Student ID: _____

Collaborators: _____

CS181 Winter 2018 – Problem Set 1

Due Friday, January 26, 11:59 pm

- Please write your student ID **and the names of anyone you collaborated with** in the spaces provided and attach this sheet to the front of your solutions. **Please do not include your name anywhere since the homework will be blind graded.**
- An extra credit of **5%** will be granted to solutions written using LaTeX. Here is one place where you can create latex documents for free: `https://www.sharelatex.com/`. The link also has tutorials to get you started. There are several other editors you can use.
- If you are writing solutions by hand, please write your answers in a neat and readable hand-writing.
- Always explain your answers. When a proof is requested, you should provide a rigorous proof.
- 20% of the points will be given if your answer is "I don't know". However, if instead of writing "I don't know" you write things that do not make any sense, no points will be given.
- The homework is expected to take anywhere between 10 to 16 hours. You are advised to start early.
- Submit your homework online on the course webpage on CCLE. You can also hand it over at the end of any class or discussion section before the deadline.

Note: *All questions in the problem sets are challenging; you should not expect to know how to answer any question before trying to come up with innovative ideas and insights to tackle the question. If you want to do some practice problems before trying the questions on the problem set,* ***we suggest trying Exercise problems 1.4, 1.5, 1.9, 1.10, and 1.11 from the book. Do not turn in solutions to problems from the book.***

The machines that we called "Finite State Machines" in class are also called "Deterministic Finite Automata (DFA)" and the machines we called "Magical Finite State Machines" in class are also called "Non-Deterministic Finite Automata (NFA)".

Hint on all construction problems: If you want to prove that L is regular, it suffices to give an NFA for it. On the other hand, if you are told to assume that L' is regular, this means that there must exist a DFA recognizing L'.

1(a).

Theorem 1a. For every DFA M there exists a penta NFA N that accepts the same language.

Proof. The machine is a DFA meaning there is only one traversed path on any given input. This is in contrast to an NFA where a machine can end up in multiple states on one input. Since DFAs can only have one current state at any given time, it is trivial to prove the theorem since it can only have one final state. If the DFA ends in an accepting state, then it entails that more than $\frac{1}{5}$ of the final states are accepting states. If instead the DFA ends in a non-accepting state, then it entails that less than $\frac{1}{5}$ of the final states are accepting states. As a result, the existence of a DFA M proves the existence of a penta NFA N that accepts the same language. □

1(b).

Theorem 1b. For every penta NFA N there exists a DFA M that accepts the same language.

Proof. To prove this theorem, let us construct a DFA from a penta NFA. We are assuming that Σ is {0,1}.

$$Let\, N = (Q, \Sigma, \delta, q_0, F). This\, is\, our\, NFA. \tag{1}$$

$$Let\, M = (Q', \Sigma, \delta', q_0', F'). This\, is\, our\, DFA. \tag{2}$$

Now we will convert the five-tuple representing the NFA to their counterparts in the DFA. Our set of states in the DFA will be the power set of the set of states in our NFA.

$$Q' = P(Q) \tag{3}$$

The alphabet stays the same through the conversion. For the transition function, each element of the power set, P(Q), becomes a state in our DFA.

$$\Sigma = \Sigma \tag{4}$$

$$\delta' : Q' \times \Sigma \to Q' \tag{5}$$

The starting state is the state that contains the subset with only q_0.

$$q_0' = \{q_0\} \tag{6}$$

The accepting states are the states where at least $\frac{1}{5}$ of the subset of states are in the original set of accepting states.

$$Let\, X = a\, state\, in\, P(Q) \tag{7}$$

$$Let\, Y = subset\, of\, states \tag{8}$$

$$F' = \{X | Y \in X, \frac{Y \in F}{Y \notin F + Y \in F} \geq \frac{1}{5}\} \tag{9}$$

This is a valid construction. Suppose an input is accepted by the DFA, meaning it ends up in one of the states in F'. This would mean that at least $\frac{1}{5}$ of the subset of states are in F, the accepting states for the NFA. We can also see that if an input is not accepted by the DFA, the opposite is true. □

2.

Theorem 2. If L is regular, then L_R is also regular.

Proof. In order to prove this theorem, we must construct an NFA such that it will accept L_R. We can construct our DFA, N, which accepts L and our NFA, N', which accepts L_R using five-tuples. We are assuming that Σ is {0,1}.

$$Let\ N = (Q, \Sigma, \delta, q_0, F) \tag{10}$$

$$Let\ N' = (Q', \Sigma, \delta', q_0{}', F') \tag{11}$$

The set of states Q can remain mostly the same in the construction since we can just reuse the states from the DFA for the NFA. However, we will add a new start state, which will be explained later. The alphabet will also remain the same.

$$Q' = Q \cup q_0{}' \tag{12}$$

$$\Sigma = \Sigma \tag{13}$$

The transition function needs to change. In our new transition function δ', we will reverse the directions of all the transitions since we will be examining the language that is reversed using our machine. If the machine is at the new start state, then the epsilon transitions will take the machine to the original final states of the DFA. If an input is received at the new start state, then it will go to an empty state and fail. However, choosing our new start state is not as simple. We cannot simply use an accepting state as our start state since it is a possibility that there are multiple start states. As a result, we will create a start state which will have ϵ transitions to each of the states represented in F. Finally, we will designate our new accepting state to be the DFA start state, q_0.

$$\delta'(x, a) = \{y : \delta(y, a) = x\} \tag{14}$$

$$\delta'(q_0{}', \epsilon) = F \tag{15}$$

$$\delta'(q_0{}', a) = \emptyset \tag{16}$$

$$q_0{}' = \{q | q \times \{\varepsilon\} \to x \in F\} \tag{17}$$

$$F' = q_0 \tag{18}$$

To validate the NFA we have constructed, we have to show that there exists a path from q_0' to a state in F'. To show this, we can use the fact that there exists a path from q_0 to a state in F. The fact that such a path exists proves that a path from a state in F to q_0 exists. Now we just need the ε transitions to connect the path to the new start state, q_0'. The path is complete and the proof is complete. □

3.

Theorem 3. Let L be any language and let L_{alt} be the set of strings in L with every other character removed. If L is regular, then L_{alt} is also regular.

Proof. We can prove this using the method we discussed in class which involved creating two separate machines that are linked by ϵ transitions. Using these two machines, we can construct our NFA to prove that L_{alt} is regular.

$$Let\, M = DFA\, accepting\, L \tag{19}$$

We will create two copies of M, named M_1 and M_2. These will be used to construct N, which is our NFA.

$$Let\, N = M_1 \,\&\, M_2 \tag{20}$$

Let's represent our machines in the form of five-tuples. Assume that Σ is {0,1}.

$$Let\, M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \tag{21}$$

$$Let\, M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \tag{22}$$

$$Let\, N = (Q', \Sigma, \delta', q_0', F') \tag{23}$$

First, let's establish our set of states, Q' in our NFA. Our set of states will be a union of Q_1 and Q_2. We will use the states in Q_1 to process inputs and we will use states in Q_2 to perform ϵ transitions to skip every other letter in the input.

$$Q' = Q_1 \cup Q_2 \tag{24}$$

The alphabet will remain the same.

$$\Sigma = \Sigma \tag{25}$$

Our transition functions involve jumping back and forth between M_1 and M_2. If an input arrives at M_1, process it and then jump to a state in M_2 using ϵ transitions. If an input arrives at M_2, then we make use of the ϵ transitions to jump back to M_1 to represent the skipping over of every other character in the input. All of this constitutes the transition function for the NFA, δ'.

$$\delta_1(q_1, a) = y \in Q_1 \tag{26}$$

$$\delta_1(x \in Q_1, \epsilon) = y \in Q_2 \tag{27}$$

$$\delta_2(x \in Q_2, \epsilon) = q_1 \tag{28}$$

$$\delta_2(x \in Q_2, a) = \emptyset \tag{29}$$

The start state for the NFA is merely the start state of M_1 as the first character in the input goes through M_1 first.

$$q_0{}' = q_1 \tag{30}$$

The accepting states are both the accepting states of M_1 and M_2. Since the machine can end up in either M_1 and M_2, we must keep both sets of accepting states.

$$F' = F_1 \cup F_2 \tag{31}$$

To prove that our NFA works, we can use the fact that if the DFA accepts an input of $x_1x_2x_3...$, then our NFA must accept an input of $x_1x_3x_5...$ Because of this fact, the proof is complete.
4.

Theorem 4. Let L be any language and let $L_{\frac{1}{2}\text{-}}$ be the set of all the first halves of strings in L. Show that if L is a regular language then $L_{\frac{1}{2}\text{-}}$ is regular.

Proof. We will once again construct an NFA to prove this theorem. Our NFA will be constructed out of 2 DFAs. We want to run these DFAs in parallel. The DFA called M will read the string from left to right. The DFA called M' will read the DFA from right to left. My plan is to have the DFAs meet in the middle of the string in order to prove that $L_{\frac{1}{2}\text{-}}$ is regular. First, let us represent our DFAs and NFA as five-tuples. We assume that Σ is $\{0,1\}$.

$$Let\ M = (Q, \Sigma, \delta, q_0, F) \tag{32}$$

$$Let\ M' = (Q', \Sigma, \delta', q_0{}', F') \tag{33}$$

$$Let\ N = (Q'', \Sigma, \delta'', q_0{}'', F'') \tag{34}$$

Before defining the elements of N, I first need to define some elements of M'. Q', Σ, and F' will remain the same as their counterparts in M. However, δ' transitions will be reversed since we are reading from the opposite direction. In addition, the start state, q_0' will instead be a fabricated state with ϵ transitions to all the states in F.

Our NFA will consist of 2 DFAs running in parallel. M will handle the first half of the string, which is x. M' will handle the second half of the string, which is y. The set of states within our NFA will be a cross product of the states in Q and Q' since we are running the DFAs in parallel. In addition, we need to add a new start state, which will be explained later.

$$Let\, Q'' = Q \times Q' \cup \{{q_0}''\} \tag{35}$$

The alphabet stays the same.

$$\Sigma = \Sigma \tag{36}$$

The transition function for the NFA is quite complicated. First, if the machine is at the start state, the next state is in both the start of x and the end of y, which are the starting states for M and M' respectively. If the machine is anywhere else, then advance to the next state in their respective DFAs if given an input.

$$\delta''(\{{q_0}''\}, \epsilon) = (q_0, {q_0}') \tag{37}$$

$$\delta''(\{\delta(x \in Q, a), \delta'(y \in Q', a)\}, b) = (x' \in Q, y' \in Q') \tag{38}$$

The start state of N is fabricated to have ϵ transitions to both of the DFAs' start states, so they can run in parallel. The ϵ transitions connect to (q_0, q_0'). Because there are two DFAs, both of these must reach a final state. Thus, both states must be in F (because F = F').

$$F'' = (x \in F, x \in F) \tag{39}$$

I don't know how to justify that my construction is correct. □