Solution to CS375 Homework Assignment 3 (40 points)
Due Date: February 19, 2016

1. Construct an NFA for the given regular expression: \((a+b)^*a\)     (5 points)

**Solution:** Any of following three cases.

2. Transform the two given regular expressions into NFA's: \(a^*b^*\), \(a^* + b^*\)     (10 points)

**Solution:**
For \(a^*b^*\), any of the following four cases.

For \(a^* + b^*\), we have
3. Transform the given NFA into a regular expression by deleting state 2 before deleting state 1. (10 points).

Solution:
First, we create a new start state and a new final state for the given NFA as follows:

State 2 is a state between state 1 and state 3, also a state between state 1 and state 1 (see the above illustration), so we have to compute the label for the edge between state 1 and state 3, and the label for the edge between state 1 and state 1 (a loop for state 1):

\[ \text{New}(1,3) = \text{old}(1,3) + \text{old}(1,2) \text{old}(2,2)^* \text{old}(2,3) = \emptyset + a a^* b = a a^* b \]
\[ \text{New}(1,1) = \text{old}(1,1) + \text{old}(1,2) \text{old}(2,2)^* \text{old}(2,1) = b + a a^* b \]

The DFA after eliminating state 2 looks like:

Now eliminate state 1. State 1 is a state between state 0 and state 3, so we have to compute the label of the new edge between state 0 and state 3:

\[ \text{New}(0,3) = \text{old}(0,3) + \text{old}(0,1) \text{old}(1,1)^* \text{old}(1,3) = \emptyset + a (b+aa^*b)^* aa^* b = a(b+aa^*b)^*aa^* b \]

After the elimination of state 1, we have

Since state 0 and state 3 do not have loops, elimination of these states will not change the label between them, so the final regular expression is  \[ a(b+aa^*b)^*aa^* b \]

4. Suppose we are given the following NFA over the alphabet \{a,b\}  (15 points):
(a) Find a regular expression for the language accepted by the NFA.
(b) Write down the transition table for the NFA.
(c) Use (11.8) (slide 7 of my notes: Regular Language & Finite Automata III) to transform the NFA into a DFA.

Solution:
(a) By attaching the given NFA to a new start state and a new final state, we have

If we eliminate state 1 first, since state 1 is a state between state 0 and state 2, the new label between state 0 and state 2 is:

$$\text{New}(0, 2) = a + \Lambda b^*b = a + b^*b$$

Hence, after the elimination of state 1, we have

State 2 is a state between state 0 and state 3 and also a state between state 3 and state 3, so we need to compute the labels of the new edge between state 0 and state 3 and the new edge between state 3 and state 3 (a loop for state 3):

$$\text{New}(0, 3) = \Phi + (a+b^*)\Lambda^*a = (a + b^*)a$$
$$\text{New}(3, 3) = \Phi + \Lambda\Lambda^*a = a$$

After eliminating state 2, the NFA turns out as:

After eliminating state 0, the NFA turns out as:
Finally, after eliminating state 3, we have

So, in this case, the regular expression for the language accepted by the NFA is \( a^*(a+b)^*a^*a^* \).

On the other hand, if we eliminate state 2 first,

since state 2 is a state between state 1 and state 3, a state between state 3 and state 3, and also a state between state 0 and state 3 (see the above figure), we need to compute the following labels:

\[
\begin{align*}
\text{New}(1, 3) &= \Phi + b\lambda^*a = ba \\
\text{New}(3, 3) &= \Phi + \lambda\lambda^*a = a \\
\text{New}(0, 3) &= \Phi + a\lambda^*a = aa
\end{align*}
\]

So, after eliminating state 2, we have

To eliminate state 1, note that the label for the new edge between state 0 and state 3 is:

\[
\text{New}(0, 3) = aa + \lambda b^*ba = aa + b^*ba
\]

Hence, after eliminating state 1, we have

Subsequently by eliminating state 0 and state 3, we get
Hence, in this case, the regular expression accepted by the given NFA is \( a^*(aa+b^*ba)a^* \). This expression is the same as the one obtained in the previous case (eliminating state 1 first) simply by factoring out the right most ‘a’ in both terms inside the parenthesis.

(b) The transition table for the given NFA is:

<table>
<thead>
<tr>
<th>( T_N )</th>
<th>( a )</th>
<th>( b )</th>
<th>( \Lambda )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{0,2}</td>
<td>( \Phi )</td>
<td>{1}</td>
</tr>
<tr>
<td>1</td>
<td>( \Phi )</td>
<td>{1,2}</td>
<td>( \Phi )</td>
</tr>
<tr>
<td>2</td>
<td>{3}</td>
<td>( \Phi )</td>
<td>( \Phi )</td>
</tr>
<tr>
<td>3</td>
<td>( \Phi )</td>
<td>( \Phi )</td>
<td>{2}</td>
</tr>
</tbody>
</table>

(c) To use the algorithm in slide 7 of the notes “Regular Language & Finite Automata III” to transform the NFA to a DFA:

First, construct \( \Lambda \)-closures of all the states of the given NFA, with \( \Lambda(0) \) to be used as the start state of the DFA:

\[
\begin{align*}
\Lambda(0) &= \{0, 1\} \\
\Lambda(1) &= \{1\} \\
\Lambda(2) &= \{2\} \\
\Lambda(3) &= \{2, 3\}
\end{align*}
\]

Next to get the states of the DFA, we construct a tree as follows:

The nodes of the tree are constructed through the following process:

\[
\begin{align*}
T_0(\{0,1\}, a) &= \Lambda(T_N(0,a) \cup T_N(1,a)) \\
&= \Lambda(\{0,2\} \cup \Phi) \\
&= \Lambda(0) \cup \Lambda(2) \\
&= \{0, 1, 2\} \\
T_0(\{0,1\}, b) &= \Lambda(T_N(0,b) \cup T_N(1,b)) \\
&= \Lambda(\Phi \cup \{1, 2\})
\end{align*}
\]
The distinct nodes in the above binary tree are: \{0, 1\}, \{0, 1, 2\}, \{1, 2\}, (0, 1, 2, 3), \{2, 3\}, \Phi.

These are the states of the DFA to be constructed and we have the following transition table for the DFA:
Now we replace \( \{0,1\} \) by 0
\( \{0,1,2\} \) by 1, \( \{0,1,2,3\} \) by 2, \( \{1,2\} \) by 3, \( \{2,3\} \) by 4 and \( \Phi \) by 5.

The transition function of the DFA is of the form:

<table>
<thead>
<tr>
<th>( S )</th>
<th>( a )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {0,1} )</td>
<td>( {0,1,2} )</td>
<td>( {1,2} )</td>
</tr>
<tr>
<td>( {0,1,2} )</td>
<td>( {0,1,2,3} )</td>
<td>( {1,2} )</td>
</tr>
<tr>
<td>( {0,1,2,3} )</td>
<td>( {0,1,2,3} )</td>
<td>( {1,2} )</td>
</tr>
<tr>
<td>( {1,2} )</td>
<td>( {2,3} )</td>
<td>( {1,2} )</td>
</tr>
<tr>
<td>( {2,3} )</td>
<td>( {2,3} )</td>
<td>( \Phi )</td>
</tr>
<tr>
<td>( \Phi )</td>
<td>( \Phi )</td>
<td>( \Phi )</td>
</tr>
</tbody>
</table>

The final DFA is given by: