DUE: Thursday, September 19

1. (8 points) Prove that for every \( n \geq 2 \) there exists a language \( L_n \) for which there exists a DFA with \( n \) states to accept \( L_n \) and there does not exist a DFA with \( n-1 \) states to accept \( L_n \).

2. (5 points) For NFA \( N = (Q, \Sigma, \Delta, S, F) \), state \( q \in Q \) is useful if there exist \( s \in S, z \in \Sigma^* \) such that \( q \in \Delta(s, z) \) and there exist \( f \in F, x \in \Sigma^* \) such that \( f \in \Delta(q, x) \), and a state is useless if it is not useful. Remove useless states from the NFA with \( \varepsilon \)-transitions

\[
N = \left( \left\{ q_0, q_1, q_2, q_3, q_4, q_5 \right\}, \{0,1\}, \Delta, \left\{ q_0, q_2 \right\}, \left\{ q_3 \right\} \right)
\]

to obtain an equivalent NFA with no useless states, where

\[
\begin{array}{c|ccc}
\Delta & 0 & 1 & \varepsilon \\
\hline
q_0 & \left\{ q_1, q_3 \right\} & \left\{ q_2 \right\} & \emptyset \\
q_1 & \emptyset & \emptyset & \emptyset \\
q_2 & \emptyset & \emptyset & \left\{ q_5 \right\} \\
q_3 & \emptyset & \emptyset & \emptyset \\
q_4 & \emptyset & \left\{ q_2, q_5 \right\} & \left\{ q_3 \right\} \\
q_5 & \emptyset & \emptyset & \left\{ q_1 \right\} \\
\end{array}
\]

3. (15 points) a Describe a decision procedure which accepts as input an NFA \( N \) and decides whether or not \( L(N) \) is infinite.

b Describe a decision procedure which accepts as input an NFA \( N \) and \( k \in \mathbb{N} \) and decides whether or not \( |L(N)|=k \).

c Describe a decision procedure which accepts as input two NFAs \( N_0 \) and \( N_1 \) and decides whether \( |L(N_0)|<|L(N_1)| \). Note that if regular languages \( L(N_0) \) and \( L(N_1) \) are each infinite, then they have the same cardinality.

4. (6 points) For the NFA \( \left( \left\{ q_0, q_1, q_2 \right\}, \{0,1\}, \Delta, \left\{ q_0, q_1 \right\}, \left\{ q_0, q_2 \right\} \right) \), where

\[
\begin{array}{c|cc}
\Delta & 0 & 1 \\
\hline
q_0 & \left\{ q_1, q_2 \right\} & \emptyset \\
q_1 & \left\{ q_1 \right\} & \left\{ q_1 \right\} \\
q_2 & \emptyset & \left\{ q_0 \right\} \\
\end{array}
\]

use the algorithm from class and our text to construct the regular expression

\[
\alpha_{\left\{ q_0, q_0 \right\}} + \alpha_{\left\{ q_0, q_1 \right\}}
\]

describing the language defined by the set of paths from \( q_0 \) to \( q_1 \) or \( q_2 \).
only possibly passing through $q_0$ or $q_1$ as intermediate states. You don't need to simplify the regular expression produced by the algorithm.
1. For \( n \geq 2 \), let \( L_n = \{0^{n-2}\} \) be defined over \{0\}. The language is accepted by 
\[ \{q_0, \ldots, q_{n-1}\}, \{0\}, \delta, \{q_0, \{q_{n-2}\}\} \], where \( \delta(q_i, 0) = q_{i+1} \) for \( 0 \leq i \leq n-2 \), and 
\( \delta(q_{n-1}, 0) = q_{n-1} \) (state \( q_{n-1} \) is a black hole).

Since \( |\Sigma| = 1 \), each state of a DFA has one edge leaving it. So beginning at the start state we encounter a (possibly trivial) path, leading to a nontrivial cycle (a path or cycle is trivial if it does not have any edges). Since \( L_n \) is finite, no states on the cycle can be final states. To accept \( L_n \), at \( n-2 \) edges from the start state is a final state (there are \( n-1 \) states on this path), and its outgoing edge is to the cycle. So any acceptor for to accept \( L_n \) must have at least \( n \) states.

2. State \( q_4 \) is useless because it can’t be reached from any start state. States \( q_1, q_2 \) and \( q_5 \) are useless since there does not exist a path from any of them to a final state. So removing them yields equivalent \( N = \{\{q_0, q_3\}, \{0, 1\}, \Delta^+, \{q_0\}, \{q_3\}\} \), where

\[
\begin{array}{ccc}
\Delta^+ & 0 & 1 \\
q_0 & \{q_3\} & \emptyset & \emptyset \\
q_3 & \emptyset & \emptyset & \emptyset \\
\end{array}
\]

3. For all three parts of this problem, we assume the NFAs \( N, N_0 \) and \( N_1 \) have been converted to equivalent DFAs \( M, M_0 \) and \( M_1 \) which have, respectively, sets of states \( Q, Q_0 \) and \( Q_1 \).

\( a \) \( L(N) \) is infinite if and only if there is a path from the start state of \( M \) to a final state of \( M \), and a vertex on the path lies on a nontrivial cycle. The Floyd-Warshall Algorithm tests for each the existence of a path from any vertex to any other vertex, and it tests whether there is a path from any vertex to itself (the equivalent of lying on a cycle).

\( b \) We use the previous algorithm to test if \( L(M) \) is finite. If it is finite, then all strings in \( L(M) \) are of length at most \( |Q|-1 \). We test all strings in \( \Sigma^* \) of length at most \( |Q|-1 \) to decide if they belong to \( L(N) \), and we test if the number of strings which do belong is equal to \( k \).

\( c \) We use the procedure from part \( a \) to decide if either of \( L(N_0) \) or \( L(N_1) \) is infinite. If either is infinite, the answer is simple. If both \( L(N_0) \) and \( L(N_1) \) are finite, then we invoke the following finite, albeit inefficient, procedure:

\[
k \leftarrow 0
\]

\[
\text{repeat}
\]

\[
\text{if } |L(N_0)| = k \text{ or } |L(N_1)| = k \text{ then}
\]

\[
\text{if not } |L(N_i)| = k \text{ then return true else return false}
\]

\[
k \leftarrow k + 1
\]

\[
\text{forever}
\]
4. \( \alpha_{q_0q_0} \cap \alpha_{q_0q_2} = \varepsilon \)
\( \alpha_{q_0q_0} = \varepsilon + 0 + 1 \)
\( \alpha_{q_0q_1} = \alpha_{q_0q_2} = 0 \)
\( \alpha_{q_0q_0} = \alpha_{q_0q_2} = \emptyset \)
\( \alpha_{q_0q_0} = 1 \)
\( \alpha_{q_0q_1} = \emptyset \)
\( \alpha_{q_0q_0} = \varepsilon \)
\( \alpha_{q_0q_1} = \alpha_{q_0q_2} = 0 \)
\( \alpha_{q_0q_1} = \varepsilon + 0 + 1 \)
\( \alpha_{q_0q_1} = \alpha_{q_0q_2} = \emptyset \)
\( \alpha_{q_0q_0} = \varepsilon \)
\( \alpha_{q_0q_1} = 1 \)
\( \alpha_{q_0q_0} = 10 \)
\( \alpha_{q_0q_1} = 0 + 0(\varepsilon + 0 + 1)^*(\varepsilon + 0 + 1) \)
\( \alpha_{q_0q_2} = 0 + 0(\varepsilon + 0 + 1)^*\emptyset \)
\( \alpha_{q_0q_0} + \alpha_{q_0q_1} = 0 + 0(\varepsilon + 0 + 1)^*(\varepsilon + 0 + 1) + 0 + 0(\varepsilon + 0 + 1)^*\emptyset \)

which simplifies to \( 0(0 + 1)^* \).