Student ID: 304990072

Collaborators: ____________

CS181 Winter 2019 – Problem Set 1

Due Tuesday, January 29, 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.overleaf.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 Gradescope.

> 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'.

Problem 1

Assume L to be a regular language, $\exists M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ s.t. $L(M_1) = L$, and M_1 is a DFA. Define another NFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ that accepts L^R in the following way:

- reverse all the arrows in M_1
- Add a new start state called q_{new} which has arrows pointing to every end state of every accept states of M_1. All the arrows are labelled with ϵ.
- Change the final states and the start state of M_1 into normal states.

More formally, define M_2 as follows:
Let $Q_2 = Q_1 \cup \{q_{new}\}$, $q_2 = q_{new}$, $F_2 = q_1$, q_{new} is as defined above. δ_2 is defind by if $p \in \delta_1(q, a)$, then $q \in \delta_1(p, a)$

Then $L(M_2) = L^R$
$L^R \subseteq L(M_2)$: Consider $x \in L^R$. Then, by definition of L^R, $x^R \in L$. Thus, on input x_R, M_1 goes from the starting state q_1 to the a state in F_1. Since M_2 is defined by reversing all transitions in M_1, and adding a new state q_{new} with ϵ-transitions to states in F_1, an input x to M_2 will be accepted with a reversed sequence of transitions of states (with an additional start state of q_{new}). Hence, $L^R \subseteq L(M_2)$.

$L(M_2) \subseteq L^R$: Consider $y \in L(M_2)$. Then, on input y, M_2 goes from state q_{new} to a state $q' \in F_2$. Since q_{new} is not in F_2,then M_2 must first transition out of q_{new}. But, the only transitions from q_{new} are the ϵ-transitions to states in F_1. From this point onwards, every transition of states M_2 takes have a corresponding transition in M_1, but in a different direction. Also, M_2 has only 1 final state, which is the start state of M_1. Hence, for every $y \in L(M_2)$, $y^R \in M_1$. $\Rightarrow y \in L^R$ by definition. Hence, $L(M_2) \subseteq L^R$.

Therefore, $L(M_2) = L^R$, L^R is a regular language. (proven)

Problem 2

Assume L to be a regular language, $\exists M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ s.t. $L(M_1) = L$, and M_1 is a DFA. Define another NFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ that accepts $L_{\frac{1}{2}-}$ in the following way:

The idea is, if a string of length l is accpeted by M_2, then at least a string of length $2l$ can be accepted by M_1. Since the length l is a variable, we need keep track of both the current state and if it is possible to reach a state $\in F_1$ in exactly l steps. More formally:

Let $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be an NFA such that

- $Q_2 = (Q_1 \times Q_1) \cup \{q_2\}$, the first entry denotes the current state in M_1 if the string is an input for M_1, the second entry denotes the states that are possible to reach a state $\in F_1$ in l steps.
- q_2 is the start state we define for M_2
- F_2 are states with format $(q, q | q \in Q_1)$ according to our definition of Q_2. This is because if the string is used as an input for M_1, we can reach a final state $\in F_1$ in l steps (l is the current length of string).
- $\delta_2(q, a) = \begin{cases} (q_1, f) \textbf{ for } f \in F_1 & \textbf{if } q = q_2,\ a = \epsilon \\ (\delta_1(q, a), m) & \textbf{if } q = (x, y)\textbf{,}\ x, y \in (Q_1 \times Q_1),\ a \neq \epsilon, y = \delta_1(m, a) \end{cases}$

For $q \in Q_2, a \in \Sigma \cup \{\epsilon\}$.
We define δ_2 in such a way because each time we get a new input, l increases by 1. Thus we need to change the second entry to a state in Q_1 that is 1 more step further from the final states of M_1. The only exception is when M_2 is in the start state, we need to indicate that an empty string is given to M_1, M_1 is at its start state q_1, and all the final states of M_1 are 0 steps away from themselves (due to a string length l of 0).

$L_{\frac{1}{2}-} \in L(M_2)$: according to out construction of M_2, all string $x \in L_{\frac{1}{2}-}$ are accepted by M_2

$L(M_2) \in L_{\frac{1}{2}-}$: all the final states of M_2 has the form $(q, q | q \in Q_1)$ according to our definition. This means if the same string is used as input for M_1, the M_1 will end in a state which can reach one of its final states in l steps, l is the length of the input string. Hence, according to the definition of $L_{\frac{1}{2}-}$, $L(M_2) \in L_{\frac{1}{2}-}$.

Therefore, $L(M_2) = L_{\frac{1}{2}-}$, $L_{\frac{1}{2}-}$ is a regular language. (proven)

Problem 3

Assume there exists a DFA $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1),\ \Sigma_1 = \Sigma \cup \{\$\}$ such that it accepts a string in the form of $x\$x$. The object is to prove $L(M_{2p}) = \{x \in \Sigma^* \mid M_1 \ accepts \ x\$x\}$ is a regular language. To do this, define NFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ by modifying M_1. We can use the same set of states as M_1, and add a new start state q_2 with transitions labelled with ϵ to every state q that accepts a transition labelled with \$. Then, delete all transitions in the original M_1 labelled with \$. more formally:

- $Q_2 = Q_1 \cup \{q_2\}$
- q_2 is the new start state we defined
- $F_2 = F_1$
- $\Sigma_3 = \Sigma \cup \{\epsilon\}$
- For $q \in Q_2, a \ \in \Sigma,$

$$\delta_2(q, a) = \begin{cases} \delta_1(q_1, a) & \textbf{if } q \neq q_2,\ a \in \Sigma \\ \{q \mid q = \delta_1(q^*, \$),\ q^* \in Q\}, & \textbf{if } q = q_2,\ a = \epsilon \end{cases}$$

Since every NFA has an equivalent DFA, we can construct a DFA M_3 equivalent to M_2, with $\Sigma_3 = \Sigma$.

$L(M_{2p}) \subseteq L(M_2) \Rightarrow L(M_{2p}) \subseteq L(M_3)$: For any input in the form of $x\$x$ (accepts by M_{2p}/M_1), M_1 will be in a state that accepts a transition labelled with \$ after reading the \$ symbol in $x\$x$. If the same x is used as an input to M_2, since the only transitions q_2 has are those $\epsilon - transitions$ to $\{q \mid q = \delta_1(q^*, \$),\ q^* \in Q\}$, M_2 can be in the same state when M_1 reads \$. Since M_1 accepts $x\$x$, M_1 will continue from that state until it reaches an accept state after reading $x\$x$. The same set of transitions will take place in M_2 since $\delta_2(q, a)$ is defined as $\delta_1(q_1, a)$ for $q \neq q_2$, and M_2 will never go back to the start state q_2 after the $\epsilon = transitions$ as defined. Thus for any x that is accepted by M_{2p}, it is accepted by M_2. Hence, $L(M_{2p}) \subseteq L(M_2)$. $\Rightarrow L(M_{2p}) \subseteq L(M_3)$

$L(M_2) \subseteq L(M_{2p}) \Rightarrow L(M_3) \subseteq L(M_{2p})$: Assume not. For any string x accepted by M_2, according to the definition of δ_2 (same as δ_1 except when $q = q_0$), there must be a string in the form of $y\$x$ that is accepted by M_1. Thus, if $L(M_2) \nsubseteq L(M_{2p})$, $y \neq x$. Since M_1 only accepts strings in the form of $x\$x$, M_1 does not accept the string $y\$x$ if $y \neq x \Rightarrow$ contradiction. Hence, $L(M_2) \subseteq L(M_{2p}) \Rightarrow$ $L(M_3) \subseteq L(M_{2p})$.

Hence, $L(M_3) = L(M_{2p})$, $L(M_{2p})$ is a regular language. (proven)