Chap. 0 Introduction, Notations and Terminology

- **Alphabet** \( \Sigma \): a finite nonempty set of symbols.
- **String** \( w \): a finite sequence of symbols chosen from some alphabet.
- The *empty string*: \( \epsilon, \varepsilon, \lambda, \text{ or } \Lambda \).
- \( |w| \): The *length* of a string \( w \).
- \( \Sigma^k \): The set of strings of length \( k \), each of whose symbols is in \( \Sigma \).
- \( \Sigma^* \): The set of strings over \( \Sigma \).
- \( \Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \cdots \).
- \( \Sigma^0 = \{ \epsilon \} \).

**Concatenations** of strings:
\( xy \) denotes the concatenation of \( x \) and \( y \).
\[ |xy| = |x| + |y| \]
\[ \epsilon w = w \epsilon = w \]

**Language**: a subset \( L \) of \( \Sigma^* \) is called a language over \( \Sigma \). Examples: \( \Sigma^*, \emptyset, \{ \epsilon \} \).

Chap. 1 Regular Languages

A *class* of languages is a set of languages.
Examples: DFA class, NFA class, \( \epsilon \)-less NFA class, RE class (Regular Expression).
We will show in Chapter 1 that all these four computation models recognize (describe) the same class of languages.
The textbook (Sipser2, p.40 Def. 1.16) defines a regular language as a set of strings \( L \) such that \( L=L(M) \) for some DFA \( M \). Other textbooks may choose otherwise.

Since the four classes have been proved equivalent, it does not matter which class was chosen by the author.

Section 1.1 (Deterministic) Finite Automata (DFA) 

\[ M = (Q, \Sigma, \delta, q_0, F) \]  

where 

\( Q \): finite set of states.  

\( \Sigma \): input alphabet.  

\( \delta \): transition function \( Q \times \Sigma \rightarrow Q \).  

\( q_0 \): start state or initial state.  

\( F \): a set of accept (final) states, \( F \subseteq Q \).  

\( \hat{\delta} \): Extended transition function \( Q \times \Sigma^* \rightarrow Q \).  

Let \( w = xa, w \in \Sigma^*, x \in \Sigma^*, a \in \Sigma \), and \( |w| \geq 1 \). Define  

\[ \hat{\delta}(q, \epsilon) = q \]  

\[ \hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a). \]  

Claim: If \( \delta \) is a function \( Q \times \Sigma \rightarrow Q \), then \( \hat{\delta} \) is a function \( Q \times \Sigma^* \rightarrow Q \).
Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and let $w = w_1w_2...w_n$ be a string where each $w_i$ is in $\Sigma$. Then, we say $M$ accepts $w$ if a sequence of states $r_0, r_1, ..., r_n$ in $Q$ exists with three conditions:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, 1, ..., n - 1$, and
3. $r_n \in F$.

We say $M$ recognizes language $A$ if $A = \{w : M$ accepts $w\}$. Using the notation $\hat{\delta}$ we say $M$ accepts a string $w$ if $\hat{\delta}(q_0, w) \in F$.

The language of the DFA $M = (Q, \Sigma, \delta, q_0, F)$ is then defined as

$L(M) = \{w : \hat{\delta}(q_0, w) \in F\}$.

(See Sipser2 p.36) Sipser prefers not to say $M$ accepts some language $L$. He prefers to say $M$ recognizes some language $L$. It is perfectly alright to say $M$ accepts some string $w$. It is wrong to say $M$ accepts $L$ when $M$ accepts every string in $L$ and also some string not in $L$. One thing to remember: we say $M$ recognizes (accepts) $L$ only if $M$ accepts exactly those strings in $L$. No more, no less.

In summary:

$M$ accepts $w$ if $\hat{\delta}(q_0, w) \in F$

$M$ recognizes $L$ if $L = L(M)$

$M$ accepts $L$ if $L = L(M)$

Section 1.2 Nondeterministic Finite Automata (NFA)

First, we define $\epsilon$-less NFA , which is a restricted kind of NFA. Later, we define NFA which is capable of making $\epsilon$-moves.
\( \epsilon \)-less NFA \( A = (Q, \Sigma, \delta, q_0, F) \) where
\( \delta : \text{transition function } Q \times \Sigma \to 2^Q \).

\( \hat{\delta} \): Extended transition function \( Q \times \Sigma^* \to 2^Q \).
Let \( w = xa, w \in \Sigma^*, x \in \Sigma^*, a \in \Sigma, \) and \( |w| \geq 1 \). Define
\[
\hat{\delta}(q, \epsilon) = \{q\}
\]
\[
\hat{\delta}(q, xa) = \bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a)
\]
In other words, suppose
\[
\hat{\delta}(q, x) = \{p_1, p_2, \ldots, p_k\}
\]
then
\[
\hat{\delta}(q, w) = \hat{\delta}(q, xa) = \bigcup_{i=1}^{k} \delta(p_i, a)
\]

\( L(A) \): The language of an \( \epsilon \)-less NFA \( A = (Q, \Sigma, \delta, q_0, F) \) is defined as:
\[
L(A) = \{ w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset \}.
\]

Notation: \( \bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a) := \{ r \mid (\exists p)[ p \in \hat{\delta}(q,x) \text{ and } r \in \delta(p, a)] \} \)

NFA

NFA \( A = (Q, \Sigma, \delta, q_0, F) \) where
\( \delta: \text{transition function } Q \times \Sigma \cup \{\epsilon\} \to 2^Q \)

\( \epsilon \)-closure of a state \( q \) is denoted by \( E(q) \).
Define \( E(q) \) as the smallest set such that
1. \( q \) is in \( E(q) \), and
2. if \( p \) is in \( E(q) \) then every state in \( \delta(p, \epsilon) \) is also in \( E(q) \).
\( \epsilon \)-closure of a set of states \( S \) is denoted by \( E(S) \).

We say a set \( S \) is \( \epsilon \)-closed if \( S = E(S) \).
Now we can define $\hat{\delta}$ formally as follows: $\hat{\delta}: Q \times \Sigma^* \to 2^Q$

$$\hat{\delta}(q, \epsilon) = E(q)$$
$$\hat{\delta}(q, xa) = E(\bigcup_{p \in \delta(q, x)} \delta(p, a))$$

Define $L(A) = \{ w | \hat{\delta}(q_0, w) \cap F \neq \emptyset \}$.

**Theorem 1.39:** (Sipser2 p.55) Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

(See also the Proof in Sipser2 p.55) Before doing the full construction, let’s first consider the easier case wherein $N$ has no $\epsilon$-transitions. Later we take the $\epsilon$-transitions into account.

**Lemma to the theorem:** If $L$ is a language of an $\epsilon$-less NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$, then there is a DFA $D = (Q_D, \Sigma, \delta_D, \{ q_0 \}, F_D)$ such that $L(D) = L(N) = L$.

**Proof of the lemma** Given $\epsilon$-less NFA $N$, how to find the corresponding $D$?

By subset construction!

$Q_D$ is the power set of $Q_N$

$F_D$ is the set of subsets $S$ of $Q_N$ such that $S \cap F_N \neq \emptyset$

Note: For any subset $T$ of $Q_N$, we have $T \in F_D$ iff $T \cap F_N \neq \emptyset$.

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$$

**Claim:** $\hat{\delta}_D(\{ q_0 \}, w) = \hat{\delta}_N(q_0, w)$ for every $w \in \Sigma^*$.

**Proof:** Induct on $|w|$

**BASIS:** $|w| = 0$

$$\hat{\delta}_D(\{ q_0 \}, \epsilon) = \{ q_0 \} = \hat{\delta}_N(q_0, \epsilon)$$

**INDUCTION:** Assume $|w| = n+1$, $w = xa$, and $\hat{\delta}_D(\{ q_0 \}, x) = \hat{\delta}_N(q_0, x)$.

Then we must show that $\hat{\delta}_D(\{ q_0 \}, xa) = \hat{\delta}_N(q_0, xa)$.
\[ \hat{\delta}_D(\{q_0\}, xa) = \delta_D(\hat{\delta}_D(\{q_0\}, x), a) \]
\[ = \delta_D(\hat{\delta}_N(q_0, x), a) \quad \text{DFA } \delta \]
\[ = \bigcup_{p \in \hat{\delta}_N(q_0, x)} \delta_N(p, a) \quad \text{Hypothesis} \]
\[ = \hat{\delta}_N(q_0, xa) \quad \text{NFA } \hat{\delta} \]

**Claim:** \( L(D) = L(N) \).

**Proof:** Recall \( L(D) = \{ w \mid \hat{\delta}_D(\{q_0\}, w) \in F_D \} \) where \( F_D \) is the set of subsets \( S \) of \( Q_N \) such that \( S \cap F_N \neq \emptyset \).

\[ L(N) = \{ w \mid \hat{\delta}_N(q_0, w) \cap F_N \neq \emptyset \} \]
\[ = \{ w \mid \hat{\delta}_D(\{q_0\}, w) \cap F_N \neq \emptyset \} \]
\[ = \{ w \mid \hat{\delta}_D(\{q_0\}, w) \in F_D \} = L(D) \]

\[ \square \]

**Theorem 1.39 restated:** If a language \( L \) is recognized by an NFA \( E = (Q_E, \Sigma, \delta_E, q_0, F_E) \), then there is a DFA \( D = (Q_D, \Sigma, \delta_D, q_D, F_D) \) such that \( L(E) = L(D) = L \).

**Proof of the theorem:**

Construct \( D \) as follows:

1. \( Q_D \) is the set of the subsets of \( Q_E \) that are \( \epsilon \)-closed.

2. \( q_D = E(q_0) \).

3. \( F_D = \{ S \mid S \text{ is in } Q_D \text{ and } S \cap F_E \neq \emptyset \} \).

4. For each \( S \in Q_D \), each \( a \in \Sigma \), define \( \delta_D(S, a) = E(\bigcup_{p \in S} \delta_E(p, a)) \).

Before we prove \( L(E) = L(D) \), we show that \( \hat{\delta}_E(q_0, w) = \hat{\delta}_D(q_D, w) \) for every \( w \in \Sigma^* \).

**Claim:** \( \hat{\delta}_E(q_0, w) = \hat{\delta}_D(q_D, w) \) for every \( w \in \Sigma^* \).

**Proof:** Induct on \( |w| \).

Let \( |w| = 0 \).

\[ \hat{\delta}_E(q_0, \epsilon) = E(q_0) \]
\[ = q_D \]
\[ = \hat{\delta}_D(q_D, \epsilon) \]

Let \( |w| = n + 1 \). Assume \( w = xa \) and \( \hat{\delta}_E(q_0, x) = \hat{\delta}_D(q_D, x) \).

Then
\[ \hat{\delta}_E(q_0, xa) = E(\bigcup_{p \in \hat{\delta}_E(q_0, x)} \delta_E(p, a)) \]
\[ = E(\bigcup_{p \in \hat{\delta}_D(q_D, x)} \delta_E(p, a)) \]
\[ = \delta_D(\hat{\delta}_D(q_D, x), a) \]
\[ (\because \text{the definition of } \delta_D \text{ in the construction of } D \text{ from } E.) \]
\[ = \hat{\delta}_D(q_D, xa) \]
\[ (\because \text{the definition of } \hat{\delta} \text{ in DFA.}) \]

**Claim:** \( L(D) = L(E) \).

This can be verified by observing that

\[ F_D = \{ S \mid S \text{ is in } Q_D \text{ and } S \cap F_E \neq \emptyset \} \]
\[ L(D) = \{ w \mid \hat{\delta}_D(q_D, w) \text{ is in } F_D \} \]
\[ L(E) = \{ w \mid \hat{\delta}_E(q_0, w) \cap F_E \neq \emptyset \} \]

\[ \square \]

**Corollary 1.40, Sipser2 p.56:** A language is regular if and only if some nondeterministic finite automaton recognizes it.

Proof: Since a DFA automaton is also an NFA automaton, the “only if” half is confirmed. The “if” half follows from Theorem 1.39.

**Section 1.3 Regular Expressions**

**Definition:** Suppose \( \Sigma = \{a, b\} \).

**BASIS:**

1. \( \epsilon \) is a regular expression; \( L(\epsilon) = \{ \epsilon \} \).
2. \( \emptyset \) is a regular expression; \( L(\emptyset) = \emptyset \).
3. \( a \) is a regular expression; \( L(a) = \{a\} \).
4. \( b \) is a regular expression; \( L(b) = \{b\} \).

**INDUCTION:**

1. **union**

   If \( E \) and \( F \) are regular expressions, then \( E \cup F \) is a regular
expression; $L(E \cup F) = L(E) \cup L(F)$, i.e., the union of the languages $L(E)$ and $L(F)$.

2. **concatenation**
   
   If $E$ and $F$ are regular expressions, then $EF$ is a regular expression; $L(EF) = L(E)L(F)$, i.e., the concatenation of the languages $L(E)$ and $L(F)$.

3. **star**
   
   If $E$ is a regular expression, then $E^*$, (also known as Kleene closure )is a regular expression; $L(E^*) = (L(E))^*$, of the language $L(E)$.

4. If $E$ is a regular expression, then $(E)$ is a regular expression; $L((E)) = L(E)$.

**Definition:** Let $L$, $M$, and $N$ be languages. Define

- **Union** $L \cup M = \{w | w \in L \text{ or } w \in M\}$
- **Concatenation** $LM = \{w_1w_2 | w_1 \in L \text{ and } w_2 \in M\}$
- **Star** $N^* = N^0 \cup N \cup NN \cup \cdots$ 
  $= N^0 \cup N^1 \cup N^2 \cup \cdots$

**Notes:**

1. $L\emptyset = \emptyset$. Why? By the definition we have $L\emptyset = \{w_1w_2 | w_1 \in L \text{ and } w_2 \in \emptyset\}$. But the empty set $\emptyset$ has no element at all, hence “$w_2 \in \emptyset$” is not true; therefore the $w_1w_2$ formation cannot succeed.

2. $L^0 = \{\epsilon\}$ for any language $L$, including the empty set $\emptyset$.

3. $\emptyset^* = \{\epsilon\}$

4. $\emptyset^0 = \{\epsilon\}$

**Lemma 1.55:** If a language is described by a regular expression, then it is regular.(Restated: Every language described by a regular expression is also recognized by an NFA.)
Proof: By structural induction on the given regular expression $R$.

**Basis:**

- $R = \epsilon$

![Image of ε transition]

- $R = \emptyset$

![Image of empty set transition]

- $R = a$

![Image of a transition]

- $R = b$

![Image of b transition]

**Induction:**

- $R' = R \cup S$

![Image of union of states]

- $R' = RS$

![Image of product of states]

- $R' = R^*$
Lemma 1.60: If a language is regular, then it is described by
a regular expression. (Restated: If \( L = L(A) \) for some DFA \( A \),
then there is a regular expression \( R \) such that \( L = L(R) \).)

Proof: Without loss of generality (W.L.O.G.) assume that \( A \)'s
states are \( \{1, 2, \ldots, n\} \) for some integer \( n \), and state 1 is the
start state of \( A \). Let \( R_{ij}^{(k)} \) denote the regular expression whose
language is the set of strings \( w \) such that \( w \) is the label of a path
from state \( i \) to state \( j \) in \( A \), and the path has no intermediate
node whose number is greater than \( k \).

It is easy to decide \( R_{ij}^{(0)} \) for every pair \((i, j)\) of states once the
DFA \( A \) is given.

Then we can decide \( R_{ij}^{(k)} \) for any \( k \in \mathbb{N} \) by:

\[
R_{ij}^{(k)} = R_{ij}^{(k-1)} \cup R_{ik}^{(k-1)} (R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}.
\]

Then the regular language \( R \) for \( L(A) \) is the union of all ex-
pressions \( R_{1j}^{(n)} \) such that state \( j \) is an accepting state of \( A \).

We complete the whole plan of the diagram:

\[ R' = (R) \quad \text{No change.} \]
Are there other concepts for regular languages? Yes!

Section 1.4 Non-regular Languages (and Pumping lemma)

Fact: There are some languages that are not regular.
Evidence? Ad-hoc proof (by contradiction).
Ex: \( L_{01} = \{0^n1^n | n \geq 1 \} \) can be shown not to be regular by an ad-hoc argument.

We prefer some method that is more general.

**Pumping Lemma for regular languages** (Theorem 1.70 Sipser2 p.78)

Let \( L \) be any regular language. There exists a constant \( n \) such that for every string \( w \) in \( L \) such that \( |w| \geq n \), there is a way to break \( w \) into three strings, \( w = xyz \), where

1. \( y \neq \epsilon \),
2. \( |xy| \leq n \), and
3. for all \( k \geq 0 \), the string \( xy^kz \) is also in \( L \).

Ex: \( L_{eq} = \{w | w \in \{0,1\}^*, \#(1) \text{ in } w = \#(0) \text{ in } w \} \). (i.e. no.
Use the pumping lemma to prove that \( L_{eq} \) is not regular.

**Proof:** By way of contradiction (B.W.O.C.) assume \( L_{eq} \) is regular. We derive a contradiction by using pumping lemma.

For any \( n \) that the adversary may pick, we choose \( w = 0^n1^n \), which is in \( L \).

For any way that the adversary may divide \( w \) into \( xyz \), since \( y \neq \varepsilon \) and \( |xy| \leq n \), we know \( x \) and \( y \) must contain only 0’s.

Hence, \( xz \) has \( n \) 1’s, and \( xz \) has fewer than \( n \) 0’s, since \( y \neq \varepsilon \) and \( y \) contains only 0’s.

\[ \therefore xz \notin L . \]

But pumping lemma says \( xy^0z \in L \), a contradiction! \( \square \)

**Closure properties of Regular Languages**

1. union \( L \cup M \)
2. intersection \( L \cap M \)
3. complement \( \overline{L} \)
4. difference \( L - M \)
5. reversal \( L^R \)
6. star \( L^* \)
7. concatenation \( LM \)
8. homomorphism \( h(L) \)
9. inverse homomorphism \( h^{-1}(L) \)

\[
L \cap M = \overline{L} \cap \overline{M} = \overline{L \cup M} \\
L - M = L \cap \overline{M}
\]
Union, star, and concatenation are already known to be closure properties of regular languages.

Proving that complement is closed will settle all of the closure properties except reversal, homomorphism, and inverse homomorphism.

**Exercise 1.14a p.85:** If $L$ is a regular language over $\Sigma$, then $\overline{L} = \Sigma^* - L$ is also regular.

**Proof outline:** Let $L = L(A)$ for a DFA $A = (Q, \Sigma, \delta, q_0, F)$. Let $B = (Q, \Sigma, \delta, q_0, Q - F)$. Then $L(B) = \overline{L}$.

**Reversal (Problem 1.31 p.88)**

**String reversal $w^R$:**

1. If $w = \epsilon$, then $w^R = w = \epsilon$.
2. If $w = xa$, then $w^R = ax^R$.

**Claim:** $(xy)^R = y^Rx^R$.

**Claim:** $(x_1x_2 \cdots x_n)^R = x_n^Rx_{n-1}^R \cdots x_1^R$.

**Set reversal:** $L^R = \{w^R \mid w \in L\}$.

**Theorem:** If $L$ is regular, so is $L^R$.

**Proof:** Let $L = L(E)$ for some regular expression $E$.

We will define an expression $E^R$ and show that $L^R = L(E^R)$. $E^R$ will be shown to be a regular expression, hence $L^R$ is regular.

Prove by structural induction on $E$.

**Basis:** If $E$ is $\epsilon$, $\emptyset$, or $a$, define $E^R$ to be $E$. $E^R$ is a regular expression.

**Induction:**

1. If $E = E_1 \cup E_2$, define $E^R = E_1^R \cup E_2^R$. $E^R$ is a regular expression.
Let $L_1 = L(E_1), L_2 = L(E_2)$. By induction hypothesis, we have $L_1^R = L(E_1^R), L_2^R = L(E_2^R)$. We know $L = L(E) = L(E_1 \cup E_2) = L(E_1) \cup L(E_2)$.

::: \[ L^R = (L(E_1))^R \cup (L(E_2))^R \]

= $L_1^R \cup L_2^R$

= $L(E_1^R) \cup L(E_2^R)$ \hspace{1cm} \text{Hypothesis}

= $L(E_1^R \cup E_2^R)$

= $L(E^R)$

2. If $E = E_1E_2$, define $E^R = E_2^RE_1^R$. $E^R$ is a regular expression.

Let $L_1 = L(E_1), L_2 = L(E_2)$. By induction hypothesis, we have $L_1^R = L(E_1^R), L_2^R = L(E_2^R)$. We know $L = L(E) = L(E_1E_2) = L(E_1)L(E_2)$. Therefore,

\[
L^R = \{w^R \mid w \in L\}
\]

= $\{(w_1w_2)^R \mid w_1 \in L(E_1) \land w_2 \in L(E_2)\}$

= $\{w_2^Rw_1^R \mid w_1 \in L(E_1) \land w_2 \in L(E_2)\}$

= $(L(E_2))^R(L(E_1))^R$

= $L_2^RL_1^R$

= $L(E_2^R)L(E_1^R)$ \hspace{1cm} \text{Hypothesis}

= $L(E_2^RE_1^R)$

= $L(E^R)$

3. If $E = E_1^*$, define $E^R = (E_1^R)^*$. $E^R$ is a regular expression.

Let $L_1 = L(E_1)$. By induction hypothesis, we have $L_1^R = L(E_1^R)$.

We know $L = L(E) = L(E_1^*) = (L(E_1))^* = L_1^*$. \hspace{1cm} \text{Hypothesis}

$\therefore$ \[ L^R = \{w^R \mid w \in L\} = \{w^R \mid w \in L_1^*\} = (L_1^*)^R \]

Since $(M^*)^R = (M^R)^*$ for any regular set $M$, we have
\[ L^R = (L^*_1)^R \]
\[ = (L^R_1)^* \]
\[ = (L(E^R_1))^* \] Hypothesis
\[ = L((E^R_1)^*) \]
\[ = L(E^R) \]

Lemma: \((M^*)^R = (M^R)^*\) for any regular set \(M\).

String Homomorphisms

Let \(h : \Sigma \to \Pi^*\).
Ex: \(h(0) = ab, h(1) = \epsilon\).

Define an extension of \(h : \Sigma^* \to \Pi^*\).
\[ h(\epsilon) = \epsilon \]
\[ h(xa) = h(x)h(a) \]

Define an extension of \(h\) that takes a language \(L\) over \(\Sigma\) and maps to a language \(h(L)\) over \(\Pi\):
\[ h(L) = \{h(w) \mid w \text{ is in } L\} \]

Theorem: If \(L\) is a regular language over \(\Sigma\), and \(h\) is a homomorphism on \(\Sigma\), then \(h(L)\) is also regular.

Proof: Prove by structural induction on the given regular expression \(E\) which satisfies \(L = L(E)\), to show that \(L(h(E)) = h(L(E))\) and \(h(E)\) is a regular expression. (\(h(E)\) is the expression we obtain by replacing each symbol \(a\) of \(\Sigma\) in \(E\) by \(h(a)\).)

Basis:

1. If \(E\) is \(\epsilon\) or \(\emptyset\), then \(h(E) = E\); hence \(h(E)\) is a regular expression, and \(L(h(E)) = L(E)\). We wish to show \(h(L(E)) = L(E)\), which is true since \(L(E)\) is either \(\{\epsilon\}\) or
2. If $E = a$ for some symbol $a$ in $\Sigma$. $h(E)$ is a regular expression: $h(a)$.

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

**INDUCTION:** Assume $E_1$, $E_2$ are sub-expressions of $E$, $h(E_1)$ and $h(E_2)$ are regular expressions, and $L(h(E_1)) = h(L(E_1))$ and $L(h(E_2)) = h(L(E_2))$.

1. If $E = E_1 \cup E_2$. Then $h(E) = h(E_1 \cup E_2) = h(E_1) \cup h(E_2)$. Hence, $h(E)$ is a regular expression.

$$L(h(E)) = L(h(E_1 \cup E_2)) = L(h(E_1) \cup h(E_2)) = L(h(E_1)) \cup L(h(E_2)) = h(L(E_1)) \cup h(L(E_2)) \text{ Hypothesis}$$

$$= h(L(E_1) \cup L(E_2)) \text{ Easy to prove}$$

$$= h(L(E_1 \cup E_2))$$

$$= h(L(E))$$

2. If $E = E_1E_2$. Then $h(E) = h(E_1E_2) = h(E_1)h(E_2)$. Hence, $h(E)$ is a regular expression.

$$L(h(E)) = L(h(E_1E_2)) = L(h(E_1)h(E_2)) = L(h(E_1)L(h(E_2))$$

$$= h(L(E_1))h(L(E_2)) \text{ Hypothesis}$$

$$= h(L(E_1)L(E_2)) \text{ Easy to prove}$$

$$= h(L(E_1E_2))$$

$$= h(L(E))$$
3. If \( E = E_1^* \). Then \( h(E) = h(E_1^*) = (h(E_1))^* \). Hence, \( h(E) \) is a regular expression.

\[
\begin{align*}
L(h(E)) &= L(h(E_1^*)) \\
&= L((h(E_1))^*) \\
&= (L(h(E_1)))^* \\
&= (h(L(E_1)))^* \text{ Hypothesis} \\
&= h((L(E_1))^*) \text{ Easy to prove} \\
&= h(L(E_1^*)) \\
&= h(L(E))
\end{align*}
\]

\[\square\]

Inverse Homomorphisms
\[
\begin{align*}
h &: \Sigma \to T^* \\
h &: \Sigma^* \to T^* \\
h &: 2^{\Sigma^*} \to 2^{T^*}
\end{align*}
\]

Define \( h^{-1}(L) = \{ w \mid w \in \Sigma^*, h(w) \in L \} \).

\[
\begin{align*}
\Sigma^* & \quad T^* \\
L & \quad h(L) \\
\Sigma^* & \quad T^*
\end{align*}
\]

\[
\begin{align*}
h^{-1}(L) & \quad L \\
h & \quad h
\end{align*}
\]

**Theorem:** If \( h \) is a homomorphism form \( \Sigma \) to \( T \), and \( L \) is a regular language over \( T \), then \( h^{-1}(L) \) is also a regular language.

**Proof:** Let DFA \( A = (Q, T, \delta, q_0, F) \) be such that \( L = L(A) \). Let DFA \( B = (Q, \Sigma, \gamma, q_0, F) \) be such that \( \gamma(q, a) = \hat{\delta}(q, h(a)) \)
for \( a \in \Sigma, q \in Q \).
We claim \( \hat{\gamma}(q_0, w) = \hat{\delta}(q_0, h(w)) \).
\( B \) accepts \( w \) iff \( A \) accepts \( h(w) \). \( \therefore L(B) = h^{-1}(L) \). \( \square \)

Decision Properties of Regular languages

Empty Language (Is \( L \) empty?)

If \( L \) is represented by any kind of finite automaton, then use graph-reachability algorithms to answer whether we can reach an accepting state from the start state.

Membership (Is \( w \) in \( L \)?)

If \( L \) is represented by a DFA, then simulate the DFA processing the input \( w \).

Equivalence of two regular languages (\( L_1 = L_2 \)?)

\[
L_1 = L_2 \iff (L_1 - L_2) \cup (L_2 - L_1) = \emptyset
\]
Hence, there are algorithms to make the decision.

Finite Language (Is \( L \) finite?)

Let DFA \( A \) be the automaton. Let \( n \) be the size of its state set. Simulate \( A \) on all strings \( w \) of length \( l \) such that \( n \leq l < 2n \). \( L \) is infinite if and only if \( A \) accepts any such string. Why? The “if” part can be easily proved by using Pumping Lemma. The “only if” part is harder.

Axiom: Every non-empty subset of \( \mathbb{N} \) has a smallest element.
Theorem: Let DFA \( A \) be the automaton for \( L \). Let \( n \) be the number of states. If \( L \) is infinite, then \( A \) accepts a string of length \( l \) such that \( n \leq l < 2n \).
**Proof:** B.W.O.C. assume $L$ is infinite and its DFA $A$ accepts no string of length $l$ such that $n \leq l < 2n$.

Among all strings in $L$ with length $\geq 2n$, choose the string $w$ that is as short as any other. From Pumping lemma, we can write $w = w_1w_2w_3$ with $1 \leq |w_2| \leq n$. And $w_1w_2^0w_3$ (which is $w_1w_3$) is in $L$. But $n \leq |w_1w_3| < 2n$ (why?), and this is a contradiction.

Explain why:

1. $n \leq |w_1w_3|$
   since $|w| \geq 2n$ and $|w_2| \leq n$.

2. $|w_1w_3| < 2n$
   Suppose not, then $|w_1w_3| \geq 2n$ and thus is one of the strings in $L$ with length $\geq 2n$. But $|w_1w_3| < |w|$, a contradiction to the assumption that $w$ is as short as any other.

\[ \square \]

**Myhill-Nerode Theorem and Minimization of DFA**
(See also Sipser2 Problem 1.51 and 1.52, pp. 90-91, p.97-98)

Given an arbitrary language $L \subseteq \Sigma^*$, define $R_L \subseteq \Sigma^* \times \Sigma^*$ such that

$$xR_Ly \text{ iff for each } z: \ xz \in L \iff yz \in L.$$ 

Then, $R_L$ is an equivalence relation. Why?

1. $R_L$ is reflexive.

2. $R_L$ is symmetric.

3. $R_L$ is transitive.
   Suppose $xR_Ly$ and $yR_Lz$. We show that $xR_Lz$. Let $w$ be any string in $\Sigma^*$. Then, if $xw \in L$, $yw$ must be in $L$. Hence, $zw$ must also be in $L$. Similarly, if $xw \notin L$, $yw$ cannot be in $L$. Hence, $zw$ cannot be in $L$. 

19
If the number of equivalence classes is finite, we say the equivalence relation has a *finite index*.

*Right invariant:* An equivalence relation $R$ is said to be right invariant if $x, y, z \in \Sigma^*$ and $xRy$ implies $xzRyz$.

**Myhill-Nerode theorem:** The following three statements about the language $L$ are equivalent:

1. $L$ is recognized by some DFA.
2. $L$ is the union of some of the equivalence classes of a right invariant equivalence relation that has a finite index.
3. $R_L$ has a finite index.

**Proof:** (1) $\Rightarrow$ (2)
Assume $L$ is recognized by DFA $M = (Q, \Sigma, \delta, q_0, F)$. Let $R_M$ be the equivalence relation such that $xR_My$ iff $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$. For any $z$, if $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$ then $\hat{\delta}(q_0, xz) = \hat{\delta}(q_0, yz)$. Hence, $R_M$ is right invariant; and $R_M$ has a finite index since $Q$ is finite. Furthermore, $L$ is the union of those equivalence classes that contain a string $x$ such that $\hat{\delta}(q_0, x)$ is in $F$.

(2) $\Rightarrow$ (3)
Suppose $E$ is an equivalence relation satisfying (2). We show that every equivalence class of $E$ is entirely contained in some equivalence class of $R_L$. Thus the index of $R_L$ cannot be greater than the index of $E$ and so is finite.

Assume $xEy$. Then for each $z \in \Sigma^*$, $xzEy z$. Thus, for each $z \in \Sigma^*$, $xz \in L \iff yz \in L$. Thus, $xR_Ly$. Therefore, $E \subseteq R_L$.

(3) $\Rightarrow$ (1)
**Claim:** $R_L$ is right invariant.

**Proof omitted.**

Let $Q'$ be the finite set of equivalence classes of $R_L$ and $[x]$ be
the element of $Q'$ containing $x$.
Define $\delta'(\lfloor x \rfloor, a) = [xa]$. We claim that $\delta'$ is well-defined. Let $q'_0 = [\epsilon], F' = \{ \lfloor x \rfloor \mid x \in L \}$. Then $M' = (Q', \Sigma, \delta', q'_0, F')$ recognizes $L$ since $\hat{\delta}'(q'_0, x) = \lfloor x \rfloor$, and thus $x$ is in $L(M')$ iff $\lfloor x \rfloor$ is in $F'$. But $\lfloor x \rfloor \in F'$ iff $x \in L$.

**Claim:** $\delta'(\lfloor x \rfloor, a) = [xa]$ is well-defined.

**Proof:** Suppose $x \neq y$ and $[xa] \neq [ya]$, we must show that $\lfloor x \rfloor \neq [y]$, i.e. $xR_Ly$ cannot be true.

B.W.O.C. assume $\lfloor x \rfloor = [y]$, then $xR_Ly$ holds. Since $R_L$ is right invariant, $xaR_Lya$ holds. Then, $[xa] = [ya]$, a contradiction. \(\square\)

Important observations can be made from the proof of the Myhill-Nerode theorem:

1. Given a regular language $L$, the DFA $M'$ in the proof ``(3) $\Rightarrow$ (1)'' has minimum state among all possible DFA’s that recognize $L$. Why?
   The number of states in $M'$ is the number of equivalence classes of $R_L$, and $R_L$ cannot have more equivalence classes than any other $R_M$ which is defined in the proof ``(1) $\Rightarrow$ (2)'' for an arbitrary DFA $M$ that recognizes $L$. Why so? Since any such $R_M$ must be a right invariant equivalence relation that has a finite index, and therefore every equivalence class of $R_M$ is entirely contained in some equivalence class of $R_L$.
   The proof ``(2) $\Rightarrow$ (3)'' has all the details.

2. Suppose a DFA $A_m$ recognizes $L$ and it has minimum state set among all DFAs that recognize $L$. Then $A_m$ and $M'$ are isomorphic. We show a one-to-one correspondence between the two state sets. Let $M' = (Q', \Sigma, \delta', q'_0, F')$ and $A_m = (Q, \Sigma, \delta, q_0, F)$. Define $f: Q \to Q'$ as follows.
   For each $q \in Q$, let $f(q) = \hat{\delta}(q_0, x)$ where $x$ is some string in $\Sigma^*$ such that $\hat{\delta}(q_0, x) = q$. Part(a) will explain why such string $x$ must exist.
(a) Given any \( q \), there exists some string \( x \in \Sigma^* \) such that \( \delta(q_0, x) = q \). Suppose otherwise, \( q \) is not reachable from \( q_0 \) and thus can be removed from \( Q \), a contradiction to the assumption that \( A_m \) has minimum state set.

(b) Claim: \( f \) is a function.

Proof: Given any \( q \), suppose \( x \) and \( y \) are two distinct strings such that \( \delta(q_0, x) = q \) and \( \delta(q_0, y) = q \).

\[ \therefore x R_{Am} y. \]

We must show that \( \delta'(q_0', x) = \delta'(q_0', y) \).

Recall the proof “(2) \implies (3)”; since the equivalence relation \( R_{Am} \) as defined by DFA \( A_m \) is a right invariant equivalence relation with finite index, we have \( x R_L y \). But,

\[
\delta'(q_0', x) = [x] = [y] \therefore x R_L y = \delta'(q_0', y)
\]

\[ \square \]

(c) Claim: \( f \) is one-to-one.

Proof: Assume \( f(q_1) = f(q_2) \), we must show that \( q_1 = q_2 \).

Recall \( f(q_1) = \hat{\delta}'(q'_0, x_1) \) where \( x_1 \) is some string in \( \Sigma^* \) such that \( \hat{\delta}(q_0, x_1) = q_1 \), and \( f(q_2) = \hat{\delta}'(q'_0, x_2) \) where \( x_2 \) is some string in \( \Sigma^* \) such that \( \hat{\delta}(q_0, x_2) = q_2 \).

From \( \hat{\delta}'(q'_0, x_1) = f(q_1) = f(q_2) = \hat{\delta}'(q'_0, x_2) \), we have \( x_1 R_L x_2 \). We have \( x_1 R_{Am} x_2 \) since otherwise \( R_{Am} \) will have more equivalence classes than \( R_L \) and hence \( Q \) has more states than \( Q' \).

Then we have \( \hat{\delta}(q_0, x_1) = \hat{\delta}(q_0, x_2) \), and thus \( q_1 = q_2 \). \( \square \)

Minimization of DFA

Equivalent states:
State \( p \) and state \( q \) are said to be equivalent if for all \( w \in \Sigma^* \),
\[ \delta(p, w) \in F \iff \delta(q, w) \in F; \text{ and distinguishable otherwise.} \]

**Table-filling algorithm:**

Given DFA \( A = (Q, \Sigma, \delta, q_0, F) \) the following algorithm finds all distinguishable pairs in \( A \).

**BASIS:** If \( p \) is in \( F \) and \( q \) is not, then \( \{p, q\} \) is distinguishable.

**INDUCTION:** If \( \{r, s\} \) is distinguishable, and \( r = \delta(p, a) \) and \( s = \delta(q, a) \) for some \( a \in \Sigma \), then \( \{p, q\} \) is distinguishable.

Why? Let \( w \) be the string that distinguishes \( \{r, s\} \).

\[
\begin{align*}
\hat{\delta}(p, aw) &= \hat{\delta}(r, w) \\
\hat{\delta}(q, aw) &= \hat{\delta}(s, w)
\end{align*}
\]

Thus, \( aw \) distinguishes \( \{p, q\} \).

The algorithm can be used to

1. decide whether two regular languages are equal,

2. obtain the unique minimum state DFA from any given DFA.

**Correctness of table-filling algorithm**

**Theorem:** If two states are not distinguished by the table-filling algorithm, then they are equivalent.

**Proof:** Let \( A \) be the DFA for which the algorithm will find all pairs of distinguishable states, and \( A = (Q, \Sigma, \delta, q_0, F) \).

B.W.O.C. assume there is at least one pair of states \( \{p', q'\} \) such that

1. there exists \( w \) such that exactly one of \( \hat{\delta}(p', w) \) and \( \hat{\delta}(q', w) \) is in \( F \), and yet

2. the algorithm does not find \( \{p', q'\} \) to be distinguished.

**Definition:** Let us call such pairs BAD pairs.

Among all strings that distinguish the bad pairs, choose one
string $w = a_1 a_2 \cdots a_n$ such that $w$ is as short as any other. Let \{\(p, q\)\} be a bad pair that $w$ distinguishes. Then exactly one of $\hat{\delta}(p, w)$ and $\hat{\delta}(q, w)$ is in $F$.

**Claim:** $|w| \geq 1$, i.e. $w$ cannot be $\epsilon$.

Consider state $r = \delta(p, a_1)$, state $s = \delta(q, a_1)$.
\{\(r, s\)\} is distinguished by $a_2 a_3 \cdots a_n$ since \{\(p, q\)\} is distinguished by $a_1 a_2 \cdots a_n$.
\{\(r, s\)\} cannot be a bad pair since $w$ is longer than $a_2 a_3 \cdots a_n$. Hence, the algorithm does find \{\(r, s\)\} to be distinguished. But the algorithm will go further and find \{\(p, q\)\} to be distinguished since $r = \delta(p, a_1)$ and $s = \delta(q, a_1)$. This is a contradiction! $\square$