

Chap. 2 Finite Automata

2.1 An Informal Picture of Finite Automata(See 2nd Edition)

A man with a wolf, goat, and cabbage is on the left bank of a river

A boat carries one man and only one of the other three.

The wolf eats the goat without the man.

The goat eats the cabbage without the man.

Is it possible to cross the river without the goat or cabbage being eaten?

states: occupants of each bank after a crossing

16 subsets of the man(m), wolf(W), goat(G), and cabbage(C).

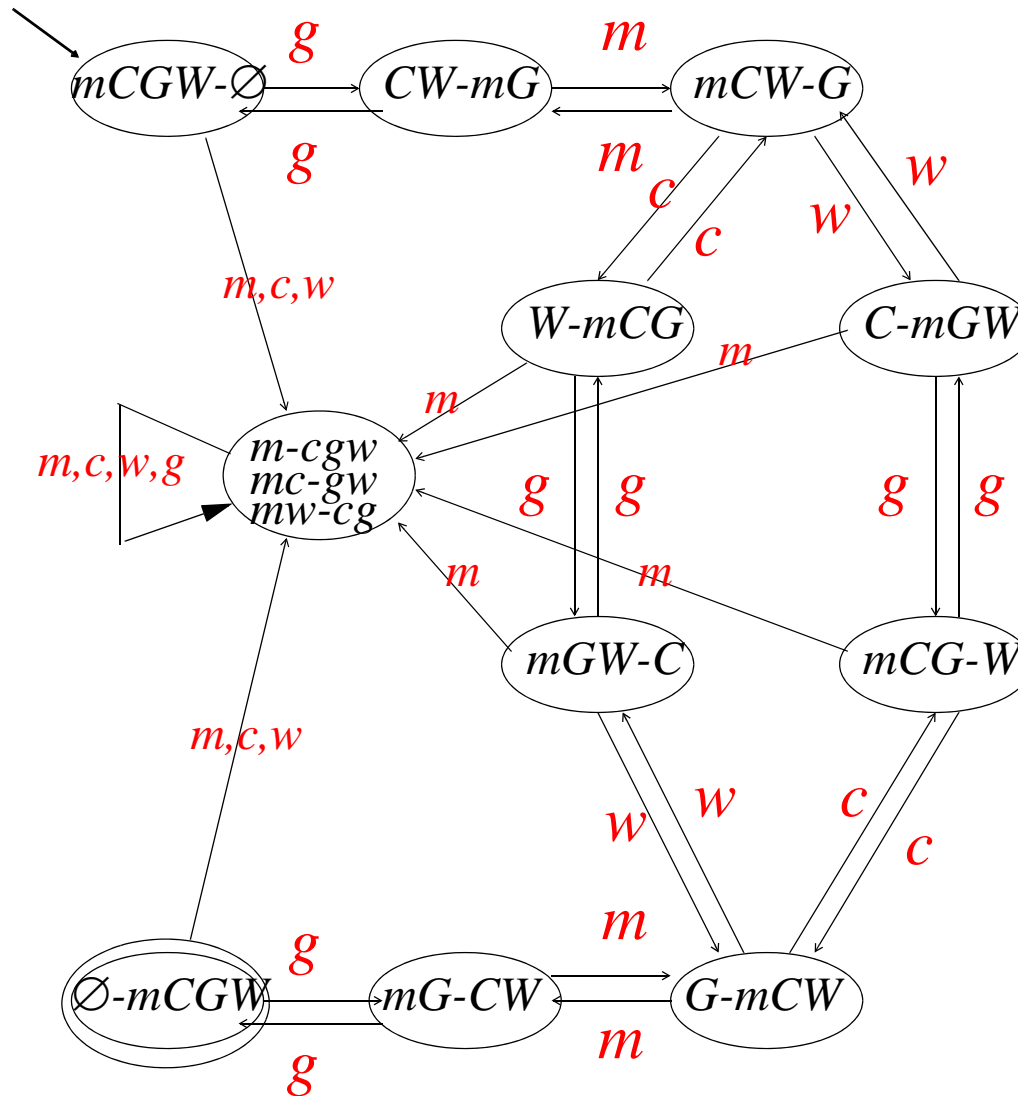
16 states: $mWGC-\emptyset$, $WC-mG$, ..., $\emptyset-mWGC$

input symbol: four ways of crossing the river

man cross alone(m) with the wolf(w), the goat(g), or cabbage(c)

input string: a sequence of crossings(input symbols)

state transition: From a state to another state by a input symbol



2.2 Deterministic Finite Automata(DFA)

2.2.1 Definition of a DFA

A *deterministic finite automaton* consists of:

1. A *finite set of states*, denoted Q ,
2. A *finite set of input symbols*, denoted Σ ,
3. A *transition function*,

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

Let $p, q \in Q$, and $a \in \Sigma$, if we write $\delta(q, a) = p$, meaning that when **current** state is in q and **input** symbol is a , the current state is changed to p , and input symbol is **changed** to the **next(right)** one.

4. A *start(initial) state*, $q_0 \in Q$, and
5. A *set of final or accepting states*, $F \subseteq Q$.

A DFA $A = (Q, \Sigma, \delta, q_0, F)$ “five-tuple” notation

2.2.2 How a DFA Processes Strings

How a DFA decides whether or not “accept”

a *input string*(sequence of input symbols)

Suppose $a_1a_2 \dots a_n$ is a sequence of input symbols

1. We start out the DFA with in its **start** state q_0 .
2. We consult the **transition function**, δ ,
with current state and current input symbol,
assume in the first trial $\delta(q_0, a_1) = q_1$,
3. **Repeat** this step with the **next** state q_1 and the **next** input symbol a_2 .
Assume $\delta(q_1, a_2) = q_2, \delta(q_2, a_3) = q_3, \dots, \delta(q_{n-1}, a_n) = q_n$.
4. **Finally**, check if $\delta(q_{n-1}, a_n) = q_n \in F$ or not.

If $q_n \in F$, **accept** $a_1a_2 \dots a_n$. If $q_n \notin F$ **does not accept** $a_1a_2 \dots a_n$.

$\delta(\delta(\dots\delta(\delta(q_0, a_1), a_2), \dots), a_{n-1}), a_n) = q_n$

where $\delta(q_{i-1}, a_i) = q_i$ for $1 \leq \forall i \leq n$.

function **DFA**(

$\underline{D} = (Q: \text{set of states}, \Sigma: \text{set of symbols}, \delta: Q \times \Sigma \rightarrow Q, q_0 \in Q, F \subseteq Q);$

$\underline{x} = a_1 a_2 \dots a_n \in \Sigma^*$) /* \underline{D} and $\underline{x} \in \Sigma^*$ are two formal parameters */

/* \underline{D} has five tuples, $(Q, \Sigma, \delta, q_0, F)$ */

return { \underline{D} accept \underline{x} , \underline{D} \neg accept \underline{x} }

$q \in Q, i \in \mathbb{N};$ /* q and i are two local variables */

$q, i := q_0, 1;$

do $i \leq n \rightarrow$ **if** $\exists \delta(q, a_i) \rightarrow q, i := \delta(q, a_i), i+1$

$/ \neg \exists \delta(q, a_i) \rightarrow$ **return** \underline{D} \neg accept \underline{x}

fi

od

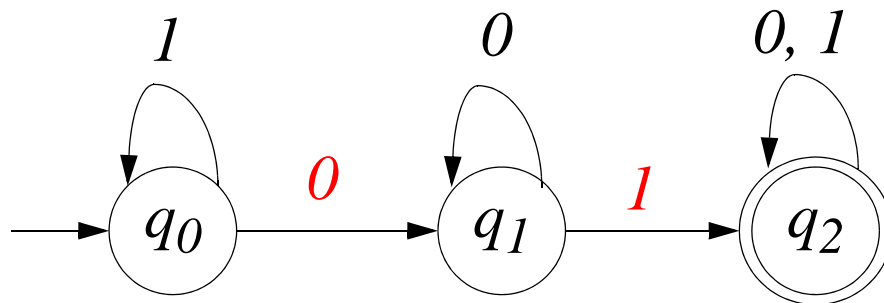
if $q \in F \rightarrow$ **return** \underline{D} accept \underline{x} | $q \notin F \rightarrow$ **return** \underline{D} \neg accept \underline{x} **fi**

end function **DFA**.

2.2.3 Simple notations for DFA's

Example 2.2 Let $\Sigma = \{0, 1\}$ and $L = \{x01y \in \Sigma^* \mid x, y \in \Sigma^*\}$.
 substring **01**. $(0 \mid 1)^*01(0 \mid 1)^*$.

1. **Transition diagram** Figure 2.4 in p. 48.



State q_0 : *initial state*

If it sees **0**, go to a **new state q_1** . If it sees **1** **stay** in the state q_0 .
 It has **never** seen 0.

State q_1 : *The last symbol it has seen is 0.*

If it sees **1**, go to a **new state q_2** . If it sees **0**, **stay** in the state q_1 .

State q_2 : *It has **ever** seen is 01. **final** state*

Whether it sees **0** or **1**, **stay** in the state q_2 .

2. Transition table Figure 2.5 in p. 49.

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

Formal definition of DFA D

$D = (Q, \Sigma, \delta, q_0, F)$ where

1. $Q = \{q_0, q_1, q_2\}$
2. $\Sigma = \{0, 1\}$
3. $\delta = \{\delta(q_0, 0)=q_1, \delta(q_0, 1)=q_0, \delta(q_1, 0)=q_1, \delta(q_1, 1)=q_2, \delta(q_2, 0)=q_2, \delta(q_2, 1)=q_2\}$
4. ?
5. $F = \{q_2\}$

2.2.4 Extend the domain of transition function from symbols to **strings**

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

$$\hat{\delta}: Q \times \Sigma^* \rightarrow Q$$

$$\hat{\delta}(q, \varepsilon) =_B q \quad \text{basis}$$

$$\hat{\delta}(q, xa) =_R \delta(\hat{\delta}(q, x), a) \quad \text{recursion} \quad (2.1)$$

$$q \in Q, x \in \Sigma^*, a \in \Sigma, xa \in \Sigma^+.$$

$$\hat{\delta}(q_0, a_1 a_2 \dots a_n) =_R \delta(\hat{\delta}(q_0, a_1 a_2 \dots a_{n-1}), a_n) \quad \text{1st recursion}$$

$$=_R \delta(\delta(\hat{\delta}(q_0, a_1 a_2 \dots a_{n-2}), a_{n-1}), a_n) \quad \text{2nd recursion}$$

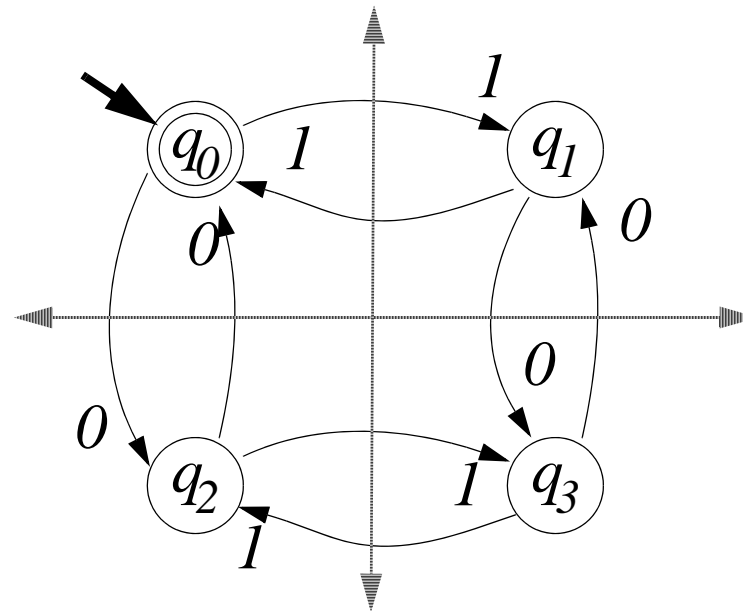
$$=_R \dots$$

$$=_R \delta(\delta(\dots \delta(\delta(\hat{\delta}(q_0, \varepsilon), a_1), a_2), \dots), a_{n-1}), a_n) \quad \text{n-th recursion}$$

$$=_B \delta(\delta(\dots \delta(\delta(q_0, a_1), a_2), \dots), a_{n-1}), a_n) \quad \text{basis}$$

We may use δ^* instead of $\hat{\delta}$ to show that the 2nd domain of δ is Σ^* .

Example 2.4 in p. 50.



Even numbers of 0's and even numbers of 1's.

2.2.5 The Language of DFA

We *define* the language of a DFA $D = (Q, \Sigma, \delta, q_0, F)$, $L(A)$,

$$L(D) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}.$$

$$L(A) \subseteq \Sigma^* \text{ or } L(A) \in 2^{\Sigma^*}.$$

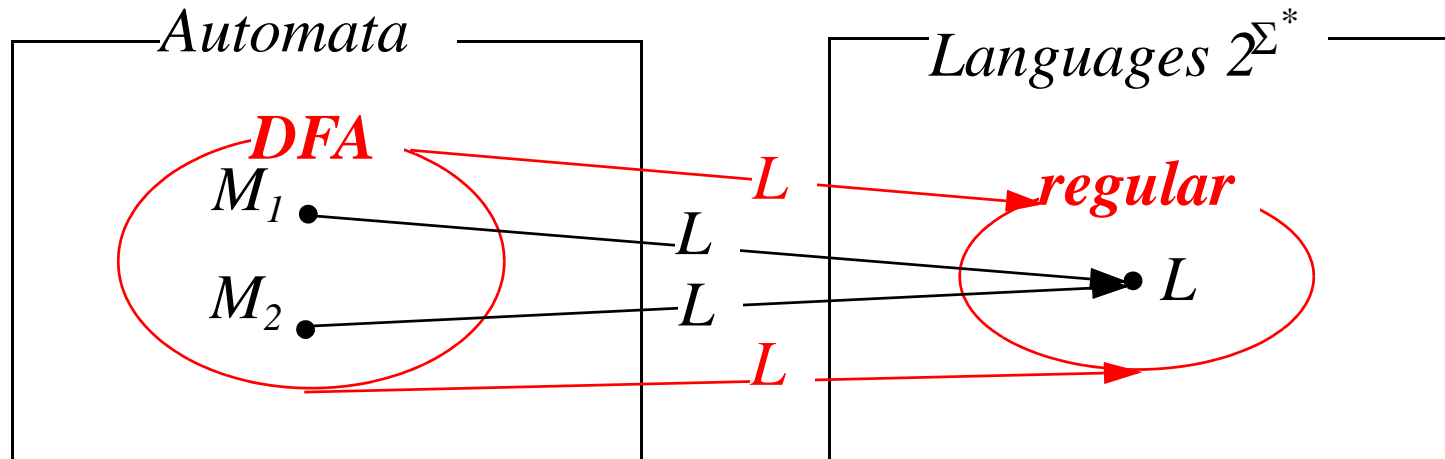
Def. A Class of Regular languages

Let L be a language.

If $L = L(A)$ for some **DFA** A , then we say L is a **regular language**.

A class of regular languages

type-3 languages in Chomsky's language hierarchy



Def. Equivalence of two Automata

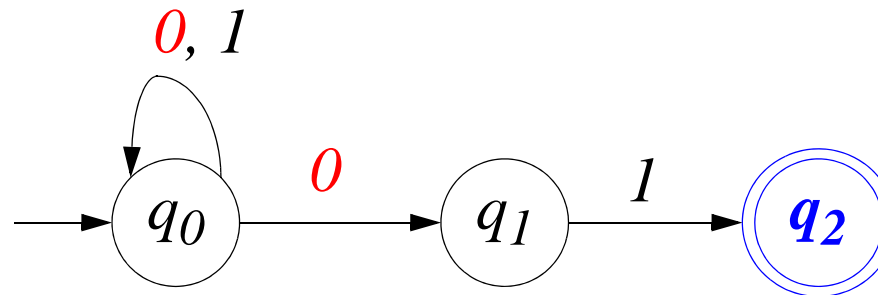
Let M_1 and M_2 be two automata. We say two automata M_1 and M_2 are **equivalent**, if $L(M_1) = L(M_2)$.

2.3 Nondeterministic Finite Automata(NFA)

2.3.1 An Informal View of NFA

Example 2.9 An NFA accepting all strings that ends in 01.

Fig. 2.9(p. 56)



Consider an input string $00101 \in \{0, 1\}^*$.

$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \notin F(\text{fail})$

$q_0 \xrightarrow{0} q_1 \xrightarrow{0} X(\text{stuck})$

$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} X(\text{stuck})$

$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \in F(\text{success})$

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

$$\forall q \in Q, \forall a \in \Sigma, \exists \delta(q, a) \in Q.$$

(total) function

$$\exists q \in Q, \exists a \in \Sigma .\exists. \nexists \delta(q, a).$$

partial function

If $A = (Q, \Sigma, \delta, q_0, F)$ be a FA with $\delta: Q \times \Sigma \dashrightarrow Q$ *partial function*,

$L(A) = L(B)$ for DFA $B = (Q \cup \{d\}, \Sigma, \delta', q_0, F)$ where $d \notin Q$,

$$\delta' = \delta \cup \{\delta(q, a) = d \mid \delta(q, a) \notin Q\} \cup \{\delta(d, a) = d \mid a \in \Sigma\}$$

d : *dead state*

B is the DFA (with *total function*) but?

Dead state:

“ $m-cgw, mc-gw, mw-cg$ ” state in the 1st Example.

eliminate the state and relevant transition functions \rightarrow *partial function*.

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

DFA but *partial function*: $\delta: Q \times \Sigma \rightarrow Q \cup \{\emptyset\}.$

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

2.3.2 Definition of NFA

$A = (Q, \Sigma, \delta, q_0, F)$ is a *nondeterministic finite automaton*(NFA), if

1. Q is a *finite set of states*,
2. Σ is a *finite set of input symbols*,
3. δ is a *transition function*,

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

If $q \in Q$, $a \in \Sigma$, and $\{p_1, p_2, \dots, p_n\} \subseteq Q$ (or $1 \leq \forall i \leq n, p_i \in Q$) then we write $\delta(q, a) = \{p_1, p_2, \dots, p_n\}$, meaning that when current state is in q and input symbol is a , the state may be changed to **any** one of p_1, p_2, \dots, p_n , and input symbol is changed to the next(right) one.

4. $q_0 \in Q$ is a *start state*, and
5. $F \subseteq Q$ is a set of *final or accepting states*.

2.3.3 The Extended Transition Function

DFA: $\delta: Q \times \Sigma \rightarrow Q$ *Definition*

$$\delta^*: Q \times \Sigma^* \rightarrow Q \qquad \hat{\delta}: Q \times \Sigma^* \rightarrow Q \text{ in the text}$$

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

$$\delta': 2^Q \times \Sigma \rightarrow 2^Q \qquad \text{1st extension in this lecture}$$

$$\delta'^*: 2^Q \times \Sigma^* \rightarrow 2^Q \qquad \text{2nd in this lecture}$$

$$\hat{\delta}: Q \times \Sigma^* \rightarrow 2^Q \text{ in the text}$$

Two step extension

1. Set extension: $\delta \Rightarrow \delta'$

We extend the 1st domain of δ' to set of states.

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

Let $P \subseteq Q$. Then we define

$$\delta'(P, a) = \{q \in Q \mid p \in P, q \in \delta(p, a)\}$$

$$\text{or} \quad = \bigcup_{p \in P} \delta(p, a)$$

We may write $\delta'(p, a)$ **instead** of $\delta'(\{p\}, a)$ when needed.

2. String extension : $\delta' \Rightarrow \delta'^*$ (same as $\delta \Rightarrow \hat{\delta}$)

$$\delta'^* : 2^Q \times \Sigma^* \rightarrow 2^Q.$$

$$\delta'^*(P, \varepsilon) =_B P \quad \text{basis}$$

$$\delta'^*(P, wa) =_R \delta'(\delta'^*(P, w), a) \quad \text{recursion}$$

$$P \subseteq Q (\text{or } P \in 2^Q), w \in \Sigma^*, a \in \Sigma, wa \in \Sigma^+.$$

We may **write** δ and δ^* instead of δ' and δ'^* since $\delta \subseteq \delta'$.

2.3.4 The Language of an NFA

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA. Then the **language defined**

by the NFA N , denoted as $L(N)$ is,

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

$$L(N) \subseteq \Sigma^* \text{ or } L(N) \in 2^{\Sigma^*}.$$

2.3.5 Equivalence of DFA and NFA

Example 2.10(Fig. 2.12)

Subset construction(NFA \Rightarrow DFA) $Q_D = 2^{Q_N}$.

Given an NFA $N = (Q_N, \Sigma, \delta_N, q_{0N}, F_N)$,

construct DFA $D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D) \exists. L(D) = L(N)$.

1. $Q_D = \{P \mid P \subseteq Q_N\} = 2^{Q_N}$.

3. $\delta_D(P, a) = \{q \in Q_N \mid p \in P \subseteq Q_N, q \in \delta_N(p, a)\} \subseteq Q_N$ or $\in Q_D$.

or $= \cup_{p \in P} \delta_N(p, a)$

or $= \delta_N'(P, a)$

set extension of δ_N .

$\delta_D(P, a) = \delta_N'(P, a)$ or $\delta_D = \delta_N'$ for short,

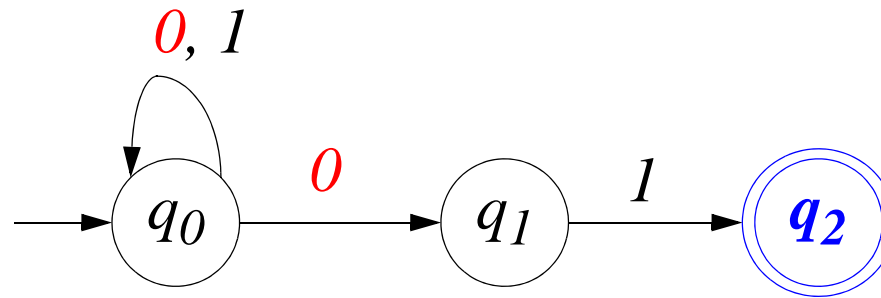
where $\delta_D: Q_D \times \Sigma \rightarrow Q_D$, $\delta_N': 2^{Q_N} \times \Sigma \rightarrow 2^{Q_N}$, and $Q_D = 2^{Q_N}$.

4. $q_{0D} = \{q_{0N}\}$.

5. $F_D = \{F \in Q_D \mid F \cap F_N \neq \emptyset\}$

Example 2.10 NFA of Fig. 2.9(p. 56)

postfix **01**.



Transition table for NFA

	0	1	
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$	$\delta(q_0, 0) = \{q_0, q_1\}$: NFA
q_1	\emptyset	$\{q_2\}$	
* q_2	\emptyset	\emptyset	

Transition table for DFA

$2^3 = 8$ subset-states (See Fig. 2.12 in p. 61)

	0	1		0	1
	\emptyset	\emptyset		$\{q_0, q_1\}$	$\{q_0, q_2\}$
→ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$	*	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	\emptyset	$\{q_2\}$	*	$\{q_1, q_2\}$	$\{q_2\}$
* $\{q_2\}$	\emptyset	\emptyset	*	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$

Starting from $\{q_0\}$

See Fig. 2.14 in p. 63.

	0	1		0	1
\rightarrow	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$	\rightarrow	$[\epsilon]$ $[0]$ $[\epsilon]$
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$		$[0]$ $[0]$ $[01]$
$*$	$\{q_0, \mathbf{q_2}\}$	$\{q_0, q_1\}$	$\{q_0\}$	$*$	$\mathbf{[01]}$ $[0]$ $[\epsilon]$

Only 3 subset-states are reachable from the initial state $\{q_0\}$.

Renaming of states

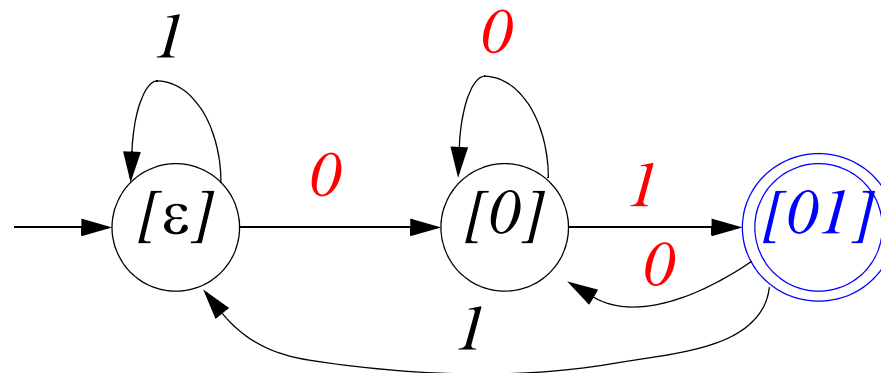


Fig. 2.14 DFA from Fig. 2.9 accepts strings that end in **01**.(postfix **01**)

Thm. 2.11 $L(D) = L(N)$ in the above subset construction

Proof $\delta_D^*(q_{0D}, w) = \delta_N'^*(\{q_{0N}\}, w)$ for all $w \in \Sigma^*$.

$$\delta_D^*: Q_D \times \Sigma^* \rightarrow Q_D (= 2^{Q_N} \times \Sigma^* \rightarrow 2^{Q_N}).$$

$\delta_N'^*: 2^{Q_N} \times \Sigma^* \rightarrow 2^{Q_N}$. We may use q_{0N} instead of $\{q_{0N}\}$.

Induction on $|w|$.

basis: Let $|w| = 0$, $\delta_D^*(q_{0D}, \varepsilon) = \delta_D^*(\{q_{0N}\}, \varepsilon) = \{q_{0N}\} = \delta_N'^*(\{q_{0N}\}, \varepsilon)$.

induction: Let $w = xa$ where $a \in \Sigma$, $x \in \Sigma^*$, and $w \in \Sigma^+$. ($|w| = |x| + 1$)

$$\begin{aligned} \delta_D^*(q_{0D}, xa) &= \delta_D(\delta_D^*(\{q_{0N}\}, x), a) && \text{by definition of } \delta_D^* \\ &= \delta_N'(\delta_D^*(\{q_{0N}\}, x), a) && \text{by subset construction} \\ &= \delta_N'(\delta_N'^*(\{q_{0N}\}, x), a) && \text{by induction hypothesis} \\ &= \delta_N'^*(\{q_{0N}\}, xa) && \text{by definition of } \delta_N'^*. \end{aligned}$$

$\delta_D^*(q_{0D}, x) \in F_D$, if and only if, $\delta_N'^*(q_0, x) \cap F_N \neq \emptyset$. $\therefore L(D) = L(N)$.

Thm. 2.12 A language L is defined by some DFA, if and only if L is defined by some NFA. ($NFA \Leftrightarrow DFA$)

Proof: (If) subset construction ($NFA \Rightarrow DFA$) **Thm. 2.11**

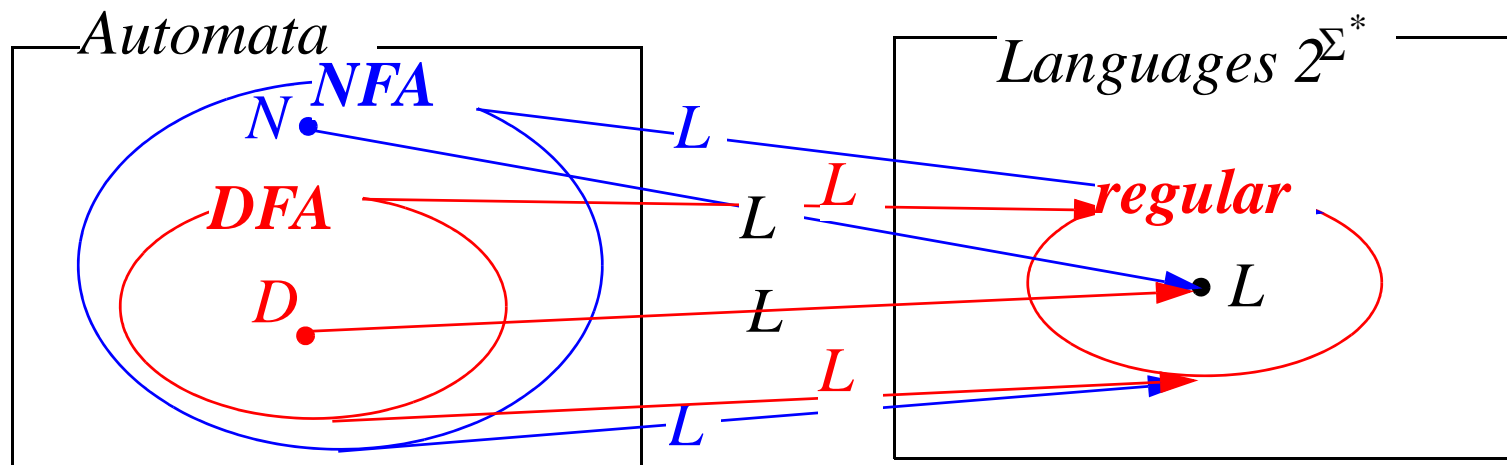
(Only if) NFA can easily simulate DFA ($DFA \Rightarrow NFA$).

Let a DFA $D = (Q, \Sigma, \delta_D, q_0, F)$. We define an NFA $(Q, \Sigma, \delta_N, q_0, F)$

$$\delta_N(q, a) = \{p \mid \delta_D(q, a) = p\}.$$

Proof for $\forall w \in \Sigma^*$, if and only if, $\hat{\delta}_D(q, w) = p$, then $\hat{\delta}_N(q, w) = \{p\}$.

$\therefore L$ is regular, iff $L = L(N)$ for some NFA N .



2.3.6 A Bad Case for the Subset Construction

See Ex. 2.13 (p64-65)

$$(0+1)^* 1(0+1)^{n-1}.$$

The state of DFA

summarizes the information concerning **past inputs** that is needed to **determine** the **behavior** of the automata on **subsequent inputs**.

Aho and Ullman in the 2nd Edition.

$$\begin{array}{ll}
 [(0+1)^*] \xrightarrow{1} [(0+1)^*1] \xrightarrow{0} [(0+1)^*10] \xrightarrow{0} [(0+1)^*100] & \Rightarrow^* [(0+1)^*10^{n-2}0] \\
 & \Rightarrow^* [(0+1)^*10^{n-2}1] \\
 & \rightarrow^1 [(0+1)^*101] \quad \dots \\
 & \dots \\
 \rightarrow^1 [(0+1)^*11] \xrightarrow{0} [(0+1)^*110] & \dots \\
 & \Rightarrow^* [(0+1)^*11^{n-2}0] \\
 \rightarrow^1 [(0+1)^*111] & \Rightarrow^* [(0+1)^*11^{n-2}1]
 \end{array}$$

2.4 An Application: Text Search Read text

2.5 Finite Automata with Epsilon-Transitions

$A = (Q, \Sigma, \delta, q_0, F)$ is a fa with epsilon transition (ϵ -NFA), if

$Q, \Sigma, q_0,$ and F are same as NFA, but

3. δ is a transition function,

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

If $q \in Q,$ and $\{p_1, p_2, \dots, p_n\} \subseteq Q,$

then we write $\delta(q, \epsilon) = \{p_1, p_2, \dots, p_n\},$ meaning that when current state is in q and **regardless** of input symbol the state may be changed to **any** one of $p_1, p_2, \dots, p_n,$ and next input symbol is **not changed**.

Ex. 2.16 in Fig. 2.18 for an ϵ -NFA accepting decimal numbers in p. 73.

Ex. 2.17 in Fig. 2.19 for an ϵ -NFA

accepting keywords “web” and “ebay” in p. 73.

2.5.3 Epsilon-closure

basis: $q \in \varepsilon^*(q)$ (same as $ECLOSE(q)$ in the text)

recursion: If $p \in \varepsilon^*(q)$, and $r \in \delta(p, \varepsilon)$, then $r \in \varepsilon^*(q)$.

Example 2.19

Why $\varepsilon^*(q)$ instead of $ECLOSE(q)$

Let $\varepsilon = \{(p, q) \mid q \in \delta(p, \varepsilon)\}$. Then $\varepsilon \subseteq Q \times Q$ (a binary relation on Q)

and $ECLOSE(q) = \varepsilon^*(q)$ reflexive-transitive closure of ε .

2.5.4 The Extended Transition and Language for ε -NFA's

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

ε -NFA: $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$.

Let's *divide* δ of ε -NFA into δ^1 and $\delta^0 (= \varepsilon)$.

$\delta^1: Q \times \Sigma \rightarrow 2^Q$.

$\delta^0: Q \times \{\varepsilon\} \rightarrow 2^Q$.

$$\delta = \delta^1 \cup \delta^0.$$

δ^1 : is *same* as δ in NFA, we use δ *instead* of δ^1 , since $\delta^1 = \delta$.

$\delta^0 \leftrightarrow_f \varepsilon$ what is the **bijection** f ? We use ε *instead* of δ^0 .

$\varepsilon = \{(p, q) \in \delta / q \in \delta(p, \varepsilon)\}$ binary **relation** on Q .

$\varepsilon^* = \bigcup_{i \in \mathbb{N}_0} \varepsilon^i \subseteq Q \times Q$ binary **relation** on Q .

$\varepsilon, \varepsilon^*: Q \rightarrow 2^Q$. **function** from Q to 2^Q .

Let's **extend** the domain of $\delta(=\delta^1)$, ε , and ε^* from **state** to **set of states**.

Let $P \subseteq Q$, we define

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

$$\delta'(P, a) = \{q \in Q \mid p \in P, \delta(p, a) = q\}$$

$$\text{or} \quad = \cup_{p \in P} \delta(p, a)$$

We may write $\delta'(p, a)$ **instead** of $\delta'(\{p\}, a)$ when needed.

We may write $\delta(P, a)$ **instead** of $\delta'(P, a)$ when needed.

$$\varepsilon': 2^Q \rightarrow 2^Q.$$

$$\varepsilon'(P) = \{q \in Q \mid p \in P, \varepsilon(p) = q\}$$

$$\text{or} \quad = \cup_{p \in P} \varepsilon(p)$$

$$\varepsilon'^*: 2^Q \rightarrow 2^Q.$$

$$\varepsilon'^*(P) = \{q \in Q \mid p \in P, \varepsilon^*(p) = q\} = \cup_{p \in P} \varepsilon^*(p)$$

We write $\varepsilon(P)$ **instead** of $\varepsilon'(P)$ and $\varepsilon^*(P)$ **instead** of $\varepsilon'^*(P)$.

Let's **extend** the 2nd domain of δ from **symbol** to **strings**.

$$\hat{\delta}: 2^Q \times \Sigma^* \rightarrow 2^Q.$$

$$\hat{\delta}(P, \varepsilon) =_B \varepsilon^*(P) \quad \text{basis}$$

$$\hat{\delta}(P, wa) =_R \varepsilon^*(\delta(\hat{\delta}(P, w), a)) \quad \text{recursion}$$

$$P \subseteq Q (\text{or } P \in 2^Q), w \in \Sigma^*, a \in \Sigma, wa \in \Sigma^+.$$

$$\hat{\delta}(q_0, a_1 a_2 \dots a_n) =_R \varepsilon^*(\delta(\hat{\delta}(q_0, a_1 a_2 \dots a_{n-1}), a_n)) \quad \text{1st recursion}$$

$$=_R \varepsilon^*(\delta(\varepsilon^*(\delta(\hat{\delta}(q_0, a_1 a_2 \dots a_{n-2}), a_{n-1})), a_n)) \quad \text{2nd recursion}$$

$$=_R \dots$$

$$=_R \varepsilon^*(\delta(\varepsilon^*(\delta(\dots \varepsilon^*(\delta(\varepsilon^*(\delta(\hat{\delta}(q_0, \varepsilon), a_1)), a_2)), \dots)), a_{n-1})), a_n)) \quad \text{n-th}$$

$$=_B \varepsilon^*(\delta(\varepsilon^*(\delta(\dots \varepsilon^*(\delta(\varepsilon^*(\delta(\varepsilon^*(q_0), a_1)), a_2)), \dots)), a_{n-1})), a_n)) \quad \text{basis}$$

$$\delta^n = \varepsilon^{* \circ} \delta^{\circ} \varepsilon^{* \circ} \dots \circ \delta^{\circ} \varepsilon^{* \circ} = \varepsilon^{* \circ} (\delta^{\circ} \varepsilon^{* \circ})^n = \varepsilon^{* \circ} (\delta^1)^n. \quad \text{where } \delta^1 = \delta^{\circ} \varepsilon^{* \circ}.$$

We may write δ^* instead of $\hat{\delta}$, since $\hat{\delta} = \cup_{i \in N_0} \delta^i = \delta^*$.

2.5.5 Eliminating ε -Transitions

Given an ε -NFA $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$, construct the equivalent DFA $D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$ such that $L(D) = L(N)$.

1. $Q_D = \{P \mid P \subseteq Q_E\} = 2^{Q_E}$.
3. $q_{0D} = \varepsilon^*({q_0})$
4. $\delta_D(P, a) = \varepsilon^*(\delta_E(P, a))$ set extension of δ_E and ε -closure ($\delta^1 = \delta \circ \varepsilon^*$)
 where $P \subseteq Q_E$ ($P \in Q_D$), $a \in \Sigma$, and
 $\delta_D(P, a) \subseteq Q_E$ (or $\delta_D(P, a) \in Q_D$)
5. $F_D = \{F \in Q_D \mid F \cap F_E \neq \emptyset\}$

Theorem 2.22 A language L is accepted by some ε -NFA if and only if L is accepted by some DFA.

Proof: (If) ε -NFA can easily simulate DFA (DFA \Rightarrow ε -NFA).

(Only if) Eliminating ε -transitions (ε -NFA \Rightarrow DFA)

Let $E = (Q_E, \Sigma, \delta_E, q_{0E}, F_E)$ be a ε -NFA and

DFA $D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$ is the one in 2.5.5.

We show $\delta_E^*(q_{0E}, w) = \delta_D^*(q_{0D}, w)$ by induction on $|w|$.

basis: $\delta_E^*(q_{0E}, \varepsilon) = \varepsilon^*(q_{0E})$ by basis definition of δ_E^* .
 $= q_{0D} =$ by **construction** of δ_D .
 $= \delta_D^*(q_{0D}, \varepsilon)$ by basis definition of δ_D^* .

Induction: Let $w = xa \in \Sigma^+$, $a \in \Sigma$, and $x \in \Sigma^*$.

$\delta_E^*(q_0, xa) = \delta_E(\varepsilon^*(\delta_E^*(q_0, x), a))$ by recursive definition of δ_E^* .
 $= \delta_E(\varepsilon^*(\delta_D^*(q_{0D}, x), a))$ by **induction hypothesis**.
 $= \delta_D(\delta_D^*(q_{0D}, x), a)$ by **construction** of δ_D .
 $= \delta_D^*(q_{0D}, xa)$ by recursive definition of δ_D^* .

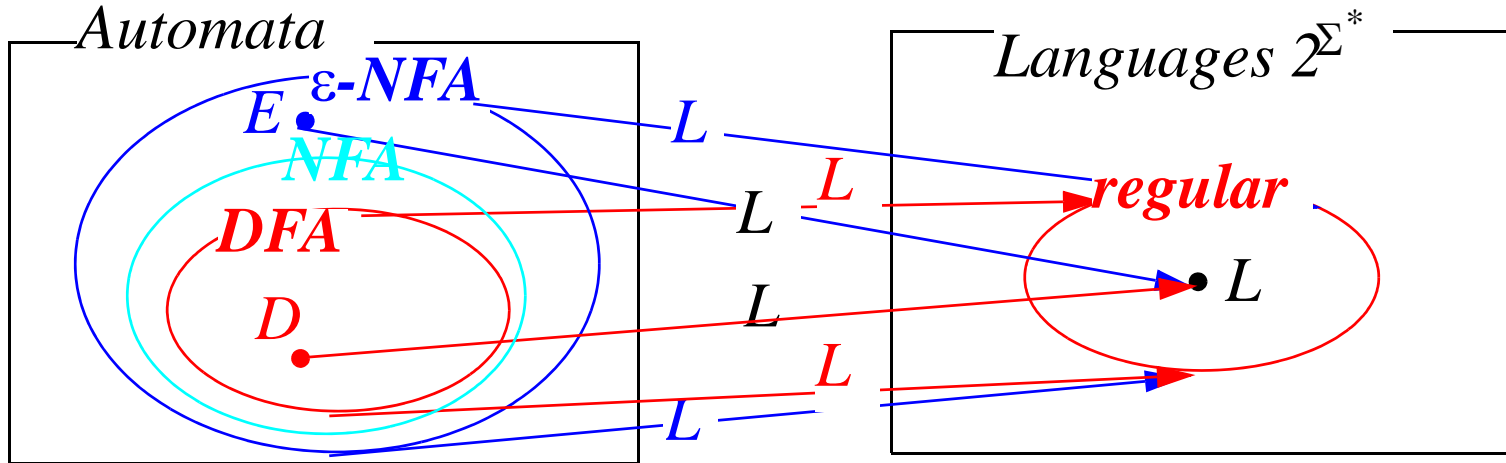
ε -NFA Transition Table for Fig. 2.18 in p. 73.

	δ^I	+, -	.	d		ε	ε^*
\rightarrow	q_0	$\{q_1\}$	\emptyset	\emptyset	q_0	$\{q_1\}$	$\{q_0, q_1\}$
	q_1	\emptyset	$\{q_2\}$	$\{q_1, q_4\}$	q_1	\emptyset	$\{q_1\}$
	q_2	\emptyset	\emptyset	$\{q_3\}$	q_2	\emptyset	$\{q_2\}$
*	q_3	\emptyset	\emptyset	$\{q_3\}$	q_3	$\{q_5\}$	$\{q_3, q_5\}$
	q_4	\emptyset	$\{q_3\}$	\emptyset	q_4	\emptyset	$\{q_4\}$
	q_5	\emptyset	\emptyset	\emptyset	q_5	\emptyset	$\{q_5\}$

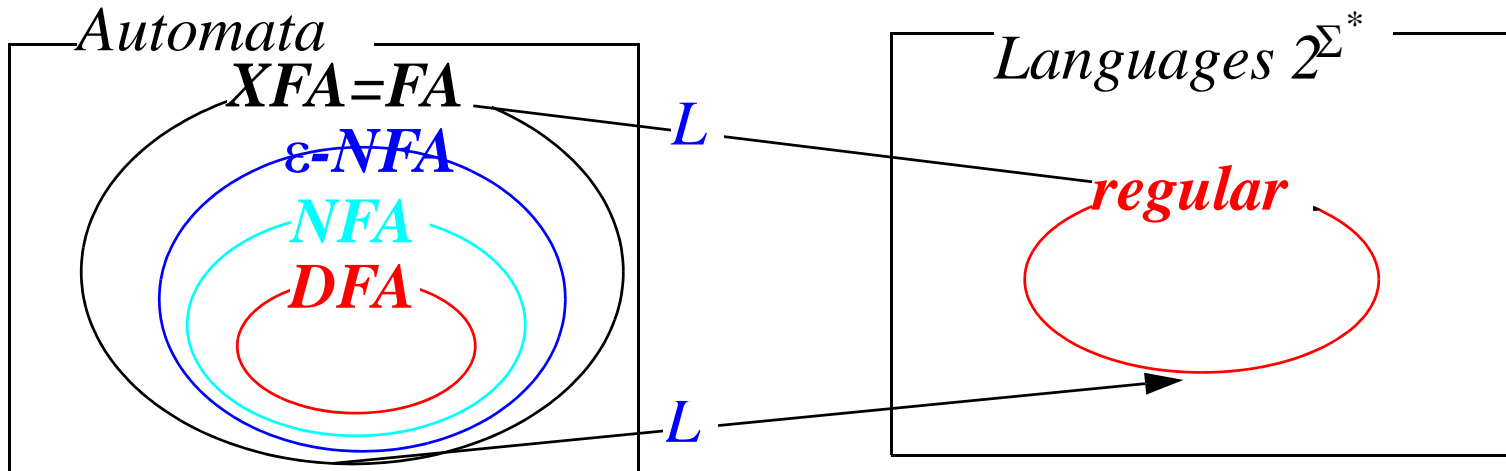
DFA Transition Table for Fig. 2.22 in p. 79.

	$\varepsilon^* \circ \delta^I \circ \varepsilon^*$	+, -	.	d
\rightarrow	$\{q_0, q_1\}$	$\{q_1\}$	$\{q_2\}$	$\{q_1, q_4\}$
	$\{q_1\}$	\emptyset	$\{q_2\}$	$\{q_1, q_4\}$
	$\{q_2\}$	\emptyset	\emptyset	$\{q_3, q_5\}$
	$\{q_1, q_4\}$	\emptyset	$\{q_2, q_3, q_5\}$	$\{q_1, q_4\}$
*	$\{q_3, q_5\}$	\emptyset	\emptyset	$\{q_3, q_5\}$
*	$\{q_2, q_3, q_5\}$	\emptyset	\emptyset	$\{q_3, q_5\}$

ϵ -NFA = NFA = DFA



XFA's (FA's) = ϵ -NFA's = NFA's = DFA's = Class of regular languages



2.A *Extended Finite Automata(XFA)*

$X = (Q, \Sigma, q_0, \delta, F)$ is an *extended finite state automata(XFA)*, if

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

Given an XFA $X = (Q_X, \Sigma, q_0, \delta_X, F_X)$, **construct**

an **equivalent ε -NFA** $E = (Q_E, \Sigma, q_0, \delta_E, F_E)$ such that $L(X) = L(E)$.

Construction algorithm

$Q_E := Q_X; F_E := F_X; \delta_E := \emptyset;$

for $\forall (\delta_X(q, x) = P) \in \delta_X$ where $x = a_1 a_2 \dots a_n$ **do**

if $0 \leq n \leq 1 \rightarrow \delta_E(q, x) := P$ where $x \in \Sigma \cup \{\varepsilon\}$

$| n \geq 2 \rightarrow \delta_E(q, a_1) = \delta_E(q, a_1) \cup \{p_1\};$

for $2 \leq \forall i \leq n-1$ **do** $\delta_E(p_{i-1}, a_i) := \{p_i\}$ **od**; $\delta_E(p_{n-1}, a_n) = P;$

$Q_E := Q_E \cup \{p_1, p_2, \dots, p_{n-1}\}$ where $Q_E \cap \{p_1, p_2, \dots, p_{n-1}\} \neq \emptyset$

fi

od

The following statements are **equivalent**,

1. A language L is **regular**.

2. $L = L(D)$ for some DFA D where $\delta : Q \times \Sigma \rightarrow Q$.

3. $L = L(P)$ for some DFA P where $\delta : Q \times \Sigma \rightarrow Q \cup \{\emptyset\}$ with partial function δ .

4. $L = L(N)$ for some NFA N where $\delta : Q \times \Sigma \rightarrow 2^Q$.

5. $L = L(E)$ for some ε -NFA E where $\delta : Q \times (\{\varepsilon\} \cup \Sigma) \rightarrow 2^Q$.

6. $L = L(X)$ for some XFA X where $\delta : Q \times \Sigma^* \rightarrow 2^Q$.

The following two statements are **equivalent**,

1. A language L is **regular**.

2. $L = L(A)$ for some FA A where $\delta : Q \times \Sigma^* \rightarrow 2^Q$.

Consider a language $L \subseteq \Sigma^*$ and a **binary relation** R_L on Σ^* where

$$\forall x, y \in \Sigma^*: x R_L y, \text{ if } \forall z \in \Sigma^*: xz, yz \in L \text{ or } xz, yz \notin L.$$

Then R_L is a **quivalent**(proof?) relation on Σ^* .

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Then we define **accessible** input strings from the initial stat, q_0 , to a state $q \in Q$, as $L(q)$,

$$L(q) = \{w \in \Sigma^* \mid \delta^*(q_0, w) = q\} \quad (\because L(M) = \cup_{q \in F} L(q))$$

Consider a DFA $M = (Q, \Sigma, \delta, q_0, F)$ and a **binary relation** R_M on Σ^*

$$\forall x, y \in \Sigma^*: x R_M y, \text{ if } \delta^*(q_0, x) = \delta^*(q_0, y). \text{ Then } Par_{R_M}(\Sigma^*) \cong_f Q.$$

Compare two **quivalent** relation on Σ^* , R_L and R_M . \exists . $L = L(M)$.

$$|Par_{R_L}(\Sigma^*)| \leq |Par_{R_M}(\Sigma^*)| \text{ in general.}$$

If $|Par_{R_L}(\Sigma^*)| = |Par_{R_{M'}}(\Sigma^*)|$ for some $M' = (Q', \Sigma, \delta', q_0', F')$.

M' is a **minimal state DFA** that accepting L .