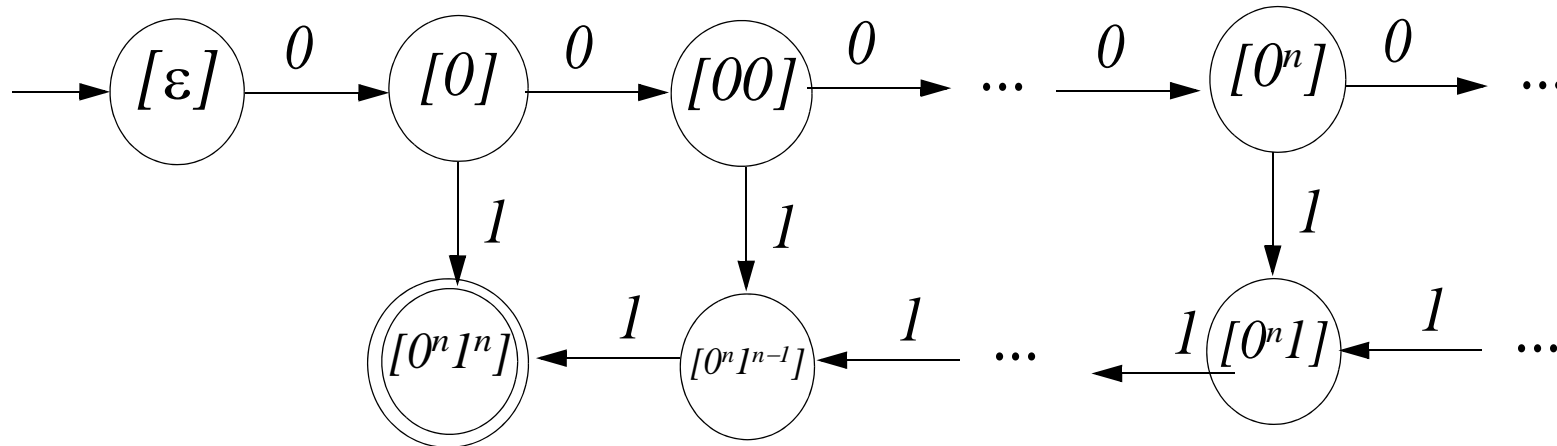


Chap. 4 Properties of Regular Languages

4.1 Proving Languages not to be Regular

Consider $L_{0^n 1^n} = \{0^n 1^n \mid n \geq 0\}$

finite automaton?



regular expression?

0^*1^* .

4.1.1 Pumping Lemma

Theorem 4.1: (The pumping lemma for regular languages)

Let L be a regular languages. Then there exists a **constant** n (which **depends on** L) such that for **every** string $w \in L$ such that $|w| \geq n$, we can break w into three substrings, $w = xyz$, such that:

- 1) $y \neq \varepsilon$, **non-empty string** y
- 2) $|xy| \leq n$, and **not so far from the beginning**
- 3) for all $k \geq 0$, the string $xy^kz \in L$. **pumping** y any number of times or deleting it

Proof

Suppose L is regular. Then $L = L(A)$ for some DFA $A = (Q, \Sigma, \delta, q_0, F)$.

Suppose A has n states. ($|Q| = n$)

Consider $\forall w = a_1a_2\dots a_m, m \geq n, 0 < \forall i \leq m: [a_i \in \Sigma, \delta^*(q_0, a_1a_2\dots a_i) = q_i]$

Consider $q_0q_1\dots q_m \in Q^*$. Since $m \geq n, \exists i, j. \exists. 0 \leq i < j \leq n$ and $q_i = q_j = q$.

Now we break $w = xyz$ as follows

1. $x = a_1a_2\dots a_i$ $\delta^i(q_0, x) = q_i = q$
2. $y = a_{i+1}a_{i+2}\dots a_j$ $\delta^{j-i}(q_i, y) = q_j = q = \delta^{|y|}(q, y)$
3. $z = a_{j+1}a_{j+2}\dots a_m$ $\delta^{m-j}(q_j, z) = \delta^{m-j}(q, z) = q_m \in F$ (Fig. 4.1)

Since, $\delta^*(q, y) = q$, $\forall k \geq 0$, $\delta^*(q, y^k) = q$. [$\delta^*(q, \varepsilon) = q$, $\delta^*(\delta^*(q, y), y^{k-1}) = q$]

- $\therefore \forall k \geq 0, xy^kz \in L$. (3) *pumping*
- $|y| = j - (i+1) + 1 = j - i > 0$ (1) *nonempty y*
- $|xy| = j \leq n$ (2) *first pump*

Contraverse of the Pumping lemma

If $L \in$ regular languages. Then

$$\exists n \geq 0,$$

$$\forall w \in L [. \exists. |w| \geq n],$$

$$\exists x, y, z [. \exists. w = xyz, y \neq \epsilon, |xy| \leq n],$$

$$\forall k \geq 0, xy^kz \in L.$$

If $\forall n \geq 0,$

$$\exists w \in L [. \exists. |w| \geq n],$$

$$\forall x, y, z [. \exists. w = xyz, y \neq \epsilon, |xy| \leq n],$$

$$\exists k \geq 0, xy^kz \notin L, \text{ Then}$$

$L \notin$ regular languages.

Two \forall 's. and two \exists 's.

\forall ': *adversarial game* “for all considered harmful”(?) pp 130

Example 4.2

$L_{eq} = \{w \in (0+1)^* \mid w \text{ has equal number of } 0\text{'s and } 1\text{'s}\}$ is **not regular**.

Proof

$$\forall n \geq 0,$$

$$\exists w = 0^n 1^n \in L_{eq} .\exists. |w| = 2n \geq n.$$

$$\forall x, y, z .\exists. w = xyz, y \neq \varepsilon, |xy| \leq n,$$

$$\Leftrightarrow \forall i, j .\exists. 0 \leq i \leq n, 1 \leq j \leq n \text{ where } i+j \leq n,$$

$$x = 0^i, y = 0^j, z = 0^{n-i-j} 1^n,$$

$$\exists k = 0, xy^k z = xz = 0^{n-j} 1^n \notin L_{eq}, \text{ since } j \neq 0.$$

$$\exists k = 2, xy^2 z = xy^2 z = 0^i 0^{2j} 0^{n-i-j} 1^n = 0^{n+j} 1^n \notin L_{eq}, \text{ since } j \neq 0.$$

...

$\therefore L_{eq}$ is **not regular**.

Example 4.3

$L_{pr} = \{w \in 1^* \mid |w| \text{ is a prime number}\}$ is **not regular**.

$$\forall n \geq 0,$$

$\exists w = 1^p \in L_{pr} .\exists. |w| = p \geq n + 2$ and p is a prime number.

$$\forall x, y, z .\exists. w = xyz, y \neq \varepsilon, |xy| \leq n,$$

$$\Leftrightarrow \forall m .\exists. m \geq 1, |y| = m, |xz| = p - m,$$

Consider $\exists k = p - m \geq 0$.

$$|xy^{p-m}z| = |xz| + |y|(p-m) = (p-m) + m(p-m) = (1+m)(p-m)$$

$$m+1 \neq 1 (\because m \geq 1), p-m \geq 2 (\because p \geq n + 2 \text{ and } m \leq n)$$

$\therefore (1+m)(p-m)$ is not prime.

$$\Leftrightarrow \exists k = p-m \geq 0, xy^{p-m}z \notin L_{pr}, \text{ since } j > 0.$$

$\therefore L_{pr}$ is **not regular**.

Theorem 4.1: (Pumping lemma)

If $\forall n \geq 0$,
 $\exists w \in L$ [. \exists . $|w| \geq n$],
 $\forall x, y, z$ [. \exists . $w = xyz$, $y \neq \varepsilon$, $|xy| \leq n$],
 $\exists k \geq 0$, $xy^kz \notin L$, Then

$L \notin$ regular languages.

Theorem 4.1': (Pumping lemma, Stronger Form)

If $\forall n \geq 0$,
 $\exists w \in L$ [. \exists . $|w| \geq n$],
 $\forall u, x, y, z, v$ [. \exists . $w = uxyzv$, $|xyz| \geq n$, $y \neq \varepsilon$, $|xy| \leq n$],
 $\exists k \geq 0$, $uxy^kzv \notin L$, Then

$L \notin$ regular languages.

4.2 Closure Properties of Regular Languages

Let \mathbb{L} be a class of languages and \otimes and \oplus be unary and binary operations on \mathbb{L} , respectively, i.e., $\otimes: \mathbb{L} \rightarrow \mathbb{L}$ and $\oplus: \mathbb{L} \times \mathbb{L} \rightarrow \mathbb{L}$

If $\forall L \in \mathbb{L}, \otimes L \in \mathbb{L}$ and $\forall L_1, L_2 \in \mathbb{L}, L_1 \oplus L_2 \in \mathbb{L}$, then we say that the class of languages \mathbb{L} has the closure property on the unary and binary operation \otimes and \oplus , respectively.

Theorem 4.4 If L and M are regular languages, then so is $L \cup M$, LM , and L^* are also regular.

Proof Since L and M are regular,

$\exists R, S \in$ regular expressions $\ni L = L(R)$, and $M = L(S)$.

$R + S$, RS , and $R^* \in$ regular expressions denoting

$L \cup M$, LM , and L^* , respectively.

$\therefore L \cup M$, LM , and L^* are also regular. (Read Closure under R.E. p133)

Theorem 4.5 *If L is a regular language, then $\bar{L}(= \Sigma^* - L)$ is also regular.*

Proof Let $L = L(A)$ for some DFA $A = (Q, \Sigma, \delta, q_0, F)$. Then $\bar{L} = L(B)$ where $B = (Q, \Sigma, \delta, q_0, \bar{F}(= Q - F))$ is a DFA.

*change the final states to nonfinal states and
nonfinal states to final states*

Example 4.6 $\overline{(0+1)^*01}$ is regular. (Figure 4.2)

Example 4.7 $L_{ne} = \{w \in (0+1)^* \mid w \text{ has not equal number of } 0\text{'s and } 1\text{'s}\}$
 $= \bar{L}_{eq}$ is not regular.

Read “What If Languages Have Different Alphabets?” in page 134

Theorem 4.8 *If L and M are regular, then $L \cap M$ is also regular.*

Proof 1. DeMorgan's law $L \cap M = \neg(\neg L \cup \neg M)$.

Proof 2. product construction *Let L and M be the languages of*

a DFA $A_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$ and $A_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$.

Then $A_{L \cap M} = (Q_L \times Q_M, \Sigma, \delta, (q_L, q_M), F_L \times F_M)$

where $\delta((p, q), a) = (\delta_L(p, a), \delta_M(q, a))$ is a DFA $L(A) = L \cap M$.

Proof: *Prove $\delta^*((q_L, q_M), w) = (\delta_L^*(q_L, w), \delta_M^*(q_M, w))$ by induction.*

Basis: $\delta^*((q_L, q_M), \varepsilon) = (q_L, q_M) = (\delta_L^*(q_L, \varepsilon), \delta_M^*(q_M, \varepsilon))$ *basis def of δ^**

Induction: $\forall x \in \Sigma^*, \forall a \in \Sigma, \delta^*((q_L, q_M), xa)$

$= \delta(\delta^*((q_L, q_M), x), a)$ *by recur. def. of δ^* .*

$= \delta((\delta_L^*(q_L, x), \delta_M^*(q_M, x)), a)$ *by ind. hyp.*

$= (\delta_L(\delta_L^*(q_L, x), a), \delta_M(\delta_M^*(q_M, x), a))$ *by def. of δ in Thm 4.8.*

$$= (\delta_L^*(q_L, xa), \delta_M^*(q_M, xa)) \quad \text{by recur. def. of } \delta^*.$$

Theorem 4.10 If L and M are **regular** languages,
then $L - M$ is also **regular**.

Proof 1. $L - M = L \cap \neg M$.

Proof 2. Production construction

$$A_{L-M} = (Q_L \times Q_M, \Sigma, \delta_{L \times M}, (q_L, q_M), F_{L-M})$$

where $\delta_{L \times M} = \delta$ in the proof 2 of Theorem 4.8 and

$$F_{L-M} = \{(f_L, f_M) \mid f_L \in F_L, f_M \notin F_M\}$$

$$A_{L \cap M} = (Q_L \times Q_M, \Sigma, \delta_{L \times M}, (q_L, q_M), F_{L \cap M})$$

where $F_{L \cap M} = \{(f_L, f_M) \mid f_L \in F_L, f_M \in F_M\} = F_L \times F_M$.

$$A_{L \cup M} = (Q_L \times Q_M, \Sigma, \delta_{L \times M}, (q_L, q_M), F_{L \cup M})$$

where $F_{L \cup M} = \{(f_L, f_M) \mid f_L \in F_L \text{ or } f_M \in F_M\}$

Reversal of Languages

Let $w = a_1 a_2 \dots a_n$. Then $w^R = a_n a_{n-1} \dots a_1$.

Let $L \subseteq \Sigma^*$. Then $L^R = \{w^R \mid w \in L\}$.

Theorem 4.11 If L is a **regular** language, then L^R is also **regular**.

Proof 1. simulate the automaton for L , $A = (Q, \Sigma, \delta, q_0, F)$ in reverse.

$B = (Q \cup \{p_0\}, \Sigma, \delta^R, p_0, \{q_0\})$ where $p_0 \notin Q$.

$$\delta^R(p, a) = \{q \mid p \in \delta(q, a)\} \cup \{\delta^R(p_0, \varepsilon) = F\}$$

Proof 2. Assume $L(E) = E$ for regular expression E .

There is a regular expression E^R , such that $L(E^R) = (L(E))^R$.

structural induction on $|E|$

Basis: If E is ε , \emptyset , and $a \in \Sigma$, then $\varepsilon^R = \varepsilon$, $\emptyset^R = \emptyset$, and $a^R = a$.

$$L(\varepsilon^R) = L(\varepsilon) = \{\varepsilon\} = L(\varepsilon)^R, L(\emptyset^R) = L(\emptyset) = \emptyset = L(\emptyset)^R, \text{ and}$$

$$L(a^R) = L(a) = \{a\} = L(a)^R.$$

Induction: $L(E^R) = (L(E))^R$.

1. If $E = E_1 + E_2$, then $E^R = E_1^R + E_2^R$.

$$\begin{aligned} L(E^R) &= L(E_1^R + E_2^R) = L(E_1^R) \cup L(E_2^R) \\ &= L(E_1)^R \cup L(E_2)^R = L(E_1 + E_2)^R = (L(E))^R. \end{aligned}$$

2. If $E = E_1E_2$, then $E^R = E_2^RE_1^R$.

$$\begin{aligned} L(E^R) &= L(E_2^RE_1^R) = (L(E_2))^R(L(E_1))^R = (L(E_1E_2))^R = (L(E))^R. \\ &\text{since } w_2^Rw_1^R = (w_1w_2)^R. \end{aligned}$$

3. If $E = E_1^*$, then $E^R = (E_1^R)^*$.

$$L(E^R) = L((E_1^R)^*) = (L(E_1)^*)^R = (L(E))^R.$$

4.2.3 Homomorphism $(h, A, B, \oplus_A, \oplus_B)$

Skip in this lecture

$h: A \rightarrow B$, $\oplus_A: A \times A \rightarrow A$, and $\oplus_B: B \times B \rightarrow B$.

$$\forall a, b \in A, h(a \oplus_A b) = h(a) \oplus_B h(b).$$

If h is onto $|A| \geq |B|$.

string homomorphism from Σ to Δ^ .* $(h, \Sigma, \Delta^*, \cdot)$

$$h: \Sigma \rightarrow \Delta^*, \oplus_A = \cdot: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*, \oplus_B = \cdot: \Delta^* \times \Delta^* \rightarrow \Delta^*.$$

$$h(a \cdot b) = h(a) \cdot h(b)$$

$$h(a_1 \cdot a_2 \cdot \dots \cdot a_n) = h(a_1) \cdot h(a_2) \cdot \dots \cdot h(a_n)$$

domain of h can be extended to Σ^ and 2^{Σ^*} .* $(h, \Sigma^*, \Delta^*, \cdot)$

Example 4.13

Theorem 4.14 *If h is a homomorphism from Σ to Δ , and L is a regular language over Σ , then $h(L)$ is also regular.*

Proof: *Let E be a regular expression over Σ .*

If $h(E)$ be the expression obtained by replacing $a \in \Sigma$ in E by $h(a)$.

We are to claim $h(E)$ is a regular expression and $L(h(E)) = h(L(E))$.

Basis: *If E is ε or \emptyset . $h(E) = \varepsilon$ or \emptyset . $L(\varepsilon) = \{\varepsilon\} = h(\{\varepsilon\})$. $L(\emptyset) = \emptyset = h(\emptyset)$.*

If $E = \mathbf{a}$, $a \in \Sigma$. $h(E) = h(\mathbf{a}) = h(a)$: r.e. over Δ . $L(h(\mathbf{a})) = L(h(a)) = h(a)$.

$h(L(\mathbf{a})) = h(\{a\}) = h(a)$.

Induction: *If $E = F + G$, $h(E) = h(F) + h(G)$, $h(F) + h(G)$: r.e. over Δ .*

$L(h(E)) = L(h(F) + h(G)) = L(h(F)) \cup L(h(G))$.

$h(L(E)) = h(L(F) \cup L(G)) = h(L(F) \cup h(L(G)))$

$L(h(F)) = h(L(F))$ and $L(h(G)) = h(L(G))$ I.H.

concatenation and closure are similar

Let R be a regular expression \exists . $L(R) = L$, $h(R)$ is also a regular expression and $L(h(R)) = h(L(R)) = h(L)$. $\therefore h(L)$ is also regular.

Inverse Homomorphism

$$h^{-1}: \Delta^* \rightarrow \Sigma^*.$$

$$h^{-1}: 2^{\Delta^*} \rightarrow 2^{\Sigma^*}.$$

$$\text{Let } L \in 2^{\Delta^*}. h^{-1}(L) = \{h^{-1}(w) \mid w \in L\}$$

Example 4.15 $h(a) = 01, h(b) = 10, L = L((\mathbf{00+1})^*) = (\mathbf{00+1})^*.$

$$h^{-1}(L) = L((\mathbf{ba})^*) = (\mathbf{ba})^*.$$

Proof: $h(w) \in L$ if and only if $w \in (\mathbf{ba})^*.$

(If) Suppose $w = (ba)^n, n \geq 0, (w = (ba)^0 = \varepsilon).$

$$h((ba)^n) = (1001)^n \in L = (\mathbf{00+1})^*.$$

(Only if) Assume $w \notin (\mathbf{ba})^*.$

1) w begins with a , then $h(w)$ begins with $01, \therefore h(w) \notin (\mathbf{00+1})^*.$

2) w ends in b , then $h(w)$ ends in $10, \therefore h(w) \notin (\mathbf{00+1})^*.$

3) w has two consecutive a 's, then $h(w)$ has substring 0101

$\therefore h(w) \notin (\mathbf{00+1})^*$.

4) w has two consecutive b 's, then $h(w)$ has substring 1010

$\therefore h(w) \notin (\mathbf{00+1})^*$.

5) Otherwise $w \in (\mathbf{ba})^*$.

Assume none of 1) through 4) holds

1) w begins with a

2) w ends in b

3) 4) a 's and b 's must alternate in w

$\therefore w \in (\mathbf{ba})^*$.

Theorem 4.16 If h is a **homomorphism** from Σ to Δ , and

L is a **regular** language over Δ , then $h^{-1}(L)$ is also **regular**.

Proof: Let $L = L(A)$ where DFA $A = (Q, \Delta, \delta, q_0, F)$. Define a DFA B

$B = (Q, \Sigma, \gamma, q_0, F)$ where $\gamma(q, a) = \delta^*(q, h(a))$, $a \in \Sigma$, $h(a) \in \Delta^*$.

It is easy to show that $\gamma^*(q_0, w) = \delta^*(q_0, h(w))$, $w \in \Sigma^*$.

$\therefore L(B) = h^{-1}(L)$. $\therefore h^{-1}(L)$ is **regular**.

4.3 Decision Properties of Regular Languages

Every finite languages are regular.

regular expressions

There are infinite languages that is regular.

closure

Finite representations of possibly infinite regular languages

DFA, NFA, ϵ -NFA, XFA(FA), regular expressions

finite automata, regular expressions

Definition of regular language?

DFA, NFA, ϵ -NFA, XFA(FA),

regular expressions, regular grammars

many-to-one mappings

Decision problems on regular languages

1. Given a regular language, is L **empty** or not?

Is it reachable from initial state to final states in finite automaton?

2. Given a regular language, is L **finite** or not?

Is there cycle for non-empty string in fa,

*Is there * except ε^* and \emptyset^* , in re?*

3. Given $w \in \Sigma^*$ and regular language L , is $w \in L$?

membership problem

simulate DFA

4. Given two regular language L_1 and L_2 , are they **equivalent** or not?

unique representation for each regular language?

minimal state DFA?

4.4 Equivalence and Minimization of Automata

Equivalence of two descriptions of regular languages.

regular expressions, FA's.

Minimize the states of DFA(m-DFA)

unique representation for regular language

*State p and q are **equivalent**, denoted $p \equiv q$ if*

$\forall w \in \Sigma^$, $\delta^*(p, w) \in F$ if and only if $\delta^*(q, w) \in F$. (DFA)*

If $\delta^(p, w) \in F$, then $\delta^*(q, w) \in F$, and*

if $\delta^(p, w) \notin F$, then $\delta^*(q, w) \notin F$.*

distinguishable, otherwise.

*The state of **minimal state DFA(m-DFA)***

summarizes** the information concerning **past inputs

*that is **needed to determine** the **behavior** of the automata*

*on **subsequent inputs**.*

Two states p and q are **distinguishable**, denoted $p \neq q$, if
 $\exists w \in \Sigma^*, \delta^*(p, w) \in F$ but $\delta^*(q, w) \notin F$ or vice versa.

Example 4.18 (pp 156)

Recursive algorithm for finding distinguishable states
table filling algorithm, repeat until no change

Basis: If $p \in F$ and $q \notin F$, then $p \neq q$.

Induction: $\forall p, q \in Q$ (. \exists . $p \neq q$ is not found yet)

if $\exists a \in \Sigma$. \exists . $\delta(p, a) \neq \delta(q, a)$, then $p \neq q$.

Repeat the above induction process until

no new indistinguishable state pairs is found $\forall p, q \in Q$

$n(n-1)/2 \in O(n^2)$

Example 4.19 (pp 157)

Theorem 4.20 *If two states are **not distinguishable** by the above alg., then the states are **equivalent**.*

Proof: *Assume state p and q are distinguishable, **but** the above alg. fails to find p and q to be distinguished. (**bad pair**)*

*Since p and q are the **bad pair**, distinguished by the **shortest** string u .*

$\exists u \in \Sigma^* . \exists . \delta^*(p, u) \in F$ but $\delta^*(q, u) \notin F$. (w.l.o.g.)

1. $u \neq \varepsilon$, since p and q must be checked distinguishable in the **basis**.

2. Let $u = av$ ($a \in \Sigma, v \in \Sigma^*$), $s = \delta(p, a)$, and $t = \delta(q, a)$.

Since $\delta^(p, av) = \delta^*(s, v) \in F$ but $\delta^*(q, av) = \delta^*(t, v) \notin F$. (w.l.o.g.), (s, t) pair must be found to be **distinguishable** by the alg.*

*If the alg. can found (s, t) pair in k -th iteration, it must have added (p, q) pair in $(k+1)$ -th iteration **contradicting** the fact that (p, q) pair couldn't be found by the alg.*

$\therefore s$ and t should be distinguishable, for the string v .

*But v is **shorter** than $u (=av)$.*

Theorem 4.23 *The equivalence of state is transitive.*

Proof *Let's assume $p \equiv q$ and $q \equiv r$.*

$\forall w \in \Sigma^*, \delta^*(p, w) \in F$ if and only if $\delta^*(q, w) \in F$ and

$\forall x \in \Sigma^*, \delta^*(q, x) \in F$ if and only if $\delta^*(r, x) \in F$.

$\therefore \forall v \in \Sigma^*, \delta^*(p, v) \in F$ if and only if $\delta^*(r, v) \in F$.

$\therefore p \equiv r$.

$\equiv \subseteq Q \times Q$ *binary relation on Q*

\equiv *is reflexive, symmetric, and transitive.*

$\therefore \equiv$ *is **equivalent** relation on Q*

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Then equivalent **minimal state DFA**

$M = (Q_{\equiv}, \Sigma, \delta_{\equiv}, [q_0]_{\equiv}, F_{\equiv})$ is defined as follow.

1. $Q_{\equiv} = \{[q]_{\equiv} \mid q \in Q\}$ **equivalence partition**
3. $\delta_{\equiv}: Q_{\equiv} \times \Sigma \rightarrow Q_{\equiv}$ where $\delta_{\equiv}([q]_{\equiv}, a) = [\delta([q]_{\equiv}, a)]$
5. $F_{\equiv} = \{[f]_{\equiv} \mid f \in F\}$

proof $[q]_{\equiv} = \{p \in Q \mid q \equiv p\}$ **equivalence class, block**

If $q \equiv p$, $[q]_{\equiv} = [p]_{\equiv}$, and if $q \not\equiv p$, $[q]_{\equiv} \cap [p]_{\equiv} = \emptyset$; $\cup_{q \in Q} [q]_{\equiv} = Q$.

$\forall q \in Q, \forall a \in \Sigma, \forall p \in [q]_{\equiv}, \delta(p, a) \in [\delta([q]_{\equiv}, a)]_{\equiv}$.

$\therefore \delta_{\equiv}([q]_{\equiv}, a) = [\delta([q]_{\equiv}, a)]_{\equiv}$.

$\forall x \in \Sigma^*, \delta^*(q_0, x) \in \delta_{\equiv}^*([q_0]_{\equiv}, w)$.

$\therefore L(A) = L(M)$

Note that $|Q_{\equiv}| \leq |Q|$ for any DFA Q .

We define an **equivalent** relation (R_L) on Σ^* for some **regular** $L \subseteq \Sigma^*$, as

$x R_L y$, iff $\forall z \in \Sigma^*$, $xz, yz \in L$ or $xz, yz \notin L$ (**right invariant**). Then

$[x]_{R_L} = \{y \in \Sigma^* \mid x R_L y\}$ is a **partion** on $L \subseteq \Sigma^*$ and

If $|\{[x]_{R_L}\}|$ is **finite**, then

$M = (\{[x]_{R_L}\}, \Sigma, \delta, q_0, F)$ is **minimal state DFA** accepting L

where $\delta([x]_{R_L}, a) = [xa]_{R_L}$, $\cup_{f \in F} f = L$.

right invariant

Myhill-Nerode Theorem

See p65-67 of

Introduction to Automata Theory, Languages, and Computation,

the 2nd Ed., J. Hopcroft and I. Ullman(1979).

omitted in the 3rd. ed.(2006)

Example 4.25(p163)

$$\begin{array}{lll}
 A = 0B + 1F & A \equiv E & \{A, E\} \\
 B = 0G + 1C & B \equiv H & \{B, H\} \\
 C = 0A + 1C + \varepsilon & & \{C\} \\
 D = 0C + 1G & D \equiv F & \{D, F\} \\
 E = 0H + 1F & & \{E\} \\
 F = 0C + 1G & & \\
 G = 0G + 1E & & \\
 H = 0G + 1C & & 8 \text{ states}
 \end{array}$$

$$\begin{array}{ll}
 \{A, E\} = 0\{B, H\} + 1\{D, F\} \\
 \{B, H\} = 0\{B, H\} + 1\{C\} \\
 \{C\} = 0\{A, E\} + 1\{C\} + \varepsilon \\
 \{D, F\} = 0\{C\} + 1\{B, G\} \\
 \{E\} = 0\{B, H\} + 1\{D, F\} & 5 \text{ states}
 \end{array}$$

The state of *minimal state DFA* (*m-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.

Following statements are logically equivalent

1. L is **regular**.
2. $L = L(D)$ for some DFA D with total function δ .
3. $L = L(P)$ for some DFA P with partial function δ .
4. $L = L(N)$ for some NFA N .
5. $L = L(E)$ for some ε -NFA E .
6. $L = L(X)$ for some XFA X .
7. $L = L(R)$ for some RE R .
8. $L = L(M)$ for some m -DFA M with **minimal states**.

Following statements are logically equivalent

1. L is **regular**.
2. $L = L(A)$ for some **finite automaton** A .
3. $L = L(R)$ for some **regular expression** R .
4. $L = L(G)$ for some **regular grammar** G . (TBD in Chap. 5)