CS46 Homework 3

This homework is due at 10:00PM on Sunday, February 16. Write your solution using $\LaTeX$. Submit this homework using **github**. This is a **10 point** homework.

This is an individual homework. It's ok to discuss approaches at a high level. In fact, I encourage you to discuss general strategies. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else. Your write-up is your own. If you use any out-of-class references (anything except class notes, the textbook, or asking Lila), then you **must** cite these in your post-homework survey. Please refer to the course webpage or ask me any questions you have about this policy.

The main **learning goal** of this homework is to develop the skills to design, understand, and analyze NFAs and regular expressions.

Part 1 — These problems should be completed and submitted[1] on Automata Tutor. You are allowed three attempts at each problem. I recommend that you *first* try to solve the problems on paper, *then* use the site to debug your solutions.

1. Let $\Sigma = \{a, b\}$, let $L_1 = \{w \mid \text{the length of } w \text{ is even}\}$, and let $L_2 = \{w \mid w \text{ begins and ends with } a\}$.
   (a) Construct an NFA for the language $L = L_1 \cup L_2$.
   (b) Construct an NFA for $L_1 \circ L_2$.

2. Write a regular expression for the language

   $$\{w \mid w \text{ contains exactly two } a\text{s or at least two } b\text{s}\}$$

   over the alphabet $\Sigma = \{a, b\}$.

   Ponder the fact that union is pretty easy with NFAs and with regular expressions.

3. Let $\Sigma = \{a, b\}$ and let $L = \{w \mid w \text{ contains an even number of } a\text{s and an odd number of } b\text{s}\}$.
   (a) Construct an NFA recognizing L^*.
   (Hint: You can use the construction of Theorem 1.49 to get a correct answer, but this NFA will be capable of being simplified. Try describing L^* in English first.)
   (b) Think for a little while about a regular expression for L. Ponder the fact that intersection was much easier in automata than in regular expressions.
   (If you choose to, you can try to submit this regular expression on Automata Tutor, but because of the number of fiddly little details and the lack of useful feedback on the site, it is **not required**.)
   You might try to approach this problem by starting with a DFA and converting it to a regular expression, or by coming up with a regular expression based on your intuition about what sort of strings are in this language.

[1] If you want to use late days on this assignment, you will need to submit solutions to these problems via github. The automatatutor site has only one deadline.

4. Let $\Sigma = \{0, 1\}$ and $L = \{w \mid w$ contains the substring $0ab0$ or $1ab1$ where $a, b \in \Sigma\}$.

   (a) Construct a regular expression for L.

   (b) Construct a DFA that recognizes L.
   (It is *strongly* recommended that you plan on paper first!)
   For fun: Take a screenshot of your masterpiece, email it to Lila, treasure it forever.

   (c) Construct an NFA that recognizes L.
   Your NFA should have substantially fewer states than your DFA. (Phew!)

5. Let $L = \{w \mid$ if w contains an a, then it contains at least three as in a row$\}$ over $\Sigma = \{a, b\}$. So for example, L contains $abaaa$, $bbaabaaabaa$, ε, and bb. L does *not* contain baa, $abaaba$, or $bbba$.

   - Construct a DFA that recognizes L.
   - Construct an NFA that recognizes L.
   - Construct a regular expression that recognizes L.

Part 2 — These problems should be typeset in LaTeX and submitted using **github**.

6. **Understanding regular expressions.**

   For each of the following regular expressions over $\Sigma = \{a, b\}$, explain in English what language they describe. (Show your thought process.)

   (a) $b^*a(b^*a^*)^*$

   (b) $(a^*\emptyset b \cup ab \cup b^*\emptyset^*a)(b \cup \emptyset)$

   (c) $\epsilon \cup a(ba)^* \cup b(ab)^*$

7. **Writing regular expressions.**

   Give regular expressions for the languages accepted by NFAs M_1 and M_2. (**Hint:** use Lemma 1.60.)

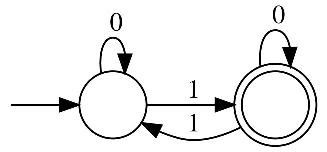

Figure 1: NFA M_1.

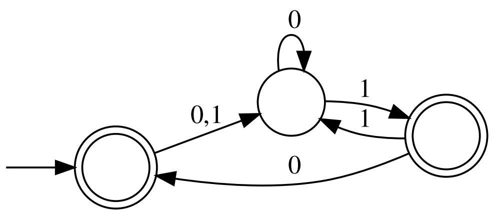

Figure 2: NFA M_2.

8. Consider the language $C = \text{op}(A, B)$ where "op" is some operation that regular languages are closed under. Suppose we know the following about A and C. What, if anything, can we conclude about B?

   (You should support your answer with a brief explanation.)

(a) A is regular and C is regular.
(b) A is regular and C is not regular.
(c) A is not regular and C is regular.
(d) A is not regular and C is not regular.

9. **Regular expression identities.**

   Let R and S be regular expressions. Prove or disprove the following "identities". To prove the identity, you must argue that a string in the language defined on the left-hand side is in the language defined on the right-hand side, and vice versa. To disprove the identity, you must give a small counterexample string with real examples of regular expressions for R and S.

   (a) $(R^*)^* = R^*$
   (b) $(R \cup S)^* = R^* \cup S^*$
   (c) $(R^* S^*)^* = (R \cup S)^*$

10. **Yes, but does the alphabet *really* matter? (extra credit)**

    We saw in lab that the language $L = \{w \mid w \text{ is a palindrome}\}$ is non-regular for some alphabet, but is regular for $\Sigma = \{a\}$. For this problem, let's set $\Sigma = \{a\}$.

    (a) Argue why there must exist at least one language $L \subseteq \Sigma^*$ which is not regular.
    (b) Give an example of a non-regular language $L \subseteq \Sigma^*$.

11. **Are "regular expressions" regular? (self-referentially cool, *definitely* extra credit)**

    We have a set of rules that describe how to build regular expressions, given an alphabet Σ and some additional symbols, '$\emptyset$', '$\cup$', '$\circ$', '*', '(', and ')'. Consider the language L over the alphabet given by Σ with these added symbols, defined as:

    $$L = \{w \mid w \text{ is a regular expression over } \Sigma\}$$

    Prove that L is not regular.

    (Thinking dizziness warning: be careful not to think yourself in circles with this one! A cool corollary of this claim is that there is no regular expression that matches all regular expressions.)