CS 381 – HW2 Solutions

1. Prove that if \( L_1 \) and \( L_2 \) are regular languages, then so is: \( L_1 \setminus L_2 = \{ w \in L_1 \mid w \notin L_2 \} \)

Method:

We can prove this by constructing a DFA for \( L_1 \setminus L_2 \) using the DFAs for \( L_1 \) and \( L_2 \). Let's denote \( \text{DFA}_1 = (Q_1, \Sigma, q_1^{\text{start}}, \delta_1, \text{ACCEPT}_1) \) as the DFA for \( L_1 \) and \( \text{DFA}_2 = (Q_2, \Sigma, q_2^{\text{start}}, \delta_2, \text{ACCEPT}_2) \) as the DFA for \( L_2 \). We will call the DFA we construct \( \text{DFA}' = (Q, \Sigma, q'^{\text{start}}, \delta, \text{ACCEPT}) \).

Construction:

\( \text{DFA}' \) clearly needs the same language \( \Sigma \) as both \( \text{DFA}_1 \) and \( \text{DFA}_2 \). Our DFA will have a state for every pair of states \( q_1 \) in \( Q_1 \) and \( q_2 \) in \( Q_2 \): \( Q = \{(q_1, q_2) \mid q_1 \in Q_1, q_2 \in Q_2 \} \). The start state \( q'^{\text{start}} \) will be the state pair \( (q_1^{\text{start}}, q_2^{\text{start}}) \). Our transition function will separately map the initial \( \text{DFA}_1 \) state to the next \( \text{DFA}_1 \) state and the initial \( \text{DFA}_2 \) state to the next \( \text{DFA}_2 \) state according to the transition functions \( \delta_1 \) and \( \delta_2 \): \( \delta((q_1, q_2)) = (\delta_1(q_1), \delta_2(q_2)) \). The accept states for \( \text{DFA}' \) will be all states \( (q_1, q_2) \) such that \( q_1 \) is an accept state of \( \text{DFA}_1 \) and \( q_2 \) is not an accept state of \( \text{DFA}_2 \): \( \text{ACCEPT} = \{(q_1, q_2) \mid q_1 \in \text{ACCEPT}_1, q_2 \notin \text{ACCEPT}_2 \} \).

Correctness:

\( \text{DFA}' \) starts with state \( (q_1^{\text{start}}, q_2^{\text{start}}) \) and separately tracks the progress of its input through \( \text{DFA}_1 \) and \( \text{DFA}_2 \). We only accept an input string \( w \) if \( \text{DFA}_1 \) would have accepted \( w \). Thus \( w \in L_1 \). Also, we only accept \( w \) if \( \text{DFA}_2 \) would have rejected \( w \). Thus \( w \notin L_2 \). This is the exact description of \( L_1 \setminus L_2 \).

Another Method:

We can also prove the claim by noting that \( L_1 \setminus L_2 = L_1 \cap L_2^c \) where we’ve used \( L_2^c \) to denote the complement of \( L_2 \). We know that regular languages are closed under complement and intersection, so \( L_1 \setminus L_2 \) must be a regular language.

2. Given a DFA \( M = (Q, \Sigma, q_0, \delta, F) \) and \( p, q \in Q \), let \( L(M, p, q) = \{ w \mid q = \delta(p, w) \} \). Prove/refute each of the following claims.

Problem i:

For every \( M, p, q \) as above and every \( x, y \in \Sigma^* \), if \( z \in L(M, p, q) \) and \( y \in L(M, q, p) \) then \( xy \in L(M, p, p) \).

Solution i:

This fact can be proven rigorously using induction on the length of \( y \) and the definition of \( \delta \). A more conceptual proof follows: \( x \in L(M, p, q) \) means that \( x \) takes our machine from state \( p \) to state \( q \). \( y \in L(M, q, p) \) means that \( y \) takes our machine from state \( q \) to state \( p \). Let's now start in state \( p \) and input \( xy \). The machine first reads \( x \), which leaves us in state \( q \). The machine then reads \( y \), which takes us to state \( p \). Thus \( xy \in L(M, p, p) \).

Problem ii:
For every $M, p, q$ as above and every $y, z \in \Sigma^*$, if $yz \in L(m, p, q)$ then there exist some $r \in Q$ such that for every $x \in L(M, r, r)$ and every $i \in \mathbb{N}$, $yx^i z \in L(M, p, q)$.

**Problem ii:**

First, let's define $r = \hat{\delta}(p, y)$. Observe that this means that $y \in L(M, p, r)$ and $z \in L(M, r, q)$. Now let's show that for this choice of $r$ it is true that for every $x \in L(M, r, r)$ and every $i \in \mathbb{N}$ we have $yx^i z \in L(M, p, q)$. Note that for any specific $i$, $x \in L(M, r, r) \rightarrow x_i \in L(M, r, r)$ because we can inductively apply the result of part (i) to reduce the length of the concatenation. Let's now denote $x^i$ as $x'$, observing that $x' \in L(M, r, r)$. We then need to show $yx' z \in L(M, p, q)$. But this is true because $y$ takes state $p$ to state $r$, $x'$ takes state $r$ to state $r$, and $z$ takes state $r$ to state $q$. Thus applying $yx' z$ in sequence takes us from state $p$ to state $r$. Hence: $yx' z = yx^i z \in L(M, p, q)$.

3. Recall that a language is regular if it is computable by some DFA.

**Problem i:**

Prove that any intersection of finitely many regular languages is a regular language.

**Solution i:**

We know that regular languages are closed under intersection. That is, for any two regular languages $L_1$ and $L_2$, we know that $L' = L_1 \cap L_2$ is regular. The problem of intersecting $N$ regular languages $L_1 \cup L_2 \ldots L_N$ can be directly translated to the problem of intersecting $N - 1$ regular languages $(L' = L_1 \cup L_2) \cup L_3 \ldots L_N$ where we know $L'$ is regular because regular languages are closed under intersection. We can repeat this translation $N - 1$ times for any finite $N$ to produce a single regular language - the intersection of the $N$ original languages. Thus the intersection of finitely many regular languages is regular.

**Problem ii:**

Prove that there exists a set $W$ of regular languages so that the intersection of all languages in $W$ is not regular.

**Solution ii:**

We can prove this constructively. First off, we know that irregular languages exist. Given an irregular language $I$ we can construct $I$ as an infinite intersection of regular languages as follows:

$$L^*_w \subseteq \Sigma^* = \Sigma^* - w$$

$$W = \{L_w \mid w \notin I\}$$

I claim first that every language in $W$ is regular and second that the intersection of all languages in $W$ leaves us with the irregular language $I$. Note that $L_w$ is just the complement of the language $w$, which is finite and therefore regular. It follows that, because regular languages are closed under complements, $L_w$
is regular. Next, the intersection of all elements in $W$ is defined as only those elements that are in every single language in $W$. The only elements in every language in $W$ are the elements of $I$. Clearly the elements of $I$ are in every language in $W$. Also, any element not $x \notin I$ is absent from some language in $W$, namely $L_x$. Thus we have constructed an irregular language from the intersection of an infinite number of regular languages.

**Problem iii:**
Find a set $W$ of regular languages such that $W$ is infinite and yet the intersection of all the languages in $W$ is an infinite regular language.

**Solution iii:**
Many examples work. We can construct one example by defining our set $W$ to be composed of individual languages $L_w$ where $L_w$ is some regular language (say $0^*$) unioned with some unique string $w \notin 0^*$. The intersection of any two of these languages will clearly be only $0^*$, a regular language (clearly the intersection of all elements of $W$ is also $0^*$ because every element contains at least $0^*$). There are an infinite number of such languages, because there are an infinite number of distinct $w \notin 0^*$. And ... that’s it.

4. Find a set $W$ consisting of infinitely many languages over $\{0, 1\}$ so that:
(i) Each language in $W$ is infinite.
(ii) Each language in $W$ is regular.
(iii) $L_1 \neq L_2 \in W \rightarrow L_1 \cap L_2 = \emptyset$.

**Solution:**
Many examples work. We can construct one example by defining:

$$L_i = \{w \mid w = 0^i1^j : j \in \mathbb{Z}^+\}$$

$$W = \{L_i \mid i \in \mathbb{Z}^+\}$$

Clearly, each $L_i$ is infinite because we can trail $0^i$ with any number of 1s we want. Also, each $L_i$ is regular because it is the concatenation of two regular languages: $\{0^i\}$ is regular for any specific $i$, and we know $1^*$ to be regular. Finally, no two languages share any element because strings from different languages have a different number of leading 0s. Thus $W$ satisfies properties (i), (ii), and (iii).

5. Construct a DFA, $M$, such that $L(M) = L(N)$ where $N$ is the given NFA (see Figure 1).

6. Construct a NFA, $M$, over $\Sigma = \{1, 2, 3, 4, 5\}$ such that $M$ has only five states and $L(M) = \{w = \sigma_1\sigma_2 \ldots \sigma_{|w|} : 1 \leq i < j \leq |w| \rightarrow \sigma_i \leq \sigma_j\}$. In other words, the numbers that are the letters in $w$ appear in non-decreasing order (see Figure 2).
Figure 1: Observe that the given NFA described the language \( L = a^* b^* c^* \). The above DFA describes the same language. We have three states to keep track of the most recent symbol read, and in the case that we ever read a ‘smaller’ symbol we go to a non-accepting garbage state.

Figure 2: Observe that the language we desire is \( 1^* 2^* 3^* 4^* 5^* \). This problem is very similar to the NFA given in Problem 5, and we can thus construct a similar NFA.