Automata Theory A-1.2

17.

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, $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

a. $\{w \mid w$ has at least three a's and at least two b's$\}$

Ab. $\{w \mid w$ has exactly two a's and at least two b's$\}$

c. $\{w \mid w$ has an even number of a's and one or two b's$\}$

Ad. $\{w \mid w$ has an even number of a's and each a is followed by at least one b$\}$

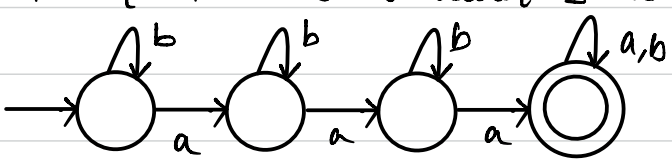
e. $\{w \mid w$ starts with an a and has at most one b$\}$

f. $\{w \mid w$ has an odd number of a's and ends with a b$\}$

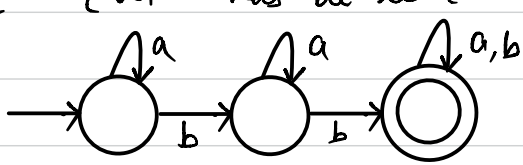
g. $\{w \mid w$ has even length and an odd number of a's$\}$

a. Let us divide it into 2 parts. M_1 that recognizes language L_1 and M_2 the recognizes language L_2, where M_1, M_2 are DFA's.

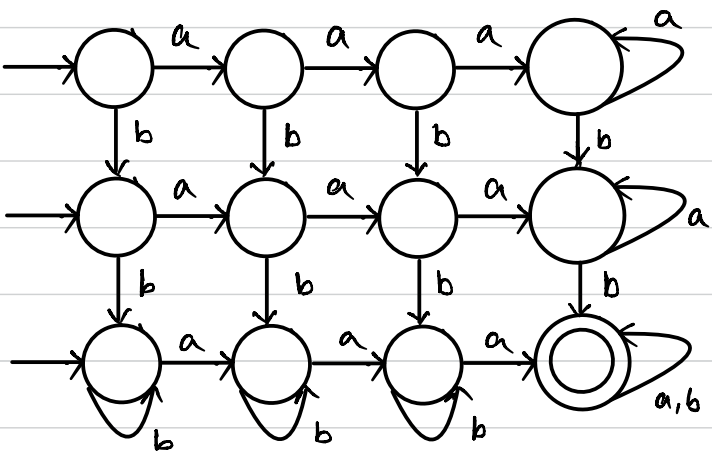
$L_1 = \{w \mid w$ has at least 3 a's$\}$



$L_2 = \{w \mid w$ has at least 2 b's$\}$



We need to construct DFA M that accepts the language $L_1 L_2$

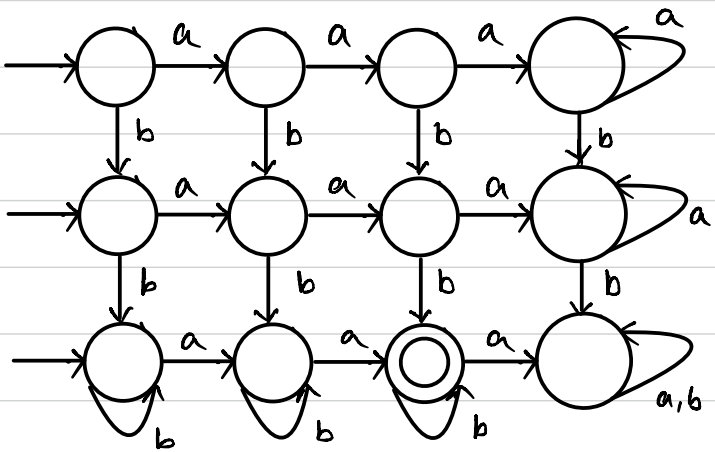


b. We will follow a similar procedure from the previous question.

$L_1 = \{w \mid w$ has exactly 2 a's$\}$

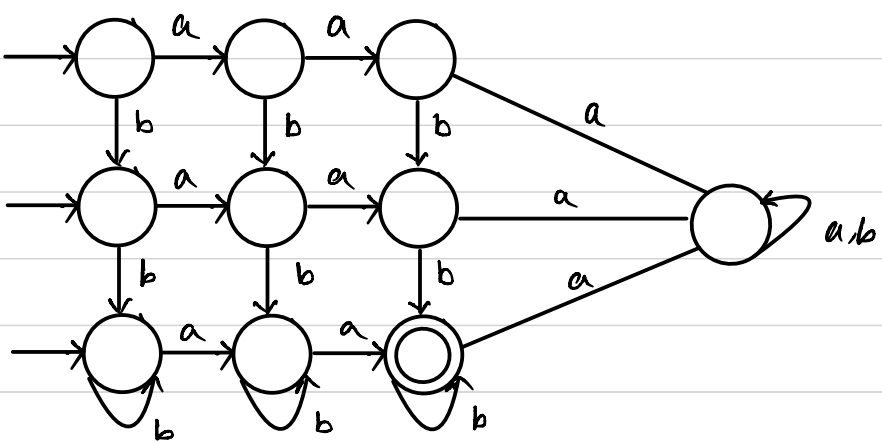
$L_2 = \{w \mid w$ has atleast 2 b's$\}$

We need to find a DFA that recognizes the language $L_1 L_2$



We can leave it as it is or simply it since the rightmost states lead to the same state.

Upon simplifying:



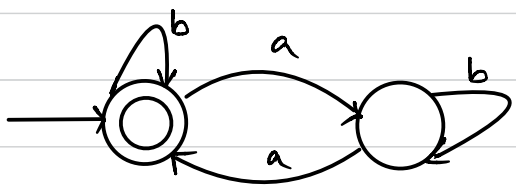
c. Following the same strategy:

$L_1 = \{w \mid w$ has even a's$\}$

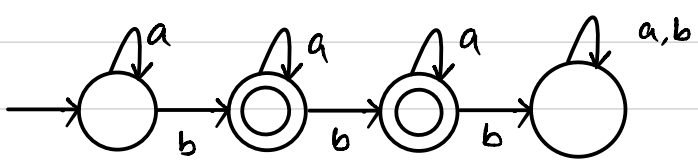
$L_2 = \{w \mid w$ has one or two b's$\}$

(we could have divided L_2 into two languages too but since it is simple enough we will use it as is).

L_1

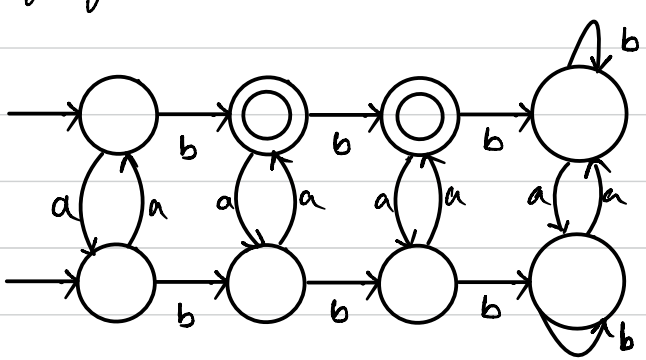


L_2

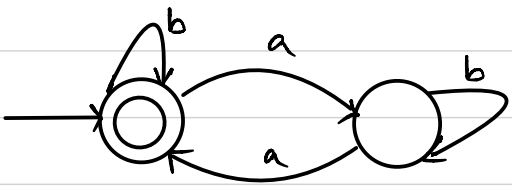


The language it recognizes is L_1L_2

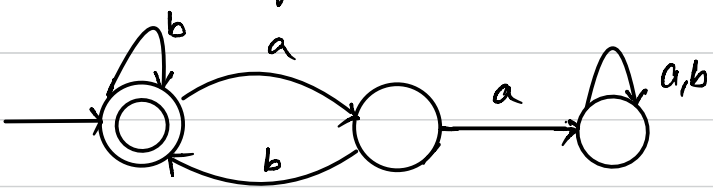
L_1L_2



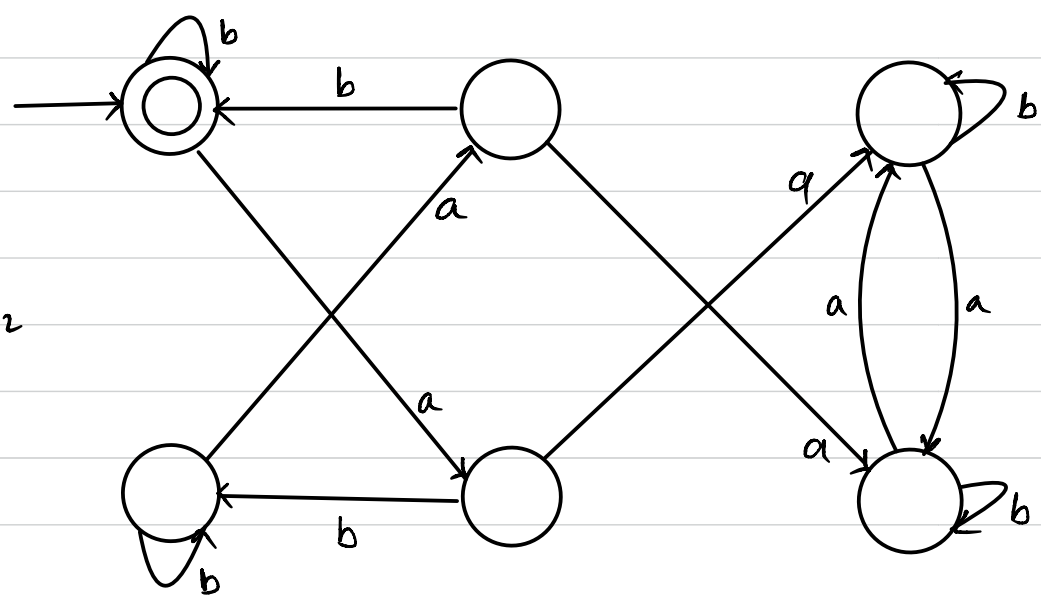
d. $L_1 = \{w \mid w$ has even a's$\}$



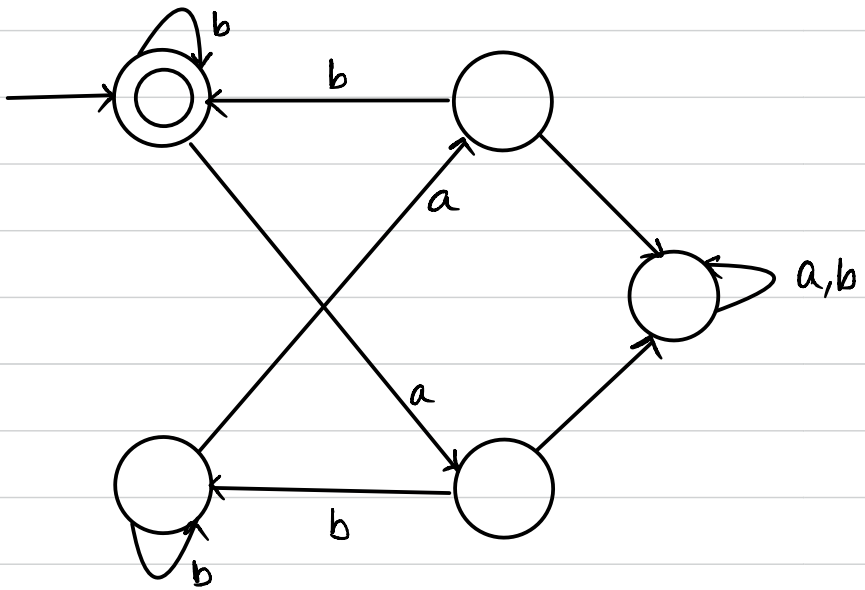
$L_2 = \{w \mid w$ has each a followed by at least one b$\}$



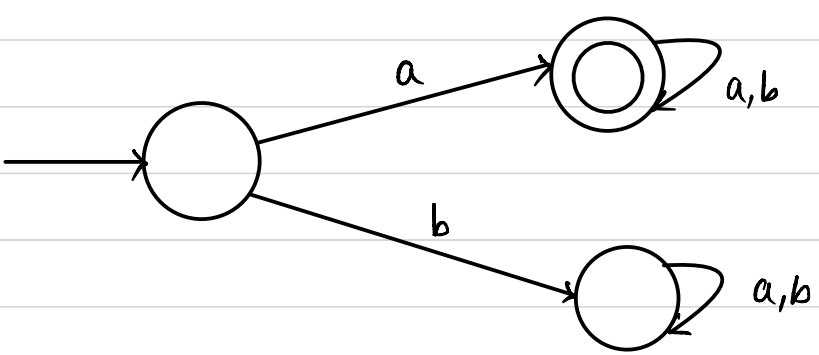
$L_1 L_2$



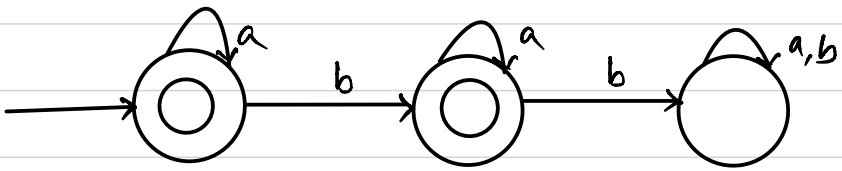
We can simplify $L_1 L_2$ that our DFA recognizes.



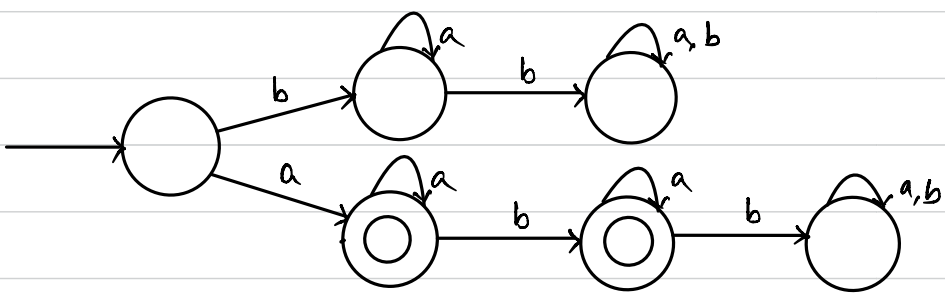
c. $L_1 = \{w \mid w \text{ starts with } a\}$



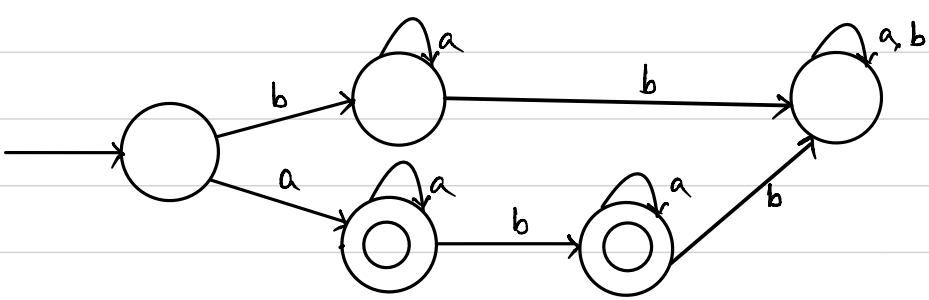
$L_2 = \{w \mid w$ has at most one b$\}$



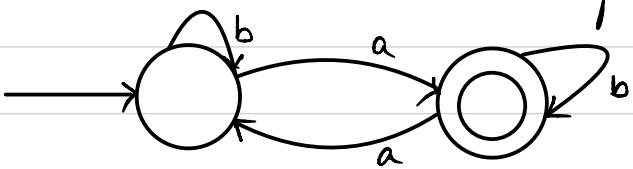
$L_1 L_2$



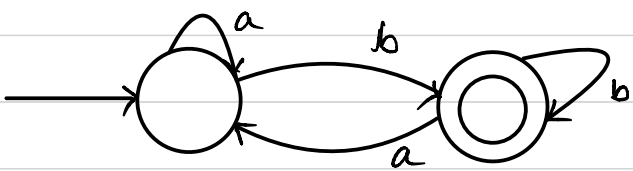
The DFA recognizes the simplified $L_1 L_2$:



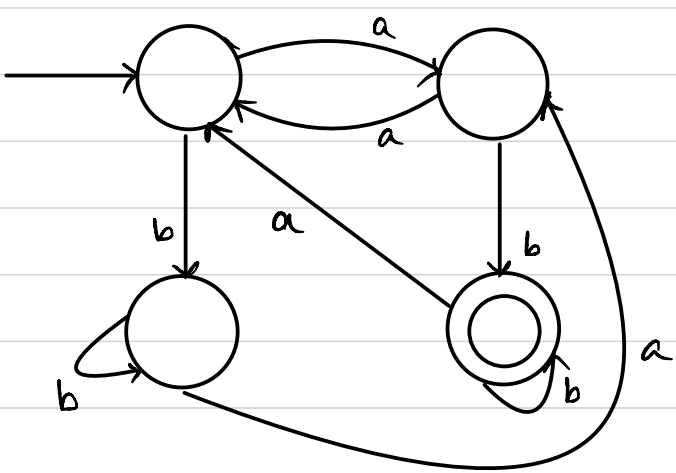
f. $L_1 = \{w \mid w$ has odd number of a's $\}$



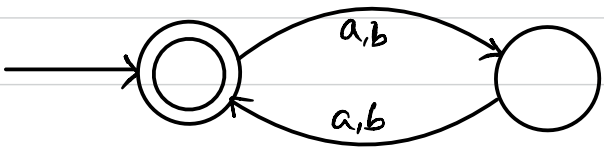
$L_2 = \{w \mid w$ ends with b $\}$



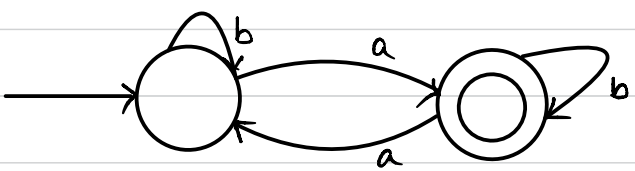
The required DFA recognizes the language $L_1 L_2$



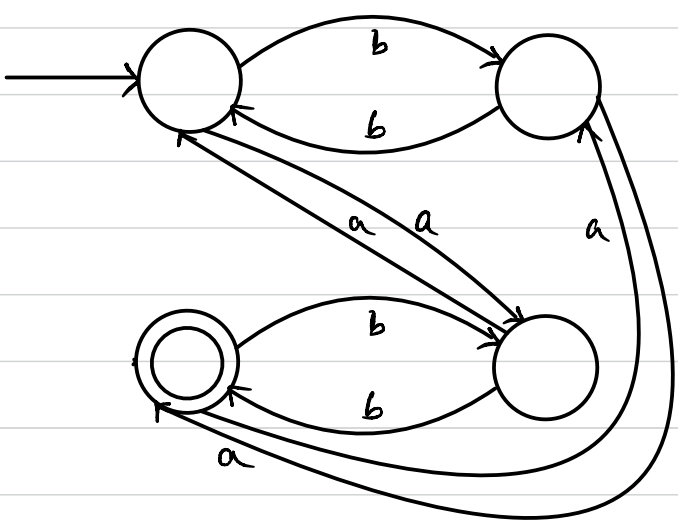
g. L_1 = {w | w has even length}



L_2 = {w | w has odd a's}



The required DFA recognizes the language L_1L_2

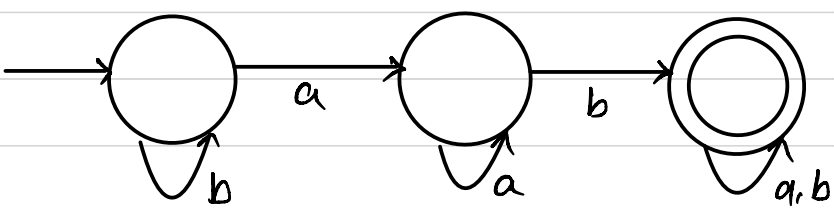


18.

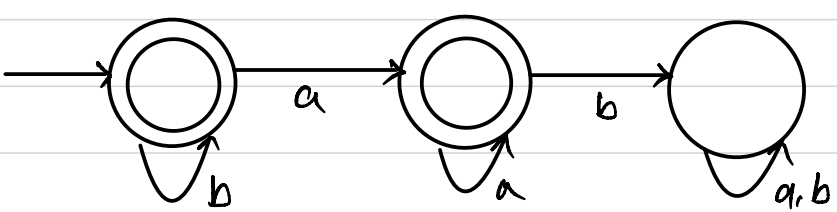
1.5 Each of the following languages is the complement of a simpler language. In each part, construct a DFA for the simpler language, then use it to give the state diagram of a DFA for the language given. In all parts, $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

- **[A]a.** $\{w \mid w$ does not contain the substring $\mathtt{ab}\}$
- **[A]b.** $\{w \mid w$ does not contain the substring $\mathtt{baba}\}$
- **c.** $\{w \mid w$ contains neither the substrings $\mathtt{ab}$ nor $\mathtt{ba}\}$
- **d.** $\{w \mid w$ is any string not in $\mathtt{a}^*\mathtt{b}^*\}$
- **e.** $\{w \mid w$ is any string not in $(\mathtt{ab}^+)^*\}$
- **f.** $\{w \mid w$ is any string not in $\mathtt{a}^* \cup \mathtt{b}^*\}$
- **g.** $\{w \mid w$ is any string that doesn't contain exactly two $\mathtt{a}$'s$\}$
- **h.** $\{w \mid w$ is any string except $\mathtt{a}$ and $\mathtt{b}\}$

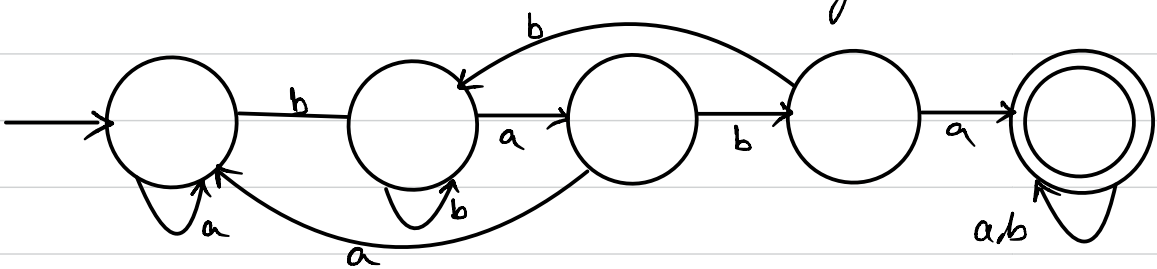
a. let $L = \{w \mid w$ contains substring $ab\}$



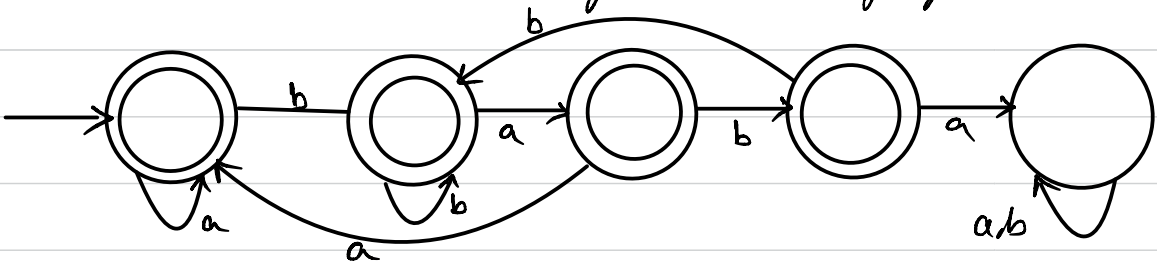
The solution DFA recognized language $\bar{L}$



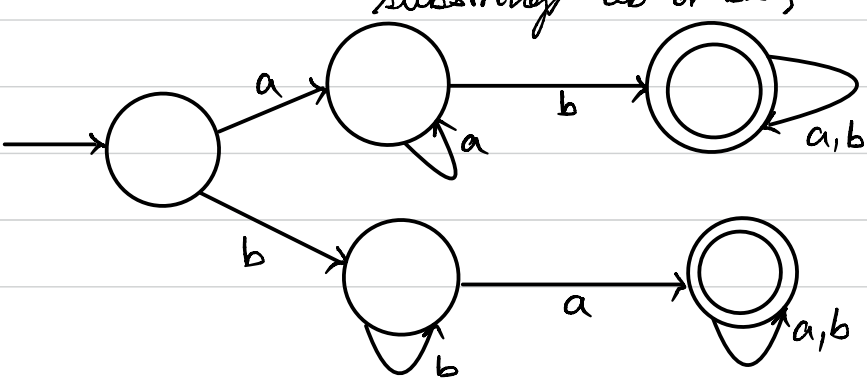
b. $L = \{w \mid w$ contains substring baba$\}$



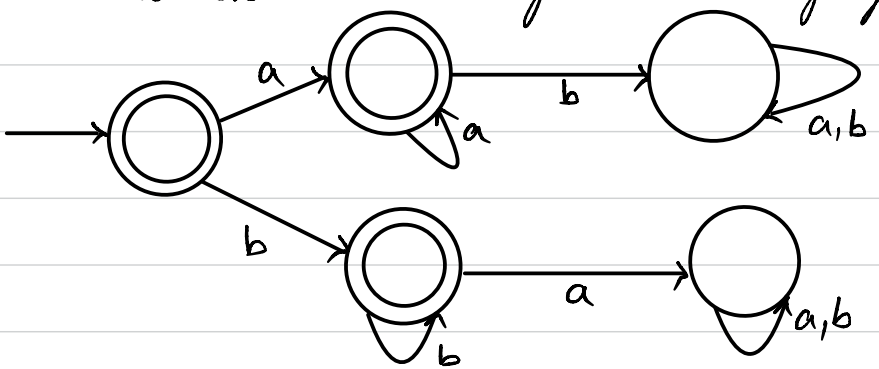
The solution DFA recognized language $\bar{L}$



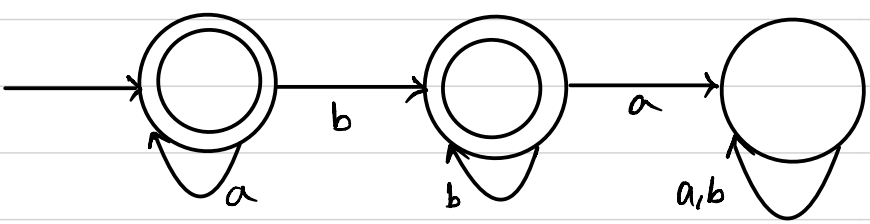
C. $L = \{w \mid w$ contains either substrings ab or ba$\}$



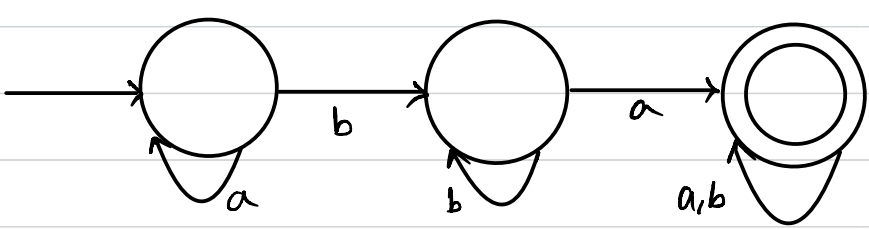
The solution DFA recognized language $\overline{L}$



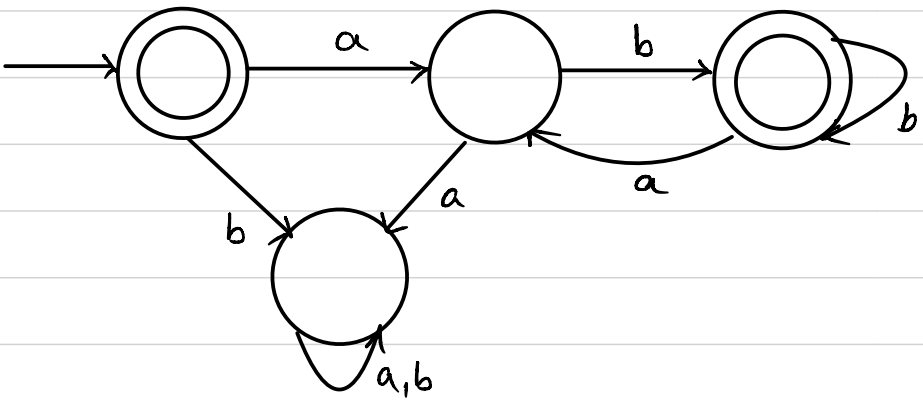
d. $L = \{w \mid w$ is any string in $a^*b^*\}$



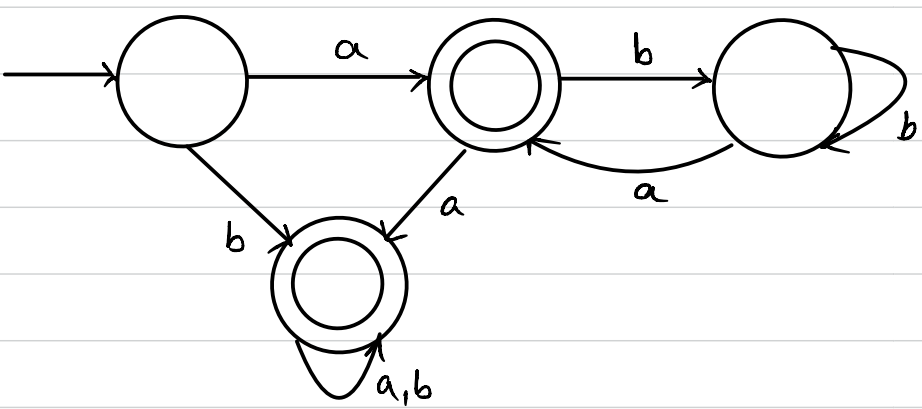
The solution DFA recognized language $\overline{L}$



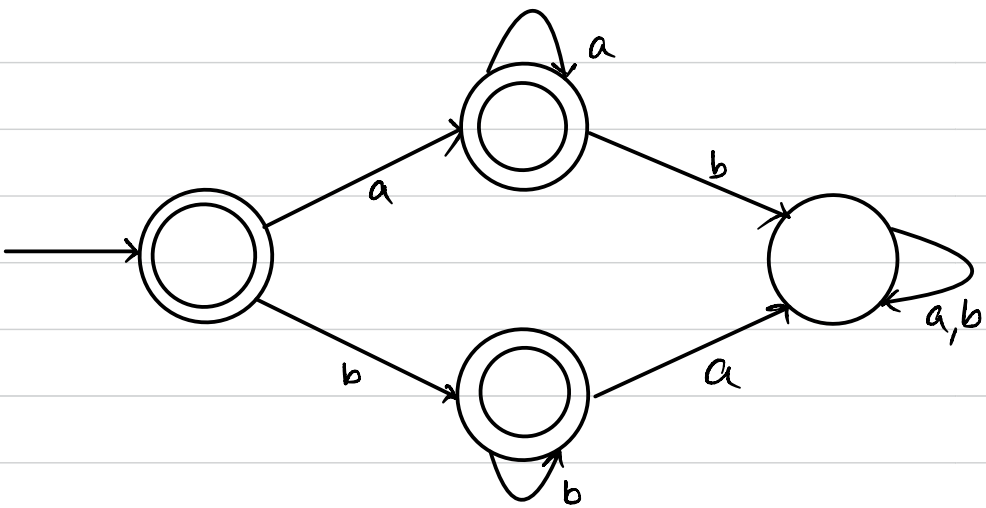
c. $L = \{w \mid w \text{ is any string in } (ab^+)^*\}$



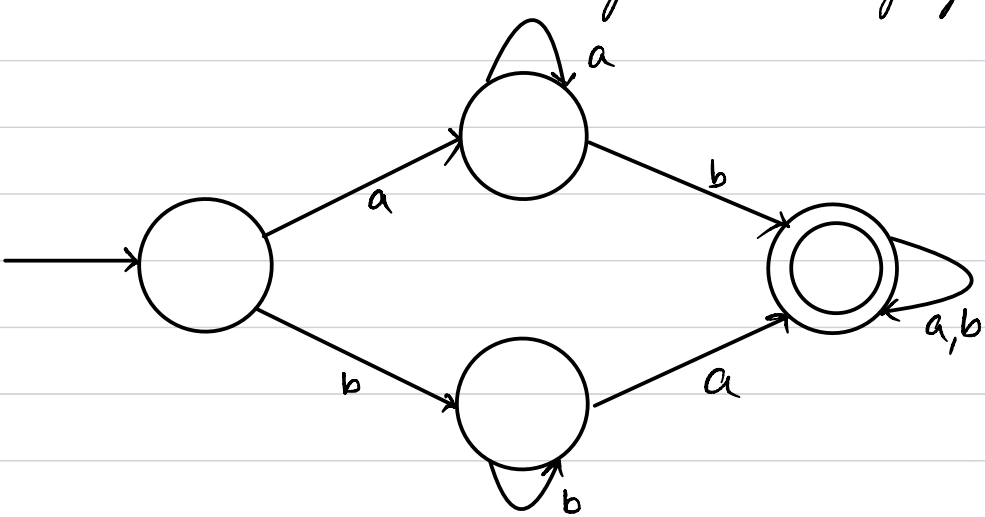
The solution DFA recognized language $\overline{L}$



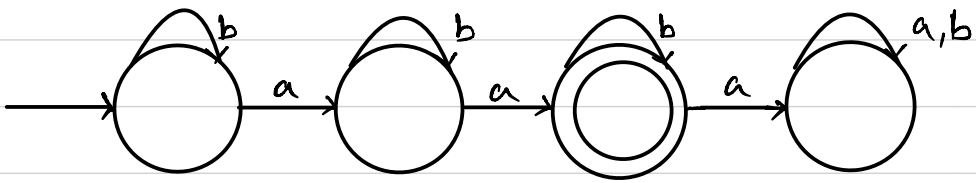
f. $L = \{w \mid w \text{ is any string in } a^* \cup b^*\}$



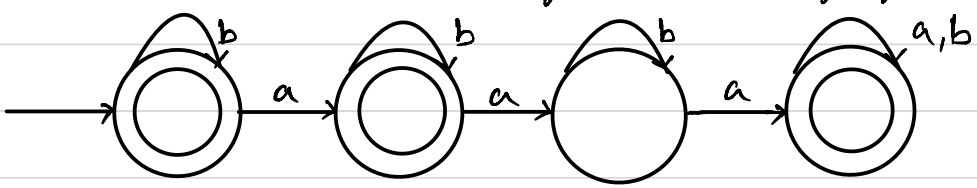
The solution DFA recognized language $\overline{L}$



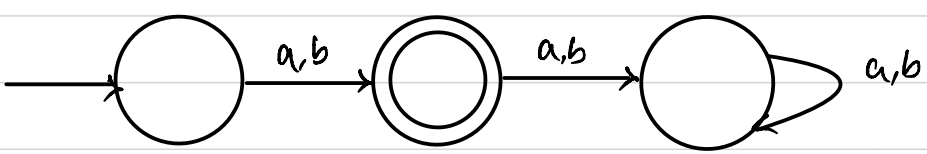
g. $L = \{w \mid w$ contains exactly two a's$\}$



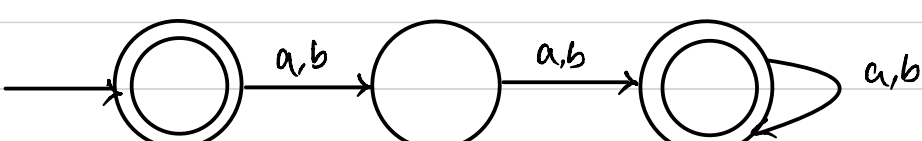
The solution DFA recognized language $\overline{L}$



h. $L = \{w \mid w$ is a and b$\}$



The solution DFA recognized language $\overline{L}$

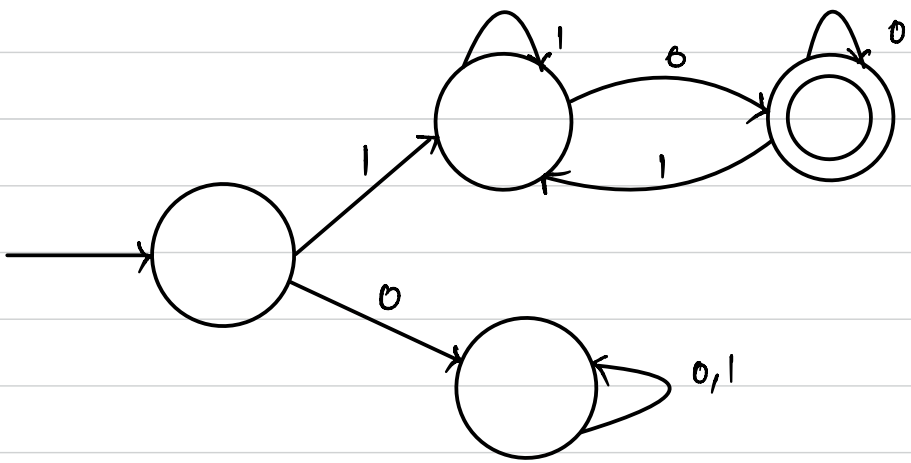


19.

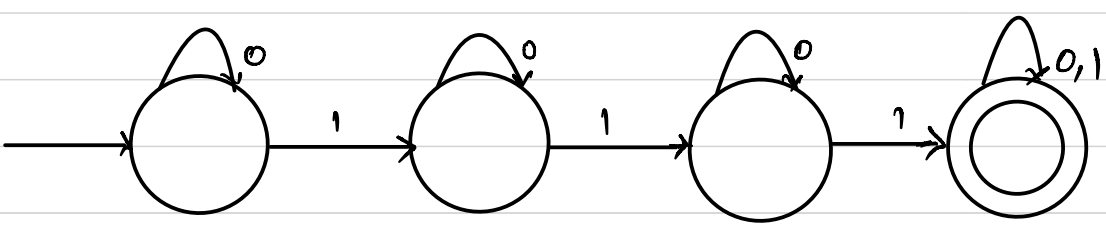
1.6 Give state diagrams of DFAs recognizing the following languages. In all parts, the alphabet is {0,1}.

- **a.** $\{w \mid w$ begins with a 1 and ends with a 0$\}$
- **b.** $\{w \mid w$ contains at least three 1s$\}$
- **c.** $\{w \mid w$ contains the substring 0101 (i.e., $w = x0101y$ for some x and $y)\}$
- **d.** $\{w \mid w$ has length at least 3 and its third symbol is a 0$\}$
- **e.** $\{w \mid w$ starts with 0 and has odd length, or starts with 1 and has even length$\}$
- **f.** $\{w \mid w$ doesn't contain the substring 110$\}$
- **g.** $\{w \mid$ the length of w is at most 5$\}$
- **h.** $\{w \mid w$ is any string except 11 and 111$\}$
- **i.** $\{w \mid$ every odd position of w is a 1$\}$
- **j.** $\{w \mid w$ contains at least two 0s and at most one 1$\}$
- **k.** $\{\epsilon, 0\}$
- **l.** $\{w \mid w$ contains an even number of 0s, or contains exactly two 1s$\}$
- **m.** The empty set
- **n.** All strings except the empty string

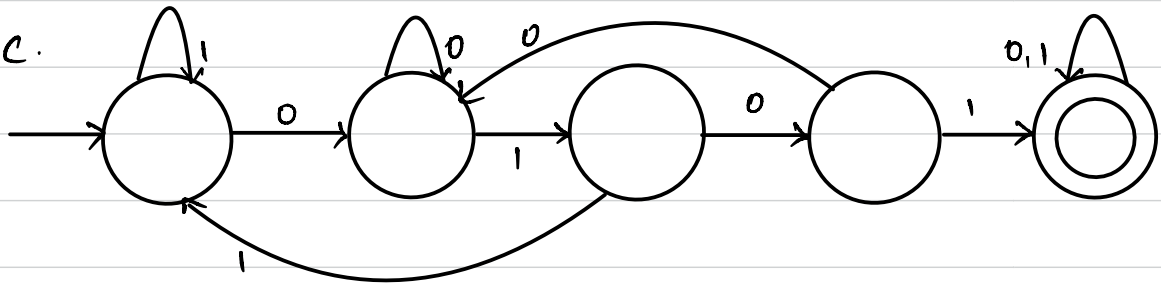
a.



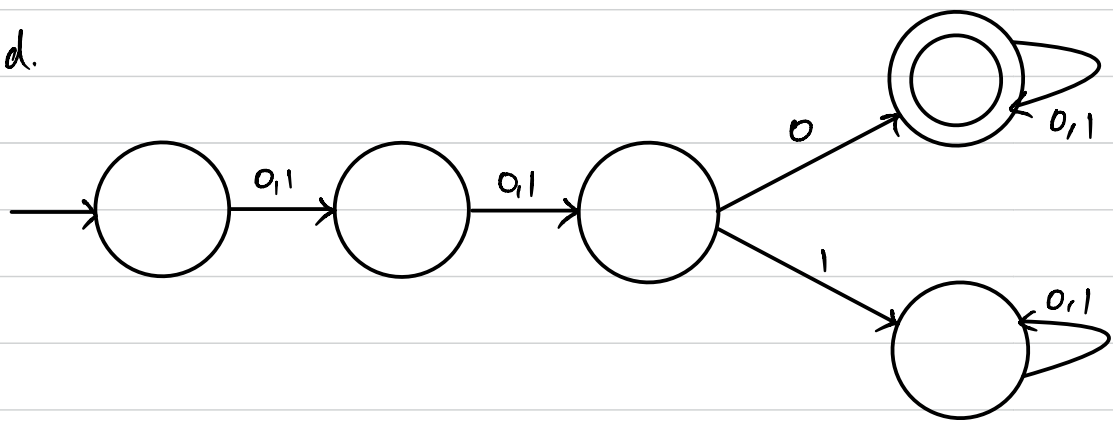
b.



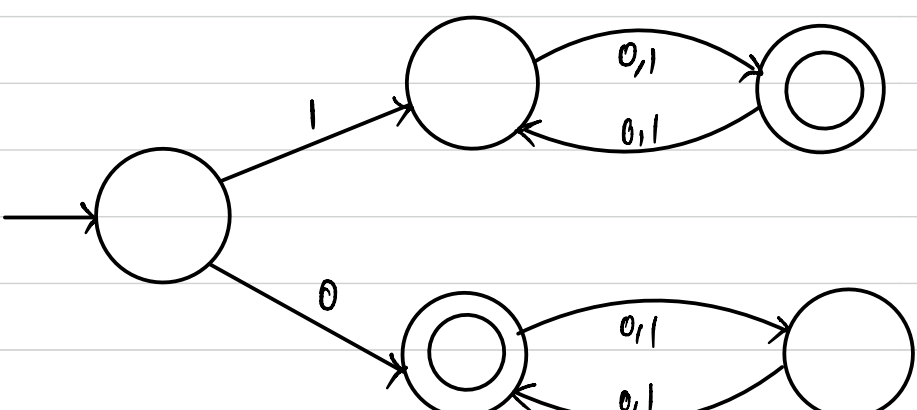
c.



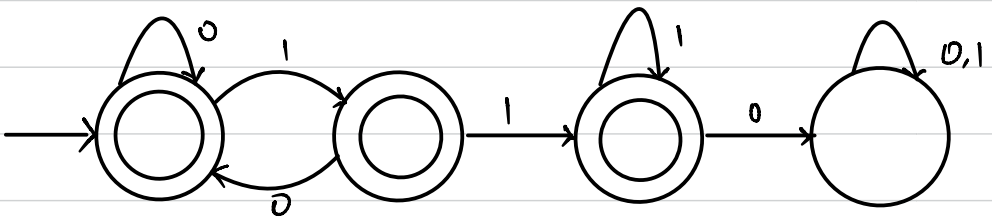
d.



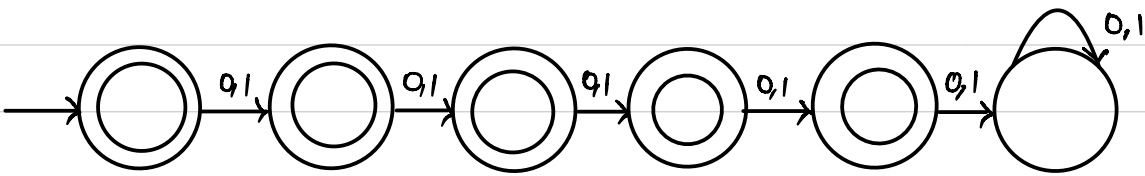
e.



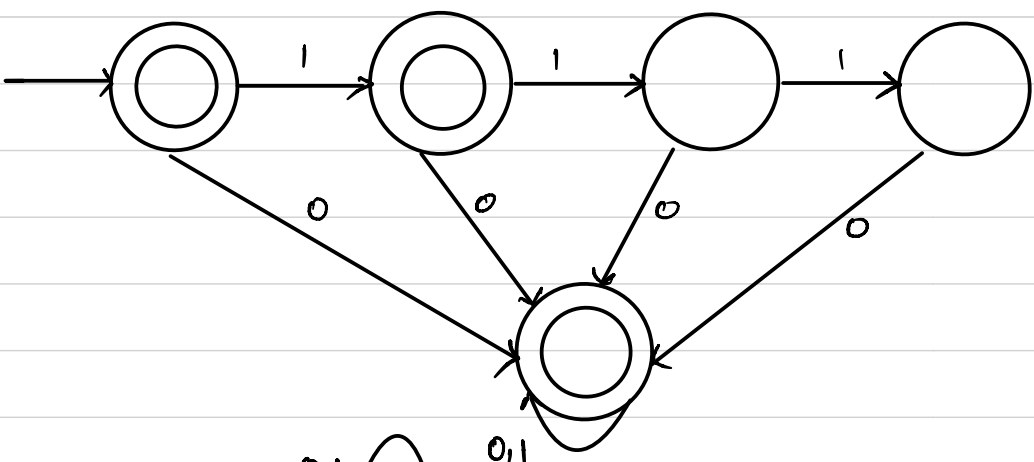
f.



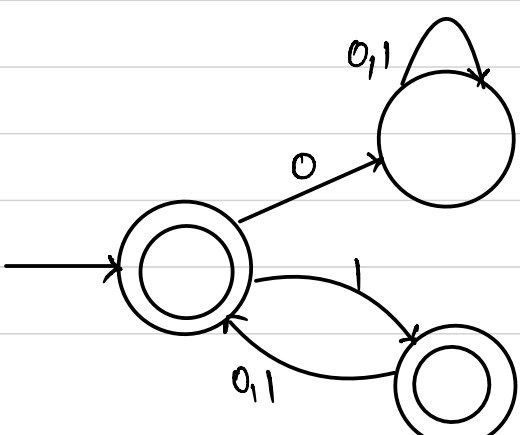
g.



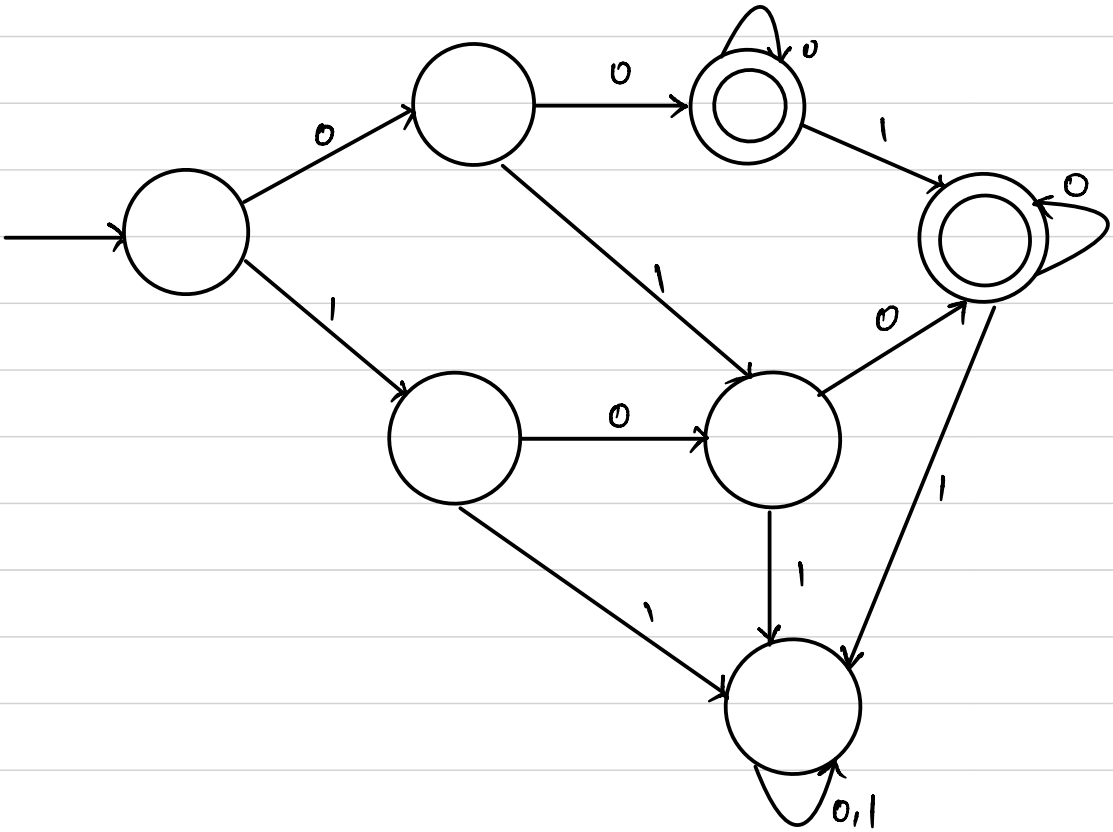
h.



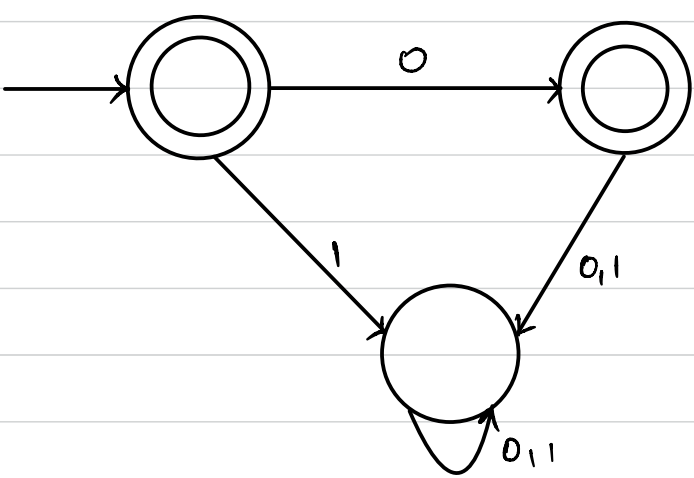
i



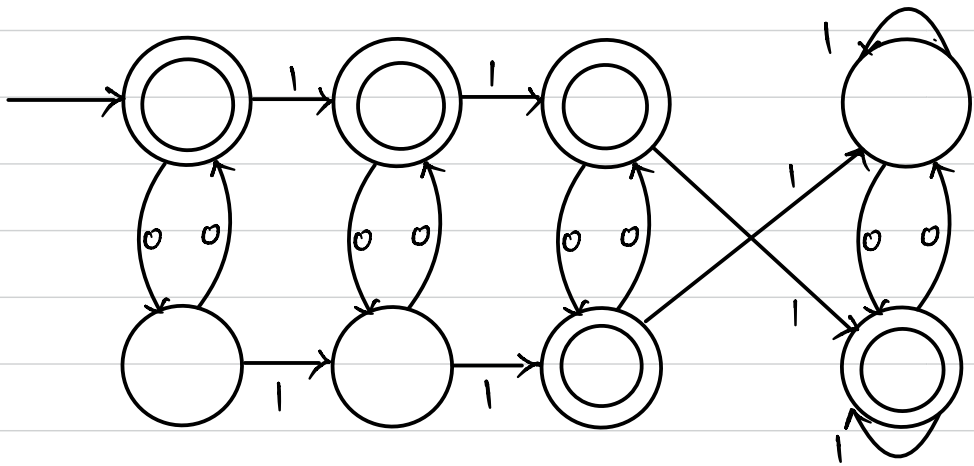
j.



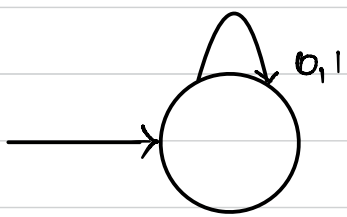
k.



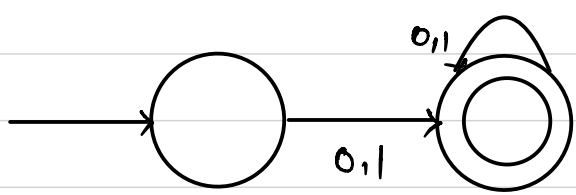
l.



m.



n.

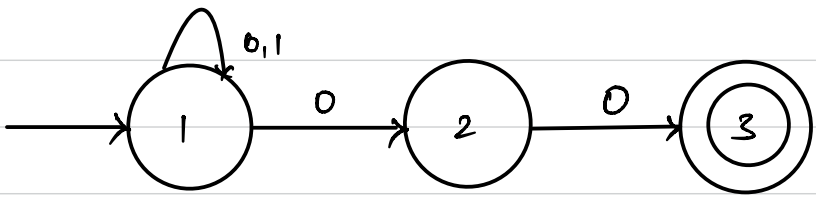


20.

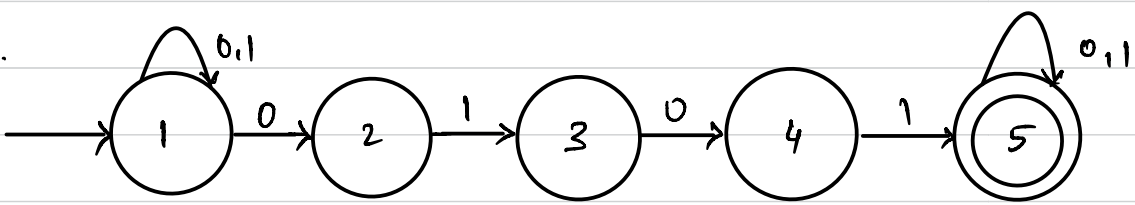
1.7 Give state diagrams of NFAs with the specified number of states recognizing each of the following languages. In all parts, the alphabet is $\{0,1\}$.

- [A] **a.** The language $\{w \mid w \text{ ends with } 00\}$ 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
- [A] **f.** The language $1^*(001^+)^*$ with three states
- **g.** The language $\{\varepsilon\}$ with one state
- **h.** The language 0^* with one state

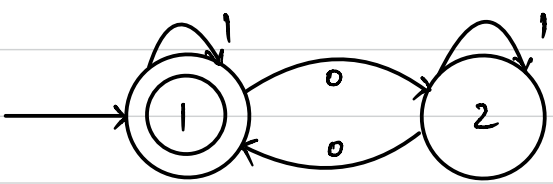
a.



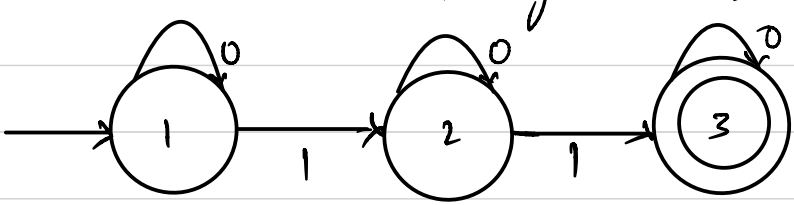
b.



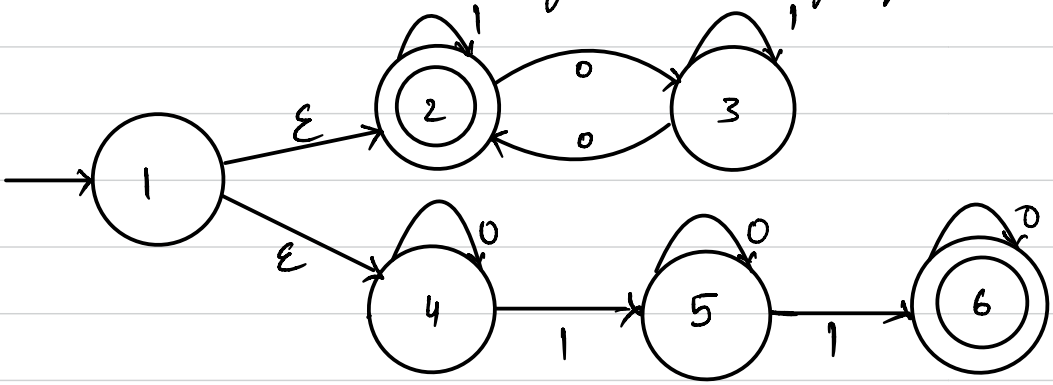
c. $L_1 = \{w \mid w \text{ contains even 0s}\}$



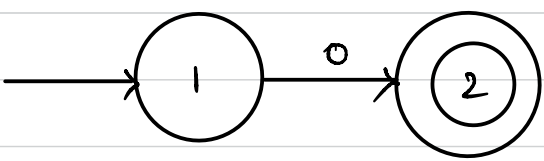
$L_2 = \{w \mid w \text{ contains exactly two 1's}\}$



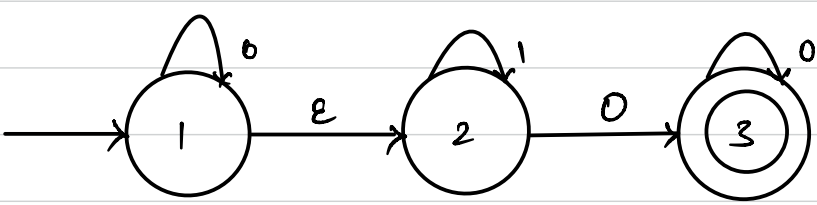
The required NFA recognized language $L_1 L_2$



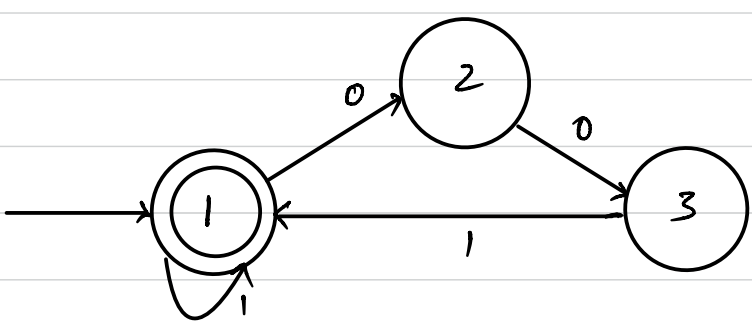
d.



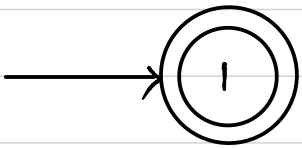
e.



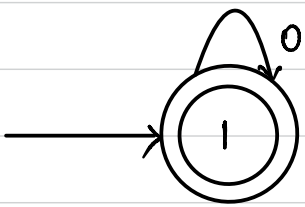
f.



g.



h.



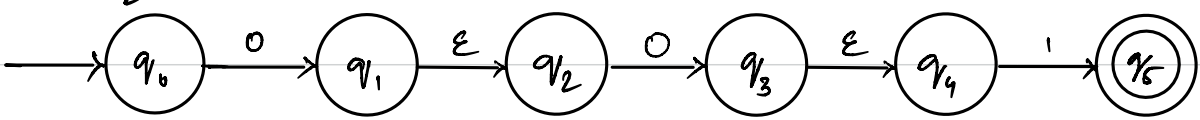
21.

1.17 **a.** Give an 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.

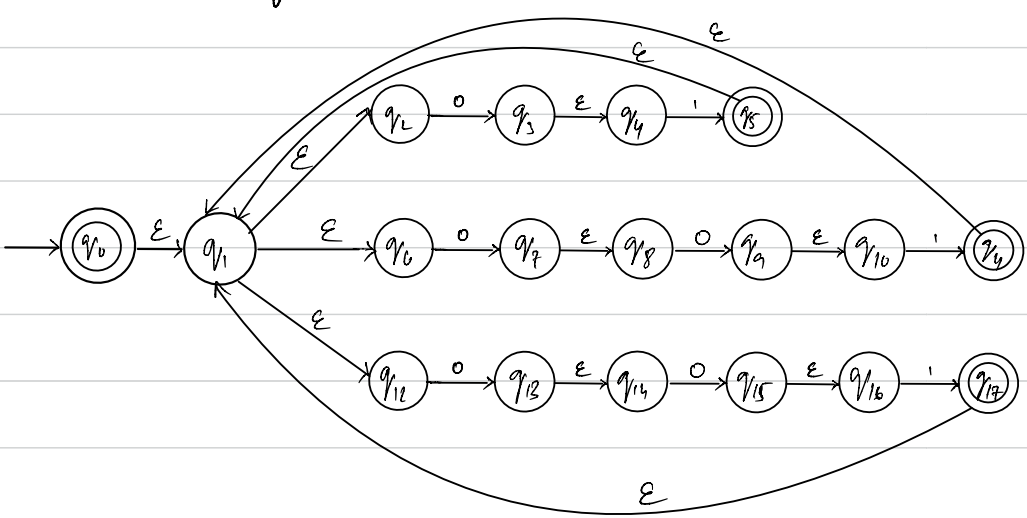
a. $L_1 = 01$

$L_2 = 001$



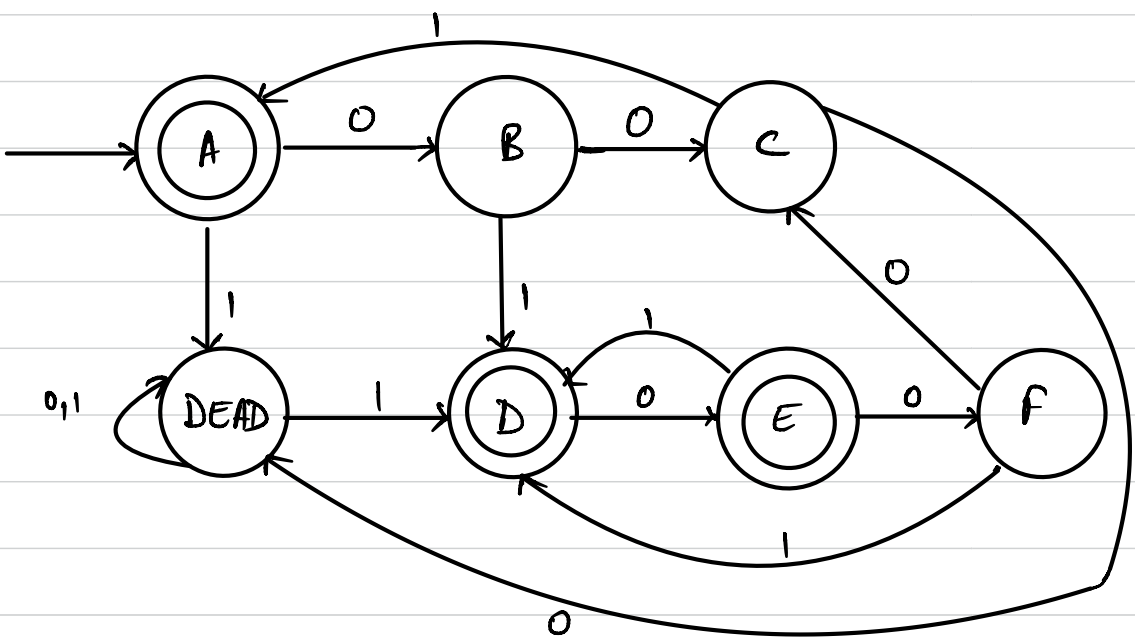
$L_3 = 010$

NFA that recognizes $L = (L_1 \cup L_2 \cup L_3)^*$



b. In order to convert this NFA to DFA, we need to get rid of all the ε's. Then draw the transition table and map the states followed by combining of any dead states if possible

DFA that recognizes $L = (01 \cup 001 \cup 010)^*$:

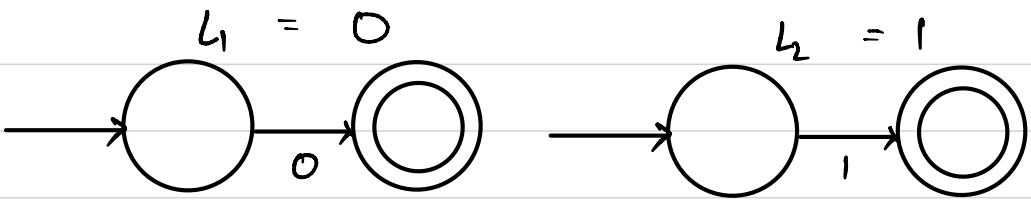


1.19 Use the procedure described in Lemma 1.55 to convert the following regular expressions to nondeterministic finite automata.

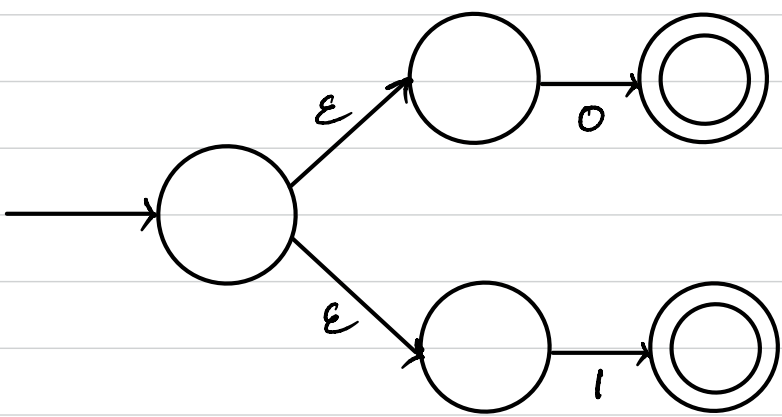
a. $(0 \cup 1)^*000(0 \cup 1)^*$
b. $(((00)^*(11)) \cup 01)^*$
c. $\emptyset^*$

22.

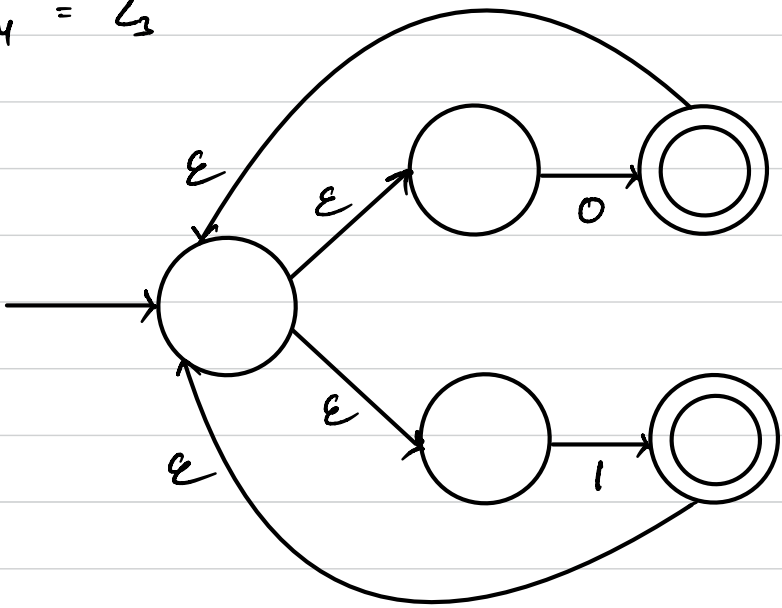
a. $R = (0 \cup 1)^* 000 (0 \cup 1)^*$



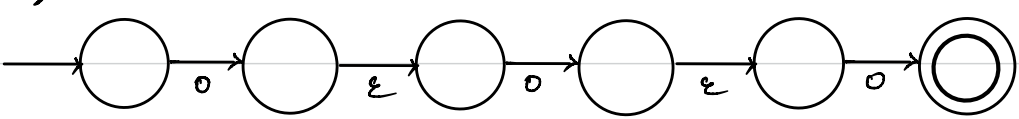
$L_3 = L_1 \cup L_2$



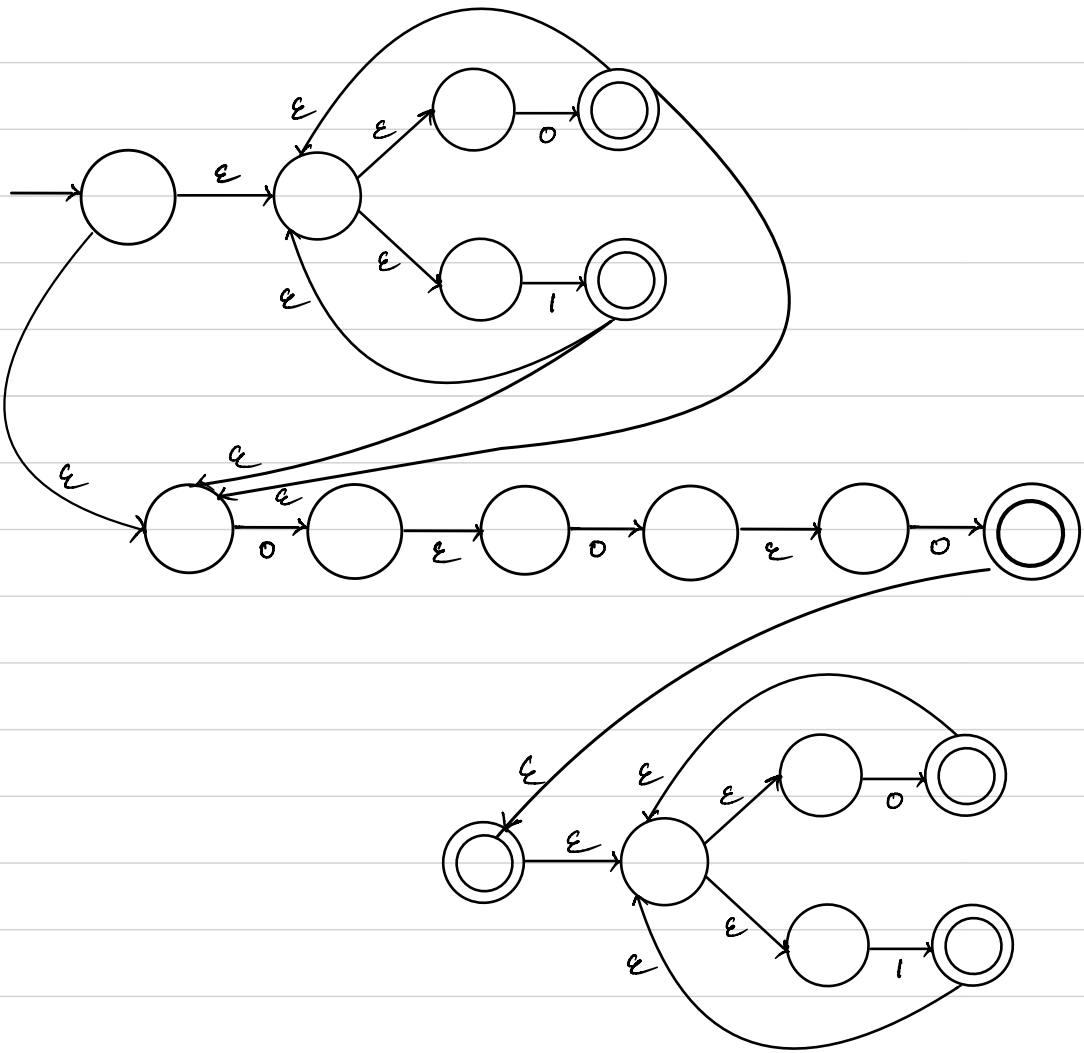
$L_4 = L_3^*$



$L_5 = 000$

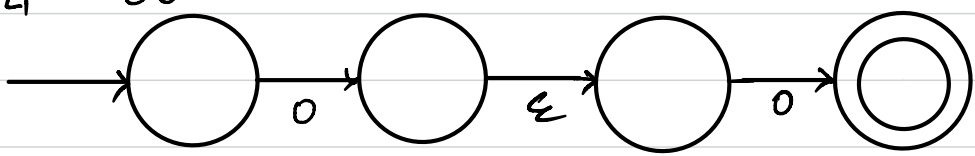


$L = L_4 L_5 L_4$

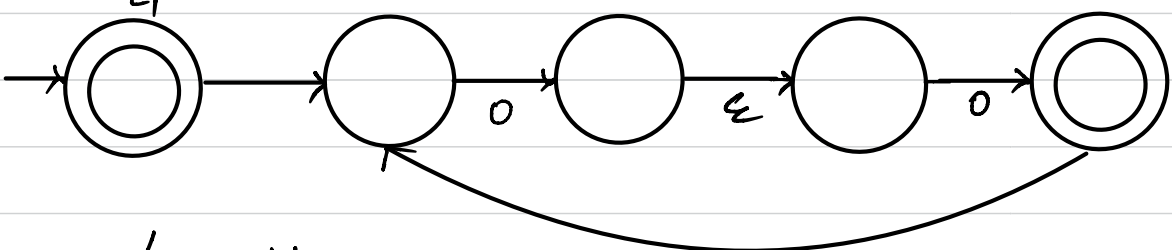


b. $((00)^*(11)) \cup 01)^*$

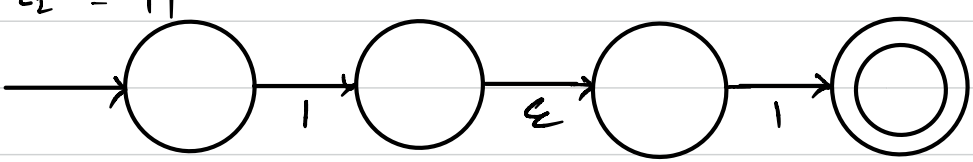
$L_1 = 00$



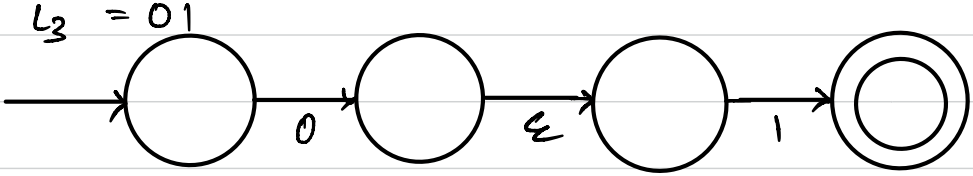
L_1^*



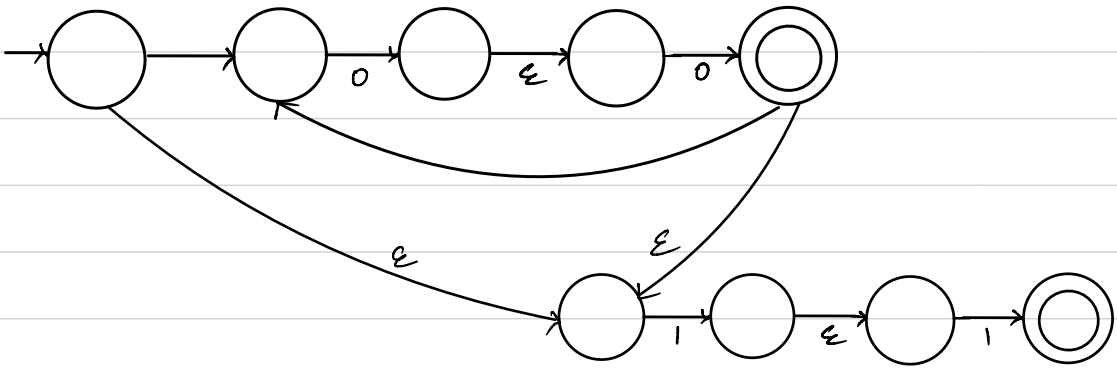
$L_2 = 11$



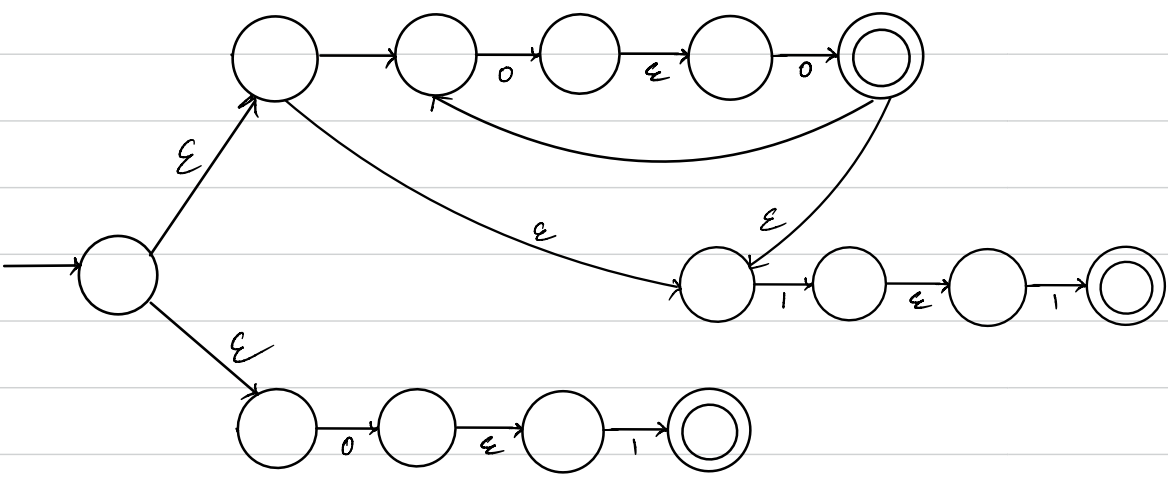
$L_3 = 01$



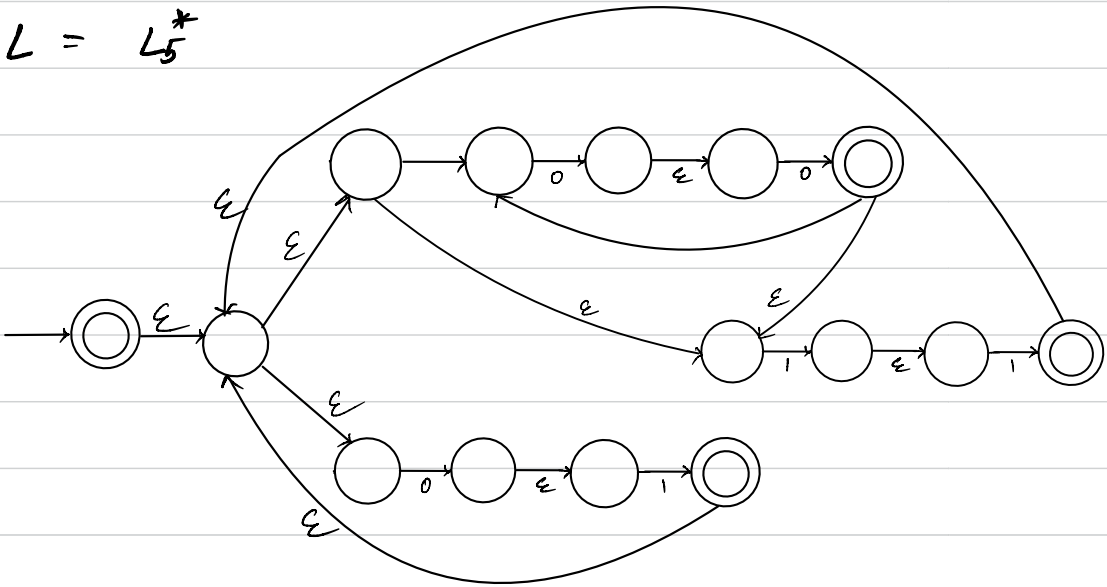
$L_4 = L_1^* L_2$



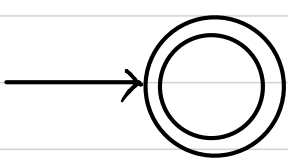
$L_5 = (L_1^* L_2) \cup L_3$



$L = L_5^*$



c. $R = \emptyset^*$



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.

23.

a. a^*b^*
b. $a(ba)^*b$
c. $a^* \cup b^*$
d. $(aaa)^*$
e. $\Sigma^*a\Sigma^*b\Sigma^*a\Sigma^*$
f. $aba \cup bab$
g. $(\varepsilon \cup a)b$
h. $(a \cup ba \cup bb)\Sigma^*$

	Members of L	Not members of L
a^*b^*	ab	ba
	aab	baa
$a(ba)^*b$	abab	b
	ababab	ba
$a^* \cup b^*$	a	ba
	b	baa
$(aaa)^*$	aaa	b
	aaaaaa	a
$\Sigma^*a\Sigma^*b\Sigma^*a\Sigma^*$	aba	a
	aabaa	b

	Members of L	Not members of L
$aba \cup bab$	aba	a
	bab	b
$(\varepsilon \cup a)b$	b	a
	ab	ba
$(a \cup ba \cup bb)\Sigma^*$	a	ε
	ba	b

24.

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 $\Sigma = \{\mathtt{a}, \mathtt{b}, /, \#\}$.

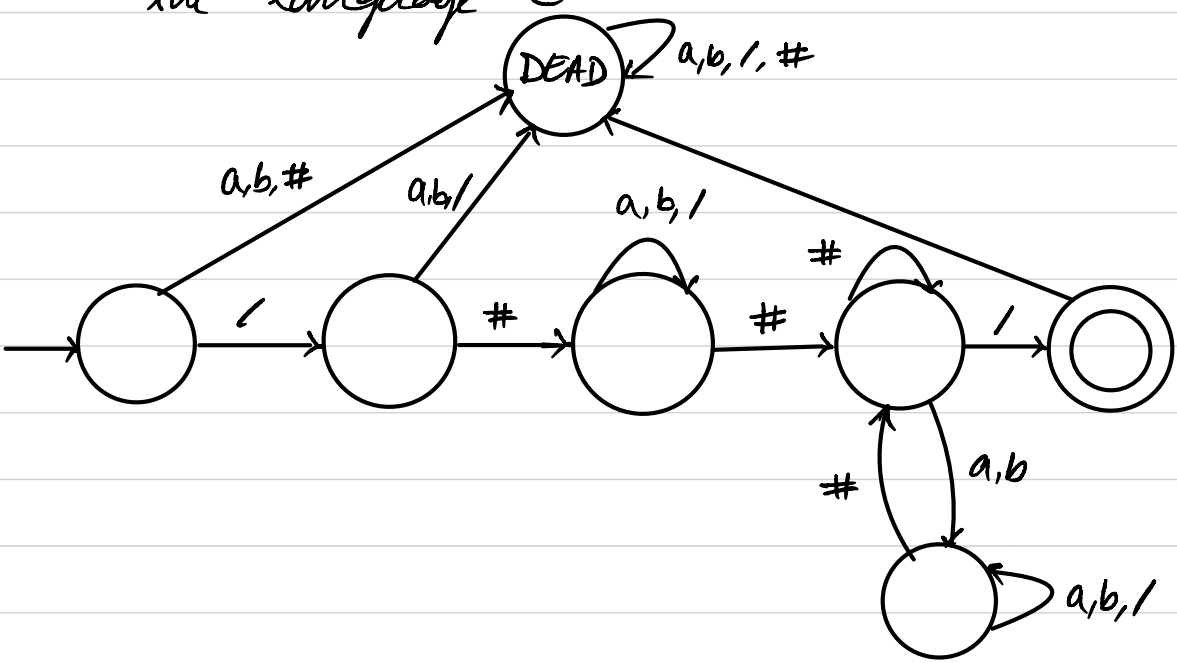
a. Give a DFA that recognizes C.

b. Give a regular expression that generates C.

a. The conditions that a valid string must follow are:

1. Start with /#
2. End with #/
3. No intervening #/

Let M be the DFA that recognizes the language C.



b. Regular expression that generates language C:

$$R = /\#(a \cup b \cup /)^{*}\#(\#\cup(a \cup b)(a \cup b \cup /)^{*}\#)^{*}/$$

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

We need to prove it both ways.

Case 1:

Consider $BB \subseteq B$ to be true

To prove: $B = B^+$

wkt for every language $BB \subseteq B^+$ -①

let us consider a string of elements 's' such that $s \in B^+$

we need to show $s \in B$

Since we said that $s \in B^+$,

s can be split into elements.

$s = x_1 x_2 x_3 \dots x_k$, $\forall x_i \in B$, $k \geq 1$

because $x_1, x_2 \in B$ & $BB \subseteq B$.

Next we shall consider the element x_3,

st $x_3 \in B$, $BB \subseteq B \Rightarrow x_1, x_2, x_3 \in B$

On continuing this process for every x_i,

we can say $x_1, x_2 \dots x_k \in B \Rightarrow s \in B$.

$\therefore s \in B$, $B^+ \subseteq B$. -②

From ①, ② we can say that $B = B^+$.

Case 2: Given $B = B^+$ we need to show that $BB \subseteq B$

wkt for every language $BB \subseteq B^+$ - ①

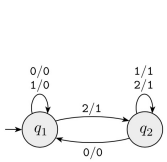
$B = B^+$ (given) - ②

From ①, ② we can say that $BB \subseteq B$

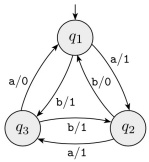
From case 1, case 2 we can say that $B = B^+$ iff $BB \subseteq B$

26.

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.



T_1



T_2

Each transition of an FST is labeled with two symbols, one designating the input symbol for that transition and the other designating the output symbol. The two symbols are written with a slash, /, separating them. In T_1, the transition from q_1 to q_2 has input symbol `2` and output symbol `1`. Some transitions may have multiple input–output pairs, such as the transition in T_1 from q_1 to itself. When an FST computes on an input string w, it takes the input symbols $w_1 \cdots w_n$ one by one and, starting at the start state, follows the transitions by matching the input labels with the sequence of symbols $w_1 \cdots w_n = w$. Every time it goes along a transition, it outputs the corresponding output symbol. For example, on input `2212011`, machine T_1 enters the sequence of states $q_1, q_2, q_2, q_2, q_2, q_1, q_1, q_1$ and produces output `1111000`. On input `abbb`, T_2 outputs `1011`. Give the sequence of states entered and the output produced in each of the following parts.

a. T_1 on input `011`
b. T_1 on input `211`
c. T_1 on input `121`
d. T_1 on input `0202`
e. T_2 on input `b`
f. T_2 on input `bbab`
g. T_2 on input `bbbbbb`
h. T_2 on input ε

Transducer	Input	Output	Sequence
T_1	011	000	q_1, q_1, q_1, q_1
T_1	211	111	q_1, q_2, q_2, q_2
T_1	121	011	q_1, q_1, q_2, q_2
T_1	0202	0101	q_1, q_1, q_2, q_1, q_2
T_2	b	1	q_1, q_2
T_2	bbab	1111	q_1, q_3, q_2, q_3, q_2
T_2	bbbbbb	110110	$q_1, q_3, q_2, q_1, q_3, q_2, q_1$
T_2	ε	ε	q_1

27.

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 (page 35). 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 $\delta: Q \times \Sigma \longrightarrow Q \times \Gamma$.)

Finite State Transducer:

1. It is a kind of finite state automaton.
2. It consists of input and output strings.
3. It converts input string to output string.

Formal definition of (FST):

FST is a 6-tuple machine represented by $M = (Q, \Sigma, \Gamma, \delta, q_0)$ where

$Q \rightarrow$ finite set of states

$\Sigma \rightarrow$ finite set of input alphabets

$\Gamma \rightarrow$ finite set of output alphabets

$\delta: Q \times \Sigma \rightarrow Q \times \Gamma$ is the transition fn

$q_0 \rightarrow$ start state ($q_0 \in Q$)

Formal definition of computation of FST:

1. Translates the input string to output string
2. Let w be an input string over Σ,
   x be an output string consisting of alphabet of Γ.

The transition carried over $q_0', q_1' \ldots q_n' \in Q'$ are such that $q_0' = q_0$

Then the transition δ is defined as:

$$\delta(q_i', w_i) = (q_i', x_i) \quad , \quad 0 < i \leq n$$

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.

28.

Formal definitions of T_1, T_2 are given by:
$M = (Q, \Sigma, T, \delta, q_0)$ where

$Q \rightarrow$ finite set of states
$\Sigma \rightarrow$ finite set of input alphabets
$T \rightarrow$ finite set of output alphabets
$\delta : Q \times \Sigma \rightarrow Q \times T$ is the transition fn
$q_0 \rightarrow$ stat state ($q_0 \in Q$)

FST T_1 is formally defined by:
$(\{q_1, q_2\}, \{0,1,2\}, \{0,1\}, \delta_1, q_1)$ where δ_1 is:

δ_1:

	0	1	2
q_1	$\{q_1, 0\}$	$\{q_1, 0\}$	$\{q_2, 1\}$
q_2	$\{q_1, 0\}$	$\{q_2, 1\}$	$\{q_2, 1\}$

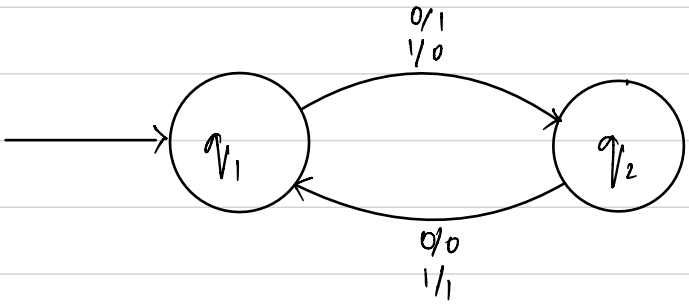
FST T_2 is formally defined by:
$(\{q_1, q_2, q_3\}, \{a, b\}, \{0,1\}, \delta_2, q_1)$ where δ_2 is:

δ_2:

	a	b
q_1	$\{q_2, 1\}$	$\{q_3, 1\}$
q_2	$\{q_3, 1\}$	$\{q_1, 0\}$
q_3	$\{q_1, 0\}$	$\{q_2, 1\}$

29.

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.



The above is the state diagram of the FST that follows the conditions given in the question.

30.

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

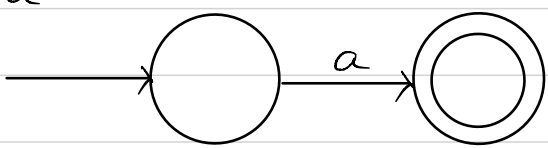
a. $a(abb)^* \cup b$

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

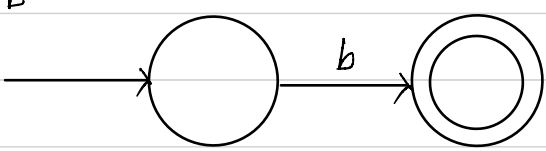
c. $(a \cup b^+)a^+b^+$

a.

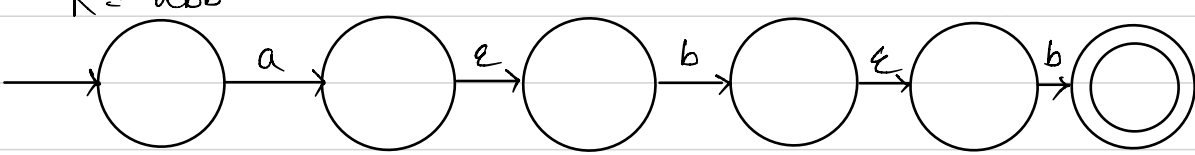
$R = a$



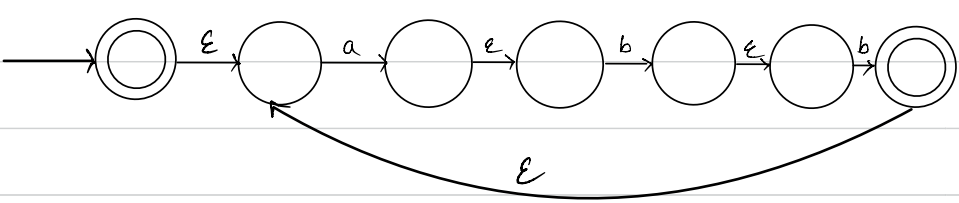
$R = b$



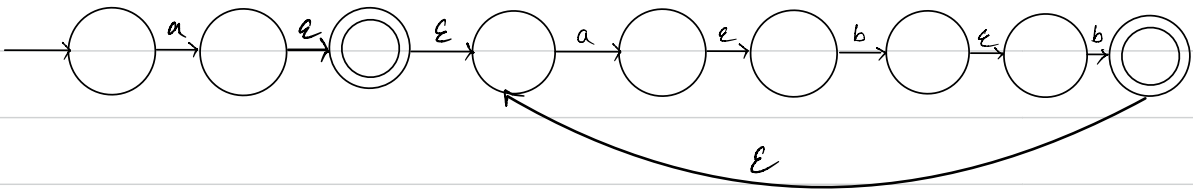
$R = abb$



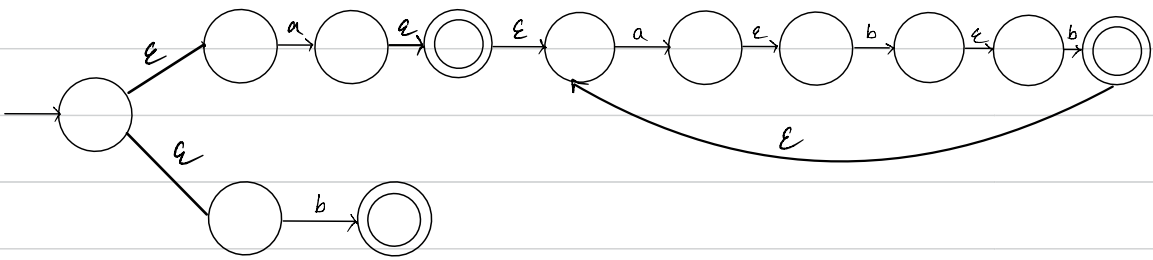
$R = (abb)^*$



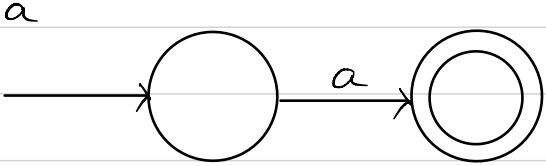
$R = a(abb)^*$



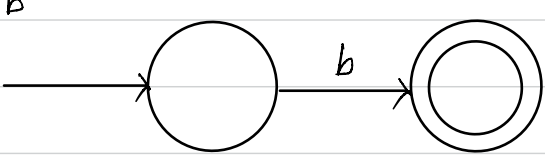
$R = a(abb)^* \cup b$



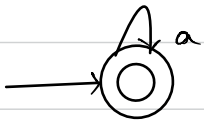
b. $R = a$



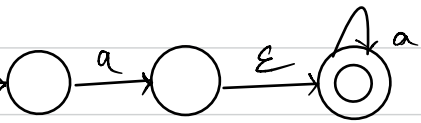
$R = b$



$R = a^*$

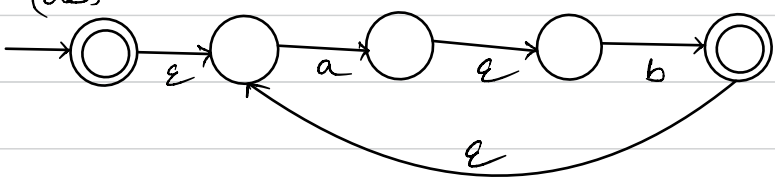


$R = a^+ = aa^*$

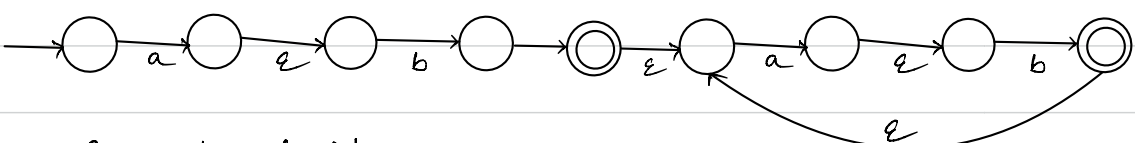


$R = ab$

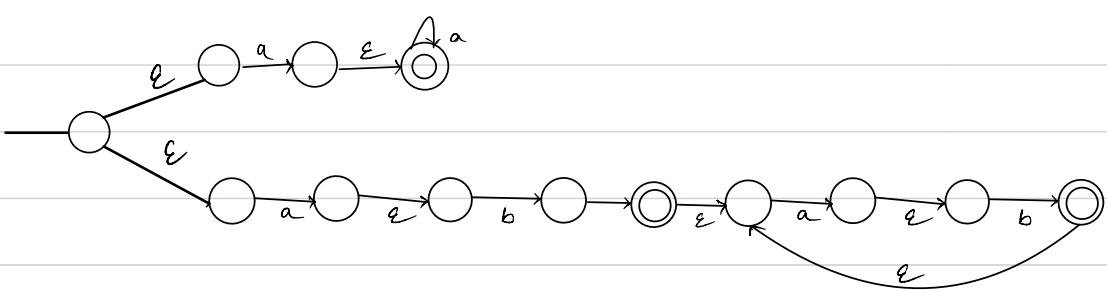
$R = (ab)^*$



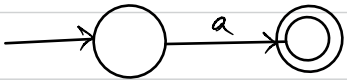
$R = (ab)^+ = (ab)(ab)^*$



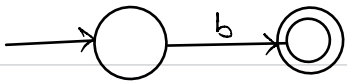
$R = a^+ \cup (ab)^+$



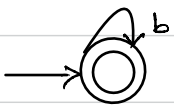
$R = a$



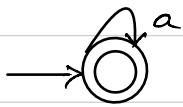
$R = b$



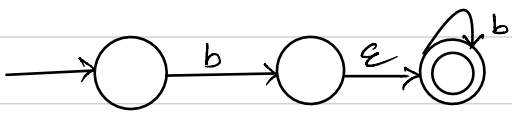
$R = b^*$



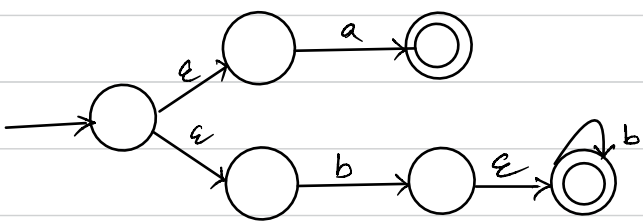
$R = a^*$



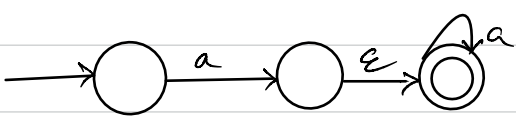
$R = b^+ = bb^*$



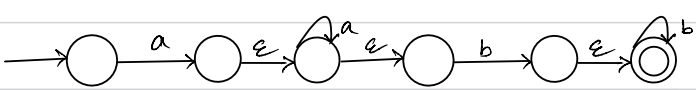
$R = a \cup b^+$



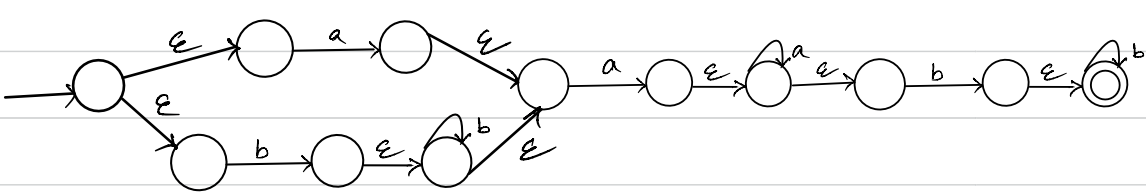
$R = a^+ = aa^*$



$R = a^+b^+$



$R = (a \cup b^+)a^+b^+$



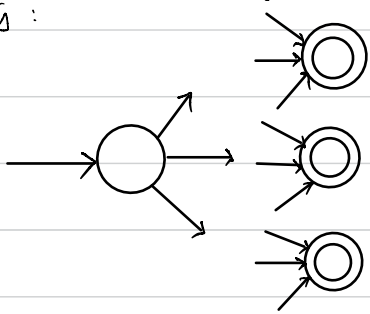
1.31 For any string $w = w_1 w_2 \cdots w_n$, the ***reverse*** of w, written w^R, is the string w in reverse order, $w_n \cdots w_2 w_1$. For any language A, let $A^R = \{w^R \mid w \in A\}$. Show that if A is regular, so is A^R.

31.

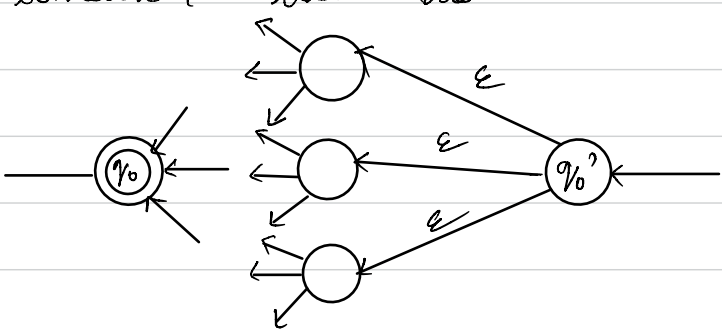
Let M be a DFA that recognizes A,

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

The state diagram of M would look somewhat like this:



The state diagram of NFA M' for A^R would look somewhat like this:



Therefore we observe that $q_0' = q'_{accept}$

$\Rightarrow$ for any $w \in \Sigma^*$,

there exists a path that follows from start state to accept state in M iff there exists a path following from w^R from q_0' to q'_{accept} in M'.

Hence we can say that,

$w \in A$ iff $w^R \in A^R$

32.

1.32 Let

$$\Sigma_3 = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}.$$

Σ_3 contains all size 3 columns of 0s and 1s. A string of symbols in Σ_3 gives three rows of 0s and 1s. Consider each row to be a binary number and let

$$B = \{w \in \Sigma_3^* | \text{ the bottom row of } w \text{ is the sum of the top two rows}\}.$$

For example,

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \in B, \quad \text{but} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \notin B.$$

Show that B is regular. (Hint: Working with $B^{\mathcal{R}}$ is easier. You may assume the result claimed in Problem 1.31.)

This problem is the use of additive operator of numbers. Application would be to find the sum of two numbers.

Our goal is to prove B^R is regular since it implies that B is regular (Regular languages are closed under reversal)

A language is said to be regular if an automaton recognizes it.

∴ let M be an automaton that recognizes B^R

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

where, $Q = \{c_0, c_1\}$

$c_0 \rightarrow$ denotes that the read part of the string leads to a carry.

$c_1 \rightarrow$ represents the carry.

$$\Sigma = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \cdots \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$$

$q_0 = C_0$

$F = \{C_0\}$

δ is defined as:

$$\delta(C_0, a) = C_0 \quad , \quad a = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\delta(C_0, a) = C_1 \quad , \quad a = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\delta(C_1, a) = C_1 \quad , \quad a = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\delta(C_1, a) = C_0 \quad , \quad a = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Any other arrows just point to the dead state.

$\Rightarrow$ M recognizes B^R $\Rightarrow$ B^R is regular $\Rightarrow$ B is a regular language

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 0s and 1s of height two. A string of symbols in Σ_2 gives two rows of 0s and 1s. 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.)

33.

Formulation of elements that belong to C:

If the top row represents a number n_1, and bottom represents n_2,

$$n_2 = n_1 + 2n_1$$

We can find $2n_1$ by simply performing a left shift followed by addition.

Let us represent the binary digits of the top row with $X_{n-1}, X_{n-2}, \cdots X_0$ and bottom row with $Y_n, Y_{n-1} \cdots Y_0$.

$$\begin{array}{cccccc} X_n & X_{n-1} & \cdots & X_1 & X_0 \\ X_{n-1} & X_{n-2} & \cdots & X_0 & 0 \\ \hline Y_n & Y_{n-1} & \cdots & Y_1 & Y_0 \end{array}$$

Let C_i^{in}, C_i^{out} be carry in and out of the i^{th} index.

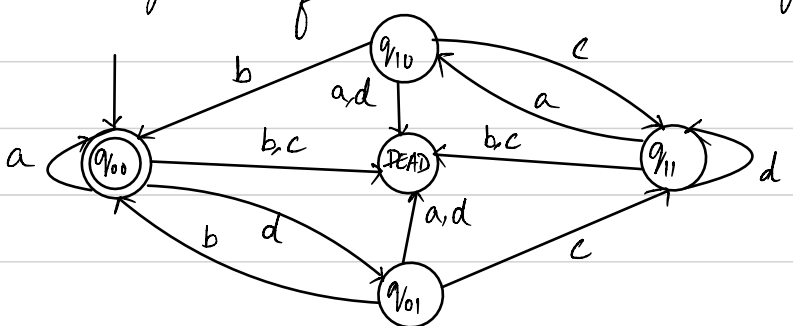
$C_0^{in}, C_0^{out} = 0$

$P_i = X_i \text{ XOR } X_{i-1} \text{ XOR } C_i^{in}$

$$C_i^{out} = \begin{cases} 0 & , \ x_i \text{ and } x_{i-1} = 0 \\ 1 & , \ x_i \text{ and } x_{i-1} = 1, \ x_i \text{ xor } x_{i-1} = 0 \end{cases}$$

$$C_i^{in} = C_{i-1}^{out}$$

The state diagram of the DFA that recognizes C:



$$a = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad d = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Consider string abda, $\begin{bmatrix} 0 \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

When read in reverse order : $a \to d \to b \to a$

the string is accepted.

∴ DFA recognizes C^k, C^k is regular

$\Rightarrow$ C is regular.

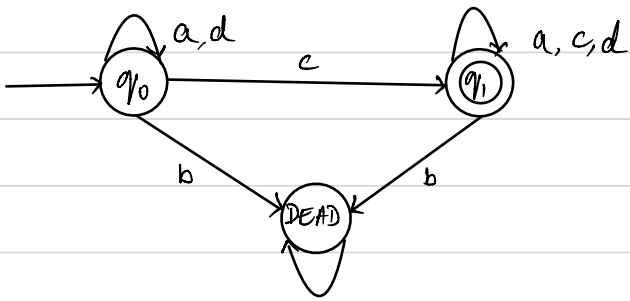
34.

1.34 Let Σ_2 be the same as in Problem 1.33. Consider each row to be a binary number and let

$$D = \{w \in \Sigma_2^* | \text{ the top row of } w \text{ is a larger number than is the bottom row}\}.$$

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.

A DFA that recognizes D over $\Sigma = \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}$ would have a starting state such that the top row is either equal to or larger than the bottom. The state diagram is:



$$a = \begin{bmatrix}0\\0\end{bmatrix}, \quad b = \begin{bmatrix}0\\1\end{bmatrix}, \quad c = \begin{bmatrix}1\\0\end{bmatrix}, \quad d = \begin{bmatrix}1\\1\end{bmatrix}$$

Consider string acda, $\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}$

When read in reverse order: $a \to c \to d \to a$ the string is accepted.

∴ DFA recognizes D^R, D^R is regular

$\Rightarrow$ 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 0s and 1s, and let

$$E = \{w \in \Sigma_2^* | \text{ the bottom row of } w \text{ is the reverse of the top row of } w\}.$$

35. Show that E is not regular.

Let us assume E is regular and disprove via contradiction using the pumping lemma.

Consider a string s, $w \in L$:

$$s = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^p \begin{bmatrix} 0 \\ 1 \end{bmatrix}^p \quad \text{where } p \rightarrow \text{positive integer}$$

$$s = 1^p 0^p = (0^p 1^p)^R$$

$$|s| = 2p \Rightarrow |s| \geq p$$

If we were to divide w into 3 pieces such that:

$$w = xyz, \quad p \geq q, \quad y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^p, \quad z = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^p$$

In order to perform further division,

$$y = uvw, \quad v = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^m, \quad 0 < m \leq p$$

let i be any value st $i \geq 0$. let $i = 2$,

$$s = xuv^i wz \qquad s = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^{p+m} \begin{bmatrix} 0 \\ 1 \end{bmatrix}^p$$

$\because m > 0$, $c^{p+m} b^p \neq (b^{p+m} c^p)^R$

$\Rightarrow$ Top row isn't the reversal of bottom row due to the above contradiction.

$xuv^2wz \notin L$

$\therefore$ The given language L is not a regular language.

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

36.

Since K is a multiple of n,

$K = ni$, $i \in Z^+$

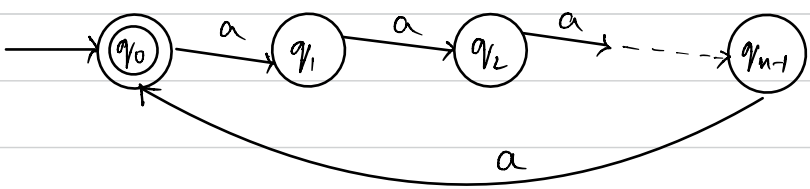
Let us see some sets of B for $i = 1$

$n = 1$ $B_1 = \{a\}$

$n = 2$ $B_2 = \{aa\}$

$n = 3$ $B_3 = \{aaa\}$

Finite automaton of this regular exprⁿ is :



From the closure property of regular exprⁿ, we can say that B_n is regular for $n \geq 1$.

B_n also obeys regular operations such as union, intersection, set difference, complement, etc.

∴ B_n is a regular expression.

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.

37.

If a number is a multiple of n, it can be represented as nx, $x \geq 1$.

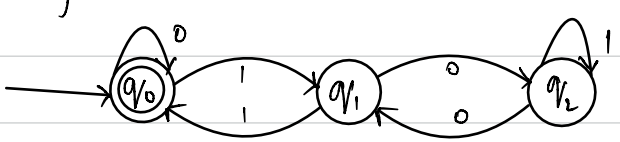
$$nx \bmod n \equiv 0$$

If we were to check any number k,

$$k \bmod n = \begin{cases} 0, & k = 3n \\ 1, & k = 3n+1 \\ \vdots & \\ n-1, & k = 3n+(n-1) \end{cases}$$

For simplicity, consider $n = 3$

The DFA for C_n :



Strings recognized are: 0, 10101, ...

0 21

This can be showed $\forall n, n \geq 1$.

$\therefore M = (\{q_0, q_1, \ldots q_n\}, \{0,1\}, \delta, q_0, \{q_0\})$.

$\therefore M$ recognizes C_n, C_n is regular.

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.

38.

Every regular language is recognized by some all-NFA.

Proof:

wkt every DFA is equivalent to an all-NFA.

let L be a regular that is recognized by a DFA, $M = (Q, \Sigma, \delta, q_0, F)$.

For each input string w, $w \in \Sigma^*$, there exists exactly one possible state $q \in Q$ that M would be in after reading w.

For M to be an all-NFA, it will recognize w iff $q \in F \Rightarrow w \in L$.

Since all-NFA recognizes L, M is an all-NFA

$\Rightarrow$ every regular language is recognized by some all-NFA.

Every all-NFA recognizes a regular language.

Proof:

let L be the language recognized by an all-NFA, $N = (Q, \Sigma, \delta, q_0, F)$.

let us construct a DFA, $M = (Q', \Sigma, \delta', q_0', F')$ that

recognizes L. Here $Q' = P(Q)$, where $P(Q) \subset Q$, for $R \in Q'$, $a \in \Sigma$ we have $\delta'(R,a) = \bigcup_{r \in R} E(\delta(r,a))$, for $S \subset Q$, $E(S)$ is the set of all states $q \in Q$ that can be reached from S by moving along arrows with ε, $q_0' = E(\{q_0\})$, $F' = \{R \in Q' \mid R \subset F\}$.

Let us introduce a string w, $w \in \Sigma^*$, R are the set of states that N lands up in after reading w. When M reads the same string, it will also end up in R.

wkt for all NFA, $w \in L$ iff $R \subset F$ and M recognizes w iff $R \in F'$ which is the same as $R \subset F$.

$\Rightarrow$ M recognizes L.

On proving the two statements we can say that all-NFA's recognizes the class of regular languages.

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.

39.

Given: If there is a two state GNFA is equivalent to GNFA.

To prove: Opposite phenomenon occurs in DFA.

Proof: We need to show that no DFA is equivalent to a DFA with lesser states.

Assume A_K is a set of words of minimum length of $K-1$. $\Rightarrow A_K$ has at least K equivalent strings of word length n, $n \geq 0$. $\therefore$ A DFA with $K-$ states is required to recognize A_K.

Using the pigeon hole principle we can say that a DFA with fewer than $K-$states will have at least two strings that would lead to the state.

For example, let us look at a case:
Let $A_k = \Sigma^* 0^{k-1} 0^*$, $k > 1$, $\Sigma = \{0, 1\}$
A DFA with k-states recognizes A_k. There is a state for every 0 it reads after every 1 that it read starting from the start state. After reading through k-1 0's it reaches the final state. If the DFA were to contain fewer than k-states then the string would cause it to loop to a single state. Hence the string is not recognized.

Hence proved.

40.

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.

[A]**a.** $NOPREFIX(A) = \{w \in A \mid \text{no proper prefix of } w \text{ is a member of } A\}$.

b. $NOEXTEND(A) = \{w \in A \mid w \text{ is not the proper prefix of any string in } A\}$.

Assume a DFA, $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes A.

a. let $L = \{w \in \Sigma^* : x \in A \text{ and } z \in \Sigma^*, \text{ st } xz = y\}$

let NFA, $N = (Q', \Sigma, \delta', q_0', F')$ st,

$Q' = Q \cup \{q_f\}$, $q_f \notin Q$,

for $q \in Q'$, $a \in \Sigma$, $\delta'(R, a) = \begin{cases} \delta(R,a), & r \notin F \\ \emptyset, & r \in F \end{cases}$,

$q_0' = q_0$, $F' = q_f$

Consider a string w, $w \in L$, $\exists$ string y, $y \in A$ st, $xz = y$ and $x \neq \emptyset$ where $x \rightarrow$ proper prefix of y. On M' reading w, computation of x leaves it at an accepting state of M and computation of z takes it to q_f and if it were to be recognized, computation ends at q_f. This means that from construction of M', it arrives at an accepting state of M before reaching q_f.

If x were a proper prefix of y, on input x, M ends at an accepting state $\Rightarrow w \in L$, $x \in A$.

Therefore, our initial consideration holds true.

NOPREFIX(A) is equivalent to $A \cap \bar{L}$.

wkt regular languages are closed under intersection and complement.

$\Rightarrow$ NOPREFIX(A) is also regular.

b. let us assume that M accepts only those languages that reach the final state only once.

For a state $q \in F$, we check its path from $q \in Q$ to a state in F exists. let $F' \subseteq F$ be the set of states where there is no such path.

By changing $F = F'$ we get a DFA for NOEXTEND(A)

$\Rightarrow$ NOEXTEND(A) is regular.

*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.

41.

Given: $A/B = \{w \mid wx \in A \text{ for some } x \in B\}$

$A \rightarrow$ regular language, $B \rightarrow$ any language

To prove: $A/B \rightarrow$ regular

Proof: let DFA, M recognize A (∵ A → regular)

$M = (Q, \Sigma, \delta, q_0, F)$ where,

Q is the set of states

Σ is the set of alphabets of A, B

δ is the transition function

q_0 is the start state

F is the set of final states

For A/B to be regular, a DFA-M' must recognize it.

let $M' = (Q', \Sigma, \delta', q_0', F')$

Q' is the set of states $= Q$

Σ is the set of alphabets of A, B

δ' is the transition function $= \delta$

q_0' is the start state $= q_0$

F' is the set of final states

$F' = \{ q_i \in Q \mid \exists x \in B$ st on reading x,
M goes from q_i to some state in $F \}$

$\therefore$ M' recognizes A/B

$\Rightarrow$ A/B is a regular language.

42.

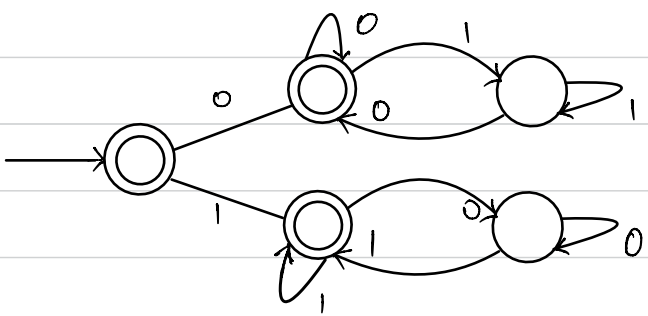
1.48 Let $\Sigma = \{0,1\}$ and let

$D = \{w | w$ contains an equal number of occurrences of the substrings 01 and 10$\}$.

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.

If we can show a DFA recognizes D then it proves that D is regular.

Let us construct the state diagram of such a DFA:



Hence D is a regular language.

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

43.

Let us assume that an FST can output w^R for every input w and disprove with an example.

Consider a string $w = w_1 w_2 \ldots w_n$

$\Rightarrow w^R = w_n, w_{n-1} \ldots w_1$

$w = 00$ $\qquad$ $w = 01$

output $= 00$ $\qquad$ output $= 10$

From the above example we observe that the first input bit is the same in both cases but first output bit differs.

However the definition of FST tells us that it can produce its first output before it reads the second input.

∴ The FST can't produce $w^R = 10$ on input $w = 01$.

This leads to contradiction of our assumption.

$\Rightarrow$ No FST outputs w^R on input w.

44.

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.

Definitions:

Distinguishable by L: x, y are distinguishable by L if $\exists$ some z st exact one of xz and $yz \in L$.

Indistinguishable by L: x, y are indistinguishable by L if for every z, $xz \in L$ whenever $yz \in L$.

To show equivalence relation, $\equiv_L$ it must obey the following properties:

1. Reflexive: $x \equiv_L x$ is true then,
   For all strings z, $xz \in L$ iff $xz \in L$
   $\Rightarrow x \equiv_L x \Rightarrow \equiv_L$ is reflexive

2. Symmetry: $x \equiv_L y \iff y \equiv_L x$ holds true then,
   $\forall z$, $xz \in L$ iff $yz \in L$ is equivalent to
   $\forall z$, $yz \in L$ iff $xz \in L$
   $\Rightarrow y \equiv_L x$ is true $\Rightarrow \equiv_L$ is symmetric

3. Transitive: if $a \equiv_L b$ and $b \equiv_L c$ then $a \equiv_L c$

$\forall z, az \in L$ iff $bz \in L$ and

$\forall z, bz \in L$ iff $cz \in L$ implies that

$\forall z, az \in L$ iff $cz \in L$ holds true

because $a \equiv_L c$ is true

$\Rightarrow \equiv_L$ is transitive

Since $\equiv_L$ obeys the above 3 properties, it is an equivalence relation.

45.

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

$ADD = \{x=y+z \mid x, y, z \text{ are binary integers, and } x \text{ is the sum of } y \text{ and } z\}$.

Show that ADD is not regular.

In order to disprove that ADD is regular, we will use the pumping lemma. Even if one of the cond^n of the lemma fail then we've disproved our assumption - ADD is regular.

Let p be the pumping length. Consider a string that follows $x = y + z$, $1^{p+1} = 10^p + 1^p \in ADD$.

Let $p = 2 \Rightarrow 111 = 100 + 11 \in ADD$

Let us divide the string into 3 parts u, v, w.

$u = 1$, $v = 1$, $w = 1 = 100 + 11$.

wkt string S: $\underbrace{1}_{u}\ \underbrace{1}_{v}\ \underbrace{1 = 100 + 11}_{w}$

After pumping $uv^i w$ when $i = 2$ we have,

S: $1111 = 100 + 11$

But S: $1111 = 100 + 11 \notin ADD$ since $1111 \neq 111$

which leads to a contradiction

$\Rightarrow$ ADD is not regular.

46.

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] **a.** 0001^*
- [A] **b.** 0^*1^*
- **c.** $001 \cup 0^*1^*$
- [A] **d.** $0^*1^+0^*1^* \cup 10^*1$
- **e.** $(01)^*$
- **f.** ε
- **g.** $1^*01^*01^*$
- **h.** $10(11^*0)^*0$
- **i.** 1011
- **j.** Σ^*

a. If we consider $p < 4$, the string simply won't accept 0, 00 or 000. $\therefore p \geq 4$ will be able to divide the string into 3 (xyz) 000 1 ... .
$\therefore$ min pumping length $= 4$

b. For $p < 1$, we will have ε which means when we divide into 3 parts, one of them will be ε.
$\therefore$ min pumping length $= 1$

c. Clearly when 001 is the output, it can't be pumped. Consider 0^*1^*, $x \to \varepsilon$, $y \to$ first symbol, $z \to$ everything that follows.
$\therefore$ min pumping length $= 4$

d. Clearly 11 can't be pumped $\Rightarrow p > 2$. Let us try $p = 3$. $S = 0^*1^+0^+1^* \rightarrow xyz$ ($x = \varepsilon$, y = first string, z = everything left) which works. Also holds true for 101.
$\therefore$ minimum pumping length = 3

e. Wkt ε can't be pumped $\Rightarrow p > 0$. $S = 01$ then $x = \varepsilon$, $y = 01$, z = everything that follows. $len(S) \neq 1$ $\because S = (01)^*$
$\therefore$ minimum pumping length = 2

f. $S = \varepsilon$ can't be pumped $\because$ p must be ≥ 1.
$\therefore$ minimum pumping length = 0

g. $S = 00$ can't be pumped $\Rightarrow p > 2$. $len(S) = 3$
$\Rightarrow S = 001, 010$ or 100 $\therefore x = \varepsilon, 0, \varepsilon$, $y = 00, 1, 00$
z = rest, 1, rest respectively which satisfies it.
$\therefore$ min pumping length = 3.

h. $s = 100$ can't be pumped $\Rightarrow s > 3$. For $s = 10100$, $x = 10$, $y = 10$, $z =$ rest then y can be pumped.
$\therefore$ min pumping length $= 4$

i. $s = 1011$ is fixed. For $p < 5$ it isn't pumpable.
for $p = 5$ it can be pumped.
$\therefore$ min pumping length $= 5$

j. $s = \Sigma^*$. p can't be 0. For $p = 1$ we will have
$x = \varepsilon$, $y \in \{\varepsilon, 0, 1\}$, $z = \varepsilon$
$\therefore$ min pumping length $= 1$

47.

*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.

let $A = \{a^* \# b^*\}$

since $\{a^* b^*\}$ is regular,

$A_{\frac{1}{3}-\frac{1}{3}} \cap \{a^* b^*\} = \{a^n b^n \mid n \geq 0\}$

will be regular if $A_{\frac{1}{3}-\frac{1}{3}}$ is regular (closed under intersection)

However we can easily prove that $\{a^n b^n \mid n \geq 0\}$ is not regular using pumping lemma.

$S = xyz$, $xyz = aabb$

$\Rightarrow x = a$, $y = a$, $z = bb$

For $p = 2$, $xy^2z = aaabb = a^3b^2 \notin \{a^n b^n \mid n \geq 0\}$

$\Rightarrow \{a^* b^*\}$ is not regular & $A_{\frac{1}{3}-\frac{1}{3}}$ is also not regular $\because$ regular languages are closed under intersection.

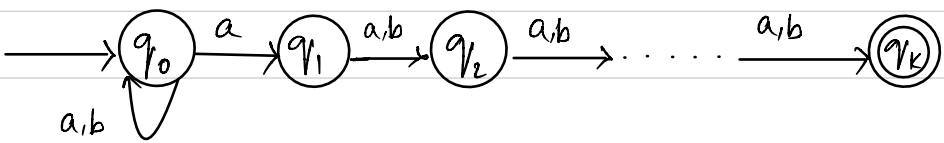
And if $A \rightarrow$ regular $\Rightarrow A_{\frac{1}{3}-\frac{1}{3}}$ might not be regular.

Hence proved.

48.

1.60 Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. 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 $C_k = \Sigma^*\mathtt{a}\Sigma^{k-1}$. Describe an NFA with $k+1$ states that recognizes C_k in terms of both a state diagram and a formal description.

The state diagram of NFA-N that consists of $k+1$ states and recognizes language C_k is given as:



$N = (Q_k, \Sigma, \delta, q_0, F)$

$Q_k = \{q_0, q_1, \cdots q_k\}$

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

$$\delta(q_i, a) = \begin{cases} \{q_0, q_1\}, & i = 0 \\ \{q_{i+1}\}, & 0 < i < k \\ \emptyset, & i = k \end{cases}$$

$$\delta(q_i, b) = \begin{cases} \{q_0\}, & i = 0 \\ \{q_{i+1}\}, & 0 < i < k \\ \emptyset, & i = k \end{cases}$$

$F = q_k$

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 2^k states.

49.

In 1.60, $C_k = \Sigma^* a \Sigma^{k-1}$ for $k \geq 1$. $\Sigma = \{a, b\}$.

Consider two strings x, y st. xz and yz have the same suffix - z. Assume x and y are unique $\Rightarrow$ DFA reaches different states on reading x, y. Let's say that x, y are of length k $\Rightarrow x = x_1 x_2 \cdots x_k$, $y = y_1 y_2 \cdots y_k$. Since they are unique, $\exists i$ st $x_i \neq y_i$, $0 < i \leq k$. $\Rightarrow x_i = a$ or b and $y_i = b$ or a respectively.

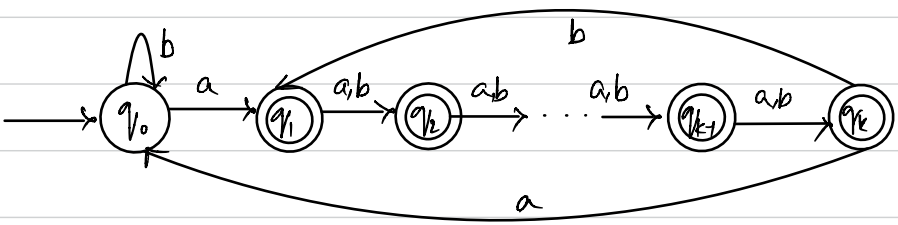
If we were to take the suffix string, $z = b^{i-1}$ then it forces one of xy or yz to have 'a' at the k^{th} bit from the end.

$\Rightarrow$ Total number of strings of length k over $\{a, b\}$ is 2^k. Hence the DFA must have 2^k in order to be able to distinguish between 2^k possible strings.

50.

1.62 Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. 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 $D_k = \Sigma^*\mathtt{a}(\Sigma \cup \varepsilon)^{k-1}$. Describe a DFA with at most $k+1$ states that recognizes D_k in terms of both a state diagram and a formal description.

A DFA with almost k+1 states that recognizes language D_k can be drawn as:



$M = (Q_k, \Sigma, \delta, q_0, F)$

$Q_k = \{q_0, q_1, \cdots q_k\}$

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

$$\delta(q_i, a) = \begin{cases} q_1, & i = 0 \\ q_1, & i \neq 0 \end{cases}$$

$$\delta(q_i, b) = \begin{cases} q_0, & i = 0 \\ q_{(i+1)\%k}, & i \neq 0 \end{cases}$$

$F = \{q_1, q_1, \cdots q_k\}$

51.

*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 $B_n = A_1 \cup \cdots \cup A_k$, for regular languages A_i, then at least one of the A_i requires a DFA with exponentially many states.

a. Proof by induction:

$n = 1$

$B_n = \{\varepsilon, 0, 1\}$, $N = (\{q_0\}, \Sigma, \delta, q_0, \{q_0\})$

where $\delta(q_0, \varepsilon|0|1) = q_0$

$n = k$

let $n = n_1 + n_2 \Rightarrow n_1, n_2 < n$

let C, D regular exprn of length n_1 n_2 respectively divide B_n. Wkt the NFA of C, D consist of at least n_1, n_2 states respectively.

∴ B_n is recognized by a NFA with n states.

b. From the above result wkt B_n, $n>0$ is recognizable by NFA with n states.

$B_n = A_1 \cup A_2 \cup \ldots A_k$ where $A_i \rightarrow$ regular.

An equivalent DFA constructed from the above NFA will have a min of n states and a max of 2^n states. Since every A_i is regular,

they can all be represented by DFA's.
From the pigeon hole principle we can say that at least one DFA that is constructed from A_i contains 2^i states.

52.

*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 $CUT(A) = \{yxz \mid xyz \in A\}$.

a. Exhibit a language B for which $CUT(B) \neq CUT(CUT(B))$.

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

a. We can show it with a help of an example.

let $B = \{0^n 1^n \mid \{0,1\} \in \Sigma^*, n \geq 0\}$

$S = xyz = 0^n 1^m 1^{n-m} \Rightarrow CUT(B) = 1^m 0^n 1^{n-m}$

and $CUT(CUT(B)) = 0^n 1^n$

$\Rightarrow CUT(B) \neq CUT(CUT(B))$

b. let $M = (Q, \Sigma, \delta, q_0, F)$ and $M' = (Q', \Sigma, \delta', q_0', F')$ be two DFA's that recognize A, $CUT(A)$ respectively.

If M' does recognize $CUT(A) \Rightarrow CUT(A) \rightarrow$ regular.

$\therefore M' = (Q', \Sigma, \delta', q_0', F')$ where,

$Q' = Q$

Σ = set of alphabets

δ' : $\delta'(q', a) = \begin{cases} \delta(q, a), & q \in Q \text{ and } q \notin F \\ \delta(q, a), & q \in F \end{cases}$

q_0' = start state

$F' = F$ ∵ $w \in A \rightarrow A \Rightarrow w \in xyz$, $yxz \in CUT(A) \rightarrow M'$

∵ DFA M' recognizes $CUT(A)$, it is regular.

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

a. Show that for each 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.

53

a. Consider 2 unique k-bit strings x, y

$x = x_1 x_2 \cdots x_k$, $y = y_1 y_2 \cdots y_k$ st $\exists i, x_i \neq y_i, 0 < i \leq k$.

Consider a suffix string z st, $z = 0^{i-1}$.

$\Rightarrow$ one of xz or yz must have the k^{th} bit from the end as 1 and the other as 0.

Since there are 2^k such possible strings of length k, a DFA that recognizes it must have 2^k states.

b. $\overline{WW_k} = \{ww \mid w \in \Sigma^* \text{ and } |w| < k\}$

$\therefore$ a NFA consisting of $k+1$ states can be built to recognize $\overline{WW_k}$.

let it be describes as $N(Q, \Sigma, \delta, q_0, F)$ where,

$Q = \{0, 1, \cdots k\}$, $\delta(0,0) = 0$, $\delta(0,1) = 1$, $\delta(i-1, 0|1) = i$ for $2 \leq i < k$, $q_0 = 0$ and $F = \{k\}$.

$\therefore$ The traversal starts at 0, followed by 1 and $k-3$ 1's to form the shorted recognized string. $\therefore$ It requires a min length of $k-1$ bits.