Automata Theory - Quiz II (Solutions)

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

1 Problems

1. Let $L$ be a language over $\Sigma = \{0, 1\}$ defined as follows: $L = \{ w \in \Sigma^* \text{ and } w \text{ ends in } 01 \text{ or } 10 \text{ or } 00 \text{ or } 11 \}$. Is $L$ regular?

Solution: There are two approaches to this problem.

In the first approach, we observe that the strings that end in 01 can be represented by the regular expression $(0+1)^*01$. Likewise, strings that end in 00 and 11 can be represented by the regular expressions $(0 + 1)^*00$ and $(0 + 1)^*11$ respectively. Since $L$ is the union of these regular languages, $L$ must be regular.

Alternatively, we can observe that every string in $\Sigma^*$, which has length at least 2, must end with 01, 10, 00 or 11. In other words, $L$ includes all strings in $\Sigma^*$ which have length at least two. Now, the language $L'$ which is constituted of strings which have length strictly less than two is finite ($\{\epsilon, 0, 1\}$) and therefore regular. Since $L = \Sigma^* - L'$, it follows that $L$ is regular. □

2. Let $L$ be a regular language over an alphabet $\Sigma$. Let $L_1$ and $L_2$ denote two languages over the same alphabet, such that $L = L_1 \cup L_2$. Should each of $L_1$ and $L_2$ also be regular?

Solution: This is somewhat of a trick question. We know that if $L_1$ and $L_2$ are regular, then so is $L_1 \cup L_2$. But the converse is not true. For instance, $\Sigma^*$ is a regular language; but it can be decomposed into two languages $L_1 = \{ w \mid w \text{ has an equal number of } 0's \text{ and } 1's \}$ and $L_2 = \{ w \mid w \text{ has an unequal number of } 0's \text{ and } 1's \}$, both of which are not regular.

In similar fashion, consider the language $L = 0^*1^*$, which is clearly a regular language. But $L$ can be written as $L_1 \cup L_2$, where $L_1 = \{0^i1^j, i \geq 0\}$ and $L_2 = \{0^i1^j, i \neq j, i, j \geq 0\}$. We have already shown (in class) that neither $L_1$ nor $L_2$ is regular. □

3. Let $L$ be a regular language over an alphabet $\Sigma$. Assume that you are given the DFA $D$ of $L$. How would you efficiently check that $L = \Sigma^*$?

Solution: Interchange the final and non-final states of $D$ to get a new DFA $D'$. Observe that $D'$ the complement of $L$, i.e., $L^c$. The crucial observation is that $L = \Sigma^*$ if and only if $L^c = \phi$. Using simple breadth-first search (polynomial time and hence efficient), check if there exists a path from the start state of $D'$ to any final state. If there exists even one such path, it means that $L^c$ contains at least one string and is therefore non-empty. Since $L^c \neq \phi$, $L \neq \Sigma^*$. Likewise, if there does not exist a path from the start state of $D'$ to a final state, then $L^c = \phi$ and hence $L = \Sigma^*$. □

4. Write a Context-Free Grammar for the language $L$ defined as follows:

$L = \{ w \mid w \in \{0, 1\}^* \text{ and } w \text{ contains two consecutive } 0's. \}$

Solution: One approach to this problem is through recognizing that $L$ is defined by the regular expression $(0 + 1)^*00(0 + 1)^*$. □
Note that \((0 + 1)^*\) can be captured by the following grammar

\[
S \rightarrow 0S \\
S \rightarrow 1S \\
S \rightarrow \epsilon
\]

Therefore, a CFG for \(L\) is given as:

\[
S \rightarrow S_i 0S_i \\
S_i \rightarrow 0S_i | 1S_i | \epsilon
\]

\(\square\)

5. Consider the CFG defined by:

\[
S \rightarrow aS \\
S \rightarrow Sb \\
S \rightarrow a \\
S \rightarrow b
\]

Argue that no string derived from \(S\) can have \(ba\) as a substring. 

\textit{Hint: Use induction on the length of the strings derived from \(S\).}

\textbf{Solution:} Let \(w\) denote a string derived from \(S\). Consider the case in which \(|w| = 1\). As per the grammar, it is clear that \(w = a\) or \(w = b\) and hence \(ba\) is not a substring of \(w\). Assume that if \(w\) is derived from \(S\) and \(|w| \leq n\), then \(ba\) is not a substring of \(w\). Now consider the case, in which \(w\) is a string of length \(n + 1\). Since \(S \Rightarrow^* w\), it must be the case that the first step in the derivation used the production \(S \rightarrow aS\) or the production \(S \rightarrow Sb\). In the former case, \(w\) must have the form \(a \cdot x\), where \(S \rightarrow x\) and \(|x| = n\). As per the inductive hypothesis, \(x\) cannot contain \(ba\) as a substring. But if \(ba\) is not a substring of \(x\), then it is not a substring of \(a \cdot x\) either and the claim holds. In the latter case, \(w\) must be of the form \(x \cdot b\), where \(S \rightarrow x\) and \(|x| = n\). Once again, as per the inductive hypothesis, \(x\) does not contain \(ba\) as a substring and hence neither does \(w \cdot x \cdot b\). We apply the principle of mathematical induction to conclude that no string derived from \(S\) can have \(ba\) as a substring. \(\square\)