CS208 Practice Problem Set 2

1. Let $\Sigma = \{0, 1\}$.

   (a) Construct DFAs for each of the following languages. Try to use as few states as you can in your construction. In the following $n_i(w)$ denotes the count of i's in the string w, for $i \in \Sigma$. Similarly, $|w|$ denotes the length of the string w.

      i. $L_1 = \{0^m 1^n \mid m + n = 0 \mod 2 \text{ and } m = 0 \mod 3\}$. Think of L_1 as the set of all even length strings of 0s followed by 1s, where the count of 0s is a multiple of 3.

      ii. $L_2 = \{0^m 1^n 0^k \mid m, n, k > 0 \text{ and } m + n + k = 0 \mod 2\}$. Think of L_2 as the set of all strings obtained by inserting a block of 1s inside a block of 0s such that the length of the overall string becomes even.

      iii. $L_3 = \{w \in \Sigma^* \mid w = u \cdot v, \text{ where } u, v \in \Sigma^* \text{ and } n_0(u) = 0 \mod 2, n_1(v) = 0 \mod 2\}$. Think of L_3 as the set of all strings that can be cut into two parts and given to two applications, one of which expects an even count of 0s while the other expects an even count of 1s.

      iv. $L_4 = \{w \in \Sigma^* \mid |w| + 2.n_1(w) = 0 \mod 3\}$. Think of L_4 as the set of all strings whose length would be a multiple of 3 if we simply repeated each 1 in the string thrice (without caring about the 0s).

   (b) Construct NFAs for each of the following languages. Feel free to use ϵ-transitions. The purpose of constructing NFAs is really to capture the intuitive structure of strings in the language in as direct a way as possible. Feel free to re-use DFAs constructed in the previous part of this question as building blocks of your NFAs.

      i. $L_5 = \{w \in \Sigma^* \mid w = u \cdot v \cdot x, \text{ where } u, v, w \in \Sigma^* \text{ and } |u| + 2.|v| + 3.|x| = 0 \mod 4\}$. You can think of a string being

broken into three parts and fed to three applications, such that the first (resep. second and third) application charges Re. 1 (resp. Rs. 2 and Rs. 3) to process each letter in the string. Then L_5 is the set of strings that can be processed by spending a muliple of 4 Rupees.

ii. $L_6 = \{w \in \Sigma^* \mid w = u \cdot v, \text{ and either } u \in L_1, v \in L_2 \text{ or } u \in L_2, v \in L_1\}$. Thus, L_6 is the set of all strings that can be broken into two parts, such that the first part belongs to L_1 and the second part belongs to L_2, or vice versa.

iii. $L_7 = \{w \in \Sigma^* \mid u_1 \cdot u_2 \cdot \ldots \cdot u_k, \text{ where } k > 0, k = 0 \mod 2, \text{ and } u_i \in L_2 \text{ for all } i\}$. Thus, L_7 is the language of all strings that can be broken up into an even number of strings from L_2.

iv. $L_8 = \{w \in \Sigma^* \mid n_1(w') = 0 \mod 2, \text{ where } w' \text{ is obtained from } w$ by replacing every alternate letter starting from the second letter by $\varepsilon\}$. Thus if $w = 0101001$, then $w' = 0001$. You can think of the string w as a sequence of bits coming in so fast that a machine can only read every alternate bit. L_8 is then the sequence of strings that can be accepted by such a lossy automaton for L_2.

2. Suppose r_1 and r_2 are regular expressions over the same alphabet Σ. We say $r_1 = r_2$ to denote equality of the languages represented by r_1 and r_2. In other words, every string in the language represented by r_1 is also included in the language represented by r_2 and vice versa. For each of the following pairs of regular expressions over $\Sigma = \{0, 1\}$, either prove that they represent the same language, or give a string that is present in the language of one but not in the language of the other. In the latter case, you must also describe why your solution string is in the language of one regular expression, but not in that of the other.

   (a) $r_1 = \mathbf{1}^*(\mathbf{1} + \mathbf{0})^*\mathbf{0}^*$ and $r_2 = (\mathbf{0}^*\mathbf{1}^*)^*$

   (b) $r_1 = ((\mathbf{0} + \mathbf{1})^*\mathbf{0})^*\mathbf{0}$ and $r_2 = (\mathbf{0} + \mathbf{1})^*\mathbf{0}^*\mathbf{0}$

   (c) $r_1 = (\mathbf{0} + \mathbf{1})^*\mathbf{01}(\mathbf{0} + \mathbf{1})^*$ and $r_2 = \mathbf{1}^*(\mathbf{0} + \mathbf{1})^*\mathbf{0}(\mathbf{0} + \mathbf{1})^*\mathbf{1}$

3. Give as small an upper bound as you can of the number of *distinct* languages over $(\mathbf{0} + \mathbf{1})^*$ that can be recognized by DFAs using at most n states. Your answer must be a function of n. You must also clearly explain your reasoning.

4. The solution to this problem is specific to your roll number. Let $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, B\}$. We will say a string $w \in \Sigma^*$ is *embedded* in a string $u \in \Sigma^*$ iff w can be obtained from u by replacing some letters in u with ε. Thus, 014 is embedded in 203421049. To see why this is so, notice that 014 can be obtained as $\varepsilon 0 \varepsilon\varepsilon\varepsilon 1 \varepsilon 4 \varepsilon$. However, 014 is not embedded in 23421049.

   (a) State your roll no. as a string in Σ^*. Let us call this string ρ.

   (b) Give a DFA with no more than $|\rho| + 1$ states that accepts the language $L_\rho = \{w \mid w \in \Sigma^*, \rho \text{ is embedded in } w\}$. You must explain what each state in your DFA represents (use a couple of lines of explanation per state), as evidence that you understood the construction of the DFA. Your answer will fetch 0 marks without proper explanation per state.

   Note: You can of course start with a NFA, and then use the subset construction to convert it to a DFA. But this is a long and tedious route, and will not give you much intuition to understand what each state in the resulting DFA represents. So you are strongly advised not to follow this route. There is enough structure in the problem to permit a much simpler construction of a DFA, and you are encouraged to think about it.