Problem Set 3 Solutions

Problem 1. Using the procedure shown in class, convert the following NFA into a DFA for the same language.

The problem is pretty mechanical—I’m not going to draw out the solution—hopefully you didn’t have trouble doing so.

Problem 2. Using the procedure shown in class, eliminate all ε-arrows from the following NFA.

The problem too is mechanical. States 1, 2, and 3 all become final (so all states are now final), since they can reach final states along ε-paths. Now we add in “bypass arcs.” The approach I explained in class for doing this: for each state of the NFA, find all states reachable from the state along ε-paths; find each transition to a state labeled by a character \(a \in \Sigma\); add in a direct connection, if needed, from the state to the labeled state. After all this is done, eliminate all ε-transitions.

Problem 3. Let \(L_1, L_2, L_3 \subseteq \Sigma^*\) be languages and let \(\text{Most}(L_1, L_2, L_3)\) be the set of all \(x \in \Sigma^*\) that are in at least two of \(L_1, L_2, L_3\). Prove: if \(L_1, L_2, L_3\) are DFA-acceptable then so is \(\text{Most}(L_1, L_2, L_3)\).

Solution 1: Extend the product construction. Let \(M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)\), \(M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)\), and \(M_3 = (Q_3, \Sigma, \delta_3, q_3, F_3)\) be DFAs for \(L_1, L_2, L_3\), respectively. Form a new DFA \(M = (Q, \Sigma, \delta, s, F)\) for \(\text{Most}(L_1, L_2, L_3)\) by defining \(Q = Q_1 \times Q_2 \times Q_3\), \(s = (q_1, q_2, q_3)\), \(\delta((p, q, r), a) = (\delta_1(p, a), \delta_2(q, a), \delta_3(r, a))\), and \(F = \{(p, q, r) \in Q_1 \times Q_2 \times Q_3 : \text{at least two of the following three things are true: } p \in F_1, q \in F_2, r \in F_3\}\). It is easy to see that \(L(M) = \text{Most}(L_1, L_2, L_3)\).

Solution 2: Use closure properties. Note that \(\text{Most}(L_1, L_2, L_3) = (L_1 \cap L_2) \cup (L_2 \cap L_3) \cup (L_1 \cap L_3)\). The regular languages are closed under \(\cap\) and \(\cup\) and so they are closed under \(\text{Most}\).

Problem 4. Let \(\text{Stutter}(L) = \{a_1a_1a_2a_2 \cdots a_na_n : a_1a_2 \cdots a_n \in L\}\). (A) Prove that the DFA-acceptable languages are closed under \(\text{Stutter}\). (B) Then, having proved it once, give another, entirely different proof.
Here three different proofs:

(1) Consider the map \( h : \Sigma \to \Sigma^* \) defined by \( h(a) = aa \) for all \( a \in \Sigma \). Then \( \text{Stutter}(L) = h(L) \). We know that the DFA/NFA-acceptable languages are closed under homomorphism (from a previous problem set), so we are done.

(2) Let \( M = (Q, \Sigma, \delta, q_0, F) \) be a DFA accepting \( L \). To make an NFA accepting \( \text{Stutter}(L) \), add a state “in the middle of each arrow” to ensure that a symbol \( a \in \Sigma \) is always followed by a symbol \( a \), and the same destination is then reached. This would give an NFA for \( \text{Stutter}(L) \). You could, if desired, make it into a DFA by the addition of a dead state that was connected up to the rest of the machine in the natural way.

(3) Use the regular-expression characterization of the DFA-acceptable languages. Let \( \alpha \) be a regular expression over \( \Sigma \). Construct from \( \alpha \) a new regular expression \( \beta \) by replacing each character \( a \in \Sigma \) that occurs in \( \alpha \) by \( (a \circ a) \). What results is a new regular expression \( \beta \) where \( L(\beta) = \text{Stutter}(L(\alpha)) \).

**Problem 5.** How many states are in the smallest possible DFA for \( \{0,1\}^* \{1^{10}\} \)? Prove your result.

First, 11 states are sufficient: there is a DFA \( M_{11} \) that accepts \( L = \{0,1\}^* \{1^{10}\} \) and has 11 states. The machine has states \( Q = \{q_0, q_1, \ldots, q_{10}\} \) with \( q_0 \) the start state, \( F = \{q_{10}\} \) the final states, \( \delta(q_0, 0) = q_0 \) for all states \( q \in Q \), while \( \delta(q_i, 1) = q_{i+1} \) for \( i < 10 \) and \( \delta(q_{10}, 1) = q_{10} \).

Second, 11 states are necessary. Suppose for contradiction that there exists a 10-state DFA \( M = (Q, \Sigma, \delta, q_0, F) \) that accepts \( L \). Consider the 11 strings \( 1^i \) for \( 0 \leq i \leq 10 \). By the pigeonhole principle we know that \( \delta^*(q_0, 1^i) = \delta^*(q_0, 1^I) \) for \( 0 \leq i < I \leq 10 \). But then \( \delta^*(q_0, 1^{10-I}) = \delta^*(q_0, 1^I) \), so \( \delta^*(q_0, 1^{10-j}) = \delta^*(q_0, 1^{10}) \) for some \( j \geq 1 \). But the lefthand state must be outside \( F \) and the righthand states must be in \( F \), a contradiction.

One could use the DFA minimization procedure to prove this, establishing that \( M_{11} \) is already a minimal-size DFA. Here one shows that no two states are equivalent, which follows, we have claimed, by showing that the algorithm of class discovers no inequivalence when looking at 0-and 1-character extensions.

**Problem 6** Let \( L_n \) (for \( n \geq 1 \)) be \( \{0,1\}^* \{1\} \{0,1\}^n \). Prove that there is an NFA for \( L_n \) having \( n + 2 \) states, but that there is no DFA for \( L_n \) having \( 2^n - 1 \) or fewer states. In a well written English sentence or two, give a high-level interpretation of your result.

As with the last problem, the first part is constructive; just draw the needed machine. For the second part, assume for contradiction that there is a \((2^n - 1)\)-state DFA \( M = (Q, \Sigma, \delta, q_0, F) \). By the pigeonhole principle, we know that some two distinct strings \( x, x' \in \{0,1\}^n \) satisfy \( \delta(q_0, x) = \delta(q_0, x') \). Since \( x \) and \( x' \) differ, they do so at some particular bit position \( \ell \in [1..n] \) (numbering from 1, starting on the left). Let \( x_0 \) be the one of \( x, x' \) with \( x_0[\ell] = 0 \) and let \( x_1 \) be the one of \( x, x' \) with \( x_1[\ell] = 1 \). Now consider the strings \( y_0 = x_0 0^\ell \) and \( y_1 = x_1 0^\ell \). The second is in \( L_n \); the first is not. But we know that \( \delta^*(q_0, y_0) = \delta^*(q_0, y_1) \), getting us our contradiction: this state cannot be both final and nonfinal.

Interpretation of the result: There can be an exponential gap between the size of the smallest NFA for a language and the size of the smallest DFA for it. Or, said differently, Some languages can be represented much more efficiently with an NFA than a DFA.