Lecture L&M 1

Languages, Regular Expressions, Deterministic Finite Automata

1.1 Induction over words

A. Using induction on i, prove that $(w^R)^i = (w^i)^R$ for any word w and all $i \geq 0$, where ^{R} is the reversal operator, defined recursively by $\lambda^R = \lambda$ and $(wa)^R = aw^R$ for $a \in \Sigma$, $w \in \Sigma^+$.

Hint: Use that $(uv)^R = v^R u^R$ for $u, v \in \Sigma^*$ (Theorem 2.1.6 in the book).

We prove by induction on i that $(w^R)^i = (w^i)^R$.
Base case: If $i = 0$ then $(w^R)^i = \lambda = \lambda^R = (w^i)^R$.
Induction step: Assume that $(w^R)^k = (w^k)^R$ applies for a given k (IH). Now

$$(w^R)^{k+1} = (w^R)^k w^R \overset{\text{(IH)}}{=} (w^k)^R w^R = (ww^k)^R = (w^{k+1})^R$$

where the second-to-last step is justified by Theorem 2.1.6.

B. Use induction on the length of w to prove that $(w^R)^R = w$ for all words $w \in \Sigma^*$.

Hint: Remember that every word w with $|w| \geq 1$ can be considered as a letter $a \in \Sigma$ followed by a word $v \in \Sigma^*$ with $|v| = |w| - 1$. You may also need Theorem 2.1.6 again.

We prove, by induction on $|w|$, that $w = (w^R)^R$ for every word $w \in \Sigma^*$. As part of the proof, we need to establish $a^R = a$ for $a \in \Sigma$; this follows from $a^R = (\lambda a)^R = a\lambda^R = a\lambda = a$.
Base case: $w = \lambda$ is the only word of length 0, and we have $(\lambda^R)^R = (\lambda)^R = \lambda$ as desired.
Induction step: Assume that $(w^R)^R = w$ for all words $w \in \Sigma^*$ of length n or less. Let w be a word of length $n + 1$. Then $w = ua$ with $a \in \Sigma$ and $u \in \Sigma^*$, and we have

$$\begin{aligned}
(w^R)^R &= ((ua)^R)^R \\
&= (a^R u^R)^R \qquad \text{(Theorem 2.1.6)} \\
&= (au^R)^R \\
&= (u^R)^R a^R \qquad \text{(Theorem 2.1.6)} \\
&= ua^R \qquad \text{(induction hypothesis)} \\
&= ua \\
&= w
\end{aligned}$$

C. *(extra)* Given an alphabet Σ consisting of k elements, let us define $f(n) := \#\{w \mid |w| \leq n\}$, i.e. $f(n)$ is the number of words over Σ with length smaller than or equal to n. Give a (non-recursive summation) formula for $f(n)$. Prove its correctness with (natural) induction.

The intended formula is $f(n) = \sum_{i=0}^{n} k^i$.
Proof by induction on n.
Base case $n = 0$:

$$f(0) = \#\{w \mid |w| \leq 0\} = \#\{\lambda\} = 1 = k^0 = \sum_{i=0}^{0} k^i$$

Induction step: Suppose (IH) $f(n) = \sum_{i=0}^{n} k^i$. Let w be some word with $|w| \leq n+1$. Then $w = \lambda$, or $w = av$, with $a \in \Sigma$ and $|v| \leq n$. Of these, there are then:

$$f(n+1) = 1 + k \cdot f(n) \overset{\text{(IH)}}{=} 1 + k \cdot \sum_{i=0}^{n} k^i = 1 + \sum_{i=0}^{n} k^{i+1} = k^0 + \sum_{i=1}^{n+1} k^i = \sum_{i=0}^{n+1} k^i$$

It follows that for all n, indeed $f(n) = \sum_{i=0}^{n} k^i$.

1.2 Regular expressions

For each of the following languages L, give a regular expression E such that $\mathcal{L}(E) = L$:

D. L is the language of all words over $\{a, b, c\}$ in which all the occurrences of a precede all the occurrences of b and c, and in turn all the b precede all the c.

An easy start: The regular expression is $a^*b^*c^*$.

E. L is the same as in Exercise D, except it does not contain λ.

Since λ is not allowed in the language, each word must contain at least one a or one b or one c. However, it need not contain one of every symbol. The +'s in the regular expression $a^+b^*c^* \cup a^*b^+c^* \cup a^*b^*c^+$ ensure the presence of at least one symbol in each word in the language. An equivalent, but shorter, expression is $a^+b^*c^* \cup b^+c^* \cup c^+$.

F. L is the language of all words over $\{a, b, c\}$ of length three.

For every symbol a choice: $(a \cup b \cup c)(a \cup b \cup c)(a \cup b \cup c)$. Can be shortened to $(a \cup b \cup c)^3$.

G. L is the language of all words in $\{a, b\}^*$ that contain the substring ab and have length > 2.

The extra symbols can be before or after the ab: $(a \cup b)^*ab(a \cup b)^+ \cup (a \cup b)^+ab$

H. L is the language over $\{a, b\}$ such that all words contain both aa and bb.

The set of word over $\{a, b\}$ that contain the substrings aa and bb is represented by

$$(a \cup b)^*aa(a \cup b)^*bb(a \cup b)^* \cup (a \cup b)^*bb(a \cup b)^*aa(a \cup b)^* \quad .$$

The two expressions joined by the $\cup$ indicate that the aa may precede or follow the bb.

I. *(extra)* L is the language over $\{a, b\}$ such that all words contain both ab and ba.

At first glance it may seem that the desired language is given by the regular expression

$$(a \cup b)^* ab (a \cup b)^* ba (a \cup b)^* \cup (a \cup b)^* ba (a \cup b)^* ab (a \cup b)^* \ .$$

Clearly every word in this language has substrings ab and ba. Every word described by the preceding expression has length four or more. Have we missed some words with the desired property? Consider aba, bab, and $aaaba$. A regular expression for the set of words containing substrings ab and ba can be obtained by adding alternative ($\cup$)

$$(a \cup b)^* aba (a \cup b)^* \cup (a \cup b)^* bab (a \cup b)^*$$

to the expression above.

1.3 Deterministic finite automata

J. Let M be the deterministic finite automaton (DFA) defined by $Q = \{q_0, q_1, q_2\}$ (with q_0 being the initial state), $\Sigma = \{a, b\}$, $F = \{q_2\}$ and transition function δ according to the following transition table:

δ	a	b
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_2	q_0

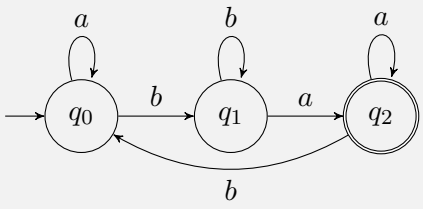
(a) Draw the state diagram for M.

(b) Trace the computations of M (i.e. the inference steps according to relation $\vdash$) when processing the words $abaa$, $bbbabb$, $bababa$ and $bbbaa$.

(c) Which of the above four words are accepted by M?

(d) Give a regular expression for $\mathcal{L}(M)$.

(a) The state diagram of M is:



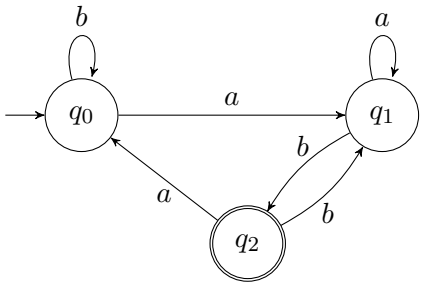
(b)

i)	ii)	iii)	iv)
$[q0, abaa]$	$[q0, bbbabb]$	$[q0, bababa]$	$[q0, bbbaa]$
$\vdash [q0, baa]$	$\vdash [q1, bbabb]$	$\vdash [q1, ababa]$	$\vdash [q1, bbaa]$
$\vdash [q1, aa]$	$\vdash [q1, babb]$	$\vdash [q2, baba]$	$\vdash [q1, baa]$
$\vdash [q2, a]$	$\vdash [q1, abb]$	$\vdash [q0, aba]$	$\vdash [q1, aa]$
$\vdash [q2, \lambda]$	$\vdash [q2, bb]$	$\vdash [q0, ba]$	$\vdash [q2, a]$
	$\vdash [q0, b]$	$\vdash [q1, a]$	$\vdash [q2, \lambda]$
	$\vdash [q1, \lambda]$	$\vdash [q2, \lambda]$	

(c) The computations in (i), (iii), and (iv) terminate in the accepting state q_2. Therefore, the words $abaa$, $bababa$, and $bbbaa$ are in $\mathcal{L}(M)$.

(d) Two possible regular expressions for $\mathcal{L}(M)$ are $a^*b^+a^+(ba^*b^+a^+)^*$ and $(a^*b^+a^+b)^*a^*b^+a^+$.

K. Let M be the DFA defined by the following state diagram:



(a) Construct the transition table of M.

(b) Which of the words $baba$, $baab$, $abab$ and $abaaab$ are accepted by M?

(c) Give a regular expression for $\mathcal{L}(M)$.

(a) The definition of M is:

$$\begin{aligned} Q &= \{q_0, q_1, q_2\} \\ \Sigma &= \{a, b\} \\ F &= \{q_2\} \end{aligned}$$

δ	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_0	q_1

(b) Only $baab$ and $abaaab$ are accepted.

(c) For example: $b^*a^+b(ab^*a^+b \cup ba^*b)^*$

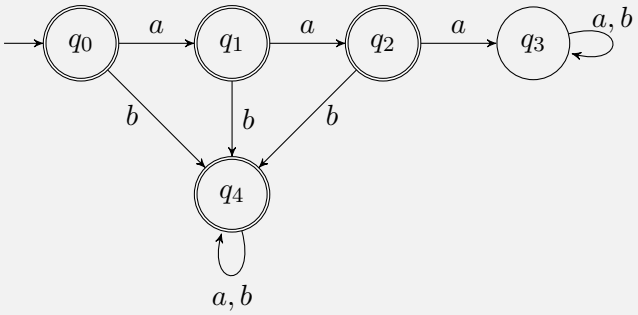
L. For each of the following languages L, draw the state diagram of a DFA that accepts the language. Can you also come up with a regular expression describing the language?

(a) L is the language of all words over $\{a, b\}$ that do not begin with aaa.

(b) L is the language of all words over $\{a, b\}$ that do not contain the substring aaa.

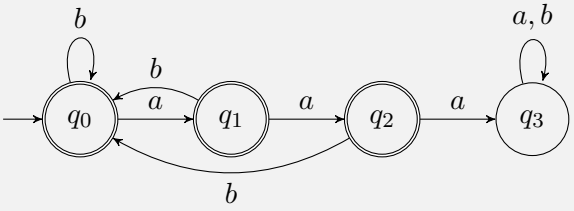
(c) *(extra)* L consists of all words of even length over $\{a, b, c\}$ that contain exactly one a.

(a) The complete DFA



accepts the set of word over $\{a, b\}$ that do not begin with the substring aaa. Its regular expression is $(\lambda \cup a \cup aa)(\lambda \cup b(a \cup b)^*)$.

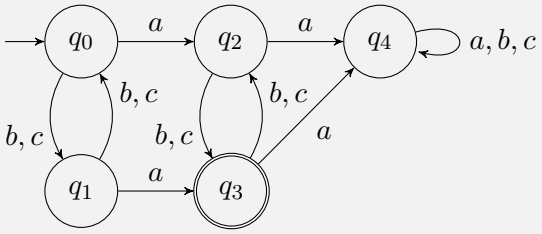
(b) A complete DFA accepting exactly the words over $\{a, b\}$ that do not contain the substring aaa is given by



The states are used to count the number of consecutive a that have been processed. When three consecutive a are encountered, the DFA enters state q_3, processes the remainder of the input, and rejects the word.

A possible regular expression is: $(\lambda \cup a \cup aa)(b^+(\lambda \cup a \cup aa))^*$

5.15 The complete DFA



accepts words of even length over $\{a, b, c\}$ that contain exactly one a. A word accepted by this machine must have exactly one a and the total number of b and c must be odd. A computation that processes an odd number of b and c terminates in state q_1 or q_3. States q_2 or q_3 are entered upon processing a single a. The state q_3 represents the combination of the two conditions required for acceptance. Upon reading a second a, the computation enters the non-accepting state q_4 and rejects.

A possible regular expression is: $((b \cup c)(b \cup c))^*(a(b \cup c) \cup (b \cup c)a)((b \cup c)(b \cup c))^*$.