Problem 1 [15 points]

Prove that, for any deterministic FSA $A = (Q, \Sigma, \delta, q_0, F)$,

$$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$$

for $x, y \in \Sigma^*$. Use the definition of $\hat{\delta}$ provided in lecture:

(1) $\hat{\delta}(q, \epsilon) = q$
(2) $\hat{\delta}(q, w\sigma) = \delta(\hat{\delta}(q, w), \sigma)$

where $\epsilon$ is the empty string, $w \in \Sigma^*$, and $\sigma \in \Sigma$.

Solution

The proof is by induction on $|y|$.

**Base:** $|y| = 0$. If $|y| = 0$, then $y = \epsilon$.

$$\hat{\delta}(q, xy) = \hat{\delta}(q, x) \quad \text{by definition of } y$$
$$= \hat{\delta}(\hat{\delta}(q, x), \epsilon) \quad \text{by definition (1) of } \hat{\delta}$$
$$= \hat{\delta}(\hat{\delta}(q, x), y) \quad \text{by definition of } y$$

**Induction:** $|y| = n + 1$. We rewrite $y$ as $w\sigma$ where $w \in \Sigma^*$ and $\sigma \in \Sigma$. Thus $|w| = n$, and we assume by the inductive hypothesis that $\delta(q, xw) = \hat{\delta}(\hat{\delta}(q, x), w)$.

$$\delta(q, xy) = \delta(q, xw\sigma) \quad \text{by definition of } y$$
$$= \delta(\hat{\delta}(q, xw), \sigma) \quad \text{by definition (2) of } \hat{\delta}$$
$$= \delta(\hat{\delta}(\hat{\delta}(q, x), w), \sigma) \quad \text{by inductive hypothesis}$$
$$= \hat{\delta}(\hat{\delta}(q, x), w\sigma) \quad \text{by definition (2) of } \hat{\delta}$$
$$= \hat{\delta}(\hat{\delta}(q, x), y) \quad \text{by definition of } y$$
Problem 2 [15 points]

Note: The following problems require you to design finite state machines. We ask that you do not use any automated tools to solve these problems as you will not have access to such tools for similar problems on the exams.

Give deterministic FSAs accepting the following languages. Please make sure your DFSAs are fully defined: clearly mark start and final states and do not use shorthand notation for arcs.

1. [5 points] The set of strings over \{a, b, c\} in which the substring bc never occurs.

![Diagram 1](image1)

Computations in state \(q_1\) have seen a prefix ending in \(b\). Computations in the sink state \(q_2\) have seen a prefix containing \(bc\).

2. [5 points] The set of strings over \{1, 7\} ending in 11711.

![Diagram 2](image2)

Computations in states \(q_1, q_2, q_3, q_4,\) and \(q_5\) have seen a prefix ending in \(1, 11, 117, 1171,\) and \(11711\), respectively.

3. [5 points] The set of strings over \{0, 1\} which are divisible by three when interpreted as a binary number (ignoring leading zeroes).

For example 00000\(_b\) = 0, which is divisible by 3, so 00000 should be accepted. 001001\(_b\) = 9, and thus should be accepted also. 101\(_b\) = 5, is not divisible by 3, and thus should be rejected.

Note: \(\epsilon\) should be interpreted as 0, and thus should be accepted.
Computations in state $q_i$ have seen a prefix $x$ where $x_b \equiv i \mod 3$. Note that appending 0 to the right of a string multiplies it by two: $(x0)_b = 2 \times x_b$. Appending 1 to a string multiplies by two and adds one: $(x1)_b = 2 \times x_b + 1$. The transitions are defined by:

$$
\delta(q_i, 0) = q(2i \mod 3)
$$
$$
\delta(q_i, 1) = q(2i+1 \mod 3)
$$

Problem 3 [10 points]

Give a non-deterministic FSA (possibly with $\epsilon$ moves) accepting the following language. Please clearly mark start and final states.

1. The set of strings over \{1, 7\} ending in 11711. How does your NDFSA differ from the DFSA you constructed in Problem 2.2?

The NDFSA is simpler than the equivalent DFSA of Problem 2 because it can non-deterministically “guess” when the fifth symbol from the right end of the input string has been read.

Problem 4 [15 points]

Give regular expressions for each of the following languages:

1. [7 points] The set of strings over \{a, b, c\} in which the substring $bc$ never occurs.

   $(a + c + bb^*a)^*b^*$,

   which can be simplified to

   $(c + b^*a)^*b^*$

   Every string of $b$'s must either end the string or be followed by an $a$.

2. [8 points] The language described in Problem 2.3: The set of strings over \{0, 1\} which are divisible by three when interpreted as a binary number (ignoring leading zeroes).

   For example $0000_0 = 0$, which is divisible by 3, so 00000 should be accepted. $001001_b = 9$, and thus should be accepted also. $101_b = 5$, is not divisible by 3, and thus should be rejected.

   Note: $\epsilon$ should be interpreted as 0, and thus should be accepted.

   $(0 + 1(01^*0)*1)^*$
Examine the FSA in Problem 2.3 and consider all of the sequences of moves beginning in \( q_0 \) that lead back to \( q_0 \). One possibility is to take the immediate loop from \( q_0 \) to \( q_0 \) emitting a 0. Another possible loop is to go to \( q_1 \) via the 1 arc, then back from \( q_1 \) to \( q_0 \) along the other 1 arc. While we’re in \( q_1 \), however, we may take zero or more detours through \( q_2 \) via the path 01∗0.

Thus, we may either follow the \( q_0 \) to \( q_0 \) loop labeled 0, or go to \( q_1 \) and back, possibly with detours. We may follow either of these two loops as many times as we want, in any order, intermingling them if we wish. This gives us a regex of the form \((\text{Loop } 1 + \text{Loop } 2)^*\). Plugging in 0 for Loop 1 and 1(01∗0)1 for Loop 2 gives the final regular expression.

**Problem 5 [20 points]**

Use the Pumping Lemma for regular languages to show that the following language over \( \Sigma = \{a, b\} \) is not regular: strings consisting of an equal number of \( a \)'s and \( b \)'s such that no prefix of the string contains more \( b \)'s than \( a \)'s. Intuitively, this is the language of balanced parentheses (imagine \( a \) is open parenthesis and \( b \) is close parenthesis).

**Solution**

We will prove that \( L \) is not regular by contradiction. If we assume that \( L \) is actually regular, then the language satisfies the Pumping Lemma, which states that there exists \( n \) such that for all \( y \in L \) with \( |y| \geq n \), \( y \) can be decomposed into \( y = uvw \) such that

1. \(|uv| \leq n\)
2. \(|v| \geq 1\)
3. \(uv^i w \in L \forall i \geq 0\)

Given any \( n \), let \( y = a^n b^n \). It’s clear that \( y \in L \) and \(|y| = 2n \geq n \), so the Pumping Lemma applies. The decomposition of \( y \) under (1) and (2) is that \( uv = a^t \), where \( t \leq n \), and that \( v = a^r \), where \( 1 \leq r \leq t \leq n \). Since \( y = uvw \), we have that \( w = a^{n-t} b^n \). By (3) we also know that \( uv^i w \in L \forall i \geq 0 \). For example, if \( i = 2 \), we insert an extra \( v \), producing the string

\[ uv^2 w = uvvw = a^{t-r} a^r a^{n-t} b^n = a^{n+r} b^n. \]

By (3), this string should be in the language.

But since \( r \geq 1 \), the string \( uv^2 w = a^{n+r} b^n \) contains an unequal number of \( a \)'s and \( b \)'s, and so is not in \( L \). The Pumping Lemma says that \( uv^2 w \) must be in \( L \) if \( L \) is regular, so this is a contradiction and we can conclude that \( L \) is in fact not regular.

**Problem 6 [25 points]**

For each of the following statements, answer whether the claim is true or false, and give a short (two- to three-sentence) explanation if true or counterexample if false.

1. **[5 points]** Let \( L_1 \subset L_2 \). If \( L_1 \) is not regular, then \( L_2 \) must also be not regular.

   **False.** As a counterexample, let \( L_1 = \{0^n 1^n \mid n > 0\} \) and \( L_2 = \Sigma^* \). \( L_1 \) is non-regular and \( L_2 \) is regular, yet \( L_1 \subseteq L_2 \).
2. [5 points] \( L = L_1 \cup L_2 \). If \( L_1 \) and \( L \) are regular languages, then \( L_2 \) must also be a regular language.

**False.** As a counterexample, let \( L_1 = \Sigma^* \), which is regular, and \( L_2 = \{0^n1^n \mid n > 0 \} \), which is not regular. Yet \( L = L_1 \cup L_2 = \Sigma^* \) is regular.

3. [5 points] \( L = \bigcap_{i=1}^{\infty} L_i \). If all of the \( L_i \) are regular languages, then \( L \) is also a regular language.

**False.** As a counterexample, let \( L_i = \{0,1\}^* \setminus \{0^i1^i\} \).

Each language \( L_i \) is the complement of a set that consists of a single word, and is thus regular. However, \( L = \{0,1\}^* \setminus \{0^i1^i \mid i \geq 1 \} \) is the complement of \( \{0^i1^i \mid i \geq 1 \} \) which is not regular, thus \( L \) is not regular.

4. [5 points] Let \( \alpha \) and \( \beta \) be regular expressions. \( L((\alpha^*\beta)^*) = L((\alpha + \beta)^*) \)

**True.**

\((\alpha + \beta)^* \) represents the set of all finite sequences of \( \alpha \)'s and \( \beta \)'s. Consider the location of the \( \beta \)'s in any such sequence. Every \( \beta \) may be preceded by 0 or more \( \alpha \)'s. The final \( \beta \) may also be followed by 0 or more \( \alpha \)'s. This implies that the set of all finite sequences of \( \alpha \)'s and \( \beta \)'s is the same as the set of sequences of the form \( \alpha^*\beta^*\alpha^* \).

5. [5 points] Let \( \alpha, \beta \) and \( \gamma \) be regular expressions.

If \( L(\beta + \alpha \gamma) \subseteq L(\gamma) \), then \( L(\alpha^*\beta) \subseteq L(\gamma) \).

**True.**

We show by induction that \( \forall n \ L(\alpha^n\beta) \subseteq L(\gamma) \).

**Base case** \( (n = 0) \): \( L(\alpha^0\beta) = L(\beta) \subseteq L(\beta + \alpha \gamma) \subseteq L(\gamma) \).

**Induction step:** Assume \( L(\alpha^n\beta) \subseteq L(\gamma) \). Then

\[
L(\alpha^{n+1}\beta) = L(\alpha)L(\alpha^n\beta)
\]

\( \subseteq L(\alpha)L(\gamma) \) (by the inductive hypothesis)

\( = L(\alpha\gamma) \subseteq L(\beta + \alpha \gamma) \subseteq L(\gamma) \).