

## Homework 3 – Due Friday, February 13, 2026 at 11:59 PM

**Reminder** Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to the course staff if asked. You must also identify your collaborators and write “Collaborators: none” if you worked by yourself. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden. Collaboration is not allowed on problems marked “INDIVIDUAL.”

**Problems** There are 8 required problems and 1 bonus problem. Problem 4 will be autograded by AutomataTutor.

1. (**Set difference**) Given languages  $A$  and  $B$  over the same alphabet  $\Sigma$ , define their “set difference” to be the language  $A \setminus B = \{w \in \Sigma^* \mid w \in A \text{ but } w \notin B\}$ .

(a) Explain how to write the set difference operation in terms of (some of the) union, concatenation, star, reverse, intersection, and complement operations. That is, describe how for arbitrary  $A, B$  one can obtain  $A \setminus B$  by applying these other operations to  $A$  and  $B$ .

(b) Use part (a) to show that the regular languages are closed under set difference.

2. (**Closure under repetition**) Let  $\Sigma$  be a finite alphabet. For a string  $w = w_1 \dots w_n$ , where each  $w_i \in \Sigma$ , define  $\text{Rep}(w) = w_1 w_1 w_2 w_2 \dots w_n w_n$ . That is,  $\text{Rep}(w)$  is the string obtained by repeating each character in  $w$  one additional time. Given a language  $L \subseteq \Sigma^*$ , define the language  $\text{Rep}(L) = \{\text{Rep}(w) \mid w \in L\}$ . For example, if  $L_1 = \{a, ab\}$ , then  $\text{Rep}(L_1) = \{aa, aabb\}$ , and if  $L_2 = \{a^n \mid n \geq 0\}$  then  $\text{Rep}(L_2) = \{a^{2n} \mid n \geq 0\}$ .

Show that the class of regular languages is closed under Rep.

3. (**Regex to description**) Give plain English descriptions of the languages generated by each of the following regular expressions. (No explanations required.)

(a)  $(0 \cup 1)((0 \cup 1)(0 \cup 1)(0 \cup 1))^*$

(b)  $(a \cup b)^* \cup (b \cup c)^*$

(c)  $\varepsilon^*$

4. (**Regular expressions vs. finite automata**) Please log on to AutomataTutor to submit solutions for this question.

(a) (**Description to regex**) Give regular expressions generating the following languages:

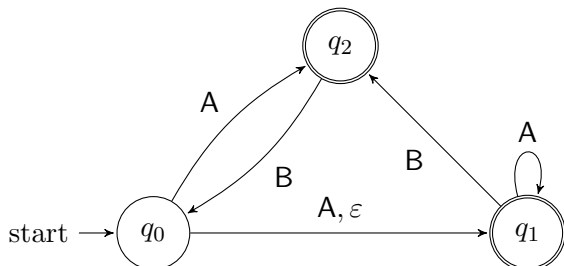
i.  $\{w \in \{0, 1\}^* \mid w \text{ has exactly one 0 and at least two 1's}\}$ .

ii.  $\{w \in \{0, 1\}^* \mid w \text{ is any string except for } 00 \text{ or } 000\}$ .

iii.  $\{w \in \{0, 1\}^* \mid w \text{ starts with 0 and has even length or } w \text{ starts with 1 and has odd length}\}$ .

(b) (**Regex to NFA**) Use the procedure described in class (also in Sipser, Lemma 1.55) to convert  $((A \cup C)^*(A \cup G)^* \cup CG)^*$  to an equivalent NFA. We recommend simplifying your NFA before entering it into AutomataTutor.

- (c) (**NFA to regex**) Convert the following NFA (over alphabet  $\{A, B\}$ ) to an equivalent regular expression.



5. (**The fault in our stars**)

- (a) If  $A$  and  $B$  are *finite* languages, what is the maximum cardinality of  $A \cup B$ ? What is the maximum cardinality of  $A \circ B$ ? Briefly explain your answers.
- (b) A *star-free regular expression* is a regex using only the symbols  $\emptyset, \varepsilon, \cup, \circ$ , and alphabet symbols in  $R$ . The *size* of a star-free regex is the number of such symbols it contains. (That is, do not count parentheses.) For a natural number  $k$ , let  $S(k)$  denote the maximum number of elements in the language generated by a star-free regex  $R$  of size  $k$ . Prove by strong induction on  $k$  that  $S(k) \leq 2^{2^k}$ .
- (c) Show that a language is finite if and only if it is generated by a star-free regex.

6. (**Regex complement**) Describe an algorithm that, given a regex  $R$ , constructs a regex  $S$  such that  $L(S) = \overline{L(R)}$ . You may assume you have access to functions such as `RegexToNFA`, `NFAToRegex`, `NFAToDFA`, etc. implementing the various constructions we've seen in class. For example, `NFAToDFA` takes as input an NFA and uses the subset construction to return an equivalent DFA. You can assume these functions handle low-level details like parsing parentheses for you. You can also use functions performing the other constructions we've seen in class; just either give a precise reference to the slides or the textbook, or a 1-2 sentence description of the construction to make it completely clear what you are referring to. You do not need to analyze the correctness or runtime of your algorithm.

7. (**Distinguishing sets**) Use distinguishing sets to prove the following statements.

- (a) Every DFA for the language  $L_1 = \{w \mid w \text{ contains the substring } 100\}$  requires at least 4 states.
- (b) Every DFA for the language  $L_3 = \{w \mid \text{every even position of } w \text{ is } 0\}$  requires at least 3 states.

8. (**Optimality of the subset construction**) Let  $\Sigma = \{a, b\}$ . For each  $k \geq 1$ , let  $C_k$  be the language consisting of all strings that contain an  $a$  exactly  $k$  places from the right-hand end. That is,  $C_k = \{wax_1 \dots x_{k-1} \mid w \in \Sigma^*, x_1, \dots, x_{k-1} \in \Sigma\}$ .

- (a) Describe an NFA with  $k + 1$  states that recognizes  $C_k$ . You can either use a state diagram or give a formal description.
- (b) Prove that no DFA with fewer than  $2^k$  states can recognize  $C_k$ .
- (c) Could there be an NFA with fewer than  $k$  states recognizing  $C_k$ ? Explain your answer.

9. (**Bonus problem**) Prove that for every natural number  $n$ , there is a language  $B_n$  such that a)  $B_n$  is recognizable by an NFA with  $n + 1$  states, but b) If  $B_n = A_1 \cup \dots \cup A_k$  for regular languages  $A_1, \dots, A_k$ , then at least one of the languages  $A_i$  requires a DFA with at least  $2^{n/k}$  states.