CSE 30151 Fall 2017 Exam 1

- SIGN YOUR NAME ON THE TOP OF ALL PAGES!!!!!!
- Sit so there is an empty chair on both sides of you.
- Do all problems and return all sheets.
- This is an in-class exam, with only open book and this year’s class notes permitted, but no computers, computer searches, cell phones, or communications with or help from others. All other aspects of the ND Honor code apply. Note you CANNOT include materials such as answer keys from non-class or prior years sources.
- Show all you work in the spaces supplied, including auxiliary equations or definitions that you may have used. **There is a blank page at the end you can use for spill-over. Indicate on the original problem page if you use it.**
- Short, clear pictures and diagrams are fine - you need not add lots of explanations.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Max Points</th>
<th>Points Off</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

1. (20pt) Multiple Choice, Short Answer. For (a) to (f) use as reference the following NFA:

(a) ____________ What is $\Sigma$?

(b) ____________ What is $\delta(4, 0)$?

(c) ____________ What is $\delta(2, 1)$?

(d) ____________ What is $E(4)$?

(e) ____________ What is $E(1)$?

(f) ____________ Is the input string 0011 accepted? In any case, write the sequence of states traversed.

(g) ____________ Write the regex for the language over \{0, 1\} where the 3rd and 2nd from the right end are 1 followed by 0.

(h) ____________ True/False: all computer programming languages can be described by regular expressions.

(i) ____________ How many elements are in the set $P(\{w, x, y, z\})$?

(j) ____________ True/False: if $L$ is a regular language, is the language $L' = \{w | w$ is not in $L\}$ always regular?
Solutions:

(a) $\Sigma = \{0, 1\}$
(b) $\{3, 2\}$ - need the set brackets
(c) $\{2, 4\}$
(d) $\{4\}$
(e) $\{1, 3, 2\}$
(f) Yes: $1 \rightarrow 3 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 2 \rightarrow 4$
(g) $(0U1)^*10(0U1)$
(h) False
(i) 16
(j) Yes

Note: 2pts per question
Define \( L_1 = \{ w \mid w = a_1b_1...a_nb_n, \ n \geq 1, \ \Sigma = \{0,1\} \} \) where both bit strings \( a_1a_2...a_n \) and \( b_1b_2...b_n \) are binary numbers with the least significant digit to the left rather than right, and where the value represented by \( b_1b_2...b_n \) is one more than \( a_1a_2...a_n \).

This is essentially a representation of binary incrementation done a bit at a time. As an example, the normal binary equivalent of 13 is 1101 so this reversed representation is \( a_1a_2a_3a_4 = 1011 \). Likewise the normal representation of 14 is 1110 so this reversed representation is \( b_1b_2b_3b_4 = 0111 \). Since 14 is one more than 13, the string from \( L_1 \) that interleaves these two strings is \( a_1b_1a_2b_2a_3b_3a_4b_4 = 1001111 \). Note also that only the first \( n \) digits of the result are represented in \( b \).

(a) (Not graded) For your own benefit, write out all strings in \( L_1 \) of length 4 or less, and then a few of length 6 and 8 until you see the pattern. Is \( \varepsilon \) in the language?

(b) (10pt) Describe a regular expression that defines \( L_1 \). It may help with grading to describe various terms in your answer in English a bit.

(c) (10pt) Develop a FA that accepts \( L_1 \). Either a state diagram or transition table is ok. Hint: there is a simple 7 state DFA solution.

The latter two questions can be done in any order, and independently of each other (you are free to do one and convert that answer into the other, but that’s more work).

Solution:

The first few members of \( L_1 = \{ 01, 10, 0100, 0111, 1001, 1010, ... \} \). \( \varepsilon \) is not in it.

One form of the regex is \( 01(00 \cup 11)^* \cup 10(10)^* \cup 10(10)^*01(00 \cup 11)^* \). The first term takes care of when \( x_1 \) is a 0, and thus no carry beyond (each \( a_i \) is duplicated in \( b_i \)). The second term is the case where \( a \) is all 1’s; the third case is where \( a_i \) is 1 and \( a_{i+1} \) is a 0 (to absorb the carry), and the rest of the bs equal the as.

![DFA Diagram]

Scoring notes:

(b)

- +1 answer is a regular expression that can be read
- +3 for each part of the regular expression (the three elements seen above could be combined)

(c)
• +1 answer is an FA (state table or diagram) that can be read
• +2 for start state and a complete diagram (diagram represents a complete thought)
• +2 for correct accept states, +1 for any right accept states that did not also accept wrong strings along the minimal path to get to the accept state
• +5 all correct transitions, +4 near perfect (1-2 mistakes), +3 some missing/incorrect transitions, +2 many mistakes, +1 a couple okay transitions with an understanding that input should be handled in pairs
3. (20pt) Show using Pumping Lemma both of the following languages are non-regular.

(a) (10pt) \( L_2 = \{ W | w = a_1a_2...a_nb_1b_2...b_n, n \geq 1, \Sigma = \{0, 1\} \) and as in \( L_1 \) the string \( b_1b_2...b_n \) represents a reversed binary number that is one more than \( a_1a_2...a_n \). For example the string 1101110 is in \( L_2 \) as it represents 13 and 14. Likewise the string \( 1^n0^n \) is in \( L_2 \) as we only show \( n \) bits of the b result. Note any string must be even in length, with equal halves for the a’s and b’s.

(b) (10pt) \( P \) is the language over \( \Sigma = \{(, )\} \) of matching parenthesis, i.e. \( P = \{ w | w = w_1...w_i...w_{2n}, n \geq 0 \) (the empty string \( \epsilon \) is included) where in any prefix substring \( w_1...w_i \) there are never more “)” than “(”, and in the whole string the numbers of “(” and “)” are equal.\}

Solution:

Scoring comments:

- The strings proposed to demo non0pumping MUST be from \( \Sigma \)
- It must be clear what are the possible options for \( y \)

(a) Assume \( L_2 \) is regular. Choose string \( 1^p0^p \).

Now \( xy \) must be in the substring of all 1s. Two cases.

- \( y \) is odd. Forming \( xy^0z \) leaves an odd number of characters and thus cannot be in \( L_2 \)
- \( y \) is even, say \( y = 1^{2k} \) for some \( k \).
  
  If we look at \( xy^2z \) we now have \( 1^{p+2k}0^p \). Since we have to look at the two halves we have \( a = 1^{p+k} \) and \( b = 1^k0^p \) b is no longer 1+a so this too fails

(b) Assume \( P \) is regular. Choose string \( (p)p \).

Now \( xy \) must be all “(”s, with \( y \) having at least one (, Either case will prove non-regular:

If \( i = 0 \) there are now fewer “(” than “)”.

If \( i \geq 2 \) there are now more “(” than “)”
4. (20pt) Use the construction process in the proof of Theorem 1.54 to convert the following regular expression to an NFA:

\[ a^* (aba)^* a^* \cup (ab)^* (aa)^* \]

Show your NFA as a state diagram. You need not show each step of the constructions, i.e. you need not be totally faithful to putting all the \( \varepsilon \) edges and intermediate states in, but your diagram should still reflect a solid understanding of the construction process. This may best be shown by annotating which parts of the diagram come from which major terms of the regex.

Solution: A reasonable form

![State Diagram](image)

<table>
<thead>
<tr>
<th>Correct implementation of</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>First ( a^* )</td>
<td>2</td>
</tr>
<tr>
<td>(aba)*</td>
<td>4</td>
</tr>
<tr>
<td>Second ( a^* )</td>
<td>2</td>
</tr>
<tr>
<td>Composition</td>
<td>3</td>
</tr>
<tr>
<td><strong>First half of regex</strong></td>
<td><strong>11</strong></td>
</tr>
<tr>
<td>(ab)*</td>
<td>3</td>
</tr>
<tr>
<td>(aa)*</td>
<td>3</td>
</tr>
<tr>
<td>Composition</td>
<td>3</td>
</tr>
<tr>
<td><strong>Second half of regex</strong></td>
<td><strong>9</strong></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>20</strong></td>
</tr>
</tbody>
</table>

Table 1: Rubric for Q4
5. (20pt) Determine what regular expression the following NFA is equivalent to, using the GNFA approach. In this figure, \( \Sigma = \{0, 1, -, .\} \), where the last two symbols are the minus sign and a decimal point. Show each step. To simplify grading, remove states in order A, B, C, D, E. It is strongly recommended that you use variables like \( R_1 = \ldots \) to simplify your edge labels. You can use the blank sheet following this if you need more room. Feel free to add stuff to the figure below for your first step.
• 3 points per step,
• 2 points for final expression,
• 0.5 points off for
• Typically 1 point taken off for first occurrence of an error, but rest of problem then graded as if error was correct. Typical errors:
  – Writing $\varepsilon x$ instead of just $x$. (0.5 points)
  – Not following the order of state removal: A,B,C,D,E
  – Dropping an edge
  – Listing a path thru states as union, not concatenation
  – Not handling self-loops properly
  – Not removing accepting state status from initial accepting states
  – Not adding new start or accept state

Solution: First modify to a GNFA by adding a new start and accept state, and edges from each old accept to new one.

Then remove A and add edge from start to C with $R_1 = -\cup \varepsilon$.

Then remove B and add edges from D to E (with “.”, no “$\varepsilon$.”) and C to E (with “0.”), Combine 2 edges from C to E with $R_2 = 1 \cup 0$. 
Then remove C and add edges from start to D (with $R_3 = R_1 1$), and from start to E (with $R_4 = R_1 R_2$)

Then remove D, with edge from start to E that is paired with $R_4$ (with $R_5 = (R_3(0 \cup 1)^* \cup R_4)$) and an edge from start to accept (with $R_6 = R_3(0 \cup 1)^*$ - note concatenating $\varepsilon$ adds nothing to string)

Then remove E, with an edge from start to accept (with $R_7 = R_5(0 \cup 1)^*$, again no concatenation with $\varepsilon$ needs be listed). Also pair this new edge up with the prior edge from start to accept with $R_8 = R_6 \cup R_7$

Final Answer: $R_8 =$

$$= R_6 \cup R_7$$
$$= (R_3(0 \cup 1)^* \cup (R_5(0 \cup 1)^*)$$
$$= (R_1 1(0 \cup 1)^* \cup ((R_3(0 \cup 1)^* \cup R_1 R_2)(0 \cup 1)^*)$$
\[ \bullet = ((-\cup \varepsilon)1(0 \cup 1)^*) \cup (((R_11(0 \cup 1)^*. \cup (-\cup \varepsilon)(1 \cup 0.))(0 \cup 1)^*) \\
\bullet = ((-\cup \varepsilon)1(0 \cup 1)^*) \cup (((-\cup \varepsilon)1(0 \cup 1)^*. \cup (-\cup \varepsilon)(1 \cup 0.))(0 \cup 1)^*) \]
This is a blank sheet for your use. Label which problem you are using it for.