Automata Theory - Homework II (Solutions)

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

1 Problems

1. Suppose that you are given the DFA D_L of a regular language L. Design an algorithm to check that L contains at least 50 strings.

   Solution: Since we are given the DFA D_L, corresponding to the regular language L, we know the integer n of the Pumping Lemma. We first check whether L is infinite, using the algorithm discussed in class. Observe that if L is determined to be infinite, then clearly it contains more than 50 strings. Now consider the case, in which L has been determined to be finite. From the Pumping Lemma, we know that all strings in L have length at most $(n-1)$. Accordingly, we generate all strings of length at most $(n-1)$ from the alphabet of L and check for membership in L. A counter which keeps track of the number of accepted strings tells us whether or not L contains 50 or more strings. $\Box$

2. A palindrome is a string that reads the same forwards and backwards. Let L_{pal} denote the set of palindromes over the alphabet $\Sigma = \{0, 1\}$. Is L regular?

   Solution: Assume that L is regular and let n be the integer of the Pumping Lemma, corresponding to L. $w = 0^n 1 0^n$ is a palindrome and therefore a member of L. As per the Pumping Lemma, w can be broken up as $w = xyz$, where

   (i) $|xy| \leq n$,
   (ii) $y \neq \epsilon$, and
   (iii) $xy^k z \in L$, for all $k \geq 0$.

   However, this means that y is a non-empty string consisting of 0's only. By pumping y up, we clearly get a string, which has more 0's to the left of the 1, as opposed to the right. Such a string cannot be a palindrome, but as per the Pumping Lemma, it should belong to L. This is the desired contradiction, from which we can conclude that L is not regular. $\Box$

3. In class, we partially proved that homomorphisms preserve regularity. In the inductive, stage, we only considered the case in which the regular expression E can be decomposed as $F+G$. Write the proof for the case in which $E = F \cdot G$.

   Solution: From the manner in which homomorphisms are applied to regular expressions, we know that

$$h(E) = h(F) \cdot h(G)$$

   Therefore,

$$\begin{aligned} L(h(E)) &= L(h(F) \cdot h(G)) \\ &= L(h(F)) \cdot L(h(G)) \end{aligned}$$

   Now, by definition of the "$\cdot$" operator, $L(E) = L(F) \cdot L(G)$. Accordingly,

$$\begin{aligned} h(L(E)) &= h(L(F) \cdot L(G)) \\ &= h(L(F)) \cdot h(L(G)) \end{aligned}$$

By the inductive hypothesis, $L(h(F)) = h(L(F))$ and $L(h(G)) = h(L(G))$. It therefore follows, that $L(h(E)) = h(L(E))$, which is what we are required to show. $\Box$

4. Let L be a language over an alphabet Σ, such that $a \in \Sigma$. The language $Qot_a(L)$ is defined as the set of strings $w \in \Sigma^*$, such that $wa \in L$. Is $Qot_a(L)$ regular?

   Solution: Consider the DFA $D_1 = (Q, \Sigma, \delta, q_0, F)$ of L; we construct the following DFA $D_2 = (Q, \Sigma, \delta, q_0, F')$, where a state $q_i \in F'$, if and only if, $\delta(q_i, a) \in F$. It is clear that D_2 accepts precisely those strings w, such that $wa \in L$. In other words, D_2 is the DFA accepting $Qot_a(L)$, thereby establishing that $Qot_a(L)$ is regular. $\Box$

5. Given two regular languages L_1 and L_2, how would you check if they have at least one string in common.

   Solution: We assume that the DFAs of the two languages, viz., D_1 and D_2, are given to us. If not, then we can always construct the DFAs from the regular expressions corresponding to the respective languages. Since L_1 and L_2 are regular, we know that $L_3 = L_1 \cap L_2$ is also a regular language; indeed, we can use the procedure on pages $135 - 136$ of [HMU01] to construct the DFA for L_3. Now, all that we need to do is to check whether there exists a path from the start state of D_3 to a final state of D_3. If there exists such a path, then L_1 and L_2 have at least one string in common; otherwise, they have no common string. $\Box$

References

[HMU01] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *"Introduction to Automata Theory, Language, and Computation"*. Addison–Wesley, 2nd edition edition, 2001.