Please remember to turn in each problem on a separate page.

In all the following problems, let $\Sigma = \{0, 1\}$.

1. Give DFAs accepting the sets of strings defined by the following regular expressions. Try to simplify your machines as much as possible.

   \begin{align*}
   (a) & \quad (0 + 1)^*10(0 + 1) \\
   (b) & \quad (0(0 + 1)^*1(00)^*) + (1(0 + 1)^*1(00)^*0) \\
   (c) & \quad ((00)^*10) + ((000)^*11)
   \end{align*}

   Argue convincingly that your answers are correct, but formal proofs are not necessary.

2. We mentioned in class that the language

   \[ \{ x \in \Sigma^* \mid \exists y \in B \ xy \in A \} \]

   is regular, given that both $A$ and $B$ are regular languages.

   Professor M. Howard claims to have a proof of this result that does not make use of regularity of $B$, so the result must hold even if $B$ is not regular. That is, the professor claims that if $A \subseteq \Sigma^*$ is a regular language and $S \subseteq \Sigma^*$ is an arbitrary (not necessarily regular) language, then the set

   \[ \{ x \in \Sigma^* \mid \exists y \in S \ xy \in A \} \]

   is regular.

   Prove or disprove the professor’s claim.
3. Let \( L_e \) be the set of all strings containing an even number of zeroes and an even number of ones:

\[
L_e = \{ x \in \Sigma^* \mid (\#0(x) \mod 2) = (\#1(x) \mod 2) = 0 \}
\]

(a) Give a deterministic finite automaton that recognizes \( L_e \). Argue convincingly that it does so; you don’t need to give a formal proof.

(b) We define the size of a union-dot-star regular expression by counting atomic patterns and operators. Formally

\[
\begin{align*}
\text{size}(\emptyset) &= \text{size}(\epsilon) = \text{size}(a) = 1 \\
\text{size}(\alpha + \beta) &= \text{size}(\alpha \cdot \beta) = 1 + \text{size}(\alpha) + \text{size}(\beta) \\
\text{size}(\alpha^*) &= 1 + \text{size}(\alpha)
\end{align*}
\]

Suppose you were to construct a regular expression \( \alpha \) for the language recognized by a given DFA, using the standard procedure discussed in lecture and given in Chapter 9 of the text. (Note the procedure works essentially unchanged for either a DFA or an NFA).

Recall the procedure constructs regular expressions

\[
\alpha_{u,v}^A \quad \text{for } A \subseteq Q \text{ and } u, v \in Q
\]

for the sets of strings that can take the machine from state \( u \) to state \( v \) with all intermediate states lying in \( A \).

Derive an (approximate) formula for the size of \( \alpha_{u,v}^A \) as a function of \(|A|\). Explain your answer.

Note an exact formula would depend on the particular FA. For your answer you may assume the machine has no self-loops; i.e.

\[
\delta(q, a) \neq q
\]

and no parallel edges; i.e.

\[
\delta(q, a) = \delta(q, b) \Rightarrow a = b
\]

You may make any other reasonable assumption you find helpful – just be sure to state your assumptions.

How big would \( \alpha \) be for the machine you constructed in part (a)?

(c) Give a more “efficient” (i.e. smaller) regular expression for this language. Correctness is more important than size, but a Gold Star will be awarded for the smallest correct solution.