Exercise 1. Draw FAs that recognize each of the following languages using the specified number of states. In all cases, the alphabet is $\Sigma = \{0, 1\}$.

Tip: using NFAs is easier!

(a) $\{w \in \Sigma^* \mid w \textit{ ends with } 11\}$ using three states.

Solution:

(b) $\{w \in \Sigma^* \mid w \textit{ contains the substring } 1010\}$ using five states.

Solution:

(c) $\{w \in \Sigma^* \mid w \textit{ contains exactly two } 0s, \textit{ or at least two } 1s\}$ using six states.

Solution:

Exercise 2. Convert each of the FAs you built in Question 1 to regular expressions (just write the regular expressions, no need to show the conversion steps).

(a) **Solution:**

(b) **Solution:**

(c) **Solution:**

Exercise 3. Prove (or disprove), **using closure properties** of Regular Languages or Finite Automata, that the language L_F (described below) is Regular.

We will describe language L_F the following way:

- start with the set of all the binary words that have any zeroes occur before any ones;
- then remove, from that set, the subset of words that have exactly two ones;
- then remove, from that set, the subset of words that have an even number of zeros and no ones;

Solution:

Exercise 4. Show that the language L_G (described below) is Regular by **1) constructing a Regular expression** and **2) one finite automaton**.

We will describe language L_G the following way:

- start with all the binary words that have all the zeroes before all the ones;
- add, to that set, all the words that have exactly two ones;
- then add, to that set, all the words that have an even number of zeros;

Solution:

Exercise 5. This Exercise is Optional!!

In lecture, we demonstrated a process for converting any NFA to a (purportedly equivalent) DFA. To complete the proof that these machines are equivalent (thereby proving that NFAs and DFAs are equally powerful), prove that the resulting DFA accepts exactly the same language as the original NFA.

Hint: try induction on the length of the word!

References

[1] Sipser, Michael. *Introduction to the Theory of Computation.* Course Technology, 2005. ISBN: 9780534950972

[2] Critchlow, Carol and Eck, David *Foundation of computation.*, Critchlow Carol, 2011