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\footnote{The Automata Tutor site} 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. (Sipser 1.7a) Construct an NFA with three states that recognizes the language \( \{w \mid w \text{ ends with } 00\} \) over \( \Sigma = \{0, 1\} \). You will need to use nondeterminism!

2. (Sipser 1.7f) Construct an NFA with three states that recognizes the language \( 1^*(001^+)^* \) over \( \Sigma = \{0, 1\} \). You will need to use nondeterminism!

3. Let \( L = \{w \mid w \text{ begins with } 1 \text{ and ends with } 0\} \) over \( \Sigma = \{0, 1\} \). 
   - Construct a DFA that recognizes \( L \).
   - Construct an NFA that recognizes \( L \). (This NFA should have fewer states than your DFA for the same language.)

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

5. Construct a regular expression for the language \( \{aa, abba\} \) over \( \Sigma = \{a, b\} \).

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

7. (Sipser 1.28a) Construct an NFA for each of the languages given by regular expressions:
   - (i) \( a(abb)^* \cup b \)
   - (ii) \( a^+ \cup (ab)^* \)
   - (iii) \( (a \cup b^+)a^+b^+ \)

8. Let \( L = \{w \mid \text{every appearance of } c \text{ in } w \text{ is in a contiguous substring with at least two other } cs\} \) over \( \Sigma = \{a, b, c\} \). So for example, \( L \) contains \( baaceca, aaab, \epsilon, \) and \( cceca \). \( L \) does \textit{not} contain \( abccccbaca, bccb, c, \) or \( cabaaaabcc \).
   - Construct a DFA that recognizes \( L \).
   - Construct an NFA that recognizes \( L \).
   - Construct a regular expression that recognizes \( L \).

\footnote{If you are stumped or looking for guidance, some of these problems are in the selected solutions in chapter 1.}
If you finish the Automata Tutor problems, here are some discussion problems for your group.

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

10. **Regular languages in real life!**
    We’ve seen several different characterizations (DFAs, NFAs, regular expressions) of regular languages. Our examples have usually been over fairly small alphabets, for the sake of readability. However, regular languages capture many actual problems from the broader CS sphere (and beyond). Can you think of an application\(^2\) where recognizing (or generating) a regular language would be useful?
    Brainstorm some creative/useful applications of regular expressions with your group.

\(^2\) Freebie example which doesn’t count: **bash** regular expressions, since they allow more operations than the regular expressions we’ve been discussing.