，满足 $L = L(A)$ ，则可以证明 L 是一个正规语言.

确定有限自动机

✧ DFA 的语言

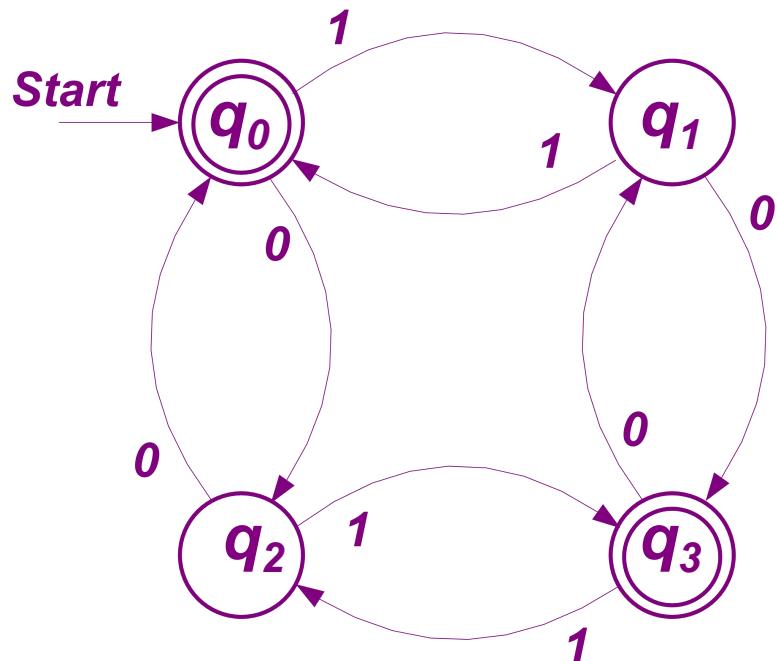
- 举例 $\Sigma = \{ 0, 1 \}$ 上的语言 $L = \{ w \mid w \text{ 中 } 0、1 \text{ 数目的奇偶性相同} \}$, 则 L 是一个正规语言. 可证 L 是如下 **DFA** 的语言.

- 证明思路

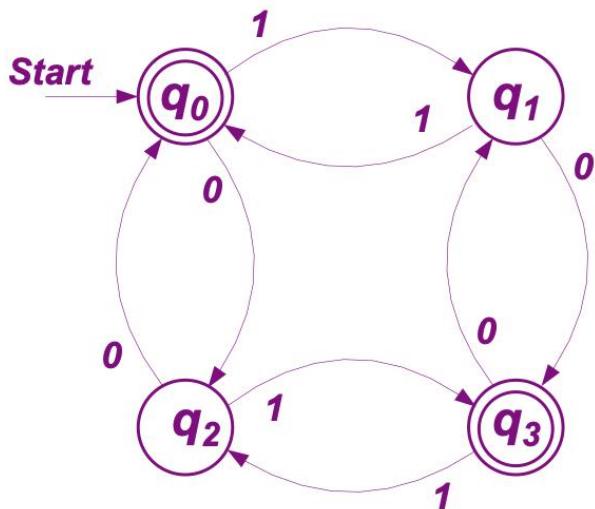
(采用互归纳法,

参考 *Example 2.4*

和 *Example 1.23*)



确定有限自动机



证：使用互归纳法，按照DFA的状态引入四个命题

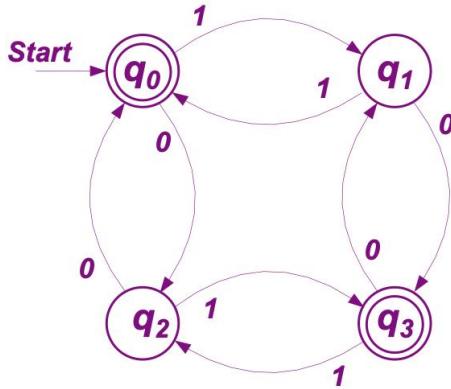
$P_0(n) : \delta'(q_0, w) = q_0$ 当且仅当 w 包含偶数个 0 偶数个 1

$P_1(n) : \delta'(q_0, w) = q_1$ 当且仅当 w 包含偶数个 0 奇数个 1

$P_2(n) : \delta'(q_0, w) = q_2$ 当且仅当 w 包含奇数个 0 偶数个 1

$P_3(n) : \delta'(q_0, w) = q_3$ 当且仅当 w 包含奇数个 0 奇数个 1

确定有限自动机



首先证明正方向的命题，归纳于字符串的长度。

(基础) 分别考察四个命题：

1. $\vec{P}_0(0)$: 当 $|w| = 0$ 时，处于初始状态 q_0 ，这是因为 $\delta'(q_0, \epsilon) = q_0$ 。由于 w 中包含 0 个 0 和 0 个 1，命题成立。
2. $\vec{P}_1(0)$: 当 $|w| = 0$ 时，并不处于状态 q_1 ，前提为假，命题为真。
3. $\vec{P}_2(0)$: 当 $|w| = 0$ 时，并不处于状态 q_2 ，前提为假，命题为真。
4. $\vec{P}_3(0)$: 当 $|w| = 0$ 时，并不处于状态 q_3 ，前提为假，命题为真。

确定有限自动机

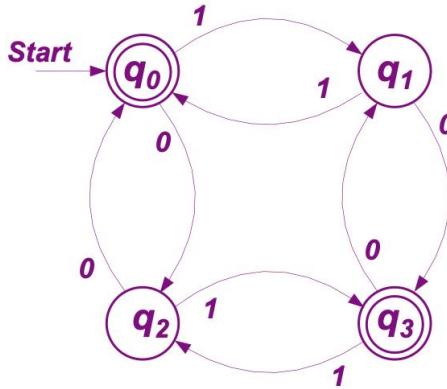
(归纳) 假设 $w = xa$, 其中 a 是一个符号, 或者是0, 或者是1。假设四个命题在 $|x| = n$ 时成立, 下面考虑 $|w| = n + 1$ 的情况。

首先考察 $\vec{P}_0(n+1)$ 。由于此时处于状态 q_0 , 根据自动机的定义, 在接受一个符号 a 进入 q_0 之前, 只可能处于 q_1 或者 q_2 (仅有两条入边进入 q_0)。分情况讨论:

1. 如果前一个状态是 q_1 , 字符串的长度为 $|x| = n$, 根据归纳前提 $\vec{P}_1(n)$, x 中包含偶数个0和奇数个1。从 q_1 到 q_0 要再接受1个1, 因此 $w = xa$ 中必然包含偶数个0和偶数个1。命题成立。
2. 如果前一个状态是 q_2 , 字符串的长度为 $|x| = n$, 根据归纳前提 $\vec{P}_2(n)$, x 中包含奇数个0和偶数个1。从 q_2 到 q_0 要再接受1个0, 因此 $w = xa$ 中必然包含偶数个0和偶数个1。命题成立。

其余三个命题可以类似处理, 在此省略。

确定有限自动机



然后证明反方向的命题，归纳于字符串的长度。

(基础) 分别考察四个命题：

1. $\overleftarrow{P}_0(0)$: 当 $|w| = 0$ 时， w 包含偶数个0和偶数个1，由于没有接受任何字符串，因此处于初始状态 q_0 ，命题成立。
2. $\overleftarrow{P}_1(0)$: 当 $|w| = 0$ 时， w 不包含偶数个0和奇数个1，前提为假，命题为真。
3. $\overleftarrow{P}_2(0)$: 当 $|w| = 0$ 时， w 不包含奇数个0和偶数个1，前提为假，命题为真。
4. $\overleftarrow{P}_3(0)$: 当 $|w| = 0$ 时， w 不包含奇数个0和奇数个1，前提为假，命题为真。

确定有限自动机

(归纳) 假设 $w = xa$, 其中 a 是一个符号, 或者是0, 或者是1。假设四个命题在 $x = n$ 时成立, 下面考虑 $|w| = n + 1$ 的情况。

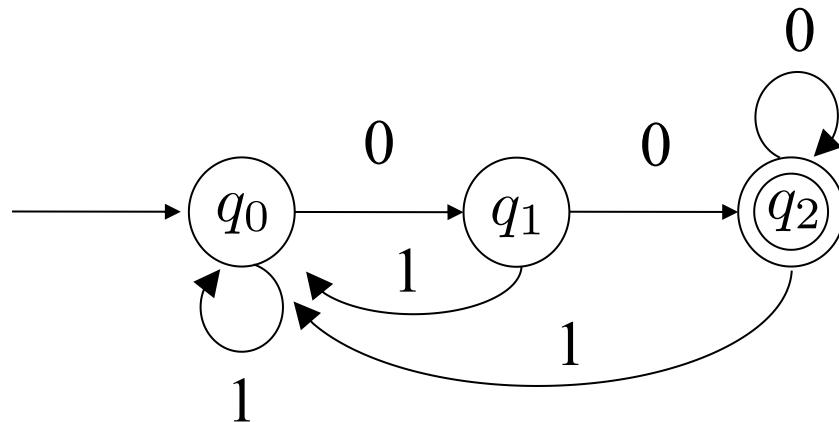
首先考察 $\overleftarrow{P}_0(n+1)$ 。由于 w 中包含偶数个0和偶数个1, 而 a 或者是0, 或者是1。所以要分两种情况讨论。

1. 如果 a 是0, 则 x 中一定包含奇数个0和偶数个1。根据前前提假设 $\overleftarrow{P}_2(n)$, DFA在接受 x 时一定处于 q_2 。从 q_2 再接受一个 $a = 0$ 到达 q_0 。命题成立。
2. 如果 a 是1, 则 x 中一定包含偶数个0和奇数个1。根据前前提假设 $\overleftarrow{P}_1(n)$, DFA在接受 x 时一定处于 q_1 。从 q_1 再接受一个 $a = 1$ 到达 q_0 。命题成立。

其他三个命题可以类似处理, 在此省略。

确定有限自动机

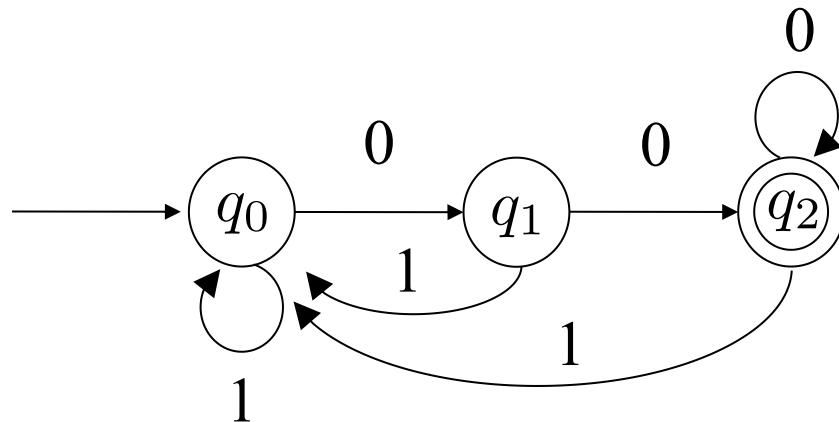
例1：设计定义在字母表 $\{0, 1\}$ 上的所有以00结尾的串的集合的DFA。



解：首先画出三个状态，接受两个连续的0到达终止状态。由于DFA要求每个节点的出边必须包含字母表上的所有符号，因此，必须要补上所缺的标为1出边，同时也不能忘记终止状态还有一个标为0的出边。最后补上状态编号。

确定有限自动机

例1：设计定义在字母表 $\{0, 1\}$ 上的所有以00结尾的串的集合的DFA。

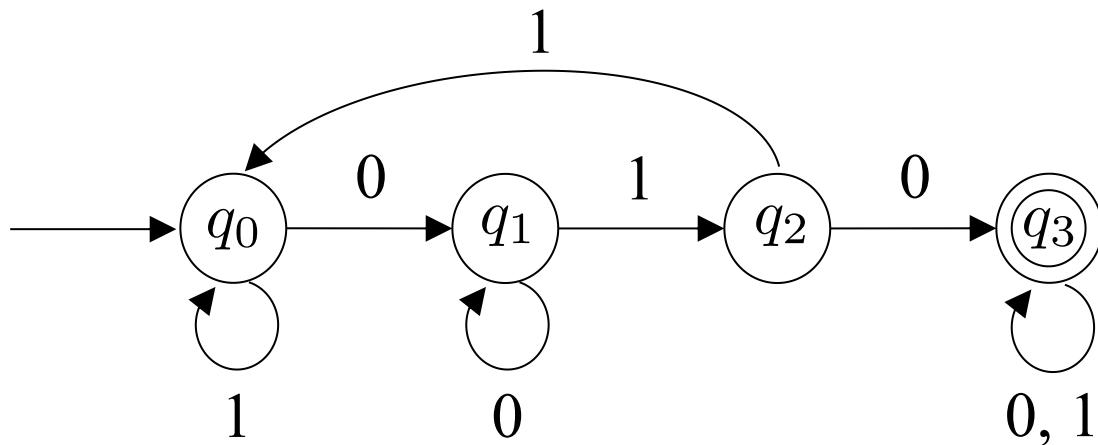


解：三个状态其实都有明确的含义。 q_0 表示出现第一对连续的0之前的任意字符串。 q_1 表示第一对连续的0之中的第一个0，一旦在该状态下碰到了1，必须转回到 q_0 。 q_2 则是终止状态，可以接受一对连续的0，也可以接受更多的连续的0。

确定有限自动机

例2：设计DFA接受以下语言

$$L = \{x \mid x \in \{0, 1\}^* \wedge x \text{ 中至少包含一个子串 } 010\}$$

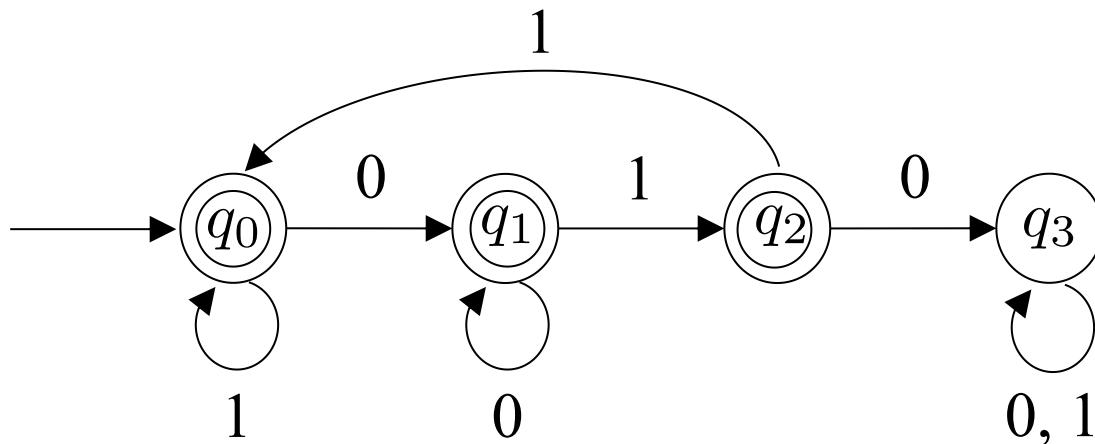


解：这道题可以采用类似与例1的技巧，首先确定主线上的节点和边，然后再补所缺的出边。一定要想清楚出边应该指向哪个节点。做完后可以列举几个符合条件的串检查一下。

确定有限自动机

例3：设计DFA接受以下语言

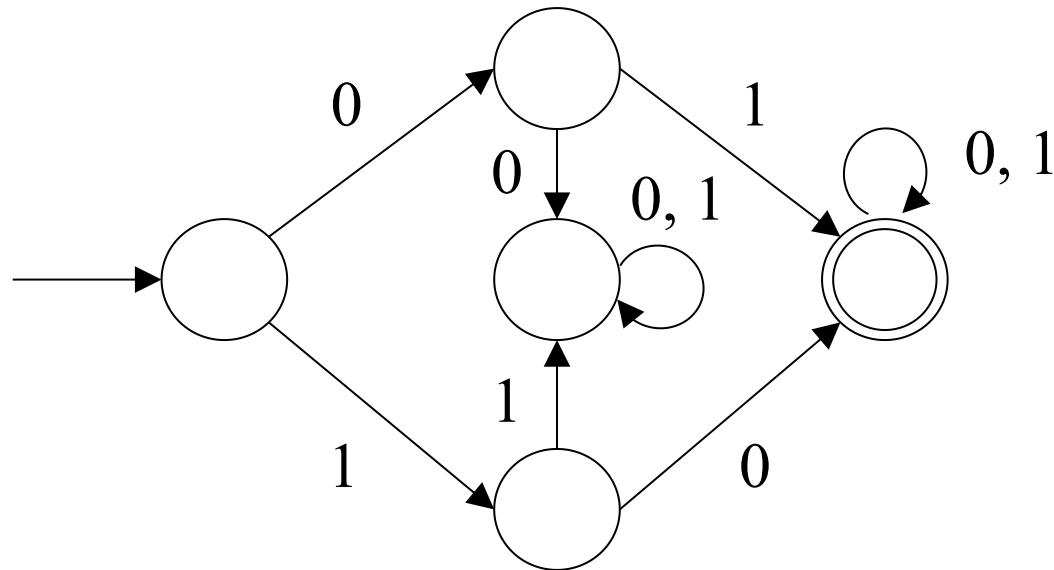
$$L = \{x \mid x \in \{0, 1\}^* \wedge x \text{中不包含子串 } 010\}$$



解：这道题看上去似乎无从下手，怎么表示“不包含”？其实可以从补集的角度去考虑，体现在DFA上，就是保留例2中DFA的结构，只不过互换终结状态和非终结状态。

确定有限自动机

例4：设计DFA表示所有长度至少为2且头两个字符不同的0和1组成的串的集合。

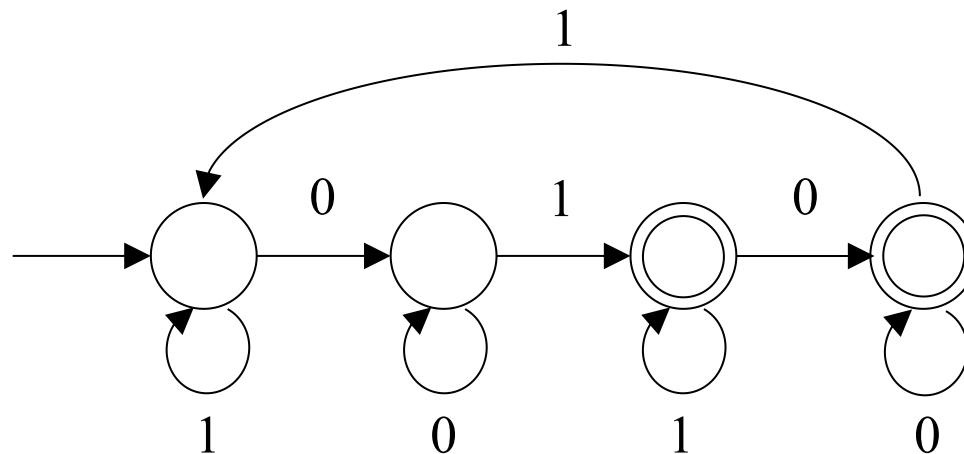


解：思路和前面一样，首先完成主线的构建，然后再对每个节点补出边。这里专门引入了一个节点充当“垃圾箱”，收无效的字符串，这是因为DFA要求每个节点必须有出边。

确定有限自动机

例5：设计DFA表示以下语言

$$L = \{x \mid x \in \{0, 1\}^* \wedge \text{子串01在} x \text{中出现的次数为奇数}\}$$

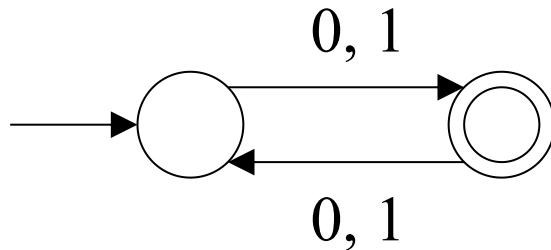


解：此题的难点在于如何实现奇数。基本的策略是即将形成偶数个01子串时就远离终结状态。需要注意的是，该串并不子串01的幂，只是包含子串01，所以还有其他的子串。初始节点的含义是包含了偶数个子串01。

确定有限自动机

例6：设计DFA表示以下语言

$$L = \{x \mid x \in \{0, 1\}^* \wedge 0 \text{的个数与1的个数之和为奇数}\}$$

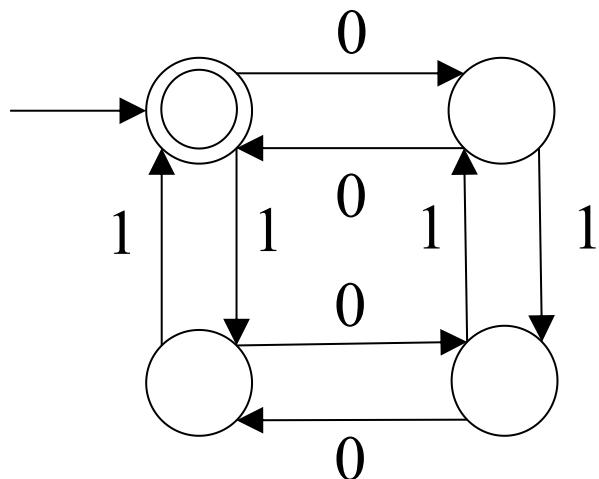


解：这道题比较简单。题目等价于字符串的长度是奇数，这样就类似于电灯开关，比如按奇数次才处于开的状态。因此，很容易设计相应的DFA。

确定有限自动机

例7：设计DFA表示以下语言

$$L = \{x \mid x \in \{0, 1\}^* \wedge 0 \text{的个数是偶数, } x \text{的长度也是偶数}\}$$

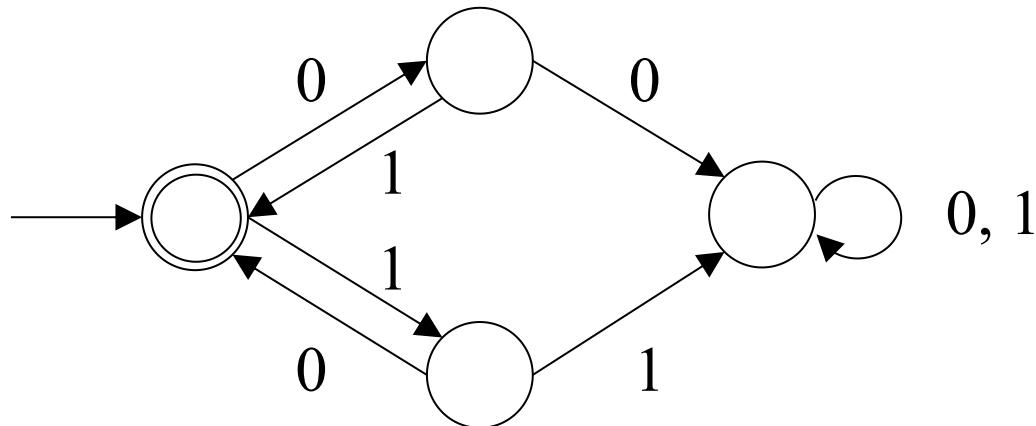


解：题目等价于0的个数是偶数，1的个数也是偶数。根据0和1的奇、偶性，可以划分为4种状态，类似于互归纳法，相互关联。这样就比较容易设计DFA了。

确定有限自动机

例8：设计DFA表示以下语言

$$L = \{x \mid x \in \{0, 1\}^* \wedge x \text{中包含相同个数的0和1} \wedge x \text{的每个前缀0和1的个数之差不超过1}\}$$

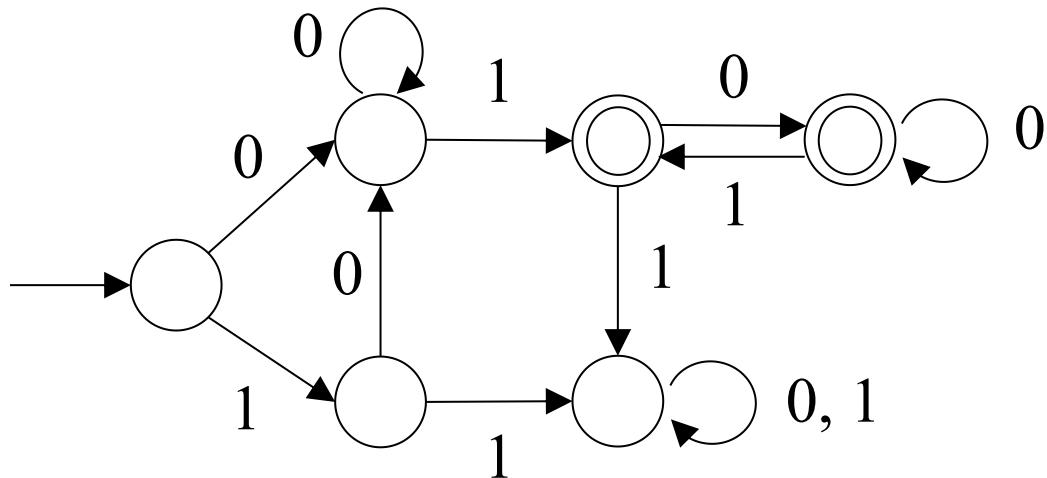


解：这道题的难点是如何实现每个前缀中0和1的个数之差不超过1。实现起来，就是只要出现0，紧接着要出现1，反之亦然。再加上垃圾回收节点，整个DFA就容易构建了。

确定有限自动机

例9：设计DFA表示以下语言

$$L = \{x \mid x \in \{0, 1\}^* \wedge x \text{ 包含子串 } 01 \text{ 但不包含子串 } 11\}$$



解：这道题看上去很复杂，既有包含关系，又有不包含关系。但基本策略不变，还是先把主线上的节点和连线确定，再补齐所有的出边。

确定有限自动机

例10：设计DFA表示所有被3整除的二进制数。

解：既然是二进制数，那么实际上就是字母表是{0, 1}的字符串集合。DFA在处理字符串时，往往采用 $w = xa$ 的形式，即在当前状态下（接受了 x ）再接受一个符号进入新的状态。我们可以把 x 看成是一个二进制数，如果再追加一个0，相当于乘以2。例如，二进制数“11”表示十进制数“3”，追加一个0之后，二进制数“110”表示十进制数“6”。同理，在二进制数上追加一个1，相当于乘以2再加上1。例如，二进制数“111”表示十进制数7。

通过上述策略，我们便建立了二进制数和DFA之间的直接关联。接下来看怎么处理“被3整除”。

确定有限自动机

例10：设计DFA表示所有被3整除的二进制数。

解：根据整除关系，我们可以把所有的整数分成三类：被3除余数为0、余数为1和余数为2。这三类可以分别对应DFA中的三个状态节点。

接下来看状态节点之间的转移情况。

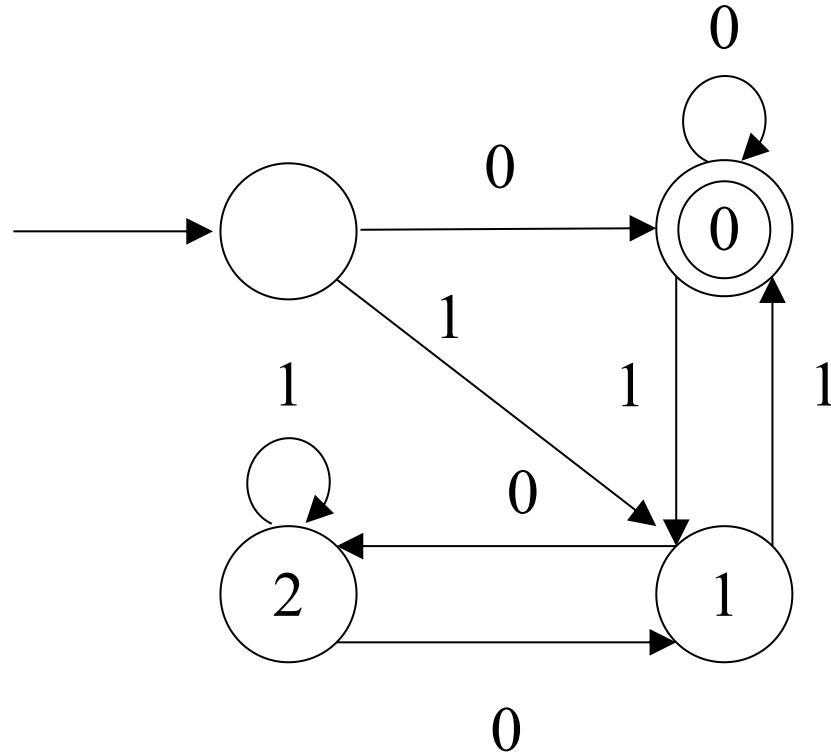
$x \% 3$	$(x0 \% 3)$	$(x1 \% 3)$
0 (e.g., 6)	0 (e.g., 12)	1 (e.g., 13)
1 (e.g., 7)	2 (e.g., 14)	0 (e.g., 15)
2 (e.g., 8)	1 (e.g., 16)	2 (e.g., 17)

因此，可以根据上表来设计DFA。

确定有限自动机

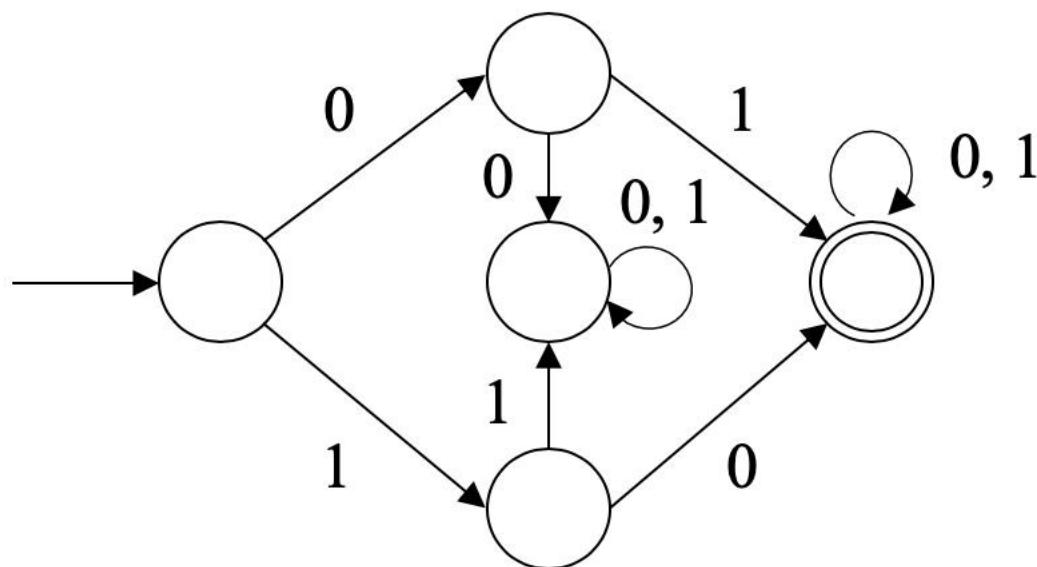
例10：设计DFA表示所有被3整除的二进制数。

解：相应的DFA如下。注意需要增加一个额外的起点。



◆ 根据定义，确定有限自动机的转移函数必须考虑字母表上的所有符号转到有效的状态，使得设计难度增加。

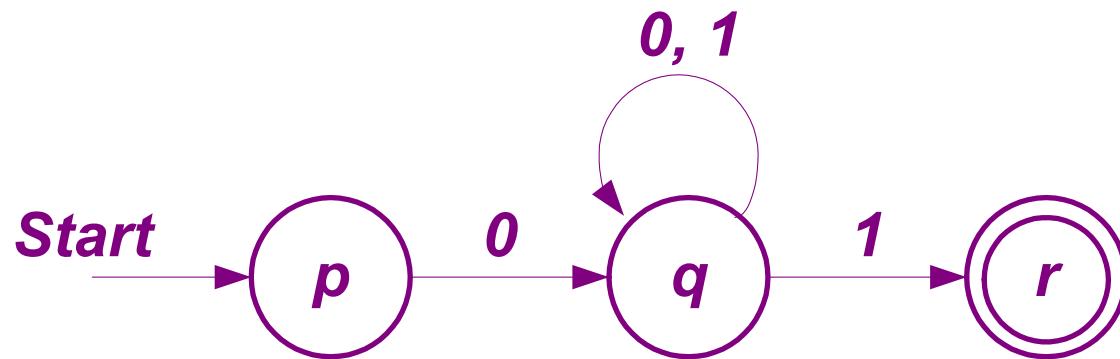
$$\delta: Q \times \Sigma \rightarrow Q$$



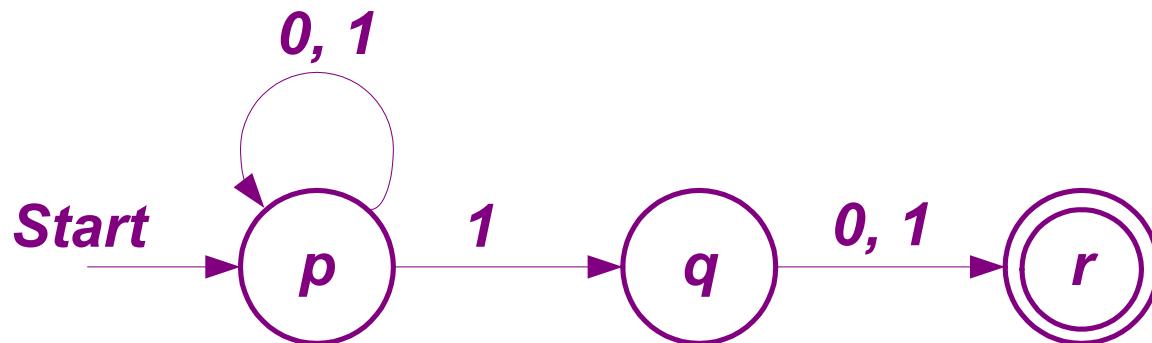
非确定有限自动机

✧ 非确定有限自动机举例

(1)



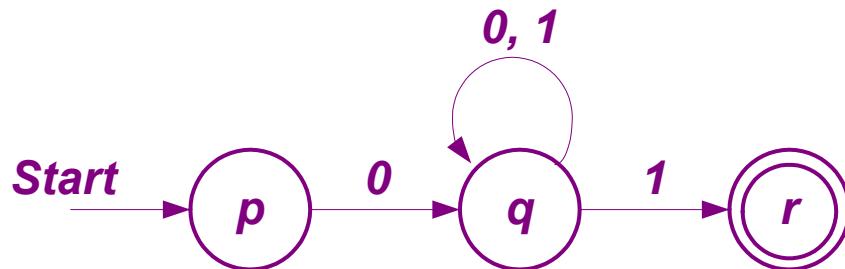
(2)



非确定有限自动机

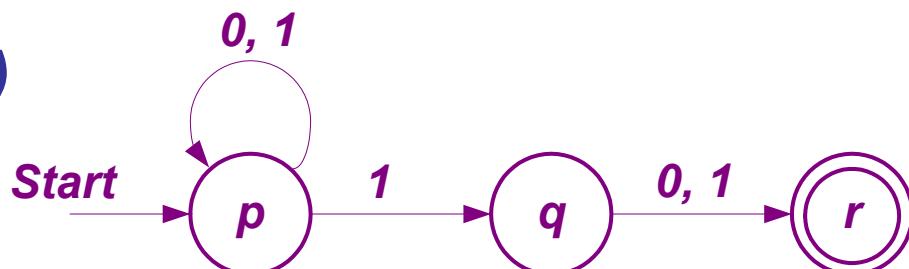
◆ 转移图和转移表表示的NFA

(1)



	0	1
$\rightarrow p$	$\{q\}$	\emptyset
q	$\{q\}$	$\{q, r\}$
$* r$	\emptyset	\emptyset

(2)



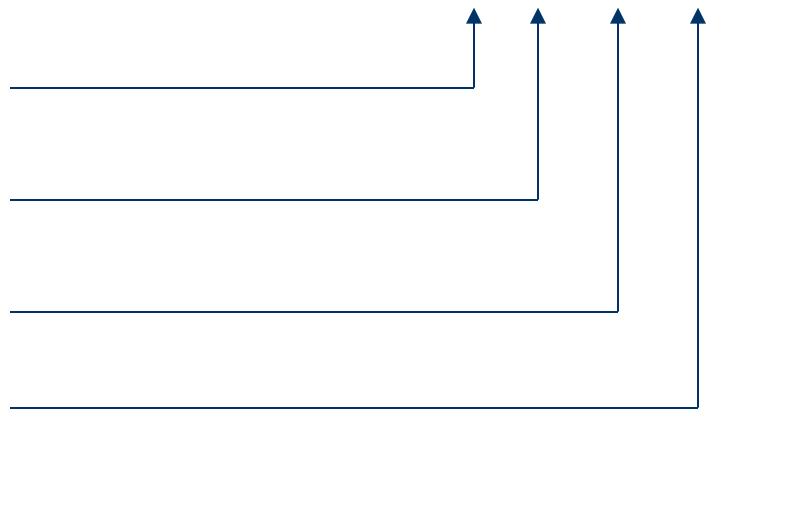
	0	1
$\rightarrow p$	$\{p\}$	$\{p, q\}$
q	$\{r\}$	$\{r\}$
$* r$	\emptyset	\emptyset

非确定有限自动机

✧ 非确定有限自动机的形式定义

一个非确定有限状态自动机 **NFA nondeterministic finite automata**) 是一个五元组 $A = (Q, \Sigma, \delta, q_0, F)$.

- 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
- 与 **DFA** 唯一不同之处



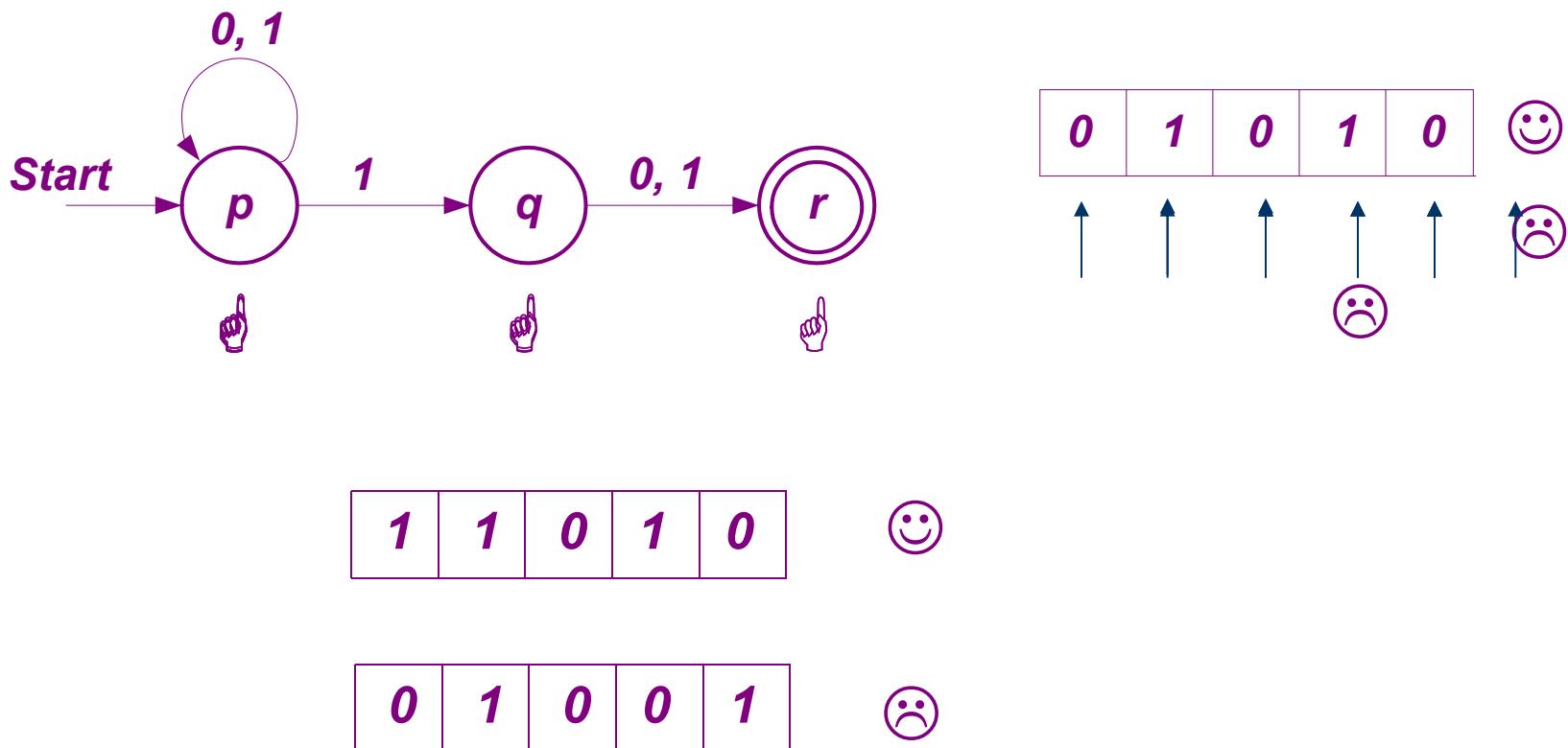
$$q_0 \in Q$$

$$F \subseteq Q$$

$$\delta: Q \times \Sigma \rightarrow 2^Q$$

非确定有限自动机

◆ NFA 如何接受输入符号串



非确定有限自动机

◆ 扩展转移函数适合于输入字符串

- 设一个 $NFA \ A = (Q, \Sigma, \delta, q_0, F)$

- $\delta: Q \times \Sigma \rightarrow 2^Q$

- 扩充定义 $\delta': Q \times \Sigma^* \rightarrow 2^Q$

- 对任何 $q \in Q$, 定义:

- $\delta'(q, \varepsilon) = \{q\}$

- 若 $w = xa$, 其中 $x \in \Sigma^*$, $a \in \Sigma$, 并且假设

$$\delta'(q, x) = \{p_1, p_2, \dots, p_k\}, \text{ 则}$$

$$\delta'(q, w) = \bigcup_{i=1}^k \delta(p_i, a)$$

非确定有限自动机

✧ 扩展转移函数适合于输入字符串

	0	1
$\rightarrow p$	$\{ q \}$	ϕ
q	$\{ q \}$	$\{ q, r \}$
$* r$	ϕ	ϕ

– 举例

$$\delta'(p, \varepsilon) = \{ p \}$$

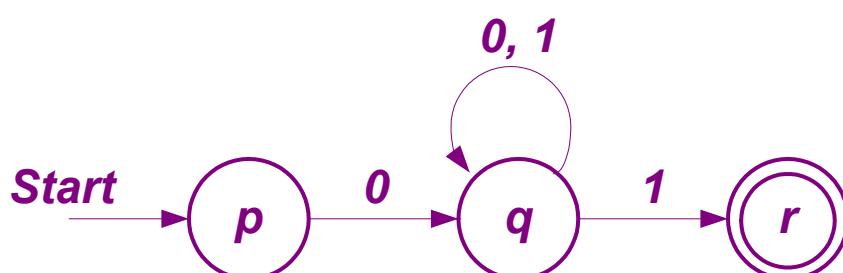
$$\delta'(p, 0) = \{ q \}$$

$$\delta'(p, 01) = \{ q, r \}$$

$$\delta'(p, 010) = \{ q \}$$

$$\delta'(p, 0100) = \{ q \}$$

$$\delta'(p, 01001) = \{ q, r \}$$



非确定有限自动机

✧ NFA 的语言

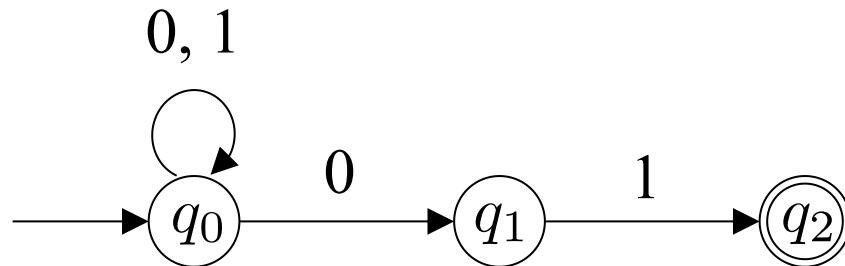
- 设一个 $NFA A = (Q, \Sigma, \delta, q_0, F)$
- 定义 A 的语言：

$$L(A) = \{ w \mid w \in \Sigma^* \wedge \delta'(q_0, w) \cap F \neq \emptyset \}$$

- 设 L 是 Σ 上的语言，如果存在一个 $NFA A = (Q, \Sigma, \delta, q_0, F)$ ，满足 $L = L(A)$ ，则可以证明 L 也是一个正规语言.

非确定有限自动机

例11：证明下图所示的NFA接受所有字母表为 $\{0, 1\}$ 且以01结尾的字符串。



证明：可以构造3个互归纳命题：

1. 对于每个 w , $\delta'(q_0, w)$ 包含 q_0 ;
2. $\delta'(q_0, w)$ 包含 q_1 , 当且仅当 w 以0结尾;
3. $\delta'(q_0, w)$ 包含 q_2 , 当且仅当 w 以01结尾。

非确定有限自动机

(基础) 如果 $|w| = 0$, 则 $w = \epsilon$ 。由于 $\delta'(q_0, \epsilon) = q_0$, 命题1成立。对于命题2和3, 前提均为假, 所以也成立。

(归纳) 假设 $w = xa$, 其中 a 是一个符号(0或1)。假设命题1、2、3对于 x 都成立(令 $|x| = n$), 则需证明 $|w| = n + 1$ 时也成立。

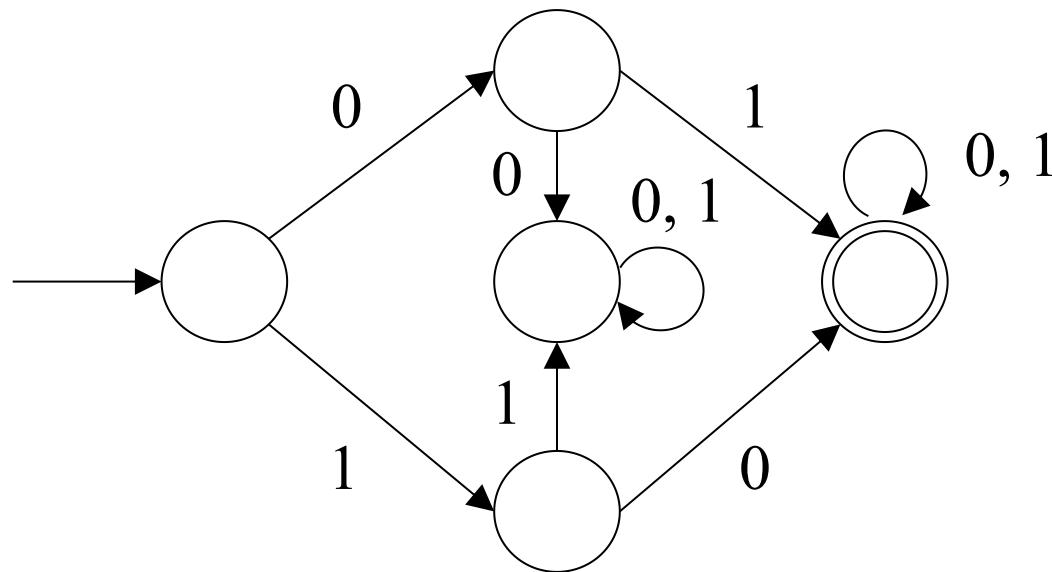
对于命题1, 由于 $\delta'(q_0, x) = q_0$, 而从 q_0 经过0或1都可以到达 q_0 , 所以成立。

对于命题2, 先考虑正方向。假定 $\delta'(q_0, w)$ 包含 q_1 。观察NFA, 则 a 只能是0, 所以成立。在考虑反方向。假定 $a = 0$, 根据命题1, 存在 $\delta'(q_0, x) = q_0$, 从 q_0 经过一个0可以达到 q_1 , 因此命题成立。

对于命题3, 先考虑正方向。假定 $\delta'(q_0, w)$ 包含 q_2 。观察NFA, 则 a 只能是1, 且 $\delta'(q_0, x)$ 包含 q_1 。根据命题2的假设, x 以0结尾, 因此 w 以01结尾, 所以成立。再考虑反方向。假定 w 以01结尾, 则 $a = 1$ 并且 x 以0结尾。根据命题2, $\delta'(q_0, x)$ 包含 q_1 。由于输入1可以实现从 q_1 到 q_2 的转换, 所以命题成立。

非确定有限自动机

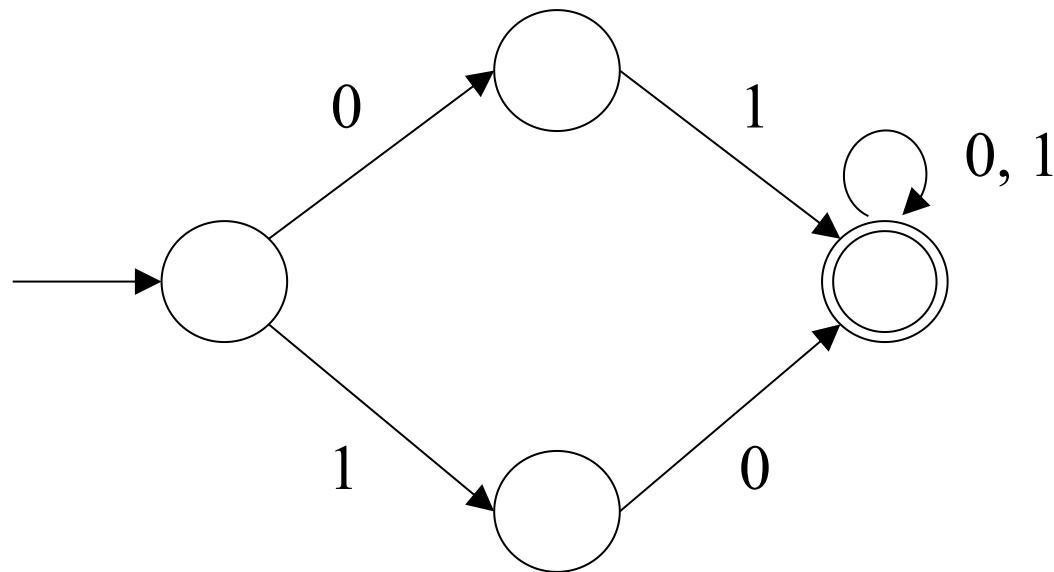
例12：设计NFA，其语言为长度至少为2且头两个字符不相同的0和1串构成的集合。



解：这道题其实和例4一样，只不过要求设计NFA而不是DFA。首先回顾一下当时是如何设计DFA的，首先按照主线来设计节点和边，再补齐所缺的出边。

非确定有限自动机

例12：设计NFA，其语言为长度至少为2且头两个字符不相同的0和1串构成的集合。

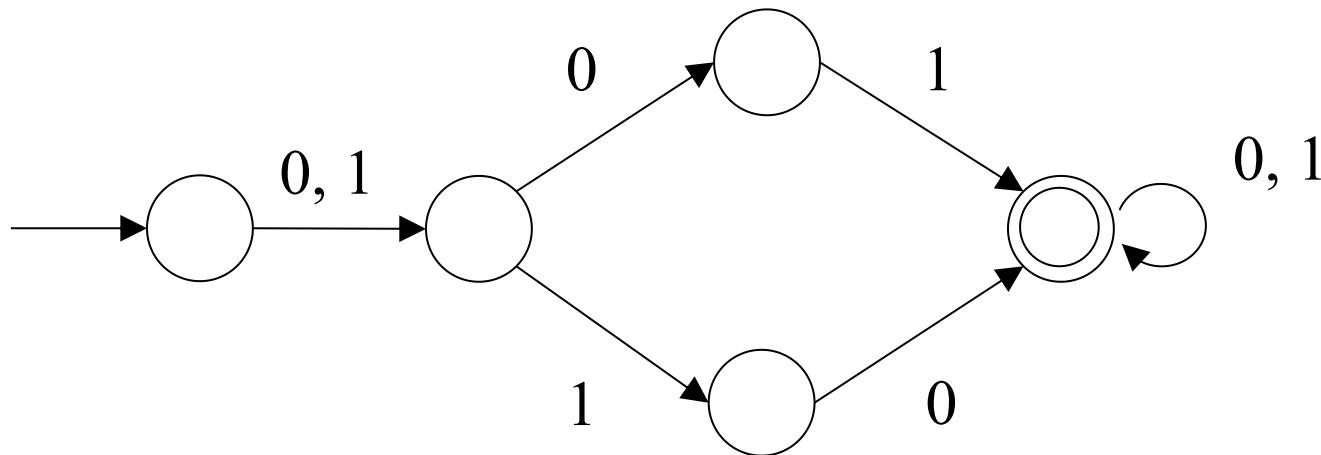


解：在设计NFA时，则根本不需要考虑补齐所缺的出边，因为NFA的转移函数允许一个节点接受一个字符后转移到空集状态。这极大简化了NFA的设计难度！

非确定有限自动机

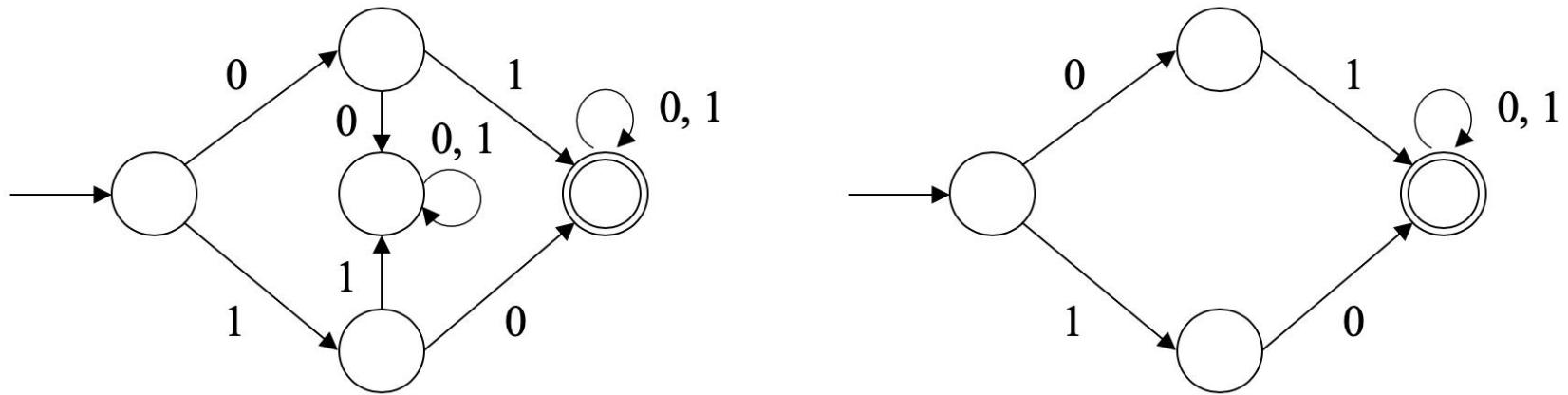
例13：设计NFA接受以下语言

$$L = \{w \mid w \in \{0, 1\}^* \wedge |w| \geq 3 \wedge w\text{中第2位和第3位不同}\}$$



解：基本设计方法不变，还是沿着主线设计。第一个位置可是0或者1，第2位和第3位分成两条路线，确保题目要求的不同，最后是任意字符串。可见NFA的设计比DFA简单很多。

DFA 和 NFA 的比较



	DFA	NFA
分析歧义	无	有
设计难度	高	低

DFA和NFA是等价的吗？

DFA 和 NFA 的等价性

定理: L 是某个 **DFA** 的语言, 当且仅当 L 也是某
个 **NFA** 的语言.

证明: 分两步证明.

(1) 设 L 是某个 **DFA** D 的语言, 则存在一个
NFA N , 满足 $L(N) = L(D) = L$;

(2) 设 L 是某个 **NFA** N 的语言, 则存在一个
DFA D , 满足 $L(D) = L(N) = L$

DFA 和 NFA 的等价性

✧ 从 DFA 构造等价的 NFA

- 设 L 是某个 DFA $D = (Q, \Sigma, \delta_D, q_0, F)$ 的语言, 则存在一个 NFA N , 满足 $L(N) = L(D) = L$.
- 证明: 定义 $N = (Q, \Sigma, \delta_N, q_0, F)$, 其中 δ_N 定义为
 - 对 $q \in Q$ 和 $a \in \Sigma$,
若 $\delta_D(q, a) = p$, 则 $\delta_N(q, a) = \{p\}$.

需要证明: 对任何 $w \in \Sigma^*$,

$$\delta'_D(q_0, w) = p \text{ iff } \delta'_N(q_0, w) = \{p\}.$$

归纳于 $|w|$ 易证上述命题.

DFA 和 NFA 的等价性

✧ 从 NFA 构造等价的 DFA (子集构造法)

– 设 L 是某个 NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ 的语言, 则存在一个 DFA D , 满足 $L(D) = L(N) = L$.

– 证明: 定义 $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$, 其中

- $Q_D = \{ S \mid S \subseteq Q_N \}$
- 对 $S \in Q_D$ 和 $a \in \Sigma$, $\delta_D(S, a) = \bigcup_{q \in S} \delta_N(q, a)$
- $F_D = \{ S \mid S \subseteq Q_N \wedge S \cap F_N \neq \emptyset \}$

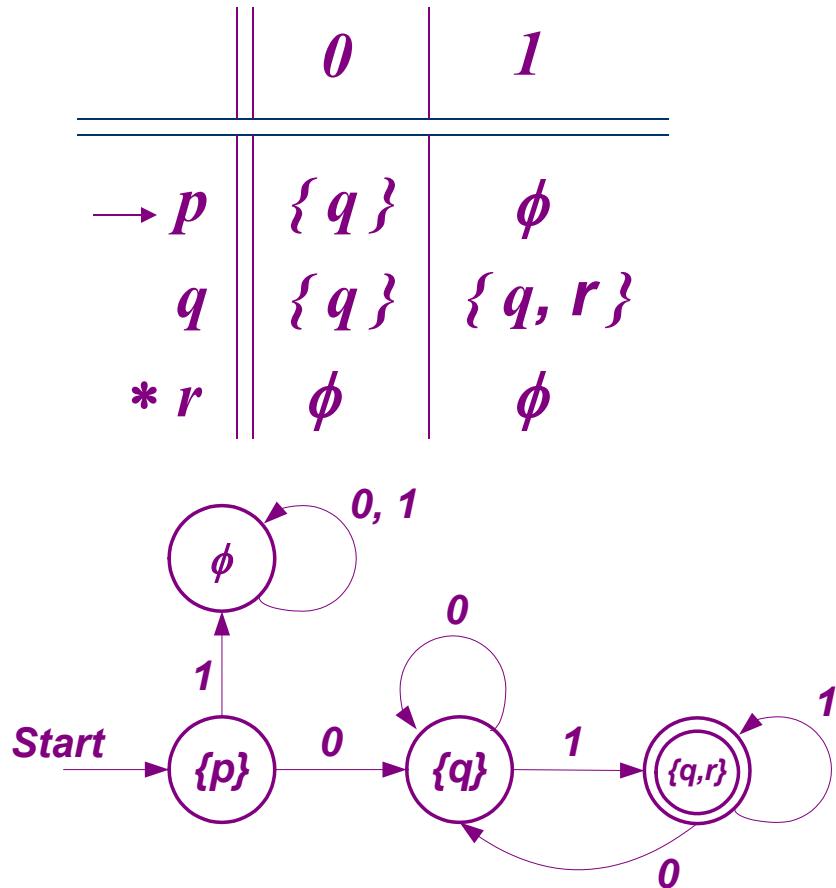
需要证明: 对任何 $w \in \Sigma^*$,

$$\delta'_D(\{q_0\}, w) = \delta'_N(q_0, w).$$

归纳于 $|w|$ 可证上述命题.

DFA 和 NFA 的等价性

✧ 子集构造法举例



	0	1
ϕ	ϕ	ϕ
$\rightarrow \{p\}$	$\{q\}$	ϕ
$\{q\}$	$\{q\}$	$\{q, r\}$
$*\{r\}$	ϕ	ϕ
$\{p, q\}$	$\{q\}$	$\{q, r\}$
$*\{p, r\}$	$\{q\}$	ϕ
$*\{q, r\}$	$\{q\}$	$\{q, r\}$
$*\{p, q, r\}$	$\{q\}$	$\{q, r\}$

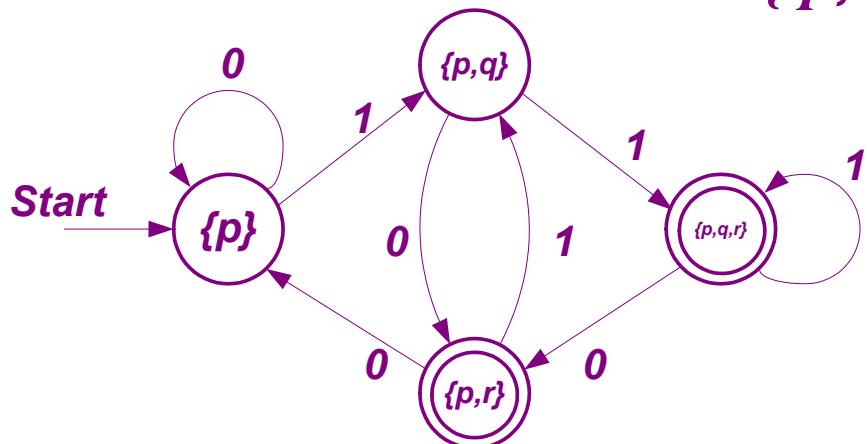
只包含从初始状态出发可达的状态

DFA 和 NFA 的等价性

✧ 子集构造法举例

	0	1
$\rightarrow p$	$\{p\}$	$\{p, q\}$
q	$\{r\}$	$\{r\}$
$* r$	ϕ	ϕ

	0	1
$\rightarrow \{p\}$	$\{p\}$	$\{p, q\}$
$\{p, q\}$	$\{p, r\}$	$\{p, q, r\}$
$* \{p, r\}$	$\{p\}$	$\{p, q\}$
$* \{p, q, r\}$	$\{p, r\}$	$\{p, q, r\}$



DFA 和 NFA 的等价性

例14：利用子集构造法将以下NFA转为DFA。

	0	1
$\rightarrow p$	$\{p, q\}$	$\{p\}$
q	$\{r\}$	$\{r\}$
r	$\{s\}$	\emptyset
$*s$	$\{s\}$	$\{s\}$

解：子集构造法的流程很清晰，从入口出发，不断求并集，得到新的状态。如果该状态包含原先NFA的终止状态，则是DFA的终止状态。执行上述过程直至没有新的状态产生为止。

DFA 和 NFA 的等价性

✧ 从 NFA 构造等价的 DFA (子集构造法)

定理: 设 $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ 是一个 **NFA** , 通过子集构造法得到相应的 **DFA** $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$, 则对任何 $w \in \Sigma^*$, $\delta'_D(\{q_0\}, w) = \delta'_N(q_0, w)$.

证明: 归纳于 $|w|$

1 设 $|w| = 0$, 即 $w = \varepsilon$.

由定义知 $\delta'_D(\{q_0\}, \varepsilon) = \delta'_N(q_0, \varepsilon) = \{q_0\}$.

2 设 $|w| = n+1$, 并 $w = xa$, $a \in \Sigma$. 注意到 $|x| = n$.

假设 $\delta'_D(\{q_0\}, x) = \delta'_N(q_0, x) = \{p_1, p_2, \dots, p_k\}$.

则 $\delta'_D(\{q_0\}, w) = \delta_D(\delta'_D(\{q_0\}, x), a)$

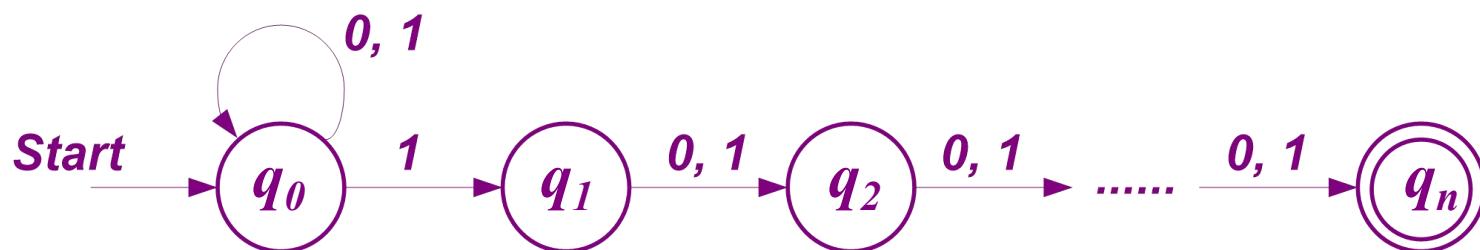
$$= \delta_D(\{p_1, p_2, \dots, p_k\}, a) = \bigcup_{i=1}^k \delta_N(p_i, a)$$

$$= \delta'_N(q_0, w)$$

DFA 和 NFA 的等价性

✧ 子集构造法得到的状态数

- 实践中, 通过子集构造法得到的 **DFA** 的状态数目与原 **NFA** 的状态数目大体相当
- 在较坏的情况下, 上述 **DFA** 的状态数目接近于所有子集的数目
- 举例 由如下 **NFA** 构造的 **DFA** 的状态数目至少为 2^n



DFA 和 NFA 的等价性

✧ 子集构造法得到的状态数

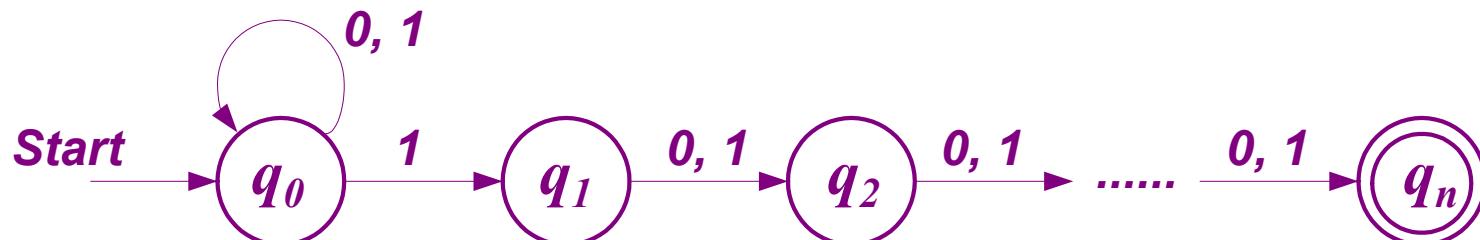
- 上页例子的证明要点，采用反证法

假设由此 **NFA** 构造的 **DFA** 的状态数目少于 2^n

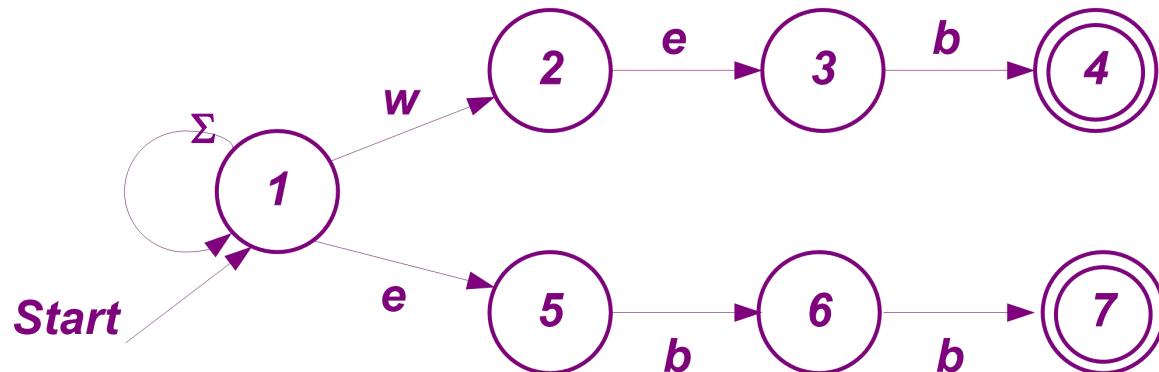
考虑长度为 n 的 $0, 1$ 串共有 2^n 个，所以存在两个不同的串 $a_1a_2\dots a_n$ 和 $b_1b_2\dots b_n$ 做为该 **DFA** 的输入，可以到达同一状态 q . (by **Pigeonhole Principle**)

若 $a_1 \neq b_1$, 则 q 既是终态又是非终态，矛盾；

一般情况，若 $a_k \neq b_k$ ，设 $a_1a_2\dots a_n 00\dots 0$ ($k-1$ 个 0) 或 $b_1b_2\dots b_n 00\dots 0$ ($k-1$ 个 0) 作为输入串时该 **DFA** 到达状态 p , 则 p 既是终态又是非终态，矛盾。

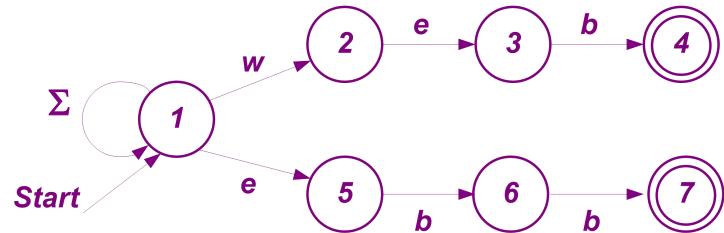


- ✧ 举例 设计一个 **NFA** 用来在文本中搜索字符串 **web** 和 **ebb.**
- ✧ 解 下图为一个满足条件的 **NFA**, 其中 Σ 代表所有 **ASCII** 字符的集合.

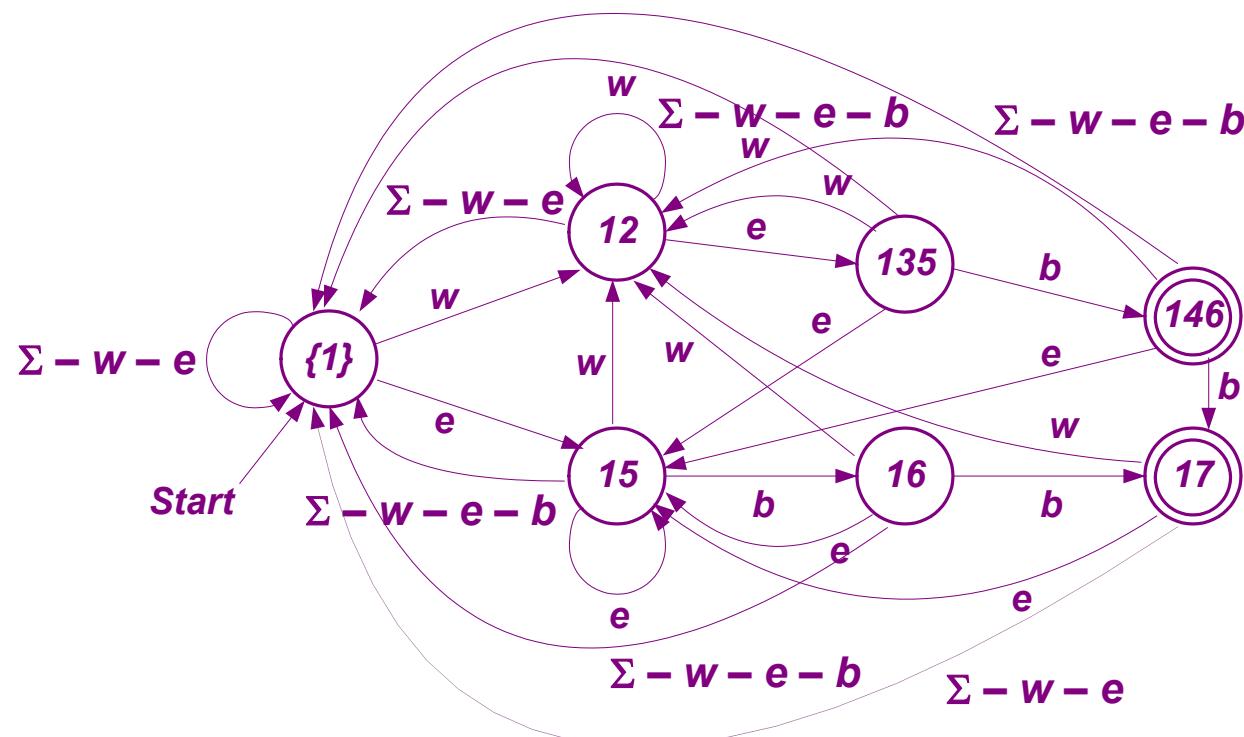
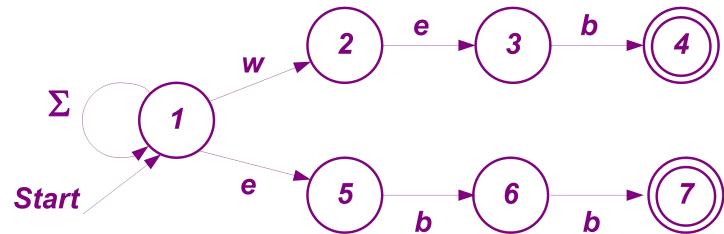


之所以是NFA是因为 Σ 包含了 w 和 e

✧ 举例 构造与前面 **NFA**
等价的 **DFA**.

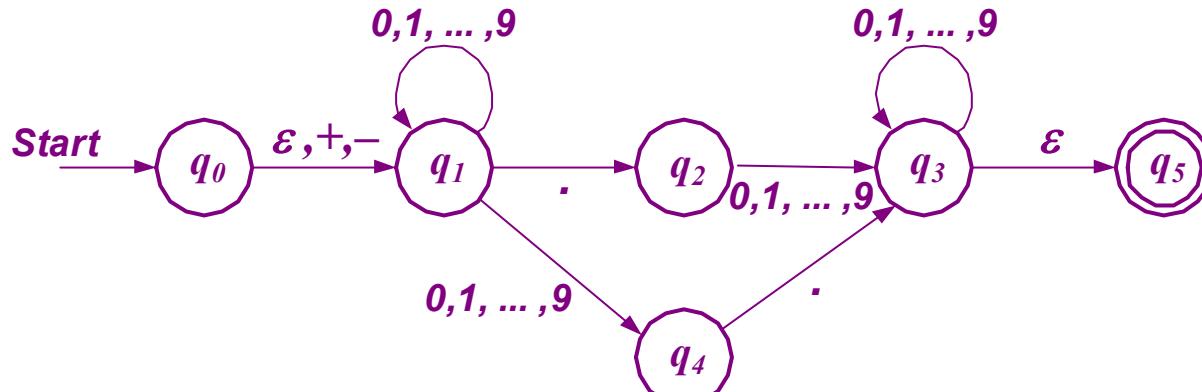


✧ 举例 构造与前面 NFA 等价的 DFA.

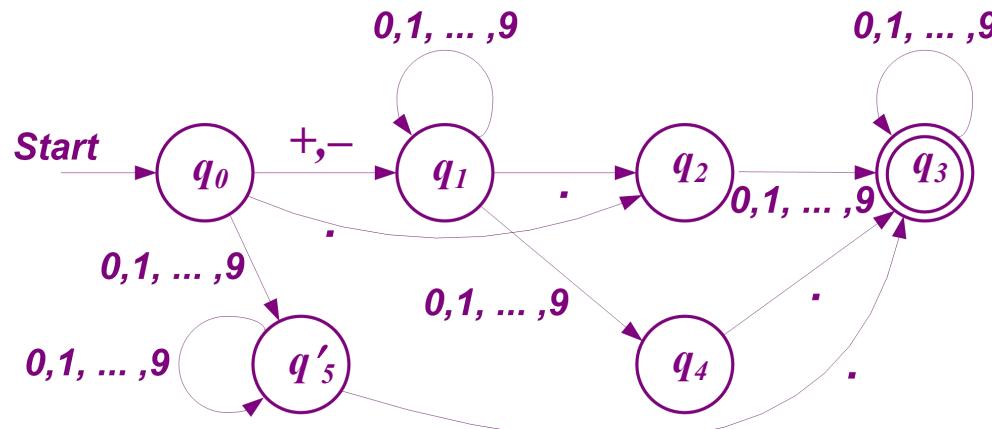


带 ϵ -转移的非确定有限自动机

✧ 举例



比较: **NFA without ϵ**



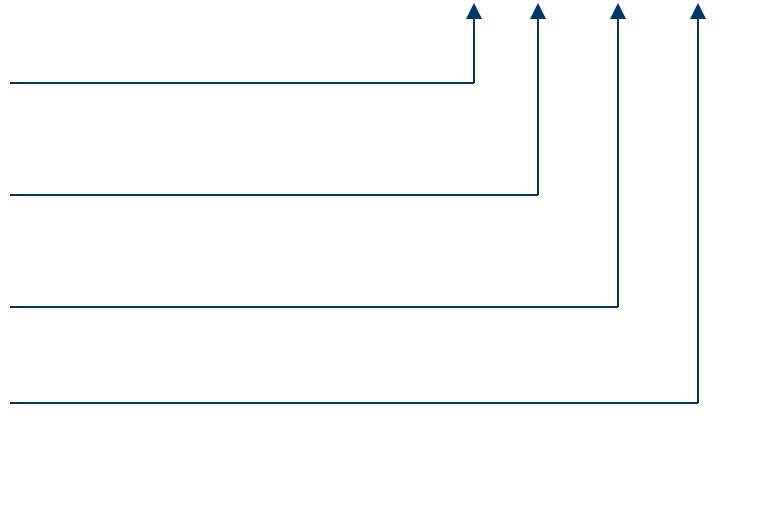
所接受的语言不变，带来额外的设计便利

带 ϵ -转移的非确定有限自动机

◆ 带 ϵ -转移的非确定有限自动机 (ϵ -NFA) 的形式定义

一个 ϵ -NFA 是一个五元组 $A = (Q, \Sigma, \delta, q_0, F)$.

- 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
- 与 NFA 的不同之处



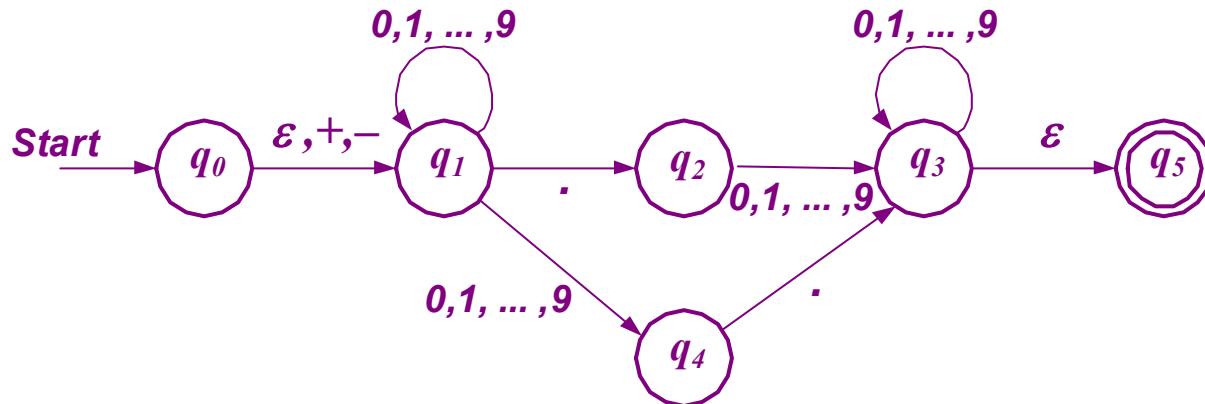
$$q_0 \in Q$$

$$F \subseteq Q$$

$$\delta: Q \times (\Sigma \cup \{ \epsilon \}) \rightarrow 2^Q$$

带 ϵ -转移的非确定有限自动机

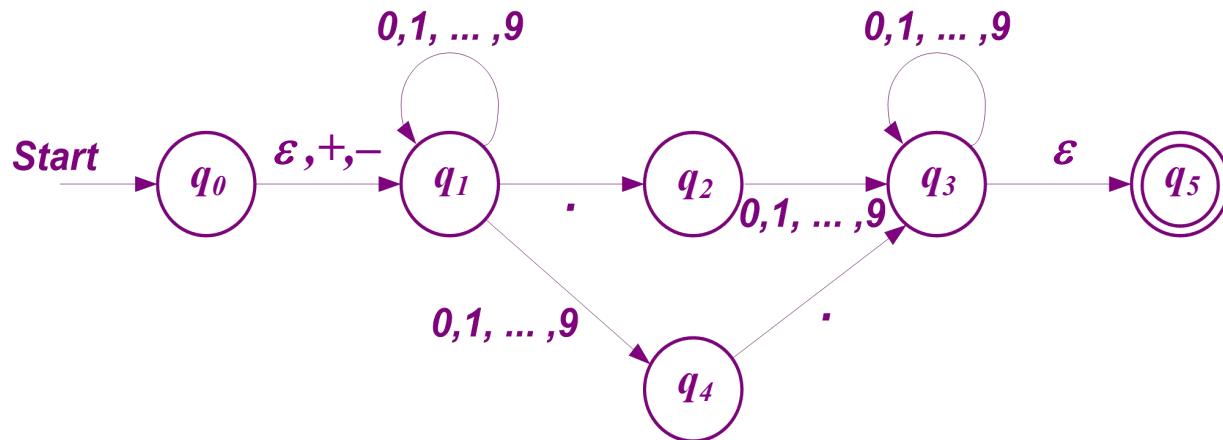
◇ 转移图和转移表表示的 ϵ -NFA



	ϵ	$+, -$.	$0, 1, \dots, 9$
$\rightarrow q_0$	$\{q_1\}$	$\{q_1\}$	\emptyset	\emptyset
q_1	\emptyset	\emptyset	$\{q_2\}$	$\{q_1, q_4\}$
q_2	\emptyset	\emptyset	\emptyset	$\{q_3\}$
q_3	$\{q_5\}$	\emptyset	\emptyset	$\{q_3\}$
q_4	\emptyset	\emptyset	$\{q_3\}$	\emptyset
$* q_5$	\emptyset	\emptyset	\emptyset	\emptyset

带 ϵ -转移的非确定有限自动机

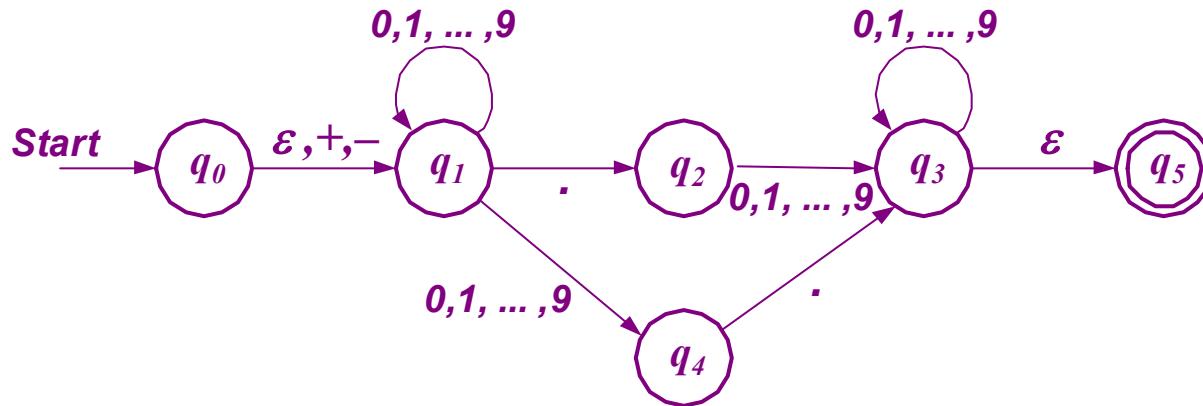
✧ ϵ -NFA 如何接受输入符号串



- 该 ϵ -NFA 可以接受的字符串如：
 - 3.14
 - +.314
 - – 314.

带 ϵ -转移的非确定有限自动机

✧ ϵ -闭包 (closure)

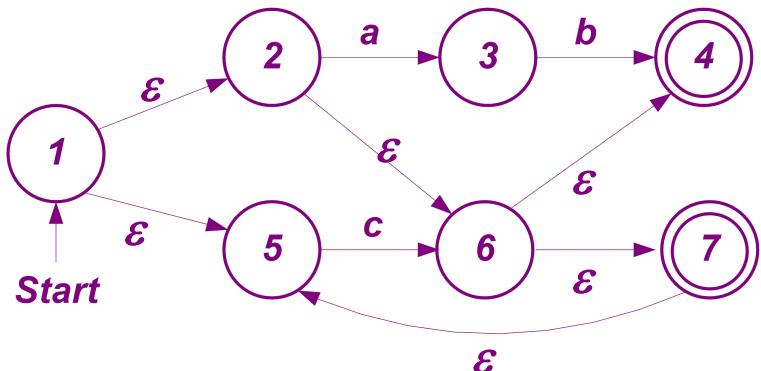


- 状态 q 的 ϵ -闭包，记为 $ECLOSE(q)$ ，定义为从 q 经所有的 ϵ 路径可以到达的状态（包括 q 自身），如：
 - $ECLOSE(q_0) = \{q_0, q_1\}$
 - $ECLOSE(q_2) = \{q_2\}$
 - $ECLOSE(q_3) = \{q_3, q_5\}$

带 ϵ -转移的非确定有限自动机

◆ ϵ -闭包

- 设 ϵ -NFA $A = (Q, \Sigma, \delta, q_0, F)$, $q \in Q$, $ECLOSE(q)$ 为满足如下条件的最小集:
 - (1) $q \in ECLOSE(q)$
 - (2) if $p \in ECLOSE(q)$ and $r \in \delta(p, \epsilon)$, then $r \in ECLOSE(q)$
- 对于右图, 有:
 - $ECLOSE(1) = \{1, 2, 4, 5, 6, 7\}$
 - $ECLOSE(2) = \{2, 4, 5, 6, 7\}$
 - $ECLOSE(7) = \{5, 7\}$



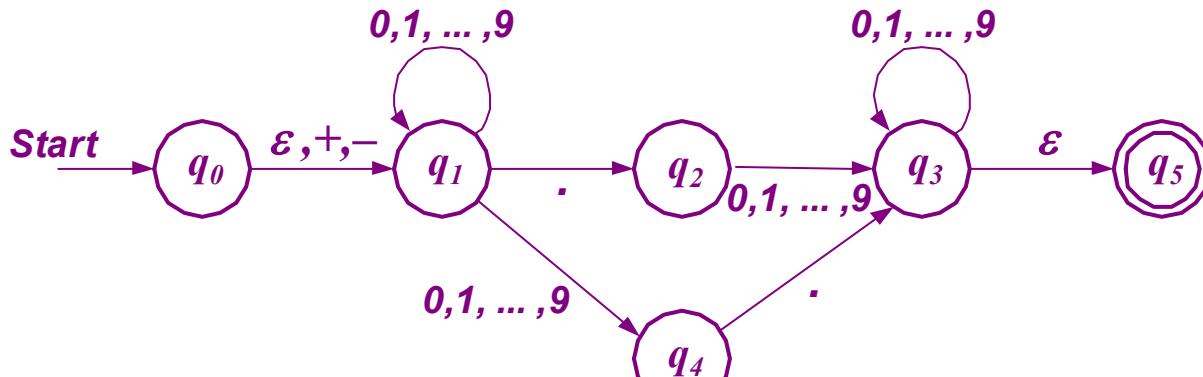
带 ϵ -转移的非确定有限自动机

◆ 扩展转移函数适合于输入字符串

- 设一个 ϵ -NFA $E = (Q, \Sigma, \delta, q_0, F)$
- $\delta: Q \times \Sigma \cup \{ \epsilon \} \rightarrow 2^Q$
- 扩充定义 $\delta': Q \times \Sigma^* \rightarrow 2^Q$
- 对任何 $q \in Q$, 定义:
 - 1 $\delta'(q, \epsilon) = ECLOSE(q)$
 - 2 若 $w = xa$, 其中 $x \in \Sigma^*$, $a \in \Sigma$, 假设
 $\delta'(q, x) = \{ p_1, p_2, \dots, p_k \}$, 并且
 令 $\bigcup_{i=1}^k \delta(p_i, a) = \{ r_1, r_2, \dots, r_m \}$, 则
 $\delta'(q, w) = \bigcup_{j=1}^m ECLOSE(r_j)$

带 ϵ -转移的非确定有限自动机

◆ 扩展转移函数适合于输入字符串



– 举例 计算 $\delta'(q_0, 5.6)$

- $\delta'(q_0, \epsilon) = ECLOSE(q_0) = \{q_0, q_1\}$
- $\delta(q_0, 5) \cup \delta(q_1, 5) = \{q_1, q_4\}$
 $\delta'(q_0, 5) = ECLOSE(q_1) \cup ECLOSE(q_4) = \{q_1, q_4\}$
- $\delta(q_1, .) \cup \delta(q_4, .) = \{q_2, q_3\}$
 $\delta'(q_0, 5.) = ECLOSE(q_2) \cup ECLOSE(q_3) = \{q_2, q_3, q_5\}$
- $\delta(q_2, 6) \cup \delta(q_3, 6) \cup \delta(q_5, 6) = \{q_3\}$
 $\delta'(q_0, 5.6) = ECLOSE(q_3) = \{q_3, q_5\}$

带 ε -转移的非确定有限自动机

✧ ε -NFA 的语言

- 设一个 ε -NFA $E = (Q, \Sigma, \delta, q_0, F)$
- 定义 E 的语言：
$$L(E) = \{ w \mid w \in \Sigma^* \wedge \delta'(q_0, w) \cap F \neq \emptyset \}$$
- 设 L 是 Σ 上的语言，如果存在一个 ε -NFA $E = (Q, \Sigma, \delta, q_0, F)$ ，满足 $L = L(E)$ ，则可以证明 L 也是一个正规语言。

带 ϵ -转移的非确定有限自动机

✧ ϵ -NFA 与 DFA 的等价性

定理: L 是某个 ϵ -NFA 的语言, 当且仅当 L 也是某个 DFA 的语言.

证明: 分两步证明.

(1) 设 L 是某个 DFA D 的语言, 则存在一个 ϵ -NFA E , 满足 $L(E) = L(D) = L$;

(2) 设 L 是某个 ϵ -NFA E 的语言, 则存在一个 DFA D , 满足 $L(D) = L(E) = L$;

带 ε -转移的非确定有限自动机

✧ 从 **DFA** 构造等价的 ε - **NFA**

– 设 L 是某个 **DFA** $D = (Q, \Sigma, \delta_D, q_0, F)$ 的语言,
则存在一个 ε - **NFA** E , 满足 $L(E) = L(D) = L$.

– 证明: 定义 $E = (Q, \Sigma, \delta_E, q_0, F)$, 其中 δ_E 定义为

- 对任何 $q \in Q$, $\delta_E(q, \varepsilon) = \phi$
- 对任何 $q \in Q$ 和 $a \in \Sigma$,
若 $\delta_D(q, a) = p$, 则 $\delta_E(q, a) = \{p\}$.

需要证明: 对任何 $w \in \Sigma^*$,

$$\delta'_D(q_0, w) = p \text{ iff } \delta'_E(q_0, w) = \{p\}.$$

归纳于 $|w|$ 易证上述命题.

带 ε -转移的非确定有限自动机

◆ 从 ε -NFA 构造等价的 DFA (修改的子集构造法)

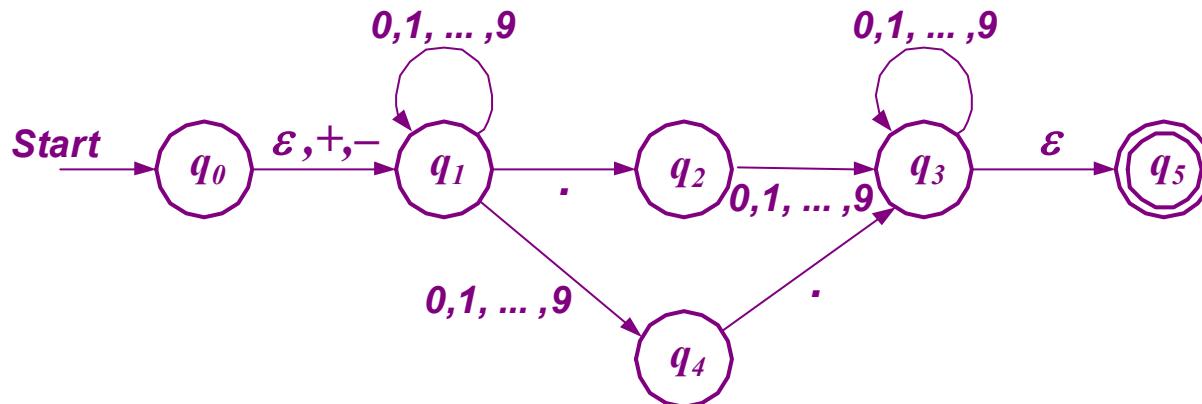
- 设 L 是某个 ε -NFA $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ 的语言, 则存在一个 DFA D , 满足 $L(D) = L(E) = L$.
- 证明: 定义 $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$, 其中
 - $Q_D = \{ S \mid S \subseteq Q_E \wedge S = ECLOSE(S) \}$
 - $q_D = ECLOSE(q_0)$
 - $F_D = \{ S \mid S \in Q_D \wedge S \cap F_E \neq \emptyset \}$
 - 对 $S \in Q_D$ 和 $a \in \Sigma$, 令 $S = \{ p_1, p_2, \dots, p_k \}$, 并设 $\bigcup_{i=1}^k \delta_E(p_i, a) = \{ r_1, r_2, \dots, r_m \}$, 则

$$\delta_D(S, a) = \bigcup_{j=1}^m ECLOSE(r_j).$$

需要证明: 对任何 $w \in \Sigma^*$, $\delta'_D(q_D, w) = \delta'_E(q_0, w)$.
归纳于 $|w|$ 可证上述命题.

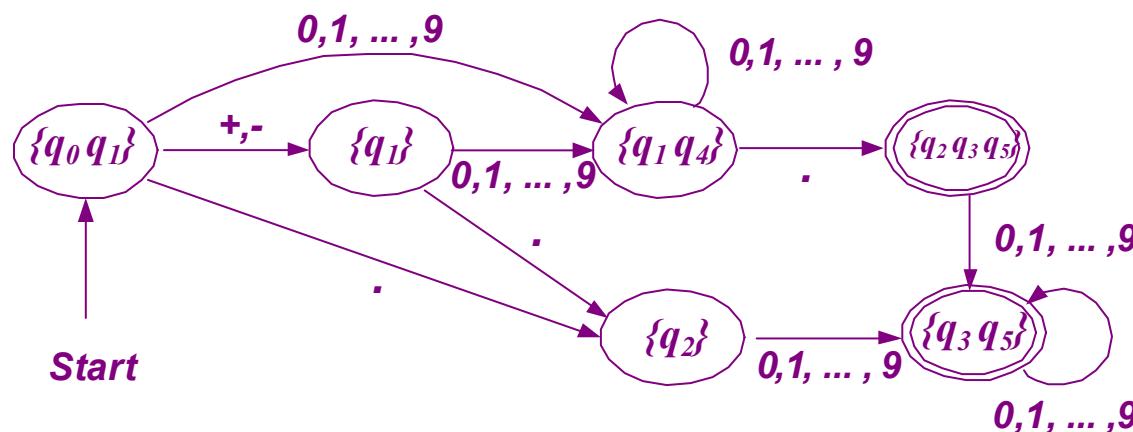
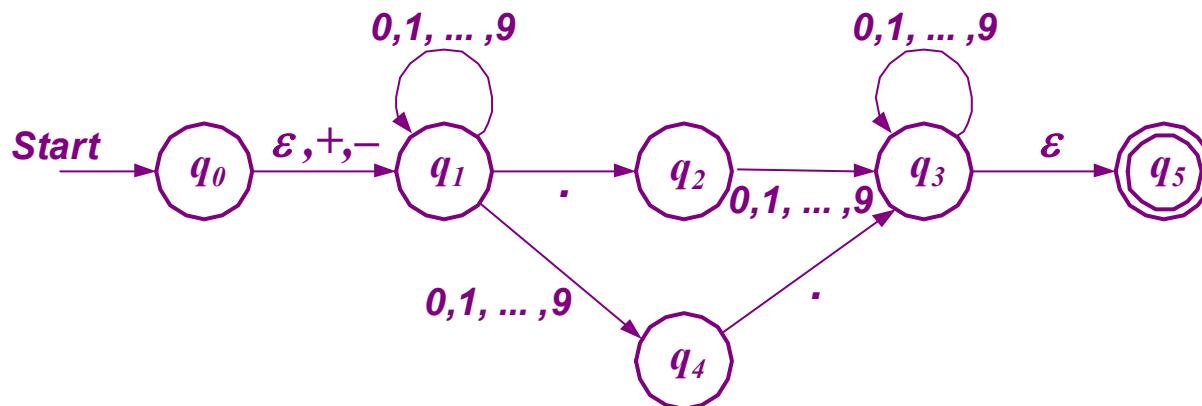
带 ϵ -转移的非确定有限自动机

✧ 修改的子集构造法举例



带 ϵ -转移的非确定有限自动机

修改的子集构造法举例



带 ε -转移的非确定有限自动机

✧ 从 ε -NFA 构造等价的 DFA (修改的子集构造法)

- 设 $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ 是一个 ε -NFA, 通过修改的子集构造法得到相应的DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$, 则
对任何 $w \in \Sigma^*$, $\delta'_D(q_D, w) = \delta'_E(q_0, w)$.
- 证明: 归纳于 $|w|$

1 设 $|w| = 0$, 即 $w = \varepsilon$.

由定义知 $\delta'_D(q_D, \varepsilon) = q_D = ECLOSE(q_0) = \delta'_E(q_0, \varepsilon)$.

2 设 $|w| = n+1$, 并 $w = xa$, $a \in \Sigma$. 注意到 $|x| = n$.

假设 $\delta'_D(q_D, x) = \delta'_E(q_0, x) = \{p_1, p_2, \dots, p_k\}$.

并设 $\bigcup_{i=1}^k \delta_E(p_i, a) = \{r_1, r_2, \dots, r_m\}$.

则 $\delta'_D(q_D, w) = \delta_D(\{p_1, p_2, \dots, p_k\}, a)$
 $= \bigcup_{j=1}^m ECLOSE(r_j) = \delta'_E(q_0, w)$

带 ϵ -转移的非确定有限自动机

例15：考虑下面的 ϵ -NFA

	ϵ	a	b	c
$\rightarrow p$	\emptyset	$\{p\}$	$\{q\}$	$\{r\}$
q	$\{p\}$	$\{q\}$	$\{r\}$	\emptyset
$*r$	$\{q\}$	$\{r\}$	\emptyset	$\{p\}$

1. 计算每个状态的 ϵ 闭包。
2. 把这个自动机转换为DFA。

解：首先根据定义，很容易计算每个状态的空闭包

$$\text{ECLOSE}(p) = \{p\}, \quad \text{ECLOSE}(q) = \{p, q\}, \quad \text{ECLOSE}(r) = \{p, q, r\}$$

然后可以在空闭包的基础上使用子集构造法。

带 ϵ -转移的非确定有限自动机

例16：考虑下面的 ϵ -NFA

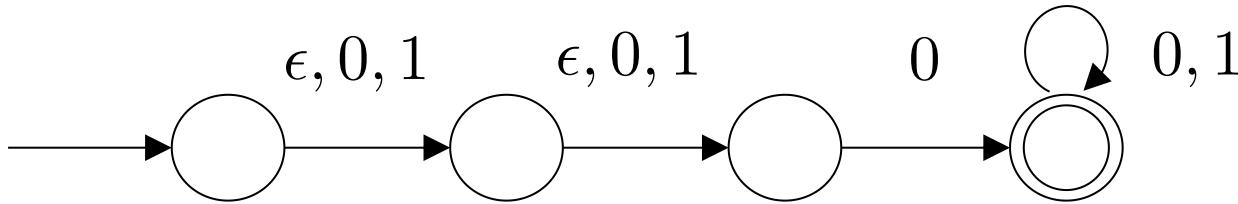
	ϵ	a	b	c
$\rightarrow p$	\emptyset	$\{p\}$	$\{q\}$	$\{r\}$
q	$\{p\}$	$\{q\}$	$\{r\}$	\emptyset
$*r$	$\{q\}$	$\{r\}$	\emptyset	$\{p\}$

- 计算每个状态的 ϵ 闭包。
- 把这个自动机转换为 DFA。

解：

	a	b	c
$\rightarrow \{p\}$	$\{p\}$	$\{p, q\}$	$\{p, q, r\}$
$\{p, q\}$	$\{p, q\}$	$\{p, q, r\}$	$\{p, q, r\}$
$*\{p, q, r\}$	$\{p, q, r\}$	$\{p, q, r\}$	$\{p, q, r\}$

例17：设计 ϵ -NFA 接受所有由0和1组成的串，要求长度至少为1，且前三个符号里面至少有一个0。

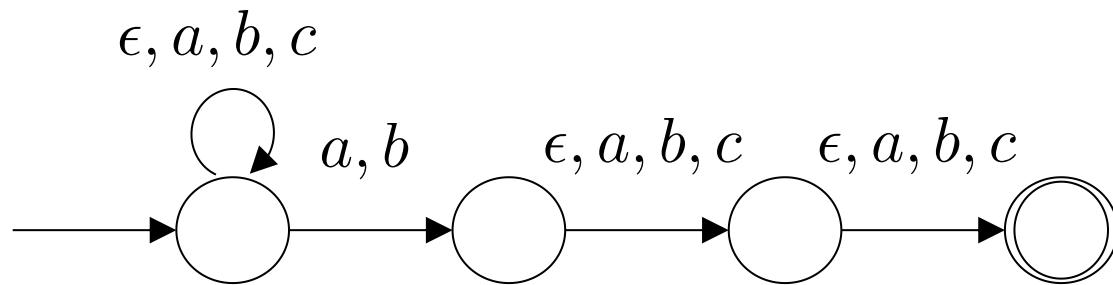


解： ϵ -NFA 特别适合处理这种情况。类似于之前设计正规式的经验，允许前两个符号出现空符号，自然就实现了题目的要求。如果用DFA或者NFA的话，设计起来难度会更大。

DFA 和 NFA 的等价性

例18：设计 ϵ -NFA 接受以下语言

$$L = \{w \mid w \in \{a, b, c\}^* \wedge |w| \geq 1 \wedge w \text{后3位中至少有1位不是} c\}$$

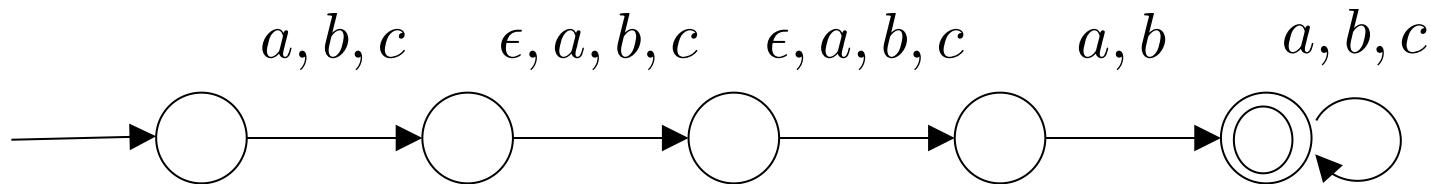


解：题目等价于“后3位至少有1位是a或b”，这样就把看似很难处理的“不是c”转化为我们非常熟悉的形式，很容易给出相应的 ϵ -NFA。

DFA 和 NFA 的等价性

例19：设计 ϵ -NFA 接受以下语言

$$L = \{w \mid w \in \{a, b, c\}^* \wedge |w| \geq 2 \wedge w \text{ 从第2位至第4位至少有一位不是 } c\}$$



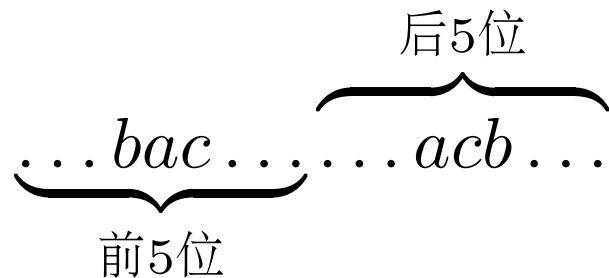
解：掌握了例17的技巧后，这道题就很简单了，注意加上偏移量。

DFA 和 NFA 的等价性

例20：设计 ϵ -NFA 接受以下语言

$$L = \{w \mid w \in \{a, b, c\}^* \wedge |w| \geq 4 \wedge \\ w \text{的前5位至少有一个子串 } bac \wedge \\ w \text{的后5位至少有一个子串 } acb\}$$

解：这道题的难点在于需要充分考虑所有的情况。第1种情况是 bac 出现在 acb 的前面并且没有重叠。如



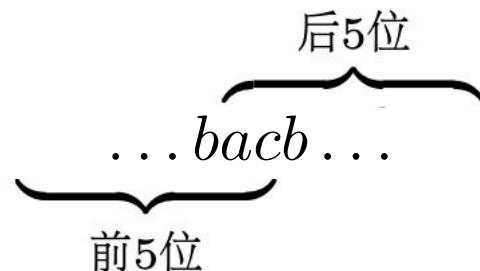
这是最容易想到的情况，但还有更多的情况需要考虑。

DFA 和 NFA 的等价性

例20：设计 ϵ -NFA 接受以下语言

$$L = \{w \mid w \in \{a, b, c\}^* \wedge |w| \geq 4 \wedge \\ w \text{的前5位至少有一个子串 } bac \wedge \\ w \text{的后5位至少有一个子串 } acb\}$$

解：由于两个子串有公共子串 ac ，因此第2种情况是 bac 在 acb 的前面并且两者交叠。



不仅如此，更有第3种情况，也是最容易漏掉的情况。

DFA 和 NFA 的等价性

例20：设计 ϵ -NFA 接受以下语言

$$L = \{w \mid w \in \{a, b, c\}^* \wedge |w| \geq 4 \wedge \\ w \text{的前5位至少有一个子串 } bac \wedge \\ w \text{的后5位至少有一个子串 } acb\}$$

解：第3种情况是 bac 在 acb 的后面，两者有公共子串 b ，整个串的长度为5。



这三种情况是或的关系，需要体现在 ϵ -NFA 中。

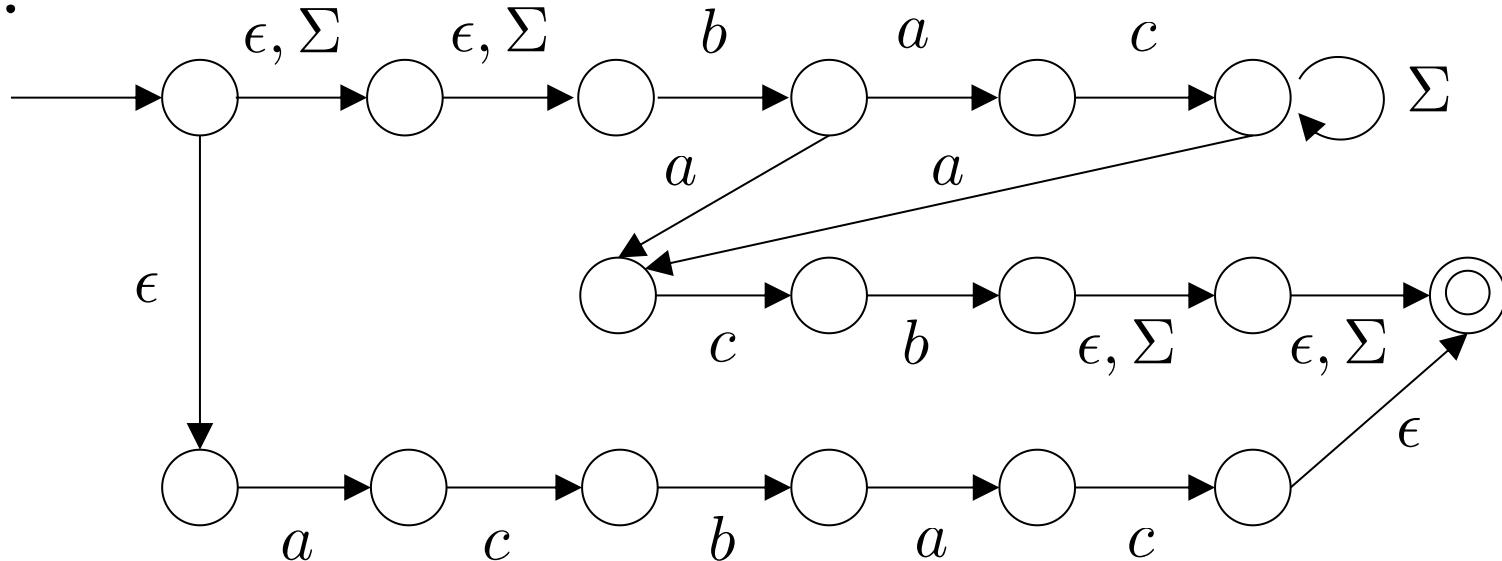
DFA 和 NFA 的等价性

例20：设计 ϵ -NFA 接受以下语言

$$L = \{w \mid w \in \{a, b, c\}^* \wedge |w| \geq 4 \wedge$$

w的前5位至少有一个子串bac \wedge
w的后5位至少有一个子串acb}

解：



- ✧ 知识回顾：集合上的等价关系与集合的划分
- ✧ **DFA** 状态集合上的一个等价关系
- ✧ 计算状态集划分的算法——填表法
- ✧ 最小化的 **DFA**

◆ 知识回顾：集合上的等价关系与集合的划分

- 等价关系

设 Q 为一个集合，二元关系 R 是 Q 上的一个等价关系，当且仅当满足以下条件：

1. **自反性** 对任何 $a \in Q$, aRa 成立；
2. **对称性** 对任何 $a, b \in Q$, 如果 aRb 成立，则有 bRa 成立；
3. **传递性** 对任何 $a, b, c \in Q$, 如果 aRb 和 bRc 成立，则有 aRc 成立.

◆ 知识回顾：集合上的等价关系与集合的划分

设 $A = \{1, 2, \dots, 8\}$, 定义 A 上的关系 R 如下：

$$xRy \Leftrightarrow \forall x, y \in A, x \equiv y \pmod{3}$$

其中, $x \equiv y \pmod{3}$ 表示 x 对 3 取余数与 y 对 3 取余数相等。
可以验证等价关系的三个属性:

1. 自反性。 $1R1$ 是显然成立的。
2. 对称性。 $4R7 \Rightarrow 7R4$ 。
3. 传递性。 $1R4 \wedge 4R7 \Rightarrow 1R7$ 。

为此, 我们说 R 是 A 上的一个等价关系。

◆ 知识回顾：集合上的等价关系与集合的划分

- 等价关系与划分

设 Q 为一个集合, R 是 Q 上的一个等价关系, 由 R 产生的所有等价类(或块)的集合构成 Q 的一个划分.

- 解释

1. 等价类 对任何 $a \in Q$, a 所在的块用 $[a]$ 表示, 定义为

$$[a] = \{x \mid xRa\} ;$$

2. 每一元素都属于唯一的块 即满足

(1) $\cup_{a \in Q} [a] = Q$; 和

(2) 对任何 $a, b \in Q$, 或者 $[a]=[b]$, 或者 $[a] \cap [b]=\emptyset$

◆ 知识回顾：集合上的等价关系与集合的划分

设 $A = \{1, 2, \dots, 8\}$, 定义 A 上的关系 R 如下：

$$xRy \Leftrightarrow \forall x, y \in A, x \equiv y \pmod{3}$$

该等价关系产生了如下等价类：

1. $[1] = \{1, 4, 7\}$
2. $[2] = \{2, 5, 8\}$
3. $[3] = \{3, 6\}$

✧ DFA 状态集合上的一个等价关系

- 设一个 DFA $D = (Q, \Sigma, \delta, q_0, F)$, 定义 Q 上的一个二元关系 R 为: 对任何 $p, q \in Q$,
$$pRq \text{ iff } \forall w \in \Sigma^*. (\delta'(p, w) \in F \leftrightarrow \delta'(q, w) \in F)$$
- 结论 上述关系 R 是等价关系.

证明:

1. 自反性 对任何 $q \in Q$, qRq 成立;
2. 对称性 对任何 $p, q \in Q$, $pRq \rightarrow qRp$ 成立;
3. 传递性 对任何 $p, q, r \in Q$, 设 pRq 和 qRr 成立,
即对任何 $w \in \Sigma^*$, $\delta'(p, w) \in F \leftrightarrow \delta'(q, w) \in F$ 和
 $\delta'(q, w) \in F \leftrightarrow \delta'(r, w) \in F$ 成立; 由此, 也有
 $\delta'(p, w) \in F \leftrightarrow \delta'(r, w) \in F$ 成立. 所以, qRr 成立

✧ DFA 状态集合上的一个等价关系

- 若 pRq , 称 p 和 q 等价 (*equivalent*) . 若 p 和 q 不等价, 则称 p 和 q 是可区别的 (*distinguishable*) .
- 关系 R 对应有限状态集 Q 的一个划分;
该划分的每个块是 Q 的一个子集;
同一划分块中的所有状态之间都是相互等价的;
分属不同划分块的任何两个状态之间都是可区别的.

✧ DFA 的优化

通过合并等价的（或不可区别的）状态
关键：如何计算上述划分？

◆ 有关可区别性的几个有用的结果

- 设状态 r 和 s 通过某个输入符号 a 可分别转移到 p 和 q (即 $\delta(r,a)=p, \delta(s,a)=q$) , 则有
 p 和 q 可区别 $\Rightarrow r$ 和 s 可区别

这是因为: 若 p 和 q 可为字符串 w 区别, 则 r 和 s 可为字符串 aw 区别.

$$(\because \delta'(r,aw) = \delta'(p,w), \delta'(s,aw) = \delta'(q,w))$$

◆ 有关可区别性的几个有用的结果

- 设状态 r 和 s 通过某个输入符号 a 可分别转移到 p 和 q (即 $\delta(r,a)=p, \delta(s,a)=q$) , 则有
 r 和 s 不可区别 $\Rightarrow p$ 和 q 不可区别

(前页结果的逆否)

✧ 有关可区别性的几个有用的结果

- 设状态 r 和 s 通过某个输入符号 a 可分别转移到 p 和 q (即 $\delta(r,a)=p, \delta(s,a)=q$) , 则有
 r 和 s 可由 ax 区别 $\Rightarrow p$ 和 q 可由 x 区别

$$(\because \delta'(r,ax) = \delta'(p,x), \delta'(s,ax) = \delta'(q,x))$$

✧ 计算状态集划分的算法—填表法

- 填表算法 (*table-filling algorithm*) 基于如下递归地标记可区别的状态偶对的过程:

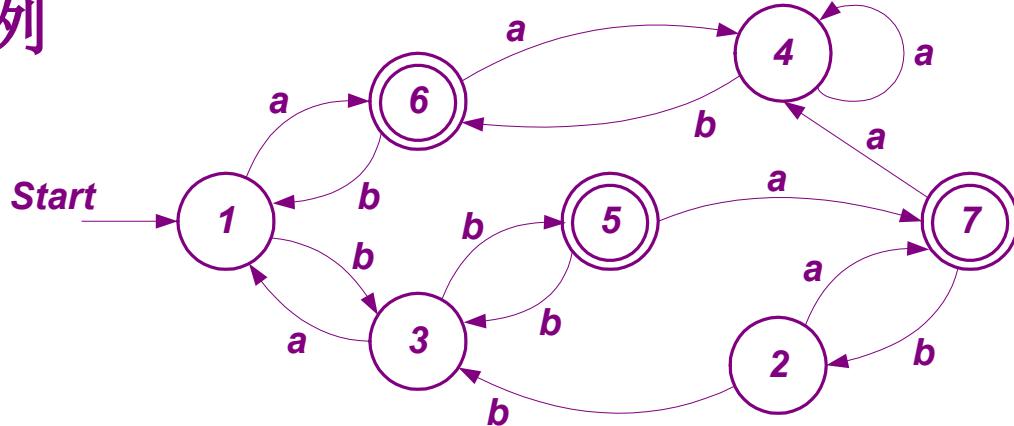
基础 如果 p 为终态, 而 q 为非终态, 则 p 和 q 标记为可区别的;

归纳 设 p 和 q 已标记为可区别的, 如果状态 r 和 s 通过某个输入符号 a 可分别转移到 p 和 q ,
 $\delta(r,a)=p$, $\delta(s,a)=q$, 则 r 和 s 也标记为可区别的;

◆ 计算状态集划分的算法—填表法

— 填表算法举例

	1	2	3	4	5	6
2						
3	X	X				
4	X	X	X			
5	X	X	X	X		
6	X	X	X	X	X	
7	X	X	X	X	X	



- (1) 区别所有终态和非终态
- (2) 区别(1,3), (1,4), (2,3), (2,4), (5,6), (5,7)
- (3) 区别 (3,4)

(4) 结束. 划分结果: {1,2}, {3}, {4}, {5}, {6,7}

◇ 计算状态集划分的算法—填表法

- 填表算法的正确性 还需证明：如果两个状态没有被填表算法标记，则这两个状态一定是等价的
- 证明 反证法. 假定状态 r 和 s 没有被填表算法标记，但这两个状态不是等价的，即是可区别的.

设字符串 w 可用于区别状态 r 和 s , 即 $\delta'(r, w)$ 和 $\delta'(s, w)$ 两个状态中，一个是终态，一个是非终态. 不妨设前者为终态，后者为非终态.

首先不可能有 $w=\varepsilon$, 否则, 状态 r 为终态, 而 s 为非终态, 依填表算法, r 和 s 第一步就被标记.

设 $w=ax$, 并且 $\delta(r, a)=p$, $\delta(s, a)=q$, 则 p 和 q 可被 x 区别. 但同样 p 和 q 不可能被填表算法标记(否则, r 和 s 将被标记). 同样也有, $x \neq \varepsilon$.

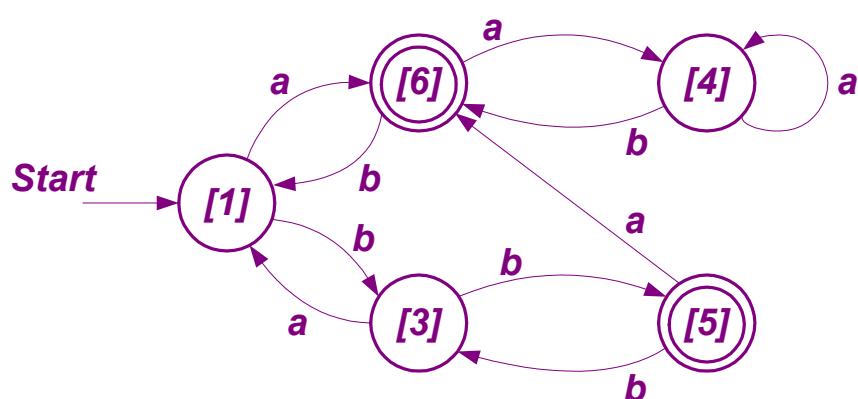
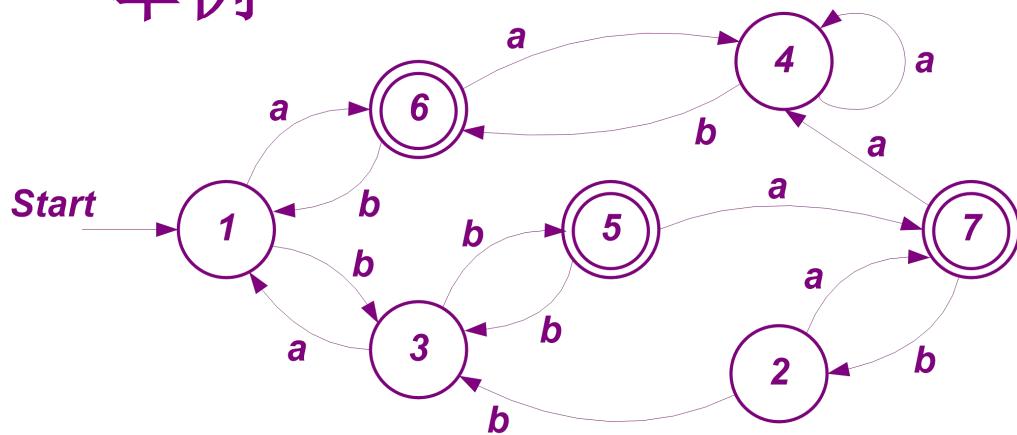
该过程不可能一直下去, 终将产生矛盾.

◆ 通过合并等价的状态进行 *DFA* 的优化

— 步骤

1. 删除所有从开始状态不可到达的状态及与其相关的边, 设所得到的 *DFA* 为 $A = (Q, \Sigma, \delta, q_0, F)$;
 2. 使用填表算法找出所有等价的状态偶对;
 3. 根据 2 的结果计算当前状态集合的划分块, 每一划分块中的状态相互之间等价, 而不同划分块中的状态之间都是可区别的. 包含状态 q 的划分块用 $[q]$ 表示.
 4. 构造与 A 等价的 *DFA* $B = (Q_B, \Sigma, \delta_B, [q_0], F_B)$, 其中 $Q_B = \{[q] \mid q \in Q\}$, $F_B = \{[q] \mid q \in F\}$, $\delta_B([q], a) = [\delta(q, a)]$
- 结论: 对任何 $w \in \Sigma^*$, $\delta'_B([q_0], w) \in F_B$, iff $\delta'(q_0, w) \in F$

◇ 通过合并等价的状态进行 *DFA* 的优化
— 举例



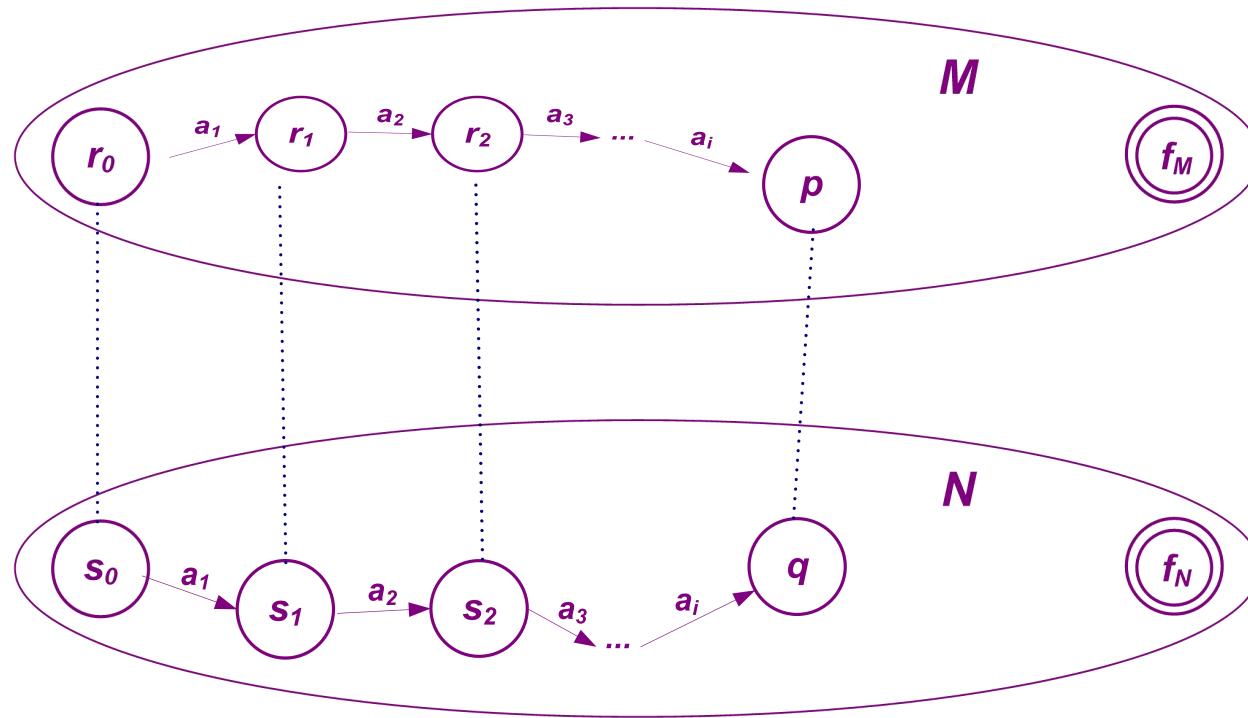
- 等价的状态偶对为:
 $(1, 2)$, $(6, 7)$
- 划分结果:
 $\{1, 2\}, \{3\}, \{4\}, \{5\}, \{6, 7\}$
- 新的状态集合:
 $[1], [3], [4], [5], [6]$

◆ 最小化的 **DFA**

- 问题 假定一个 **DFA** 为 A , 用上述优化步骤构造出与 A 等价的 **DFA** M ; 那么是否存在一个状态数目比 M 还少的 **DFA** N , 它接受的语言同 A 和 M 完全一样?

假设存在一个这样 **DFA** N . 现将 M 和 N 相并, 即状态、转移规则都相并, 这里假定 M 和 N 之间没有重名的状态, 因而也没有相交的转移边, 原来的终态还是终态, 原来的两个初态中任选一个作为新的初态. 同时还假定 M 和 N 的每一状态都是从其相应的初态可以到达的, 否则我们将去掉不可达状态, 得到状态数目更小的 **DFA**.

✧ 最小化的 DFA



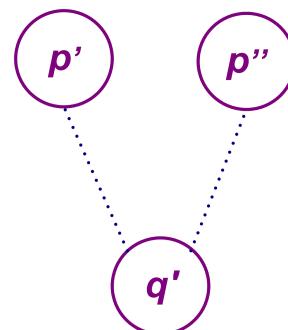
✧ 最小化的 DFA

对 M 和 N 相并后的 DFA 运用填表算法可以得出：

1. M 和 N 的初态是不可区别的, 因为 $L(M)=L(N)$;
2. 若 r 和 s 是不可区别的, 则对于任何输入符号, r 和 s 的后继状态之间也是不可区别的;
3. M 的任一状态至少与 N 的一个状态是不可区别的.

根据假设, N 的状态数目比 M 少, 所以 M 中必然存在两个状态, 它们分别与 N 中的同一个状态不可区别.

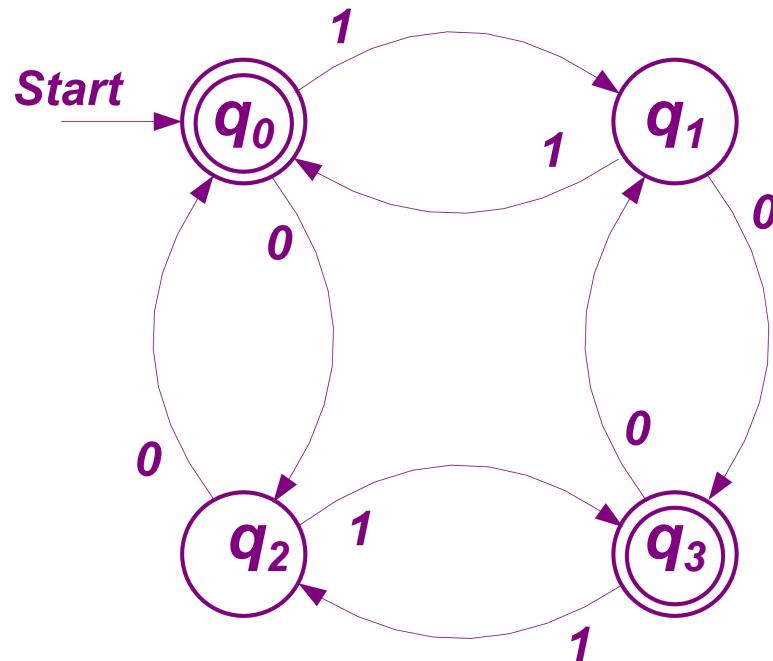
根据不可区别关系的传递性, M 的这两个状态是不可区别的, 这与 M 的构造过程矛盾.



- 结论 对任何 DFA A , 用前述优化步骤构造出与 A 等价的 DFA M ; 那么 M 的状态数目不多于任何语言为 $L(A)$ 的 DFA

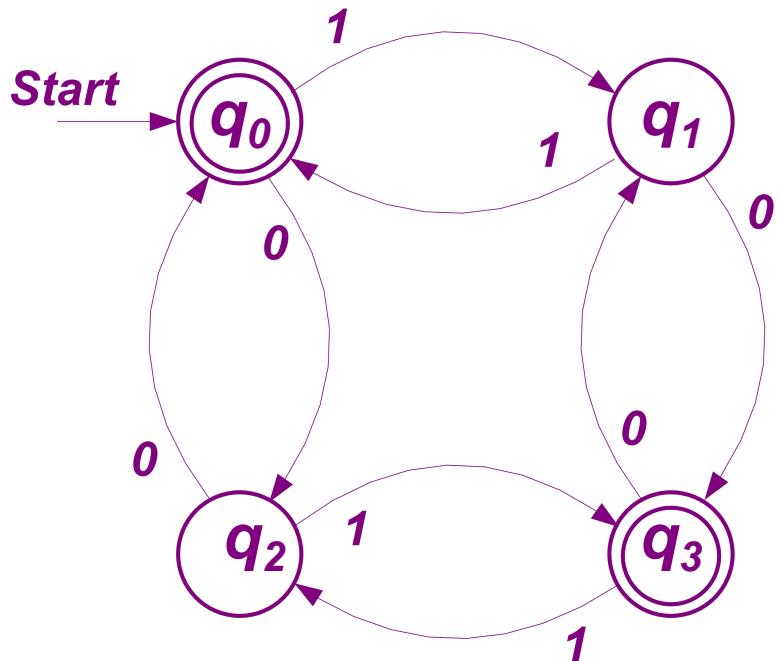
✧ 课堂练习

- 最小化下图表示的 DFA



✧ 课堂练习

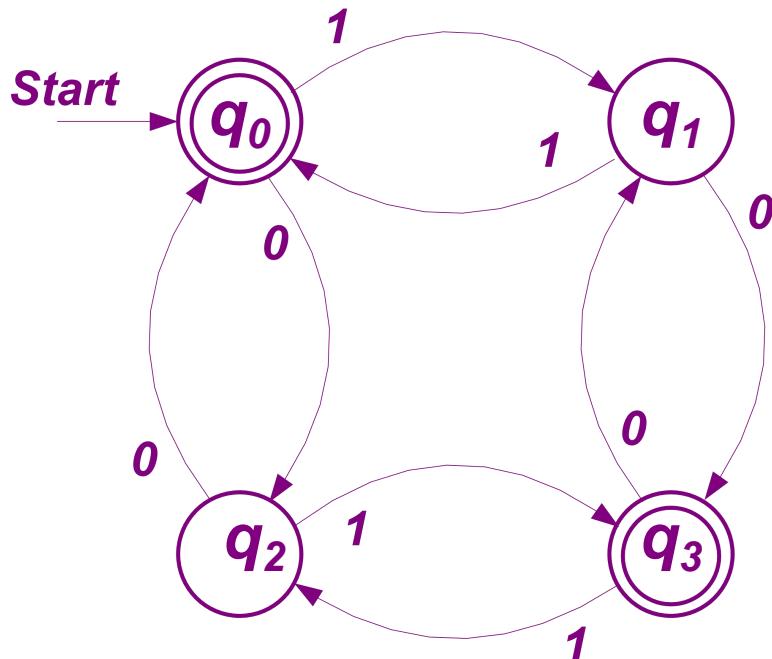
- 最小化下图表示的 DFA



1			
2			
3			
	0	1	2

✧ 课堂练习

- 最小化下图表示的 DFA

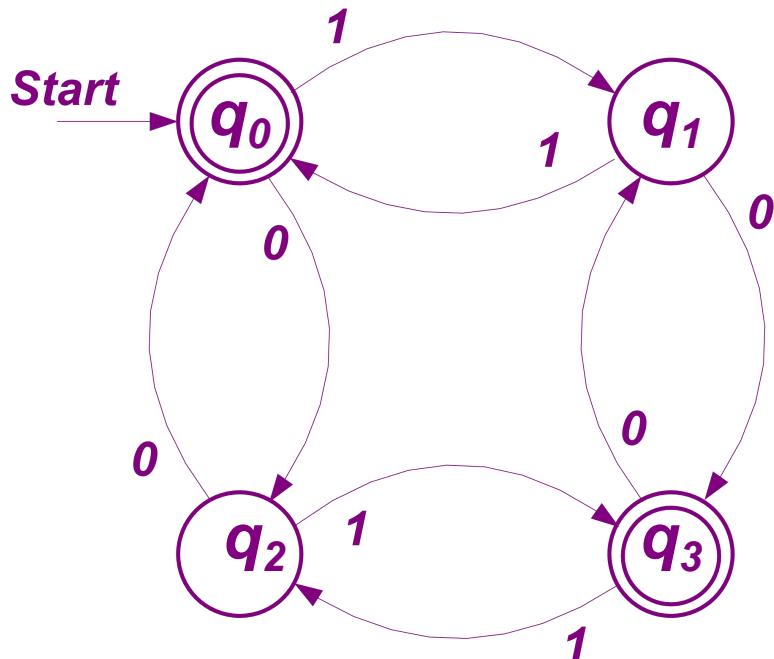


1	X		
2			
3			
	0	1	2

0是终态，1是非终态

✧ 课堂练习

- 最小化下图表示的 DFA

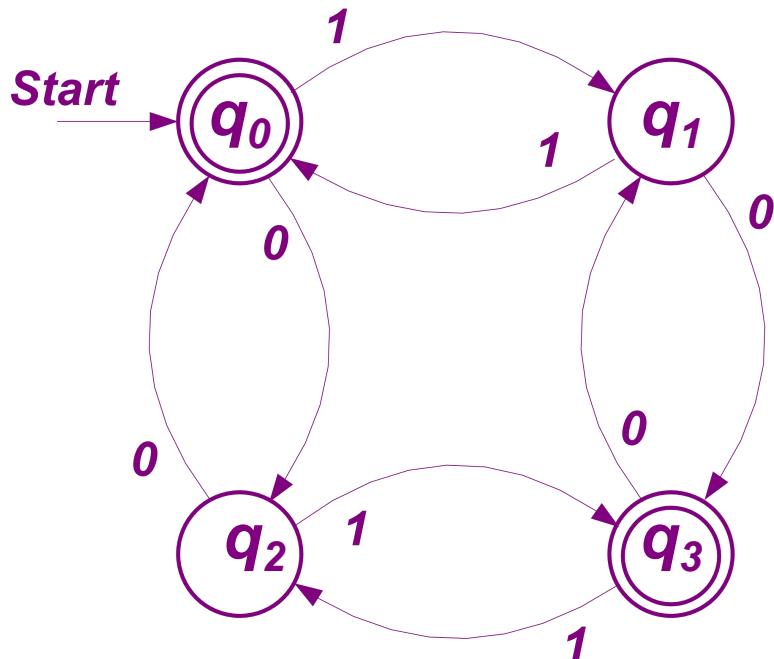


1	X		
2	X		
3			
	0	1	2

0是终态，2是非终态

✧ 课堂练习

- 最小化下图表示的 DFA

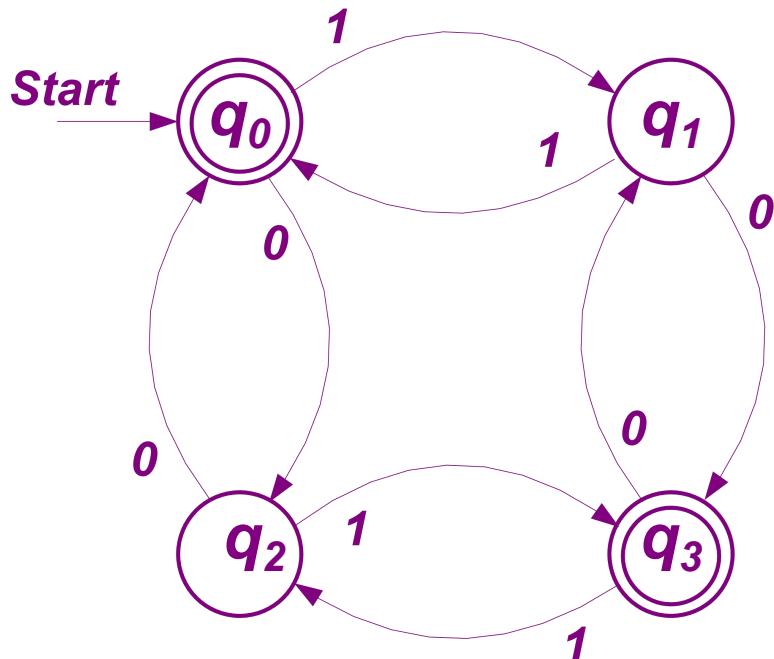


1	X		
2	X		
3		X	
	0	1	2

3是终态，1是非终态

✧ 课堂练习

- 最小化下图表示的 DFA

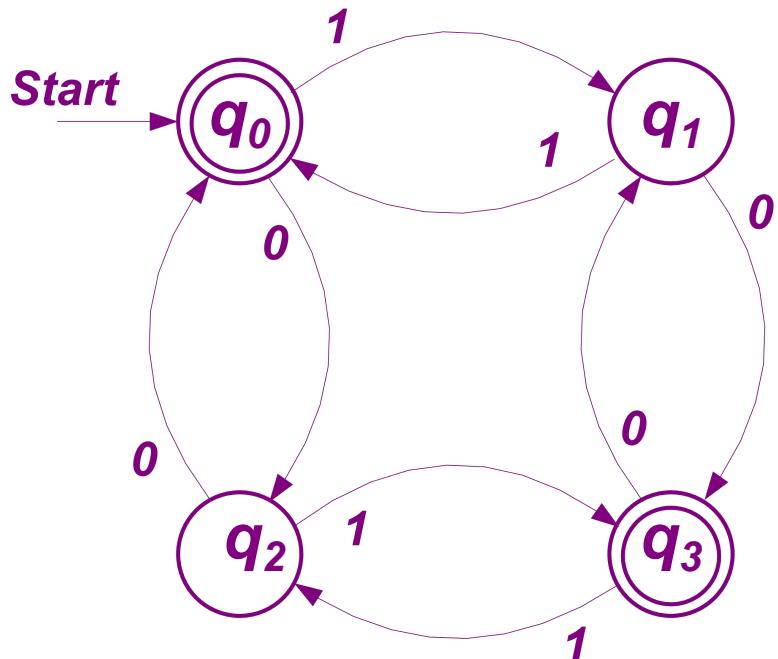


1	X		
2	X		
3		X	X
	0	1	2

3是终态，2是非终态

✧ 课堂练习

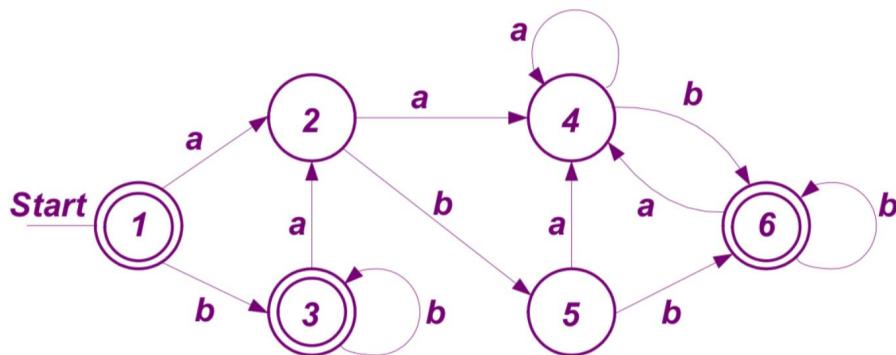
- 最小化下图表示的 DFA



1	X		
2	X		
3		X	X
	0	1	2

因此，0和3等价，1和2等价。

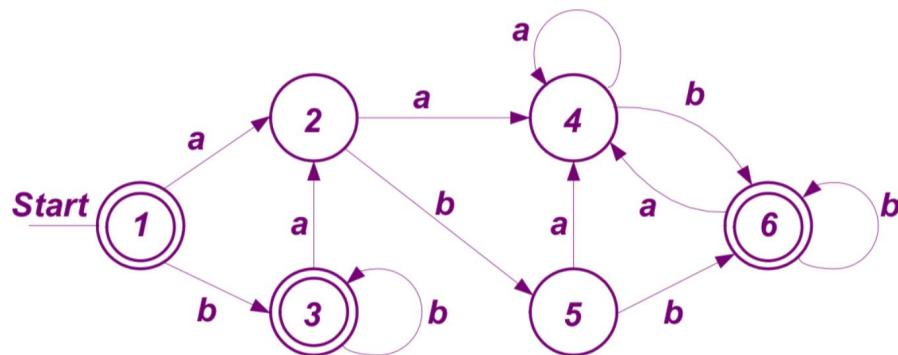
例21：最小化以下DFA



2	X				
3		X			
4	X		X		
5	X		X		
6		X		X	X
	1	2	3	4	5

首先先区分终态和非终态。

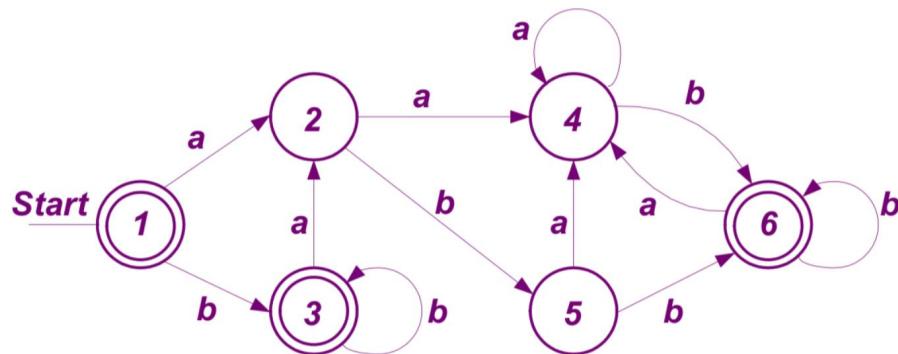
例21：最小化以下DFA



2	X				
3	O	X			
4	X		X		
5	X		X		
6		X		X	X
	1	2	3	4	5

考察状态1和状态3，由于它们接受a都到达状态2，接受b都到达状态3，因此状态1和状态3不可区分。

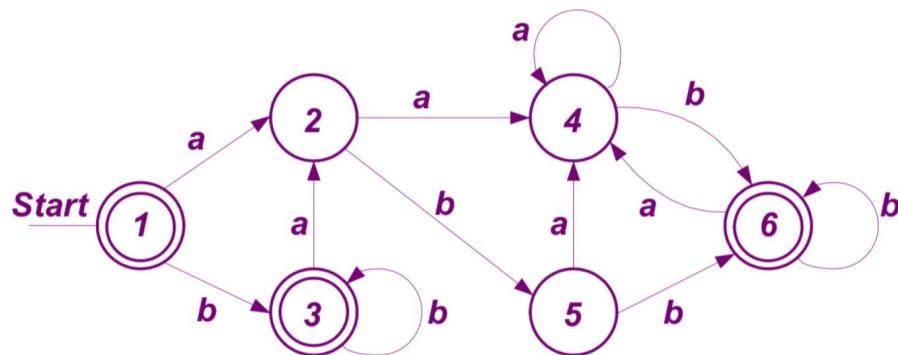
例21：最小化以下DFA



2	X				
3	O	X			
4	X		X		
5	X		X		
6	?	X		X	X
	1	2	3	4	5

考察状态1和状态6，由于它们的后继的等价关系目前还不清楚，所以暂时搁置，等后面处理完了再看。

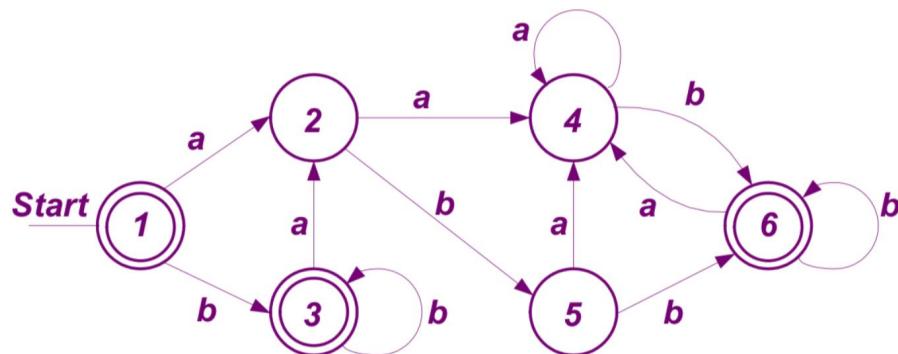
例21：最小化以下DFA



2	X				
3	O	X			
4	X	X	X		
5	X		X		
6	?	X		X	X
	1	2	3	4	5

考察状态2和状态4，因为状态2经过b到达状态5，状态4经过b到达状态6，由于状态5和状态6可区分，因此状态2和状态4可区分。

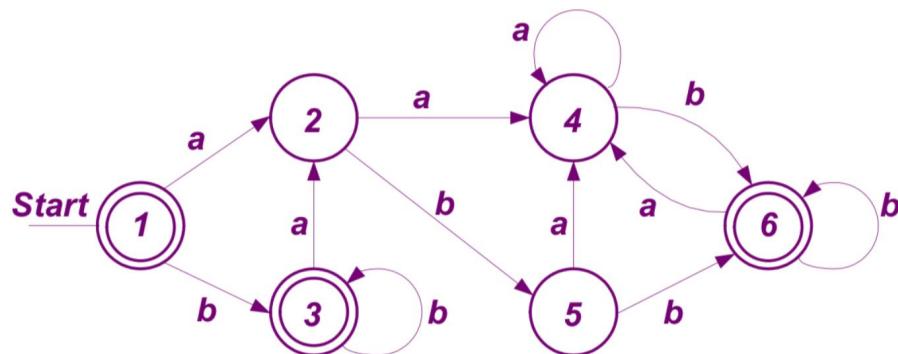
例21：最小化以下DFA



2	X				
3	O	X			
4	X	X	X		
5	X	X	X		
6	?	X		X	X
	1	2	3	4	5

考察状态2和状态5，因为状态2经过b到达状态5，状态5经过b到达状态6，由于状态5和状态6是可区分的，因此状态2和状态5也是可区分的。

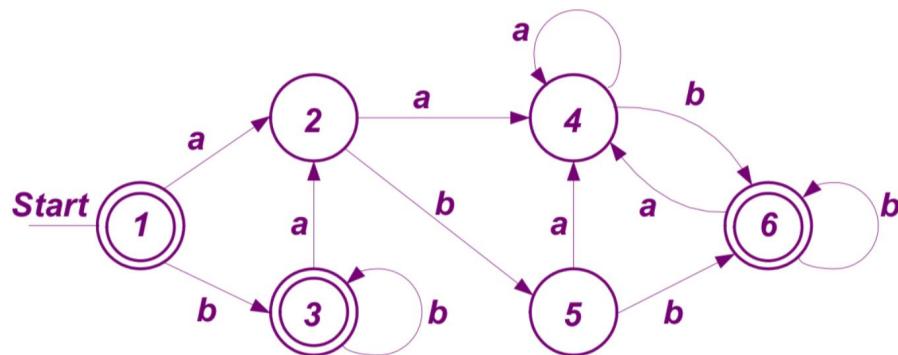
例21：最小化以下DFA



2	X				
3	O	X			
4	X	X	X		
5	X	X	X		
6	?	X	X	X	X
	1	2	3	4	5

考察状态3和状态6，因为状态3经过a到达状态2，状态6经过a达到状态4，由于状态2和状态4是可区分的，因此状态3和状态6也是可区分的。

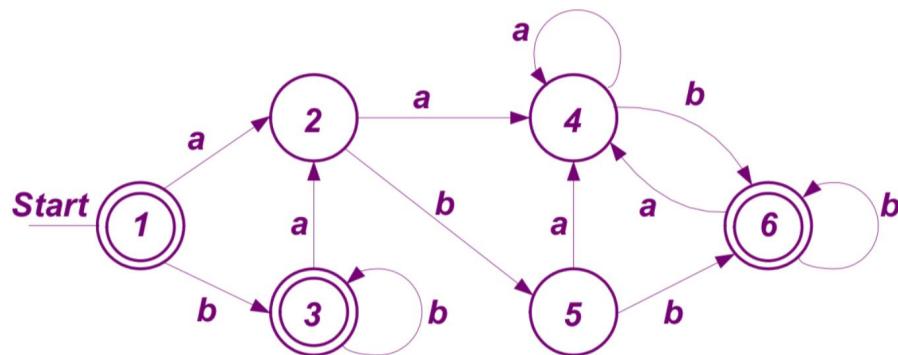
例21：最小化以下DFA



2	X				
3	O	X			
4	X	X	X		
5	X	X	X	O	
6	?	X	X	X	X
	1	2	3	4	5

考察状态4和状态5，因为状态4经过a到达状态4，状态5经过a到达状态4，状态4经过b到达状态6，状态5经过b也到达状态6，因此状态4和状态5不可区分。

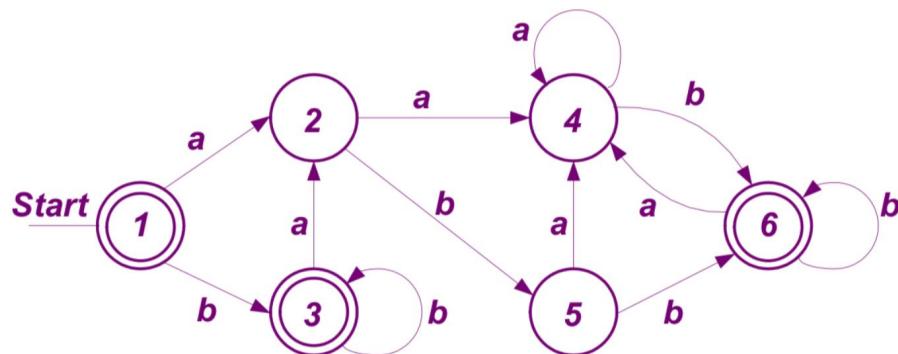
例21：最小化以下DFA



2	X				
3	O	X			
4	X	X	X		
5	X	X	X	O	
6	X	X	X	X	X
	1	2	3	4	5

最后再考察状态1和状态6，状态1经过a到达状态2，状态6经过a到达状态4，由于状态2和状态4是可区分的，因此状态1和状态6也是可区分的。

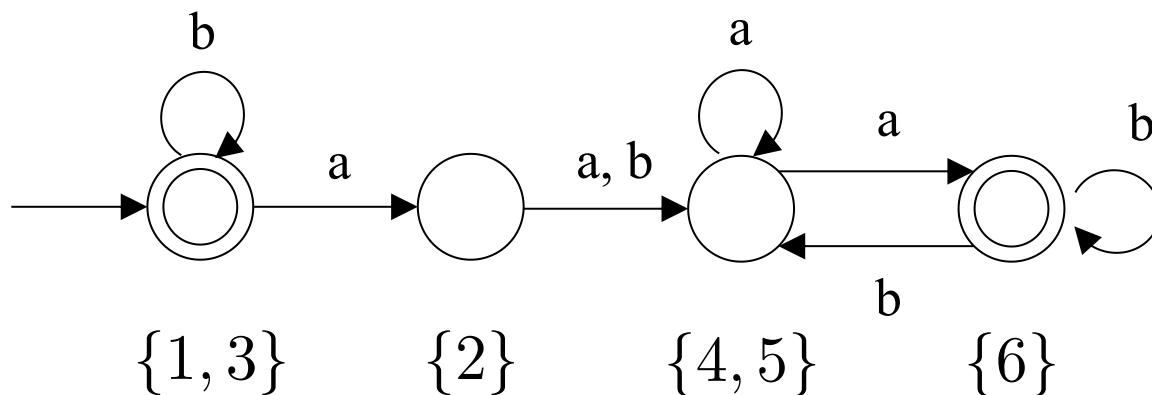
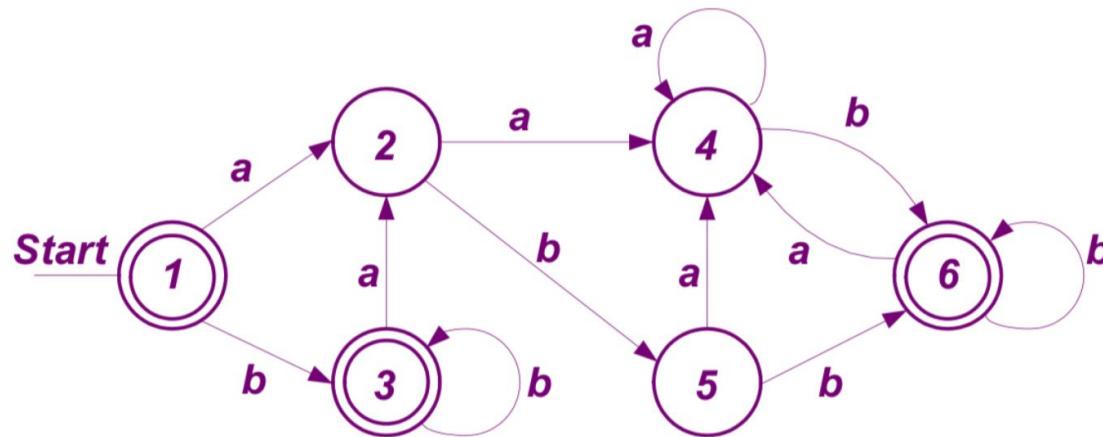
例21：最小化以下DFA



2	X				
3		X			
4	X	X	X		
5	X	X	X		
6	X	X	X	X	X
	1	2	3	4	5

因此，状态1和状态3等价，状态4和状态5等价。

例21：最小化以下DFA



例22：最小化以下DFA

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

B							
C							
D							
E							
F							
G							
H							
	A	B	C	D	E	F	G

首先建立起表格，注意行和列的元素设置。

例22：最小化以下DFA

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

B							
C		X					
D	X	X	X				
E				X			
F				X			
G				X			
H				X			
	A	B	C	D	E	F	G

接下来的技巧是根据可区分状态对倒推，例如(A, D)，看看哪些状态经过同样的符号能达到(A, D)。

例22：最小化以下DFA

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

B							
C		X					
D	X	X	X				
E				X			
F				X			
G				X			
H				X			
	A	B	C	D	E	F	G

接下来的技巧是根据可区分状态对倒推，例如(A, D)，看看哪些状态经过同样的符号能达到(A, D)。

例22：最小化以下DFA

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

B							
C		X					
D	X	X	X				
E		X		X			
F				X			
G				X			
H				X			
	A	B	C	D	E	F	G

接下来的技巧是根据可区分状态对倒推，例如(A, D)，看看哪些状态经过同样的符号能达到(A, D)。

例22：最小化以下DFA

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

B							
C		X					
D	X	X	X				
E		X		X			
F				X			
G				X			
H	X			X			
	A	B	C	D	E	F	G

接下来的技巧是根据可区分状态对倒推，例如(A, D)，看看哪些状态经过同样的符号能达到(A, D)。

(确定) 有限自动机的最小化

FL&A



例22：最小化以下DFA

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

B								
C		X						
D	X	X	X					
E		X			X			
F					X			
G					X			
H	X				X			
	A	B	C	D	E	F	G	

接下来的技巧是根据可区分状态对倒推，例如(A, D)，看看哪些状态经过同样的符号能达到(A, D)。

例22：最小化以下DFA

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

B							
C	X	X					
D	X	X	X				
E		X		X			
F					X		
G					X		
H	X			X			
	A	B	C	D	E	F	G

再考察可区分的状态对(B, D), 进行倒推。

例22：最小化以下DFA

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

B							
C	X	X					
D	X	X	X				
E	X	X			X		
F					X		
G					X		
H	X				X		
	A	B	C	D	E	F	G

再考察可区分的状态对(B, D), 进行倒推。

例22：最小化以下DFA

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

B	X						
C	X	X					
D	X	X	X				
E	X	X		X			
F	X		X	X	X		
G		X	X	X	X	X	
H	X	X	X	X	X	X	X
	A	B	C	D	E	F	G

依次类推，可以得到完整的表格。基本的技巧还是在左侧表格用倒推法，需要注意一下右侧表格单元的遍历顺序。

例22：最小化以下DFA

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
$*D$	D	A
E	D	F
F	G	E
G	F	G
H	G	D

	0	1
$\rightarrow AG$	BF	AG
BF	AG	CE
CE	D	BF
$*D$	D	AG
H	AG	D

根据等价类划分，可以得到新的DFA。由于状态H不可达，可以删去。

例22：最小化以下DFA

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
$*D$	D	A
E	D	F
F	G	E
G	F	G
H	G	D

	0	1
$\rightarrow AG$	BF	AG
BF	AG	CE
CE	D	BF
$*D$	D	AG

这是最终的结果。事实上，从一开始左表中E、F、G、H就都不可达，可以直接删去，只是为了演示填表法。

课后练习

✧ 必做题:

- *Ex.2.2.2
- Ex.2.2.4 (b),(c)
- Ex.2.2.5 (d)
- Ex.2.2.7
- Ex.2.2.9
- Ex.2.3.2
- Ex.2.3.4 (b),(c)

- Ex.2.4.2 (c) (请依所介绍的算法做)
- Ex.2.5.2
- Ex.2.5.3 (a), ! (b)
- Ex.4.4.2
- !Ex.2.5.3 (c)

✧ 思考题:

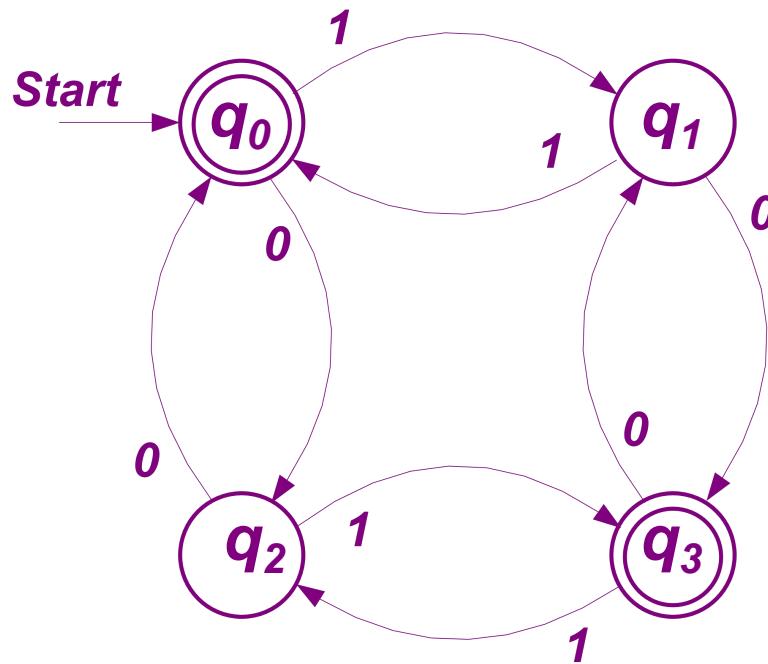
- Ex.2.2.5 (b)
- Ex.2.2.6 *(a) (b)

课后练习

✧ 思考题:

– 附加

Let $L = \{ w \mid \text{In } w, \text{ the number of 0's and the number of 1's are the same in parity} \}$ be a language over $\Sigma = \{0, 1\}$. Prove that L is the language of following DFA.



课后练习

✧ 自测题：

- 设计一个 DFA，其语言为长度至少为2且头两个字符不相同的0, 1串构成的集合。
- 试构造接受下列语言的一个 DFA:
 - 1) $\{w \in \{a, b\}^* \mid w \text{ 中不包含子串 } aa\}$
 - 2) $\{w \mid w \in \{a, b\}^*, w \text{ 中包含且仅包含奇数个子串 } ab\}$
 - 3) $\{w \mid w \in \{a, b\}^*, \text{ 且 } w \text{ 中 } a \text{ 的个数和 } b \text{ 的个数之和是奇数}\}$
 - 4) $\{w \mid w \in \{a, b\}^*, w \text{ 中 } a \text{ 的个数是偶数, 且 } w \text{ 的长度也为偶数}\}$
 - 5) $\{w \mid w \in \{a, b\}^*, w \text{ 含相同个数的 } a \text{ 和 } b, \text{ 且 } w \text{ 的每个前缀中 } a \text{ 和 } b \text{ 个数之差不超过 } 1\}$
 - 6) $\{w \mid w \in \{a, b\}^*, w \text{ 包含子串 } ab, \text{ 但不包含子串 } bb\}$
- 设计一个NFA，其语言为长度至少为2且末尾两个字符不相同的0, 1串构成的集合。
- 试给出下列正规语言 L 的一个 NFA (不是 ε -NFA) :

$$\{w \mid w \in \{a, b\}^*, |w| \geq 3, \text{ 且 } w \text{ 中第2位和第3位不同}\}$$

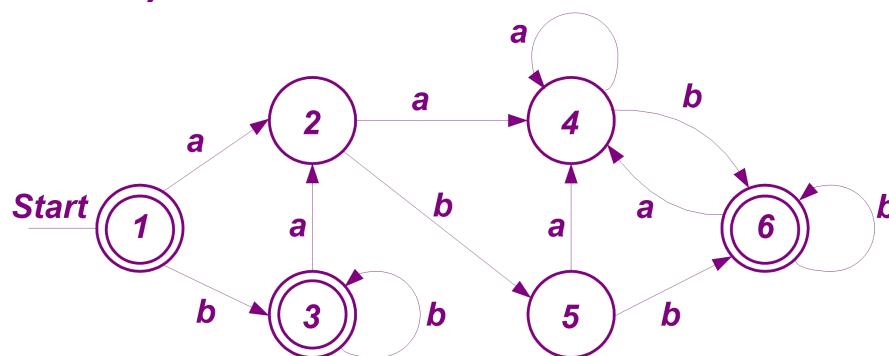
课后练习

✧ 自测题：

- 设计一个 ϵ -NFA，其语言由满足下述条件的0, 1串构成：长度至少为1，且前三个符号中至少有一个“0”。（以状态转移图的形式给出，要能够体现 ϵ -NFA的设计特点）。
- 试构造接受下列正规语言 L 的一个 ϵ -NFA：
 - 1) $\{ w \mid w \in \{a, b, c\}^*, |w| \geq 1, \text{且 } w \text{ 后 3 位中至少有一位不是 } c \}$
 - 2) $\{ w \mid w \in \{a, b, c\}^*, |w| \geq 2, \text{且 } w \text{ 中从第 2 位到第4位至少有一位不是 } c \}$
 - 3) $\{ w \mid w \in \{a, b, c\}^*, |w| \geq 4, \text{且 } w \text{ 的前5位至少有一个子串 bac, 且 } w \text{ 的后5位至少有一个子串 acb } \}$

✧ 自测题：

- 左下图表示一个 DFA，构造出与该DFA等价的最小化的 DFA（即拥有的状态数目最少）。(分主要步骤或直接写出结果均可)



That's all for today.

Thank You