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-∅, WC-mG, …, ∅-mWGC

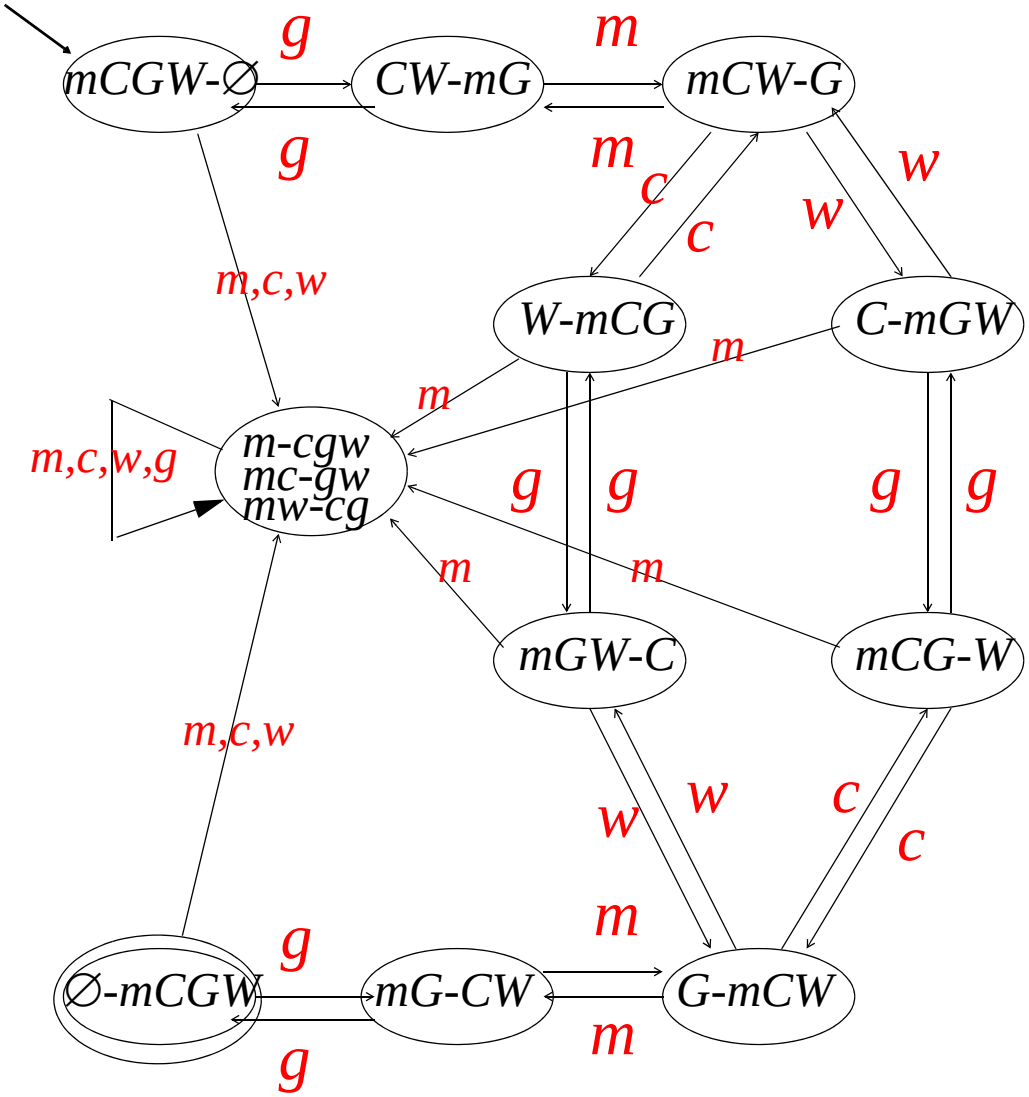
***input symbol**: four ways of crossing the river*

A man crosses alone(m) with the wolf(w), goat(g), or cabbage(c).

***input string**: a **sequence** of crossings (input symbols = {m, w, g, c})*

state transition**: From a **state** to another **state

*by an **input symbol** (crossing)*


mCGW-∅
CW-mG
mCW-G
W-mCG
C-mGW
m-cgw
mc-gw
mw-cg
mGW-C
mCG-W
∅-mCGW
mG-CW
G-mCW
g
m
c
w
m,c,w
m,c,w,g

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 $a_1a_2 \ldots a_n$ 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 $\delta(q_1, a_2) = q_2$, $\delta(q_2, a_3) = q_3$, …, $\delta(q_{n-1}, a_n) = q_n$.

*4. **Finally**, check if $\delta(q_{n-1}, a_n) = q_n \in F$ or not.*

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

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

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

function *DFA*(

$\underline{D} = (Q$:**set** *of states*, Σ:**set** *of symbols*, δ: $Q \times \Sigma \to Q$, $q_0 \in Q$, $F \subseteq Q)$;

$\underline{x} = a_1 a_2 \ldots a_n \in \Sigma^*$) /* $\underline{D}$ *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 $i \leq n \to$ **if** $^{\exists}\delta(q, a_i) \to q, i := \delta(q, a_i), i+1$

$\mid \neg^{\exists}\delta(q, a_i) \to$ **return** $\underline{D}$ $\neg$**accept** $\underline{x}$

fi

od

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

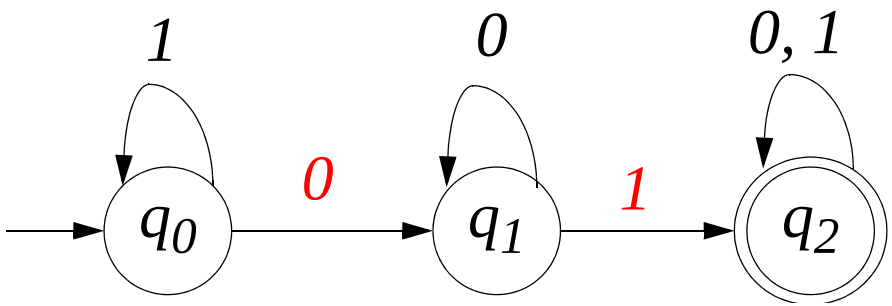
end function *DFA*.

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

$\delta: Q \times \Sigma \to Q$

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

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

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

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

$\hat{\delta}(q_0, a_1a_2 \ldots a_n) =_R \delta(\hat{\delta}(q_0, a_1a_2 \ldots a_{n-1}), a_n)$ 1st **recursion**

$=_R \delta(\delta(\hat{\delta}(q_0, a_1a_2 \ldots a_{n-2}), a_{n-1}), a_n)$ 2nd **recursion**

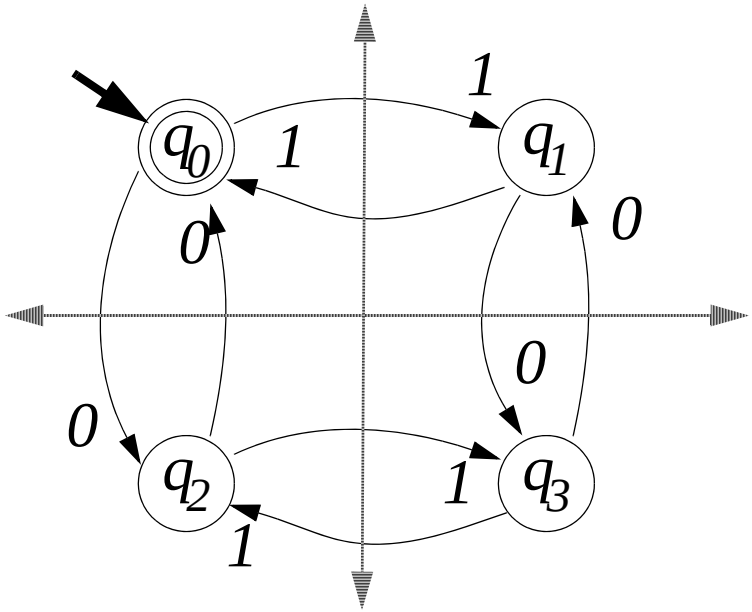
$=_R \ldots$

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

$=_B \delta(\delta(\ldots\delta(\delta(q_0, a_1), a_2), \ldots), a_{n-1}), a_n)$ **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* $D = (Q, \Sigma, \delta, q_0, F)$, $L(D)$,

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

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

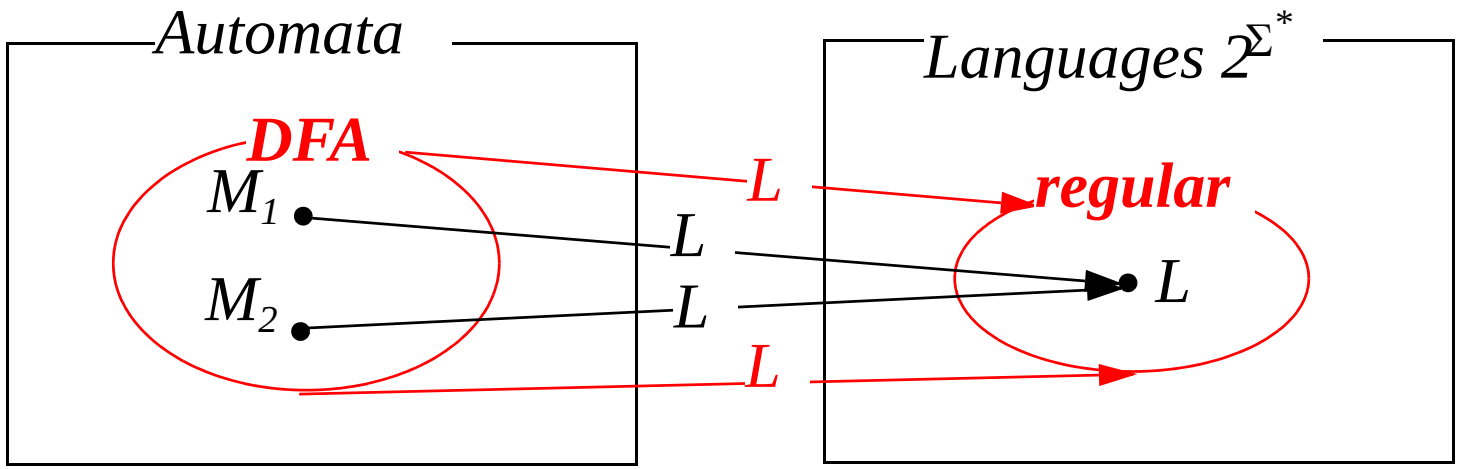
Def. *A* ***Class*** *of* ***Regular languages***

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***

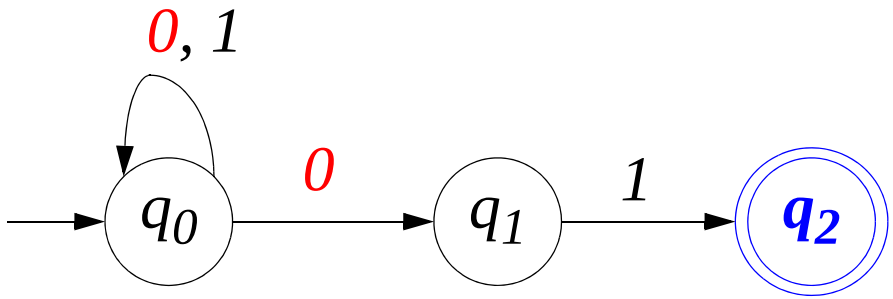
Let M_1 and M_2 be two automata. We say two automata M_1 and M_2 are ***equivalent****, if $L(M_1) = L(M_2) = L$.*

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(**postfix** 01).*

Fig. 2.9(p. 56)



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

$q_0 \rightarrow^0 q_0 \rightarrow^0 q_0 \rightarrow^1 q_0 \rightarrow^0 q_0 \rightarrow^1 q_0 \notin F$(*fail*)

$q_0 \rightarrow^0 q_1 \rightarrow^0 X$(*stuck*)

$q_0 \rightarrow^0 q_0 \rightarrow^0 q_1 \rightarrow^1 q_2 \rightarrow^0 X$(*stuck*)

$q_0 \rightarrow^0 q_0 \rightarrow^0 q_0 \rightarrow^1 q_0 \rightarrow^0 q_1 \rightarrow^1 q_2 \in F$(***success***)

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

$^{\forall}q \in Q, {}^{\forall}a \in \Sigma, {}^{\exists}\delta(q, a) \in Q.$ ***(total)*** *function*

$^{\exists}q \in Q, {}^{\exists}a \in \Sigma$.ə. $^{\nexists}\delta(q, a).$ ***partial*** *function*

If $A = (Q, \Sigma, \delta, q_0, F)$ *be a FA with* $\delta: Q \times \Sigma \not\to Q$ ***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 $\to$ ***partial*** *function.*

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

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

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

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* $\{p_1, p_2, \ldots, p_n\} \subseteq Q$ $(1 \leq {}^{\forall}i \leq n, p_i \in Q)$

   then we write $\delta(q, a) = \{p_1, p_2, \ldots, p_n\}$, *meaning that*

   when current state is in q *and input symbol is* a,

   the state may be changed to ***any*** *one of* $p_1, p_2, \ldots, p_n$,

   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: $\delta: Q \times \Sigma \to Q$ *Definition*

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

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

$\delta': 2^Q \times \Sigma \to 2^Q$ *1st set extension in this lecture*

$\delta'^*: 2^Q \times \Sigma^* \to 2^Q$ *2nd string extension in this lecture*

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

Two step extensions

1. Set extension: $\delta \Rightarrow \delta'$: *We extend the 1st domain of* δ' *to* ***set of states.***

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

Let $P \subseteq Q$. *Then we define*

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

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

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

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

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

$\delta'^{*}(P, \varepsilon) =_{B} P$ ***basis***

$\delta'^{*}(P, wa) =_{R} \delta'(\delta'^{*}(P, w), a)$ ***recursion***

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

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,*

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

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

2.3.5 Equivalence of DFA and NFA

Example 2.10(Fig. 2.12)

Thm 2.11 Subset Construction(NFA $\Rightarrow$ DFA) $Q_D = 2^{Q_N}$.

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

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

1. $Q_D = \{P | P \subseteq Q_N\} = 2^{Q_N}$.
3. $\delta_D(P, a) = \{q \in Q_N | p \in P \subseteq Q_N, q \in \delta_N(p, a)\} \subseteq Q_N$ or $\in Q_D$.

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

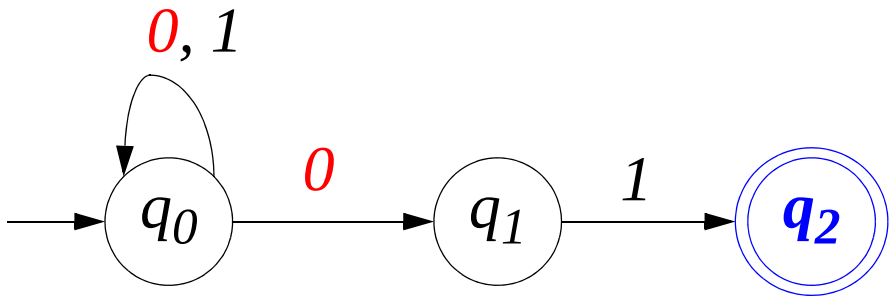
   or $= \delta_N'(P, a)$ **set extension** of δ_N.

   $\delta_D(P, a) = \delta_N'(P, a)$ 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. $F_D = \{F \in Q_D | F \cap F_N \neq \varnothing\}$

Example 2.10 *NFA of Fig. 2.9(p. 56)* *postfix 01.*



Transition table *for NFA*

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

$\delta(q_0, 0) = \{q_0, q_1\}$: *NFA*

Transition table *for DFA* $2^3 = 8$ *subset-states(See Fig. 2.12 in p. 61)*

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

		0	1
	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, \boldsymbol{q_2}\}$
*	$\{q_0, \boldsymbol{q_2}\}$	$\{q_0, q_1\}$	$\{q_0\}$
*	$\{q_1, \boldsymbol{q_2}\}$	$\varnothing$	$\{q_2\}$
*	$\{q_0, q_1, \boldsymbol{q_2}\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

Starting from $\{q_0\}$ *See Fig. 2.14 in p. 63.*

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

		0	1
→	[ε]	[0]	[ε]
	[0]	[0]	[01]
*	[01]	[0]	[ε]

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

Renaming of states

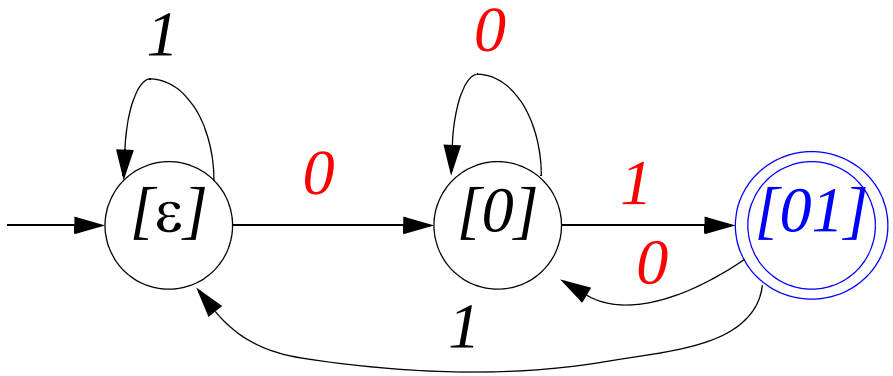


*Fig. 2.14 DFA from Fig. 2.9 accepting strings that end in 01.(**postfix** 01)*

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* $w \in \Sigma^*$.

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

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

Induction *on* $|w|$.

basis*: Let* $|w| = 0$, $\delta^*_D(q_{0D}, \varepsilon) = \delta^*_D(\{q_{0N}\}, \varepsilon) = \{q_{0N}\} = \hat{\delta'}_N(\{q_{0N}\}, \varepsilon)$.

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

$\delta_D^*(q_{0D}, xa) = \delta_D(\delta^*_D(\{q_{0N}\}, x), a)$ *by definition of* δ^*_D.

$= \delta_N'(\delta_D^*(\{q_{0N}\}, x), a)$ *by* ***subset construction***

$= \delta_N'(\delta_N'^*(\{q_{0N}\}, x), a)$ *by induction hypothesis*

$= \delta_N'^*(\{q_{0N}\}, xa)$ *by definition of* $\delta_N'^*$.

$\delta_D^*(q_{0D}, x) \in F_D$, *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 ⇔ DFA)*

Proof*: (If)* ***subset construction****(NFA ⇒ DFA)* ***Thm. 2.11***

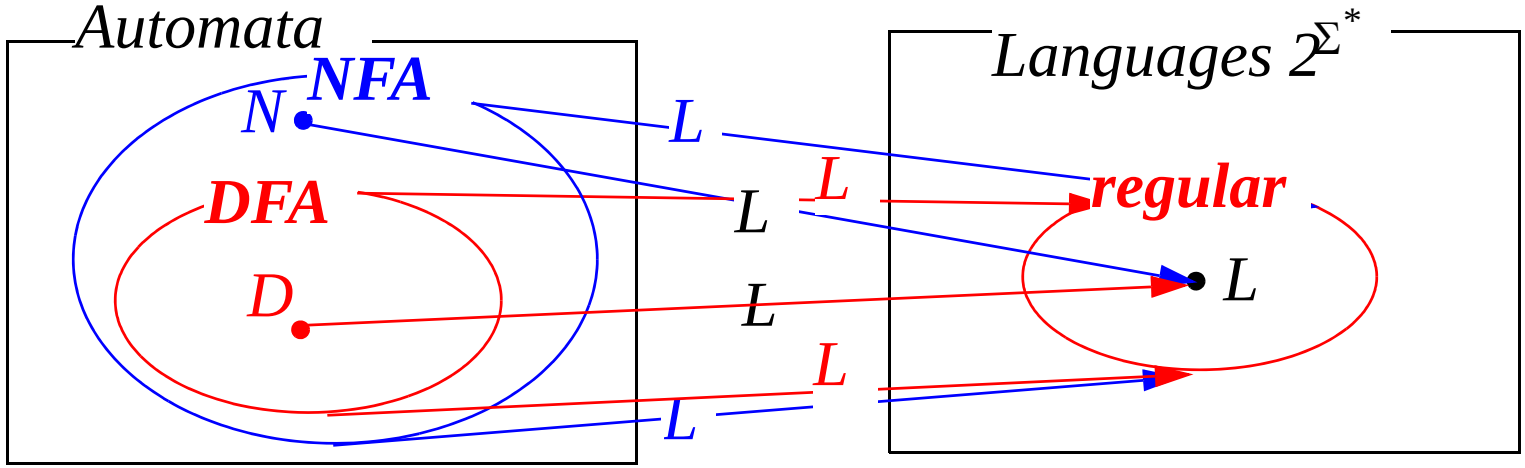
(Only if) NFA can ***easily simulate*** *DFA(DFA ⇒ NFA).*

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* $\hat{\delta}_D(q, w) = \{p\}$.

$\therefore$ *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 **behaviour** of the automata*
*on **subsequent inputs**.*

Aho and Ullman in the 2nd Edition.

$[(0+1)^*] \rightarrow^1 [(0+1)^*1] \rightarrow^0 [(0+1)^*10] \rightarrow^0 [(0+1)^*100]$ $\Rightarrow^* [(0+1)^*10^{n-2}0]$

$\Rightarrow^* [(0+1)^*10^{n-2}1]$

$\rightarrow^1 [(0+1)^*101]$ ...

...

$\rightarrow^1 [(0+1)^*11] \rightarrow^0 [(0+1)^*110]$ ...

$\Rightarrow^* [(0+1)^*11^{n-2}0]$

$\rightarrow^1 [(0+1)^*111]$ $\Rightarrow^* [(0+1)^*11^{n-2}1]$

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, Σ, q_0, and F are same as NFA, but

3. δ is a **transition function**,

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

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

then we write $\delta(q, \varepsilon) = \{p_1, p_2, \ldots, p_n\}$, meaning that

when current state is in q and **regardless** of input symbol

the state may be changed to **any** one of $p_1, p_2, \ldots, p_n$,

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 in p. 74.

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

Let $\varepsilon = \{(p, q) \in Q \times Q | \, q \in \delta(p, \varepsilon)\}$.

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

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

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

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

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

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

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.

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

$\varepsilon^* = \cup_{i \in N_0} \varepsilon^i$ $\subseteq Q \times Q$ *binary **relation** on Q.*

$\varepsilon, \varepsilon^*: Q \to 2^Q$. ***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

δ': $2^Q \times \Sigma \to 2^Q$.

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

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

*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.*

ε': $2^Q \to 2^Q$.

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

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

ε'^*: $2^Q \to 2^Q$.

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

*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**.*

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

$\delta^*(P, \varepsilon) =_B \varepsilon^*(P)$ ***basis***

$\delta^*(P, wa) =_R \varepsilon^*(\delta(\delta^*(P, w), a))$ ***recursion***

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

$\delta^n(q_0, a_1a_2 \ldots a_n) =_R \varepsilon^*(\delta(\delta^{n-1}(q_0, a_1a_2 \ldots a_{n-1}), a_n))$ *1st* ***recursion***

$=_R \varepsilon^*(\delta(\varepsilon^*(\delta(\delta^{n-2}(q_0, a_1a_2 \ldots a_{n-2}), a_{n-1})), a_n))$ *2nd* ***recursion***

$=_R \ldots$

$=_R \varepsilon^*(\delta(\varepsilon^*(\delta(\ldots\varepsilon^*(\delta(\varepsilon^*(\delta(\delta^0(q_0, \varepsilon), a_1)), a_2)), \ldots)), a_{n-1})), a_n))$ ***n-th***

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

$\delta^n = \varepsilon^* \circ \delta \circ \varepsilon^* \circ \ldots \circ \delta \circ \varepsilon^* = \varepsilon^* \circ (\delta \circ \varepsilon^*)^n = \varepsilon^* \circ (\delta^1)^n$. *where* $\delta^1 = \delta \circ \varepsilon^*$.

$\therefore \delta^* = \cup_{i \in \mathbb{N}_0} \delta^i = \delta^*$.

2.5.5 Eliminating ε-Transitions

Given an ε-NFA $E = (Q_E, \Sigma, \delta_E, q_{0E}, F_E)$, *construct the*

equivalent *DFA* $D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$ *such that* $L(D) = L(N)$.

1. $Q_D = \{P | P \subseteq Q_E\} = 2^{Q_E}$.

3. $q_{0D} = \varepsilon^*(\{q_{0E}\})$

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*

   $\delta_D(P, a) \subseteq Q_E$ $(\text{or } \delta_D(P, a) \in Q_D)$

5. $F_D = \{F \in Q_D | F \cap F_E \neq \emptyset\}$

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* <u>*basis*</u> ***definition*** *of* δ_E^*.

$= q_{0D}$ *by* ***construction*** *of* δ_D.

$= \delta_D^*(q_{0D}, \varepsilon)$ *by* <u>*basis*</u> ***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* <u>*recursive*</u> ***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* <u>*recursive*</u> ***definition*** *of* δ_D^*.

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

	δ^1	+,–	.	**d**		ε	ε^*
→	q_0	$\{q_1\}$	∅	∅	$\boldsymbol{q_0}$	$\{q_1\}$	$\{\boldsymbol{q_0}, q_1\}$
	q_1	∅	$\{q_2\}$	$\{q_1, q_4\}$	q_1	∅	$\{q_1\}$
	q_2	∅	∅	$\{\boldsymbol{q_3}\}$	q_2	∅	$\{q_2\}$
*	q_3	∅	∅	$\{\boldsymbol{q_3}\}$	$\boldsymbol{q_3}$	$\{q_5\}$	$\{\boldsymbol{q_3}, q_5\}$
	q_4	∅	$\{\boldsymbol{q_3}\}$	∅	q_4	∅	$\{q_4\}$
	q_5	∅	∅	∅	q_5	∅	$\{q_5\}$

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

	$\varepsilon^*\circ\delta^1\circ\varepsilon^*$	+,–	.	**d**
→	$\{\boldsymbol{q_0}, q_1\}$	$\{q_1\}$	$\{q_2\}$	$\{q_1, q_4\}$
	$\{q_1\}$	∅	$\{q_2\}$	$\{q_1, q_4\}$
	$\{q_2\}$	∅	∅	$\{\boldsymbol{q_3}, q_5\}$
	$\{q_1, q_4\}$	∅	$\{q_2, \boldsymbol{q_3}, q_5\}$	$\{q_1, q_4\}$
*	$\{\boldsymbol{q_3}, q_5\}$	∅	∅	$\{\boldsymbol{q_3}, q_5\}$
*	$\{q_2, \boldsymbol{q_3}, q_5\}$	∅	∅	$\{\boldsymbol{q_3}, q_5\}$

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 $X = (Q_X, \Sigma, q_0, \delta_X, F)$, ***construct***

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

Construction algorithm

$Q_E := Q_X$; $\delta_E := \varnothing$;

for $^{\forall}(\delta_X(q, x) = P) \in \delta_X$ *where* $x = a_1a_2\ldots a_n$ ***do***

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

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

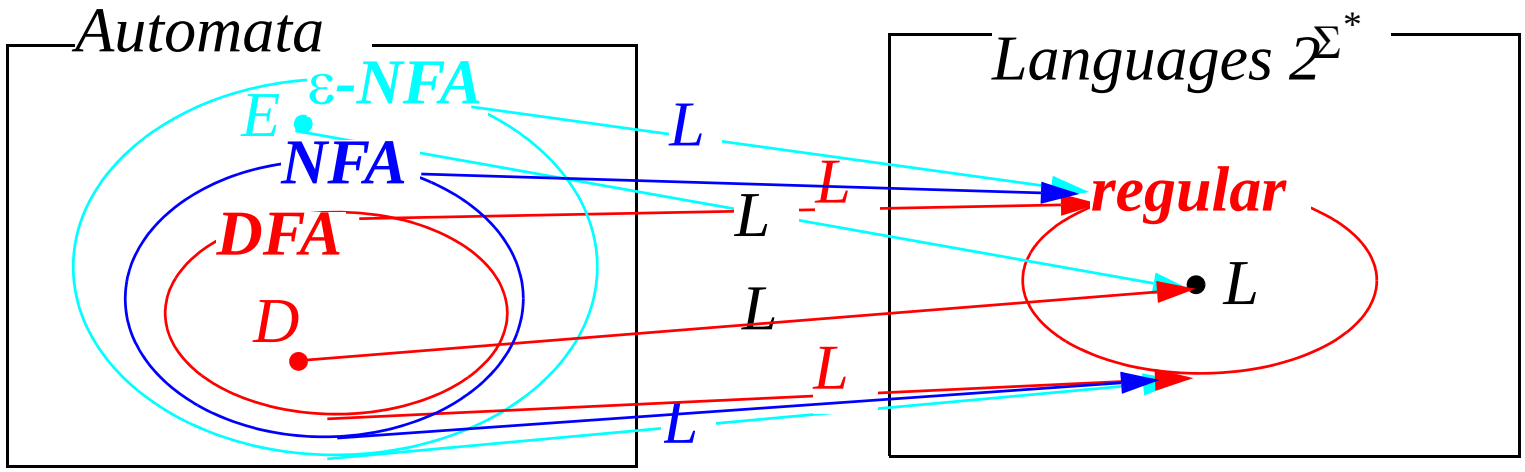
for $2 \le {}^{\forall}i < n$ ***do*** $Q_E := Q_E \cup \{p_i\}$; $\delta_E := \delta_E \cup \{\delta_E(p_{i-1}, a_i) = \{p_i\}\}$ ***od***;

$\delta_E := \delta_E \cup \{\delta_E(p_{n-1}, a_n) = P\}$ *where* $Q_X \cap \{p_1, p_2, \ldots, p_{n-1}\} \neq \varnothing$

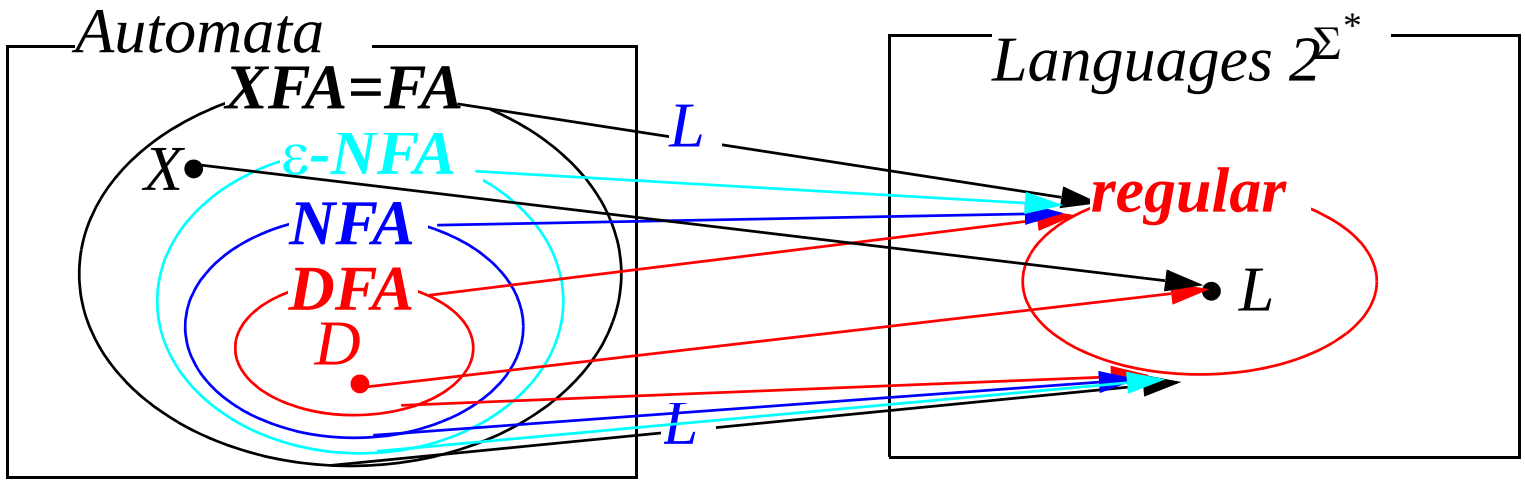
fi

od

ε-NFA's = NFA's = DFA's = Class of regular languages



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



*The following **six** statements are **equivalent**,*

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

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

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

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

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

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

*The following **two** statements are **equivalent**,*

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

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

We've changed the definition of FA from DFA to XFA.

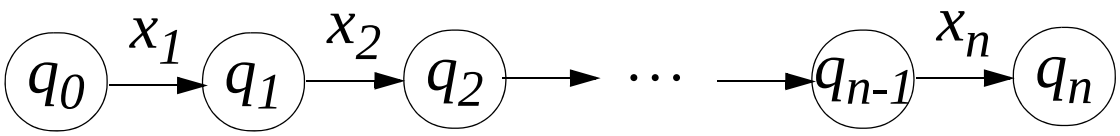
2.B Configuration rewriting system for FA

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a Finite automata where $\delta: Q \times \Sigma^* \to 2^Q$. Then

A **configuration rewritng system** $R_A = (Q \times \Sigma^*, \to)$ on $Q \times \Sigma^*$ is

$\to = \{(q, x) \to (p, \varepsilon) \mid p \in \delta(q, x)\}$. $(q, x), (p, \varepsilon) \in Q \times \Sigma^*$: **configurations**

Assume $x = x_1x_2...x_n \in \Sigma^*$ and $1 \le {}^{\forall}i \le n$: $q_i \in \delta(q_{i-1}, x_i)$. Then



$\delta(...\delta(\delta(q_0, x_1), x_2), ..., x_{n-1}), x_n) =^{q_1 \in \delta(q_0, x_1)} \delta(\delta(...\delta(q_1, x_2),...,x_{n-1}) x_n) =$

$... = \delta(q_{n-2}, x_{n-1}), x_n) =^{q_{n-1} \in \delta(q_{n-2}, x_{n-1})} \delta(q_{n-1}, x_n) =^{q_n \in \delta(q_{n-1}, x_n)} q_n.$

$(q_0, x_1x_2...x_{n-1}x_n) \Rightarrow^{(q_0, x_1) \to (q_1, \varepsilon)} (q_1, x_2...x_{n-1}x_n) \Rightarrow$

$... \Rightarrow (q_{n-2}, x_{n-1}x_n) \Rightarrow^{(q_{n-2}, x_{n-1}) \to (q_{n-1}, \varepsilon)} (q_{n-1}, x_n) \Rightarrow^{(q_{n-1}, x_n) \to (q_n, \varepsilon)} (q_n, \varepsilon).$

$L(A) = \{x \in \Sigma^* \mid (q_0, x) \Rightarrow^* (f, \varepsilon), f \in F\}.$

Read this page for the ***minimal state DFA*** *TBD in Chap. 4.*

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

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

Then R_L *is an* ***equivalent(proof*** $^?$***)*** *relation on* Σ^*.

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

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

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

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