## “CS 374” Spring 2014 — Homework 2

**Due Tuesday, February 11, 2014 at 5:00pm**

<table>
<thead>
<tr>
<th>Name:</th>
<th>NetID:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>NetID:</td>
</tr>
<tr>
<td>Name:</td>
<td>NetID:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Grader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Note that while each homework may be worth a different number of points, they will be scaled so that each contributes the same amount to your total course percentage.
- For any problem involving the construction of a DFA or NFA, you *must always* explain your construction.
1. Note the inductive definition of the reverse of a string \( w \in \Sigma^* \):

\[
w^R := \begin{cases} 
\varepsilon & \text{if } w = \varepsilon \\
x^Ra & \text{if } w = ax \text{ for some } a \in \Sigma \text{ and } x \in \Sigma^* 
\end{cases}
\]

A palindrome is a string that reads the same forwards and backwards, or in other words, a string \( w \) such that \( w = w^R \). Some interesting palindromes (if you ignore spaces and punctuation):

- I
- Poop!
- Madam in Eden, I'm Adam
- A man, a plan, a canal, panama
- Are we not drawn onward, we few, drawn onward to new era?

(a) Give an inductive definition for a palindrome over the alphabet \( \Sigma = \{a, b\} \).

(b) Prove (by induction) that for any string \( w \in \{a, b\}^* \), we have \( w = w^R \) if and only if \( w \) satisfies your inductive definition. In other words, prove your inductive definition is correct. You might find it useful to use the fact that for any strings \( u \) and \( v \), \((uv)^R = v^Ru^R\), as shown in the handout “definitions for strings” posted on the course website.

2. Design a DFA whose language is the set of strings over the alphabet \( \{/, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \) that correspond to valid dates of the form \( m/y \), where \( m \) is the number of a month, with no leading 0’s, and \( y \) is the number of a valid day within that month, with no leading 0’s. Thus, strings such as 4/1, 2/28, 2/29, and 12/31 should be accepted, while strings such as 04/01, 2/30, and 13/99 should be rejected. Important: Use as few states as possible; draw your DFA carefully and neatly; and explain exactly how your DFA works. You can omit transitions that go to a reject state.

3. Let \( L \) be a regular language. For each of the following variants of \( L \), show that the variant is also regular, by describing either a DFA or an NFA that recognizes it. Typically, your DFA or NFA will be described in terms of modifications of the DFA \( M \) that recognizes \( L \). Your construction should be completely specified; if you construct one machine \( M' = (Q', \Sigma, \delta', q'_0, F') \) from another \( M = (Q, \Sigma, \delta, q_0, F) \), you must give precise definitions for \( Q', \delta', q'_0, \) and \( F' \), in terms of \( Q, \delta, q_0, \) and \( F \). Briefly explain why your construction is correct.

   (a) \( \text{STUBS}(L) = \{x \mid xy \in L \text{ for some string } y \in \Sigma^* \} \)
   (b) \( \text{MINIMUM}(L) = \{x \in L \mid \text{no proper prefix of } x \text{ is in } L \} \)
   (c) \( \text{CYCLICSHIFT}(L) = \{vu \mid uv \in L \} \)

4. A linear set of natural numbers is a set of the form \( \{a + bi \mid i \in \mathbb{N}\} \). A set is semi-linear if it is the union of a finite number of linear sets. For all unary languages \( L \subseteq 0^* \), prove that \( L \) is regular if and only if \( \{i \mid 0^i \in L\} \) is semi-linear.
5. A Mealy machine is, in essence, a DFA that outputs symbols on an output tape as it reads its input and changes states. Instead of labeled edges $p \xrightarrow{a} q$ we have edges of the form $p \xrightarrow{a/b} q$ indicating that if $M$ is in state $p$ and it reads the next symbol $a$ from its input tape, then it writes the symbol $b$ on its output tape and goes to state $q$. You can find the formal definition of a Mealy machine on Wikipedia\(^1\) if this informal explanation was not sufficient. For purposes of this problem, let us extend the definition of a Mealy machine to allow output on null input, that is, transitions of the form $p \xrightarrow{\epsilon/a} q$ for any character $a$.

Draw a Mealy machine with as few states as possible that reads a ternary (base 3) number $t$ from right to left (least-significant trit first) and outputs the ternary number with value $2t$, again from right to left (least-significant trit first). For example, reading the input string 2122 from right to left, your machine should output the string 12021 from right to left. (The output on null input is needed to print the final (leftmost) trit.) Explain clearly how and why your machine works.

*6. Completely optional fun problem!

The “Firing Squad” problem is a famous problem with many interesting solutions which are easily found online. If you don’t want to spoil the fun, don’t look it up, and if you do, don’t tell anybody what you’ve read.

Design a collection of deterministic finite automata to solve the firing squad problem: Imagine a row of $n$ DFAs. The leftmost DFA is the “general”, and all of the other DFAs are “soldiers”. Each DFA is adjacent to exactly two others, one on its right, and one on its left, except for the general and the rightmost soldier, who are adjacent to only one other DFA. Initially, all DFAs start in their initial state. At some arbitrary time, the general is placed in a special state called “fire when ready”. At some later time, all soldier DFAs must simultaneously switch into a “fire” state; moreover, this must be the first time that any soldier is in the “fire” state.

The next state of a DFA may depend only on its own current state and the current state(s) of its neighbors. All DFAs must be identical, except perhaps for the general and the rightmost soldier. **The number of states cannot depend on $n$, the number of DFAs in the line.** More specifically, your design should consist of the description of a general DFA, a soldier DFA, and a rightmost soldier DFA, such that no matter how many soldiers we line up between the general and the rightmost soldier, the squad will still function properly. This restriction specifically disallows solutions where each soldier determines his distance from the general, or the middle, or the right end, by “counting” the delay of some message passed back and forth.

Here is an example of a design which fails miserably: If either of your neighbors are in “fire when ready” state, and your current state is not “fire when ready”, then switch to “fire when ready” state. If your current state is “fire when ready”, then switch to “fire” state.

In the above example, if at time 0 the general is in state “fire when ready”, then the $i$th soldier (not counting the general) enters the “fire” state at time $i + 2$. But the problem is to have all soldiers enter the “fire” state for the first time simultaneously, so this is not a solution.

Be sure to describe in English (not just with formal notation) exactly how your solution works, and argue that it does in fact work. For even more fun (i.e., not fun at all), program your solution and hand in a graphical display of instantaneous “snapshots” showing how the squad performs.

---

\(^1\) [http://en.wikipedia.org/wiki/Mealy_machine](http://en.wikipedia.org/wiki/Mealy_machine)