Written Assignment I Solutions

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

   (a) All strings that do not end with aa.

   $$\epsilon + a + b + (a + b)^*(ab + ba + bb)$$

   (b) All strings that contain an even number of b's.

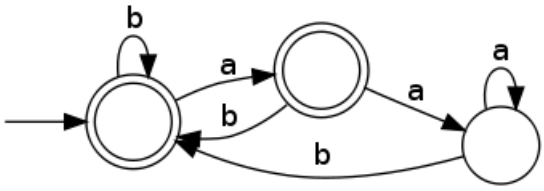
   $$a^*(ba^*ba^*)^*$$

   (c) All strings which do not contain the substring ba.

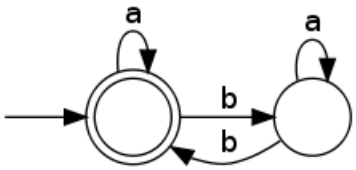
   $$a^*b^*$$

2. Draw DFAs for each of the languages from question 1. None of your DFAs may contain more than 4 states.

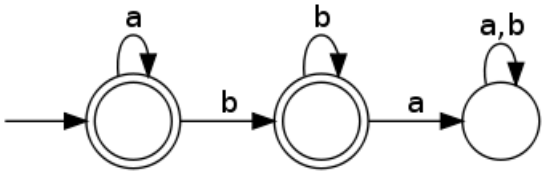
   (a)



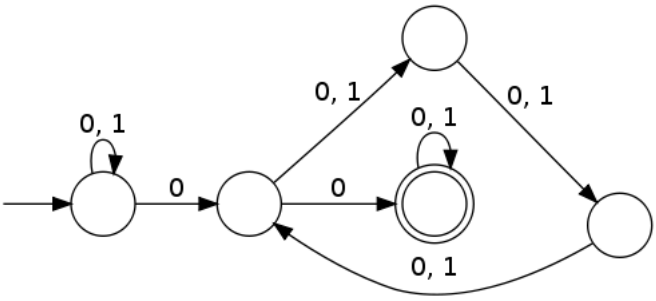
   (b)



   (c)



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



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

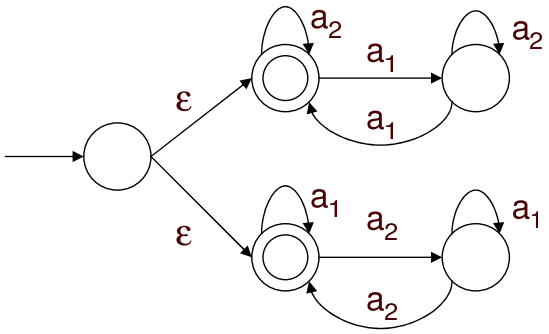
- The NFA recognizes all strings that contain two 0's separated by a substring whose length is a multiple of 3.

- A regular expression for this language is $(0+1)^*0((0+1)(0+1)(0+1))^*0(0+1)^*$.

4. Let $\Sigma_m = \{a_1, \ldots, a_m\}$ be an alphabet containing m elements, for some integer $m \geq 1$. Let L_m be the following language:

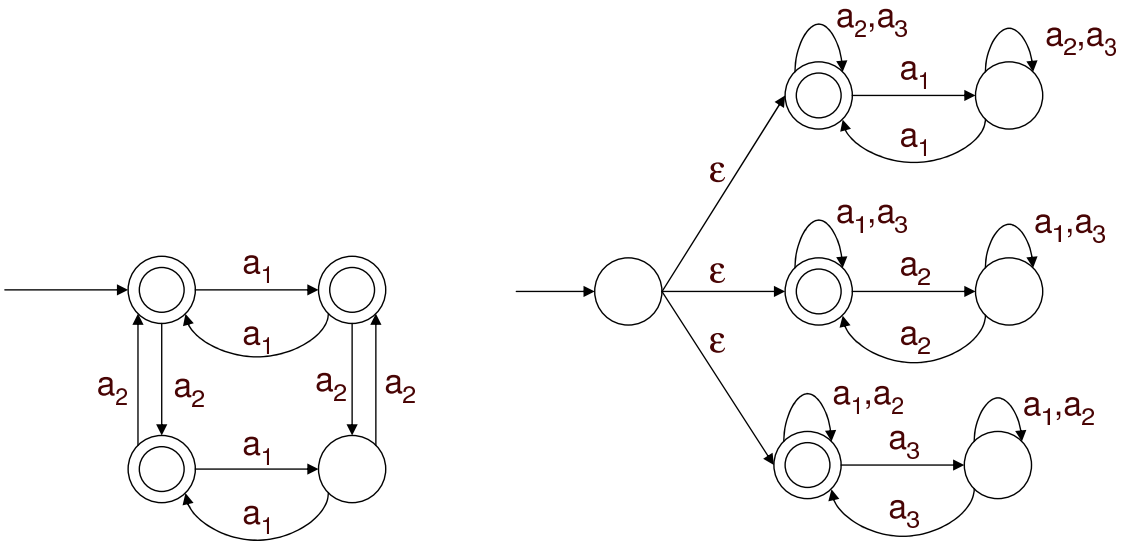
All strings in which at least one a_i occurs an even number of times (not necessarily consecutively), where $1 \leq i \leq m$.

The following figure shows an NFA for the language L_2.



Construct a DFA for the language L_2 that has at most 6 states. Also construct an NFA for the language L_3 that has at most 7 states.

Aside: Non-deterministic finite automata (NFAs) are no more powerful than DFAs in terms of the languages that they can describe. However, NFAs can be exponentially more succinct than DFAs, as this problem demonstrates. For the language L_m, there exists an NFA of size at most $2m+1$ while any DFA must have size at least 2^m. Note that the DFA for the language L_3 is not as easy to construct as the NFA for the language L_3.



DFA for L_2 NFA for L_3

5. Consider the string

```
abaabaababababbaaabaab
```

and its tokenization

```
abaa ba ababa bba aa ba a b
```

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., $R1 + R2$) in your solution. Do not give any actions; just assume that the rule returns the string that it matches.

- `(ab)*a+`
- `b*a?`