Midterm Examination 1  
CS 4232: Theory of Computation  

Wednesday 20 September 2017, Duration 40 Minutes

Matriculation Number: ____________

**Rules:** This test carries 20 marks and consists of 4 questions. Each question carries 5 marks; full marks for a correct solution; a partial solution can give a partial credit. Use the backside of the page if the space for a question is insufficient.

**Question 1 [5 marks]**

Determine all derivatives of the set

\[ L = \{0\}^* \cdot \{1\}^* \cdot \{0\}^* \cdot \{1\}^* \]

and convert the set of these derivatives to a minimal and complete deterministic finite automaton with alphabet \{0, 1\}. The derivative of \( L \) at \( x \) is the set \( L_x = \{ y \in \{0, 1\}^* : xy \in L \} \). Recall that a finite automaton is deterministic and complete iff for every state \( q \) and every symbol \( a \) there is exactly one successor state \( \delta(q, a) \) to which the automaton can go.

**Solution.** The derivatives are the following set: 

- \( L_0 = \{0\}^* \cdot \{1\}^* \cdot \{0\}^* \cdot \{1\}^* \)
- \( L_{01} = \{1\}^* \cdot \{0\}^* \cdot \{1\}^* \)
- \( L_{010} = \{0\}^* \cdot \{1\}^* \)
- \( L_{0101} = \{1\}^* \)
- \( L_{01010} = \emptyset \)

A table for the automaton is given as follows:

<table>
<thead>
<tr>
<th>state</th>
<th>succ at 0</th>
<th>succ at 1</th>
<th>acc/rej</th>
<th>start</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>01</td>
<td>acc</td>
<td>yes</td>
</tr>
<tr>
<td>01</td>
<td>010</td>
<td>01</td>
<td>acc</td>
<td>no</td>
</tr>
<tr>
<td>010</td>
<td>010</td>
<td>0101</td>
<td>acc</td>
<td>no</td>
</tr>
<tr>
<td>0101</td>
<td>01010</td>
<td>0101</td>
<td>acc</td>
<td>no</td>
</tr>
<tr>
<td>01010</td>
<td>01010</td>
<td>01010</td>
<td>rej</td>
<td>no</td>
</tr>
</tbody>
</table>

The finite automaton uses as names for states words \( x \) in the corresponding derivatives \( L_x \); there are many words \( x \) giving the same derivative \( L_x \) and exactly one such word is used for each derivative as state name. The automaton has five states and as each state represents a different derivative, the number of states is minimal by the Theorem of Myhill and Nerode.
Question 2 [5 marks]  

Provide a context-free grammar for the set $H = \{0^{2n}1^{3m} : n \text{ is odd}\}$ using as few non-terminals as possible.

Solution. The correct grammar for $H$ is $(\{S\}, \{0, 1\}, \{S \to 000S111111|00111\}, S)$. This grammar is context-free, as there is always only $S$ on the left side of a rule. The number of non-terminals for any grammar has to be at least one, as there must always be the start symbol $S$ and therefore no smaller number of non-terminals is possible.
Question 3 [5 marks]

Construct an nfa with 8 states for the language

\[ I = \{ w \in \{0, 1, 2, 3, 4, 5, 6, 7\}^* : w \text{ contains at least two different symbols} \}. \]

Solution. One divides the set of states into six groups

\[ Q_{0?} = \{0, 1, 2, 3\}, \quad Q_{1?} = \{4, 5, 6, 7\}, \quad Q_{20?} = \{0, 1, 4, 5\}, \quad Q_{21?} = \{2, 3, 6, 7\}, \quad Q_{30?} = \{0, 2, 4, 6\}, \quad Q_{31?} = \{1, 3, 5, 7\} \]

and makes eight states as follows: start state \( s \), one accepting state \( t \) and a state \( q_u \) for each set \( Q_u \) with \( u \in \{??0, ?0?, 0??, ??1, ?1?, ??1\} \). The state \( s \) and all states \( q_u \) are rejecting.

From the start symbol, the nfa can go to each \( q_u \) on symbols from the set \( Q_u \). Furthermore, assume that the nfa is in \( q_u \); if the next symbol \( a \) is in \( Q_u \) then the nfa goes to \( q_u \) else the nfa goes to \( t \). From \( t \), the nfa goes on all symbols to \( t \). When a word \( w \) comes up, on the first symbol \( a \) of \( w \), the non-deterministic automaton guesses a symbol \( b \) different from \( a \) in \( w \) and selects an \( u \) such that \( a \in Q_u \) and \( b \notin Q_u \); in the case that \( b \) is indeed in the word \( w \) then the automaton will eventually go from \( q_u \) to \( t \) and therefore accept the word \( w \). On the other hand, one can easily see that the nfa cannot accept the empty word and also not any word which consists only of repetitions of one symbol \( a \), as on \( a \), the nfa goes from \( s \) to a state \( q_u \) with \( a \in Q_u \) and then the nfa will remain in \( q_u \) until all \( a \) in the word are processed.

For understanding the principle behind this construction, note that the indices \( u \) indicate which bit is in common to all members of \( Q_u \) when the symbols in \( Q_u \) are written as binary three-digit numbers; the values of bits written as ? are irrelevant for membership in \( Q_u \).
Question 4 [5 marks]  
Recall that the weakest form of the Pumping Lemma for regular languages (Corollary 2.16 in the Lecture Notes) states that every regular language $J$ satisfies the following statement ($*$):

\[ \text{There is a constant } k \text{ such that every word } x \in J \text{ of length } k \text{ or more can be split into } u, v, w \text{ such that } v \neq \varepsilon \text{ and } x = uvw \text{ and } \{u\} \cdot \{v\}^* \cdot \{w\} \subseteq J. \]

Assume that the language $J$ consisting of all words in $\{0, 1, 2\}^+$ which contain as many 0 as 1. Does $J$ satisfy ($*$)?

If $J$ satisfies ($*$) then determine the smallest constant $k$ which works and explain why $k$ is correct; furthermore, prove that $J$ is not regular.

If $J$ does not satisfy ($*$), then prove that this fact; note that the non-regularity of $J$ follows in this case directly from Corollary 2.16.

**Solution.** Yes, $J$ satisfies ($*$) with $k = 3$.

To see that $k \leq 2$ is impossible, consider the word 01 and assume that it is pumped down, that is, made shorter. The possible outcomes are 0, 1 and $\varepsilon$ which are all not in $J$.

Now, if a word $x$ of length at least 3 contains 2, one can split $x$ into $u \cdot 2 \cdot w$ and $\{u\} \cdot \{2\}^* \cdot \{w\} \subseteq J$, as pumping the 2 up or down does not change the number of 0 and 1 and as $|uw| \geq 2$.

If a word $x$ of length at least 3 does not contain any 2 and is non-empty, then it contains for some number $n > 0$ exactly $n$ 0s and $n$ 1s. So these are in the same quantity and one 0 must be next to one 1. Thus the word is of the form $x = uvw$ with $v \in \{01, 10\}$. Now $\{u\} \cdot \{v\}^* \cdot \{w\} \subseteq J$, as removing or inserting $v$ into the word $x$ does put in or take out the same quantity of 0 and 1, so that in the resulting word there are as many 0 and 1; furthermore, due to $|uvw| \geq 3$ and $|v| = 2$, $|uw| \geq 1$.

The language is not regular, as it does not satisfy the traditional pumping lemma. If that would be satisfied with constant $h$ then the word $0^h1^h$ which is in $J$ would be pumped within the first $h$ symbols, that is, $0^h$ would be split into $uvw$ with $\{u\} \cdot \{v\}^* \cdot \{w\}^h \subseteq J$ where $v \in \{0\}^+$. Thus pumping would change the number of 0 without changing the number of 1, for example, pumping up once gives $0^{h+|v|}1^h$ and this word is not in $J$ as $|v| > 0$. 

END OF PAPER