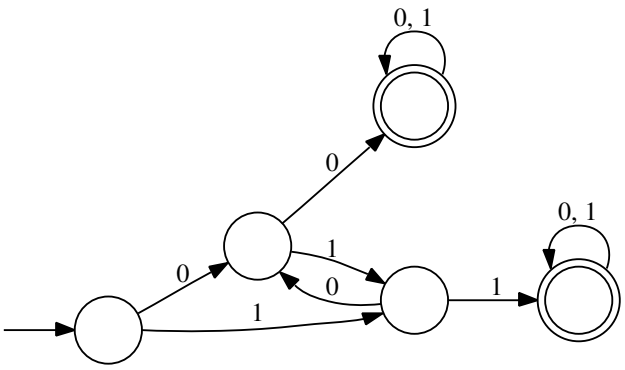
Solutions to Written Assignment 1

1. Consider the following deterministic finite automaton (DFA) over the alphabet $\Sigma = \{0, 1\}$.



Give a one-sentence description of the language recognized by the DFA. Write a regular expression for this language.

Solution:

- All strings that contain two consecutive 0s or two consecutive 1s.
- $(0 + 1)^*(00 + 11)(0 + 1)^*$

2. Consider the following languages over the alphabet $\Sigma = \{0, 1\}$.

- L_1: All strings that contain at least two 0s
- L_2: All strings that contain at least one 1
- L_3: All strings that contain at least two 0s and at least one 1
- L_4: All strings that contain at most one 0 or no 1s

Give DFAs for each of the languages L_1, L_2, L_3, and L_4.

Aside: This example illustrates that regular languages are closed under intersection and complementation. Note that $L_3 = L_1 \cap L_2$ and $L_4 = \Sigma^* - L_3$, where Σ^* represents the language containing all strings over the alphabet Σ.

Solution: See Figure 1.

3. Let E_3 be the language over the alphabet $\Sigma = \{a_1, a_2, a_3\}$ defined as follows.

E_3: All strings in which a_i occurs an even number of times for some $i \in \{1, 2, 3\}$

Give a non-deterministic finite automaton (NFA) for the language E_3.

Solution: See Figure 2.

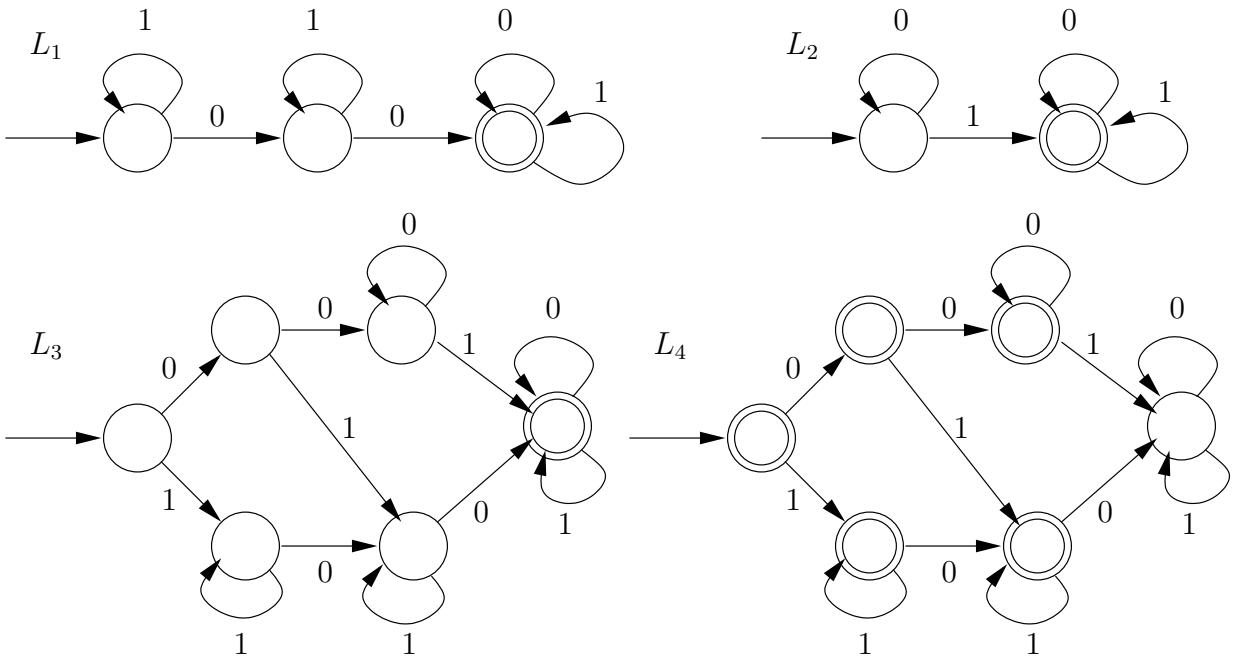


Figure 1: Example DFAs for Question 2.

4. Write regular expressions for the following languages over the alphabet $\Sigma = \{0, 1\}$:

   (a) All strings that contain at least one 0 and at least one 1 and that also end with at least two 1s.

   $$(0+1)^*0(0+1)^*11$$

   (b) All strings that do not begin with 01.

   $$\varepsilon + 0 + 1 + (00 + 10 + 11)(0+1)^*$$

   (c) All strings that contain an odd number of 1s.

   $$0^*10^*(0^*10^*10^*)^*$$

5. Give a DFA for each of the following languages over the alphabet $\Sigma = \{0, 1\}$.

   (a) The language of the NFA in Figure 3.
   (b) The language of the regular expression $(0 + 01)^*1^*$.

   Solution: See Figure 4.

6. Consider the string

   `aaabaabbababbb`

   and its tokenization

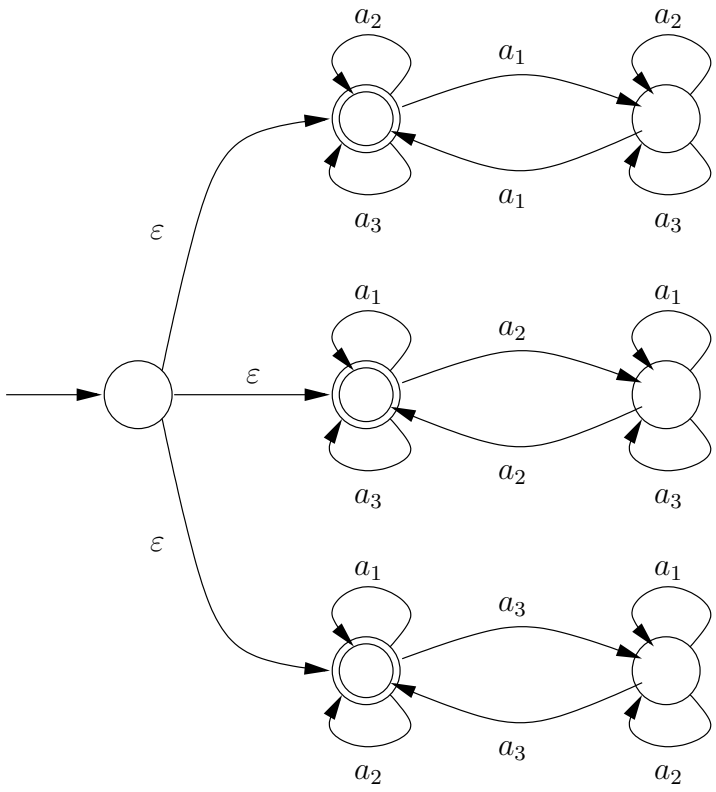


Figure 2: Example NFA for Question 3.

```
aa a b aabb a b a bbb
```

Give a flex specification with the minimum number of rules that produces this tokenization. Each flex rule should be as simple as possible as well. You may not use regular expression union (i.e., $R_1 + R_2$) in your solution. Do not give any actions; just assume that the rule returns the string that it matches.

Solution:

```
(aa)*b*
a
```

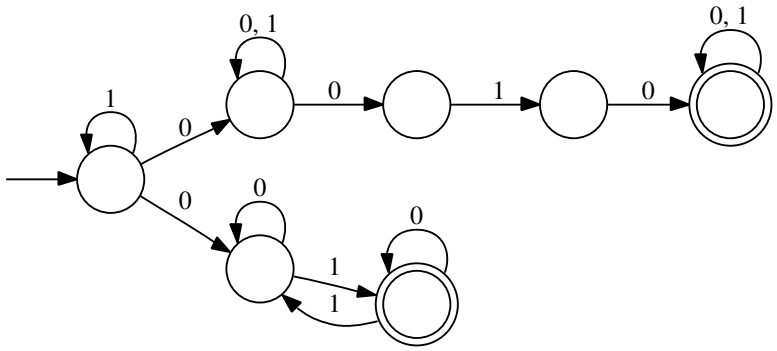


Figure 3: An NFA for Question 5.

(a)

(b)

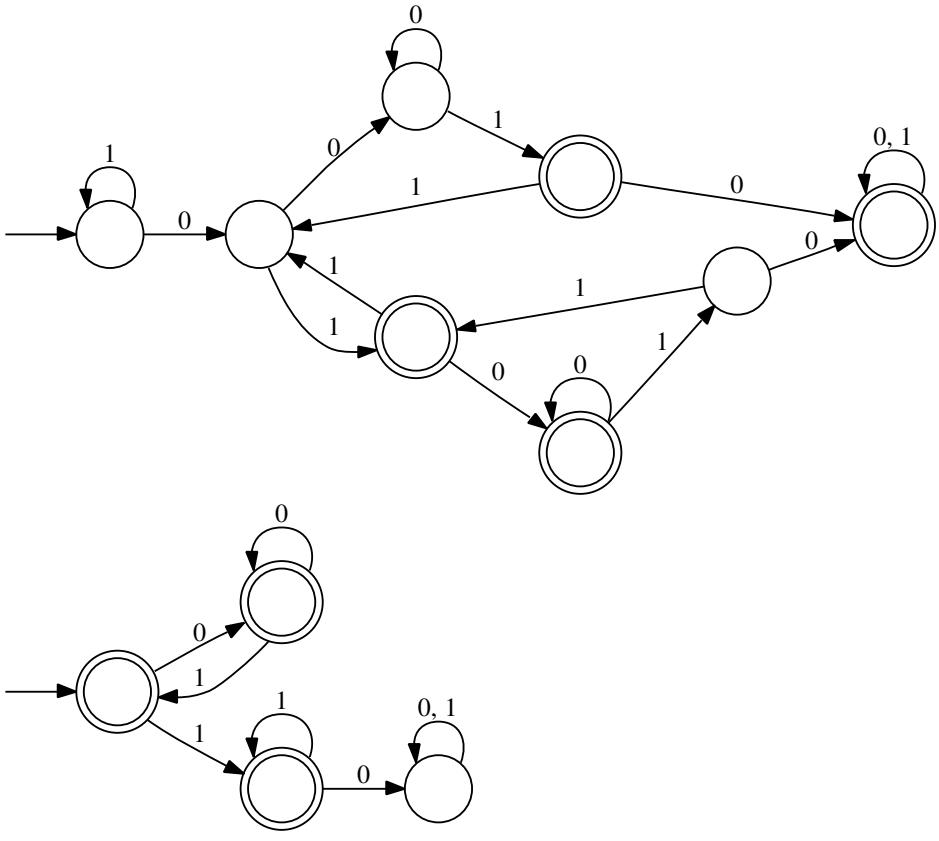


Figure 4: Example DFAs for Question 5.