CFL Homework 3 (John Pham: john.pham@kcl.ac.uk, student number 1629417)

1. The regular expression matchers in Java, Python & Ruby can be very slow with some (basic) regular expressions. What is the main reason for this inefficient computation?

   Translating a regular expression into a Non-Deterministic Finite Automata is the common way for languages such as Java, Ruby and Python to perform matching on regular expressions. NFAs grow in size linearly with respect to the size of the regular expression. However, as NFAs are non-deterministic, the problem of finding out whether a string is accepted or not, might not be computationally cheap. This is because in a certain state, there can be more than one "next" state given an input. In the implementation of Java etc. depth first search is used as the mechanism for determining if a string is accepted, as such, if a wrong decision is made, the language has to backtrack and explore all potential candidates.

2. What is a regular language? Are there alternative ways to define this notion? If yes, give an explanation why they define the same notion.

   A regular language is a language that can be represented with a regular expression. We are able to convert a regular expression into a NFA and DFA using Thompsons' and Subset construction, and similarly, from a DFA back to a Regular expression using Brzozowski's method. We can therefore say that automatas and regular expressions can recognise the same set of languages. That is:

   A language is regular if and only if there exists a regular expression that recognises all its strings

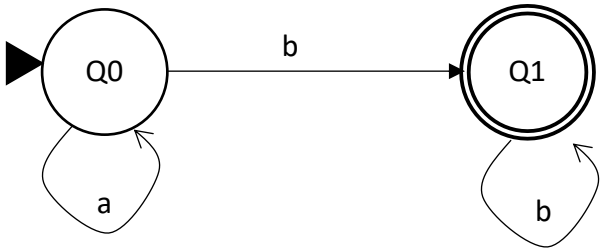
3. Why is every finite set of strings a regular language?

   If a set of strings can be defined with a regular expression, then it is considered to be *regular*. We know from basic regular expressions of the ALT (+) operator which gives us a choice between two regular expressions r1, and r2. As long as the set of strings is finite, it will always be possible (although naïve) to express each of these strings as a choice.

4. Assume you have an alphabet consisting of the letters a, b, and c only. (1) Find a regular expression that recognises the two strings ab and ac. (2) Find a regular expression that matches all strings except these two strings.

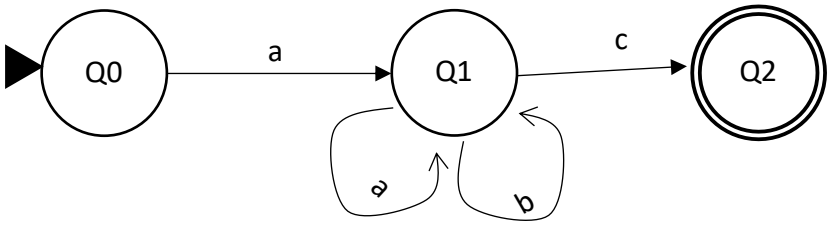
   (1) $a \bullet (b + c)$
   (2) $1 + (a + b + c) + (a \bullet a) + ((b + c) \bullet (a + b + c) \bullet (1 + a + b + c))$

5. Given the alphabet {a, b}. Draw the automaton that has two states, say Q0 and Q1. The starting state is Q0 and final state is Q1. The transition function is given by:
   (Q0, a) -> Q0
   (Q0, b) -> Q1
   (Q1, b) -> Q1
   What is the language recognised by this automaton?



   any string that starts with the letter b

6. Give the NFA that can recognise the language L(a . (a+b)* . c)



7. Given a DFA define which language is recognised by this automaton. Can you define also the language defined by a NFA?

   We lift the transition function (delta) from characters to strings:

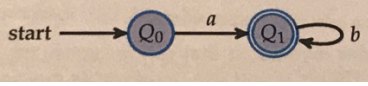
   Delta-hat(q, []) = q
   Delta-hat(q, c::s) = Delta-hat(Delta-hat(q, c), s)

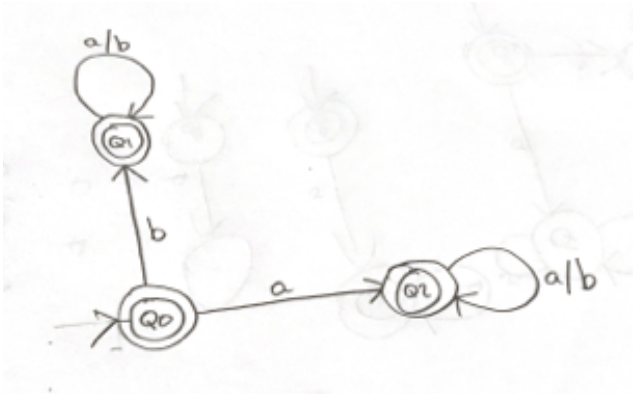
   A string s is in the language accepted by the automaton A if and only if:

   Delta-hat(Q0, s) is in the set of final states F

8.

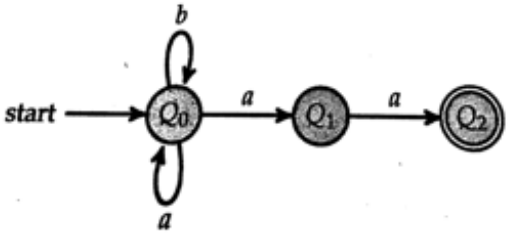
8. Given the following deterministic finite automaton over the alphabet $\{a, b\}$, find an automaton that recognises the complement language. (Hint: Recall that for the algorithm from the lectures, the automaton needs to be in completed form, that is have a transition for every letter from the alphabet.)



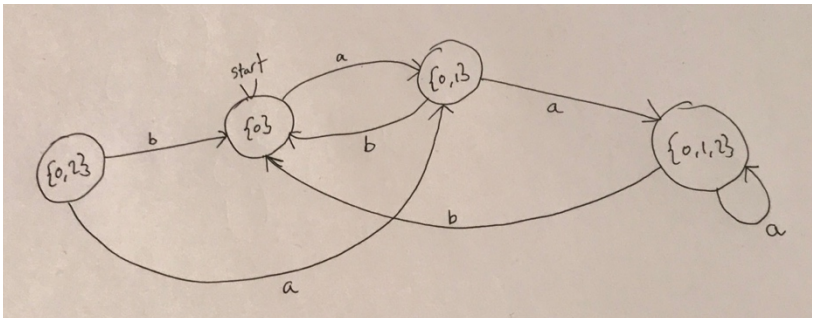


9.

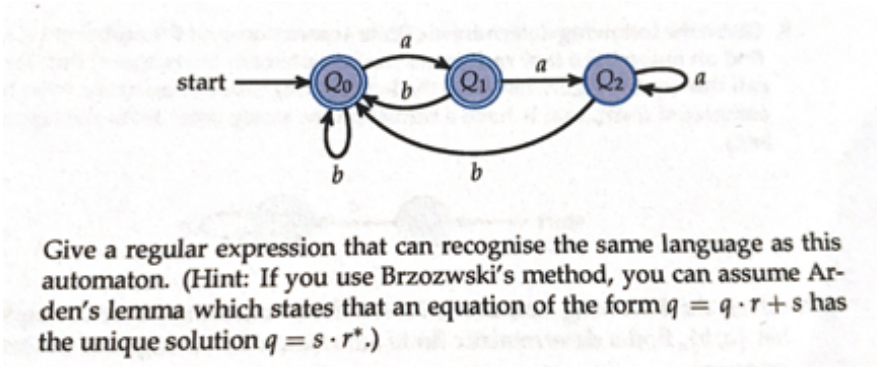
9. Given the following non-deterministic finite automaton over the alphabet $\{a, b\}$, find a deterministic finite automaton that recognises the same language:



nodes	a	b
{}	{}	{}
{0}	{0, 1}	{0}
{1}	{2}	{}
{2}	{}	{}
{0, 1}	{0, 1, 2}	{0}
{0, 2}	{0, 1}	{0}
{1, 2}	{2}	{}
{0, 1, 2}	{0, 1, 2}	{0}



11. Given the following DFA over the alphabet {a, b}:



Give a regular expression that can recognise the same language as this automaton. (Hint: If you use Brzozwski's method, you can assume Arden's lemma which states that an equation of the form $q = q \cdot r + s$ has the unique solution $q = s \cdot r^*$.)

((b + ab + aa(a*)b)*) + ((b + ab + aa(a*)b)*a)

12. If a NFA has n states. How many states does a DFA that can recognise the same language as the NFA maximal need?

If an NFA has n states, then a DFA will (in the worst case scenario) an exponential (2^n) number of states as per the subset construction.