CS46 practice problems 3

These practice problems are an opportunity for discussion and trying many different solutions. They are **not counted towards your grade**, and **you do not have to submit your solutions.** The purpose of these problems is to get more comfortable with NFAs and regular expressions. I recommend trying to solve these problems on paper *first*, then trying with the online tool. Once you are ready to test your solutions, the Automata Tutor site will give you troubleshooting feedback.

Note that Automata Tutor uses slightly different notation for regular expressions.

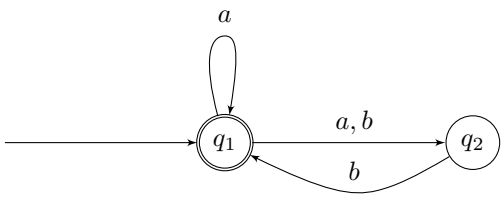
1. (a) Construct a DFA for the language $\{w \mid w$ begins with a and ends with $b\}$.
   (b) Construct an NFA for the language $\{w \mid w$ begins with a and ends with $b\}$. (Try to use nondeterminism so that it has fewer states than the previous DFA for the same language.)

2. Construct an NFA with *three states* that recognizes the language

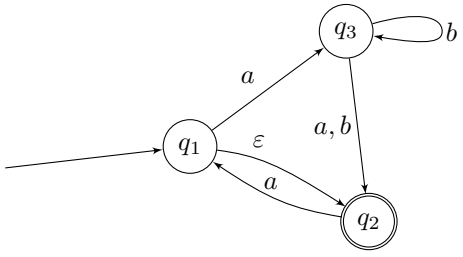
$$\{w \mid w \text{ ends with } bb\}$$

   . You will need to use nondeterminism!

3. Construct a DFA equivalent to the following NFA:



4. Construct a DFA equivalent to the following NFA:



5. Construct an NFA for the language $L = L((aa \cup ab)^*)$ over the alphabet $\Sigma = \{a, b\}$.

6. Construct an NFA for each of the languages given by regular expressions:

   (i) `a(abb)`*`∪b`
   (ii) `a`$^+$`∪(ab)`*
   (iii) `(a∪b`$^+$`)a`$^+$`b`$^+$

7. Construct a regular expression for the language Σ^* over $\Sigma = \{a, b\}$.

8. Construct a regular expression for the language $\{w \mid w \text{ contains the substring } ab\}$ over $\Sigma = \{a, b\}$.

9. Construct a regular expression for the language $\{aa, abba\}$ over $\Sigma = \{a, b\}$.

10. Construct a regular expression for the language $\{w \mid w \text{ contains exactly two } a\text{s}\}$ over $\Sigma = \{a, b\}$.

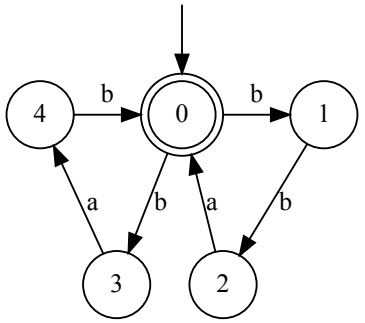
11. Let $L = \{w \mid \text{every appearance of } c \text{ in } w \text{ is in a contiguous substring with at least two other } c\text{s}\}$ over $\Sigma = \{a, b, c\}$. So for example, L contains *baaccca*, *aaab*, ε, and *cccca*. L does *not* contain *abcccbaca*, *bccb*, *c*, or *cabaaaabcc*.

    (a) Construct a DFA that recognizes L.

    (b) Construct an NFA that recognizes L.

    (c) Construct a regular expression that recognizes L.

12. Construct a DFA equivalent to the following NFA:



    Write a regular expression for the language accepted by this machine.

These problems are for discussion in small groups. You should start with problem 13 and then do one of either 14 or 15 (they are both quite complicated).

13. **Does the alphabet matter?**

    A palindrome is a string that reads the same forwards and backwards. Consider

    $$L = \{w \in \Sigma^* \mid w \text{ is a palindrome}\}$$

    - When $\Sigma = \{a, b, c\}$, is L regular?
      If yes, then prove it (directly, using a construction); if no, then disprove it (using the pumping lemma).
    - When $\Sigma = \{a\}$, is L regular?
      If yes, then prove it (directly, using a construction); if no, then disprove it (using the pumping lemma).

14. **C++ comments**

    In C++, it is possible to write a comment as a string beginning with '`/*`' and ending with '`*/`'. In between, it can contain any characters, including '`/`' and '`*`', as long as it does *not* contain the substring '`*/`' before the end of the string. **For the sake of clarity in this problem, we will substitute '#' for the '*' character, so that it cannot be confused with the Kleene star operator.**

    For example, '`/#ab/ab/c/b####ac#/`' is a valid comment, but '`/#cab/baaaa###/a//bb#/`' is not a valid comment.

    Construct a regular expression for the language L of this style of well-formed C++ comments over the alphabet $\Sigma = \{a, b, c, \#, /\}$.

    $$L = \{w \mid w = \texttt{/\#}x\texttt{\#/}, \text{ where } x \in \Sigma^* \text{ does not contain the substring '\#/'}\}$$

    You should explain your reasoning/show your work. A formal proof is not required, but you should be able to justify (at a high level) why your regular expression is correct.

15. (Sipser 1.66) A **homomorphism** is a function $f : \Sigma \to \Gamma^*$ from one alphabet to strings over another alphabet. We can extend f to operate on strings by defining $f(w) = f(w_1)f(w_2)\cdots f(w_n)$ where $w = w_1 w_2 \cdots w_n$ and each $w_i \in \Sigma$. We further extend f to operate on languages by defining $f(\epsilon) = \epsilon$ and $f(A) = \{f(w) \mid w \in A\}$, for any language A.

    (a) Show, by giving a formal construction, that the class of regular languages is closed under homomorphism. In other words, given a DFA M that recognizes B and a homomorphism f, construct a finite automaton M' that recognizes $f(B)$.
    You need to provide an argument why your construction recognizes the right language.
    Once you're done, consider the machine M' that you constructed. Is it a DFA in every case?

    (b) Show, by giving an example, that the class of non-regular languages is not closed under homomorphism.