Quiz 2

Your name:

Think. Be careful, clear, and precise.

1. Complete the following narrative, following the conventions of lecture and your text.
   
   A **DFA** was defined as a five-tuple \( M = (Q, \Sigma, \delta, q_0, F) \) where \( Q \) is a finite set, \( \Sigma \) is an alphabet, \( q_0 \in Q \), \( F \subseteq Q \), and \( \delta : Q \times \Sigma \to Q \).

   To define an **NFA** \( M' \) we modified the conventions above to say that an NFA is a 5-tuple \( M = (Q, \Sigma, \delta, q_0, F) \) where \( Q, \Sigma, q_0, F \) were as before, but now \( \delta \) has a domain of \( Q \) and range \( Q \).

   We showed that DFAs and NFAs accept the same class of languages. For the “easy” direction of this, we said that, informally, every DFA \( M = (Q, \Sigma, \delta, q_0, F) \) is an NFA. But that’s not formally true, because the transition functions have different signatures. So, formally, given a DFA \( M = (Q, \Sigma, \delta, q_0, F) \) you need to construct an NFA \( M' = (Q, \Sigma, \delta', q_0, F) \), where \( L(M') = L(M) \), by saying that \( \delta'(q, a) = \) when \( a \in \Sigma \), and \( \delta'(q, \varepsilon) = \) .

   For the nontrivial direction, we are given an NFA \( M = (Q, \Sigma, \delta, q_0, F) \). We saw how to eliminate the \( \varepsilon \)-arrows, so we can assume, without loss of generality, that \( \delta(q, \varepsilon) = \emptyset \) for all \( q \in Q \). Construct from \( M \) a DFA \( M' = (Q', \Sigma, \delta', q'_0, F') \) where \( Q' = \) and, additionally, \( \delta'(S, a) = \) (for \( S \in Q' \), \( a \in \Sigma \), \( q'_0 = \{q_0\} \), and, \( F' = \) )

2. You are given a first DFA \( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \) with \( |Q_1| = 10 \) states, \( |F_1| = 5 \) of them final. You are given a second DFA \( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \) with \( |Q_2| = 10 \) states, \( |F_2| = 5 \) of them final. Suppose you use the product construction to make a DFA \( M = (Q, \Sigma, \delta, s, F) \) for \( L(M_1) \cup L(M_2) \). It will have \( |Q| = \) states and \( |F| = \) of them will be final.

3. Similarly, suppose you mindlessly convert \( 0 \cup 10^* \) into an NFA \( M \) using the procedures shown in class and in the book. Then \( M \) will have states.

4. Suppose \( L \subseteq \Sigma^* \) is accepted by an \( n \)-state DFA. For any pair of strings \( x, y \in \Sigma^* \), say \( x \sim y \) if for every \( z \in \Sigma^* \), \( xz \in L \iff yz \in L \). Say something interesting about the number of equivalence classes, \( m \), of this relation.

Please turn the page over!
5. Circle the correct answer. Missing answers will be treated as wrong, so if you don’t know an answer, please guess.

(a) **True** or **False**: There exists a function \( f : \mathbb{N} \rightarrow \mathbb{N} \) such that no function \( F : \mathbb{N} \rightarrow \mathbb{N} \) that upperbounds it\(^1\) can be computed.

(b) **True** or **False**: If \( M = (Q, \Sigma, \delta, q_0, F) \) is a DFA and \( F = Q \) then \( L(M) = \Sigma^* \).

(c) **True** or **False**: If \( M = (Q, \Sigma, \delta, q_0, F) \) is an NFA and \( F = Q \) then \( L(M) = \Sigma^* \).

(d) **True** or **False**: If \( A \) and \( B \) are regular then so is \( A \cap B \).

(e) **True** or **False**: If \( L^* \) is regular then \( L \) is regular.

(f) **True** or **False**: If \( L \) is finite then \( L \) is regular.

(g) **True** or **False**: Every subset of a regular language is regular.

(h) **True** or **False**: A regular expression is a string.

(i) **True** or **False**: We have seen that the pumping lemma is a useful tool for proving languages regular.

(j) **True** or **False**: An efficient procedure\(^2\) is known that takes a regular expression \( \alpha \) and a word \( w \) and decides if \( w \in L(\alpha) \).

---

\(^1\) \( F \) upperbounds \( f \) if \( F(x) \geq f(x) \) for all \( x \).

\(^2\) Eg, linear, quadratic, or cubic time in \(|\alpha| + |w|\).