Lecture 4. Finite Automata (Finite-state machines)

Hà Chí Trung, BM KHMT, KCNTT, HVKTQS

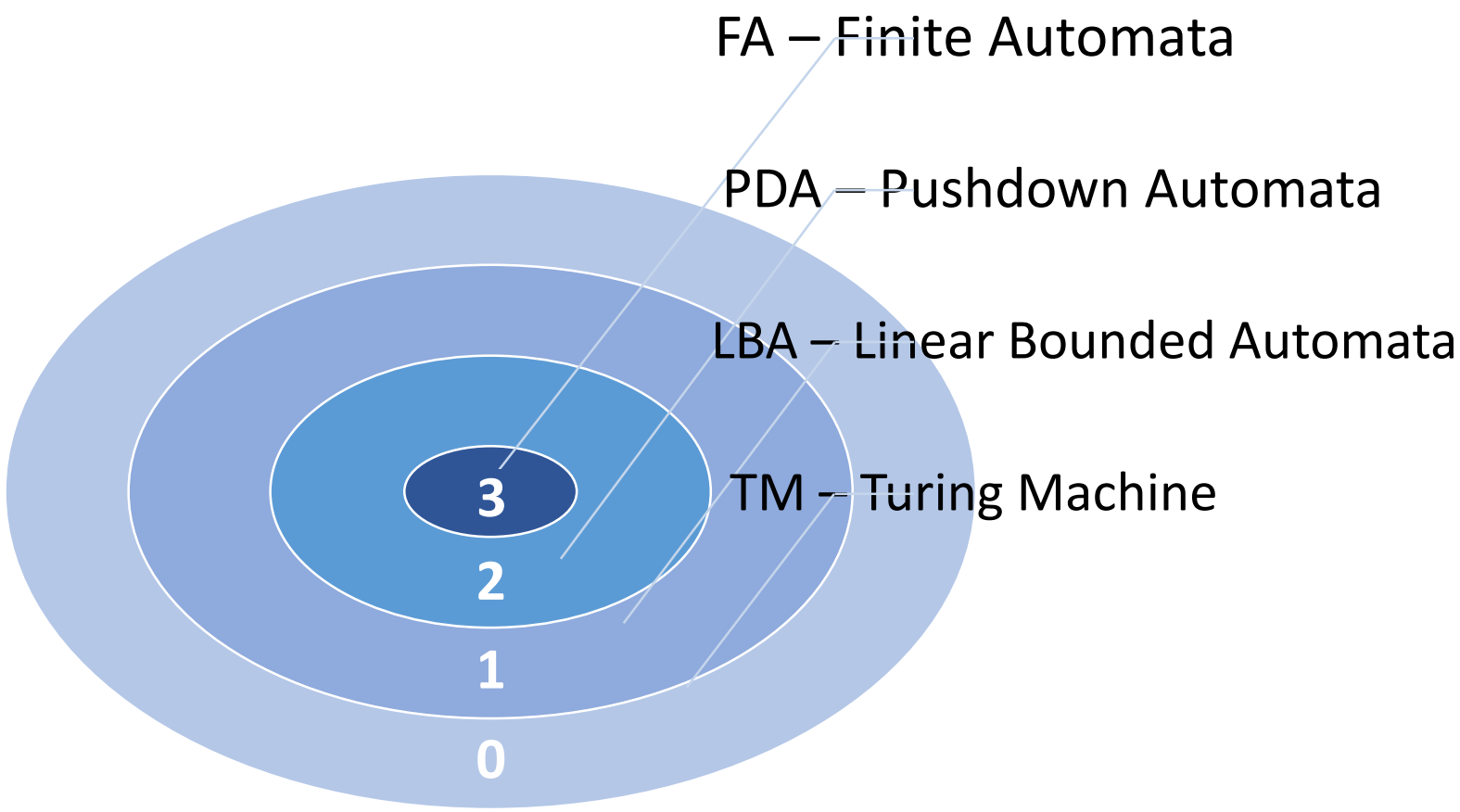
hct2009@yahoo.com

01685-582-102

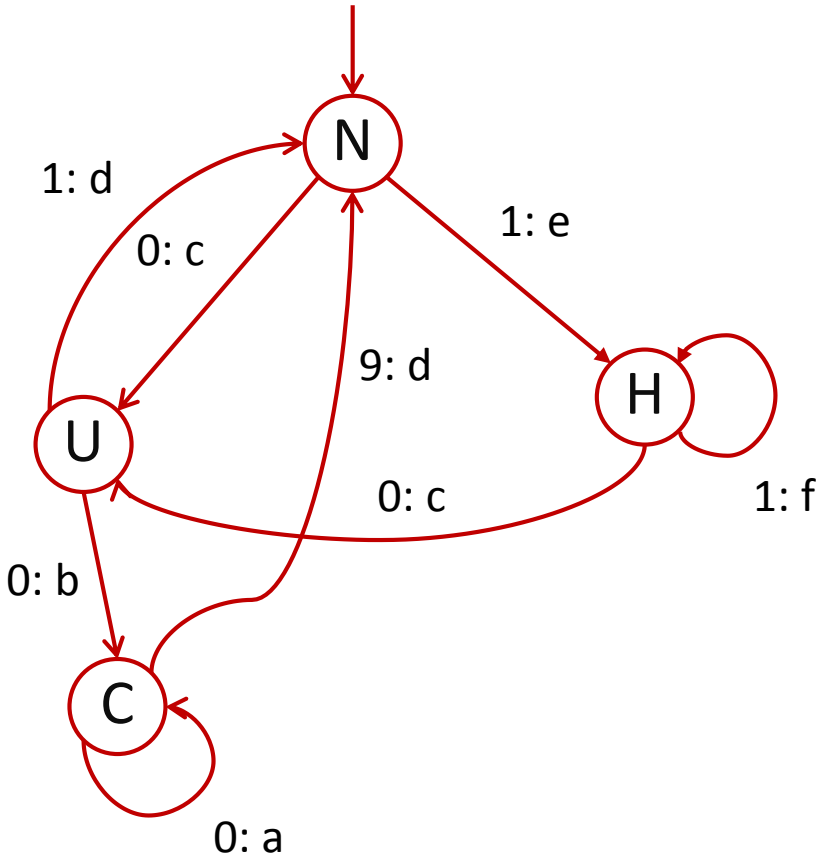
Finite Automata (Finite-state machines)

1. **Finite-State Machines with Output**
2. **Finite-State Machines with No Output**
    - **2.1. Deterministic Finite Automata**
    - **2.2. Nondeterministic Finite Automata**
    - **2.3. Finite Automata with ε-transitions**
    - **2.4. Equivalence of DFA, NFA, NFAε**
    - **2.5. Conversion algorithms with FA**

Automata


FA – Finite Automata
PDA – Pushdown Automata
LBA – Linear Bounded Automata
TM – Turing Machine
3
2
1
0

1. Finite-State Machines with Output



- **An đi học…**

  01011000…

  01110…

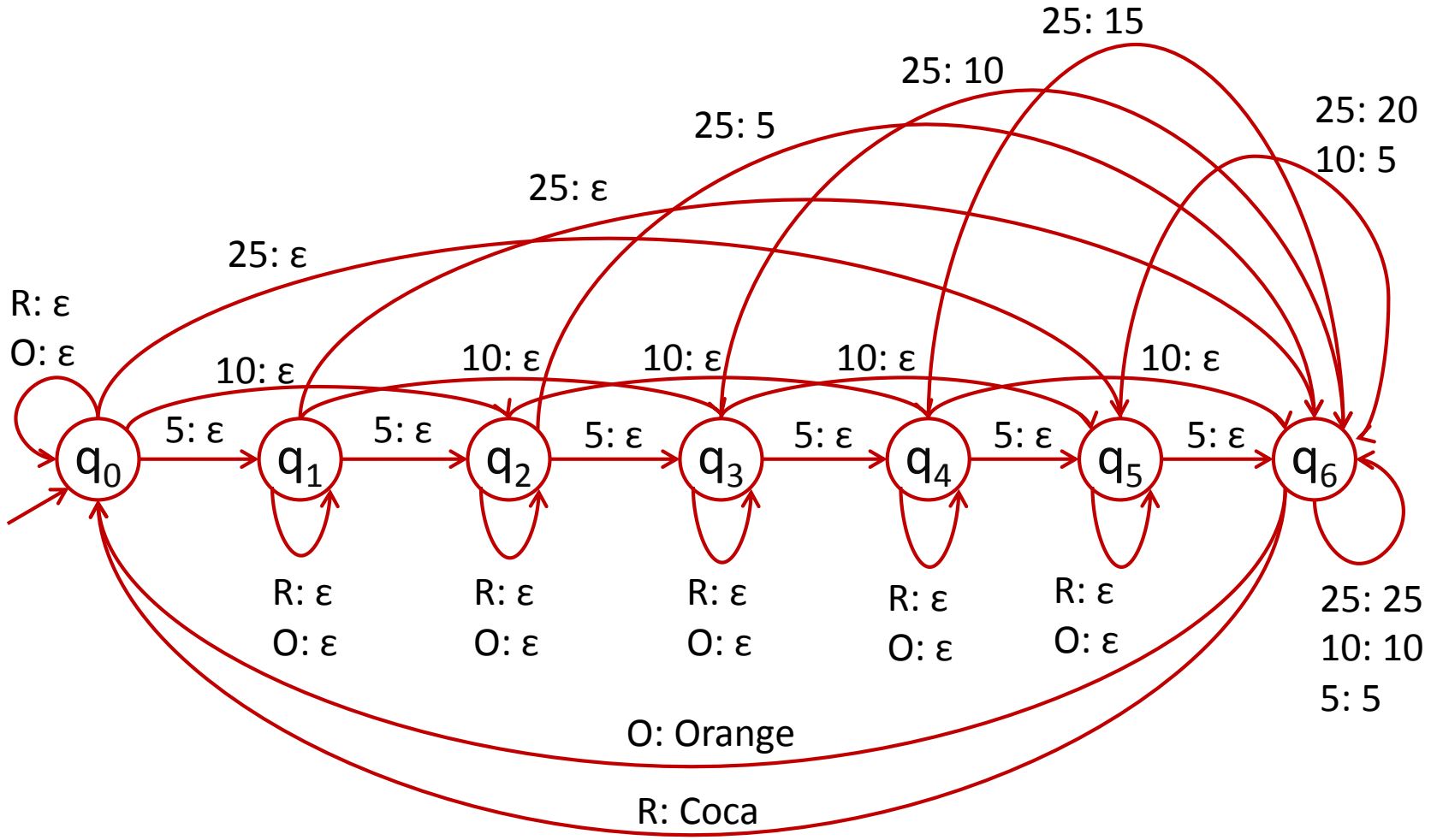
- ***Bố An (Bình):***

a = ”cầm roi”; b= “mắng chửi”; c = “dỗ dành”; d = “hy vọng”; e = “vui sướng”; f = “khen ngợi”.

Kenneth H. Rosen (Ch.13)

- **Ví dụ:** Thiết kế một máy bán hàng tự động đáp ứng các yêu cầu sau:
- Máy bán nước Cocacola và nước cam, cùng trị giá 30 xu/lon;
- Máy chấp nhận các đồng 5 xu, 10 xu, 25 xu;
- Khi thả vào máy lớn hơn 30 xu, máy lập tức trả lại số tiền vượt quá, khi người dùng nhấn nút màu đỏ (R) sẽ nhận được nước Coca, nhấn nút màu cam (O) sẽ nhận được nước cam;

Kenneth H. Rosen (Ch.13)



Kenneth H. Rosen (Ch.13)

- **Máy bán hàng tự động**

	Next State					Output				
	Input					*Input*				
State	**5**	**10**	**25**	***O***	***R***	**5**	**10**	**25**	***O***	***R***
s_0	s_1	s_2	s_5	s_0	s_0	n	n	n	n	n
s_1	s_2	s_3	s_6	s_1	s_1	n	n	n	n	n
s_2	s_3	s_4	s_6	s_2	s_2	n	n	5	n	n
s_3	s_4	s_5	s_6	s_3	s_3	n	n	10	n	n
s_4	s_5	s_6	s_6	s_4	s_4	n	n	15	n	n
s_5	s_6	s_6	s_6	s_5	s_5	n	5	20	n	n
s_6	s_6	s_6	s_6	s_0	s_0	5	10	25	OJ	AJ

1. Finite-State Machines with Output

- **Transducers:**
  - **Mealy machine** (Developed by G.H. Mealy in 1955)
  - **Moore machine** (Developed by E.F. Moore in 1956)
- **Recognizers**: FA (classifiers)

1. Finite-State Machines with Output

- **Mealy machine** là bộ 6:

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

trong đó:

Σ : tập hữu hạn gọi là tập các kí hiệu vào;

Δ : tập hữu hạn gọi là tập các kí hiệu ra;

Q : tập hữu hạn gọi là tập các trạng thái;

q_0 : trạng thái bắt đầu ;

δ : ánh xạ từ $Q \times \Sigma$ sang Q (gọi là hàm chuyển trạng thái);

ϕ : là ánh xạ từ tập $Q \times \Sigma$ sang tập Δ (gọi là hàm ra).

$$\Delta(t) = \phi[q(t), x(t)]$$

- Automat Mealy hoạt động theo cơ chế phát tín hiệu ra phụ thuộc vào tín hiệu vào và trạng thái nhận tín hiệu.

1. Finite-State Machines with Output

- **Moore machine** là bộ 6:

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

trong đó:

Σ : tập hữu hạn gọi là tập các kí hiệu vào;

Δ : tập hữu hạn gọi là tập các kí hiệu ra;

Q : tập hữu hạn gọi là tập các trạng thái;

q_0 : trạng thái bắt đầu;

δ : ánh xạ từ Q x Σ sang Q (gọi là hàm chuyển trạng thái);

ϕ : là ánh xạ từ tập Q x Σ sang tập Δ (gọi là hàm ra).

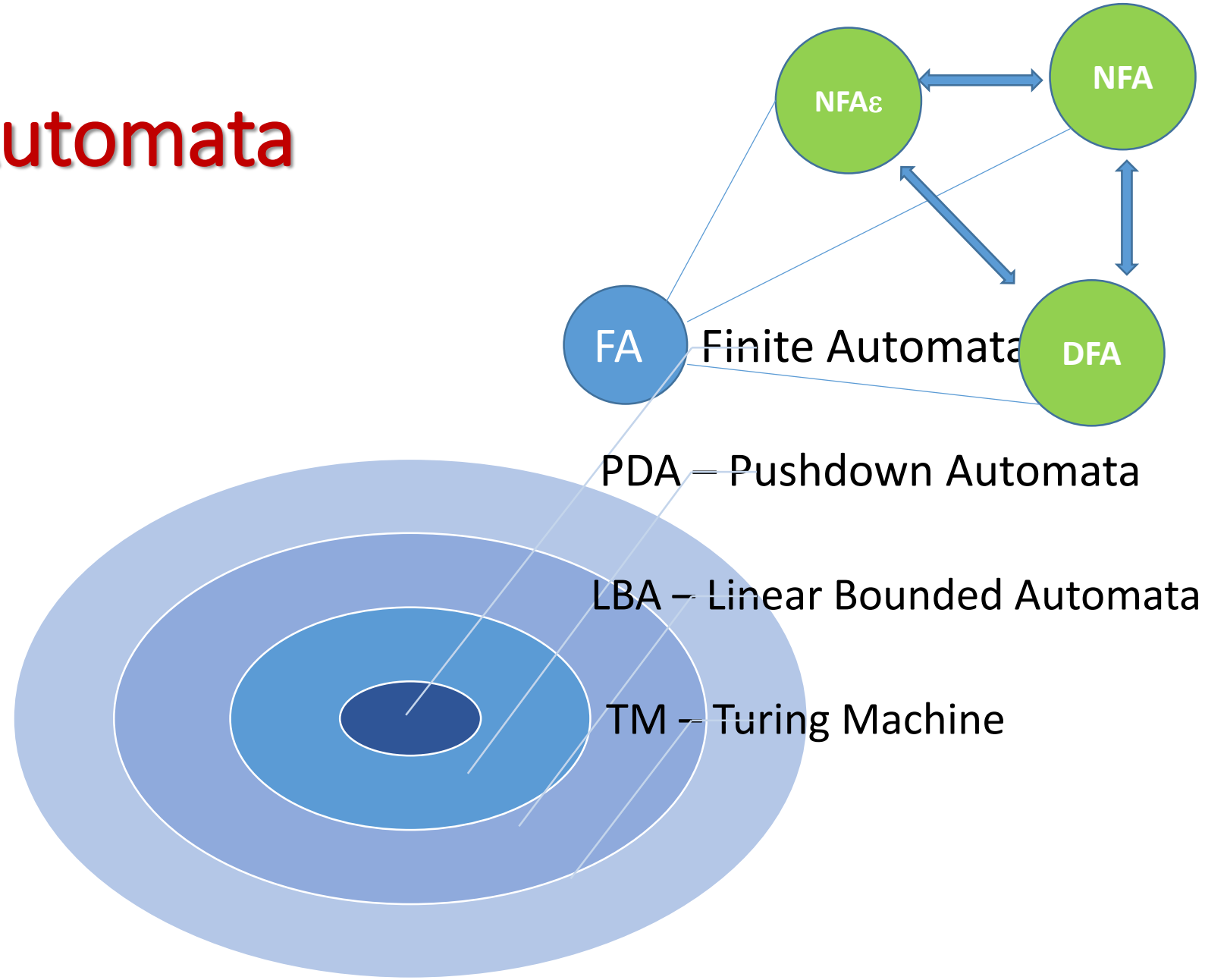
$$\Delta(t) = \phi[q(t)]$$

- Automat Moore hoạt động theo cơ chế phát tín hiệu ra chỉ phụ thuộc trạng thái.

2. Finite-State Machines with No Output

- **Finite Automata**
  - **NFAε:** Nondeterministic Finite Automata with ε-transitions
  - **NFA:** Nondeterministic Finite Automata
  - **DFA:** Deterministic Finite Automata

2. Finite Automata



2. Finite Automata

DFA:

$$A=<Q, \Sigma, \delta, q_0, F>,$$

where:

Q : **states** (p, q…);

Σ : finite set of *symbols*, called the *alphabet* (a, b, c …);

δ : $Q \times \Sigma \rightarrow Q$, **transition function,**

$\delta(p, a) = q$ or $\delta(p, a) = \emptyset$, $p, q \in Q$, $a \in \Sigma$;

$\mathbf{q_0} \in Q$: **start state**;

$\mathbf{F} \subseteq Q$: **final (accept) states**.

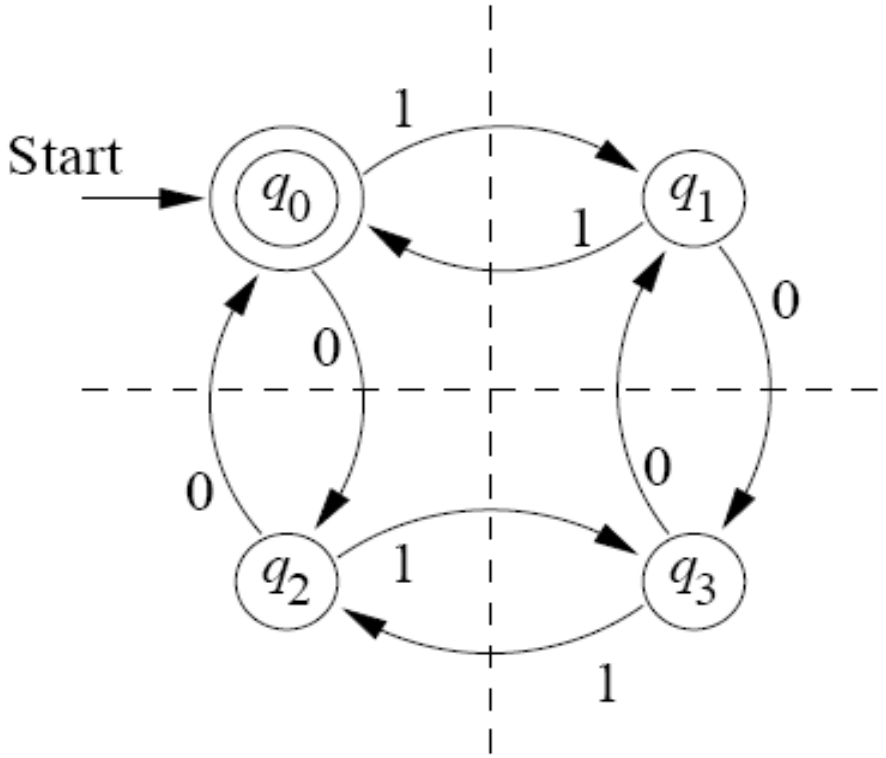
NFA: $\delta: Q \times \Sigma \rightarrow 2^Q$

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

2. Finite Automata

- **Đoán nhận ngôn ngữ**: xét xem từ nào thuộc về ngôn ngữ đó.
  - Hoạt động của automata bắt đầu từ một trạng thái đặc biệt, trang thái đầu tiên (**start state**);
  - Giả sử rằng tại mỗi thời điểm (**step**, **time step**), automata đang ở một trạng thái nào đó (**current state**), đầu vào của automata đón nhận một ký tự của chuỗi cần xử lý, dưới tác động của ký tự đó automata chuyển sang một trạng thái kế tiếp (**next state**);
  - Chuỗi được đoán nhận nếu trạng thái cuối cùng của automata rơi vào một trong các trạng thái kết thúc (**finish states**).

2.1. Deterministic Finite Automata



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

2.1. Deterministic Finite Automata

- **Thuật toán đoán nhận chuỗi**:

```
q:= q0;
c:= getnextchar();
while (C < > ε) do {
        q:= δ(q, c);
        c:= getnextchar();
};
if (q in F) return (true);
else return (false);
```

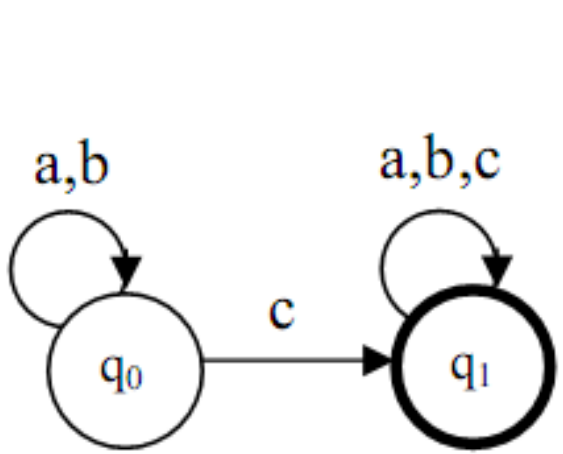
2.1. Deterministic Finite Automata

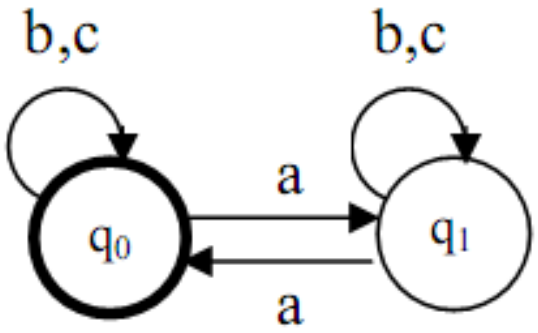
- **Hàm chuyển trạng thái mở rộng δ'**:
  1. δ'(q, ε) = q;
  2. δ'(q, wa) = δ'(δ'(q,w), a) với ∀ w, a.
- Chuỗi ω = $a_1a_2...a_n$ được đoán nhận bởi DFA A (có nghĩa là $\delta'(q_0, \omega) \in F$) nếu tồn tại một đường đi bắt đầu từ trạng thái đầu q_0 và kết thúc ở trạng thái kết với dãy nhãn của đường đi là $a_1a_2...a_n$.
- Ngôn ngữ được chấp nhận bởi một automata A:

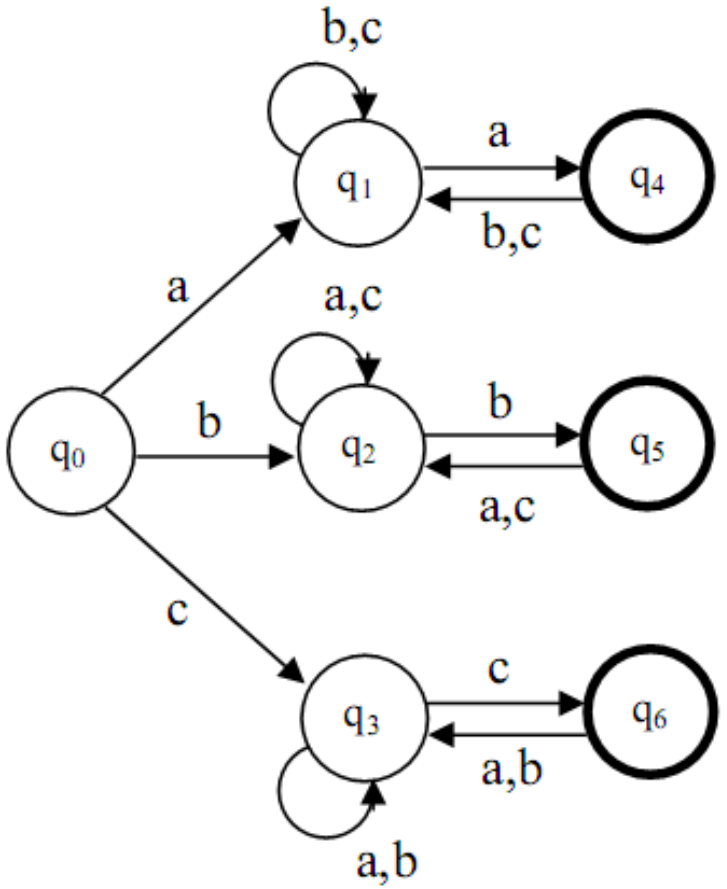
$$L(A) = \{ x \mid \delta'(q_0, x) \in F \}$$

- Các ngôn ngữ chấp nhận bởi automata hữu hạn đơn định được gọi là **ngôn ngữ chính quy**.

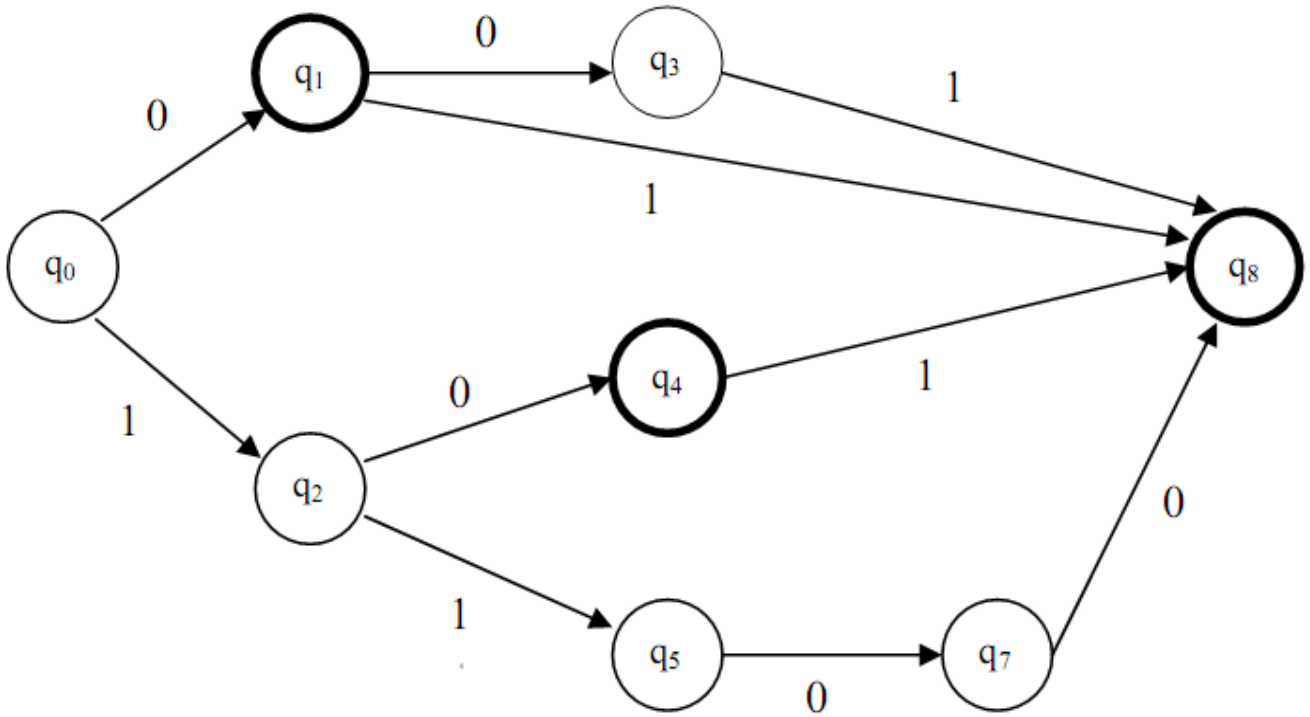
2.1. Deterministic Finite Automata



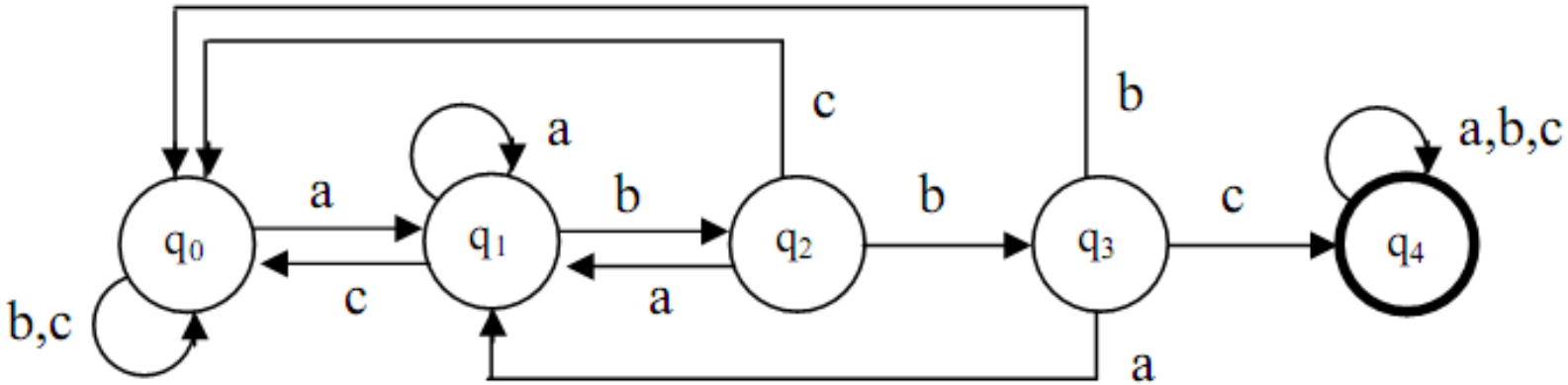




2.1. Deterministic Finite Automata



2.1. Deterministic Finite Automata



2.2. Nondeterministic Finite Automata

NFA:

$$A=<Q, \Sigma, \delta, q_0, F>,$$

where:

Q : **states** (p, q…);

Σ : finite set of *symbols*, called the *alphabet* (a, b, c …);

δ : $Q\times\Sigma \rightarrow 2^Q$, **transition function,**

$\delta(p, a) = P$ or $\delta(p, a) = \emptyset$, $p \in Q$, $\mathbf{P} \subseteq Q$, $a \in \Sigma$;

$\mathbf{q_0} \in Q$: **start state**;

$\mathbf{F} \subseteq Q$: **finish (accept) states**.

2.2. Nondeterministic Finite Automata

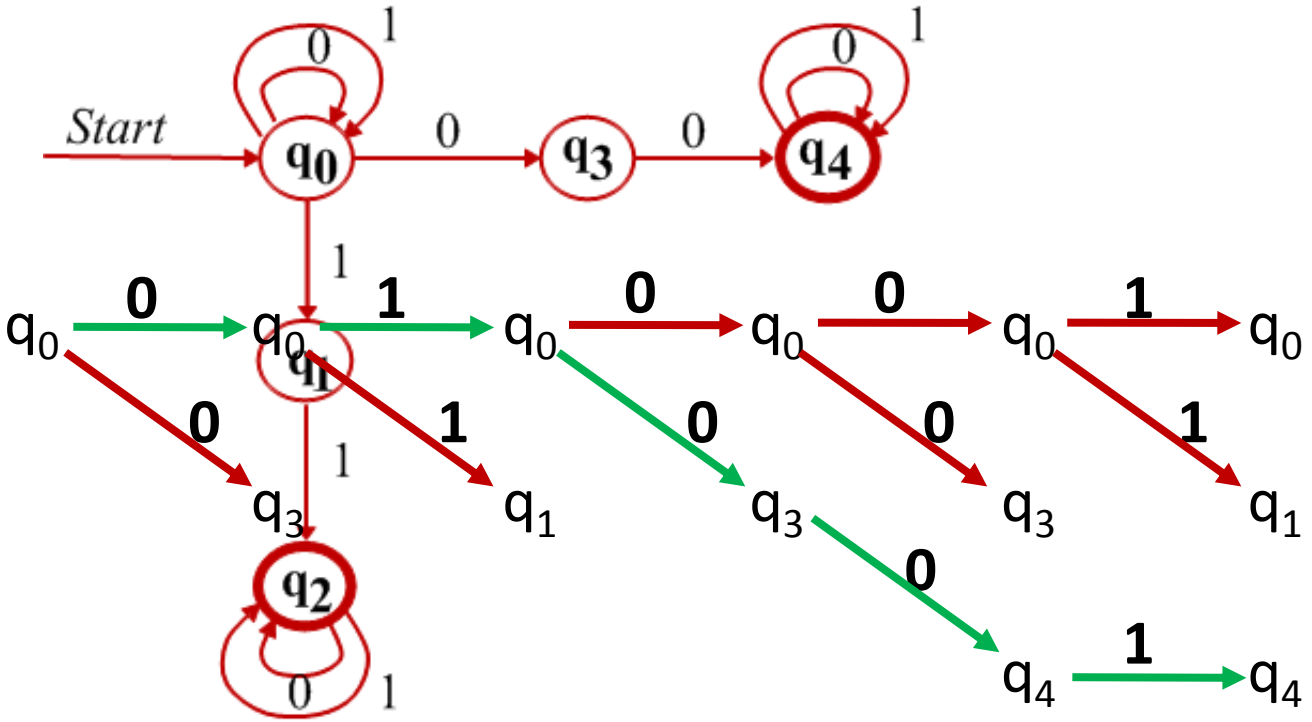
- **Chú ý**: khái niệm **δ(q, a)** là *tập hợp tất cả các trạng thái p* sao cho có phép chuyển từ trạng thái q trên nhãn a.
- **Hàm chuyển trạng thái mở rộng** (của NFA):
  1. $\delta(q, \varepsilon) = \{q\}$
  2. $\delta(q, wa) = \{ p \mid r \text{ trong } \delta(q, w) \text{ mà } p \in \delta(r, a) \}$

     $= \delta(\delta(q,w), a)$
  3. $\delta(P, w) = \cup_{q \in P} \delta(q, w)$ với $\forall P \subseteq Q$
- Ngôn ngữ L(A), với A là NFA $(Q, \Sigma, \delta, q_0, F)$ là tập hợp:

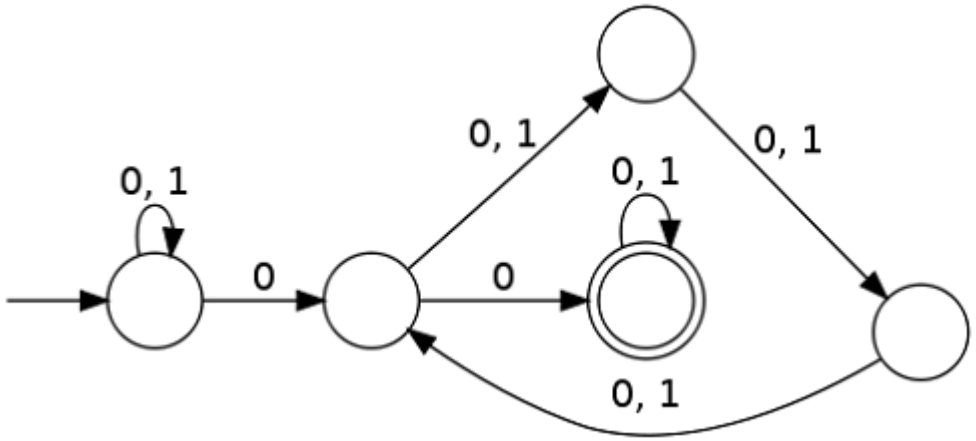
  $L(A) = \{w \mid \delta(q_0, w) \text{ có chứa một trạng thái trong } F\}$

2.2. Nondeterministic Finite Automata

- **Ví dụ:** Cho automaton (hình vẽ) và xét chuỗi nhập **01001**

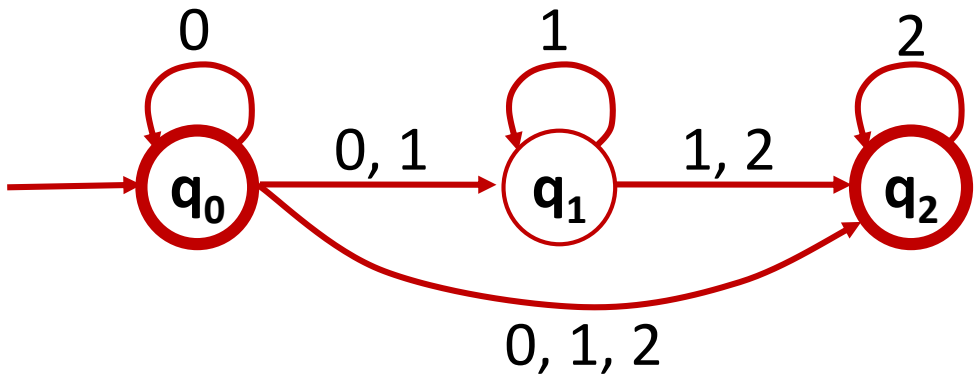


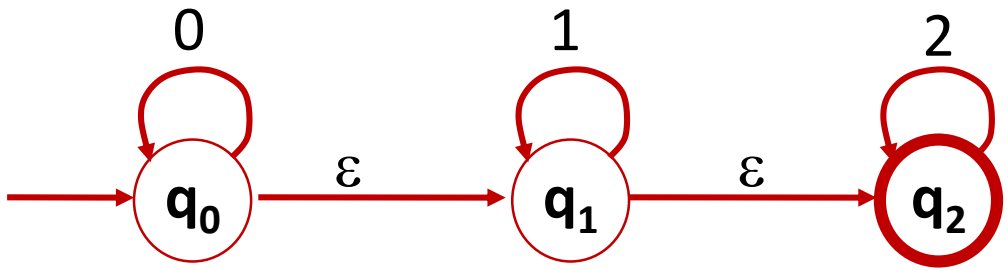
2.2. Nondeterministic Finite Automata



2.3. Finite Automata with ε-transitions (NFAε)

- **Ví dụ:** xây dựng NFA chấp nhận chuỗi $0^*1^*2^*$





2.3. Finite Automata with ε-transitions (NFAε)

NFAε:

$$A=<Q, \Sigma, \delta, q_0, F>,$$

where:

Q : **states** (p, q…);

Σ : finite set of *symbols*, called the *alphabet* (a, b, c …);

δ : $Q\times(\Sigma \cup\{\varepsilon\}) \rightarrow 2^Q$, **transition function,**

$\delta(p, a) = P$ or $\delta(p, a) = \emptyset$, $p \in Q$, $\mathbf{P} \subseteq Q$, $a \in \Sigma$;

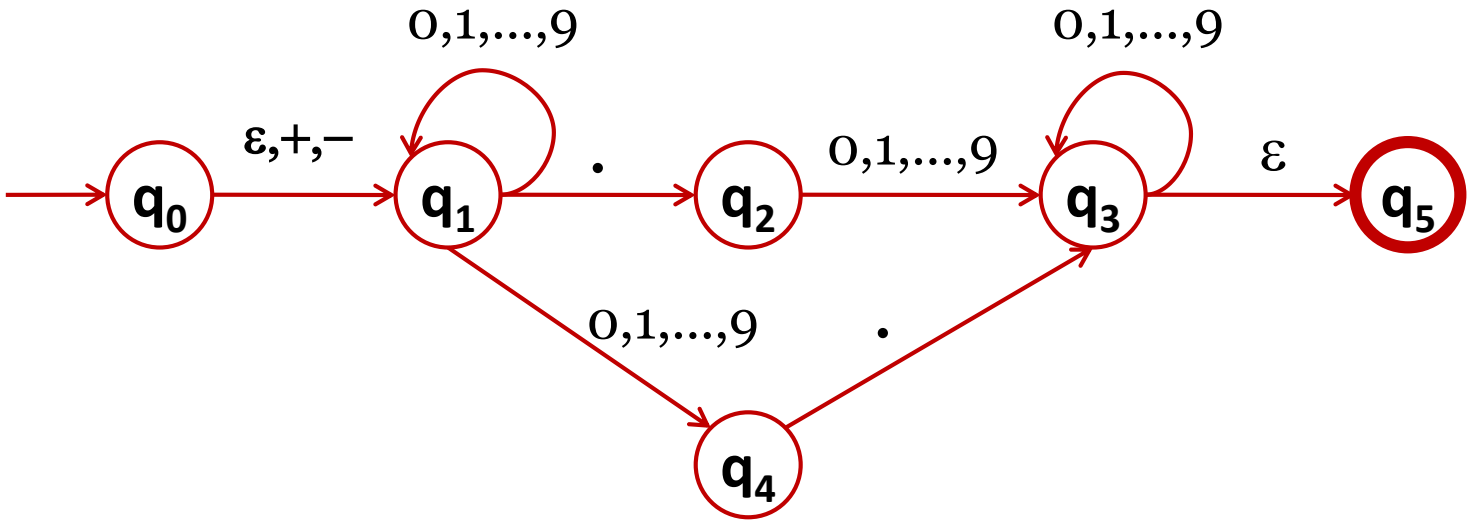
$\mathbf{q_0} \in Q$: **start state**;

$\mathbf{F} \subseteq Q$: **finish (accept) states**.

2.3. Finite Automata with ε-transitions (NFAε)

Ví dụ: Automaton biểu diễn các số thực.

1. Có thể có một dấu + hoặc - ở đầu.
2. Sau đó là một xâu ký tự "0",..."9"
3. Dấu chấm.
4. Một xâu ký tự "0",..."9" khác.



2.3. Finite Automata with ε-transitions (NFAε)

- **Hàm chuyển trạng thái mở rộng (của NFAε):** mở rộng **δ** thành **δ***
  1. **δ*** **:** Q x **Σ*** → 2^Q
  2. **δ***(q, w) = { p | có đường đi từ q tới p theo nhãn **w**, trên đường đi có thể chứa cạnh nhãn **ε** }
- **Bao đóng ε (ký hiệu ε-CLOSURE(), ε*)** là tất cả các trạng thái có thể chuyển tới bằng dãy **{ε}***:
  1. **ε***(q) = { p | có đường đi từ q tới p theo nhãn **ε** }
  2. **ε***(P) = $\cup_{q \in P}$ **ε***(q)

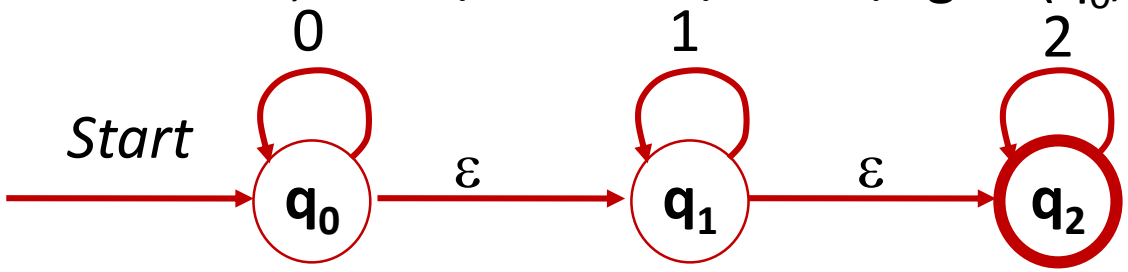
2.3. Finite Automata with ε-transitions (NFAε)

- **Ta có:**
  1. $\delta^*(q, \varepsilon) = \varepsilon^*(q)$
  2. $\delta^*(q,a) = \varepsilon^*(\delta(\delta^*(q, \varepsilon),a)) = \varepsilon^*(\delta(\varepsilon^*(q),a))$
  3. $\delta^*(q, wa) = \varepsilon^*(\delta(\delta^*(q, w), a))$ hoặc

     $\delta^*(q, wa) = \varepsilon^*(P)$ với $P = \{ p \mid r \in \delta^*(q, w), p \in \delta(r, a)\}$
  4. $\delta^*(R, w) = \cup_{q \in R} \delta^*(q, w)$

2.3. Finite Automata with ε-transitions (NFAε)

- **Ví dụ 3.12**: Cho automata sau, xác định ánh xạ mở rộng $\delta^*(q_0, 012)$



- **w = 012**

  $\delta^*(q_0, \varepsilon) = \varepsilon^*(q_0) = \{q_0, q_1, q_2\}$

  $\delta^*(q_0, \mathbf{0}) = \varepsilon^* (\delta(\delta^*(q_0, \varepsilon), 0)) = \varepsilon^*(\delta(\{q_0, q_1, q_2\}, 0)) = \varepsilon^*(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$

  $= \varepsilon^*(\{q_0\} \cup \varnothing \cup \varnothing) = \varepsilon^*(\{q_0\}) = \{q_0, q_1, q_2\}$

  $\delta^*(q_0, \mathbf{01}) = \varepsilon^*(\delta(\delta^*(q_0, 0), 1)) = \varepsilon^*(\delta(\{q_0, q_1, q_2\}, 1)) = \varepsilon^*(\{q_1\}) = \{q_1,q_2\}$

  $\delta^*(q_0, \mathbf{012}) = \varepsilon^*(\delta(\delta^*(q_0, 01), 2)) = \varepsilon^*(\delta(\{q_1, q_2\}, 2)) = \varepsilon^*(\{q_2\}) = \{q_2\}$

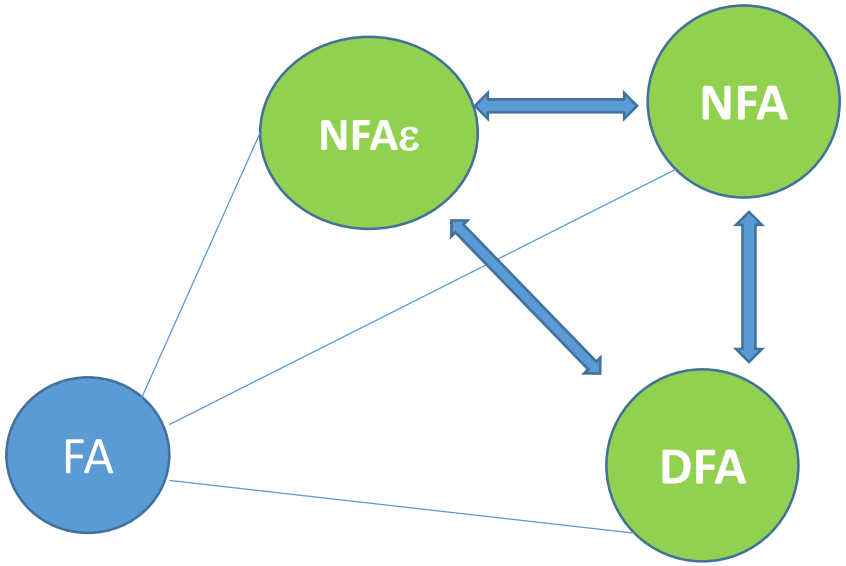
- Do $\mathbf{q_2 \in F}$ nên $w \in L(A)$.

2.3. Finite Automata with ε-transitions (NFAε)

- **Giải thuật đoán nhận chuỗi của NFAε**
  - **Input:** chuỗi nhập **x$**
  - **Output:** true(x được chấp nhận) hoặc false.

```
P = ε*(q0);
c = nextchar;
while (c <> $)
{
        P = ε* (δ(P, c));
        c = nextchar;
}
if (p ∈ P in F)return true;
else return false;
```

2.4. Equivalence of DFA, NFA, NFAε



2.4. Equivalence of DFA, NFA, NFAε

- **Định lý 1:** Nếu L là tập được chấp nhận bởi một NFA thì tồn tại một DFA chấp nhận L.
- **Định lý 2:** Nếu L được chấp nhận bởi một NFAε thì L cũng được chấp nhận bởi một NFA không có ε-dịch chuyển.

- **Hệ quả:** Nếu L là tập hợp được chấp nhận bởi một NFAε thì cũng tồn tại một DFA chấp nhận L.

2.4. Equivalence of DFA, NFA, NFAε

- **Giải thuật tổng quát xây dựng DFA từ NFA:**

  Giả sử NFA A={Q, Σ, δ, q_0, F} chấp nhận L, giải thuật xây dựng DFA A'={Q', Σ, δ', q_0', F'} chấp nhận L như sau:

  - Q' = 2^Q , phần tử trong Q' được ký hiệu là [q_0, q_1, …, q_i] với q_0, q_1, …, q_i ∈ Q;
  - q_0' = [q_0];
  - F' là tập hợp các trạng thái của Q' **có chứa ít nhất một trạng thái kết thúc** trong tập F của A;
  - Hàm chuyển δ'([q_1, q_2,..., q_i], a) = [p_1, p_2,..., p_j] nếu và chỉ nếu δ({q_1, q_2,..., q_i }, a) = {p_1, p_2,..., p_j}.
  - Đổi tên các trạng thái [q_0, q_1, …, q_i].

2.4. Equivalence of DFA, NFA, NFAε

- Giải thuật xây dựng NFA từ NFAε:
- **Giả sử ta có NFAε** A(Q, Σ, δ, q_0, F) chấp nhận L, ta xây dựng: **NFA** A'={Q, Σ, **δ'**, q_0, **F'**} như sau:
  1. **F' = F ∪ q_0** nếu **ε*(q_0)** chứa ít nhất một trạng thái thuộc F. Ngược lại, **F' = F;**
  2. **δ'**(q, a) = **δ***(q, a).

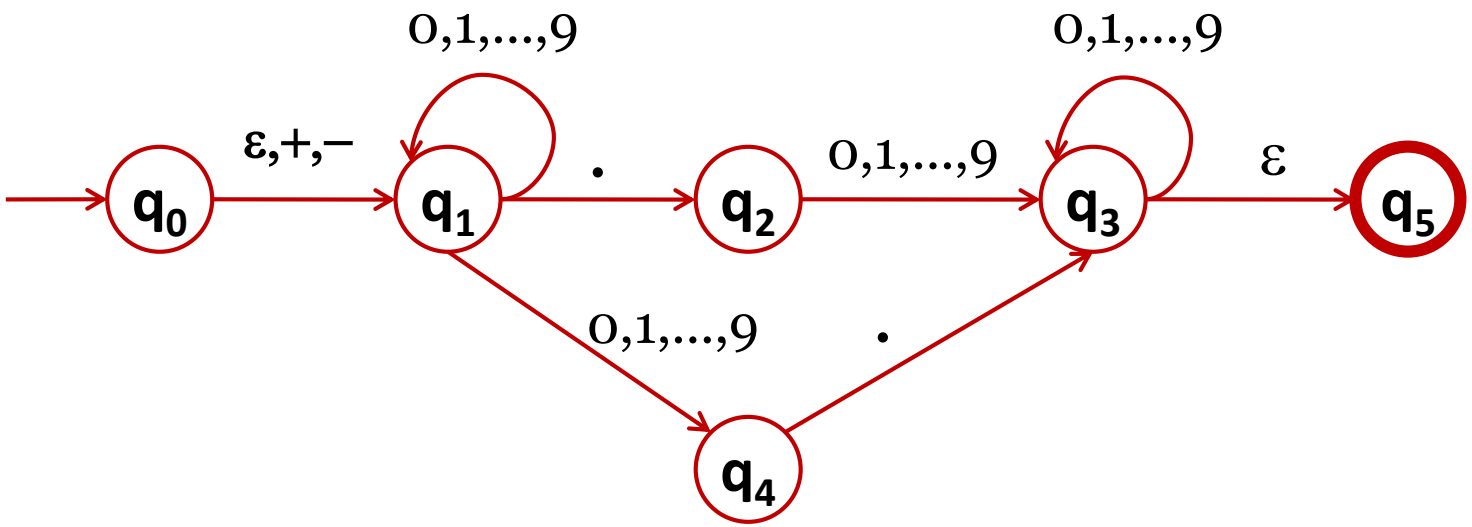
2.4. Equivalence of DFA, NFA, NFAε

- **Giải thuật xây dựng DFA từ NFAε:**

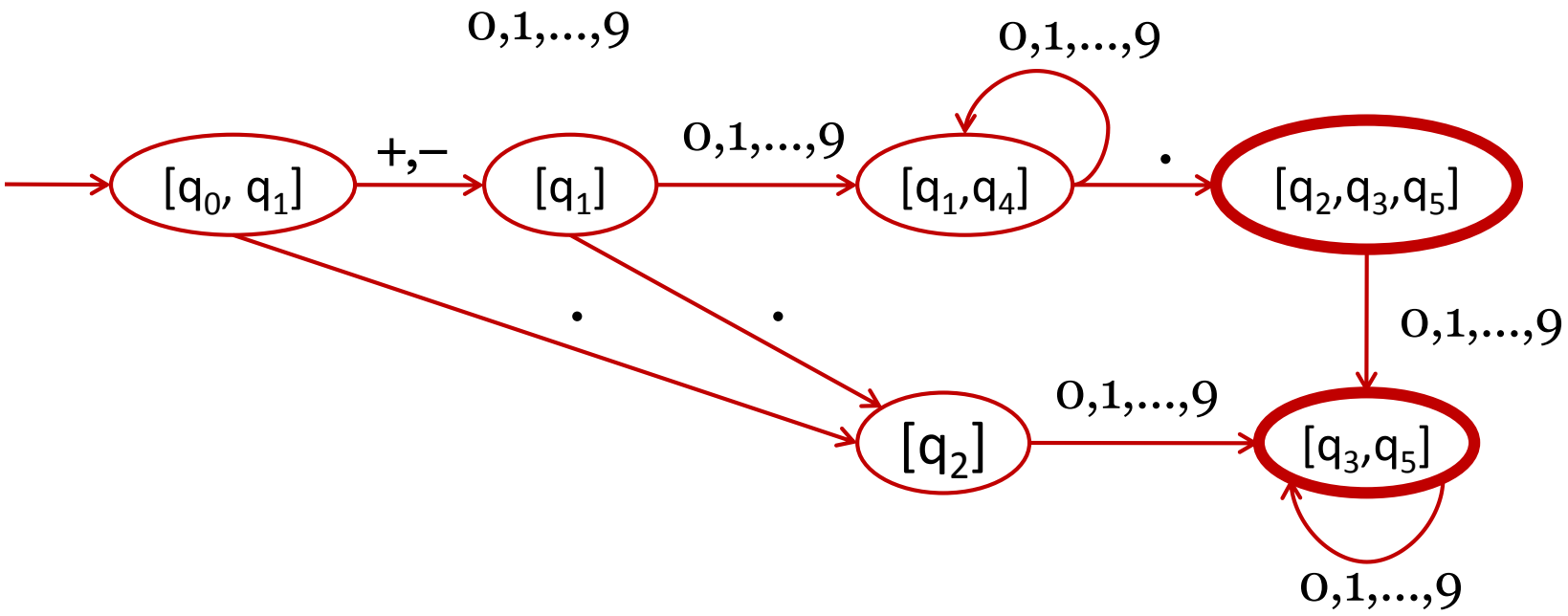
```
Tìm kiếm T = ε*(q0) ; T chưa được đánh dấu;
Thêm T vào tập Q' (of DFA);
while (xét trạng thái T ∈ Q' chưa đánh dấu){
    Đánh dấu T;
    forearch (với mỗi ký hiệu nhập a){
        U:= ε*(δ(T,a));
        if(U không thuộc tập trạng thái Q'){
            add U to Q';
            Trạng thái U chưa được đánh dấu;
        }
        δ[T,a]= U;
    }
}
```

2.4. Equivalence of DFA, NFA, NFAε

- **Ví dụ:** xây dựng **DFA** tương đương cho **NFA** sau:



2.4. Equivalence of DFA, NFA, NFAε



2.5. Conversion algorithms with FA

- Biến đổi trên 1 automaton (các phép toán 1 tập hợp chính quy):
  - phép lặp, lặp cắt, phần bù, đảo ngược…
- Biến đổi trên nhiều automata (các phép toán trên các tập hợp chính quy):
  - Cắt trái, cắt phải, phép hợp (∪), phép giao (∩),…
- *Chú ý: tự đọc*