MA/CSSE 474 Homework #8 33 Points **Updated for Spring, 2018**

2.1 means Exercise 1 from Chapter 2. **This is a moderately substantial assignment.**

Key:

(No symbol)	Not required to be turned in. Just be sure that you can do it.
(t -7)	To be turned in and graded, worth 7 points.

1. 8.1acegklptu (additional practice problems)
2. (t-3) 8.1b
3. (t-3) 8.1d

Previous questions and answers from Piazza:

Q The last part of the definition of this language is $|x|.|y| \equiv_5 0$. Does this simply mean, the length of x times the length of y is a multiple of 5? Or does the dot notation imply something else? **A** It means what you think it means.

4. (t-3) 8.1h
5. (t-3) 8.1i
6. (t-3) 8.1j Note that $\#_n(w)$ means "the number of times that the symbol *n* appears in string *w*."
7. (t-3) 8.1n
8. (t-9) 8.7a. Do this problem by construction; produce an algorithm that takes as input a DFSM $M = (K, \Sigma, \delta, s, A)$ that accepts language L. The algorithm returns the description of a DFSM $M' = (K', \Sigma, \delta', s', A')$ that accepts *pref*(*L*). You do not have to write a formal proof that the new machine actually accepts *pref*(*L*). **Hint:** M' will have a lot of its elements in common with M, but it takes a somewhat complex calculation (based on M) to determine exactly what has to be changed. **Below I show the textbook author's solutions to parts b-d** so that you have more examples (this is in response to students who filled out the HW8 survey in Winter, 2015-16)
9. 8.2ac
10. 8.3
11. (t-6) 8.4a
12. 8.4b
13. 8.7

Solutions to problem 8.7, parts b-d are on the rest of the pages:

b) $suff(L) = \{w: \exists x \in \Sigma^* \ (xw \in L)\}$.

By construction. Let $M = (K, \Sigma, \delta, s, A)$ be any FSM that accepts L. Construct $M' = (K', \Sigma', \delta', s', A')$ to accept $suf(L)$ from M:

```
1) Initially, let $M'$ be $M$.
2) Determine the set $X$ of states that are reachable from $s$:
    a) Let $n$ be the number of states in $M'$.
    b) Initialize $X$ to $\{s\}$. .
    c) For $i = 1$ to $n - 1$ do:
        i. For every state $q$ in $X$ do:
            a. For every character $c$ in $\Sigma \cup \{\varepsilon\}$ do:
                i. For every transition $(q, c, q')$ in $M'$ do:
                    a. $X = X \cup \{q'\}$
3) Add to $M'$ a new start state $s'$. Create an $\varepsilon$-transition from $s'$ to $s$.
4) Add to $M'$ an $\varepsilon$-transition from $s'$ to every element of $X$.
```

Comments on this algorithm: The basic idea is that we want to accept the end of any string in L. Another way of thinking of this is that M'' must act as though it had seen the beginning of the string without having actually seen it. So we want to skip from the start state to anyplace that the beginning of any string could have taken us. But we can't necessarily skip to all states of M', since there may be some that aren't reachable from s by any input string. So we must first find the reachable strings.

To find the reachable strings, we must follow all possible paths from s. We need only try strings of length up to n-1 where n is the number of states of M, since any longer strings must simply follow loops and thus cannot get anywhere that some shorter string could not have reached.

We must also be careful that we only skip the beginning of a string. We do not want to be able to skip pieces out of the middle. If the original start state had any transitions into it, we might do that if we allow ε-jumps from it. So we create a new start state s' that functions solely as the start state. We then create the ε-transitions from it to all the states that were marked as reachable.

c) *reverse(L)* = $\{x \in \Sigma^* : x = w^R$ for some $w \in L\}$.

By construction. Let $M = (K, \Sigma, \delta, s, A)$ be any FSM that accepts L. M must be written out completely, without an implied dead state. Then construct $M' = (K', \Sigma', \delta', s', A')$ to accept *reverse*(L) from M:

1) Initially, let M' be M.
2) Reverse the direction of every transition in M'.
3) Construct a new state q. Make it the start state of M'. Create an ε-transition from q to every state that was an accepting state in M.
4) M' has a single accepting state, the start state of M.

d) letter substitution (as defined in Section 8.3).

Let *sub* be any function from Σ_1 to Σ_2^*. Let *letsub* be a letter substitution function from L_1 to L_2. So *letsub*(L_1) = $\{w \in \Sigma_2^* : \exists y \in L_1$ and $w = y$ except that every character c of y has been replaced by $sub(c)\}$. There are two ways to prove by construction that the regular languages are closed under letter substitution:

First, we can do it with FSMs: if L is regular, then it is accepted by some DFSM $M = (K, \Sigma, \delta, s, A)$. From M we construct a machine $M' = (K', \Sigma', \delta', s', A')$ to accept *letsub*(L). Initially, let $M' = M$. Now change M' as follows. For each arc in M' labelled x, for any x, change the label to $sub(x)$. M' will accept exactly the strings accepted by M except that, if M accepted some character x, M' will accept $s(x)$.

Or we can do it with regular expressions: If L is a regular language, then it is generated by some regular expression α. We generate a new regular expression α' that generates *letsub*(L). $\alpha' = \alpha$, except that, for every character c in Σ_1, we replace every occurrence of c in α by $sub(c)$.