Do problems 0 and 1 and any two of 2, 3, or 4. Graded on a scale of 100 points.

0. (**5 points**) Your name: Mark Greenstreet Your student #: 00000000

1. (**35 points**) (Sipser exercise 1.47)
Let $\Sigma = \{\texttt{1}, \texttt{\#}\}$ and let

$$A \quad = \quad \{w \mid w = x_1\texttt{\#}x_2\texttt{\#}\cdots\texttt{\#}x_k,\ k \geq 0, \text{ each } x_i \in \texttt{1}^* \text{ and } (i \neq j) \Rightarrow (x_i \neq x_j)\}$$

In English, A is the set of all strings consisting of zero or more strings of `1`'s separated by `#`'s such that no two of these strings of `1`'s have the same length. For example `1`, `1#11#111`, `1111##11#1111111` and `111#1111#11111#111111` are in A, but `1#1` and `1#11#111#11` are not.

Prove that A is not regular.

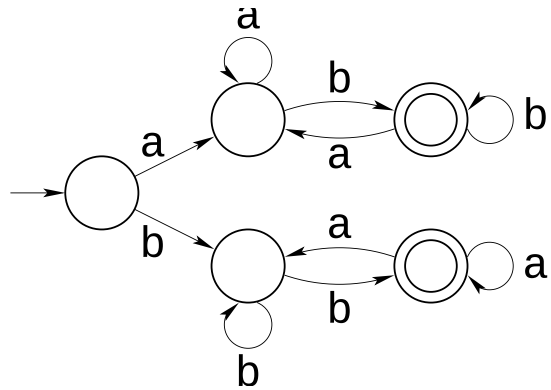
Solution:

(a) Let p be a proposed pumping lemma constant for A.
(b) Let $u = \texttt{1}^p\texttt{\#}\texttt{1}^{p+1}\texttt{\#}\cdots\texttt{\#}\texttt{1}^{2p}$.
Note that we can write $u = u_0\texttt{\#}u_1\texttt{\#}\cdots\texttt{\#}u_k$, where $k = p$ and $u_i = \texttt{1}^{p+i}$.
(c) Let $xyz = u$ such that $|y| > 0$ and $|xy| \leq p$.
(d) Let $v = xy^2z$.
Note that we can write $v = v_0\texttt{\#}v_1\texttt{\#}\cdots\texttt{\#}v_k$, where $k = p$, $v_0 = \texttt{1}^{p+|y|}$ and for $1 \leq i \leq p$, $v_i = \texttt{1}^{p+i}$.
Because $1 \leq |y| \leq p$, we conclude $p + 1 \ \leq \ (p + |y|) \ \leq \ 2p$ and $v_0 = v_{p+|y|}$. Thus, $v \notin A$.
(e) A does not satisfy the conditions of the pumping lemma. Therefore, A is not regular.

2. (**30 points**)

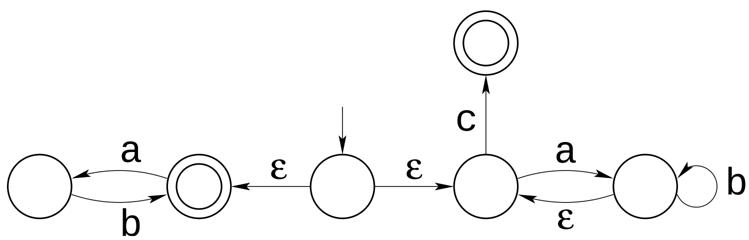
(a) (**10 points**) Give a DFA that recognizes the language $\mathtt{a}(\mathtt{a} \cup \mathtt{b})^*\mathtt{b} \cup \mathtt{b}(\mathtt{b} \cup \mathtt{a})^*\mathtt{a}$.
The input alphabet is $\{\mathtt{a}, \mathtt{b}\}$. Drawing a state diagram for your DFA is sufficient.

Solution:

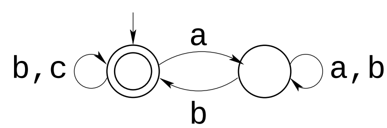


(b) (**10 points**) Give a NFA that recognizes the language $(\mathtt{ab}^*)^*\mathtt{c} \cup (\mathtt{ab})^*$.
The input alphabet is $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Drawing a state diagram for your NFA is sufficient.

Solution:



(c) (**10 points**) Give a regular expression corresponding to the NFA:



Solution: $(\mathtt{a}^*\mathtt{b} \cup \mathtt{c})^*$

3. (**35 points**) Let B be any language. Define

$$f(B) \quad = \quad \{w \mid \exists x \in B.\, x = ww^{\mathcal{R}}\}$$

where $x^{\mathcal{R}}$ denotes the reverse of string x. For example,

$$f(\{\texttt{cattac}, \texttt{doggod}, \texttt{mouseesoum}\}) \quad = \quad \{\texttt{cat}, \texttt{dog}, \texttt{mouse}\}$$

Show that if B is any regular language, then $f(B)$ is regular as well. It is sufficient to describe the construction of a DFA, NFA or regular expression for $f(B)$ and/or use closure properties that we have already proven. You don't need to give a formal proof that your construction is correct.

Solution: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that recognizes B. My solution builds an NFA, N, that runs M backwards starting from a state in F. The construction of N is pretty much the same as the one used in $HW2$ to show that the regular languages are closed under string reversal. Let's say that M reaches state q after reading w. If N can reach state q by reading w, then that means that M will reach a state in F by reading $w^{\mathcal{R}}$. This means that M accepts $ww^{\mathcal{R}}$. In fact, these are the only strings that M can accept.

The preceeding paragraph is an acceptable answer to the question. I'll also include the details of the construction of N, but won't require them in a solution (as long as the solution points out the connection with the previously solved problem from the homework).

$$\begin{array}{rcll} N & = & (Q \cup \{q_x\}, \Sigma, \delta^{\mathcal{R}}, q_x, X) & \\ q_x & \notin & Q & \\ \delta^{\mathcal{R}}(q, c) & = & \{p \in Q \mid \delta(p, c) = q\}, & \text{for } q \in Q \\ \delta^{\mathcal{R}}(q_x, \epsilon) & = & F & \end{array}$$

and X doesn't matter, because we're just going to combine N with M to create the machine that recognizes $f(B)$. Here it is:

$$\begin{array}{rcl} N' & = & (Q \times (Q \cup \{q_x\}), \Sigma, \delta', (q_0, q_x), F') \\ \delta'((p, q), c) & = & \{\delta(p, c)\} \times \delta^{\mathcal{R}}(q, c) \\ F' & = & \{(q, q) \in Q \times Q\} \end{array}$$

4. (**35 points**) Ever had a broken keyboard that dropped or repeated characters? If so, this problem is for you. Let Σ be a finite alphabet, and let $RE(\Sigma)$ denote all regular expressions over strings in Σ^*.
Define $\mathit{flakeyKeys} : \Sigma^* \rightarrow RE(\Sigma *)$ as shown below

$$\begin{array}{rcl} \mathit{flakeyKeys}(\epsilon) & = & \epsilon \\ \mathit{flakeyKeys}(x \cdot c) & = & x \circ c^*, \quad \text{for any } c \in \Sigma \end{array}$$

In other words, $\mathit{flakeyKeys}(x)$ maps the string x to a regular expression that matches any string that can be derived from x by dropping or repeating symbols. For example, $\mathit{flakeyKeys}(\texttt{cat})$ is the regular expression $\texttt{c}^*\texttt{a}^*\texttt{t}^*$

Let C be any language. Define

$$\mathit{flakeyKeys}(C) \quad = \quad \{w \mid \exists x \in C.\ w \in \mathit{flakeyKeys}(x)\}$$

Show that if C is regular, then $\mathit{flakeyKeys}(C)$ is regular as well. It is sufficient to describe the construction of a DFA, NFA or regular expression for $\mathit{flakeyKeys}(C)$ and/or use closure properties that we have already proven. You don't need to give a formal proof that your construction is correct.

Solution 1: The key idea in my solution is to construct a GNFA (see Sipser p. 70ff, esp. def. 1.64) that recognizes $\mathit{flakeyKeys}(C)$.

Let $M = (Q, \Sigma, \delta, q_a, F)$ be a DFA that recognizes C. Let $Q' = Q \cup \{q_s, q_a\}$ where q_s and q_a (i.e. "start" and "accept") are not in Q. Let

$$\begin{array}{rcll} G & = & (Q', \Sigma, \delta', q_s, \{q_a\}, & \text{a GNFA} \\ \delta'(q_a, q_0) & = & \epsilon & \\ \delta'(q_a, q) & = & \emptyset, & q \neq q_0 \\ \delta'(p, q) & = & c_1^* \cup c_2^* \cup \cdots \cup c_k^*, & (c \in \{c_1, c_2, \ldots c_k\} \Leftrightarrow \delta(p, c) = q,\ p, q \in Q \\ \delta'(q, q_a) & = & \epsilon, & \text{if } q_a \in F \\ \delta'(q, q_a) & = & \emptyset, & \text{if } q_a \notin F \\ \delta'(q_a, q) & = & \emptyset, & q \in Q' \end{array}$$

By construction, $L(G) = \mathit{flakeykeys}(C)$, and $L(G)$ is regular because GNFAs recognize the regular languages. Thus, $\mathit{flakeyKeys}(C)$ is regular.

Solution 2: One might object that I said you would never need to know the details of the proof that every DFA can be converted into a regular expression. If so, here's an alternative solution.

Let $M = (Q, \Sigma, \delta, q_a, F)$ be a DFA that recognizes C. For each state $q_i \in Q$ and each symbol $c \in \Sigma$ such that M has an outgoing arc from q labeled c, define a new state, $q_{i,c}$. Add an ϵ arc from q_i to $q_{i,c}$ and another ϵ arch fro $q_{i,c}$ to $\delta(q_i, c)$. Finally, add a self-loop arc from $q_{i,c}$ to $q_{i,c}$ labelled c. This produces an NFA that recognizes $\mathit{flakeyKeys}(C)$.