Chapter 1

Maciej Harbuz

June 13, 2024

1 Exercises

1.1 The following are the state diagrams of two DFAs, M_1 and M_2. Answer the following questions about each of these machines.

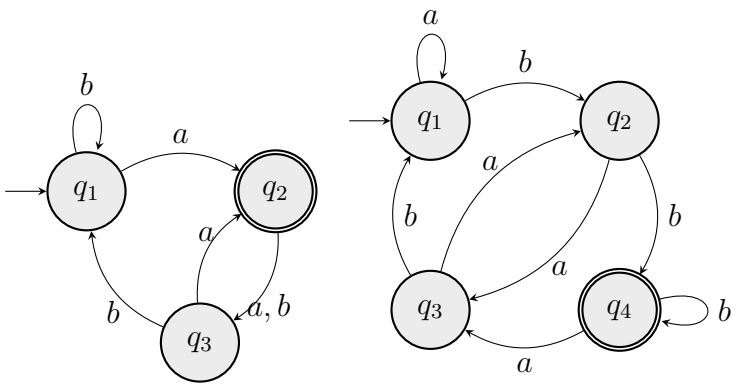


Figure 1: M_1 and M_2

(a) What is the start state?

$M_1 : q_1$

$M_2 : q_1$

(b) What is the set of accept states?

$M_1 : \{q_2\}$

$M_2 : \{q_4\}$

(c) What sequence of states does the machine go through on input $aabb$?

$M_1 : q_1 \to q_2 \to q_3 \to q_1 \to q_1$

$M_2 : q_1 \to q_2 \to q_4 \to q_4 \to q_4$

(d) Does the machine accept the string $aabb$?

M_1 : No

M_2 : Yes

(e) Does the machine accept the string ϵ?

M_1 : No

M_2 : No

1.2 Give the formal description of the machines M_1 and M_2 pictured in Exercise 1.1.

$M_1 = (Q, \Sigma_1, \delta_1, q_1, \{q_2\})$

$Q = \{q_1, q_2, q_3\}$

$\Sigma_1 = \{a, b\}$

$\delta_1 = \{((q_1, a), q_2), ((q_1, b), q_1), ((q_2, a), q_3), ((q_2, b), q_3), ((q_3, a), q_2), ((q_3, b), q_1)\}$

$M_2 = (Q, \Sigma_2, \delta_2, q_1, \{q_4\})$

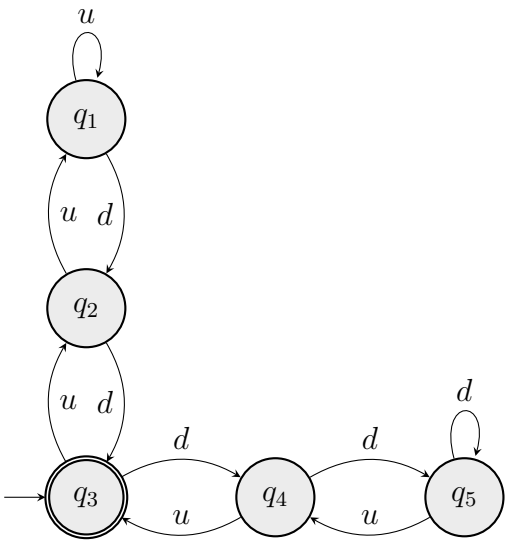
$Q = \{q_1, q_2, q_3, q_4\}$

$\Sigma_2 = \{a, b\}$

$\delta_2 = \{((q_1, a), q_1), ((q_1, b), q_2), ((q_2, a), q_3), ((q_2, b), q_4),$
$((q_3, a), q_2), ((q_3, b), q_1), ((q_4, a), q_3), ((q_4, b), q_4)\}$

1.3 The formal description of a DFA M is $\{q_1, q_2, q_3, q_4, q_5\}, \{u, d\}, \sigma, q_3, \{q_3\}$, where σ is given by the following table. Give the state diagram of this machine.

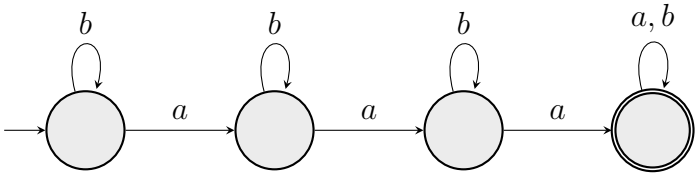
	u	d
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_2	q_4
q_4	q_3	q_5
q_5	q_4	q_5



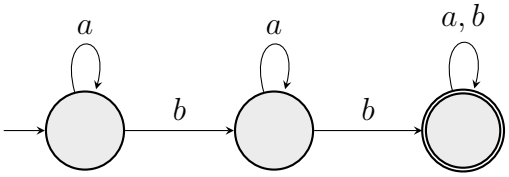
1.4 Each of the following languages is the intersection of two simpler languages. In each part, construct DFAs for the simpler languages,then combine them using the construction discussed in footnote 3 (page 46)

to give the state diagram of a DFA for the language given. In all parts, $\Sigma = \{a, b\}$.

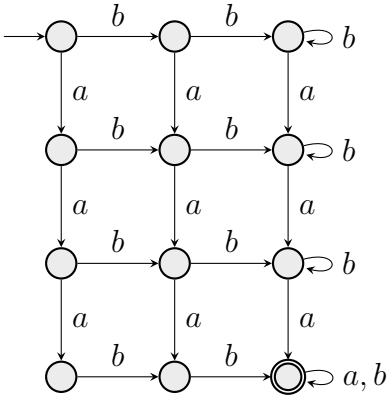
(a) $\{w|w$ has at least three a's and at least two b's$\}$
DFA_1 for at least three a's:



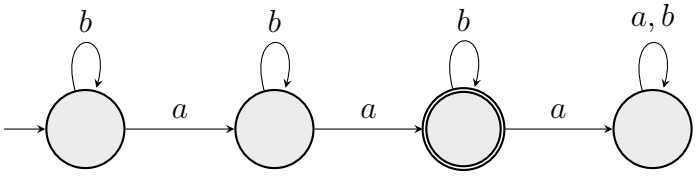
DFA_2 for at least two b's:



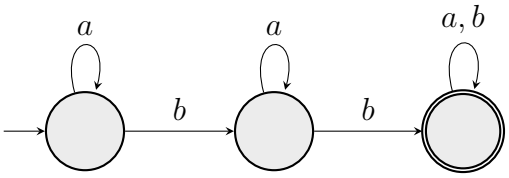
DFA for at least three a's and at least two b's:



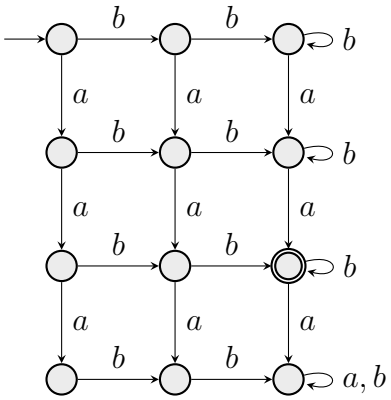
(b) $\{w|w$ has exactly two a's and at least two b's$\}$
DFA_1 for exactly two a's:



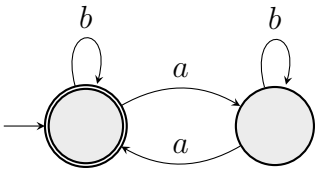
DFA_2 for at least two b's:



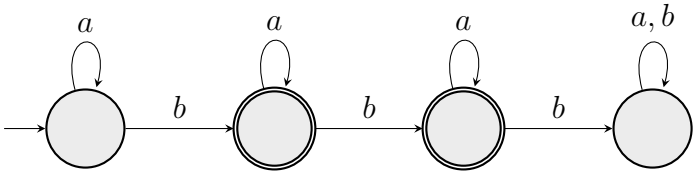
DFA for exactly two a's and at least two b's:



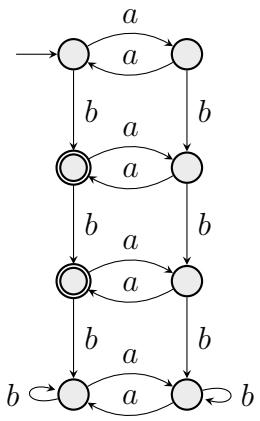
(c) $\{w|w$ has an even number of a's and one or two b's$\}$
DFA_1 for an even number of a's:



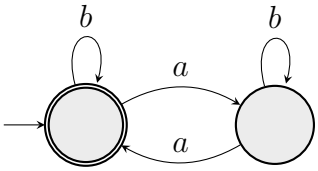
DFA_2 for one or two b's:



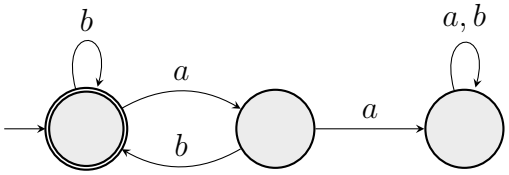
DFA for an even number of a's and one or two b's:



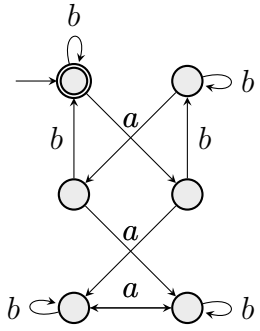
(d) $\{w|w$ has an even number of a's and each a is followed by at least one $b\}$
DFA_1 for an even number of a's:



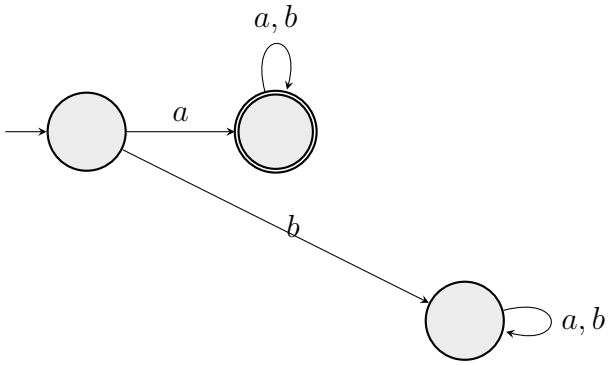
DFA_2 for each a is followed by at least one b:



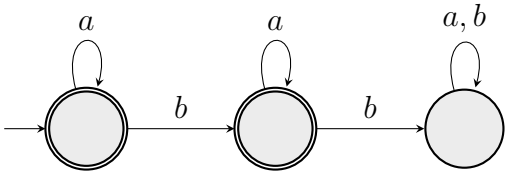
DFA for an even number of a's and each a is followed by at least one b:



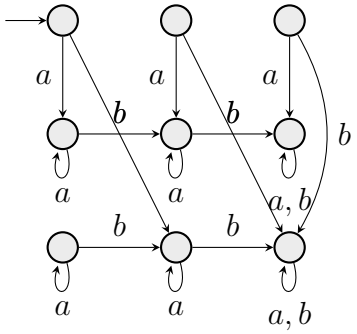
(e) $\{w|w$ starts with an a and has at most one $b\}$
DFA_1 for starts with an a:



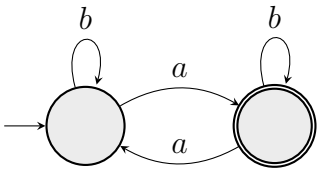
DFA_2 for has at most one b:



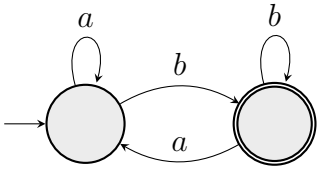
DFA for starts with an a and has at most one b:



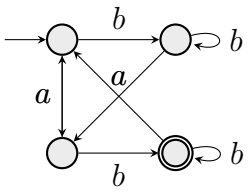
(f) $\{w|w$ has an odd number of a's and ends with a $b\}$
DFA_1 for an odd number of a's:



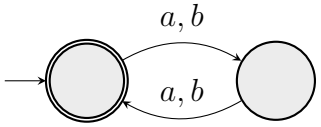
DFA_2 for ends with a b:



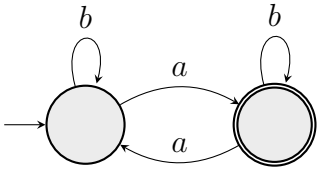
DFA for an odd number of a's and ends with a b:



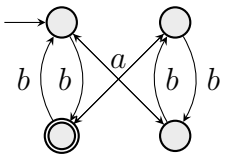
(g) $\{w|w$ has even length and an odd number of a's$\}$
DFA_1 for even length:



DFA_2 for an odd number of a's:



DFA for even length and an odd number of a's:



1.5 Each of the following languages is the complement of a simpler language. In each part, construct a DFA for the simpler language, then use it to give the state diagram of a DFA for the language given. In all parts, $\Sigma = \{a, b\}$.

(a) $A = \{w|w$ does not contain the substring $ab\}$
$\overline{A} = \{w|w$ contains the substring $ab\}$

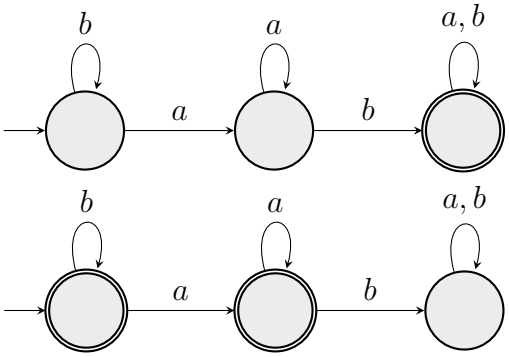


Figure 2: $\overline{A}$ and A

(b) $B = \{w|w$ does notcontain the substring $baba\}$
$\overline{B} = \{w|w$ contains the substring $baba\}$

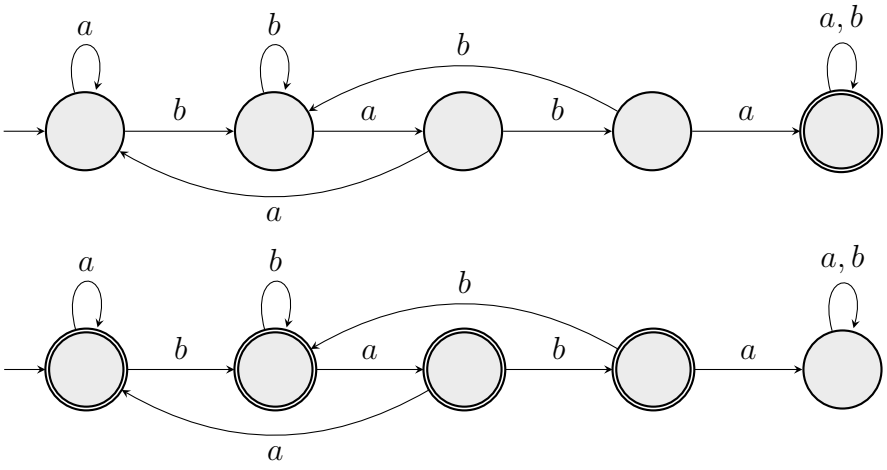


Figure 3: $\overline{B}$ and B

(c) $C = \{w|w$ contains neither the substrings ab nor $ba\}$
$\overline{C} = \{w|w$ contains the substring ab or $ba\}$

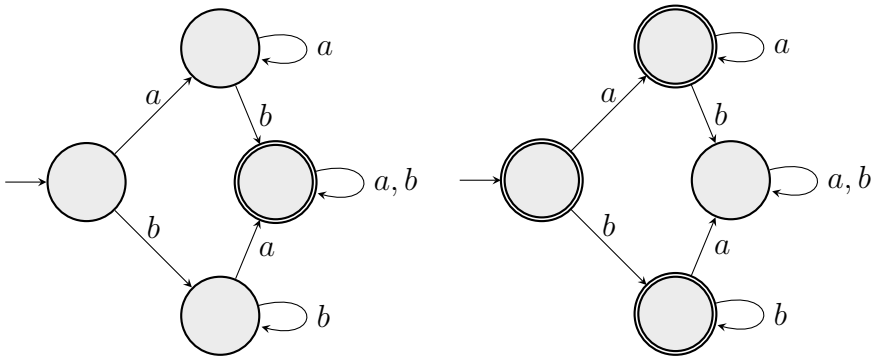


Figure 4: $\overline{C}$ and C

(d) $D = \{w | w \text{ is any string not in } a^*b^*\}$
$\overline{D} = \{w | w \text{ is any string in } a^*b^*\}$

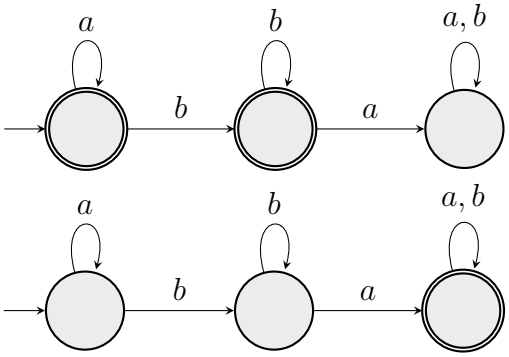


Figure 5: $\overline{D}$ and D

(e) $E = \{w | w \text{ is any string not in } (ab^+)^*\}$
$\overline{E} = \{w | w \text{ is any string in } (ab^+)^*\}$

- assuming ab^+ means a followed by one or more b.

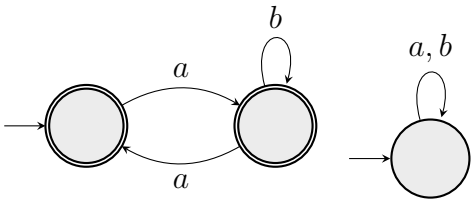


Figure 6: $\overline{E}$ and E

- assuming ab^+ means ab one or more times.

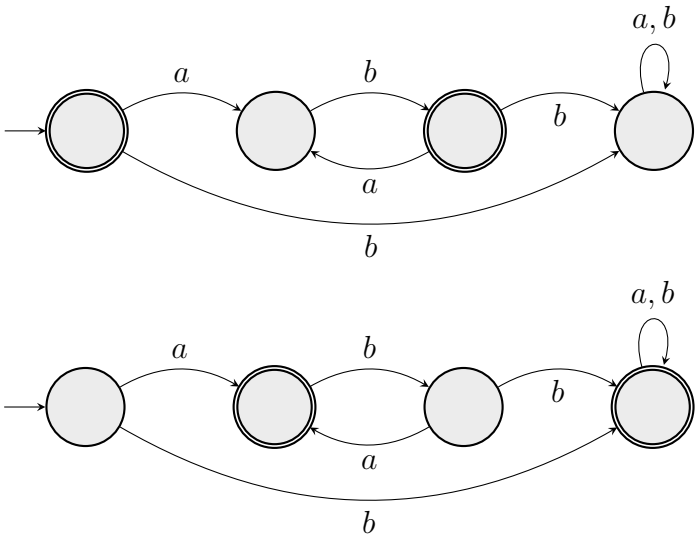


Figure 7: $\overline{E}$ and E

(f) $F = \{w | w \text{ is any string not in } a^* \cup b^*\}$
$\overline{F} = \{w | w \text{ is any string in } a^* \cup b^*\}$

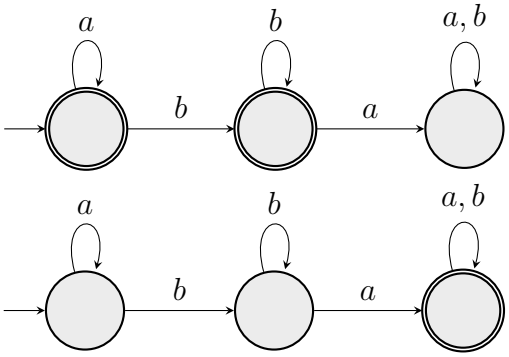


Figure 8: $\overline{F}$ and F

(g) $G = \{w | w \text{ is any string that doesn't contain exactly two } a\text{'s}\}$
$\overline{G} = \{w | w \text{ is any string that contains exactly two } a\text{'s}\}$

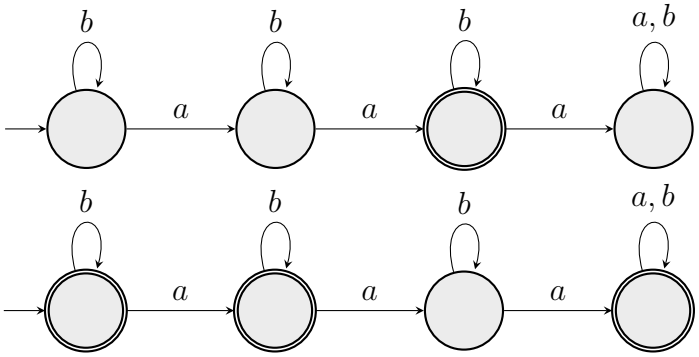


Figure 9: $\overline{G}$ and G

(h) $H = \{w | w$ is any string except a and $b\}$
$\overline{H} = \{w | w$ is string a or $b\}$

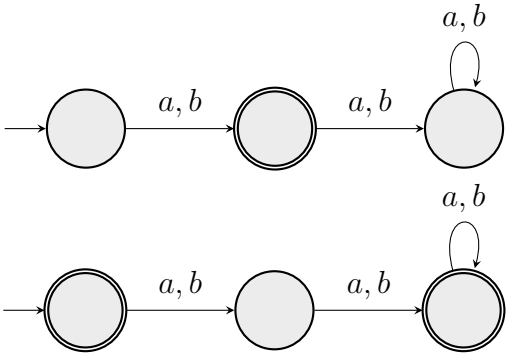
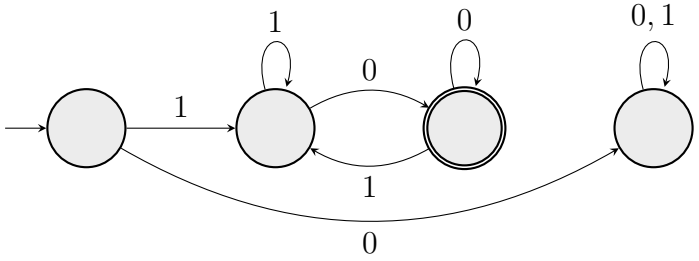


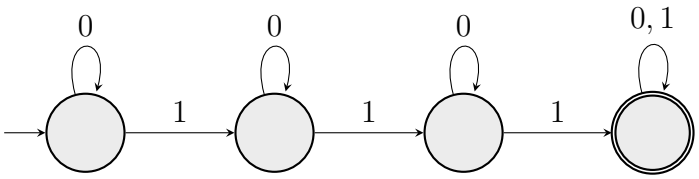
Figure 10: $\overline{H}$ and H

1.6 Give state diagrams of DFAs recognizing the following languages. In all parts,the alphabet is $\{0, 1\}$.

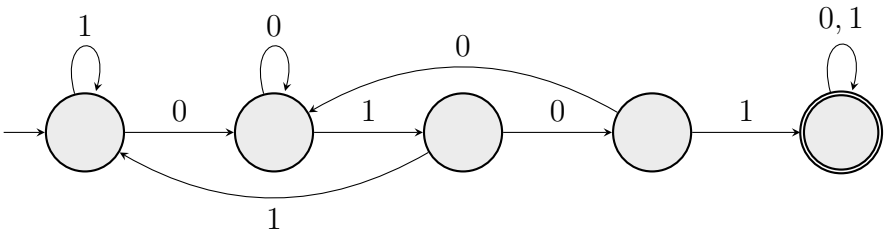
(a) $\{w | w$ begins with a 1 and ends with a $0\}$



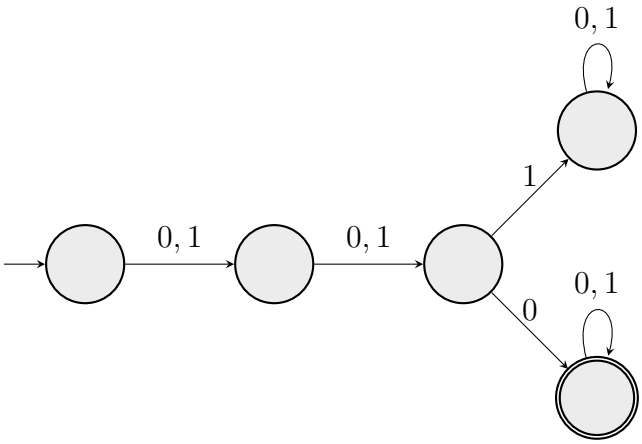
(b) $\{w|w$ contains at least three 1's$\}$



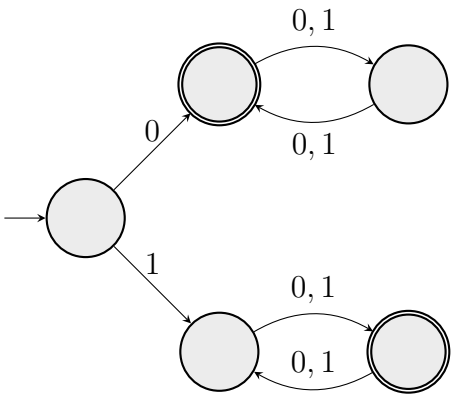
(c) $\{w|w$ contains the substring 0101 (i.e., $w = x0101y$ for some x and $y)\}$



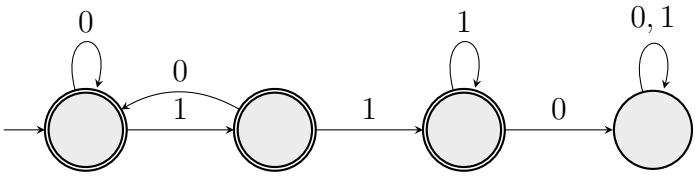
(d) $\{w|w$ has length at least 3 and its third symbol is a 0$\}$



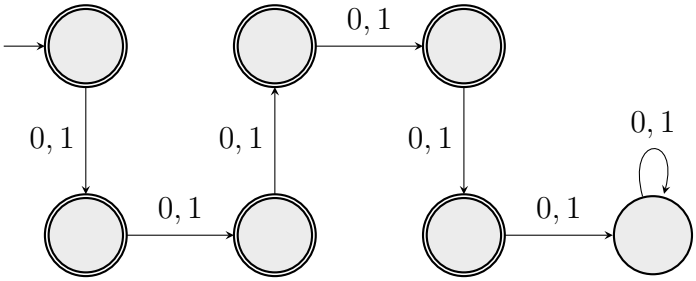
(e) $\{w|w$ starts with 0 and has odd length, or starts with 1 and has even length$\}$



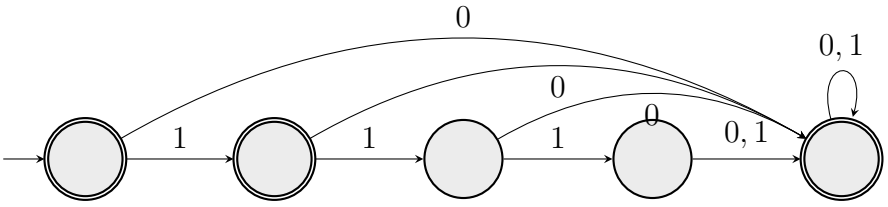
(f) $\{w|w$ doesn't contain the substring 110$\}$



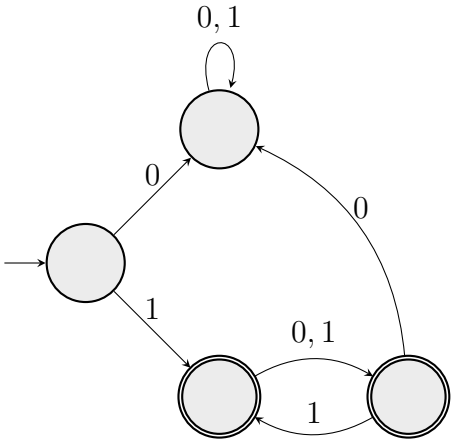
(g) $\{w|$the length of w is at most 5$\}$



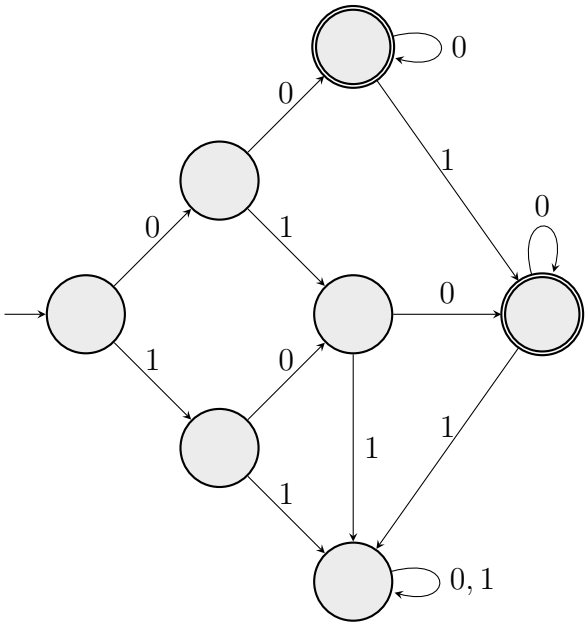
(h) $\{w|w$ is any string except 11 and 111$\}$



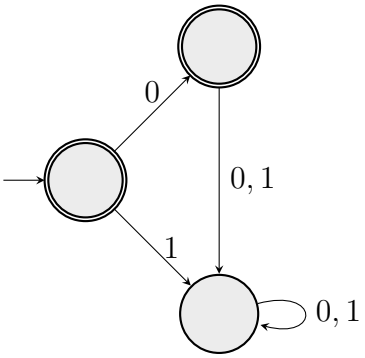
(i) $\{w|$ every odd position of w is a 1$\}$



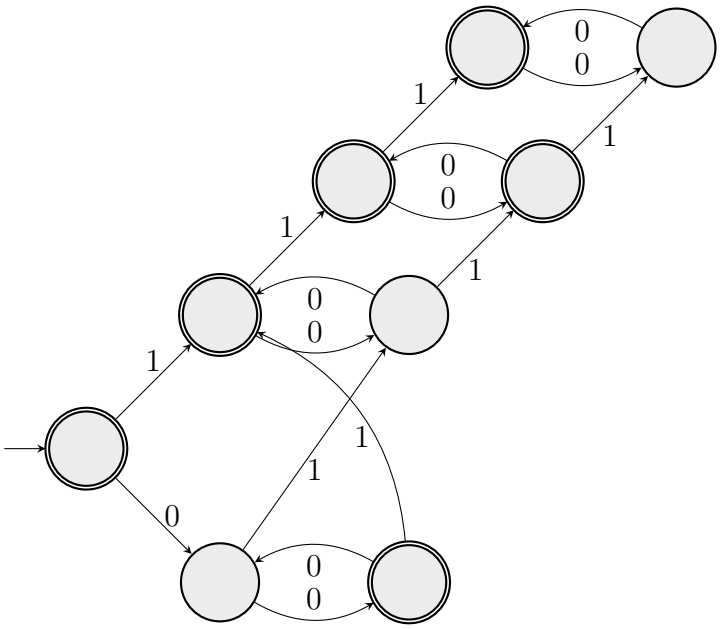
(j) $\{w|w$ contains at least two 0's and at most one 1$\}$



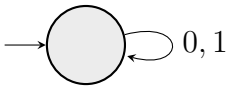
(k) $\{\epsilon, 0\}$



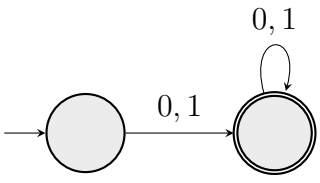
(l) $\{w|w$ contains an even number of 0's, or contains exactly two 1's$\}$



(m) The empty set

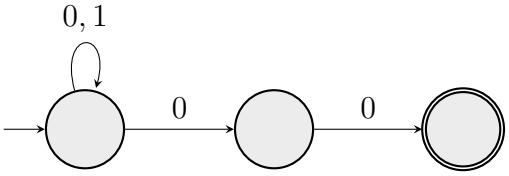


(n) All strings except the empty string

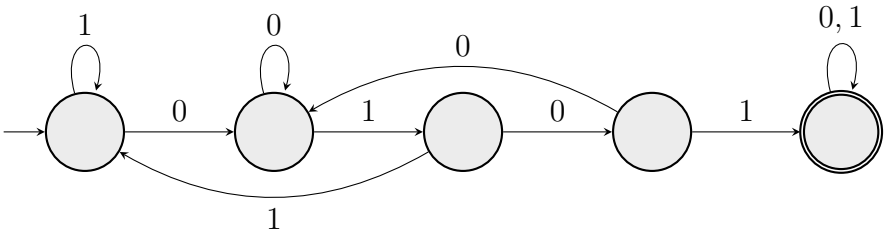


1.7 Give state diagrams of NFAs with the specified number of states recognizing each of the following languages. In all parts, the alphabet is 0,1.

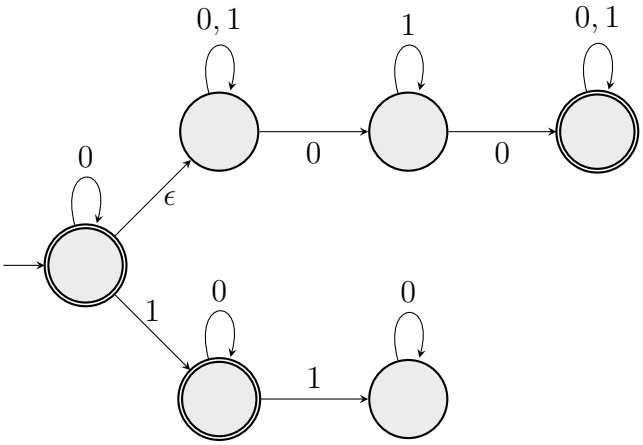
(a) The language $\{w|w$ ends with $00\}$ with three states



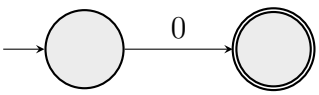
(b) The language of Exercise 1.6c with five states



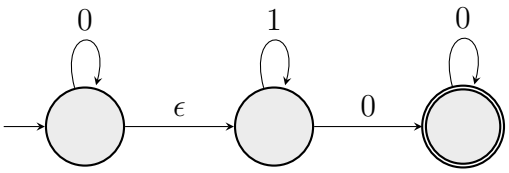
(c) The language of Exercise 1.6l with six states



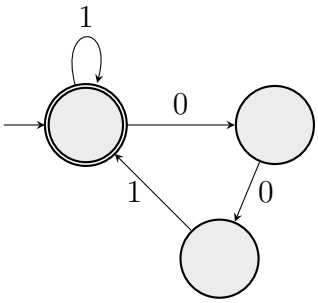
(d) The language $\{0\}$ with two states



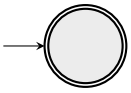
(e) The language 0^*1^*0+ with three states



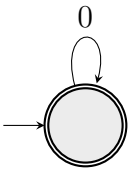
(f) The language $1^*(001+)^*$ with three states



(g) The language ϵ with one state

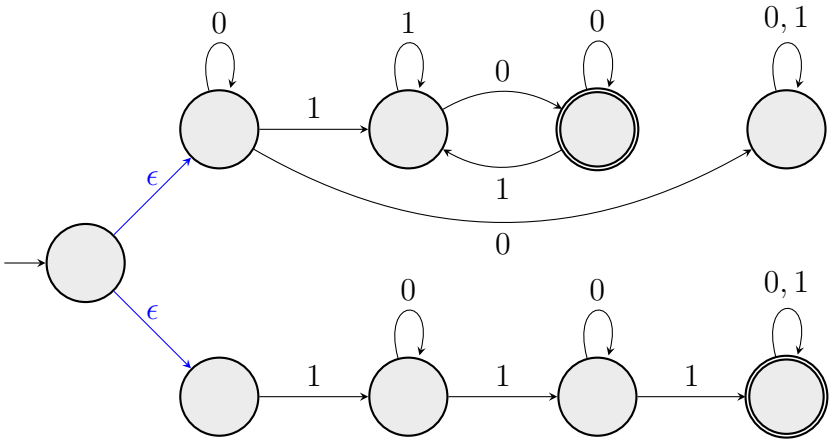


(h) The language 0^* with one state

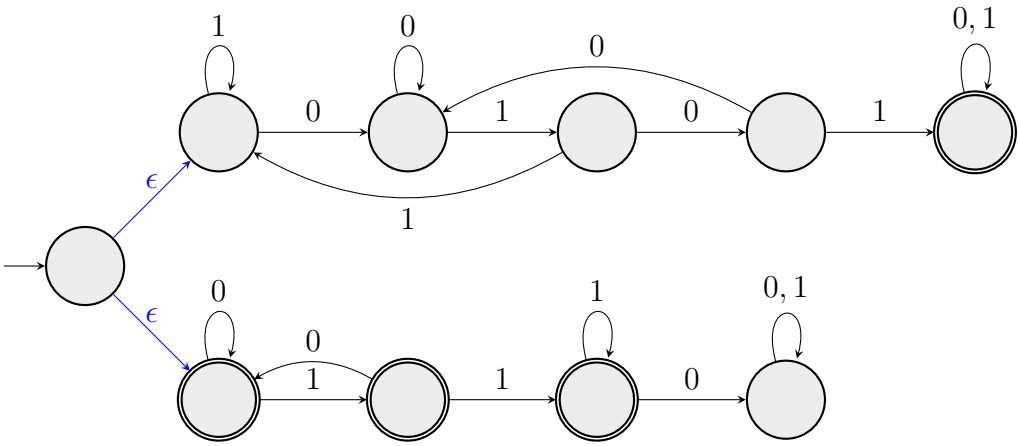


1.8 Use the construction in the proof of Theorem 1.45 to give the state diagrams of NFAs recognizing the union of the languages described in:

(a) Exercises 1.6a and 1.6b

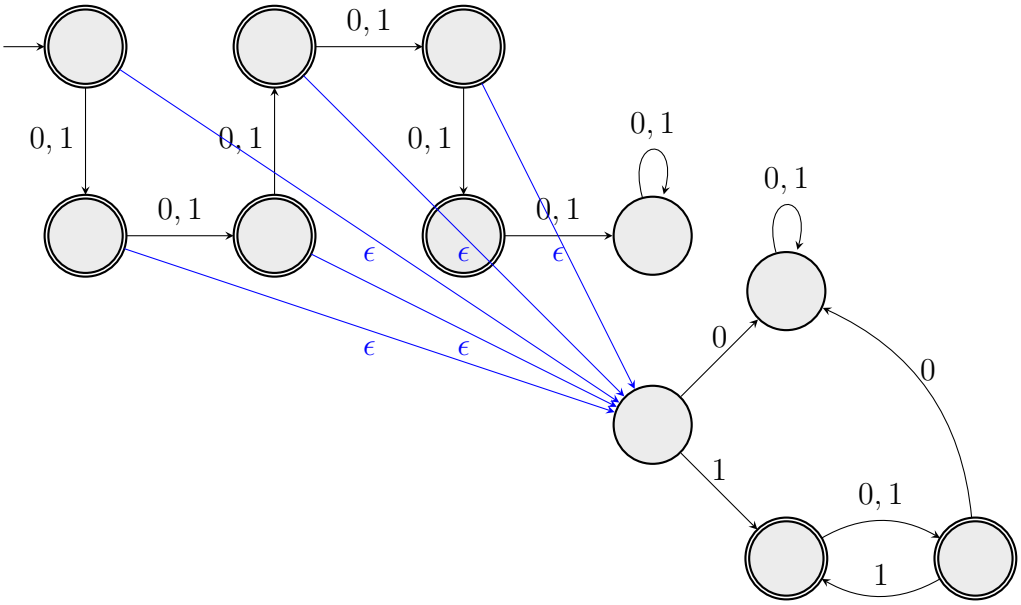


(b) Exercises 1.6c and 1.6f

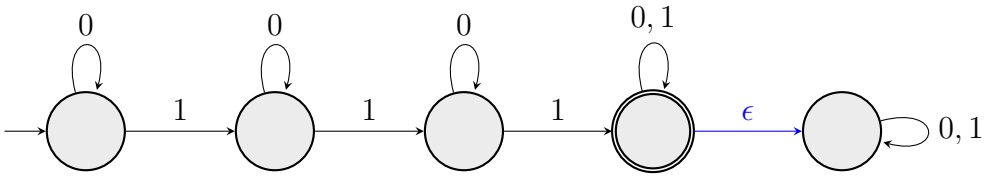


1.9 Use the construction in the proof of Theorem 1.47 to give the state diagrams of NFAs recognizing the concatenation of the languages described in

(a) Exercises 1.6g and 1.6i.

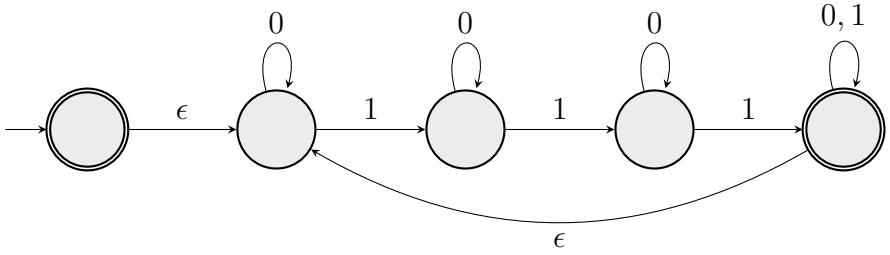


(b) Exercises 1.6b and 1.6m.

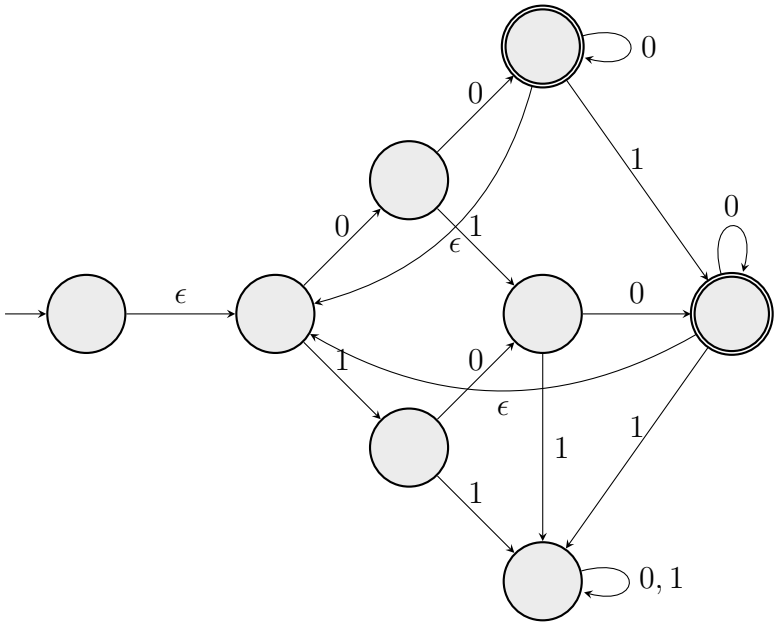


1.10 Use the construction in the proof of Theorem 1.49 to give the state diagrams of NFAs recognizing the star of the languages described in

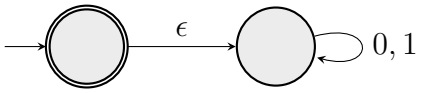
(a) Exercise 1.6b.



(b) Exercise 1.6j.



(c) Exercise 1.6m.



1.11 Prove that every NFA can be converted to an equivalent one that has a single accept state.

It is enough to show that every NFA can be converted to an equivalent one that has a single accept state and no transitions into the accept state. Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA. We construct an NFA $N' = (Q \cup \{q_f\}, \Sigma, \delta', q_0, \{q_f\})$ where δ' is the same as δ with additional transitions: for each $q \in F$ $\delta'(q, \epsilon) = \{q_f\}$. It is clear that $L(N) = L(N')$ and that N' has a single accept state. Therefore, every NFA can be converted to an equivalent one that has a single accept state.

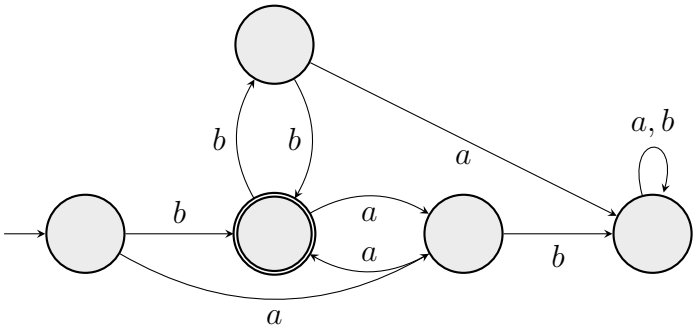
1.12 Let

$$D = \{w | w \text{ contains an even number of } a\text{'s and an odd number of } b\text{'s}$$

$$\text{and does not contain the substring } ab\}$$

Give a DFA with five states that recognizes D and a regular expression that generates D. (Suggestion: Describe D more simply.)

$D = \{w | w$ contains odd number of b's followed by even number of a's$\}$



The regular expression that generates D is $b(bb)^*(aa)^*$.

1.13 Let F be the language of all strings over $\{0, 1\}$ that do not contain a pair of 1s that are separated by an odd number of symbols. Give the state diagram of a DFA with five states that recognizes F. (You may find it helpful first to find a 4-state NFA for the complement of F.) blab

$\overline{F} = \{w | w \text{ contains a pair of 1's that are separated by an odd number of symbols}\}$

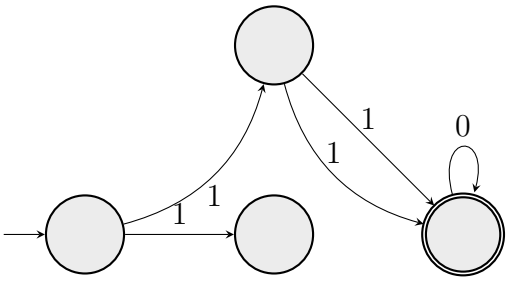


Figure 11: NFA with four states that recognizes $\overline{F}$

1.14 (a) Show that if M is a DFA that recognizes language B, swapping the accept and nonaccept states in M yields a new DFA recognizing the complement of B.Conclude that the class of regular languages is closed under complement.

Every state in DFA has a transition for every symbol in the alphabet. For every possible word, the DFA will end up in either an accept state or a nonaccept state. If we swap the accept and nonaccept states in M, the new DFA will accept the complement of the language B.

(b) Show by giving an example that if M is an NFA that recognizes language C, swapping the accept and non-accept states in M doesn't necessarily yield a new NFA that recognizes the complement of C. Is the class of languages recognized by NFAs closed under complement? Explain your answer.

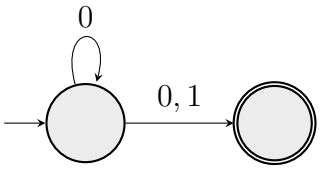


Figure 12: example NFA

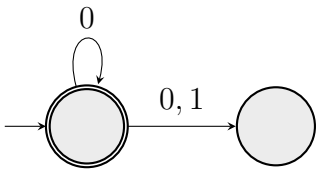


Figure 13: example NFA with changed states acceptance

NFA accepts a string if there is at least one path that leads to an accept state. If we swap the accept and non-accept states in M, the new NFA will not necessarily accept the complement of the language C. The class of languages recognized by NFAs is not closed under complement.

1.15 Give a counter example to show that the following construction fails to prove Theorem 1.49, the closure of the class of regular languages under the star operation. Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1. Construct $N = (Q_1, \Sigma, \delta, q_1, F)$ as follows. N is supposed to recognize A_1^*.

(a) The states of N are the states of N_1.

(b) The start state of N is the same as the start state of N_1.

(c) $F = \{q_1\} \cup F_1$.
The accept states F are the old accept states plus its start state.

(d) Define δ so that for any $q \in Q_1$ and any $a \in \Sigma_\epsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \notin F_1 \text{ or } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \epsilon. \end{cases}$$

(Suggestion: Show this construction graphically, as in Figure 1.50.)

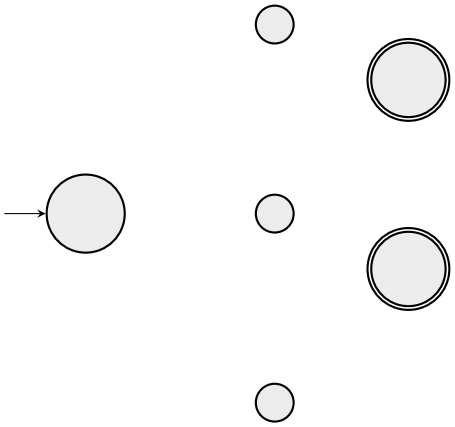


Figure 14: NFA N_1

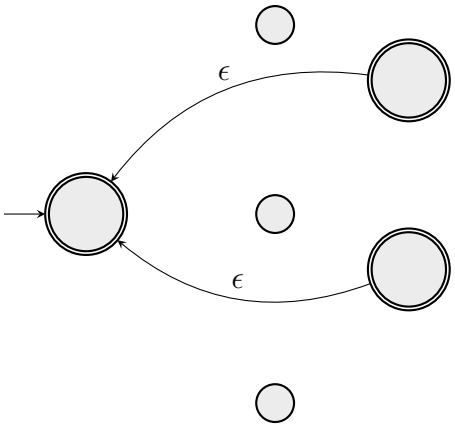


Figure 15: construction of NFA N_1

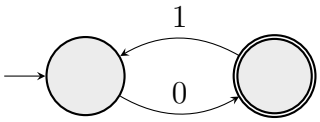


Figure 16: counter example of NFA N_1

1.16 Use the construction given in Theorem 1.39 to convert the following two nondeterministic finite automata to equivalent deterministic finite automata.

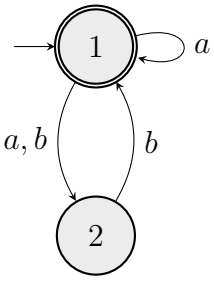


Figure 17: NFA A

(a) $A' = (Q', \Sigma, \delta', \{1\}, F')$
$Q' = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$
$\Sigma = \{a, b\}$
$F' = \{\{1\}, \{1,2\}\}$

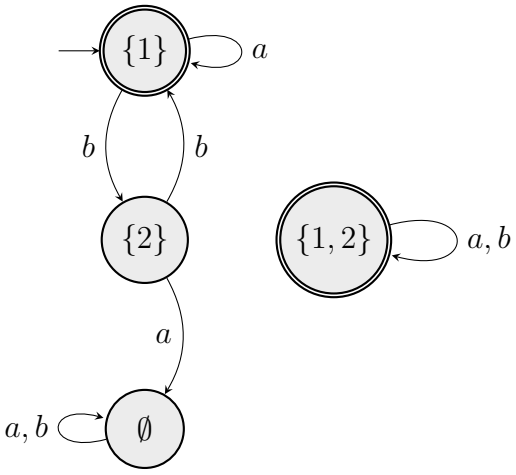


Figure 18: DFA A'

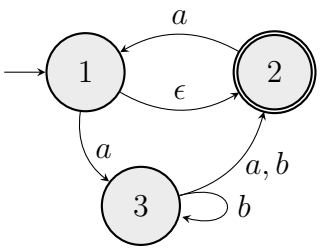


Figure 19: NFA B

(b) $B' = (Q', \Sigma, \delta', \{1\}, F')$
$Q' = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$
$\Sigma = \{a, b\}$
$F' = \{\{2\}, \{1,2\}, \{2,3\}, \{1,2,3\}\}$

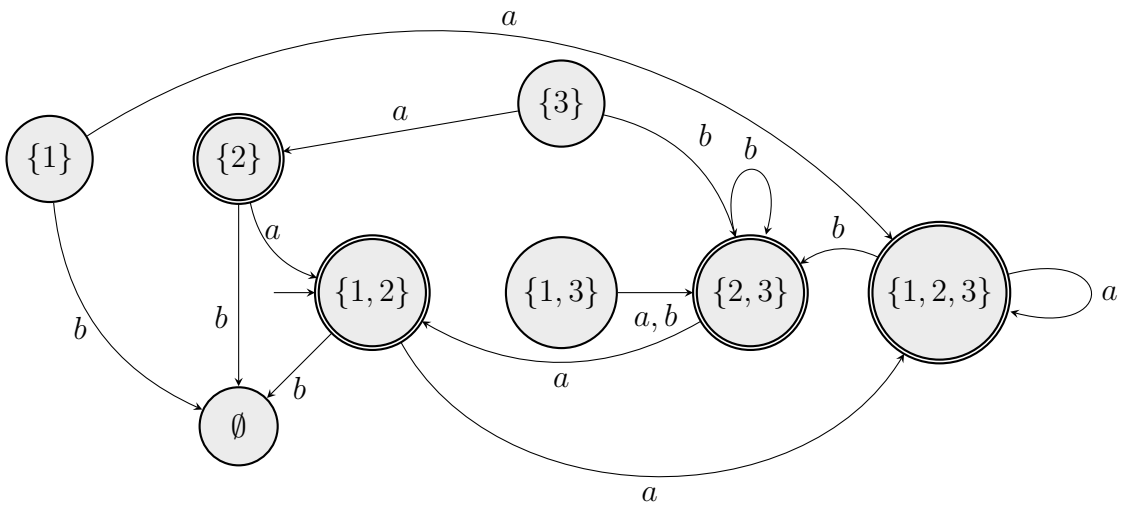


Figure 20: DFA B'

1.17 (a) Give an NFA recognizing the language $(01 \cup 001 \cup 010)^*$

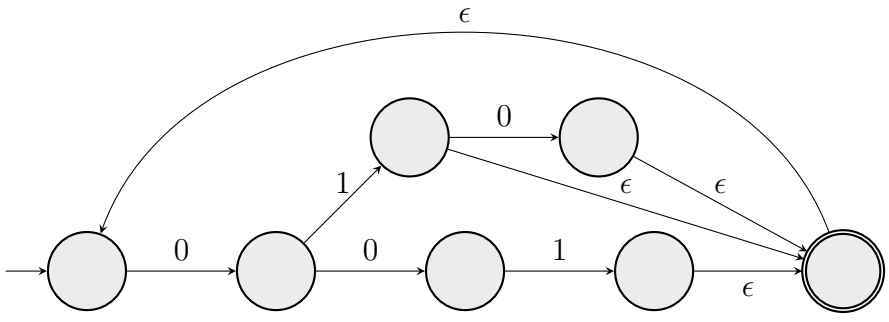


Figure 21: NFA recognizing the language $(01 \cup 001 \cup 010)^*$

(b) Convert this NFA to an equivalent DFA. Give only the portion of the DFA that is reachable from the start state.

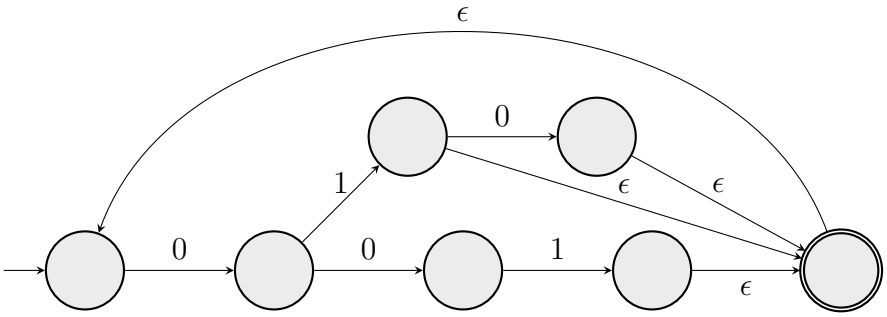


Figure 22: DFA recognizing the language $(01 \cup 001 \cup 010)^*$

1.18 Give regular expressions generating the languages of Exercise 1.6.

(a) $\{w|w$ begins with a 1 and ends with a $0\}$

$$1\Sigma^*0$$

(b) $\{w|w$ contains at least three 1's$\}$

$$\Sigma^*1\Sigma^*1\Sigma^*1\Sigma^*$$

(c) $\{w|w$ contains the substring 0101$\}$

$$\Sigma^*0101\Sigma^*$$

(d) $\{w|w$ has length at least 3 and its third symbol is a $0\}$

$$\Sigma\Sigma0\Sigma^*$$

(e) $\{w|w$ starts with 0 and has odd length, or starts with 1 and has even length$\}$

$$0(\Sigma\Sigma)^* \cup 1\Sigma(\Sigma\Sigma)^*$$

(f) $\{w|w$ doesn't contain the substring 110$\}$

$$(0 \cup 10)^*(1 \cup 111^* \cup \epsilon) = (0 \cup 10)^*1^*$$

in details:

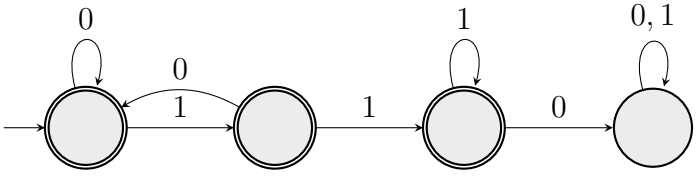


Figure 23: DFA that recognizes the language

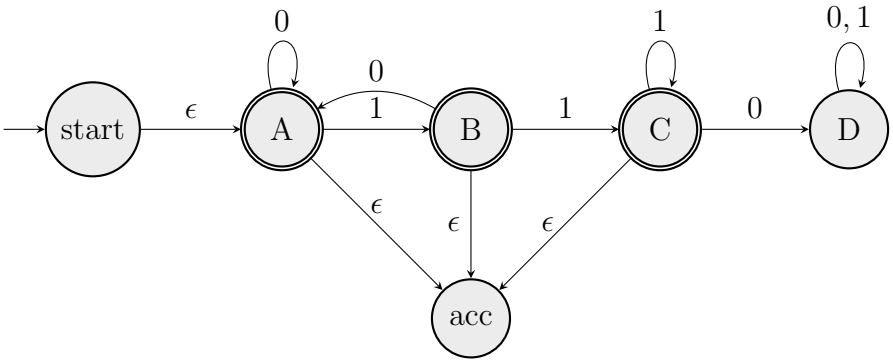


Figure 24: GNFA that recognizes the language

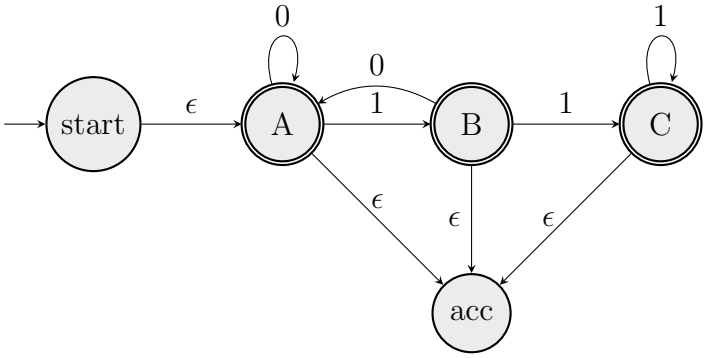


Figure 25: GNFA - removed state D

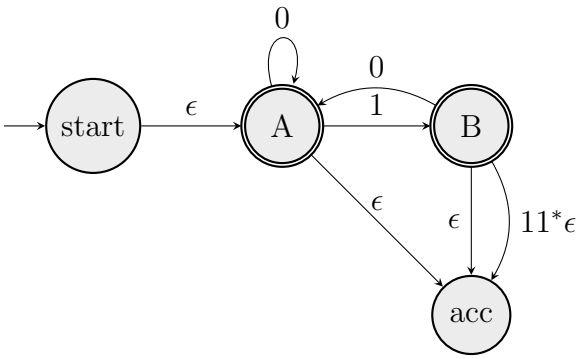


Figure 26: GNFA - removed state C

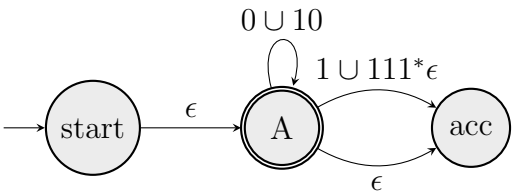


Figure 27: GNFA - removed state B

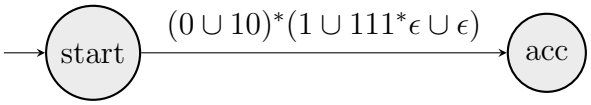


Figure 28: GNFA - removed state A

(g) $\{w|$the length of w is at most $5\}$

$$(\epsilon \cup 0 \cup 1)(\epsilon \cup 0 \cup 1)(\epsilon \cup 0 \cup 1)(\epsilon \cup 0 \cup 1)(\epsilon \cup 0 \cup 1)$$

(h) $\{w|w$ is any string except 11 and 111$\}$

$$\epsilon \cup 0\Sigma^* \cup 1 \cup 10\Sigma^* \cup 110\Sigma^* \cup 1110\Sigma^* \cup 1111\Sigma^*$$

(i) $\{w|$ every odd position of w is a 1$\}$

$$1((\Sigma)1)^*$$

(j) $\{w|w$ contains at least two 0's and at most one 1$\}$

$$00^*0 \cup 100^*0 \cup 00^*10^*0 \cup 000^*1$$

(k) $\{\epsilon, 0\}$

$$\emptyset^* \cup 0$$

(l) $\{w|w$ contains an even number of 0's, or contains exactly two 1's$\}$

$$(1^*01^*01^*)^* \cup 0^*10^*10^*$$

(m) The empty set

$$1\emptyset$$

(n) All strings except the empty string

$$\Sigma\Sigma^*$$

1.19 Use the procedure described in Lemma 1.55 to convert the following regular expressions to nondeterministic finite automata.

(a) $(0 \cup 1)^*000(0 \cup 1)^*$

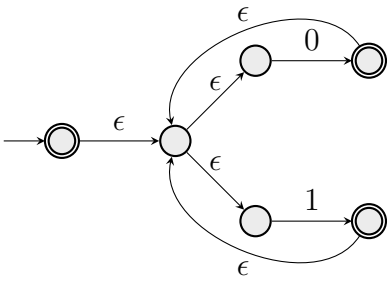


Figure 29: $(0 \cup 1)^*$

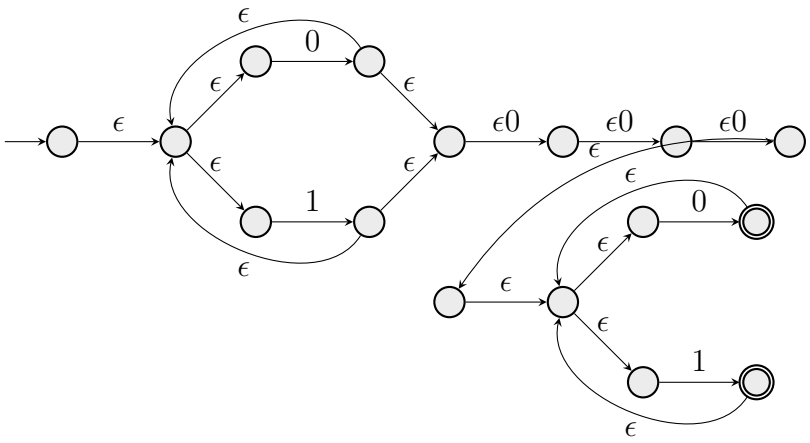


Figure 30: $(0 \cup 1)^*000(0 \cup 1)^*$

(b) $(((00)^*(11)) \cup 01)^*$

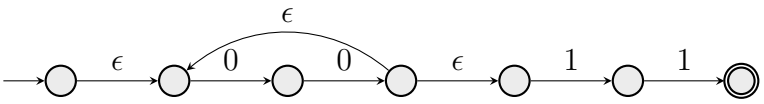


Figure 31: $((00)^*(11))$

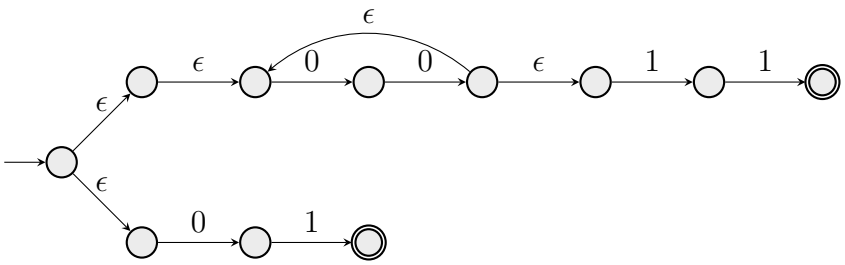


Figure 32: $((00)^*(11)) \cup 01$

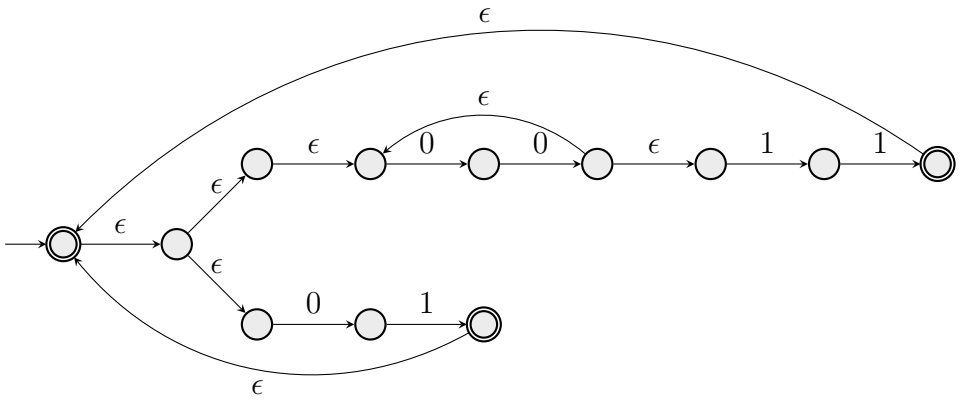
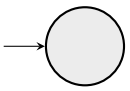


Figure 33: $(((00)^*(11)) \cup 01)^*$

(c) $\emptyset$



1.20 For each of the following languages, give two strings that are members and two strings that are not members—a total of four strings for each part. Assume the alphabet $\Sigma = \{a, b\}$ in all parts.

(a) a^*b^*

Members	Not Members
ϵ	$bbba$
ab	$abab$

(b) $a(ba)^*b$

Members	Not Members
ab	b
$aababbabb$	$aaab$

(c) $a^* \cup b^*$

Members	Not Members
aaa	ab
$bbbb$	$baba$

(d) $(aaa)^*$

Members	Not Members
ϵ	b
$aaaaaa$	aa

(e) $\Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$

Members	Not Members
aba	$aaaa$
$ababbbb$	bb

(f) $aba \cup bab$

Members	Not Members
aba	$ababab$
bab	abb

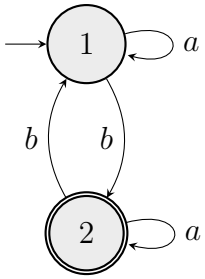
(g) $(\epsilon \cup a)b$

Members	Not Members
b	ϵ
ab	$aaba$

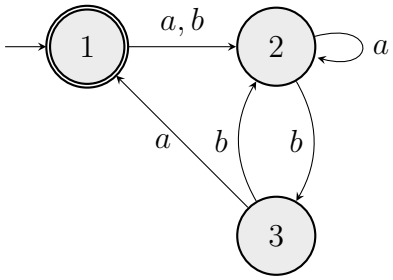
(h) $(a \cup ba \cup bb)\Sigma$

Members	Not Members
aa	$bbaa$
bbb	abb

1.21 Use the procedure described in Lemma 1.60 to convert the following finite automata to regular expressions.



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



(b) $a(a^* \cup bb)a \cup b(a^* \cup bb)a$

1.22 In certain programming languages, comments appear between delimiters such as /# and #/.Let C be the language of all valid delimited comment strings. A member of C must begin with /# and end with # /but have no intervening # /. For simplicity, assume that the alphabet for C is $\Sigma = \{a, b, /, \#\}$.

(a) Give a DFA that recognizes C.

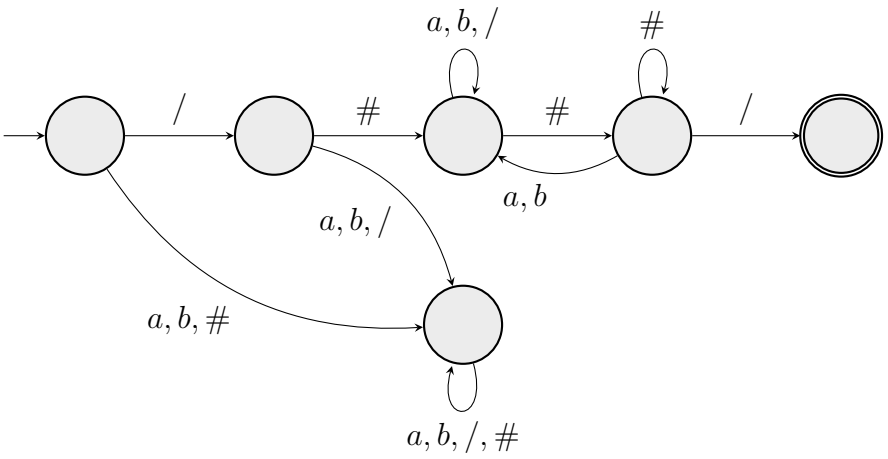


Figure 34: DFA recognizing C

(b) Give a regular expression that generates C.

$$/\#(a \cup b \cup / \cup (\#^*(a \cup b)))^*\#/$$

1.23 Let B be any language over the alphabet Σ.
Prove that $B = B^+$ iff $BB \subseteq B$.

Step 1
Assume $B = B^+$.
Let $x \in BB$.
Then $x = uv$ for some $u, v \in B$.
Since $B = B^+$, $u \in B^+$ and $v \in B^+$.
Therefore $uv \in B^+$.
Since $B^+ = B$, $uv \in B$.
Therefore $BB \subseteq B$.

Step 2
Assume $BB \subseteq B$.
Let $x \in B^+$.

Then $x = uv$ for some $u, v \in B$.
Since $BB \subseteq B$, $uv \in B$.
Therefore $B^+ \subseteq B$.
Since $B \subseteq B^+$ and $B^+ \subseteq B$ then $B = B^+$.
Therefore $B = B^+$ iff $BB \subseteq B$.

1.24 A finite state transducer (FST) is a type of deterministic finite automaton whose output is a string and not just accept or reject. The following are state diagrams of finite state transducers T_1 and T_2.

(a) T_1 on input 011

States	Output
$q1, q1, q1, q1$	000

(b) T_1 on input 211

States	Output
$q1, q2, q2, q2$	111

(c) T_1 on input 121

States	Output
$q1, q1, q2, q2$	011

(d) T_1 on input 0202

States	Output
$q1, q1, q2, q1, q2$	0101

(e) T_2 oninput b

States	Output
$q1, q3$	1

(f) T_2 on input $bbab$

States	Output
$q1, q3, q2, q3, q2$	1111

(g) T_2 on input *bbbbbb*

States	Output
$q1, q3, q2, q1, q3, q2, q1$	110110

(h) T_2 on input ϵ

States	Output
$q1$	ϵ

1.25 Read the informal definition of the finite state transducer given in Exercise 1.24. Give a formal definition of this model, following the pattern in Definition 1.5. Assume that an FST has an input alphabet Σ and an output alphabet Γ but not a set of accept states. Include a formal definition of the computation of an FST. (Hint: An FST is a 5-tuple. Its transition function is of the form $\delta : Q \times \Sigma \longrightarrow Q \times \Gamma$.)

A finite state transducer is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

(a) Q is a finite set called the states,

(b) Σ is a finite set called the alphabet,

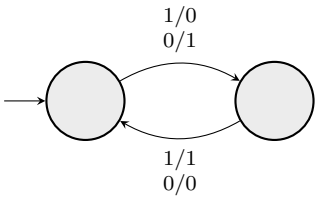
(c) $\delta : Q \times \Sigma \longrightarrow Q \times \Gamma$ is the transition and print function,

(d) $q_0 \in Q$ is the start state, and

(e) $F \subseteq Q$ is the set of accept states.

1.26 Using the solution you gave to Exercise 1.25, give a formal description of the machines T_1 and T_2 depicted in Exercise 1.24.

1.27 Read the informal definition of the finite state transducer given in Exercise 1.24. Give the state diagram of an FST with the following behavior. Its input and output alphabets are $\{0, 1\}$. Its output string is identical to the input string on the even positions but inverted on the odd positions. For example, on input 0000111 it should output 1010010.



1.28 Convert the following regular expressions to NFAs using the procedure given in Theorem 1.54. In all parts, $\Sigma = \{a, b\}$.

(a) $a(abb)^* \cup b$

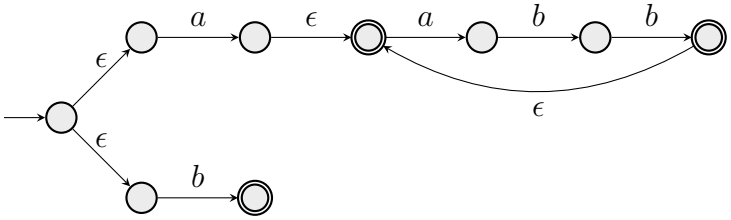


Figure 35: NFA for $a(abb)^* \cup b$

(b) $a^+ \cup (ab)^+$

$a^+ \cup (ab)^+ = aa^* \cup ab(ab)^*$

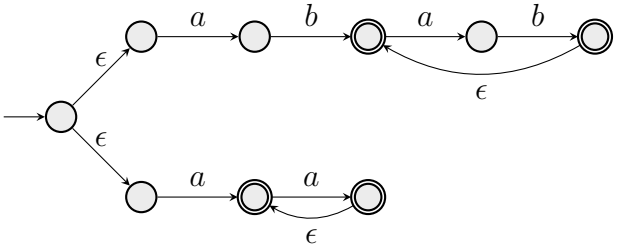


Figure 36: NFA for $a^+ \cup (ab)^+$

(c) $(a \cup b^+)a^+b$
$(a \cup b^+)a^+b = (a \cup bb^*)aa^*b$

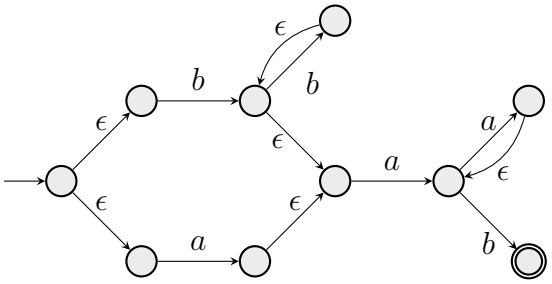


Figure 37: NFA for $(a \cup b^+)a^+b$

1.29 Use the pumping lemma to show that the following languages are not regular.

(a) $A_1 = \{0^n1^n2^n \mid n \geq 0\}$

(b) $A_2 = \{www \mid w \in \{a, b\}^*\}$

(c) $A_3 = \{a^{2^n} \mid n \geq 0\}$ (Here, a^{2^n} means a string of 2^n a's.)

(a) Let p be the pumping length, and consider the string $0^p1^p2^p$. We can pump one symbol at a time (0, 1 or 2) but it changes the number of symbols of that type. So, the string is not in the language. Or we can pump two different symbols at a time (zeros and ones or ones and twos) but the order of symbols will change.

Therefore, A_1 is not regular.

(b) Let w be any string of length p with form $a^{p-1}ba^{p-1}ba^{p-1}b$. We can pump part of string with only one b, but it will change the number of b's in the string but number of b's won't be divided by 3. Therefore, A_2 is not regular.

(c) Choose s to be the string a^{2^p}. Because s is a member of A_3 and s is longer than p, can be split into three pieces, $s = xyz$. $|xy| \geq p$. Furthermore, $p < 2^p$ and so $|y| < 2^p$. Therefore, $|xyyz| = |xyz| + |y| < 2^p + 2^p = 2^{p+1}$. $|y| > 0$ so $2^p < |xyyz| < 2^{p+1}$. The length of $xyyz$ cannot be a power of 2. Hence $xyyz$ is not a member of A_3, a contradiction.Therefore, A_3 is not regular.

1.30 Describe the error in the following "proof" that 0^*1^* is not a regular language. (An error must exist because 0^*1^* is regular.) The proof is

by contradiction. Assume that 0^*1^* is regular. Let p be the pumping length for 0^*1^* given by the pumping lemma. Choose s to be the string 0^p1^p. You know that s is a member of 0^*1^*, but Example 1.73 shows that s cannot be pumped. Thus you have a contradiction. So 0^*1^* is not regular.

We can split s into three pieces, $s = xyz$ with y contains only 0's or only 1's. Then, we can pump y and the number of 0's or 1's will change. Therefore, s isn't good choice for pumping lemma 0^*1^* can be regular.

2 Problems

1.31 For any string $w = w_1 w_2 \dots w_n$, the reverse of w, written w^R, is the string w in reverse order, $w_n \dots w_2 w_1$. For any language A, let $A^R = \{w^R \mid w \in A\}$. Show that if A is regular, so is A^R.

Let

$$D = \{Q, \Sigma, \delta, q_0, F\}$$

be DFA recognizes A. We will construct a NFA

$$D^R = \{Q^R, \Sigma, \delta^R, q_0^R, \{q_0\}\}$$

that recognizes A^R.

$Q^R = Q \cup \{q_0^R\}$ where q_0^R is new starting state connected with ϵ-edges/transitions to each accepting state of Q from F
$F^R = \{q_0\}$ (starting state from D is accepting state of D')
$\delta^R(q, a) = \{p \mid \delta(p, a) = q\}$
and $\delta^R(q_0^R, \epsilon) = F$.

1.32 Let $\Sigma_3 = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, ..., \begin{bmatrix}1\\1\\1\end{bmatrix} \right\}$. Σ_3 contains all size 3 columns of 0's and 1's. A string of symbols in Σ_3 gives three rows of 0's and 1's. Consider each row to be a binary number and let
$B = \{w \in \Sigma_3^* \mid$ the bottom row of w is the sum of the top two rows$\}$.
For example, $\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} \in B$, but $\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix} \notin B$. Show that B is regular. (Hint: Working with B^R is easier. You may assume the result claimed in Problem 1.31.)

Let

$0 = \begin{bmatrix}0\\0\\0\end{bmatrix}$ $1 = \begin{bmatrix}0\\0\\1\end{bmatrix}$ $2 = \begin{bmatrix}0\\1\\0\end{bmatrix}$ $\dots$ $7 = \begin{bmatrix}1\\1\\1\end{bmatrix}$ be shorthand notation for each element of Σ_3.

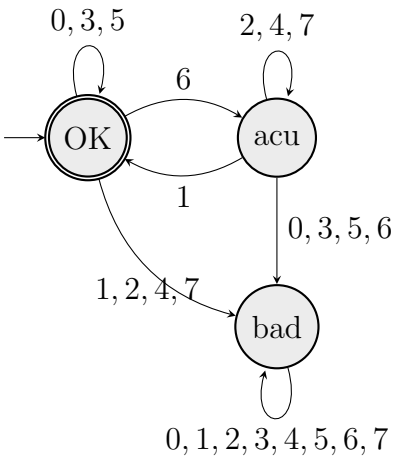


Figure 38: DFA that recognizes $\Sigma_3^{*R} = B^R$

The DFA above recognizes B^R. By Problem 1.31, B is regular.

1.33 Let $\Sigma_2 = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$. Here, Σ_2 contains all columns of 0's and 1's of height two. A string of symbols in Σ_2 gives two rows of 0's and 1's. Consider each row to be a binary number and let $C = \{w \in \Sigma_2^* \mid \text{the bottom row of } w \text{is three times the top row}\}$. For example, $\begin{bmatrix} 0 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix} \in C$, but $\begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} \notin C$. Show that C is regular. (You may assume the result claimed in Problem 1.31.)

Let $0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, and $3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ be shorthand notation for each element of Σ_2.

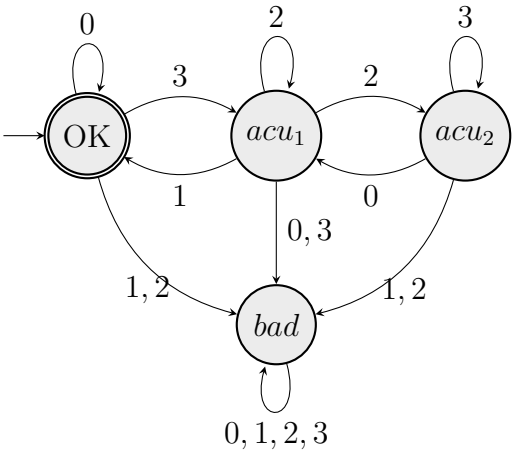


Figure 39: DFA that recognizes C^R

The DFA above recognizes C^R. By Problem 1.31, C is regular.

1.34 Let Σ_2 be the same as in Problem 1.33. Consider each row to be a binary number and let
$D = \{w \in \Sigma_2^* \mid \text{the top row of } w \text{ is a larger number than is the bottom row}\}$.
For example, $\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix} \in D$, but $\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix} \notin D$.
Show that D is regular.

Let $0 = \begin{bmatrix}0\\0\end{bmatrix}$, $1 = \begin{bmatrix}0\\1\end{bmatrix}$, $2 = \begin{bmatrix}1\\0\end{bmatrix}$, and $3 = \begin{bmatrix}1\\1\end{bmatrix}$ be shorthand notation for each element of Σ_2
and assume that $\epsilon \notin D$.

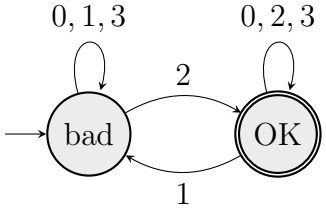


Figure 40: DFA that recognizes D^R

The DFA above recognizes D^R. By Problem 1.31, D is regular.

1.35 Let Σ_2 be the same as in Problem 1.33. Consider the top and bottom rows to be strings of 0's and 1's, and let
$E = \{w \in \Sigma_2^* |$ the bottom row of w is the reverse of the top row of w$\}$.
Show that E is not regular.

Let $e = \begin{bmatrix}0\\0\end{bmatrix}^p \begin{bmatrix}1\\1\end{bmatrix} \begin{bmatrix}0\\0\end{bmatrix}^p$
$|e| = 2p + 1 \geq p$
All possible divisions of e into xyz such that $|xy| \leq p$ and $|y| \geq 1$ are of the form $e = xyz = \left(\begin{bmatrix}0\\0\end{bmatrix}^k\right)\left(\begin{bmatrix}0\\0\end{bmatrix}^m\right)\left(\begin{bmatrix}0\\0\end{bmatrix}^{p-k-m}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}^p\right)$ where $0 \leq k \leq p$.
$xy^2z = xyyz = \left(\begin{bmatrix}0\\0\end{bmatrix}^k\right)\left(\begin{bmatrix}0\\0\end{bmatrix}^m\right)\left(\begin{bmatrix}0\\0\end{bmatrix}^m\right)\left(\begin{bmatrix}0\\0\end{bmatrix}^{p-k-m}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}^p\right) = \left(\begin{bmatrix}0\\0\end{bmatrix}^m\right)\left(\begin{bmatrix}0\\0\end{bmatrix}^p\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}^p\right) \notin E$
Therefore, by pummping lemma, E is not regular.

1.36 Let $B_n = \{a^k |\ k \text{ is a multiple of } n\}$. Show that for each $n \geq 1$, the language B_n is regular.

Lets construct NFA accepting

For $n = 2$:

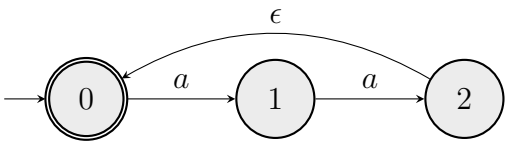


Figure 41: NFA recognicing B_2

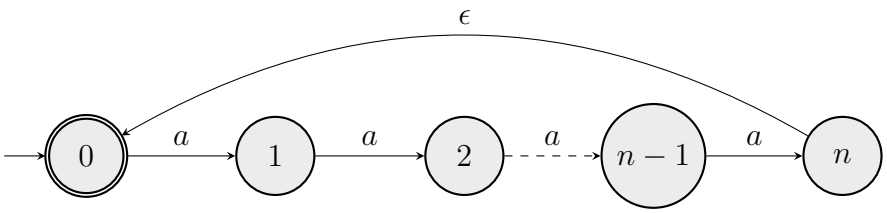


Figure 42: NFA recognicing B_n

1.37 Let $C_n = \{x \mid x \text{ is a binary number that is a multiple of } n\}$. Show that for each $n \geq 1$, the language C_n is regular.

Lets build DFA_C recognizing C_n:

$DFA_C = (Q, \Sigma, \delta, q_0, F)$

- $Q = \{0', 1', 2', \ldots, (n-1)'\}$
- $\Sigma = \{0, 1\}$
- $q_0 = 0'$
- $F = \{0'\}$
- δ is defined as follows:

$$\delta(q', 0) = 2q \pmod{n}$$
$$\delta(q', 1) = (2q + 1) \pmod{n} = (2q \pmod{n} + 1) \pmod{n}$$

where q is the integer part of q'.

We build DFA recognizing C_n so C_n is regular.

Idea

i-th state represents integers which modulo from dividing by n is i, because n is multiple of n.

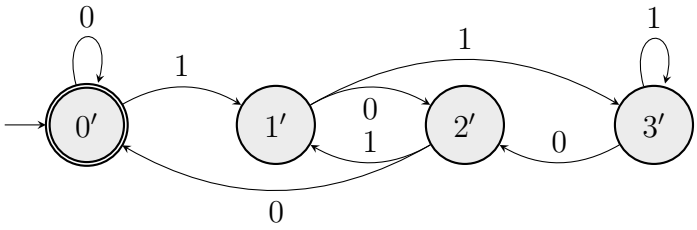


Figure 43: DFA_C for $n = 4$

$0'$: mod $4 = 0$	$1'$: mod $4 = 1$	$2'$: mod $4 = 2$	$3'$: mod $4 = 3$
$0 = (0)_2$	$1 = (1)_2$	$2 = (10)_2$	$3 = (11)_2$
$4 = (100)_2$	$5 = (101)_2$	$6 = (110)_2$	$7 = (111)_2$
$8 = (1000)_2$	$9 = (1001)_2$	$10 = (1010)_2$	$11 = (1011)_2$
...	...	...	...

1.38 An all-NFA M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ that accepts $x \in \Sigma^*$ if every possible state that M could be in after reading input x is a state from F. Note, in contrast, that an ordinary NFA accepts a string if some state among these possible states is an accept state. Prove that all-NFAs recognize the class of regular languages.

In procedure to change NFA to DFA just change step which marks state as accepting when state of DFA corresponds to states of NFA with at least one is acccepting to accepting only when all states are accepting.

Now we have DFA accepting the same language as all-NFA M so language is regular.

1.39 The construction in Theorem 1.54 shows that every GNFA is equivalent to a GNFA with only two states. We can show that an opposite phenomenon occurs for DFAs. Prove that for every $k > 1$, a language $A_k \subseteq \{0, 1\}^*$ exists that is recognized by a DFA with k states but not by one with only $k - 1$ states.

TODO: check solution

Let $A_k = \{0^*10^* \dots 0^*10^* \mid \text{where it is exactly } k - 1 \text{ ones}\}$.

We cannot build DFA with $k-1$ states recognizing A_k because we need to remember how many ones we have seen so far. We can't do this with $k-1$ states because we can't remember how many ones we have seen so far. (not sure if this is enough to prove it)

1.40 Recall that string x is a **prefix** of string y if a string z exists where $xz = y$, and that x is a **proper prefix** of y if in addition $x \neq y$. In each of the following parts, we define an operation on a language A. Show that the class of regular languages is closed under that operation.

- $L = \text{NOPREFIX}(A) = \{w \in A \mid \text{no proper prefix of } w \text{ is a member of } A\}$.
  Let be DFA $M = (Q, \Sigma, \delta, q_0, F)$ recognizes A (as A is regular language).
  We need to construct L which contains strings so that DFA M will go to accepting state only at the end of processing string. In other words for all $w = w_1 w_2 \dots w_n \in A$, $\delta(q_{i-1}, w_i) = q_i$ and $q_i \in F$ only for $i = n$.
  DFA $M_{NOPREFIX} = (Q \cup q_f, \Sigma, \delta_{NOPREFIX}, q_0, F)$
  where $\delta_{NOPREFIX}(q, a) = \delta(q, a)$ for $q \in Q$ and $\delta_{NOPREFIX}(q, a) = q_f$ for $q \in F$.
- $\text{NOEXTEND}(A) = \{w \in A \mid w \text{ is not the proper prefix of any string in } A\}$.
  During processing string w DFA M can't go from accepting state to another accepting state by any possible way.
  DFA $M_{NOEXTEND} = (Q, \Sigma, \delta, q_0, F_{NOEXTEND})$
  where $F_{NOEXTEND} = F - F'$
  where F' is set of states which can be reached from accepting state by any possible way.
  To build F' lets go from all accepting states and mark all states which can be reached from them. Then F' is set of all marked states. We can stop after $\overline{Q}$ steps where $\overline{Q}$ is number of states in DFA M. Each step we go every $a \in \Sigma$ from every accepting state or state reached from previous step.

1.41 For languages A and B,let the **perfect shuffle** of A and B be the language $\{w \mid w = a_1 b_1 \dots a_k b_k, \text{where} a_1 \dots a_k \in A \text{and} b_1 \dots b_k \in B, each a_i, b_i \in \Sigma\}$. Show that the class of regular languages is closed under perfect shuffle.

TODO - double check this solution

Let A and B be regular languages. Then there exist DFAs M_A and M_B that accept A and B, respectively. We construct a DFA M that accepts the perfect shuffle of A and B.

Let $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$. Let $Q = Q_A \times Q_B$, $q_0 = (q_{0A}, q_{0B})$, and $F = F_A \times F_B$. The transition function δ is defined as follows: for each $q \in Q$ and $a \in \Sigma$, $\delta(q, a) = (\delta_A(q_A, a), \delta_B(q_B, a))$.

We claim that $L(M) = A \circ B$. Let $w = a_1 b_1 \dots a_k b_k \in A \circ B$. Then there exists a sequence of states $r_0, r_1, \dots, r_k$ such that $r_0 = q_0$, $r_k \in F$, and $\delta(r_{i-1}, a_i) = r_i$ for $i = 1, \dots, k$. Let $r_i = (r_{iA}, r_{iB})$. Then $r_{iA} \in Q_A$ and $r_{iB} \in Q_B$. Since $r_k \in F$, $r_{kA} \in F_A$ and $r_{kB} \in F_B$. Thus $a_1 \dots a_k \in A$ and $b_1 \dots b_k \in B$. Therefore $w \in A \circ B$.

Conversely, let $w = a_1 \dots a_k b_1 \dots b_k \in A \circ B$. Then $a_1 \dots a_k \in A$ and $b_1 \dots b_k \in B$. Let $r_i = (r_{iA}, r_{iB})$ for $i = 0, \dots, k$. Then $r_0 = q_0$, $r_k \in F$, and $\delta(r_{i-1}, a_i) = r_i$ for $i = 1, \dots, k$. Therefore $w \in L(M)$.

1.42 For languages A and B, let the **shuffle** of A and B be the language $\{w \mid w = a_1 b_1 \dots a_k b_k, where a_1 \dots a_k \in A \text{and} b_1 \dots b_k \in B, \text{each } a_i, b_i \in \Sigma^*\}$. Show that the class of regular languages is closed under **shuffle**.

1.43 Let A be any language. Define DROP-OUT(A) to be the language containing all strings that can be obtained by removing one symbol from a string in A. Thus,

DROP-OUT(A) = $\{xz \mid xyz \in A \text{ where } x, z \in \Sigma^*, y \in \Sigma\}$.

Show that the class of regular languages is closed under the DROP-OUT operation. Give both a proof by picture and a more formal proof by construction as in Theorem 1.47.

1.44 Let B and C be languages over $\Sigma = \{0, 1\}$. Define $B \overset{1}{\leftarrow} C = \{w \in B \mid \text{for some} y \in C, \text{strings } w \text{ and } y \text{ contain equal numbers of 1s}\}$. Show that the class of regular languages is closed under the $\overset{1}{\leftarrow}$ operation.

1.45 Let $A/B = \{w \mid wx \in A \text{for some } x \in B\}$.Show that if A is regular and B is any language, then A/B is regular.

1.46 Prove that the following languages are not regular. You may use the pumping lemma and the closure of the class of regular languages under union, intersection, and complement.

- $\{0^n1^m0^n \mid m, n \geq 0\}$
- $\{0^m1^n \mid m \neq n\}$
- $\{w \mid w \in 0, 1^*\text{is not a palindrome}\}$
- $\{wtw \mid w, t \in 0, 1^+\}$

Let $\Sigma = \{1, \#\}$ and let $Y = \{w \mid w = x_1\#x_2\#\ldots\#x_k \text{ for } k \geq 0, \text{each} x_i \in 1^*, and x_i \neq x_j \text{for} i \neq j\}$. Prove that Y is not regular

1.47 Let $\Sigma = \{0, 1\}$ and let $D = \{w \mid w \text{contains an equal number of occurrences of the substring}$

Thus $101 \in D$ because 101 contains a single 01 and a single 10, but $1010 \notin D$ because 1010 contains two 10s and one 01. Show that D is a regular language.

1.49
- Let $B = \{1^k y \mid y \in \{0, 1\}^*\text{and } y \text{ contains at least } k \text{ 1s, for } k \geq 1\}$. Show that B is a regular language.
- Let $C = \{1^k y \mid y \in \{0, 1\}^*\text{and } y \text{ contains at most } k \text{ 1s, for } k \geq 1\}$. Show that C isn't a regular language.

1.50 Read the informal definition of the finite state transducer given in Exercise 1.24. Prove that no FST can output w^R for every input w if the input and output alphabets are $\{0, 1\}$.

1.51 Let x and y be strings and let L be any language. We say that x and y are **distinguishable by** L if some string z exists whereby exactly one of the strings xz and yz is a member of L; otherwise, for every string z, we have $xz \in L$ whenever $yz \in L$ and we say that x and y are **indistinguishable by** L.

If x and y are indistinguishable by L, we write $x \equiv_L y$. Show that $\equiv_L$ is an equivalence relation.

1.52 **Myhill–Nerode theorem**.

Refer to Problem 1.51. Let L be a language and let X be a set of strings. Say that X is **pairwise distinguishable by L** if every two distinct strings in X are distinguishable by L. Define the index of L to be the maximum number of elements in any set that is pairwise distinguishable by L. The index of L may definite or infinite.

(a) Show that if L is recognized by a DFA with k states, L has index at most k.

(b) Show that if the index of L is a finite number k, it is recognized by a DFA with k states.

(c) Conclude that L is regular iff it has finite index. Moreover, its index is the size of the smallest DFA recognizing it.

1.53 Let $\Sigma = \{0, 1, +, =\}$ and

$$ADD = \{a = b{+}c \mid a, b, c \text{ are binary integers, and } a \text{ is the sum of } b \text{ and } c\}.$$

Show that ADD is not regular.

Let p from pumping lemma be p'. Let construct word from ADD in form $10^p1 = 10^{p+1} + 1$. when we divide word s to x, y, z from pumping lemma we can get y only without $=$ or $+$ (in other case we get word with more than one $=$, $+$ sign which isn't in language ADD).

So we can pum only a or b. But when we change only sum or only one of factors we get word which isn't in ADD.

1.54 Consider the language $F = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and if } i = 1 \text{ then } j = k\}$.

(a) Show that F is not regular.

(b) Show that F acts like a regular language in the pumping lemma. In other words, give a pumping length p and demonstrate that F satisfies the three conditions of the pumping lemma for this value of p.

(c) Explain why parts (a) and (b) do not contradict the pumping lemma.

1.55 The pumping lemma says that every regular language has a pumping length p, such that every string in the language can be pumped if it has length p or more. If p is a pumping length for language A, so is any length $p' \geq p$.The minimum pumping length for A is the smallest p that is a pumping length for A. For example, if $A = 01^*$,the minimum pumping length is 2. The reason is that the string $s = 0$ is in A and has length 1 yet s cannot be pumped; but any string in A of length 2 or more contains a 1 and hence can be pumped by dividing it so that $x = 0, y = 1$, and z is the rest. For each of the following languages, give the minimum pumping length and justify your answer.

(a) 0001^*

Shortest word in language is 000 but it cannot be pumped. So, minimum pumping length is 4 as 0001 can be pumped.

(b) 0^*1^*

Minimum pumping length is 1 as 0 or 1 can be pumped. Any longer word can be pumped by pumping 0's or 1's.

(c) $001 \cup 0^*1^*$

Minimum pumping length is 1 as 0 or 1 can be pumped. Any longer word can be pumped by pumping 0's or 1's. Even word 001 can be pumped by pumping 0's or 1's as it belongs also to second of two languages sums to that language.

(d) $0^*1^+0^+1^+1^* \cup 10^*1$

Minimum pumping length has to be greater than 2 as 11 cannot be pumped. Minimum pumping length is 3 as words of form 10^+1 can be pumped. Also words from first part of union (with minimum length of 3) can be pumped by pumping 0's or 1's (as 101 belong also to second part of union but cannot be pumped within only first part of union).

(e) $(01)^*$

Minimum pumping length is 2 as 01 can be pumped.

(f) ϵ

Minimum pumping length is 1 as ϵ with length 0 cannot be pumped.

(g) $1^*01^*01^*$

Minimum pumping length is greater than 2 as 00 can be pumped. Minimum pumping length is 3 as every word with at least one 1 can be pumped.

(h) $10(11^*0)^*0$

Minimum pumping length has to be greater than 3 as 100 cannot be pumped. Minimum pumping length is 4 as words of form 10110 can be pumped (which is of length 5 but 4 also satisfies pumping lemma requirements).

(i) 1011

Minimum pumping length is 5 as 1011 (with length = 4) cannot be pumped.

(j) Σ^*

Minimum pumping length is 1 as any word can be pumped. Assuming empty string ϵ cannot be pumped as stated in selected solutions (but not sure from which lemma assumptions it can be stated).

1.56 If A is a set of natural numbers and k is a natural number greater than 1, let $B_k(A) = \{w \mid w \text{ is the representation in base } k \text{ of some number in} A\}$. Here, we do not allow leading 0s in the representation of a number. For example, $B_2(3,5) = \{11, 101\}$ and $B_3(3,5) = \{10, 12\}$. Give an example of a set A for which $B_2(A)$ is regular but $B_3(A)$ is not regular.

Prove that your example works.

1.57 If A is any language, let $A_{\frac{1}{2}-}$ be the set of all first halves of strings in A so that $A_{\frac{1}{2}-} = \{x \mid \text{for some } y, |x| = |y| \text{ and } xy \in A\}$.

Show that if A is regular, then so is $A_{\frac{1}{2}-}$.

1.58 If A is any language, let $A_{\frac{1}{3}-\frac{1}{3}}$ be the set of all strings in A with their middle thirds removed so that $A_{\frac{1}{3}-\frac{1}{3}} = \{xz \mid \text{for some } y, |x| = |y| = |z| \text{ and } xyz \in A\}$. Show that if A is regular, then $A_{\frac{1}{3}-\frac{1}{3}}$ is not necessarily regular.

1.59 Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and let h be a state of M called its "home". A synchronizing sequence for M and h is a string $s \in \Sigma^*$ where $\delta(q, s) = h$ for every $q \in Q$.

(Here we have extended δ to strings, so that $\delta(q, s)$ equals the state where M ends up when M starts at state q and reads input s.)

Say that M is synchronizable if it has a synchronizing sequence for some state h. Prove that if M is a k-state synchronizable DFA, then it has a synchronizing sequence of length at most k^3.

Can you improve upon this bound?

1.60 Let $\Sigma = \{a, b\}$. For each $k \geq 1$, let C_k be the language consisting of all strings that contain an a exactly k places from the right-hand end. Thus $C_k = \Sigma^* a \Sigma^{k-1}$. Describe an NFA with $k+1$ states that recognizes C_k in terms of both a state diagram and a formal description.

1.61 Consider the languages C_k defined in Problem 1.60. Prove that for each k, no DFA can recognize C_k with fewer than $2k$ states.

1.62 Let $\Sigma = \{a, b\}$. For each $k \geq 1$, let D_k be the language consisting of all strings that have at least one a among the last k symbols. Thus $D_k = \Sigma^* a (\Sigma \cup \epsilon)^{k-1}$. Describe a DFA with at most $k + 1$ states that recognizes D_k in terms of both a state diagram and a formal description.

1.63 (a) Let A be an infinite regular language. Prove that A can be split into two infinite disjoint regular subsets.

(b) Let B and D be two languages. Write $B \Subset D$ if $B \subseteq D$ and D contains infinitely many strings that are not in B. Show that if B and D are two regular languages where $B \Subset D$, then we can find a regular language C where $B \Subset C \Subset D$.

1.64 Let N be an NFA with k states that recognizes some language A.

(a) Show that if A is non empty, A contains some string of length at most k.

(b) Show, by giving an example, that part (a) is not necessarily true if you replace both A's by $\overline{A}$.

(c) Show that if $\overline{A}$ is non empty, $\overline{A}$ contains some string of length at most $2k$.

(d) Show that the bound given in part (c) is nearly tight; that is, for each k, demonstrate an NFA recognizing a language $\overline{A_k}$ where $\overline{A_k}$ is non empty and where $\overline{A_k}$'s shortest member strings are of length exponential in k. Come as close to the bound in (c) as you can.

1.65 Prove that for each $n > 0$, a language B_n exists where

(a) B_n is recognizable by an NFA that has n states, and

(b) if $B_n = A_1 \cup \ldots \cup A_k$, for regular languages A_i, then at least one of the A_i requires a DFA with exponentially many states.

1.66 A **homomorphism** is a function $f : \Sigma \longrightarrow \Gamma^*$ from one alphabet to strings over another alphabet. We can extend f to operate on strings by defining $f(w) = f(w_1)f(w_2)\ldots f(w_n)$, where $w = w_1w_2\ldots w_n$ and each $w_i \in \Sigma$. We further extend f to operate on languages by defining $f(A) = \{f(w)|\ w \in A\}$, for any language A.

(a) Show, by giving a formal construction, that the class of regular languages is closed under homomorphism. In other words, given a DFA M that recognizes B and a homomorphism f, construct a finite automaton M' that recognizes $f(B)$. Consider the machine M' that you constructed. Is it a DFA in every case?

(b) Show, by giving an example, that the class of non-regular languages is not closed under homomorphism.

1.67 Let the rotational closure of language A be $RC(A) = \{yx|\ xy \in A\}$.

(a) Show that for any language A, we have $RC(A) = RC(RC(A))$.

(b) Show that the class of regular languages is closed under rotational closure.

Every string s in language $RC(A)$ has a form yx where xy is in language A. If $\epsilon \in A$ then $A \subseteq RC(A)$. So $RC(A) \subseteq RC(RC(A))$.

Suppose $w \in RC(RC(A))$ and $w = yx$ where $xy \in RC(A)$. Then $xy = zt$ where $zt \in A$. So $yx = zt$ and $zt \in A$. Therefore $w \in RC(A)$.

1.68 In the traditional method for cutting a deck of playing cards, the deck is arbitrarily split two parts, which are exchanged before reassembling the deck. In a more complex cut, called Scarne's cut, the deck is broken into three parts and the middle part in placed first in the reassembly. We'll take Scarne's cut as the inspiration for an operation on languages. For a language A, let $CUT(A) = yxz|\ xyz \in A$.

(a) Exhibit a language B for which $CUT(B) = CUT(CUT(B))$.

(b) Show that the class of regular languages is closed under CUT.

1.69 Let $Sigma = \{0, 1\}$. Let $WW_k = \{ww|\ w \in \Sigma^*$ and w is of length $k\}$.

(a) Showthat foreach k, no DFA can recognize WW_k with fewer than 2^k states.

(b) Describe a much smaller NFA for $\overline{WW_k}$, the complement of WW_k.

1.70 We define the **avoids** operation for languages A and B to be

$A \textit{ avoids } B = \{w |\ w \in A \text{ and } w \text{ doesn't contain any string in } B \text{ as a substring}\}$.

Prove that the class of regular languages is closed under the avoids operation.

1.71 Let $\Sigma = \{0, 1\}$.

(a) Let $A = \{0^k u 0^k |\ k \geq 1 \text{ and } u \in \Sigma^*\}$. Show that A is regular.

(b) Let $B = \{0^k 1 u 0^k |\ k \geq 1 \text{ and } u \in \Sigma^*\}$. Show that B is not regular.

(a) construct NFA accepting A:

from starting state we can go k times to state after accepted k zeros (otherwise go to unaccepting state from which we cannot escape). Then we can read any symbol from alphabet staying in that state, or, if it is zero we can go to k states accepting k zeros.

(b) above construction couldn't work beacouse we need to remember first 1 through don't know how much symbols from u.

1.72 Let M_1 and M_2 be DFAs that have k_1 and k_2 states, respectively, and then let $U = L(M_1) \cup L(M_2)$.

(a) Show that if $U \neq \emptyset$, then U contains some string s, where $|s| < max(k_1, k_2)$.

(b) Show that if $U = \Sigma^*$, then U excludes some string s, where $|s| < k_1 k_2$.

(a) Let M_1 be a DFA defined as follows: $M_1 = (Q, \Sigma, \delta, q_s, F)$ and string $s \in L(M_1)$.

There is a path $(q_s, q_1, q_2, \ldots, q_{|s|})$ from q_s to q_f in M_1 of length $|s|$ where $q_{|s|} \in F$. It represents states in DFA during accepting s. It is of length $k_1 + 1$.

If $|s| < k_1$ the proof is done.

If $|s| \geq k_1$ then there are two states q_i and q_j such that $i < j$ and $q_i = q_j$. It means that there is a loop in DFA and we can skip it. So we can find shorter path from q_s to q_f and so we can choose s' such that $|s'| < k_1$.

1.73 Let $\Sigma = \{0, 1, \#\}$. Let $C = \{x \# x^{\mathcal{R}} \# x |\ x \in \{0, 1\}^*\}$. Show that C is a CFL.