Exam Time: 1h & 15m. Each problem is worth 10 points. 50 pts total. Choose any 5 problems. If needed, use the back of each page.

1. **Regular Expressions.**

   Using only the definition of regular languages in terms of regular expressions, show that, if $L$ is a regular language, $L'' = \{u \mid \exists v, uv \in L\}$ is regular.

   Hint: think structural induction. Provide the base cases, the statements of all the inductions (how many?), and at least one induction proof.

   **Solution:** The complete proof is almost identical to that of Ex. 1.21 on p. 13. I was asking for the basis steps, the statements of the inductions and the proof of one induction. The inductive proof for union is quite trivial.

2. **Closure of Regular Languages.** Let $A$ be a regular language over some alphabet $\Sigma$. Prove that the language $\{x \mid x^R \in A\}$, is regular.

   **Solution:** Either the last paragraph of Theorem 2.33 on p. 59, or Ex. 2.34 on the same page.
3. **NFA to DFA.** Given the NFA $M = (\{0, 1, 2, 3\}, \{0, 1\}, \delta, 0, \{2\})$, where:

$$
\begin{array}{|c|c|c|c|}
\hline
\delta & 0 & 1 & \epsilon \\
\hline
0 & \{1\} & \{1\} & - \\
1 & \{3\} & - & - \\
2 & - & - & \{0\} \\
3 & \{3\} & \{2, 3\} & - \\
\hline
\end{array}
$$

convert it into an equivalent DFA, showing all intermediate steps.

**Solution:** all the new states include $\epsilon$-closure operations.

$$
\begin{array}{|c|c|c|}
\hline
\delta & 0 & 1 \\
\hline
s_0 = \{0\} & s_1 = \{1\} & s_1 \\
s_1 & s_2 = \{3\} & s_3 = \emptyset \\
s_2 & s_2 & s_4 = \{0, 2, 3\} \\
s_3 & s_3 & s_3 \\
s_4 & s_5 = \{1, 3\} & s_6 = \{0, 1, 2, 3\} \\
s_5 & s_2 & s_4 \\
s_6 & s_5 & s_6 \\
\hline
\end{array}
$$

If we want to construct the graph corresponding to this DFA:

![The DFA as a directed, labeled graph.](image)

4. **Myhill-Nerode Theorem.**

   a) Define the relation $R_L$ used by the theorem.

   b) Find the equivalence classes of $R_L$ for the language \( \{ x \in \{0, 1\}^* \mid \#_0(x) = 2\#_1(x) \} \), where \( \#_a(w) \) is the number of occurrences of the symbol \( a \) in \( w \). Make sure you explain the construction.
c) What does the Myhill-Nerode theorem tell you about this language and why? Start by quoting the Theorem and the appropriate Corollary...

Solution:

a): p. 70

b): we start with: $[\epsilon]_{R_L} = \{ y \in \Sigma^* \mid (\forall w \in \Sigma^*) \, \epsilon w \in L \Rightarrow yw \in L \}$. Since $w$ must be an element of $L$ by $\epsilon w \in L$, this will work for all $y \in L$ and will not work if $y \notin L$, since in that case $\#_0(yw) \neq 2\#_1(yw)$. So $[\epsilon]_{R_L} = L$.

Now consider any two strings $0^m$ and $0^n$ with $m \neq n$. Consider any other string $w \in \Sigma^*$. Then $\#_0(0^m w) \neq \#_0(0^n w)$. Since both strings, $0^m w$ and $0^n w$, have the same number of 1s, only one can satisfy the condition of being in $L$, while the other must be not in $L$.

What we have shown is that $R_L$ has an infinite number of equivalence classes (and we now have the classes $\{[0^n]_{R_L} \mid n \geq 0\}$). It may be instructive to try to identify all the classes (would $\{[1^n]_{R_L} \mid n > 0\}$ do it? Why or why not?), but it is not needed to answer the third part of the question.

c): Thm. 2.48 and Corollary 2.49 (you were supposed to quote them) give that the language is not regular.

5. Pumping Lemma. Show that the language $\{w \in \{0,1,2\}^* \mid \#_0(w) + \#_1(w) = \#_2(w)\}$ is not regular, by applying one of the versions of the Pumping Lemma. You are expected to quote the version you will use, and then to apply it.

Solution: This uses the Strong form of the Pumping Lemma.

a): the full text of the Lemma is on p. 81, Lemma 2.59.

b): how to apply it.

Assume the language, call it $L$, is regular. Let $K$ be the positive integer guaranteed by the Lemma. Choose $\alpha \in L$; in particular, choose $\alpha = 0^K 1^K 2^K$. The Lemma lets us decompose $\alpha = x y z$, where $x = 0^K$, $y = 1^K$, and $z = 2^K$. Since $|y| \geq K$, the Lemma guarantees that $y = 1^K$ can be written as the concatenation of three strings, $y = u w v$, with $|v| \geq 1$ and such that $x u w^0 w z \in L \forall n \geq 0$. Since $v$ contains at least one 1, the number of 1s in $x u w^0 w z = x w w z$ is smaller than in $x w^1 w z = \alpha$, with no change in the number of 0s and 2s. This implies that the defining equation no longer holds for $x u w^0 w z$, and we must conclude that this last string is not in $L$. Contradiction.

6. Minimum DFAs. Construct the minimum DFA for the language $(0 + 1)^*010$. Use any one of the three methods discussed in the textbook, but be explicit and detailed.

Solution: there are several ways to do this. We will look at two.

a): first solution. Equivalence classes. Let $L$ denote the language. start with: $[\epsilon]_{R_L} = \{ y \in \Sigma^* \mid (\forall w \in \Sigma^*) \, \epsilon w \in L \Rightarrow yw \in L \}$. Since $w$ must be an element of $L$ to have $\epsilon w \in L$, this will work for $y = 1$ or $y$ ending in 11, and will not work otherwise (why? look at $y$ ending in 0, 01, 11, 010). So $[\epsilon]_{R_L} = \epsilon + 1 + (0 + 1)^*11 = [1]_{R_L}$. Next, let’s compute $[0]_{R_L}$. What strings belong to this class? Clearly, if we concatenate 0 with any string in $L$, we will obtain a string in $L$. Concatenating 0, or 10, with 10 will also give a string in $L$. Any string ending in 00 or 110 will also have this property. Concatenating with any 3-or-fewer character
strings not in $L$ will not give strings in $L$. So $[0]_{R_L} = 0 + (0 + 1)^*00 + 10 + (0 + 1)^*110$. Next we compute $[01]_{R_L} = (0 + 1)^*011$ and $[10]_{R_L} = [0]_{R_L}$. Note that this completes the 2-character strings. The only one of the 3-character strings not already taken care of is 010: $[010]_{R_L} = (0 + 1)^*010 = L$. Since every string with 4 or more characters must end with one of the eight 3-character patterns, we are done. The classes are $[\epsilon]_{R_L}$, $[0]_{R_L}$, $[01]_{R_L}$ and $[010]_{R_L}$. The minimum DFA has 4 states: $q_0 = [\epsilon]_{R_L}$, $q_1 = [0]_{R_L}$, $q_2 = [01]_{R_L}$ and $q_3 = [010]_{R_L}$. The transition function should be now easy, giving us the graph representation (if you feel the need to give it - the transition table would be enough:

![Graph](image1.png)

The minimal DFA as a directed, labeled graph.

b): second solution. Start with a checker automaton:

![Graph](image2.png)

The initial checker automaton.

which you extend to:

![Graph](image3.png)

A closer approximation.

and you complete, constructing the DFA of the first solution. At this point you do not know whether the automaton you constructed is minimal or not, since you have not computed the equivalence classes. The easiest method to apply is the one presented on p. 77: divide the set of nodes into two subsets: the accepting nodes and the non-accepting ones and then apply the method until you can find no more blocks. You will end up with 4 blocks, giving a 4-state minimal automaton.