Automata Theory - Homework I (Solutions)

K. Subramani
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

1 Problems

1. A tree is defined as an undirected connected graph without any cycles. Argue that if a tree has n nodes, it must have precisely $(n-1)$ edges. *Hint: Use structural induction.*

   Solution: The hypothesis is clearly correct in the base case, since if a tree has one node, it must have zero edges. Assume that the hypothesis is true whenever a tree has at most k nodes, i.e., the inductive hypothesis is that if a tree has k nodes, then it has precisely $(k-1)$ edges. Now consider a tree having $(k+1)$ nodes. As discussed in class, every tree *must* have a pendant node, i.e., a node with degree 1. Observe that this node, say v_a, connects to the rest of the tree through an edge, say e_a. Remove v_a (and hence v_a) from the tree to get a tree having k nodes. As per the inductive hypothesis, this tree has precisely $(k-1)$ edges. Accordingly, the original tree with v_a in it, must have had precisely $(k-1)+1 = k$ edges. Thus, when the hypothesis is true for structures of size k, it must be true for structures of size $(k+1)$. Applying the principle of mathematical induction, we can conclude that a tree with n nodes has precisely $(n-1)$ edges. □

2. Let $\Sigma = \{0,1\}$ denote an alphabet. Enumerate five elements of the following languages:

   (a) Even binary numbers,

   (b) The number of zeros is not equal to the number of ones in a binary string.

   (c) The number of zeros is exactly one greater than the number of ones.

   Solution:

   (a) Even binary numbers: $\{0,\ 10,\ 100,\ 110,\ 1000\}$.

   (b) The number of zeros is not equal to the number of ones in a binary string: $\{0,\ 1,\ 100,\ 110,\ 001\}$.

   (c) The number of zeros is exactly one greater than the number of ones: $\{0,\ 100,\ 001,\ 010,\ 00011\}$.

   □

3. Let $\Sigma = \{0,1\}$. The language L_3 is defined as follows:
   $L_3 = \{x \mid x \in \Sigma^*,\ x \ mod\ 3 \equiv 0,\ \text{when interpreted as a number in binary}\}$.
   Is L regular? Justify your answer with a proof or a counterexample.

   Solution: The language L_3 accepts precisely those binary strings which when interpreted as numbers are exactly divisible by 3. Figure (1) presents a DFA for this language; the existence of a DFA for the language establishes its regularity. A formal inductive proof establishing that the DFA accepts L_3 is beyond the scope of this question.

   The first step in the design is to identify that the DFA must have 3 states, viz., one state to denote strings that are exactly divisible by 3, one state to denote strings that result in a remainder of 1, when divided by 3 and another state to denote strings that result in a remainder of 2, when divisible by 3.

   The second observation is that appending a 0 to the right of a binary number causes its *value* as a number to double, whereas adding a 1 results in a number that is the sum of 1 and twice the original value.

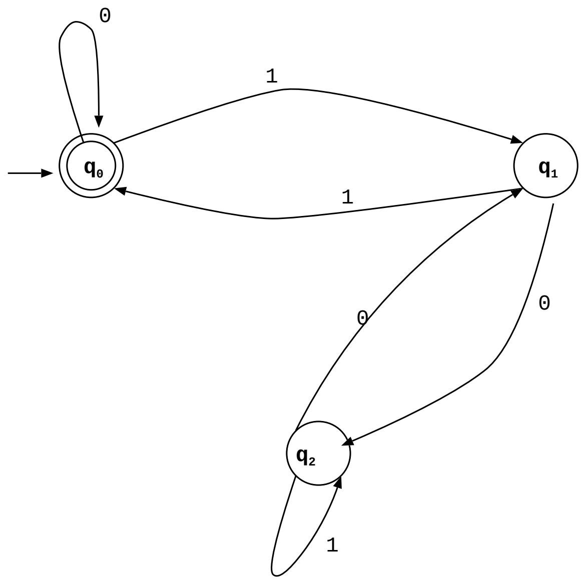

Figure 1: A DFA for divisibility by 3.

The third set of observations are as follows:

(a) If $p \equiv 0\ mod\ 3$, then $2 \cdot p \equiv 0\ mod\ 3$ and $(2 \cdot p + 1) \equiv 1\ mod\ 3$.

(b) If $p \equiv 1\ mod\ 3$, then $2 \cdot p \equiv 2\ mod\ 3$ and $(2 \cdot p + 1) \equiv 0\ mod\ 3$.

(c) If $p \equiv 2\ mod\ 3$, then $2 \cdot p \equiv 1\ mod\ 3$ and $(2 \cdot p + 1) \equiv 2\ mod\ 3$.

□

4. Let L_1 and L_2 denote two languages over an alphabet Σ. For any language $L \subseteq \Sigma^*$, the language L^R consists of those strings in Σ^*, whose reverses are in L. Prove or disprove the following claim: $(L_1 \cup L_2)^R = L_1^R \cup L_2^R$.

   Solution: Rather surprisingly, the claim is correct. Let us use x^R to denote the reverse of string x. As per the definition of L^R, $x \in L$ if and only if $x^R \in L^R$.

   Let $y \in \Sigma^*$ denote an arbitrary string in the set $(L_1 \cup L_2)^R$. Assume that $y \notin (L_1^R \cup L_2^R)$. It follows that $y \notin L_1^R$ **and** $y \notin L_2^R$. Since $y \notin L_1^R$, it must be the case that $y^R \notin L_1$. Arguing similarly, $y^R \notin L_2$. Therefore, $y^R \notin (L_1 \cup L_2)$. But this immediately implies that $y \notin (L_1 \cup L_2)^R$, contradicting the hypothesis.
   We thus have, $(L_1 \cup L_2)^R \subseteq (L_1^R \cup L_2^R)$.

   Let $y \in \Sigma^*$ denote an arbitrary string in the set $(L_1^R \cup L_2^R)$. As per the definition of set union, **either** $y \in L_1^R$ **or** $y \in L_2^R$. Observe that if $y \in L_1^R$, then $y^R \in L_1$. Hence, $y^R \in (L_1 \cup L_2)$ and therefore, $y \in (L_1 \cup L_2)^R$. In similar fashion, we can deduce that if $y \in L_2^R$, then $y \in (L_1 \cup L_2)^R$. It therefore follows that $(L_1^R \cup L_2^R) \subseteq (L_1 \cup L_2)^R$.

   From the above discussion, we can conclude that $(L_1 \cup L_2)^R = L_1^R \cup L_2^R$. □

5. Convert the λ-NFA in Figure (2) into a DFA. Note that the L in the figure represents λ and that $\Sigma = \{0, 1\}$.

   Solution:

   Figure (3) represents the direct application of the conversion algorithm discussed in class.

   I did get rid of the unreachable states and the dead state. □

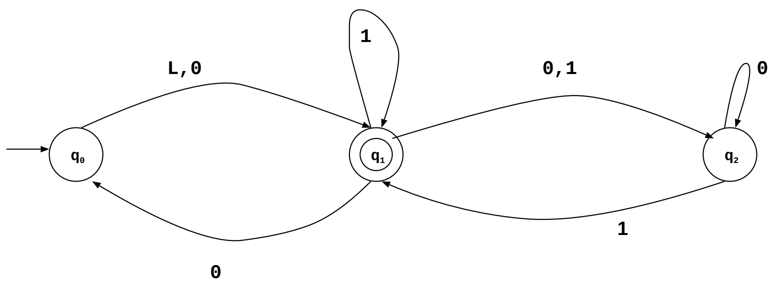

Figure 2: λ-NFA

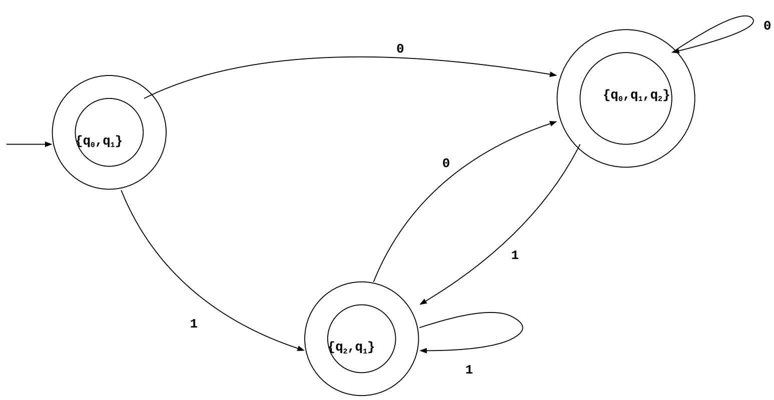

Figure 3: Conversion of λ-NFA to DFA