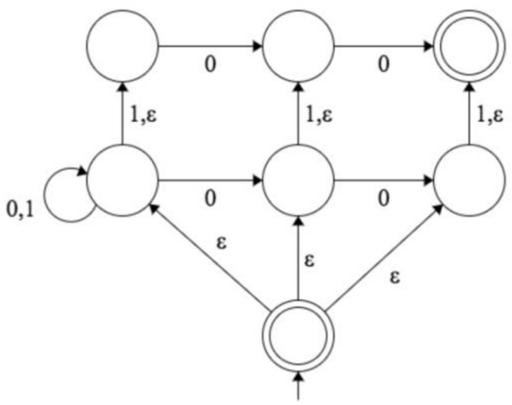
Exercise 4

Submit by Wednesday 21/04/21

Question 1 (20 pts)

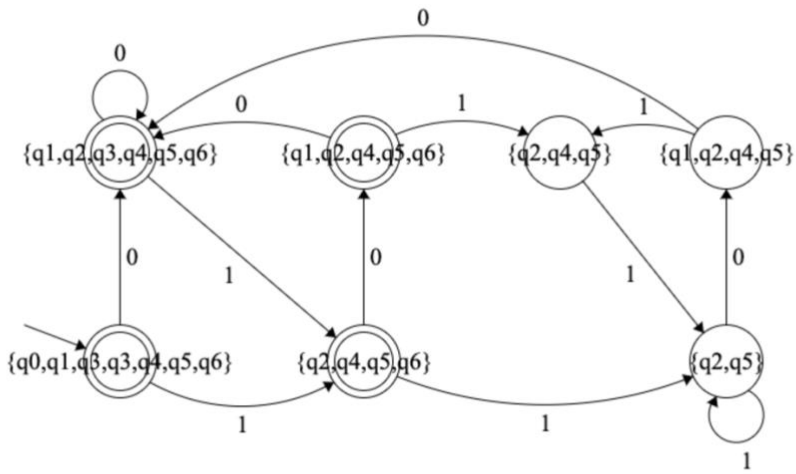
Formally describe the languages of the following NFAs and draw an equivalent DFAs (a state diagram):

a.

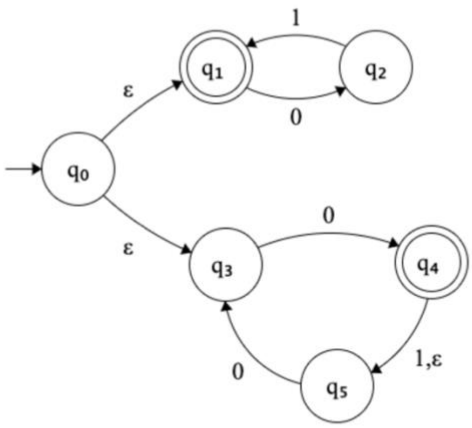


From bottom center going clockwise: q0, q1, q6, q5,q4,q3 and in the center q2

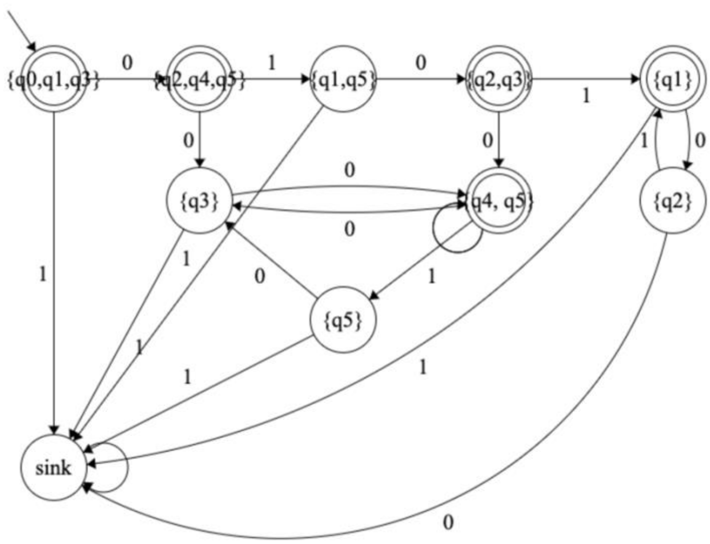
$$L_a = \{w = uv \in \{0,1\}^* \mid |v| \leq 3, \#_1(v) \leq 1\}$$



b.



$$L_b = \{w \in \{01\}^*\} \cup \{w \in \{0{,}010\}^* | \#_0(w)\%2 = 1\}$$



Question 2 (20 pts)

Prove (by regular closures) or disprove (by a counter example) the following claims.
You can assist the language $L = \{a^n b^n \mid n \geq 0\}$ which is known to be non-regular.

a. If R is a regular and R-L is regular, then $L \cap R$ is regular.

If R is regular, and R \ L is regular, then $R \cap L$ is regular. We can let M_1, M_2 be the automatons which recognize R and R\L respectively. $M_1 = \left(Q_1, \Sigma, \delta_{M_1}, q_1, F_1\right)$, $M_2 = (P_1, \Sigma, \delta_{M_2}, p_1, F_2)$. We construct our product automaton of M_1, M_2 as we learned in class with accepting states F defined as: $F = \{(q_i, p_j) | q_i \in F_1, p_j \in (P_2 \backslash F_2)\}$, which is to say we accept states in R but not in R\L, so we accept states in the intersection of R and L.

b. If R is a regular and $R \cup L$ is regular, then L is regular.

False, counterexample:

Let $R = \Sigma^*,\ L = a^n b^n,$ then $R \cup L = \Sigma^*$, but we know that L is not regular, so contradiction.

c. If R is a regular and $R \cap L$ is regular, then L is regular.

False, counterexample:

Let $R = \emptyset,\ L = a^n b^n$, then $R \cap L = \emptyset$, but we know that the L we defined is not regular, contradiction ☺

d. If $R \cup L$ is a regular and L is finite, then R is regular.

We know that if L is finite then it is also regular, therefore we continue, as $L, R \cup L$ are regular, we know that L\R is also regular, because removing something from a finite language L means the remainder is still finite. Now we can see if we have $(R \cup L) \backslash (L \backslash R)$ which is the set difference of two regular languages that it therefore is regular. So, we have $(R \cup L) \backslash (L \backslash R) = R$ and therefore R is regular.

Question 3 (60 pts)

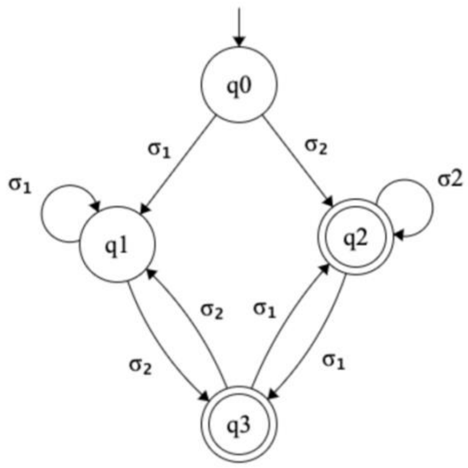
Prove by construction and/or regular closures that the following languages are regular. In your answer **define formally** the automaton recognizing the language and describe the idea behind it. If you prove by construction, **draw also a diagram** describing visually your construction. No need to prove your construction. Follow the notations shown in class. **Points will be reduced for awkward or imprecise notations.**

a. $L^R = \{w^R |\ w \in L\}$ where L is regular

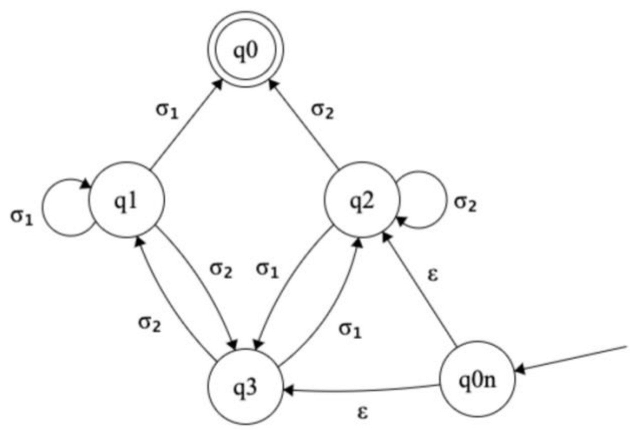
Given that L is regular, that means that there exists a DFA for it. Let $A = \{Q, \Sigma, \delta, q_0, F\}$. We build our NFA $A' : A' = \{Q', \Sigma',\ \delta', q_0', F'\}$ for the language L^R where $Q' = Q \cup \{q_0^n\}$, $\Sigma' = \Sigma$, $q_0' = q_0^n, F' = q_0$, and we define δ' as $\delta' : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$. Then we have $\forall \sigma \in \Sigma,\ \forall q \in$

$Q, \delta'(q, \varepsilon) = \delta'(q_0^n, \sigma) = \emptyset, \delta'(q, \sigma) = \{p | \delta(p, \sigma) = q\}, \delta'(q_0^n, \varepsilon) = \{q | q \in F\}$. A' enables us to simulate reading words in reverse from q_0^n, and we use ε transitions to traverse so we don't change the word. We get from an accepting state of A to the starting state of A. We can use an NFA to define the language, and since L^R is a regular language the NFA is equivalent to the DFA, so we reverse the language, and go back on the edges of the original A, and flip the direction of all the original edges.

For A:



For A':



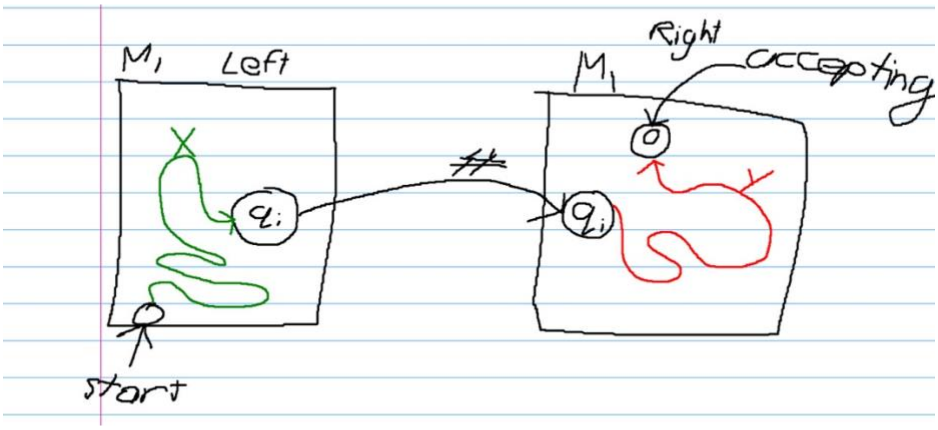
b. $L' = \{w \mid ww^R \in L\}$ where L is regular.

L is a regular language, so it has a DFA 'A' such that $L(A) = L$, and $A = \{Q, \Sigma, \delta, q_0, F\}$. We know from the lecture that if the language is regular then the language $L_{pq} = \{w \mid \delta(p, w) = q\}, \forall (p, q) \in$

$Q \times Q$ is regular as well. From the previous section, we know that the reverse language is a regular language too so L_{pq}^R is a RL by the closure under reverse. For each pair $(p, f) \in Q \times F$ we can define L_{pf} as $L_{pf} = L_{q_0 p} \cap L_{pf}^R$ and we know that L_{pf} is a regular language by closure of intersection. $L_{pf} = \{w \mid ww^R \in L, \delta(q_0, w) = p, \delta(p, w^R) = f \in F\}$, and for $\forall (p, f) \in Q \times F$ we can take the finite union: $L'' = \bigcup_{(p,f) \in Q \times F} L_{pf}$, so L'' is a regular language by the closure of a finite union. As we hav, $L'' = L'$, we can see L' is a regular language

c. $L' = \{x\#y \mid xy \in L,\ yx \notin L\}$ where L is regular.

We approach this question by constructing two separate regular languages, such that their product automaton recognizes exactly L'. We begin with the following language: $L_1' = \{x\#y \mid xy \in L\}$. As L is regular, there exists a DFA M_1 that accepts it. Consider the following diagram:



This diagram illustrates the broad idea of the construction of a machine recognizing L_1', it works by duplicating M_1 and forcing any path leading to an accepting state to traverse exactly one edge reading a # from the input.

We now describe the above mathematically: suppose $M_1 = \{Q, \Sigma, \delta, q_0, F\}$ we define $M_1' = \{Q', \Sigma', \delta', q_0^L, F^R\}$. $Q' = Q_L \cup Q_R$ where Q_L, Q_R are simply two copies of the state Q. $\Sigma' = \Sigma \cup \{\#\}$, $q_0^L = q_0 \in Q_L, F^R = F \subseteq Q_R$. δ' of any two states both appearing in the same "half" of the machine is exactly the same as δ. We add the following transitions to δ': $\delta'(q_i^L, \#) = q_i^R$, that is each state in the left half is connected to the corresponding state in the right half by a # transition.

$xy \in L \Rightarrow x\#y \in L(M_1')$

$xy \in L \Rightarrow$ there exists a path from q_0 to $q_f \in F$ which the machine follows upon reading xy.

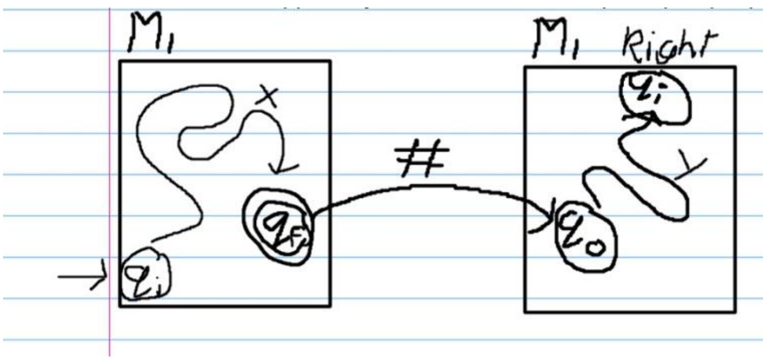
Notice, that by construction of M_1' we can decompose this path into two halves, one completely contained in the left half of the machine (denoted by $q_0^L \dots q_i^L$) and the other completely contained in the right half (denote by $q_i^R \dots q_f^R$). Observe that there exists a transition $\delta'(q_i^L, \#) = q_i^R$ therefore, since there is a # in the string x#y at position i, we see that there exists an accepting path in M_1', for the string x#y, ie, $x\#y \in L(M_1')$.

If $xy \notin L \Rightarrow x\#y \notin L(M_1')$

By near identical arguments used in the previous case the above holds. ☺ Effectively if an accepting path did exist in our constructed machine it would imply the existence of an accepting path for xy in M_1.

We now construct L_2' which recognizes the language $\{x\#y | yx \notin L\}$.

The broad idea is to create |Q| copies of M_1, each of which will be attached to |F| other copies of M_1, for a total of $|Q| * |F|$ copies of M_1. If $xy \in L$, then there exists an accepting path $q_0 \dots . q_i \dots q_f$ where q_i is the state of M_1 once it has read x. As we cannot know ahead of time the value of q_i, we use ε transitions to |Q| copies of M_1 to simulate a guess.



The above diagram ☺ is beautiful ☺ and illustrates one “gadget” of the machine.

$M_2' = \{Q', \Sigma', \delta', q_0', F'\}$. $Q' = \{q_0'\} \cup \bigcup_{i=1}^{|Q|\times|F|} Q$, $\Sigma' = \Sigma \cup \{\#\}, F'$ will be described during construction. Denote by “left machine” the machine following the initial epsilon transitions, denote by “right machine” the machines following a hashtag transition. $\delta'(q_0', \varepsilon) = q_i$ for each of the |Q| possible q_is, each qi occurring in a left machine. Within a left machine $\delta' = \delta$ with the addition that for each $q_f \in F$, $\delta'(q_f, \#) = q_0$ where q_0 occurs in an otherwise unused right machine. The accepting states of the right machine are $Q \backslash q_i$.

Suppose $yx \in L$. We will show that $x\#y \notin L'$.

If $yx \in L$, then there exists a path $q_0 \to q_i \to q_f$ where q_i is the state of the machine once it reads y. by construction of M_2' we know that x#y will follow a path ending in q_i which by construction will not be an accepting state.

Suppose $x\#y \notin L'$, then we show that $yx \in L$.

By near identical arguments used in the previous section, the above holds ☺