Solutions to Problem Set 2

1. Let k be a positive integer. Let $\Sigma = \{0, 1\}$, and L be the language consisting of all strings over $\{0, 1\}$ containing a 1 in the kth position from the end (in particular, all strings of length less than k are not in L). **[8 + 8 + 14 = 30 points]**

   (a) Construct a DFA with exactly 2^k states that recognizes L.

   SOLUTION: Construct a DFA with one state corresponding to every k-bit string. Formally, let $Q = \{0, 1\}^k$. We keep track of the last k bits read by the machine. Thus, for a state $x_1 \ldots x_k$, we define the transition on reading the bit b as $\delta(x_1 \ldots x_k, b) = x_2 \ldots x_k b$. Take $q_0 = 0^k$ and $F = \{x_1 \ldots x_k \in Q \mid x_1 = 1\}$. Note that this does not accept any strings of length less than k (as we start with the all zero state) since the kth position from the end does not exist, and is hence not 1.

   (b) Construct a NFA with exactly $k + 1$ states that recognizes L.

   SOLUTION: We construct an NFA with $Q = \{0, 1, \ldots, k\}$, with the names of the states corresponding to how many of the last k bits the NFA has seen. Define $\delta(0, 0) = 0$, $\delta(0, 1) = \{0, 1\}$ and $\delta(i - 1, 0/1) = i$ for $2 \le i \le k$. We set $q_0 = 0$ and $F = \{k\}$. The machine starts in state 0, on seeing a 1 it may guess that it is the kth bit from the end and proceed to state 1. It then reaches state k and accepts if and only if there are exactly $k - 1$ bits following the one on which it moved from 0 to 1.

   (c) Prove that any DFA that recognizes L has at least 2^k states.

   SOLUTION: Consider any two different k-bit strings $x = x_1 \ldots x_k$ and $y = y_1 \ldots y_k$ and let i be some position such that $x_i \ne y_i$ (there must be at least one). Hence, one of the strings contains a 1 in the ith position, while the other contains a 0. Let $z = 0^{i-1}$. Then z distinguishes x and y as exactly one of xz and yz has the kth bit from the end as 1. Since there are 2^k binary strings of length k, which are all mutually distinguishable by the above argument, any DFA for the language must have at least 2^k states.

2. **[10 + 10 + 10 = 30 points]**

   (a) Let A be the set of strings over $\{0, 1\}$ that can be written in the form $1^k y$ where y contains at least k 1s, for some $k \ge 1$. Show that A is a regular language.

   SOLUTION: It is easy to see that any string in A must start with a 1, and contain at least one other 1 (in the matching y segment). Conversely, any string that starts with a 1 and contains at least one other 1 matches the description for $k = 1$. Hence, A is described by the regular expression $1 \circ 0^* \circ 1 \circ (0 \cup 1)^*$, and is therefore regular.

   (b) Let B be the set of strings over $\{0, 1\}$ that can be written in the form $1^k 0 y$ where y contains at least k 1s, for some $k \ge 1$. Show that B is not a regular language.

   SOLUTION: Assume to the contrary that B is regular. Let p be the pumping length given by the pumping lemma. Consider the string $s = 1^p 0^p 1^p \in B$. The pumping lemma guarantees that s can be split into 3 pieces $s = abc$, where $|ab| \le p$. Hence, $y = 1^i$ for some $i \ge 1$. Then, by the pumping lemma, $ab^2c = 1^{p+i} 0^p 1^p \in B$, but cannot be written in the form specified, a contradiction.

(c) Let C be the set of strings over $\{0, 1\}$ that can be written in the form $1^k z$ where z contains at most k 1s, for some $k \geq 1$. Show that C is not a regular language.

SOLUTION: Assume to the contrary that C is regular. Let p be the pumping length given by the pumping lemma. Consider the string $s = 1^p 0^p 1^p \in B$. The pumping lemma guarantees that s can be split into 3 pieces $s = abz$, where $|ab| \leq p$. Hence, $b = 1^i$ for some $i \geq 1$. Then, by the pumping lemma, $ac = 1^{p-i} 0^p 1^p \in C$, but cannot be written in the form specified, a contradiction.

3. Write regular expressions for the following languages: **[12 + 8 = 20 points]**

(a) The set of all binary strings such that every pair of adjacent 0's appears before any pair of adjacent 1's.

SOLUTION: Using $R(L)$, to denote the regular expression for the given language L, we must have $R(L) = R(L_1)R(L_2)$, where L_1 is the language of all strings that do not contain any pair of 1's and L_2 is the language of all strings that do not contain any pair of 0's. For a string in L_1, every occurrence of a 1, except possibly the last one, must be followed by a 0. Hence, $R(L_1) = (0 + 10)^*(1 + \epsilon)$. Similarly, $R(L_2) = (1 + 01)^*(0 + \epsilon)$. Thus, $R(L) = (0+10)^*(1+\epsilon)(1+01)^*(0+\epsilon)$, which simplifies to $(0+10)^*(1+01)^*(0+\epsilon)$.

(b) The set of all binary strings such that the number of 0's in the string is divisible by 5.

SOLUTION: Any string in the language must be composed of 0 or more blocks, each having exactly five 0's and an arbitrary number of 1's between them. This is given by the regular expression (1*01*01*01*01*1*01*). However, this does not capture the strings containing all 1's, which can be included separately, giving the expression (1*01*01*01*01*01*)+ 1^* for the language.

4. We say a string x is a *proper prefix* of a string y, if there exists a non-empty string z such that $xz = y$. For a language A, we define the following operation

$$NOEXTEND(A) = \{w \in A \mid w \text{ is not a proper prefix of any string in } A\}$$

Show that if A is regular, then so is $NOEXTEND(A)$.**[20 points]**

SOLUTION: Given a DFA for the language A, we want to accept only those strings which reach a final state, but to which no string can be added to reach a final state again. Hence, we want to accept strings ending in exactly those final states, from which there is no (directed) path to any final state (not even itself).

For a given state $q \in F$, we can check if there is a path from q to any state in F (or a cycle involving q) by a DFS. Let $F' \subseteq F$ be the set of all the states from which there is no such path. Then changing the set of final states of the DFA to F' gives a DFA for $NOEXTEND(A)$.