Homework 2 solutions

1. (10 + 5 = 15 points)
(i) Define \( \text{Rev}(L) = \{ x^R \mid x \in L \} \). Note \( x^R \) is the reversal of the string \( x \).
Thus, e.g., if \( L = \{ ab, aba, babb \} \) then \( \text{Rev}(L) = \{ ba, aba, bbab \} \).

Let \( L_1 \) be a regular language that is recognized by a DFA \( M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1) \).
Provide a construction (be precise) to create a DFA or an NFA to accept \( \text{Rev}(L_1) \).

Intuitively, the idea is to ‘reverse’ the starting and final states as well as all the transitions in \( M_1 \) to generate an NFA \( N_2 \) which accepts \( \text{Rev}(L_1) \).

The reversal of the transitions causes \( \Delta_2(p, a) = \{ q \mid \delta_1(q, a) = p \} \)
for each state \( p \in Q_1 \) (note that \( \Delta_2(p, a) \) can be a set of states).

In order to have a single ‘Start State’ in \( N_2 \), a ‘Start State’ is added with \( \epsilon \)-transitions to each state in \( F_1 \) such that \( \Delta_2(\text{‘Start State’}, \epsilon) = F_1 \).

Ultimately, the construction of the NFA \( N_2 \) that will accept \( \text{Rev}(L_1) \) is as follows:
\( N_2 = (Q_2, \Sigma, \Delta_2, s_2, F_2) \) where:

\[ Q_2 = Q_1 \cup \text{‘Start State’} \]
\[ \Sigma = \Sigma \]
\( \Delta_2 \): as described above
\( s_2 = \text{‘Start State’} \) (which has \( \epsilon \) transitions to each state in \( F_1 \))
\( F_2 = \{ s_1 \} \)

(ii) Apply your construction to the DFA given below.

Input DFA \( M_1 \) which accepts \( L \):

![DFA Diagram]

NFA \( N_2 \) that accepts \( \text{Rev}(L) \) where \( L \) is accepted by the input DFA \( M_1 \):
2. (10 X 3 = 30 points)
Give regular expressions that denote the following sets:

a. The set of strings where the third last symbol is an ‘a’. Here, \( \Sigma = \{a, b\} \).

Strings in this set start with any number \( \geq 0 \) of a’s or b’s in any order; this is represented by \((a \mid b)^*\). Then, the third to last symbol in the string must be an ‘a’, represented by ‘a’. Finally, the last two symbols can be either a or b, represented by \((a \mid b)(a \mid b)\).

Resulting regular expression: \((a \mid b)^*a(a \mid b)(a \mid b)\)

b. The set of strings of the form \( w_1w_2 \) where \( w_1 \) and \( w_2 \) belong to \( \{a, b\}^* \) and \( w_2 \) contains a ‘b’ if and only if \( w_1 \) has at least 2 occurrences of ‘a’.

First, look at the case where \( w_1 \) has at least 2 occurrences of ‘a’, so \( w_2 \) contains a ‘b’:

\( w_1 \) with at least 2 occurrences of ‘a’ begins with any number \( \geq 0 \) of a’s or b’s in any order; this is represented in a regular expression by \((a \mid b)^*\). Then ‘must’ contain an ‘a’, which is represented as an ‘a’ in a regular expression. Then there is another set of any number \( \geq 0 \) of a’s or b’s in any order, another ‘a’, and finally a third set of any number \( \geq 0 \) of a’s or b’s in any order.

The regular expression for \( w_1 \) here is \((a \mid b)^*a(a \mid b)^*a(a \mid b)^*\).

Next, \( w_2 \) with a ‘b’ begins with any number \( \geq 0 \) of a’s or b’s in any order, then it ‘must’ contain an ‘b’, then concludes another set of any number \( \geq 0 \) of a’s or b’s.

The regular expression for \( w_2 \) here is \((a \mid b)^*b(a \mid b)^*\).
The regular expression for the ‘c’ between \( w_1 \) and \( w_2 \) is simply ‘c’. 

Therefore, the complete regular expression for this ‘first’ case is

\[(a | b)^*a(a | b)^*a(a | b)^*c(a | b)^*b(a | b)^*\].

Next, look at the case where \( w_1 \) has \( \leq 2 \) occurrences of ‘a’, so \( w_2 \) does not contain any b’s:

\( w_1 \) with \( \leq 2 \) occurrences of ‘a’ begins with any number \( \geq 0 \) of b’s; this is represented in a regular expression by \( b^* \). Next, it can contain an ‘a’ but doesn’t have to; this is represented by \((a | \epsilon)\) in a regular expression, with \( \epsilon \) corresponding to an empty string. \( w_1 \) in this case concludes with \( \geq 0 \) b’s, represented by \( b^* \).

The regular expression for \( w_1 \) here is \( b^*(a | \epsilon)b^* \).

\( w_2 \) without ‘b’ simply contains any number \( \geq 0 \) of a’s, represented by \( a^* \) in a regular expression.

The regular expression for the ‘c’ between \( w_1 \) and \( w_2 \) is simply ‘c’.

Therefore, the complete regular expression for this ‘second’ case is

\[b^*(a | \epsilon)b^*ca^*\].

The final regular expression is simply a disjunction of the two cases, resulting in:

\[((a | b)^*a(a | b)^*a(a | b)^*c(a | b)^*b(a | b)^*) | (b^*(a | \epsilon)b^*ca^*)\]

c. The set of strings that do not contain the substring ‘ab’. Here, \( \Sigma = \{a, b\} \).

A string without the substring ‘ab’ cannot contain any ‘b’ once there has been an ‘a’, resulting in the regular expression ‘b*a’* where the string can contain any number of b’s followed by any number of a’s.

Resulting regular expression: \( b^*a^* \)

3. (10 X 3 = 30 points) Use the pumping lemma to show that the following sets are not regular.

a. Set \( A = \{a^ib^mc^n | l=100, m > l, n > m \} \)

Given the ‘demon’s’ k value...(see p. 71 of book)

Let \(xyz = a^{100}b^{k+100}c^{k+100+1} \in \text{set } A \)

Let \(x = a^{100}, y = b^{k+100}, z = c^{k+100+1} \)

Now v in y = uvw must contain at least 1 ‘b’ since y only contains b’s and v cannot be empty

Note that \( m = k + 100 \) and \( n = k + 100 + 1 \), so \( n = m + 1 \) when \( l=1 \).

Now let \( i = 2 \).

The presence of \( i=2 \) forces the addition if at least 1 ‘b’ to y,
so now \( m \geq (k + 100 + 1) \rightarrow m \geq n \).
Therefore, it is no longer the case that \( n > m \).
And therefore, \( xuv^iwz \notin A \) when \( i = 2 \)
Therefore, the set is not regular.

b. The set \( A \) of strings of the form \( ww \) where \( w \in \{a, b\}^* \).
Thus abbaabba is included in this set but abba is not.

Given the ‘demon’s’ \( k \) value...(see p. 71 of book)
Let \( xyz = a^k b^k a^k b^k \in A \)
Let \( x = a^k b^k a^k, y = b^k, z = \epsilon \)
Now \( v \) in \( y = uvw \) must contain at least 1 ‘b’ since \( y \) only contains b’s and \( v \) cannot be empty.
Now when \( i = 2, uv^2w = b^j \) where \( j > k \)
Since \( a^k b^k a^k b^j \notin \text{language when } j > k, xuv^iwz \notin A \) when \( i = 2 \).
Therefore the set is not regular.

c. The set \( A \) of strings of the form \( a^{n_1}b^{n_2}c^{n_3}d^{n_4} \), where \( n_1 = n_3 \) or \( n_2 = n_4 \). (\( n_1, n_2, n_3, n_4 \geq 1 \))

Given the ‘demon’s’ \( k \) value...(see p. 71 of book)
Let \( xyz = a^k b^{k+1}c^k d^{k+2} \in A \)
Let \( x = \epsilon, y = a^k, z = b^{k+1}c^k d^{k+2} \)
Now \( v \) in \( y = uvw \) must contain at least 1 ‘a’ since \( y \) only contains a’s and \( v \) cannot be empty...
Now when \( i = 2, y = uv^2w = a^j \) where \( j > k \)
Since \( a^j b^{k+1}c^k d^{k+2} \notin \text{A when } j > k, xuv^iwz \notin A \) when \( i = 2 \).
Therefore, the set \( A \) is not regular.

4. (15 points) Exercise 47 (Parts a and b only) on Page 326 of the textbook.

Minimize the following DFAs. Indicate clearly which equivalence class corresponds to each state of the new automaton.

Part a:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>3F</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4F</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Using algorithm in book (see p. 84) for computing the collapsing relation \( \approx \)
for a given DFA $M$ with no inaccessible states. The algorithms marks (unordered) pairs of states \{p, q\}, where a pair \{p, q\} is marked as a reason is discovered as to why $p$ and $q$ are NOT equivalent.

\[
\begin{array}{cccc}
1 & 2 \\
- & 3 \\
- & - & 4 \\
- & - & - & 5 \\
- & - & - & - & 6
\end{array}
\]

Pass 1: mark all pairs consisting of once accept state and one nonaccept state.

\[
\begin{array}{cccc}
1 & 2 \\
X & X & 3 \\
X & X & - & 4 \\
- & - & X & X & 5 \\
- & - & X & X & - & 6
\end{array}
\]

Repeat until no changes; if there is an unmarked pair \{p, q\} such that $\{\delta(p, a), \delta(q, a)\}$ is marked for some $a \in \Sigma$, then mark \{p, q\}.

Mark \{1, 2\} since it goes to marked pair \{3, 6\} on input 'b'
Mark \{5, 1\} since it goes to marked pair \{1, 3\} on input 'b'
Mark \{6, 2\} since it goes to marked pair \{5, 1\} on input 'a'
Mark \{6, 5\} since it goes to marked pair \{1, 2\} on input 'a'

Table now looks as follows; no more values can be marked...

\[
\begin{array}{cccc}
1 & 2 \\
X & X & 3 \\
X & X & - & 4 \\
X & - & X & X & 5 \\
- & X & X & X & X & 6
\end{array}
\]

Therefore, it is the case that state 1 $\approx$ state 6, state 2 $\approx$ state 5, and state 3 $\approx$ state 4.

Here is the resulting minimized DFA:
Again using algorithm in book (see p. 84) for computing the collapsing relation \( \approx \) for a given DFA M with no inaccessible states. The algorithms marks (unordered) pairs of states \( \{p, q\} \), where a pair \( \{p, q\} \) is marked as a reason is discovered as to why p and q are NOT equivalent.

\[
\begin{array}{cc}
1 & 2 \\
2 & 5 \\
3F & 1 \\
4F & 6 \\
5 & 2 \\
6 & 4 \\
\end{array}
\]

Pass 1: mark all pairs consisting of once accept state and one nonaccept state.
Repeat until no changes: if there is an unmarked pair \( \{p, q\} \) such that \( \{\delta(p, a), \delta(q, a)\} \) is marked for some \( a \in \Sigma \), then mark \( \{p, q\} \).

Mark \( \{1, 2\} \) since it goes to marked pair \( \{3, 6\} \) on input 'b'
Mark \( \{5, 1\} \) since it goes to marked pair \( \{1, 3\} \) on input 'b'
Mark \( \{6, 2\} \) since it goes to marked pair \( \{5, 1\} \) on input 'a'
Mark \( \{6, 5\} \) since it goes to marked pair \( \{1, 4\} \) on input 'a'

Table now looks as follows; no more values can be marked...

\[
\begin{array}{cccc}
1 & 2 \\
X & X & 3 \\
X & X & - & 4 \\
- & - & X & X & 5 \\
- & - & X & X & - & 6 \\
\end{array}
\]

Therefore, it is the case that state 1 \( \approx \) state 6, state 2 \( \approx \) state 5, and state 3 \( \approx \) state 4.

Here is the resulting minimized DFA: