第 3 章　有 穷 状 态 自 动 机

3.1 确定性有穷自动机

- Finite automata(FA/DFA)
  - EXP: Automatic door of supermarket, Digital Watch, elevator, Markov Chain etc.

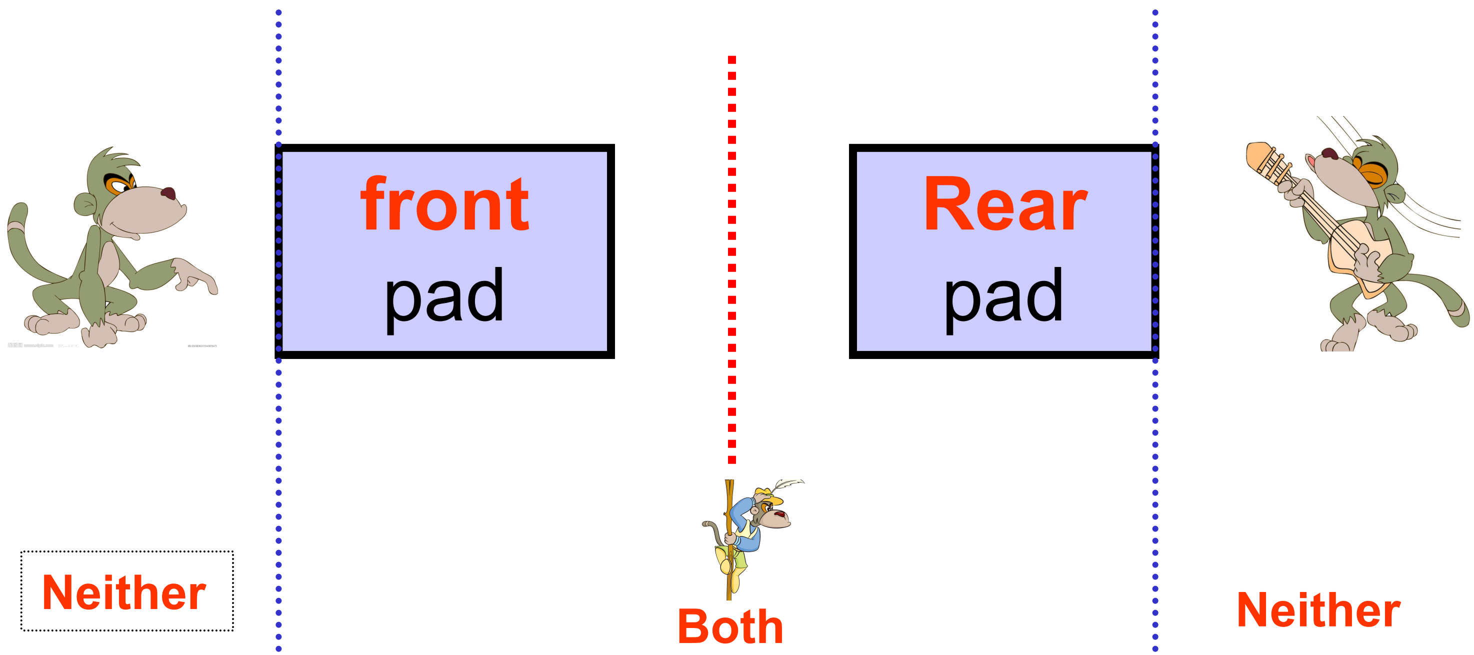


图 3.1 自动门示意图

有穷自动机的表示

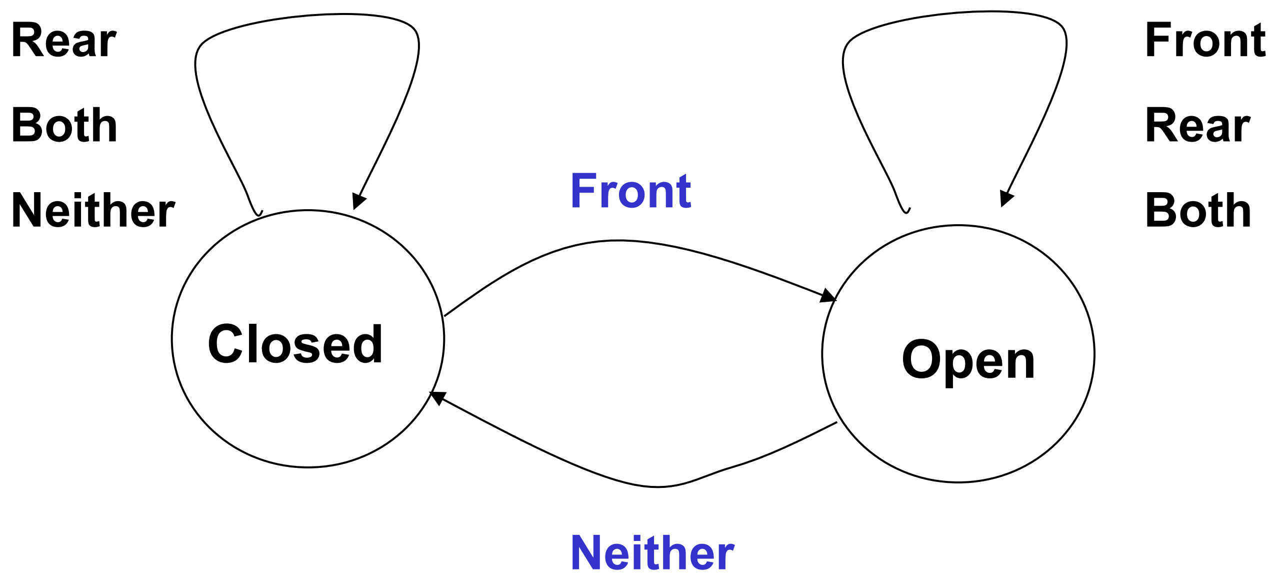


图 3.2 有穷自动机的转移图表示

	Neither	Front	Rear	Both
Closed	Closed	Open	Closed	Closed
Open	Closed	Open	Open	Open

图 3.3 有穷自动机的矩阵表示

有穷自动机的表示

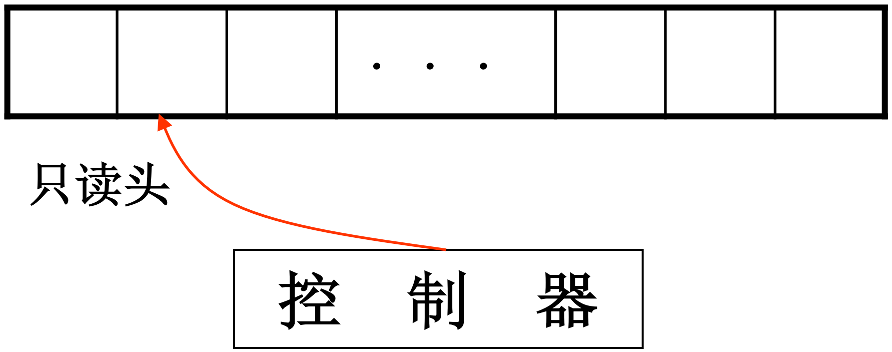


图 3.4 有穷自动机的模型

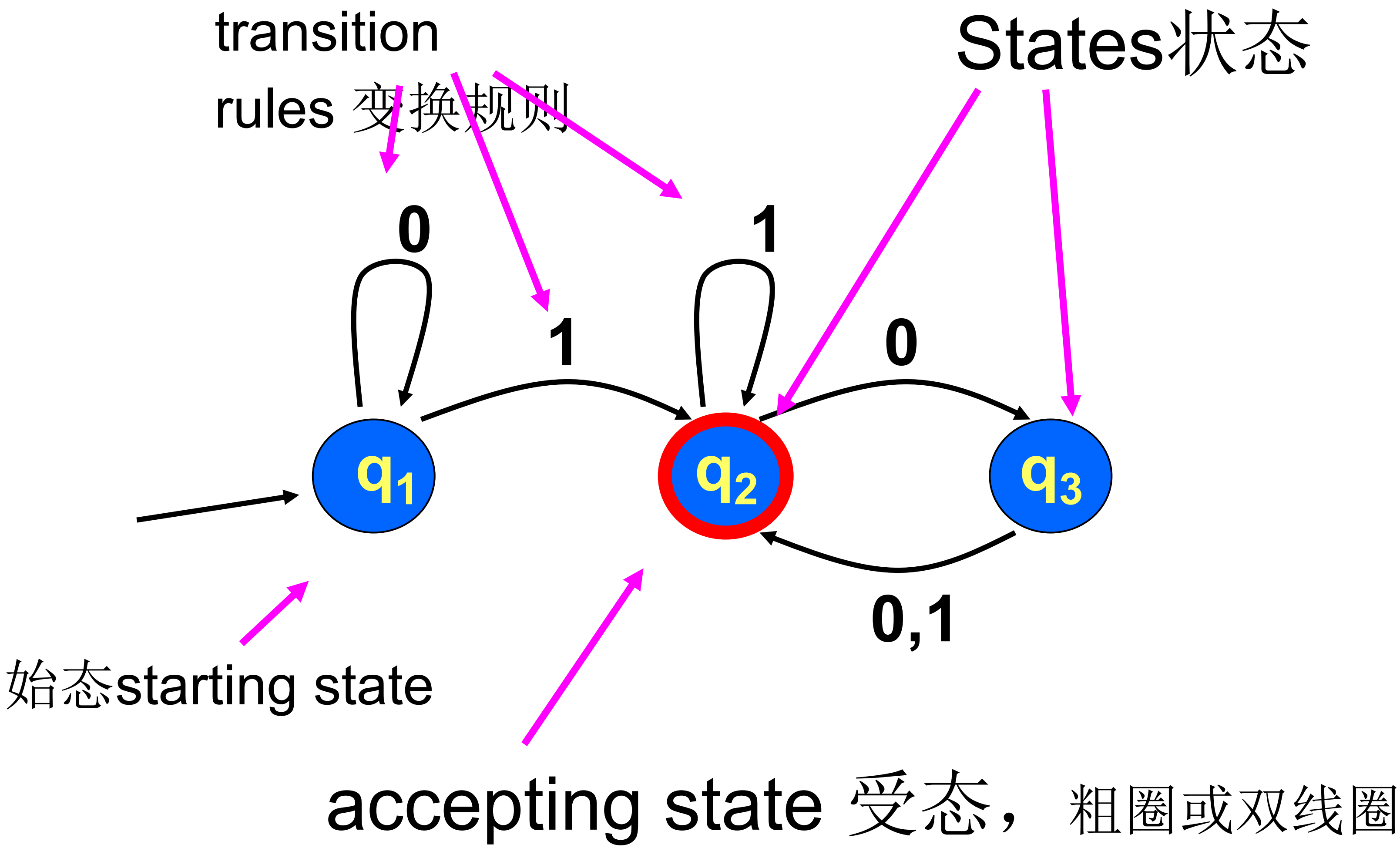
FA的特点：

- 只有一条磁带，用于存放字符串
- 带头只能读，不能写
- 带头只能往一个方向移动（从左往右）

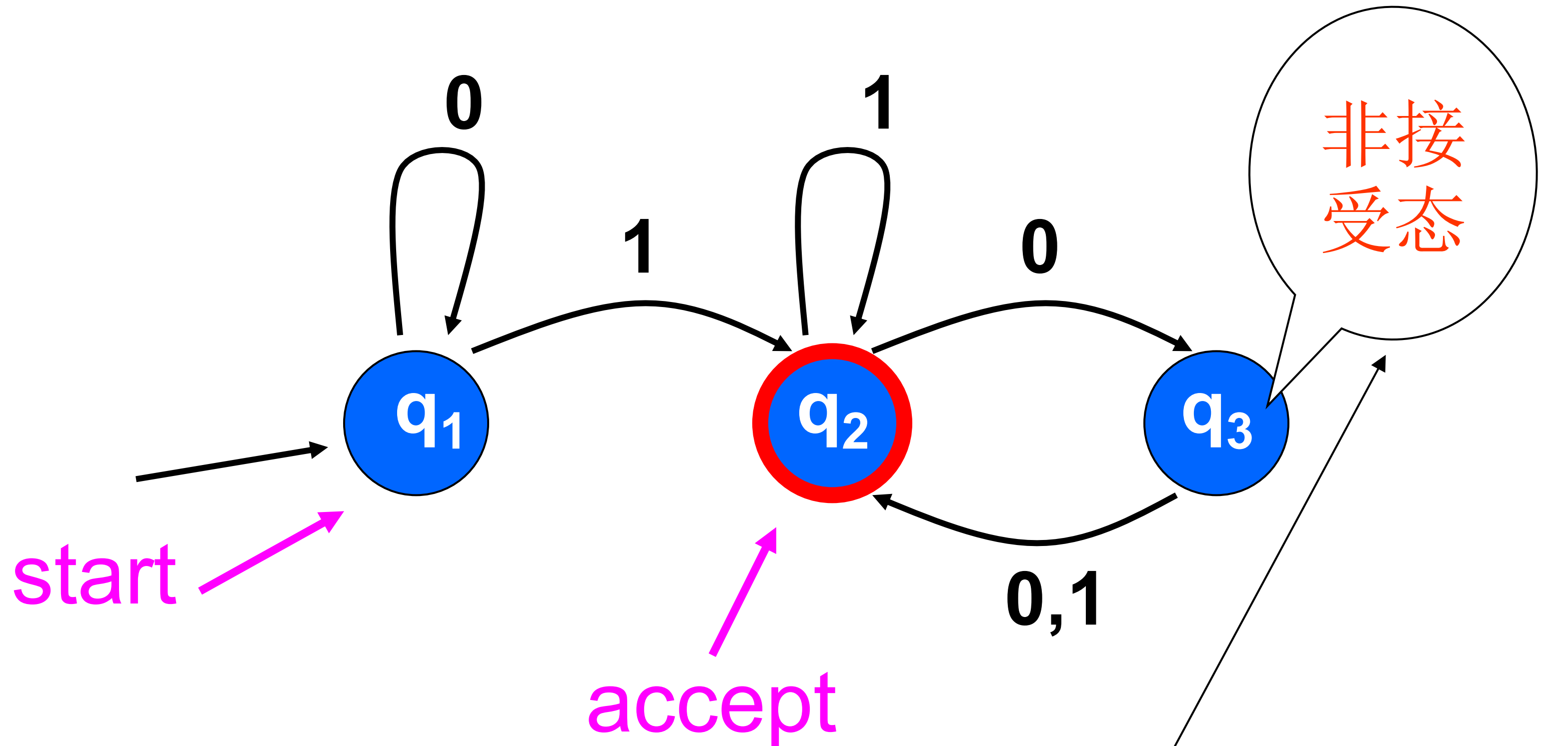
有穷自动机的表示

- 自动机的实质：**现状＋输入→下一状态**
- “Read once”，“no write” procedure.
- 只读，不写，无内存，无变量，无数组，无堆栈，无推理，无想象力
- Typical is its limited memory. 只有寄存器，能记住状态（窍门：造自动机时遇到困难加状态，等于是加寄存器扩大了记忆力）

状态转移图


transition
rules 变换规则
States状态
0
1
1
0
q_1
q_2
q_3
0,1
始态starting state
accepting state 受态，粗圈或双线圈

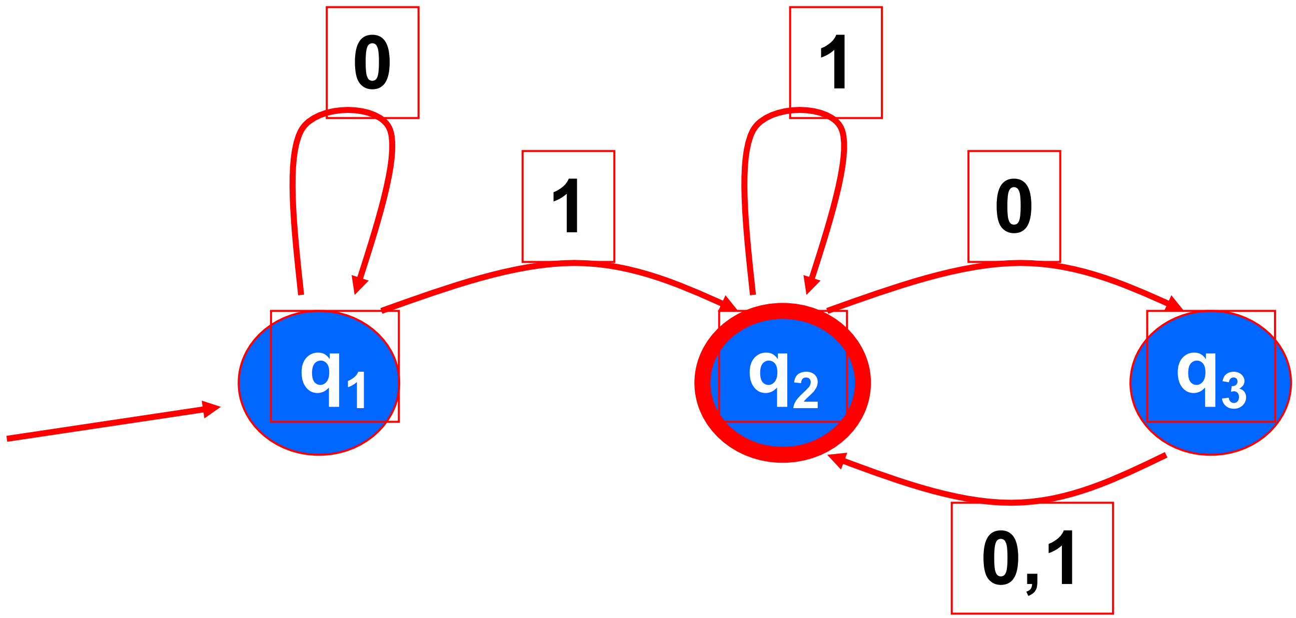
状态转移图



on input “0110”, the machine goes:

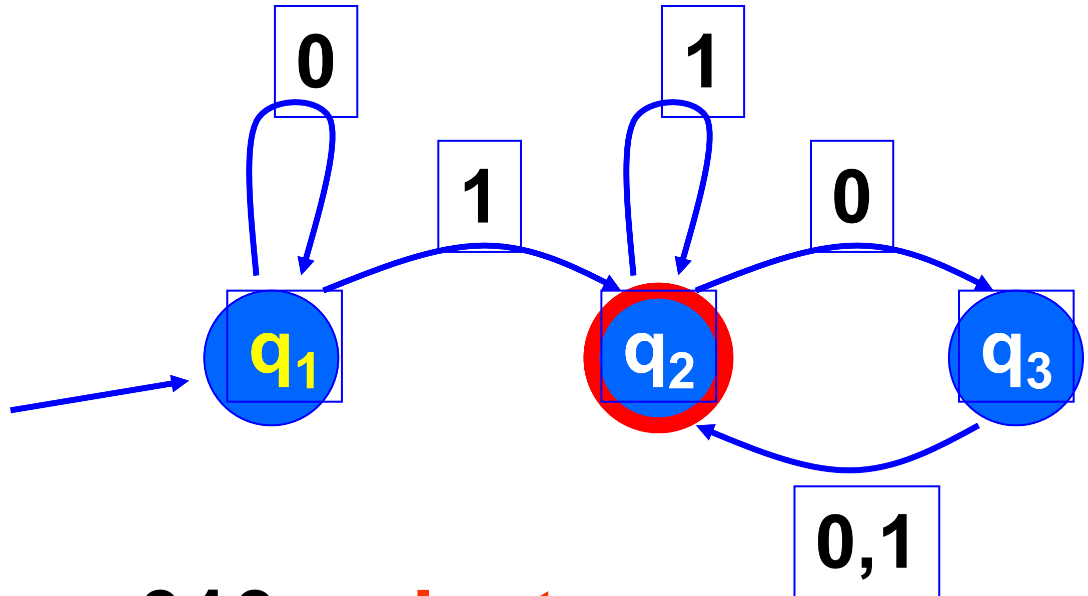
$q_1 \rightarrow q_1 \rightarrow q_2 \rightarrow q_2 \rightarrow q_3$ = “reject”

状态转移图



on input “101”, the machine goes:

状态转移图



010: reject

11: accept

0101001001001001100: accept

0100000010010: reject

ε: reject

有穷自动机的形式化定义

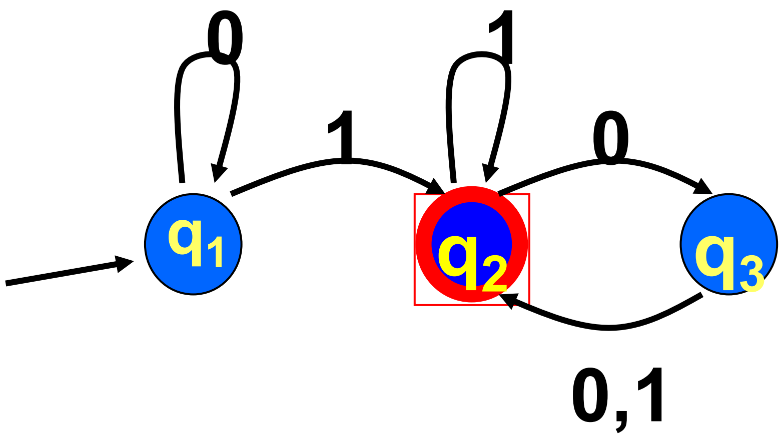
Definition 3.1 A deterministic finite automaton (DFA) is defined by a 5-tuple $(Q,\Sigma,\delta,q0,F)$

1. Q: finite set of states 状态集
2. Σ: finite alphabet 字母表
3. δ: transition function $\delta:Q\times\Sigma\rightarrow Q$ 转换函数
4. $q_0\in Q$: start state 始态
5. $F\subseteq Q$: set of accepting states 受态

Define of NFA

有穷自动机的形式化定义

EXP3-1：



1. <u>states</u> Q = $\{q_1,q_2,q_3\}$
2. <u>alphabet</u> $\Sigma = \{0,1\}$
3. <u>start state</u> q_1
4. <u>accept states</u> F=$\{q_2\}$

5. <u>transition function</u> δ:

转换函数用矩阵表示

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

有穷自动机接受的语言

The language recognized by a finite automaton M is denoted by L(M).We say that M recognizes A.

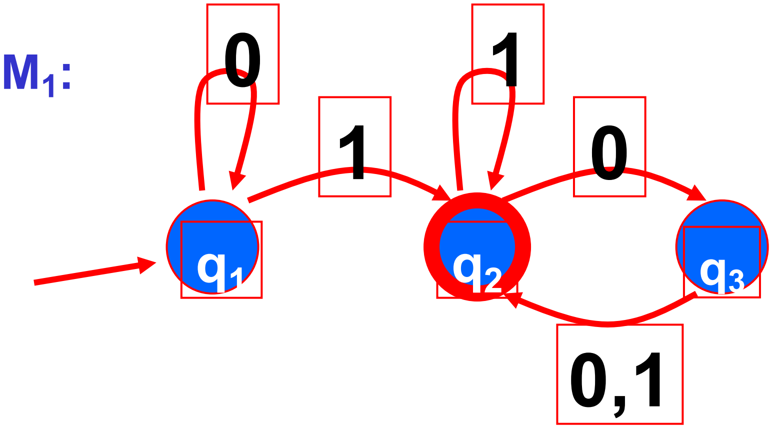
Def 3.2 A regular language is a language for which there exists a recognizing finite automaton.
正则语言，3型语言 ←→被FA接受的语言

Def 3.3 设M_1, M_2为FA。如果$L(M_1)=L(M_2)$，则称自动机M_1与M_2相互等价。

有穷自动机接受的语言

EXP3-2：

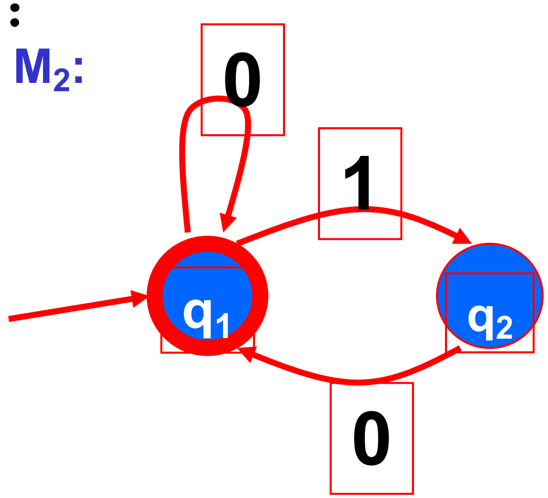
M$_1$:



L(M$_1$)={ω| ω contains at least 1 and an even number of 0s follow the last 1}

EXP3-3：

M$_2$:



L(M$_2$)={ω| ω is the empty string ε or ends in a 0}

如何设计有穷自动机

A creative process , using “Read as Automata”

设计步骤：

① 逐个读取字符，判断至今是否属于给定的语言

② 列出关键信息，每个信息关键对应一个状态

③ 给出transitions

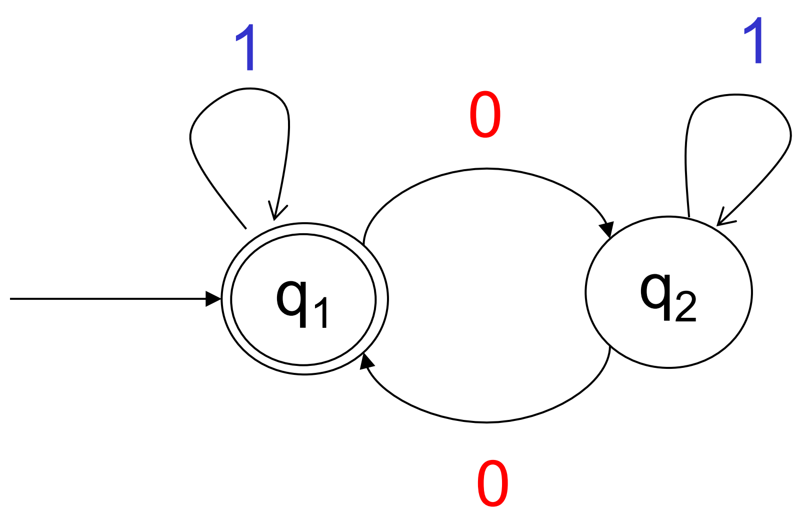
④ 将没有输入或输入为ε的状态，指定为start state

⑤ 将对应可接受的输入作为accept state

如何设计有穷自动机

EXP3-4：设计一台确定性有穷自动机M，M识别语言L(M)={ω∈{0,1}| ω中0的个数是偶数（含 0 ）}。

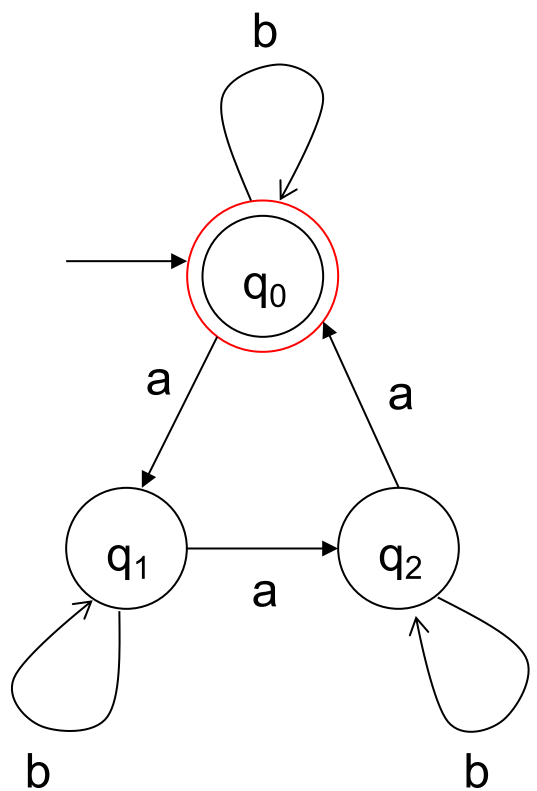
思考：关键信息是什么?



Regular operations

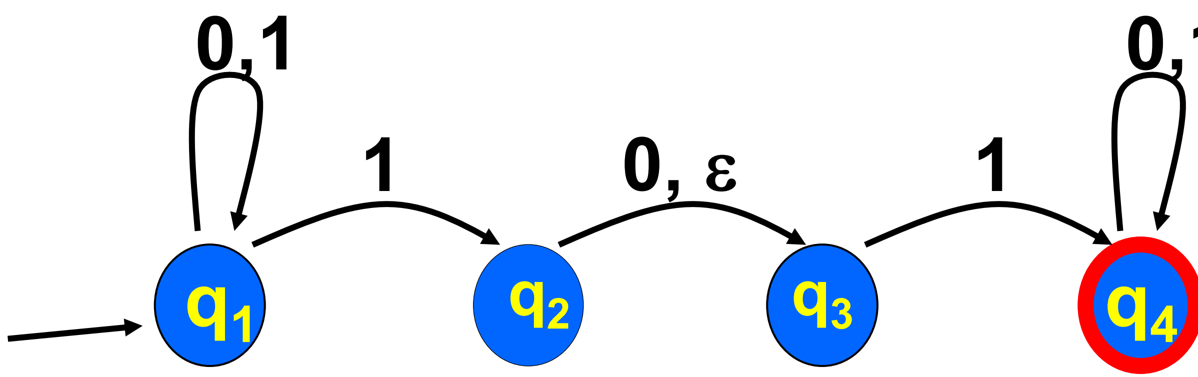
EXP3-5：设计一个DFA，它能识别语言

L(M)={ω∈{a, b}| ω中a的个数是3的倍数（含 0 ）}



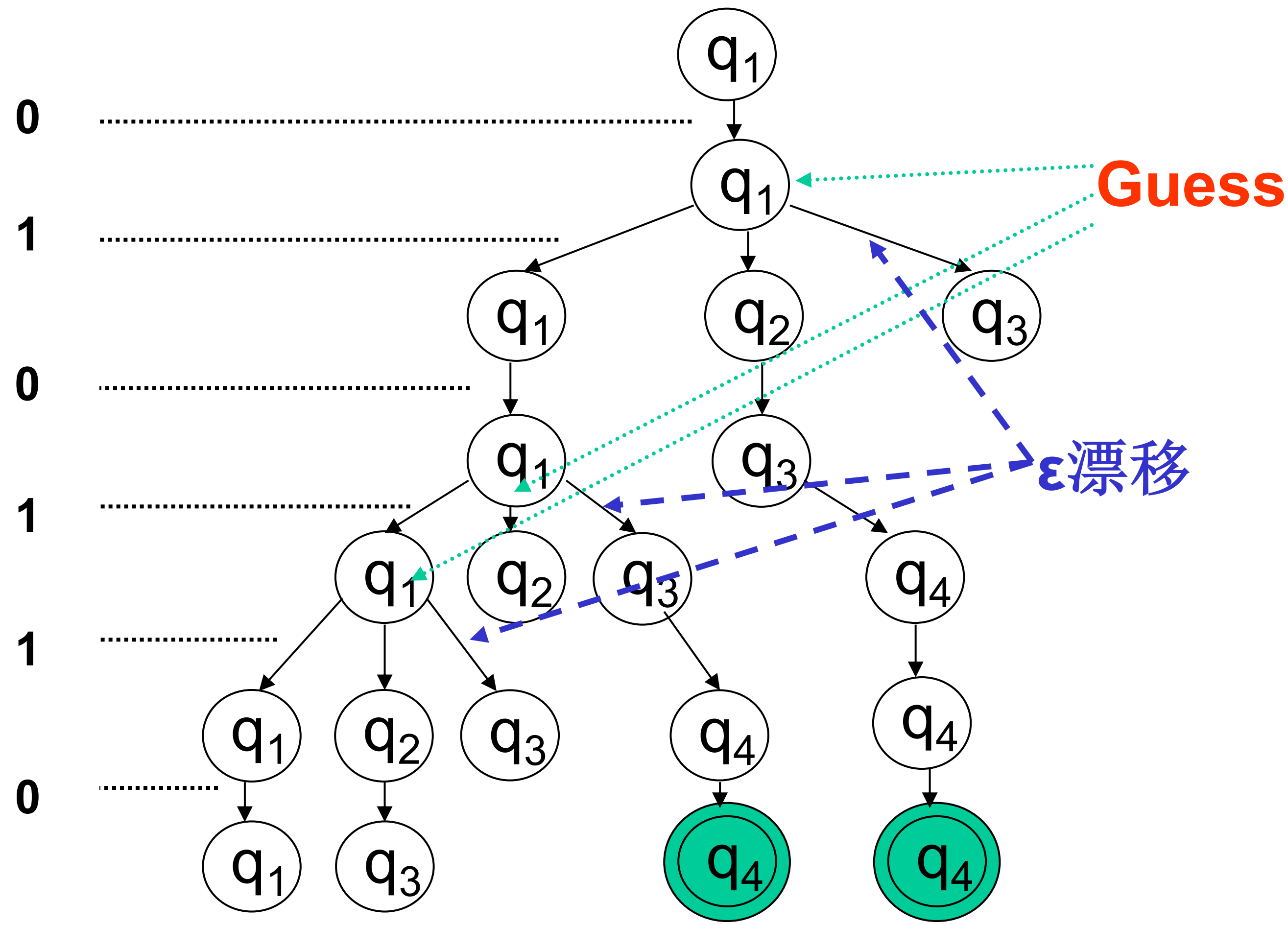
3.2 非确定性有穷自动机

What's NFA（Nondeterministic Finite Automaton）?



- Difference between NFA and DFA
  - NFA has zero,one or more exiting arrows for each input. 一入多出
  - NFA can deal with the **ε** (自动飘移)
  - NFA works as parallel computation or Tree

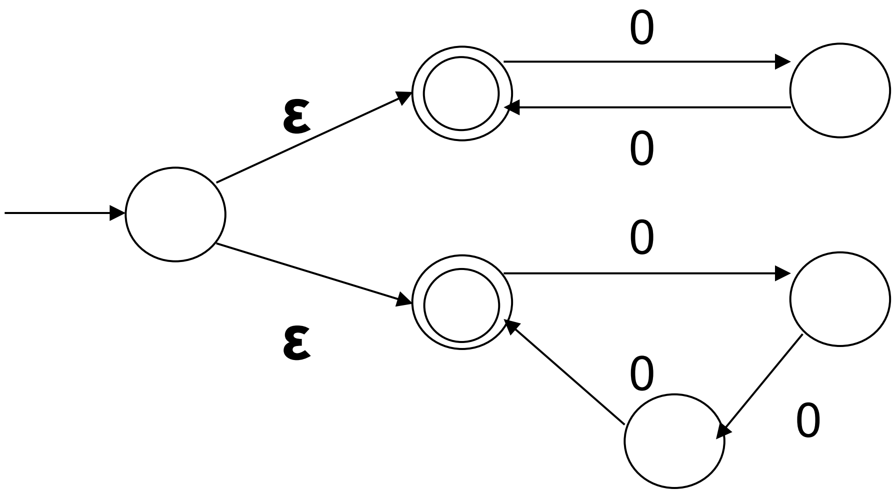
NFA的计算过程



- ➢ **The NFA works as a TREE**
- ➢**L（N1）= {ω| ω contains either ‘101’ or ’11’ as substring}**

ε的作用

EXP3-6：



L(N3) = {ω| ω =0^k, k is a multiple of 2 or 3}

ε 可以用于漂移或猜测

NFA is sometimes easier than DFA, constructing or understanding .

非确定性有穷自动机的形式化定义

Def 3.5 A nondeterministic finite automaton (NFA) M is defined by a 5-tuple $M=(Q,\Sigma,\delta,q_0,F)$, with

1. Q: finite set of states
2. Σ: finite alphabet
3. δ: transition function $\delta:Q\times\Sigma_\varepsilon\rightarrow P(Q)$
4. $q_0\in Q$: start state
5. $F\subseteq Q$: set of accepting states

- $\Sigma_\varepsilon = \sum \cup \{\varepsilon\}$
- P(Q): all subsets of Q, called Power set（幂集） of Q.

Define of DFA

NFA与DFA的主要区别

1. The function $\delta:Q\times\Sigma\varepsilon\rightarrow P\ (Q)$ is the crucial difference From DFA. It means:
   "When reading symbol "a" while in state q, one can go to one of the states in $\delta(q,a)\subseteq Q$."

   在 q态读a ,可到 $\delta(q\ ,\ a)$中某一状态

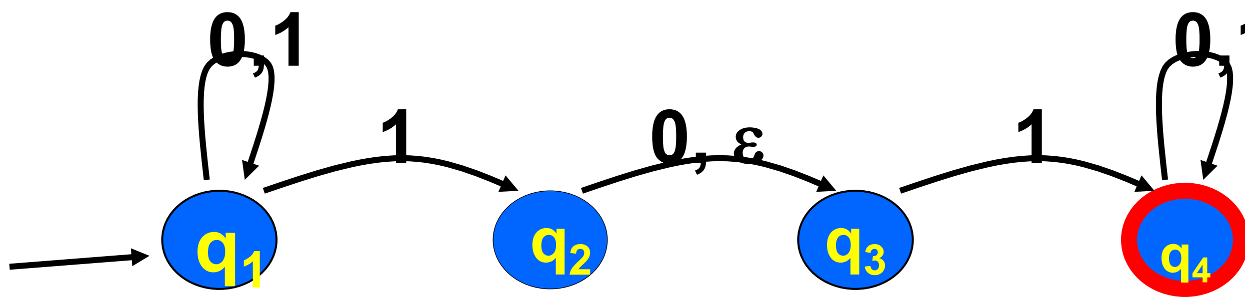
2. The ε in $\Sigma\varepsilon = \Sigma\cup\{\varepsilon\}$ takes care of the empty string transitions.

   ε 无输入 跳转状态，某些状态可无原因漂移

3. DFA的计算是线性的，NFA的计算是树形的。

非确定性有穷自动机的形式化定义

EXP3-7：



1. $Q = \{q_1,q_2,q_3,q_4\}$

2. $\Sigma = \{0,1\}$

3. is given as

	0	1	ε
q_1	$\{q_1\}$	$\{q_1,q_2\}$	Φ
q_2	$\{q_3\}$	Φ	$\{q_3\}$
q_3	Φ	$\{q_4\}$	Φ
q_4	$\{q_4\}$	$\{q_4\}$	Φ

4. q_1 is the start state, and

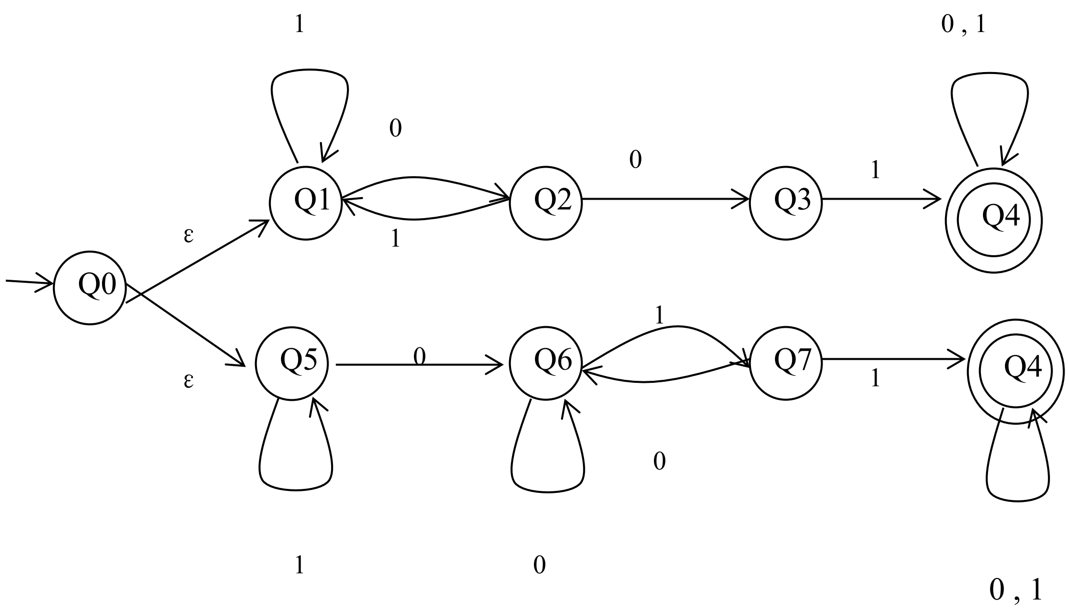
5. $F = \{q_4\}$

非确定性有穷自动机的设计

EXP3-8:构造一个接受“含有子串011或001”的NFA，字母表为{0,1}。要求：

① 采用文字形式，简要说明设计思路。

② 给出所设计的非确定性自动机的状态转移图或形式化定义。



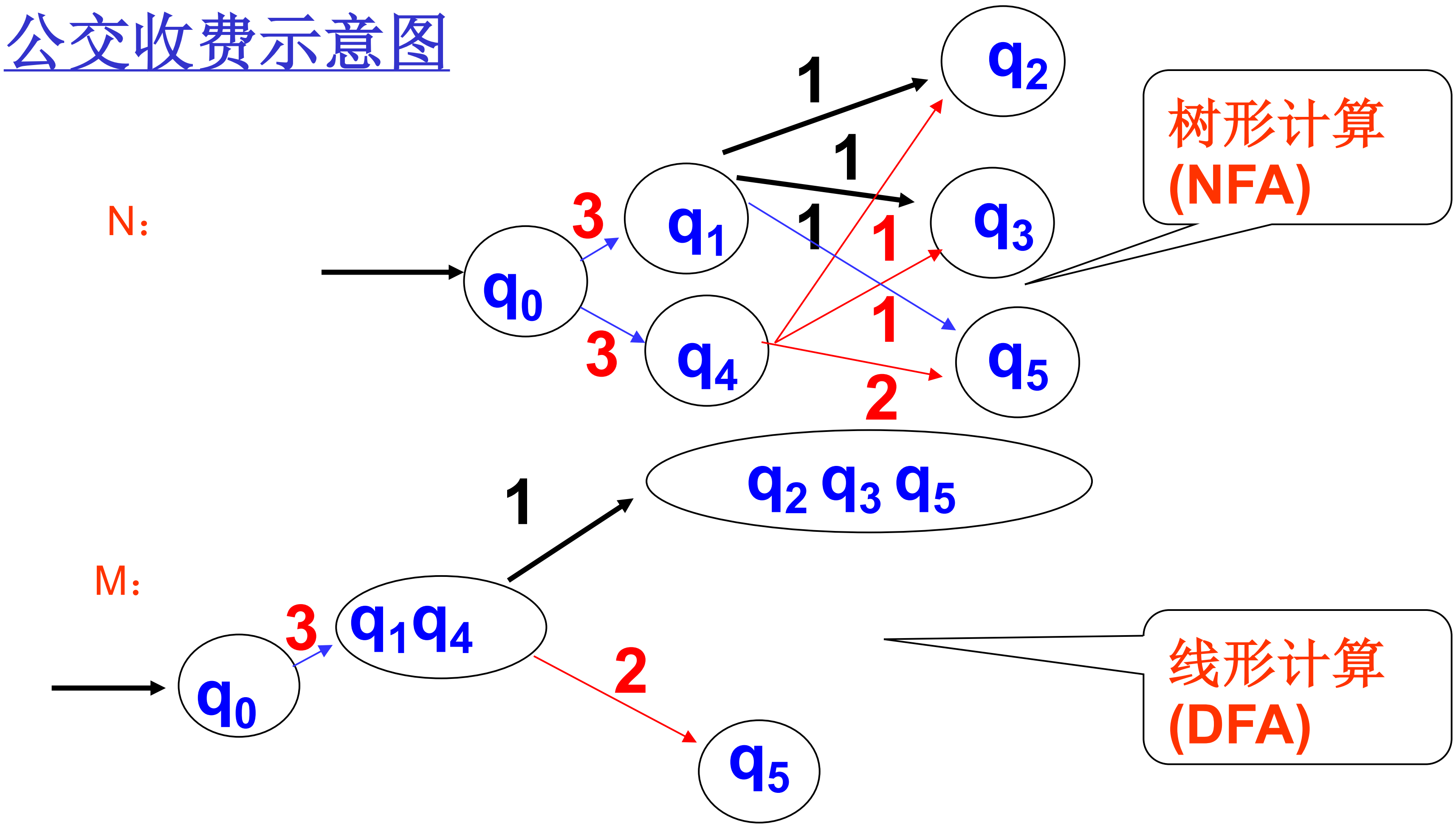
3.3 DFA与NFA的等价性

Theorem 3.1　NFA与DFA是等价的。

Proof idea：

1. FA等价的含义？
2. DFA的计算过程是直线形的，NFA的计算是树形的。
3. 关键是：如何将树形的计算转换成直线形的计算。

DFA与NFA的等价性



以3，1序列为例子，其路线组合为：$\{q_0\} \rightarrow \{q_1,q_4\} \rightarrow \{q_2,q_3,q_5\}$。从而，完成了树形（NFA）→线形（DFA）的转换。

DFA与NFA的等价性

从上面的例子分析来看，**DFA M**的状态实际上就是**NFA N**的状态集合的子集，即**Power Set (幂集)**。

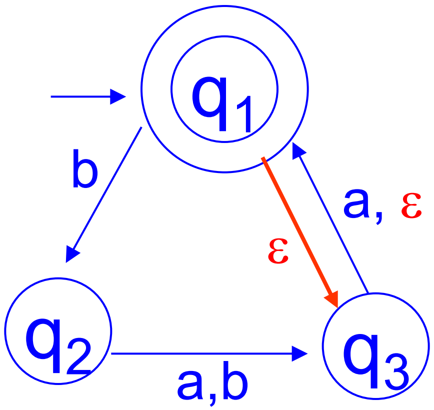
若**NFA**的状态**Q**包含**n**个状态，则$\mathcal{P}$（Q）= 2^n;
由此可见:

1. 与NFA等价的DFA的状态是NFA状态的幂集;
2. 一般来讲, DFA比它等价的NFA要复杂(状态多了);

DFA与NFA的等价性

0步集E(R)的概念:

E(R) = {q | q can be reached from R by traveling along 0 or more ε arrows， q ∊ Q， R⊆Q}.



$E(q_1)=\{q_1,q_3\}$

$E(q_3)=$?

DFA与NFA的等价性证明

Proof： Let $N = (Q,\sum,\delta,q_0,F)$ be the NFA， recognizing language **A**. Now , we construct a DFA M recognizing **A**.

Construct $M = (Q', \sum, \delta', q_0', F')$

1. $Q' = \mathcal{P}(Q)$

2. $\delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a)) \text{ for } r \in R\}$

3. $q_0' = E(\{q_0\})$

4. $F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}$

结果：**L(M) = L(N)**，**DFA与NFA**识别相同的语言，所以，**DFA与NFA**是等价的。

如何把NFA转化为等价性的DFA

NFA→DFA

1. First, determine D’ s states.
2. Next, we determine the start and accept states of D.
3. Finally, we determine D’ s transition function.
4. Optionally, simplify the machine D, removing unnecessary states.

NFA的应用举例

EXP 3-9 试证两个正则语言的并，还是正则语言。

Let $N_1=(Q_1, \sum,\delta_1,q_1,F_1)$ recognize A_1,$N_2=(Q_2, \sum,\delta_2,q_2,F_2)$

Construct $N=(Q, \sum,\delta,q_0,F)$ to recognize $A_1 \cup A_2$.

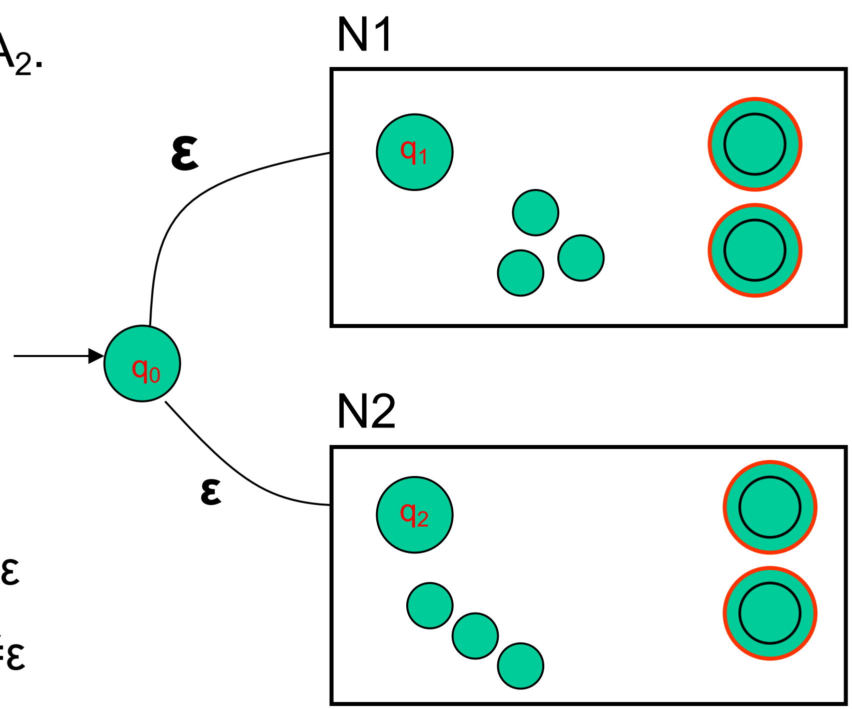
1.$Q=\{q_0\} \cup Q_1 \cup Q_2$

2.The State q_0 is the start state of N.

3.The accept state $F = F_1 \cup F_2$

4.

$$\delta(q, a) = \begin{cases} \delta_1(q,a), & q \in Q_1 \\ \delta_2(q,a), & q \in Q_2 \\ \{q_1,q_2\}, & q=q_0 \text{ and } a=\varepsilon \\ \varnothing, & q=q_0 \text{ and } a \neq \varepsilon \end{cases}$$



其中：$q \in Q$, $a \in \sum_\varepsilon$

3.4 带空转移的有穷自动机

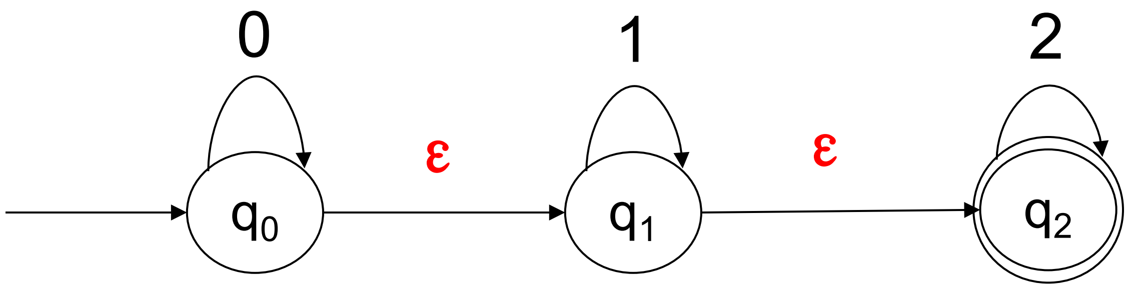
Def 3-9 带空移动的不确定的有穷状态自动机(ε-NFA)，$M = (Q, \Sigma, \delta, q_0, F)$，其中

- Q、Σ、q_0、F的意义同DFA。
- δ：$Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$

- 带空转移的有穷自动机，实际上就是允许输入空字符ε的非确定性有穷自动机，是一种特殊的NFA。
- **语义**：允许自动机在某一状态下，不读入字符（读写头不移动），而只改变状态。
- **作用**：不改变、不扩大有穷自动机的语言类，仅提供表达上的便利（形式上，比一般的NFA更简洁）。

3.4 带空转移的有穷自动机

EXP 3-10 具有 ε 转移的有穷自动机



对于上图的 ε -NFA，转移函数如下：

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

$\delta(q_1,1) = \{q_1\}$，$\delta(q_1,\varepsilon) = \{q_2\}$

$\delta(q_2,2) = \{q_2\}$

3.4 带空转移的有穷自动机

Def 3-10 在一个具有ε转移的有穷自动机中，对于状态q，它的ε-CLOSURE(q)是按下列规则递归定义的状态集：

（1） q在ε-CLOSURE(q)中；

（2） 如p在ε-CLOSURE(q)中，则δ(p, ε)也在ε-CLOSURE(q)中；

（3） 重复（2），直到ε-CLOSURE(q)中的状态不再增加为止。

进一步，对于状态集P的ε-CLOSURE，规定

$$\varepsilon\text{-CLOSURE}(P) = \bigcup_{q \in P} \varepsilon\text{-CLOSURE}(q)$$

ε-CLOSURE(P) 就是0步集

在EXP3-10中，

$\varepsilon\text{-CLOSURE}(q_0) = \{q_0,q_1,q_2\}$， $\varepsilon\text{-CLOSURE}(q_1)=\{q_1,q_2\}$，

$$\varepsilon\text{-CLOSURE}(\{q_0, q_1\}) = \varepsilon\text{-CLOSURE}(q_0) \cup \varepsilon\text{-CLOSURE}(q_1) = \{q_0,q_1,q_2\}$$

3.4 带空转移的有穷自动机

Def 3-11 对于一个具有ε转移的有穷自动机，它的**扩充转移函数**$\hat{\delta}$定义如下：

(1) $\hat{\delta}(q, \varepsilon) = \varepsilon - \mathrm{CLOSURE}(q)$

(2) $\hat{\delta}(q, \omega a) = \varepsilon\text{-}\mathrm{CLOSURE}(P)$, $\quad P = \bigcup_{r \in \hat{\delta}(q,\omega)} \delta(r, a)$

其中，q∈Q，a∈$\boldsymbol{\Sigma}$，ω∈$\boldsymbol{\Sigma}$*。

Theorem3.2 ε-NFA 与 NFA 是等价的。
证明见课本p54-55，归纳法，自学。

3.5 有穷自动机的应用---文本搜索

■ 问题的提出：

给定一个单词（或单词集合），在web或其它在线文本库中，如何查找包含一个（或全部）单词的所有文档。

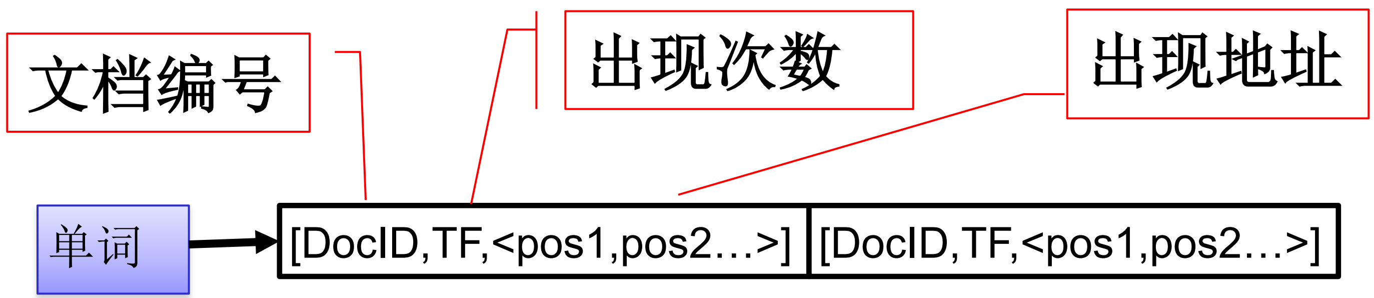
■ 解决办法：

倒排索引（**Inverted index**），也常被称为反向索引、置入档案或反向档案，是一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。现代搜索引擎的索引都是基于倒排索引。

倒排索引是搜索引擎的主要工作机制。

3.5.1 问题的分析

■ 倒排索引项



最终，所有单词的倒排序索引项构成了倒排索引列表。

■ 倒排索引的缺点

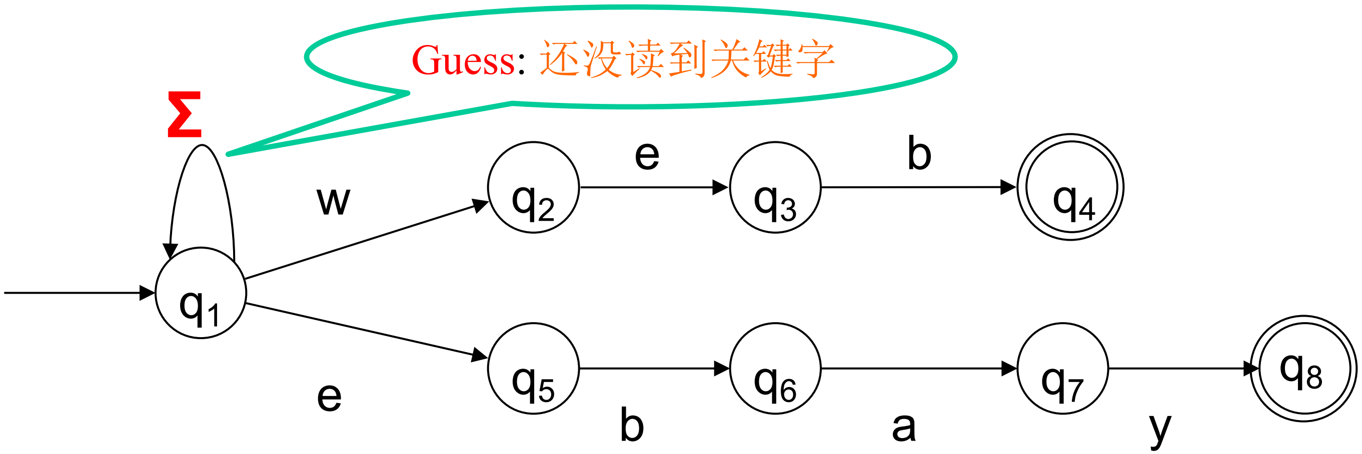
倒排索引不适用下列情况：

① 搜索的文本库快速变化，如在线新闻，股票行情；

② 搜索的文档不能建立目录，如商业销售网站；

3.5.2 文本搜索的NFA

EXP 3-11：设计一台识别单词web和ebay的NFA。



两种程序实现的选择：

1：编写一个程序模拟该NFA，计算出每个输入符号后所处的状态集合，检查什么时候到达可接受状态。

2：将NFA转化成DFA，然后用程序模拟这个DFA。

3.5.3 识别关键字的DFA

第一步：构建 DFA 的状态（子集构造法）

1、如果q_1是NFA的初始状态，则$\{q_1\}$是DFA的一个状态；

2、如果p是NFA的一个状态，从初始状态，沿着带$a_1a_2...a_m$符号的路径可达该p，则有一个DFA状态是由下列NFA状态组成的集合：

（a）q_1;

（b）p;

（c）每一个从q_1出发，沿着带$a_1a_2...a_m$后缀标记的路径可达的NFA状态。

举例说明：

1、按照上述的子集构造法，上例NFA中的起始状态q_1，必将出现在对应DFA的每个状态中；

2、从起始状态q_1出发，沿着带符号web的路径（q_1、q_2、q_3、q_4）可到达q_4；web的后缀为eb及b，其中eb满足2(c)的要求，即从q_1出发，沿着带eb标记的路径可达q_6。所以，q_1、q_4、q_6构成DFA的一个状态。

3.5.3 识别关键字的DFA

第二步： 根据子集构造，计算DFA 的状态转移

1、对于任一状态集合 Q(q_1， $p_1,\cdots,p_n$)，考察每个可能的输入x，如果在NFA中存在转移$q_i=\delta(p_i，x)$，则，在DFA中，构造转移$\{q_i\}=\delta(Q，x)$;

2、如果不存在从任何pi出发的带x的转移，则，在DFA中，构造转移$\{q_1，\delta(q_1，x)\}=\delta(Q，x)$;

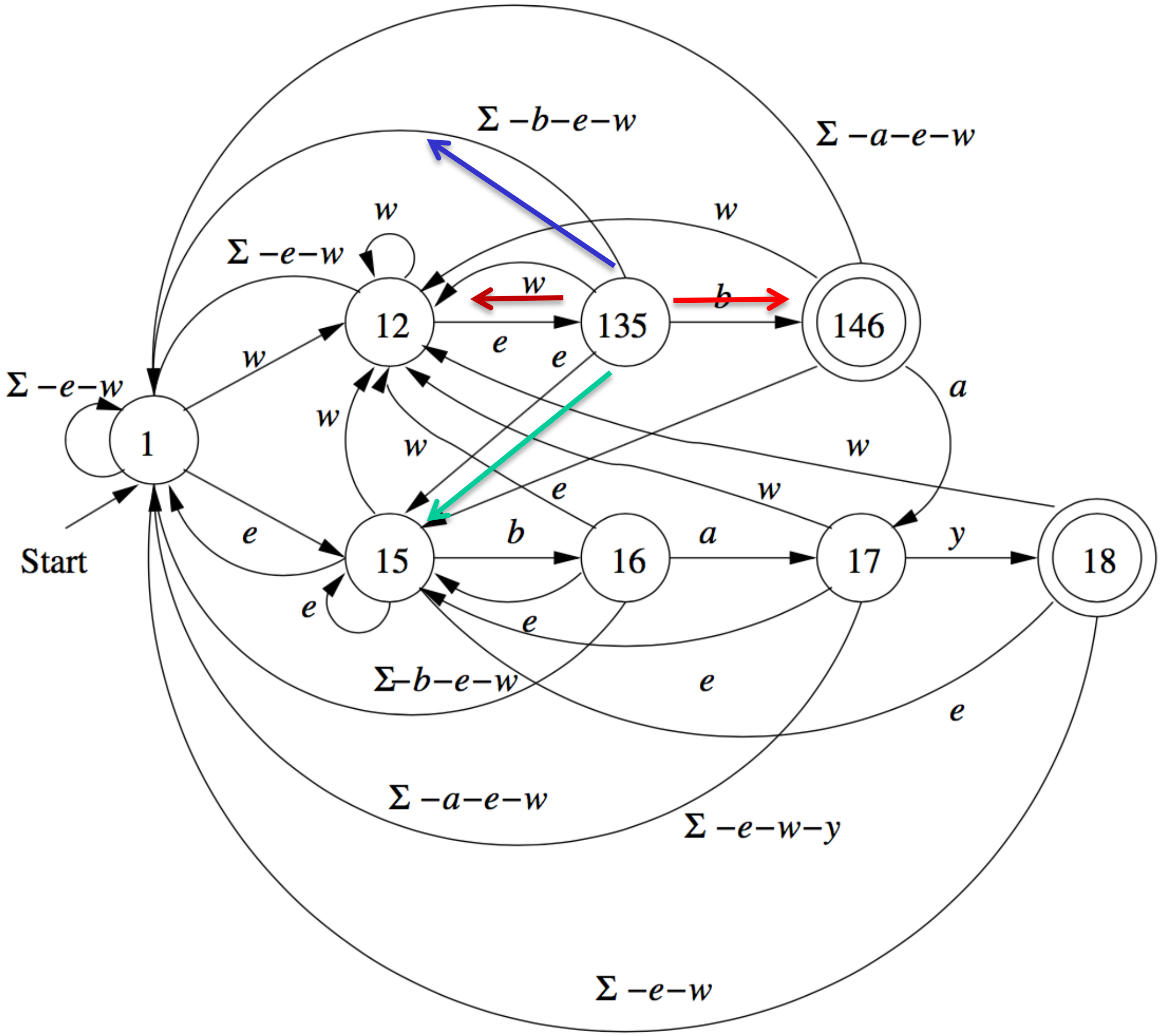
举例说明：

1、考虑DFA的状态**135**，对于输入b，在NFA中，状态1转移到状态1 ，状态3转移到状态4，状态5转移导状态6。根据上述第1点，在DFA中，135输入b后转移到146；

2、对于输入e，NFA中，状态3、5都没有发生转移，但有状态1到状态5的转移。根据上述第2点，在DFA中，135输入e后转移到15。同样，输入w后，135转移到12。

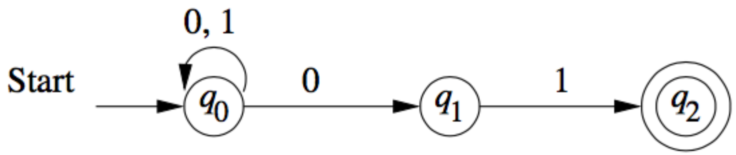
3、对于其它输入，NFA没有从3和5出发的转移，状态1只有到自身的转移。因此，在DFA中，输入∑-b-e-w后，135转移到状态1;

3.5.3 识别关键字的DFA



3.5.3 识别关键字的DFA

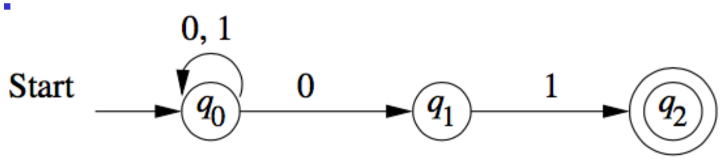
EXP3-12 将下列NFA转化为等价的DFA。



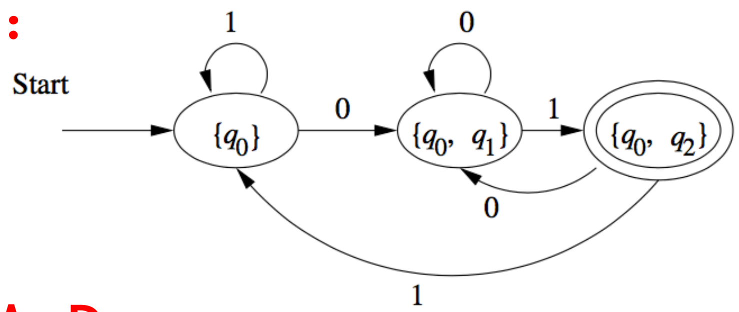
解题思路：方法一：**基于定理3.1的方法**；方法二：**子集构造法**。两种方法有什么区别？

3.5.3 识别关键字的DFA

N:



D:



方法一：按照定理3.1构造DFA D。

step1：从$\{q_0\}$开始：

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

step2：对于新增加的状态$\{q_0,q_1\}$，再求转移函数：

$\delta'(\{q_0,q_1\}, 0) = \{q_0,q_1\}$，$\delta'(\{q_0,q_1\}, 1) = \{q_0,q_2\}$

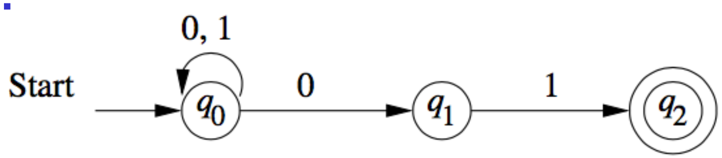
step3：对于新增加的状态$\{q_0,q_2\}$，再求转移函数：

$\delta'(\{q_0,q_2\}, 0) = \{q_0,q_1\}$，$\delta'(\{q_0,q_2\}, 1) = \{q_0\}$

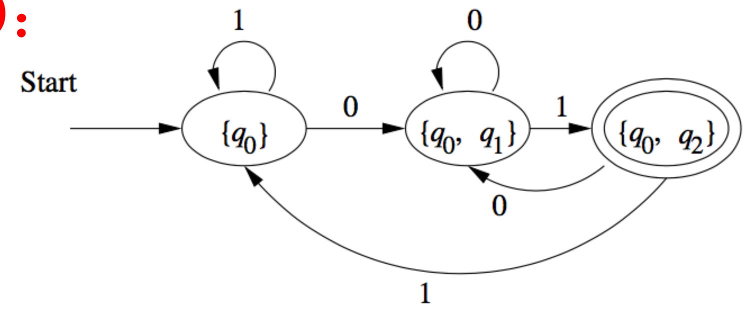
至此，不再增加新的状态，所求的DFA共有$\{q_0\}$、$\{q_0,q_1\}$、$\{q_0,q_2\}$ 3个状态，如上图所示。

3.5.3 识别关键字的DFA

N:



D：



方法二：子集法构造DFA D。

注：求解过程略，课后练习。

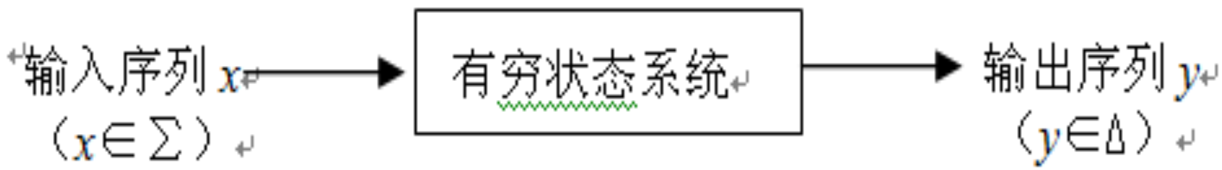
结果如上图所示，DFA共有$\{q_0\}$、 $\{q_0,q_1\}$、$\{q_0,q_2\}$ 3个状态，与方法一的结果一致。

3.5.3 识别关键字的DFA

NFA→DFA的两种方法比较：

1. 子集构造法，DFA的状态数总是不超过NFA的状态数。基于定理3.1的方法，最坏情况下，NFA转化为DFA时，状态数会呈指数增长。
2. 本质上，两种方法的结果是一致的。

3.6 带输出的有穷自动机



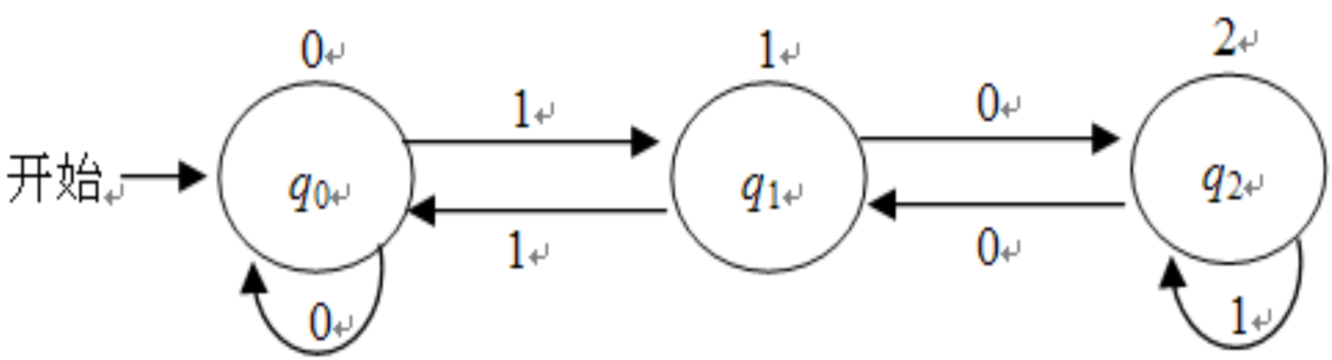
定义 3.12 一个Moore机是一个六元组

$$M=(Q,\Sigma,\Delta,\delta,\lambda,q_0)$$

其中Q,Σ,δ,q_0的意义与DFA中的相同。Δ是一个输出字母表，λ是从Q到Δ的映射，称为输出函数。

3.6 带输出的有穷自动机

- EXP 3.13 设计一个Moore机，$\sum=\{0, 1\}$，若将输入串看成一个二进制数，要求在读入过程中，能输出它已读过子串的模3余数。
- 因为模3余数只能有0，1，2三个值，因此取$\Delta=\{0, 1, 2\}$，并且只设三个状态q_0, q_1, q_2，分别对应这三种余数。这个Moore机如下图所示（有向边上的符号仍然代表δ函数的第二个变元，状态q_i上面的符号代表λ函数的值）。



3.6 带输出的有穷自动机

定义 3.13 一个Mealy机是一个六元组

$$M=(Q,\sum,\Delta,\delta,\lambda,q_0)$$

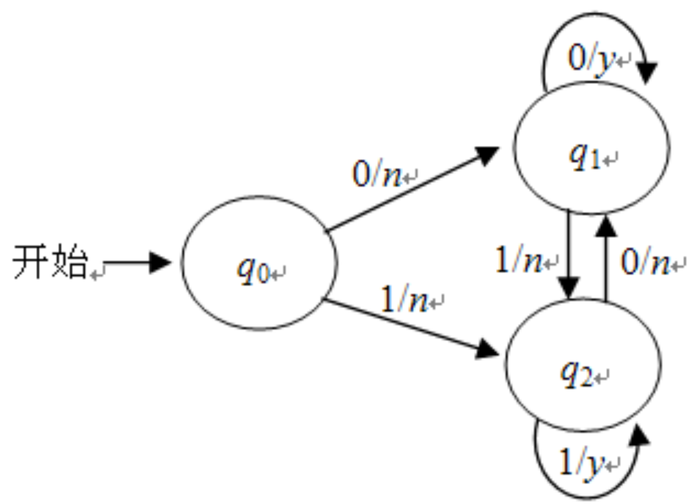
这里除λ是从$Q\times\sum$到Δ的映射外，其余符号的意义都和Moore机相同。$\lambda(q,a)$指出了当状态q遇到符号a时的输出。当输入串为$a_1a_2\ldots a_n$时，设$\delta(q_0,a_1)=q_1,\ldots,\delta(q_{i-1},a_i)=q_i,\ldots,$

$\delta(q_{n-1},a_n)=q_n$。这时的输出符号序列应为:

$\lambda(q_0,a_1)\lambda(q_1,a_2)\ldots\lambda(q_{i-1},a_i)\ldots\lambda(q_{n-1},a_n)$。注意，Mealy机与Moore机不同，当输入串长度为n时，它输出n个符号，而Moore机是输出n+1个符号。

3.6 带输出的有穷自动机

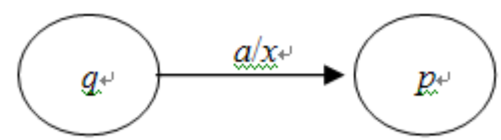
- EXP3.14 给出一个0，1串的集合S，该集合中的串都以00或11结尾。要求设计一个只有两个输出符号（Δ={y,n}）的Mealy机，当它读过属于集合S的串时，输出y，表示接受；当它读过不属于集合S的串时，输出n，表示不接受。这个Mealy机如下图所示：



Moore机与Mealy机的等价性

- 定理 3.3 如果$M_1=(Q,\sum,\Delta,\delta,\lambda,q_0)$是一个Moore机，则有一个Mealy机$M_2$与之等价。

  证明 令$M_2=(Q,\sum,\Delta,\delta,\lambda',q_0)$，其中$\lambda'$定义为:对一切$q\in Q,a\in\sum$，$\lambda'(q,a)=\lambda(\delta(q,a))$。由于$M_2$和$M_1$具有相同的状态和δ函数，故对同样的输入序列，状态转移的序列相同。不同的只是将M_1的输出向前移半个节拍，即若M_1有



则M_2有

在不考虑M_1第一个输出符号的情况下，M_2和M_1的输出序列必然相同。定理得证。

Moore机与Mealy机的等价性

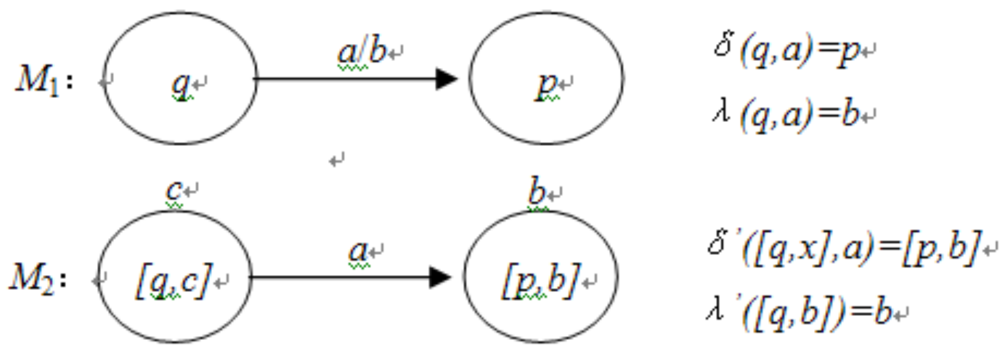
- 定理 3.4 如果$M_1=(Q,\sum,\Delta,\delta,\lambda,q_0)$是一个Mealy机，则有一个Moore机$M_2$与之等价。
- 证明 令$M_2=(Q\times\Delta,\sum,\Delta,\delta',\lambda',[q_0,b_0])$，这里$b_0$是Δ中任意一个符号。δ´和λ´的具体定义是：

  $\delta'([q,b],a)=[\delta(q,a),\ \lambda(q,a)]$,

  $\lambda'([q,b])=b$。

  M_2的状态由两个分量组成，第一个分量是Q中的状态，第二个分量是Δ中的符号。对于同一输入序列，M_2的状态序列中每个状态的第一分量，与M_1的状态序列中每个状态依次相同；而将M_1的输出符号“吸收”到M_2的状态的第二分量上，延迟半个节拍以后，再输出该符号。这一过程可用下图表示。

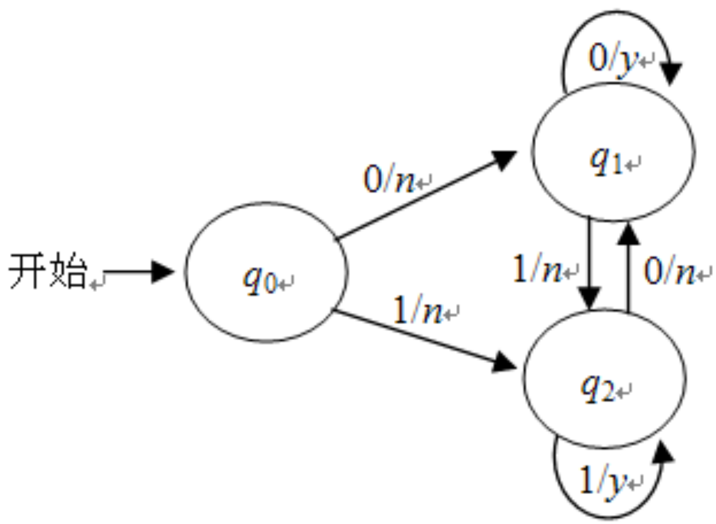
Moore机与Mealy机的等价性



- 因在同一个输入串$a_1a_2\ldots a_n$上，若M_1的状态变化序列为$q_0,q_1,\ldots,q_n$，输出序列为$b_1,b_2,\ldots,b_n$，根据M_2的构造，则M_2的状态变化序列就是$[q_0,b_0]$ $[q_1,b_1],\ldots,[q_n,b_n]$，输出序列就是$b_0$，$b_1,b_2,\ldots,b_n$。不计$M_2$的第一个输出符号$b_0$，$M_2$与$M_1$的输出序列相同，因此它们是等价的。定理证完。

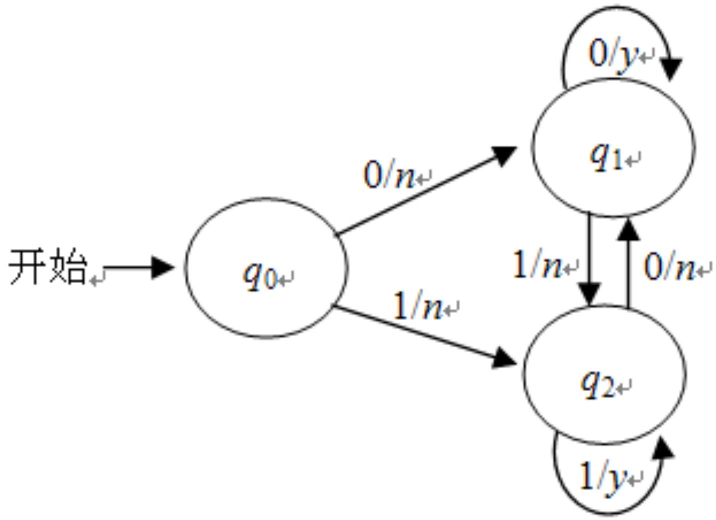
Moore机与Mealy机的等价性

- EXP 3.15 设M_1是下图所示的Mealy机，根据定理 3.4中的构造方法，将它化为等价的Moore机M_2。M_2共有6个状态：$[q_0,n],[q_0,y],[q_1,n],[q_1,y],[q_2,n],[q_2,y]$。



Moore机与Mealy机的等价性

- EXP 3.16 设M_1是下图所示的Mealy机，根据定理 3.4中的构造方法，将它化为等价的Moore机M_2。M_2共有6个状态：$[q_0,n],[q_0,y],[q_1,n],[q_1,y],[q_2,n],[q_2,y]$。



补充几个概念

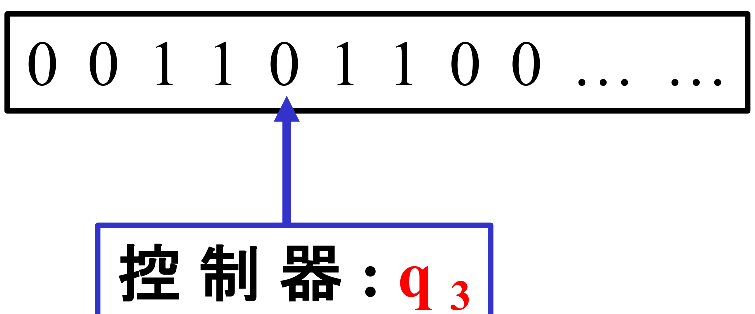
1. 关于ε的理解：

- ε是空字符串，$|\varepsilon|=0$，$\varepsilon010=0\varepsilon10=010$，$\{\varepsilon\} \neq \boldsymbol{\Phi}$；
- NFA接受ε，表示NFA什么也不读（即读写头不移动），但可以实现状态转移；

2.即时描述（instantaneous description,ID）

就是自动机某时刻的一个“快照”，保存当前自动机所处的状态、磁带内容、读写头的位置。通常，称之为格局（configuration）。

一般记作：xqy，如0011$q_3$01100，其中x，y$\in\Sigma^*$，q $\in$Q，且$\delta(q_0,x) = q$。



补充几个概念

3. 格局演化 & 计算

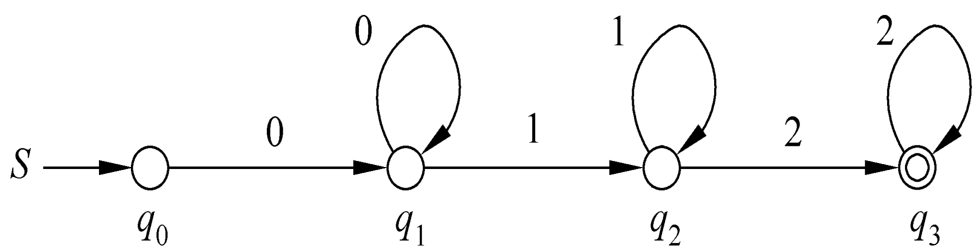
从一个格局到另一个格局的变化序列，称为格局演化，这也是一个计算过程。通常记作：

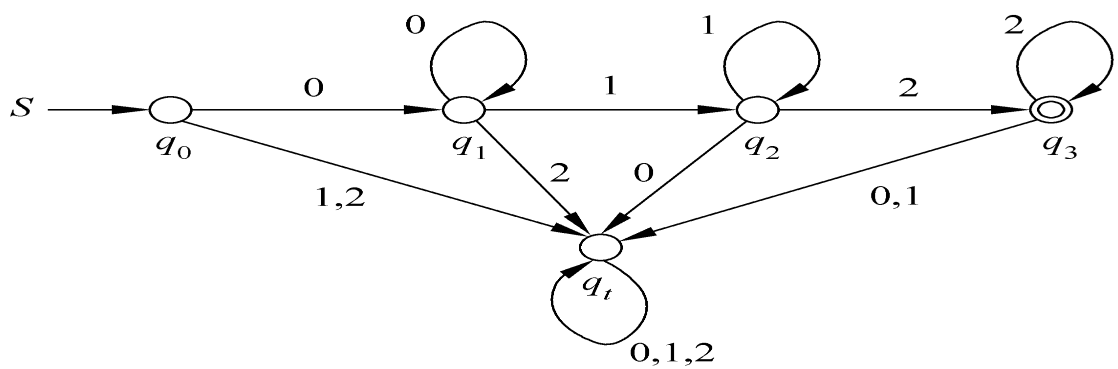
通常记法	教材上的记法	含 义
$C1 \rightarrow C2$	$C1 \vdash C2$	一步演化，$q_0 1100 \rightarrow 1 q_2 100$
$C1 \Rightarrow^{*} C2$	$C1 \vdash^{*} C2$	多步演化(包括0步)
$C1 \Rightarrow^{+} C2$	$C1 \vdash^{+} C2$	多步演化(至少1步)

一个有穷的格局演化序列，而且，最终处于接受格局，那么，这个格局演化过程就是**可计算的**。

补充几个概念

3. 陷进状态





本 章 小 结

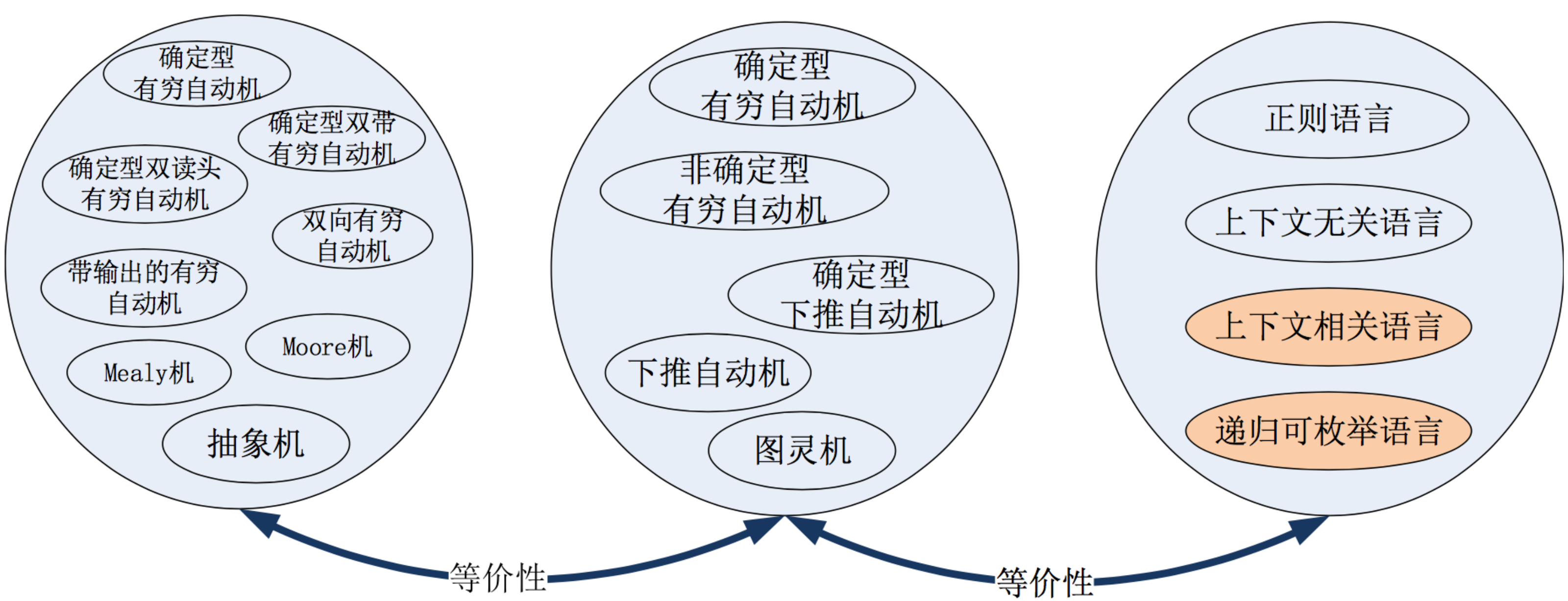
知识要点

1. 有穷自动机是一种计算装置（结构、计算机理）、一种计算模型（表示、计算方法）、与语言（文法、性质）相关。

计算装置:
结构
计算机理

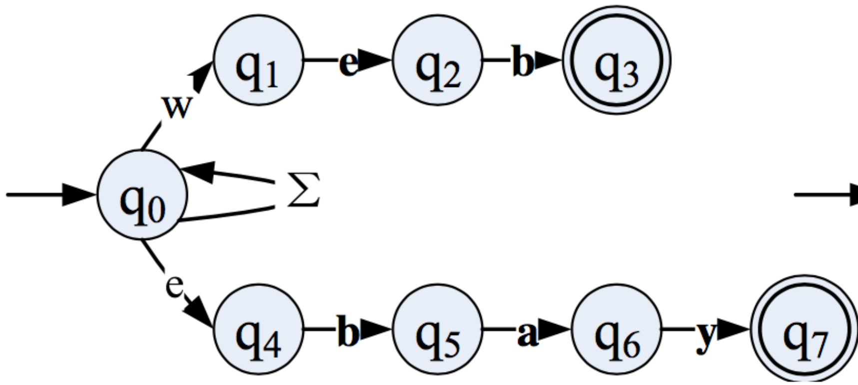
计算模型:
表示
计算方法

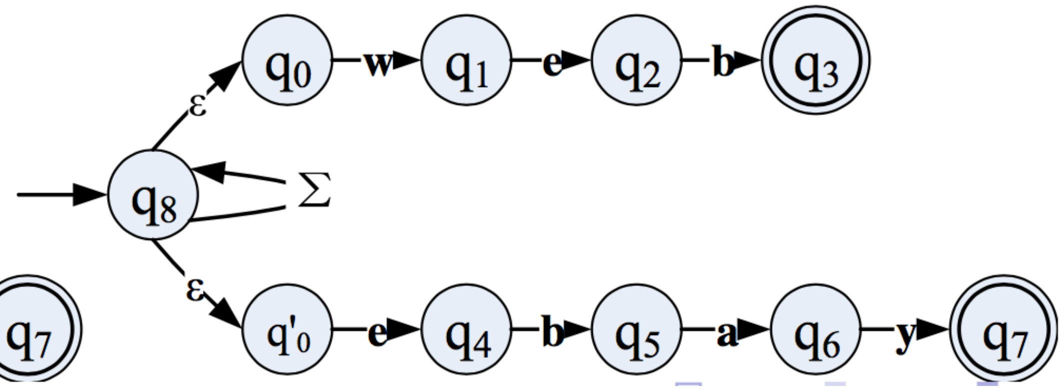
语言：
文法
性质

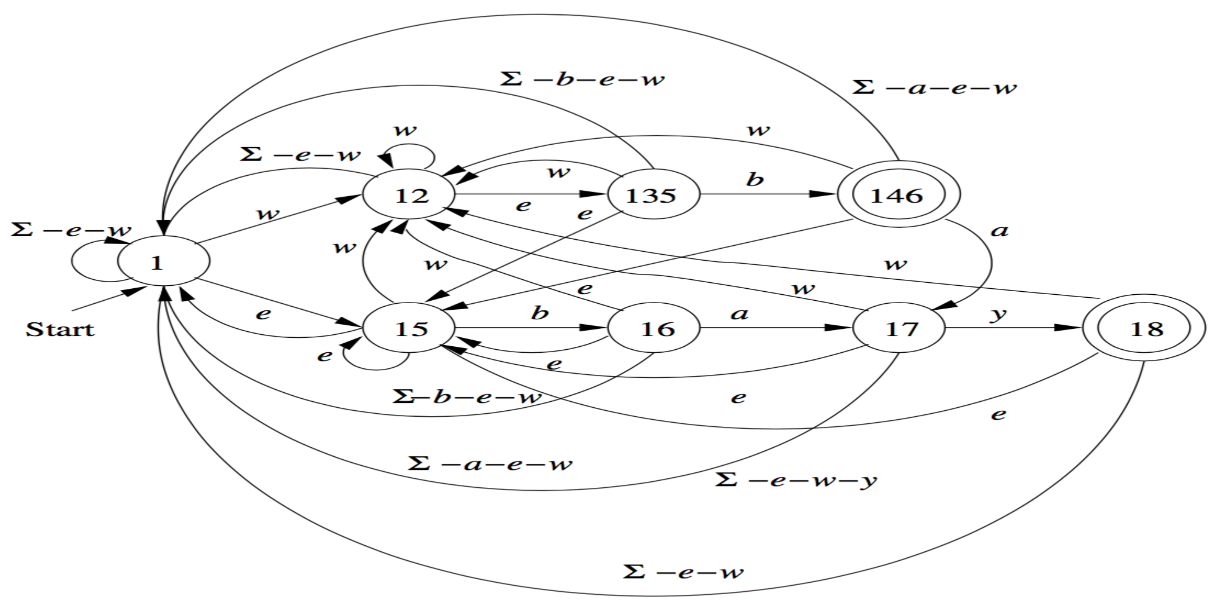


本 章 小 结

2. DFA、NFA、ε-NFA之间的主要区别

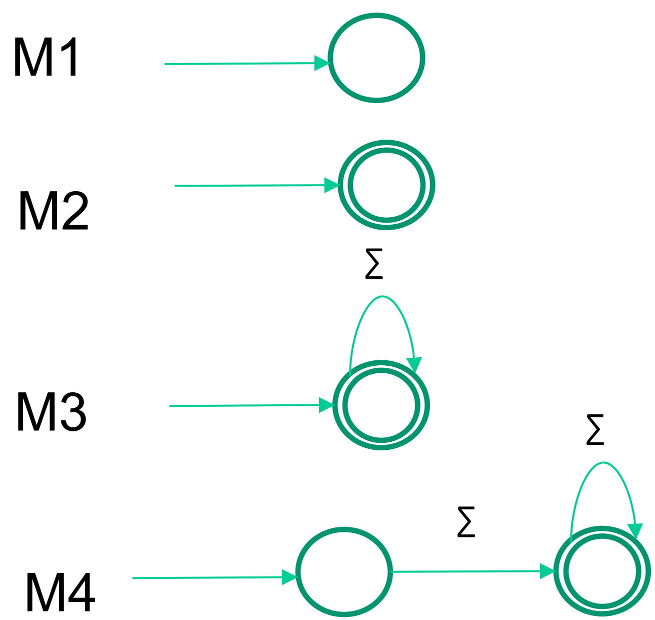






本 章 小 结

问题：判断下列**FA**是**DFA**，还是**NFA**？分别识别什么语言？



M1是NFA， $L(M1) = \Phi$

M2是NFA， $L(M2) = \{\varepsilon\}$

M3是DFA， $L(M3) = \Sigma^*$

M4是DFA， $L(M4) = \Sigma^+$

本 章 小 结

3. **FA**（**DFA/NFA**）接受的语言是正则语言（**Regular Language**）。

4. 如果两个**FA**接受的语言相同，那么，这两个**FA**是等价的。

5. 将**NFA**转化为**DFA**的方法有两种，即子集构造法和基于定理**3.1**的方法，两者是等价的。

6. **Moore**机和**Mealy**是带输出的有穷自动机。