Solved selected problems of Introduction to the Theory of Computation by Michael Sipser.

Franco Zacco

1 Regular Languages

1.1 Finite Automata

Proof. **1.1**

a. The start state of machine M_1 and $M2$ is q_1.

b. The set of accept states of machine M_1 is $\{q_2\}$.

The set of accept states of machine M_2 is $\{q_1, q_4\}$.

c. The sequence of states for machine M_1 when the input is *aabb* is

$$q_1 \to^a q_2 \to^a q_3 \to^b q_1 \to^b q_1$$

The se1quence of states for machine M_2 when the input is *aabb* is

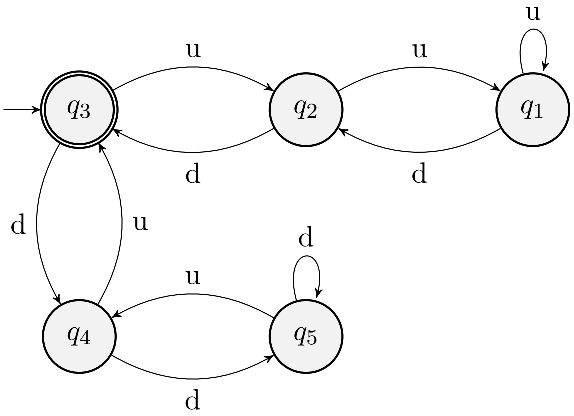
$$q_1 \to^a q_1 \to^a q_1 \to^b q_2 \to^b q_4$$

d. The machine M_1 doesn't accept the string *aabb* but the machine M_2 accepts it.

e. The machine M_1 doesn't accept the empty string ε but the machine M_2 accepts it.

□

Proof. **1.3** □



Proof. **1.4**

a. Let M_1 be a machine that checks if the input string has at least 3 a's then

$$M_1 = \{\{q_1, q_2, q_3, q_4\}, \{a, b\}, \delta_1, q_1, \{q_4\}\}$$

where δ_1 is

	a	b
q_1	q_2	q_1
q_2	q_3	q_2
q_3	q_4	q_3
q_4	q_4	q_4

Let also, M_2 be a machine that checks if the input string has at least 2 b's then

$$M_2 = \{\{q_1, q_2, q_3\}, \{a, b\}, \delta_2, q_1, \{q_3\}\}$$

where δ_2 is

	a	b
q_1	q_1	q_2
q_2	q_2	q_3
q_3	q_3	q_3

Now we combine both machines into M defined as

$$M = \{Q, \{a, b\}, \delta, (q_1, q_1), \{(q_4, q_3)\}\}$$

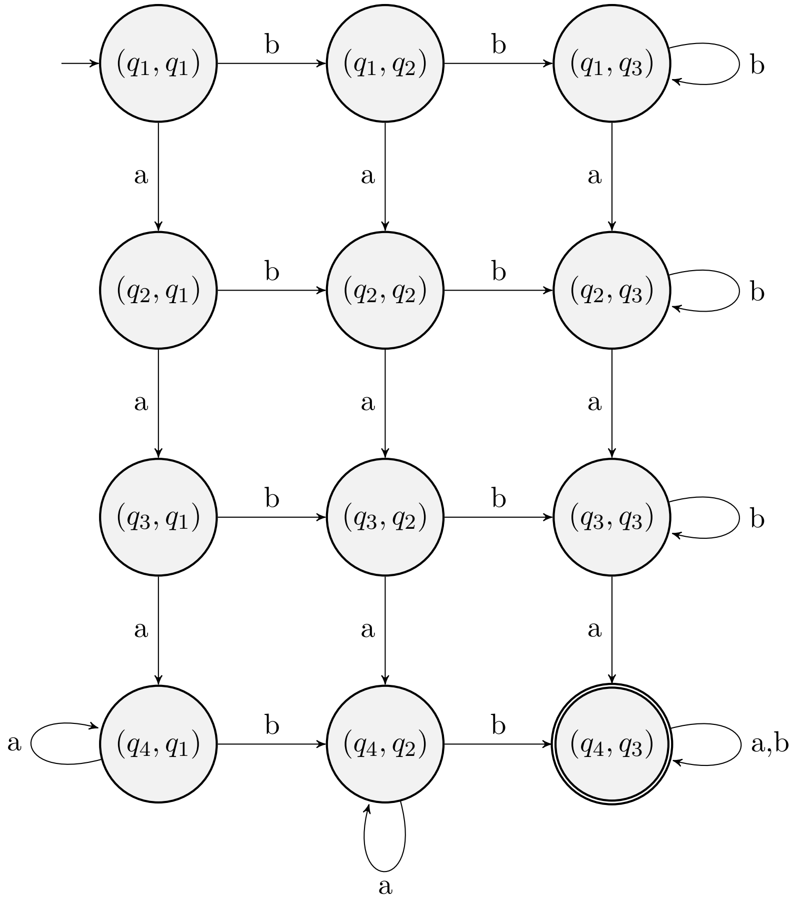
where Q is

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

and δ is

	a	b
(q_1, q_1)	(q_2, q_1)	(q_1, q_2)
(q_1, q_2)	(q_2, q_2)	(q_1, q_3)
(q_1, q_3)	(q_2, q_3)	(q_1, q_3)
(q_2, q_1)	(q_3, q_1)	(q_2, q_2)
(q_2, q_2)	(q_3, q_2)	(q_2, q_3)
(q_2, q_3)	(q_3, q_3)	(q_2, q_3)
(q_3, q_1)	(q_4, q_1)	(q_3, q_2)
(q_3, q_2)	(q_4, q_2)	(q_3, q_3)
(q_3, q_3)	(q_4, q_3)	(q_3, q_3)
(q_4, q_1)	(q_4, q_1)	(q_4, q_2)
(q_4, q_2)	(q_4, q_2)	(q_4, q_3)
(q_4, q_3)	(q_4, q_3)	(q_4, q_3)

Finally, the state diagram for this machine is given by



b. Let M_1 be a machine that checks if the input string has exactly 2 a's then

$$M_1 = \{\{q_1, q_2, q_3, q_4\}, \{a, b\}, \delta_1, q_1, \{q_3\}\}$$

where δ_1 is

	a	b
q_1	q_2	q_1
q_2	q_3	q_2
q_3	q_4	q_3
q_4	q_4	q_4

Let us take M_2 as defined in part **a.** Now we combine both machines into M defined as

$$M = \{Q, \{a, b\}, \delta, (q_1, q_1), \{(q_3, q_3)\}\}$$

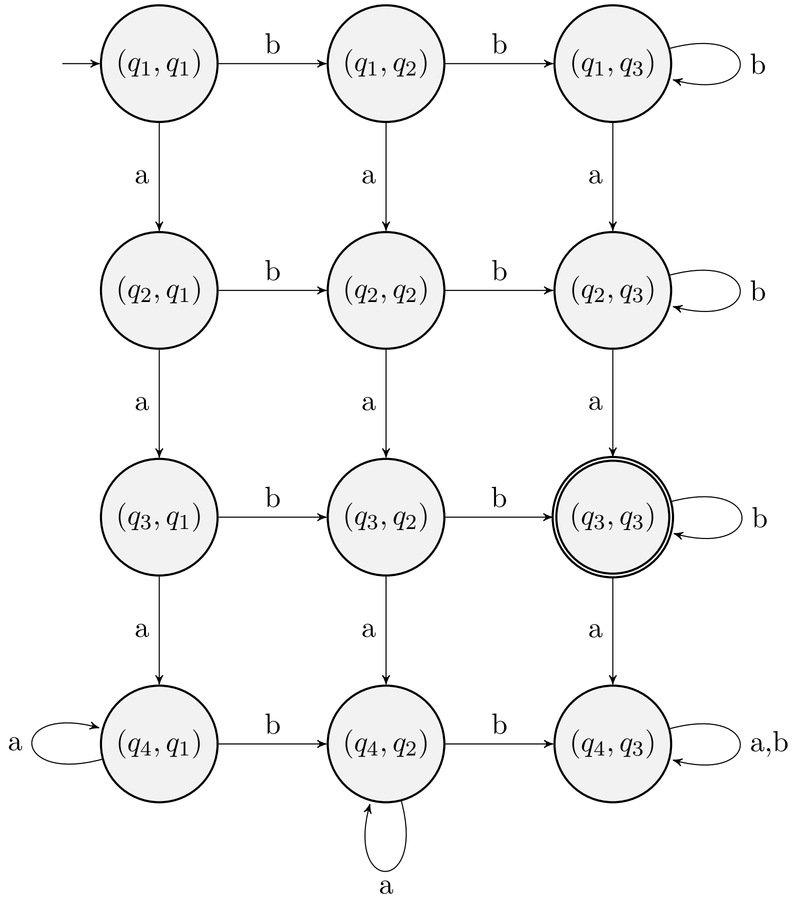
where Q is

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

and δ is

	a	b
(q_1, q_1)	(q_2, q_1)	(q_1, q_2)
(q_1, q_2)	(q_2, q_2)	(q_1, q_3)
(q_1, q_3)	(q_2, q_3)	(q_1, q_3)
(q_2, q_1)	(q_3, q_1)	(q_2, q_2)
(q_2, q_2)	(q_3, q_2)	(q_2, q_3)
(q_2, q_3)	(q_3, q_3)	(q_2, q_3)
(q_3, q_1)	(q_4, q_1)	(q_3, q_2)
(q_3, q_2)	(q_4, q_2)	(q_3, q_3)
(q_3, q_3)	(q_4, q_3)	(q_3, q_3)
(q_4, q_1)	(q_4, q_1)	(q_4, q_2)
(q_4, q_2)	(q_4, q_2)	(q_4, q_3)
(q_4, q_3)	(q_4, q_3)	(q_4, q_3)

Finally, the state diagram for this machine is given by



c. Let M_1 be a machine that checks if the input string an even number of a's then

$$M_1 = \{\{q_1, q_2\}, \{a, b\}, \delta_1, q_1, \{q_1\}\}$$

where δ_1 is

	a	b
q_1	q_2	q_1
q_2	q_1	q_2

Let also, M_2 be a machine that checks if the input string has one or two b's then

$$M_2 = \{\{q_1, q_2, q_3, q_4\}, \{a, b\}, \delta_2, q_1, \{q_2, q_3\}\}$$

where δ_2 is

	a	b
q_1	q_1	q_2
q_2	q_2	q_3
q_3	q_3	q_4
q_4	q_4	q_4

Now we combine both machines into M defined as

$$M = \{Q, \{a, b\}, \delta, (q_1, q_1), \{(q_1, q_2), (q_1, q_3)\}\}$$

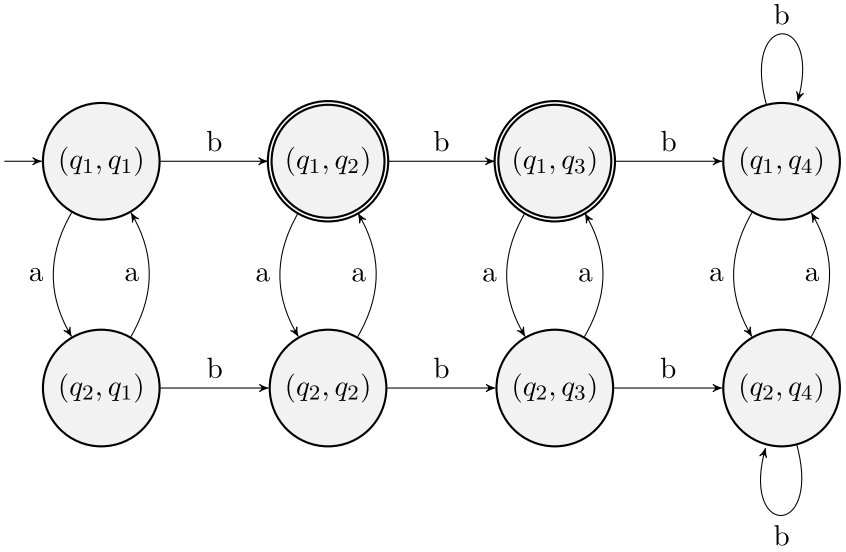
where Q is

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

and δ is

	a	b
(q_1, q_1)	(q_2, q_1)	(q_1, q_2)
(q_1, q_2)	(q_2, q_2)	(q_1, q_3)
(q_1, q_3)	(q_2, q_3)	(q_1, q_4)
(q_1, q_4)	(q_2, q_4)	(q_1, q_4)
(q_2, q_1)	(q_1, q_1)	(q_2, q_2)
(q_2, q_2)	(q_1, q_2)	(q_2, q_3)
(q_2, q_3)	(q_1, q_3)	(q_2, q_4)
(q_2, q_4)	(q_1, q_4)	(q_2, q_4)

Finally, the state diagram for this machine is given by



d. Let M_1 be the machine defined in part **c**. Let also, M_2 be a machine that checks if the input string has each a followed by at least one b then

$$M_2 = \{\{q_1, q_2, q_3\}, \{a, b\}, \delta_2, q_1, \{q_1\}\}$$

where δ_2 is

	a	b
q_1	q_2	q_1
q_2	q_3	q_1
q_3	q_3	q_3

Now we combine both machines into M defined as

$$M = \{Q, \{a, b\}, \delta, (q_1, q_1), \{(q_1, q_1)\}\}$$

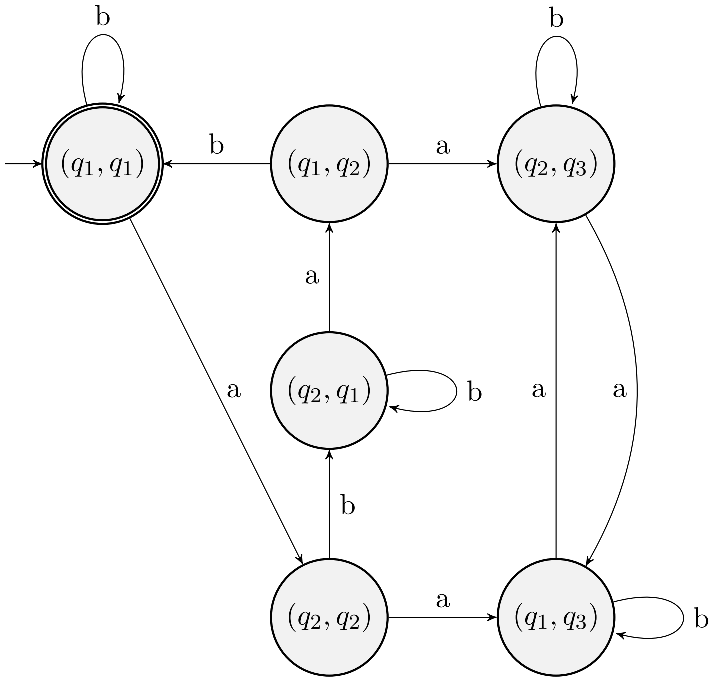
where Q is

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

and δ is

	a	b
(q_1, q_1)	(q_2, q_2)	(q_1, q_1)
(q_1, q_2)	(q_2, q_3)	(q_1, q_1)
(q_1, q_3)	(q_2, q_3)	(q_1, q_3)
(q_2, q_1)	(q_1, q_2)	(q_2, q_1)
(q_2, q_2)	(q_1, q_3)	(q_2, q_1)
(q_2, q_3)	(q_1, q_3)	(q_2, q_3)

Finally, the state diagram for this machine is given by



e. Let M_1 be a machine that checks if the input string starts with an a then

$$M_1 = \{\{q_1, q_2, q_3\}, \{a, b\}, \delta_1, q_1, \{q_2\}\}$$

where δ_1 is

	a	b
q_1	q_2	q_3
q_2	q_2	q_2
q_3	q_3	q_3

Let also, M_2 be a machine that checks if the input string has at most one b then

$$M_2 = \{\{q_1, q_2, q_3\}, \{a, b\}, \delta_2, q_1, \{q_2\}\}$$

where δ_2 is

	a	b
q_1	q_1	q_2
q_2	q_2	q_3
q_3	q_3	q_3

Now we combine both machines into M defined as

$$M = \{Q, \{a, b\}, \delta, (q_1, q_1), \{(q_2, q_2)\}\}$$

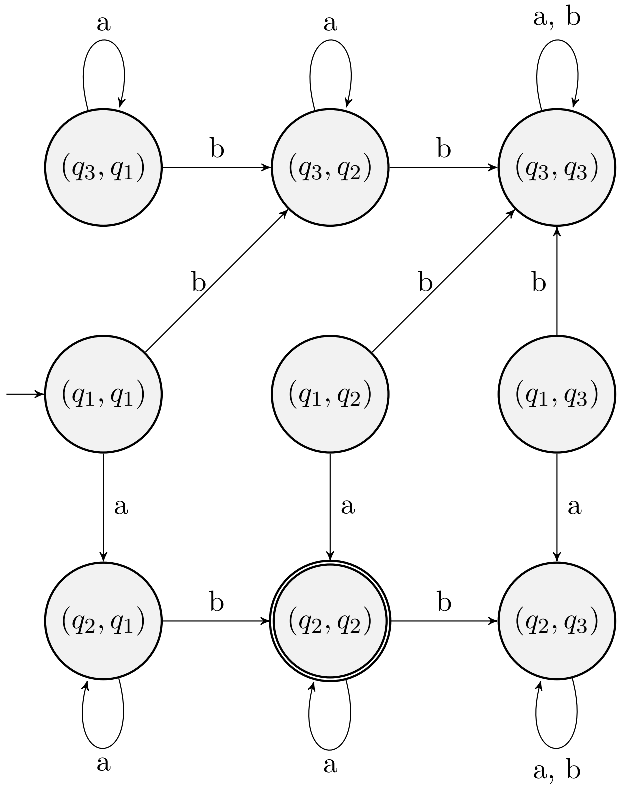
where Q is

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

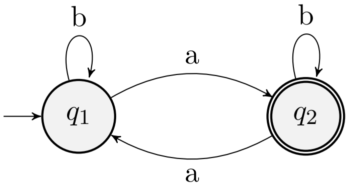
and δ is

	a	b
(q_1, q_1)	(q_2, q_1)	(q_3, q_2)
(q_1, q_2)	(q_2, q_2)	(q_3, q_3)
(q_1, q_3)	(q_2, q_3)	(q_3, q_3)
(q_2, q_1)	(q_2, q_1)	(q_2, q_2)
(q_2, q_2)	(q_2, q_2)	(q_2, q_3)
(q_2, q_3)	(q_2, q_3)	(q_2, q_3)
(q_3, q_1)	(q_3, q_1)	(q_3, q_2)
(q_3, q_2)	(q_3, q_2)	(q_3, q_3)
(q_3, q_3)	(q_3, q_3)	(q_3, q_3)

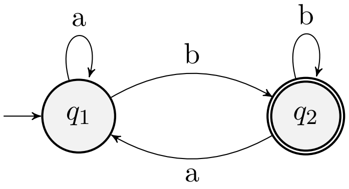
Finally, the state diagram for this machine is given by



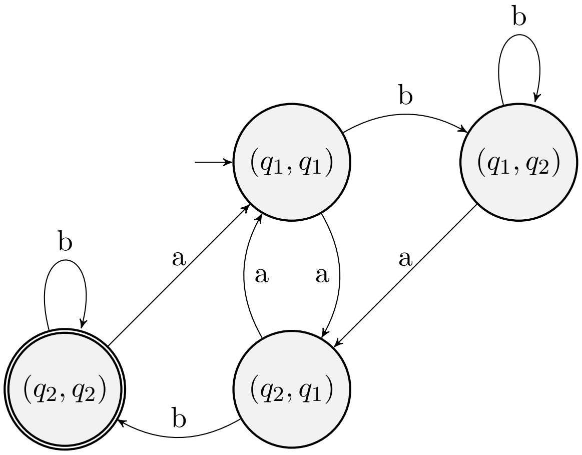
f. Let M_1 be a machine that checks if the input string has an odd number of a's then M_1 is defined as



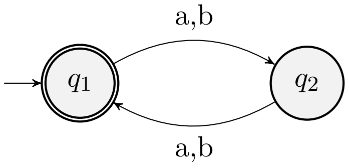
Let also, M_2 be a machine that checks if the input string ends with a b then M_2 is defined as



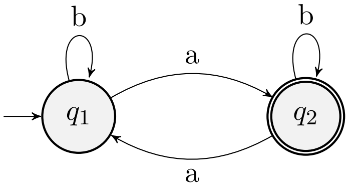
Now we combine both machines into M defined as



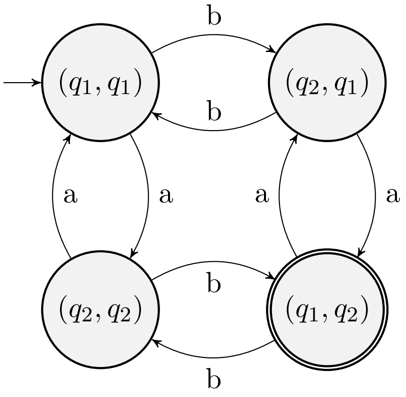
g. Let M_1 be a machine that checks if the input string has an even length then M_1 is defined as



Let also, M_2 be a machine that checks if the input string has an odd number of a's then M_2 is defined as



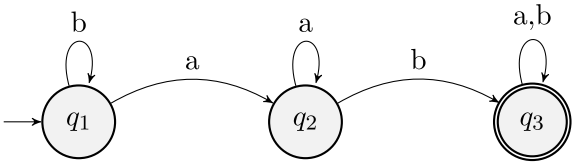
Now we combine both machines into M defined as



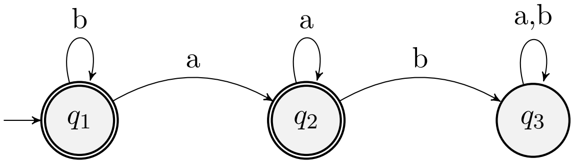
□

Proof. **1.5**

a. Let M be a machine that checks if the input string contains the substring "ab" then M is defined as

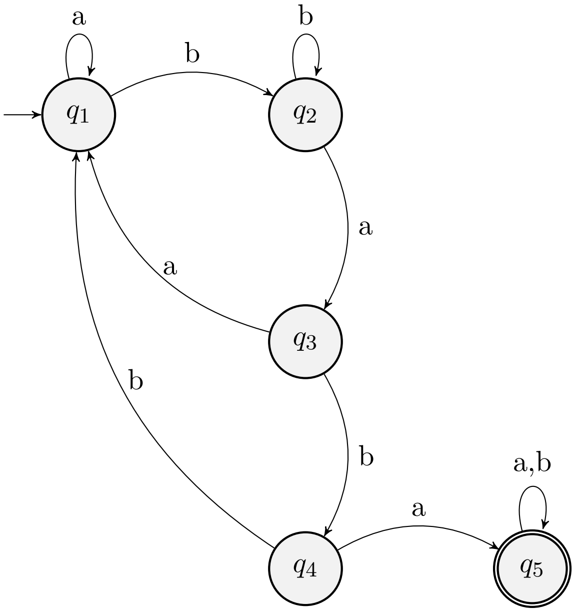


Then the complement of the machine M is

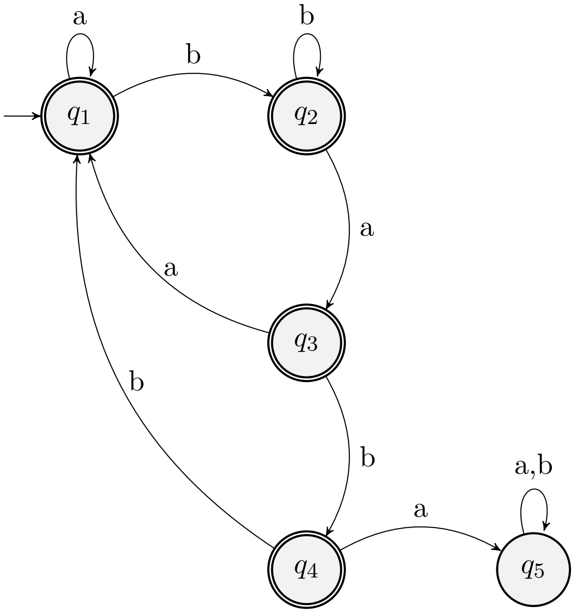


Since it will accept any string that doesn't contain the substring "ab".

b. Let M be a machine that checks if the input string contains the substring "baba" then M is defined as

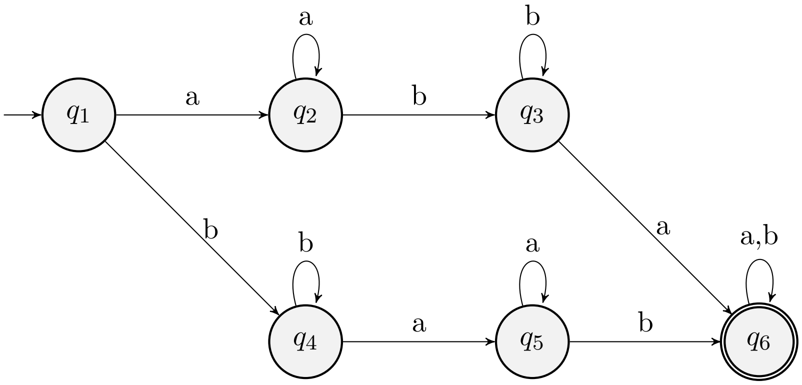


Then the complement of the machine M is

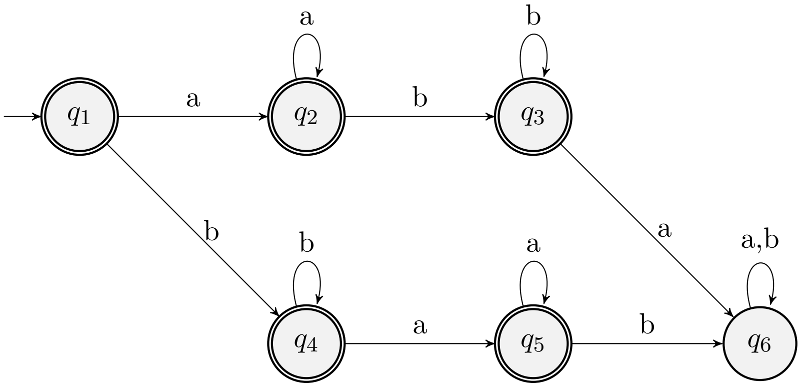


Since it will accept any string that doesn't contain the substring "baba".

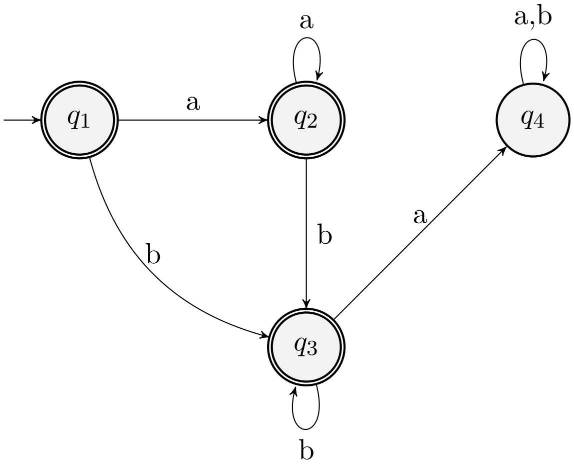
c. Let M be a machine that checks if the input string contains both the substring "ab" and "ba" then M is defined as



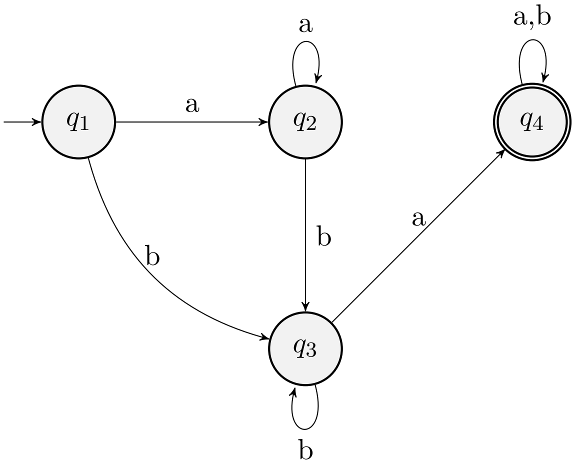
Then the complement of the machine M is



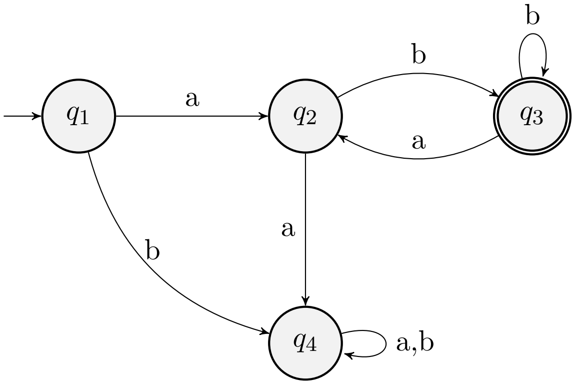
d. Let M be a machine that checks if the input string is in a^*b^* then M is defined as



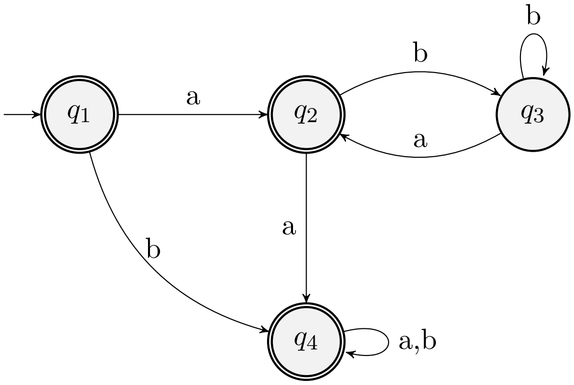
Then the complement of the machine M is



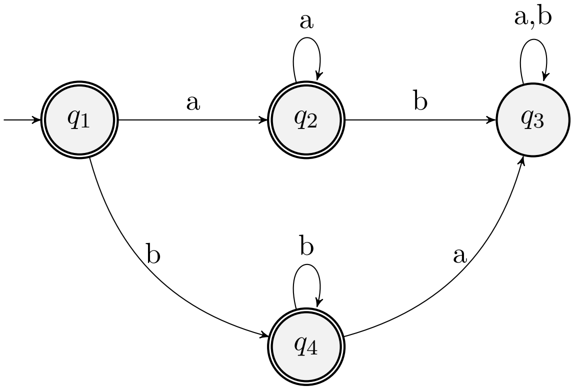
e. Let M be a machine that checks if the input string is in $(ab^+)^*$ then M is defined as



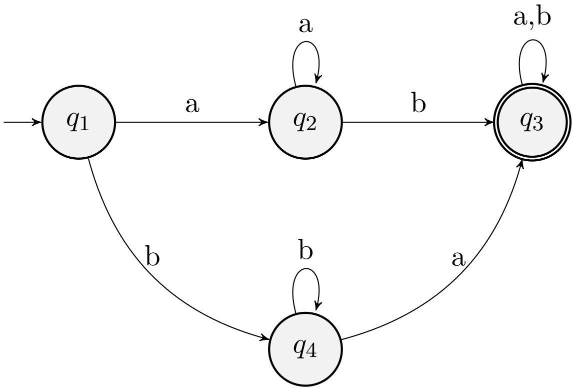
Then the complement of the machine M is



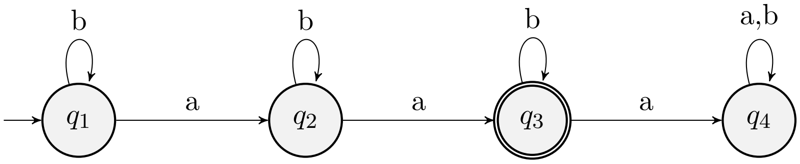
f. Let M be a machine that checks if the input string is in $a^* \cup b^*$ then M is defined as



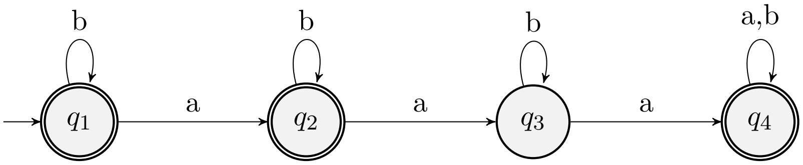
Then the complement of the machine M is



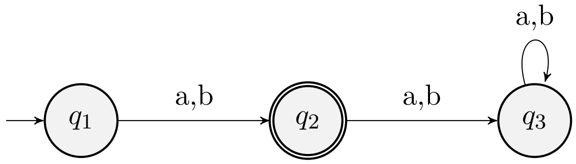
g. Let M be a machine that checks if the input string contains exactly two a's then M is defined as



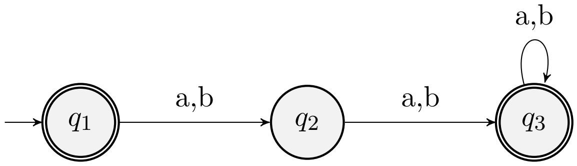
Then the complement of the machine M is



h. Let M be a machine that checks if the input string is a or b any other string is not accepted then M is defined as



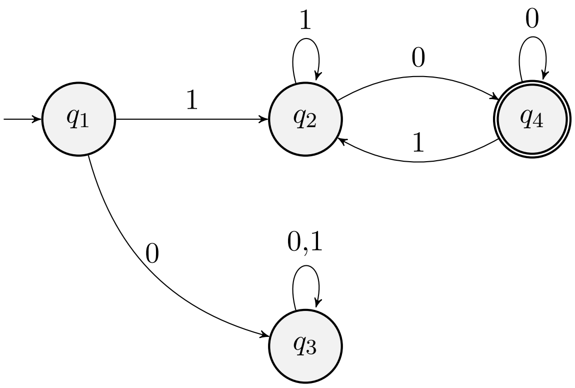
Then the complement of the machine M is



□

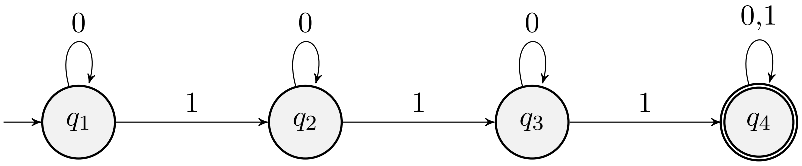
Proof. **1.6**

a. Let M be a machine that checks if the input string begins with a 1 and ends with a 0 then M is defined as

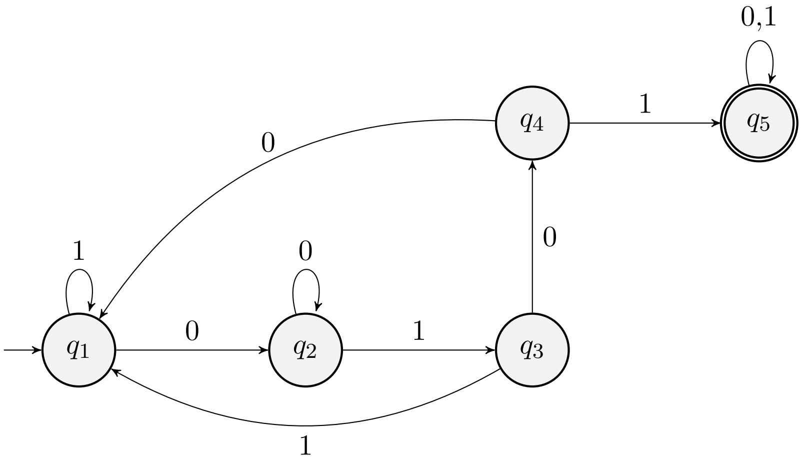


4

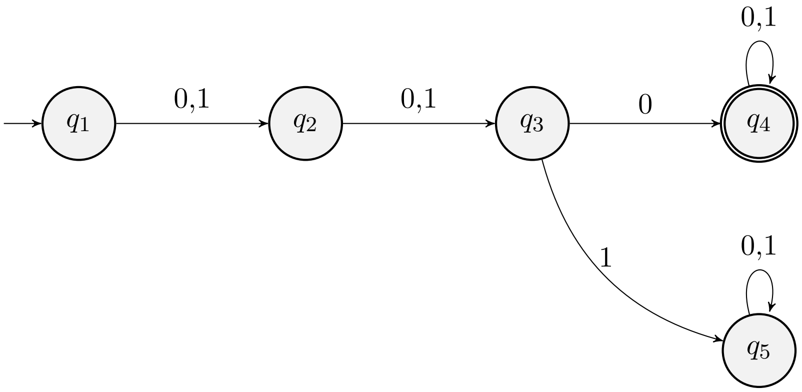
b. Let M be a machine that checks if the input string contains at least three 1s then M is defined as



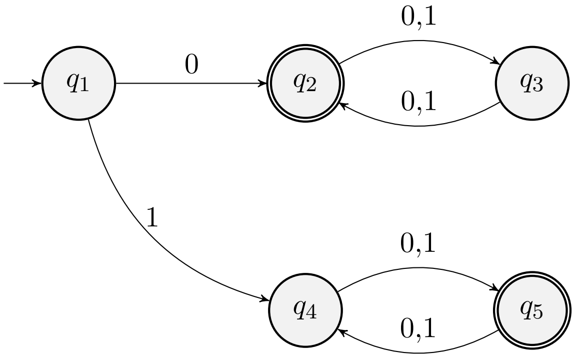
c. Let M be a machine that checks if the input string contains the substring 0101 then M is defined as



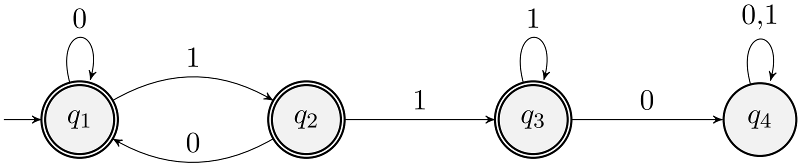
d. Let M be a machine that checks if the input string has length at least 3 and its third symbol is a 0 then M is defined as



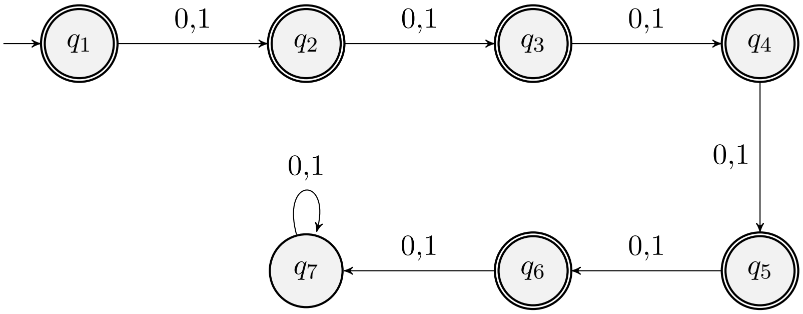
e. Let M be a machine that checks if the input string starts with 0 and has odd length or starts with 1 and has even length then M is defined as



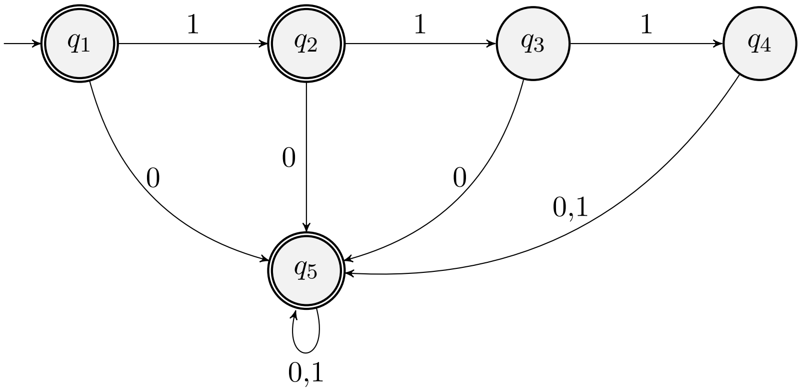
f. Let M be a machine that checks if the input string doesn't contain the substring 110 then M is defined as



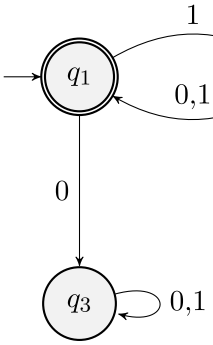
g. Let M be a machine that checks if the input string has a length at most 5 then M is defined as



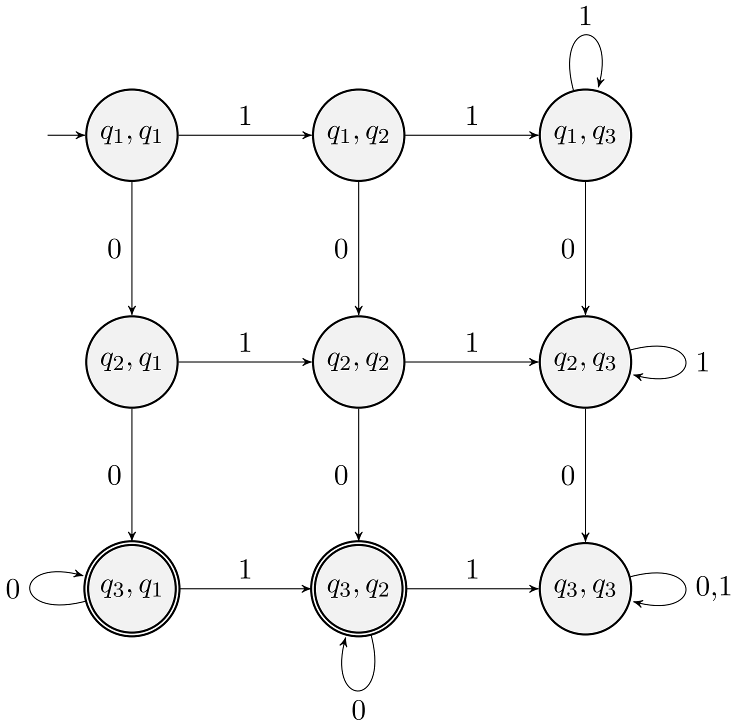
h. Let M be a machine that checks if the input string is any string except 11 or 111 then M is defined as



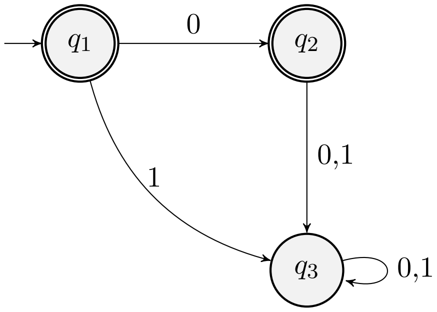
i. Let M be a machine that checks if the input string has a 1 in every odd position then M is defined as



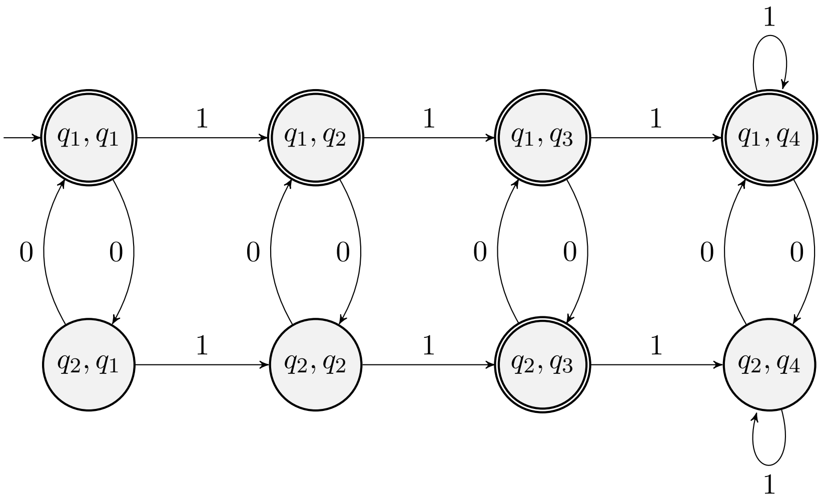
j. Let M be a machine that checks if the input string has at least two 0s and at most one 1 then M is defined as



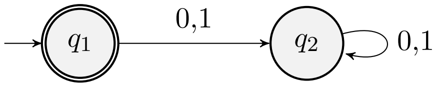
k. Let M be a machine that checks if the input string is ε or 0 then M is defined as



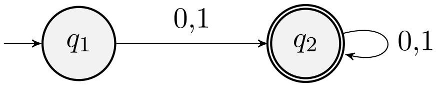
l. Let M be a machine that checks if the input string contains an even number of 0s or exactly two 1s then M is defined as



m. Let M be a machine that checks if the input string is the empty set then M is defined as



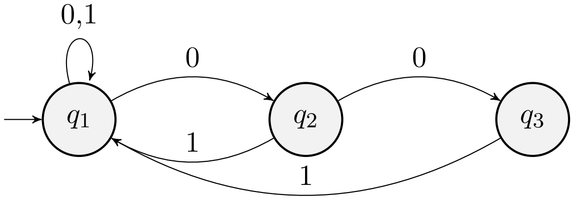
n. Let M be a machine that checks if the input string is any string except the empty set then M is defined as



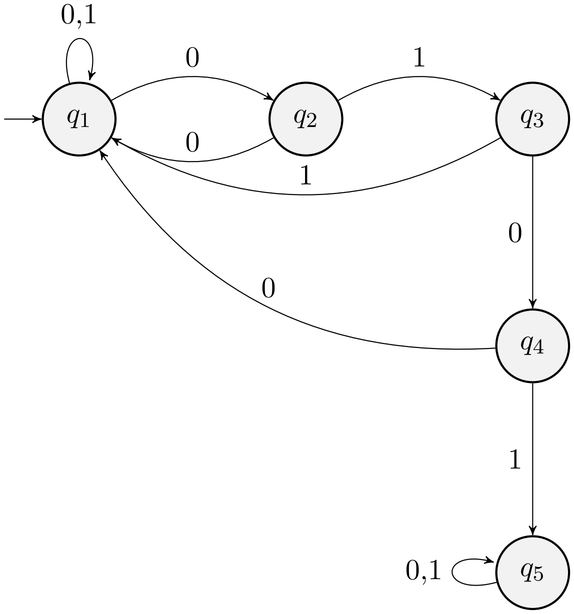
□

Proof. **1.7**

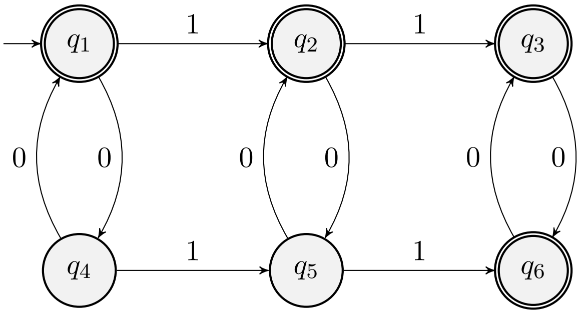
a. Let M be the NFA that recognizes the language $\{w : w \text{ ends with } 00\}$ with only three states



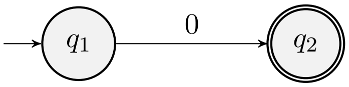
b. Let M be the NFA that recognizes the strings that have the substring 0101 then M is defined as



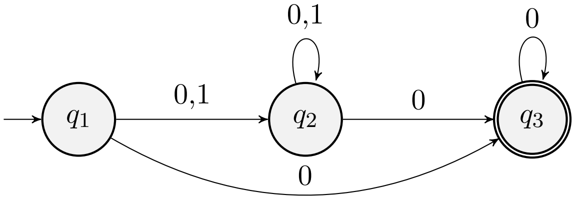
c. Let M be the NFA that recognizes the strings that have an even number of 0s or exactly two 1s then M is defined as



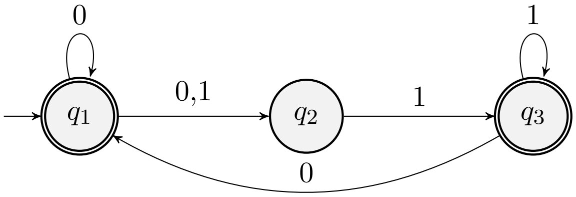
d. Let M be the NFA that recognizes the string 0 then M is defined as



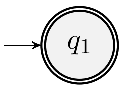
e. Let M be the NFA that recognizes the strings of the form $0^*1^*0^+$ then M is defined as



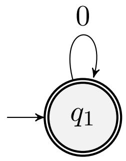
f. Let M be the NFA that recognizes the strings of the form $1^*(001^+)^*$ then M is defined as



g. Let M be the NFA that recognizes the language $\{\varepsilon\}$ then M is defined as



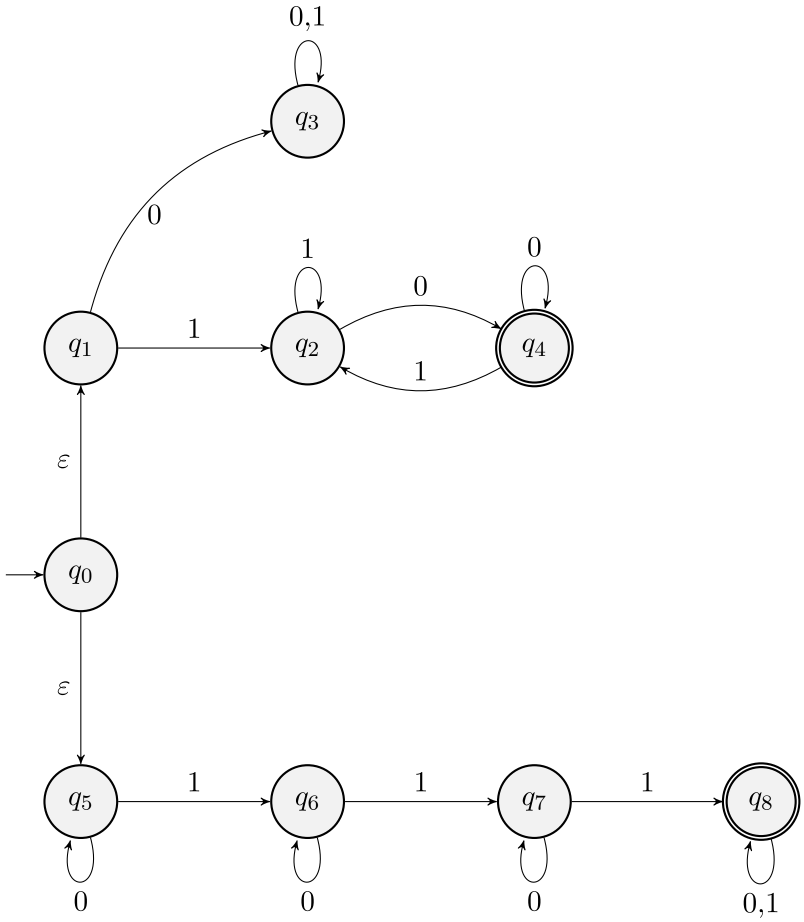
h. Let M be the NFA that recognizes the language 0^* then M is defined as



□

Proof. **1.8**

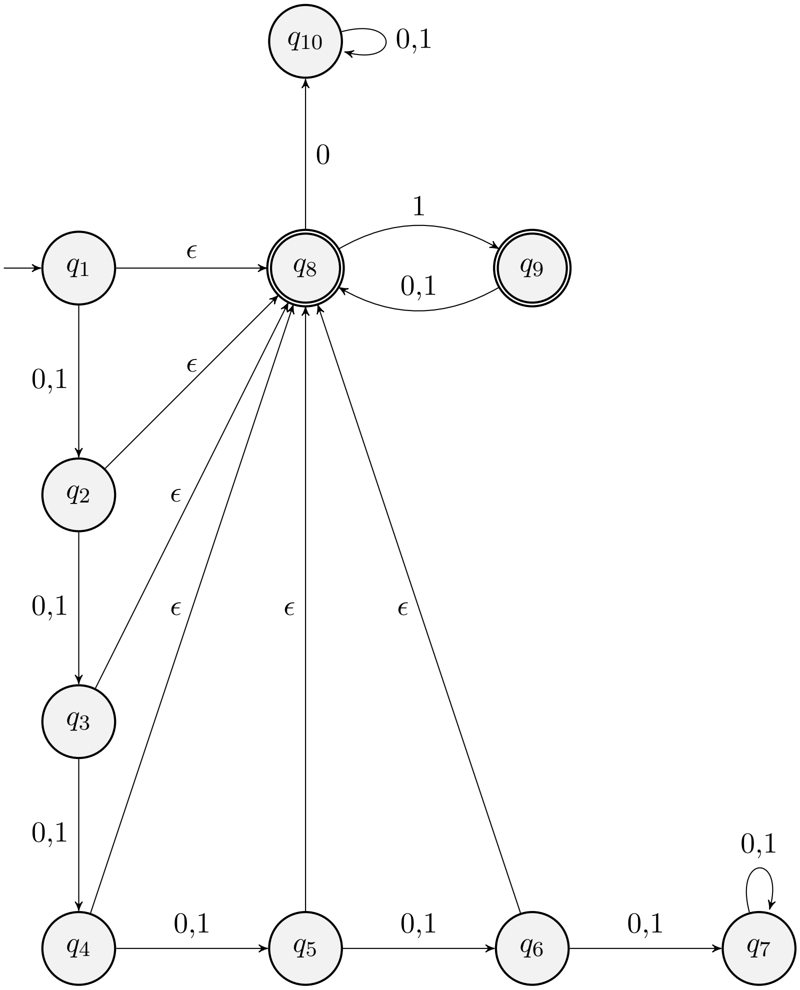
a. Let M be a machine that recognizes both languages described in Exercises 1.6a and 1.6b then



□

Proof. **1.9**

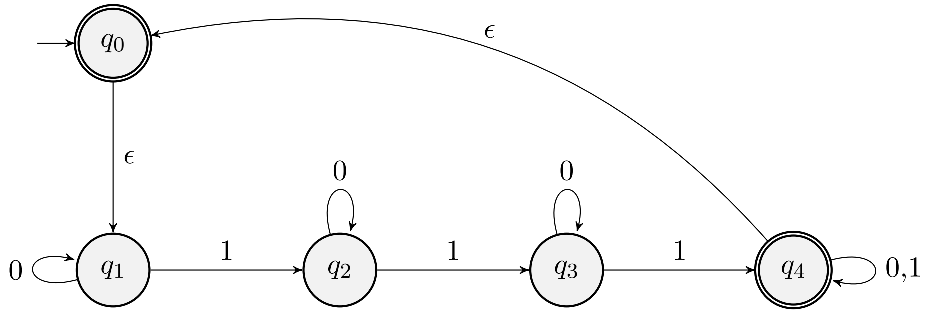
a. Let M be a machine that recognizes the concatenation of the languages described in Exercises 1.6g and 1.6i then



□

Proof. **1.10**

a. Let M be a machine that recognizes the star of the language described in Exercise 1.6b then



□

Proof. **1.11** Given that every NFA has an equivalent DFA, and we can convert a DFA into a GNFA by adding an ϵ arrow from every accept state to one accept state, then we can convert every NFA to an equivalent one that has a single accept state. □

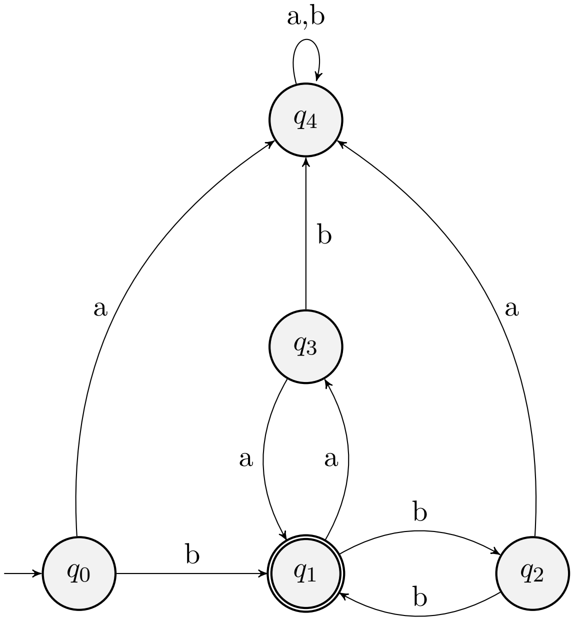
Proof. **1.12** Let

$$D = \{w|w \text{ contains an even number of a's and an odd number of b's and does not contain the substring } ab\}$$

Then we can write the regular expression that generates D as follows

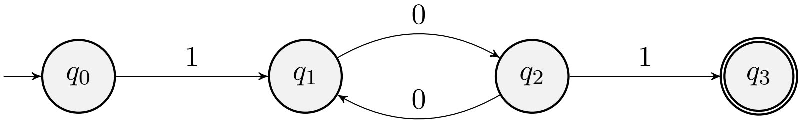
$$(bb)^*b(aa)^*$$

Finally, the DFA that recognizes D is given by

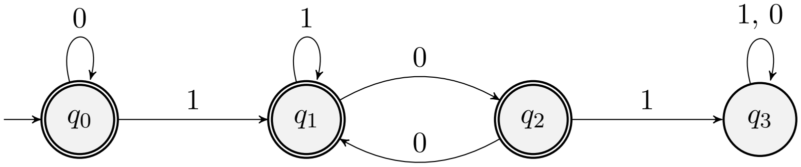


□

Proof. **1.13** Let $\overline{F}$ be the language that recognizes all strings that contain a pair of 1s separated by an odd number of 0s then the NFA with four states for this language is given by



Then F the complement of $\overline{F}$ is the language that recognizes strings that do not contain a pair of 1s that are separated by an odd number of symbols. A DFA for this language is shown below

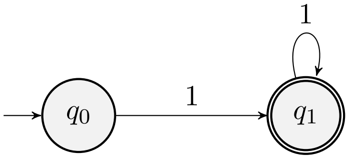


□

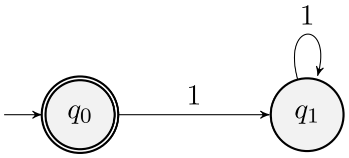
Proof. **1.14**

a. If M is a DFA that recognizes the language B, then there are finitely many states that are accept states and hence the rest are non-accept states. Taking the non-accept states as accept states for a new DFA M' we would accept all the strings that M doesn't accept. Hence M' recognizes the complement of B i.e. $\overline{B}$. Therefore given that there is always a DFA M' that accepts $\overline{B}$ then the class of regular languages is closed under complement.

b. Let M be an NFA that recognizes the language C described by the regular expression 1^+ over $\{0, 1\}$ then M is given by



Swapping the accept and nonaccept states in M gives us



But this NFA accepts only the empty string ϵ which is not the complement of C over $\{0, 1\}$.

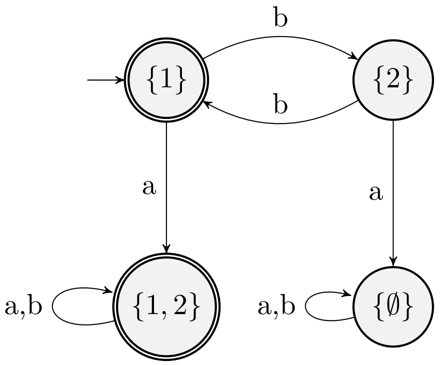
If a language A is recognized by an NFA then it's also recognized by a DFA, since there is always a DFA equivalent to the NFA then the the language A is closed under complement. This implies that the class of languages recognized by NFAs is closed under complement.

But from the example we see that they are not always obtained by swapping the accept and non-accept states.

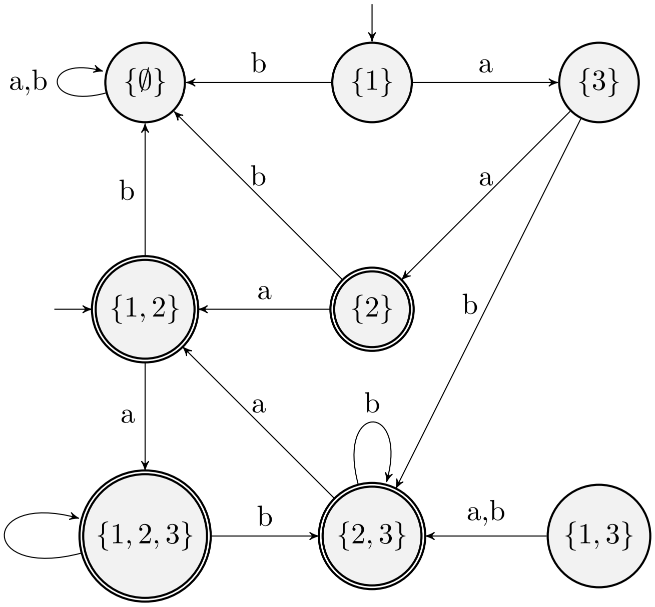
□

Proof. **1.16** Following the construction given in Theorem 1.39 we transform the given NFAs into the following DFAs

(a) The first one give us



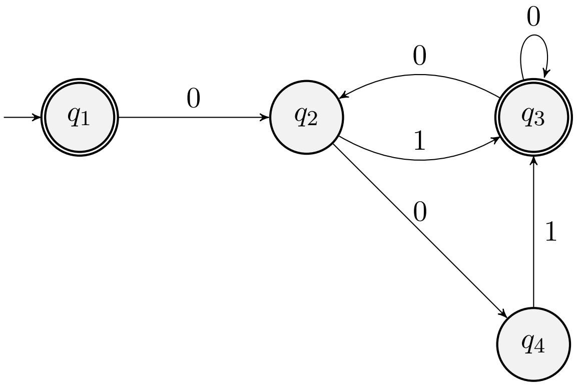
b. For this DFA we need $2^3 = 8$ states, hence



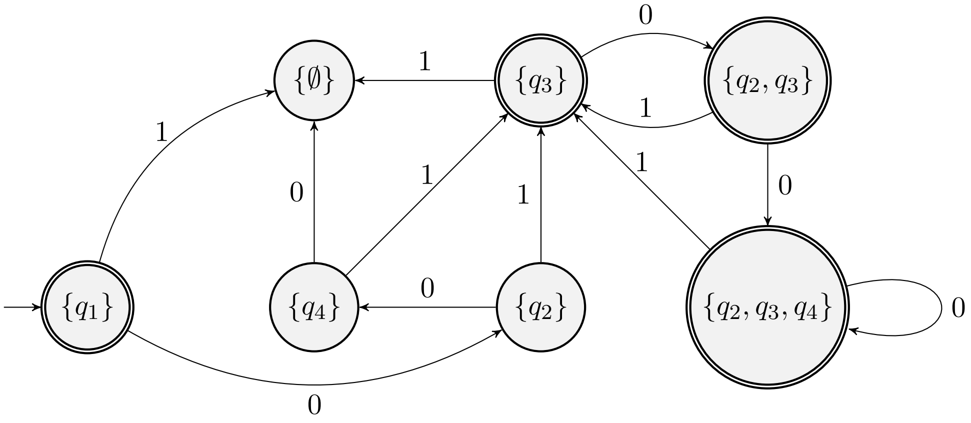
□

Proof. **1.17**

(a) The NFA that recognizes the language $(01 \cup 001 \cup 010)^*$ is



(b) The equivalent DFA is then



□

Proof. **1.18**

a. The regular expression generating the language

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

is

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

b. The regular expression generating the language

$$\{w|w \text{ contains at least three 1s}\}$$

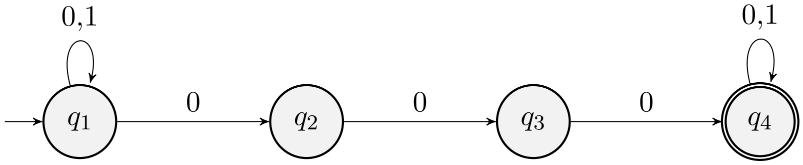
is

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

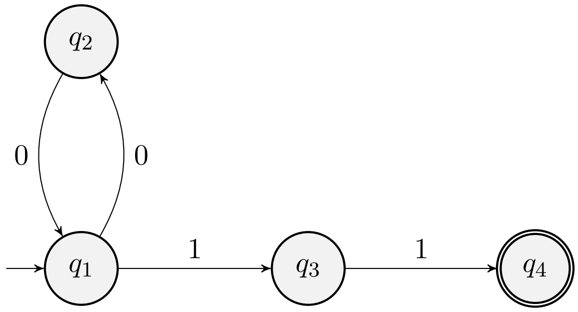
□

Proof. **1.19**

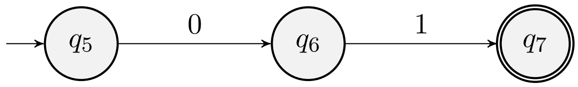
a. The NFA recognizing the regular expression $(0\cup1)^*000(0\cup1)^*$ is given by



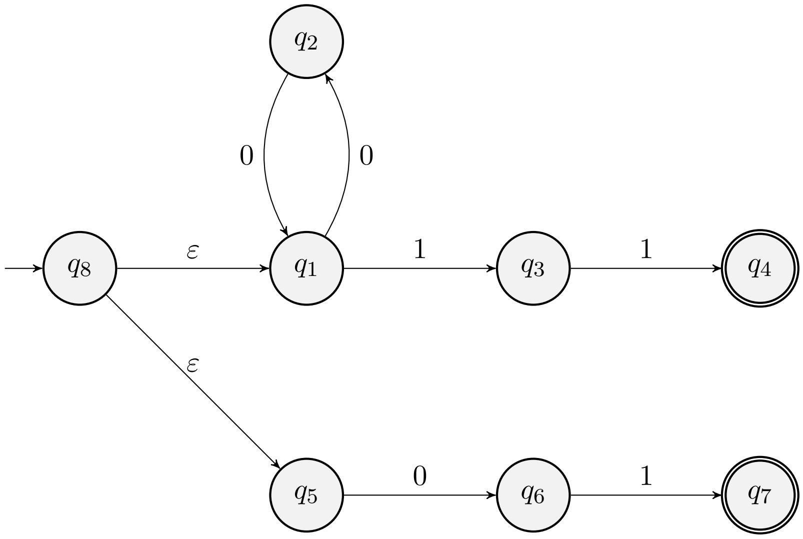
b. The NFA recognizing the regular expression $(00)^*(11)$ is given by



And the NFA recognizing the regular expression 01 is



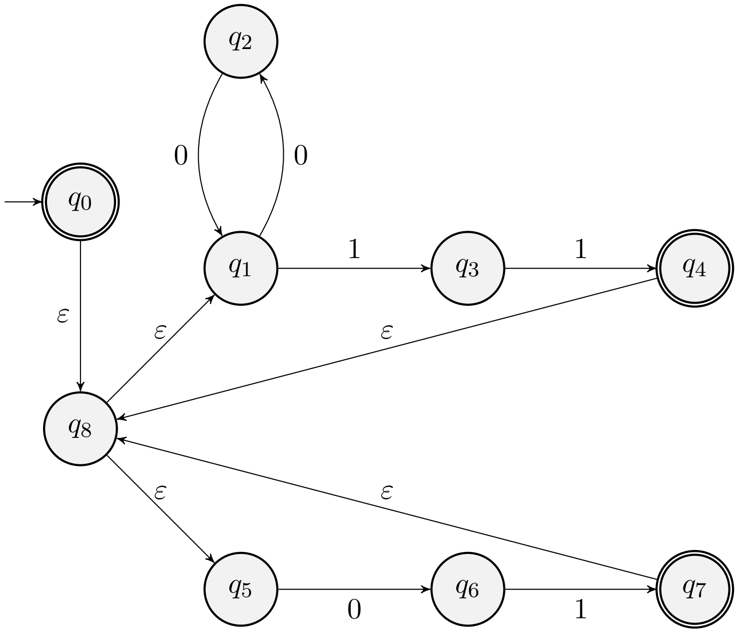
Given that regular languages are closed under union operation we can build an NFA that recognizes the regular expression $((00)^*(11))\cup01$ as follows



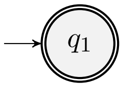
Finally, given that regular languages are closed under star operation we can build an NFA that recognizes the regular expression

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

as follows



c. The NFA recognizing the regular expression $\emptyset^*$ is given by



□

Proof. **1.20**

a. a^*b^*:
Member strings are a and b and not member strings are aba and ba.

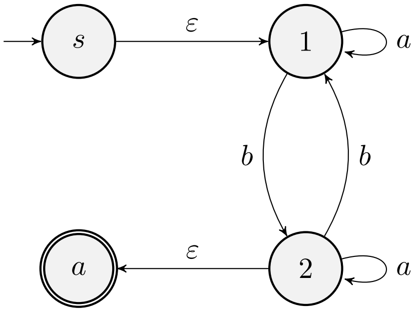
b. $a(ba)^*b$:
Member strings are ab and $abab$ and not member strings are abb and aa.

g. $(\varepsilon \cup a)b$:
Member strings are b and ab and not member strings are bb and aba.

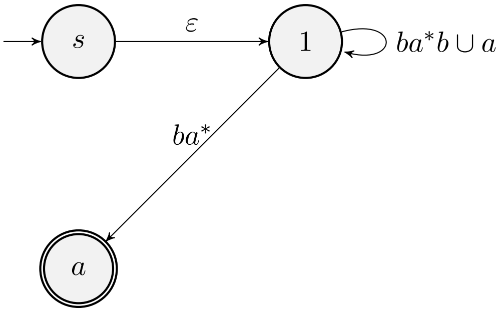
□

Proof. **1.21**

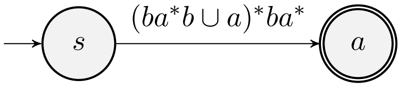
(a) Let us begin by creating the equivalent GNFA as follows



Now we remove state 2 as Lemma 1.60 explains to get

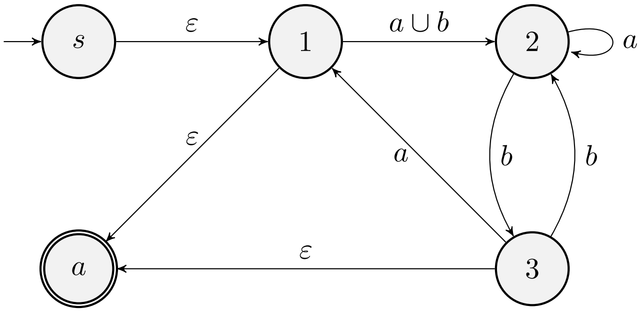


Finally, removing state 1 we get that

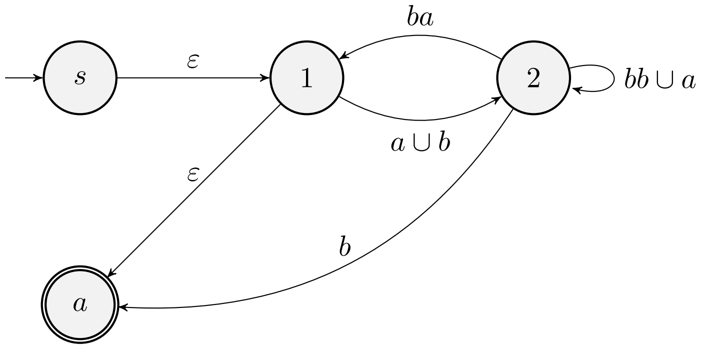


Therefore the regular expression accepting the language the DFA accepts is $(ba^*b \cup a)^*ba^*$.

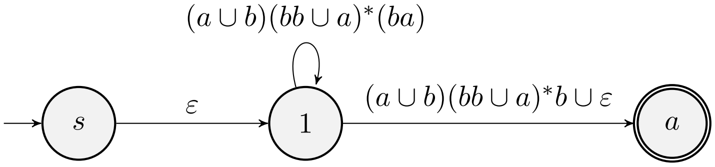
(b) As before we convert the DFA to a GNFA as follows



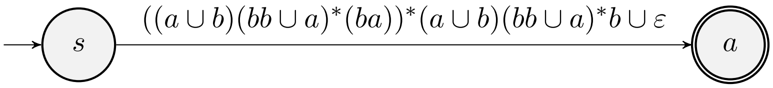
Now we remove state 3



Then we remove state 2


$(a \cup b)(bb \cup a)^*(ba)$
s
ε
1
$(a \cup b)(bb \cup a)^*b \cup \varepsilon$
a

And finally, we remove state 1 to get


s
$((a \cup b)(bb \cup a)^*(ba))^*(a \cup b)(bb \cup a)^*b \cup \varepsilon$
a

Therefore the regular expression accepting the language the DFA accepts is $((a \cup b)(bb \cup a)^*(ba))^*(a \cup b)(bb \cup a)^*b \cup \varepsilon$.

$\square$

Proof. **1.24**

a. On input 011, T_1 enter the sequence of states q_1, q_1, q_1, q_1 and outputs 000.

b. On input 211, T_1 enter the sequence of states q_1, q_2, q_2, q_2 and outputs 111.

c. On input 121, T_1 enter the sequence of states q_1, q_1, q_2, q_2 and outputs 011.

d. On input 0202, T_1 enter the sequence of states q_1, q_1, q_2, q_1, q_2 and outputs 0101.

e. On input b, T_2 enter the sequence of states q_1, q_3 and outputs 1.

f. On input $bbab$, T_2 enter the sequence of states q_1, q_3, q_2, q_3, q_2 and outputs 1111.

g. On input $bbbbbb$, T_2 enter the sequence of states $q_1, q_3, q_2, q_1, q_3, q_2, q_1$ and outputs 110110.

h. On input ε, T_2 enters the state q_1 and outputs ε.

□

Proof. **1.25** A finite state transducer (FST) is a 5-tuple $(Q, \Sigma, \delta, q_0, \Gamma)$, where

1. Q is a finite set called the states

2. Σ is a finite set called the input alphabet

3. $\delta : Q \times \Sigma \to Q \times \Gamma$ is the transition function

4. $q_0 \in Q$ is the start state

5. Γ is a finite set called the output alphabet

□

Proof. **1.26**

Machine T_1 can be described formally as

$$T_1 = (\{q_1, q_2\}, \{0, 1, 2\}, \delta, q_1, \{0, 1\})$$

Where $\delta : Q \times \Sigma \to Q \times \Gamma$ is given by

$$\begin{aligned}(q_1, 0) &\to (q_1, 0)\\ (q_1, 1) &\to (q_1, 0)\\ (q_1, 2) &\to (q_2, 1)\\ (q_2, 0) &\to (q_1, 0)\\ (q_2, 1) &\to (q_2, 1)\\ (q_2, 2) &\to (q_2, 1)\end{aligned}$$

Machine T_2 can be described formally as

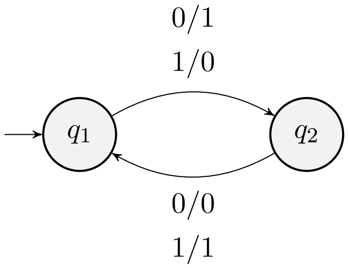
$$T_2 = (\{q_1, q_2, q_3\}, \{a, b\}, \delta, q_1, \{0, 1\})$$

Where $\delta : Q \times \Sigma \to Q \times \Gamma$ is given by

$$\begin{aligned}(q_1, a) &\to (q_2, 1)\\ (q_1, b) &\to (q_3, 1)\\ (q_2, a) &\to (q_3, 1)\\ (q_2, b) &\to (q_1, 0)\\ (q_3, a) &\to (q_1, 0)\\ (q_3, b) &\to (q_2, 1)\end{aligned}$$

□

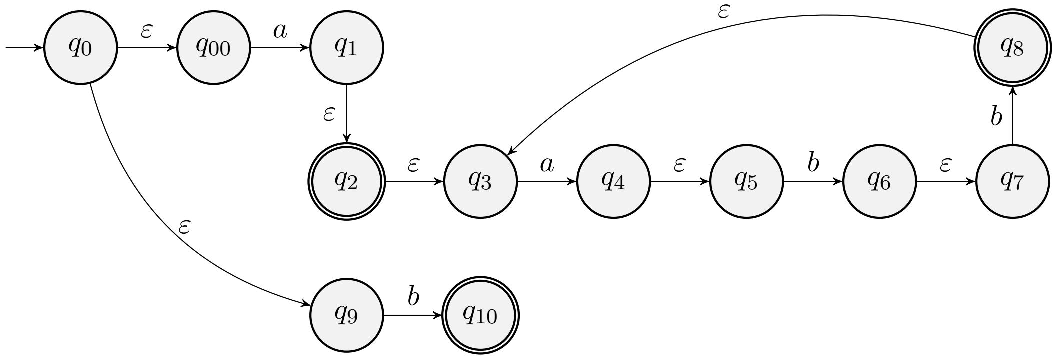
Proof. **1.27** The state diagram for the machine that outputs the string identical to the input string on the even positions but inverted on the odd positions is



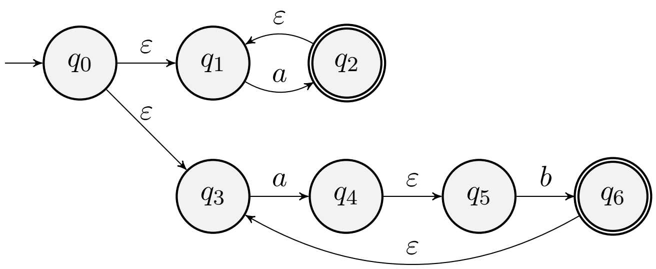
□

Proof. **1.28** The NFAs for each regular expression is shown below

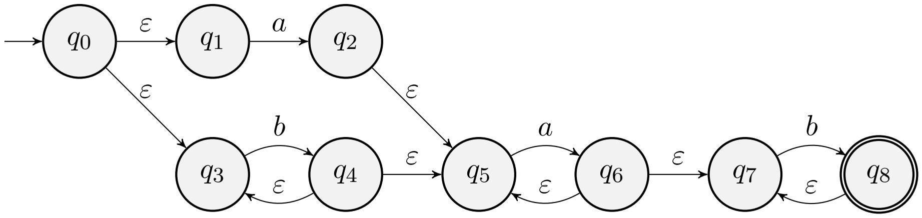
a. $a(abb)^* \cup b$



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



c. $(a \cup b^+)a^+b^+$



□