Homework 3

Question 1

Part a:

Set NFA $N_1 = (\Sigma, Q_1, s_1, A_1, \delta_1), N_2 = (\Sigma, Q_2, s_2, A_2, \delta_2)$. N_1 accepts all string that contains odd number of 0s, N_2 accept all string that contains substring 101.

Then we can construct an NFA $N = (\Sigma, Q, s, A, \delta)$ based on N_1, N_2:

$$N = \begin{cases} \Sigma \\ Q = Q_1 \times Q_2 \\ s = (s_1, s_2) \\ A = \{(x, y) \mid x \in A_1 \text{ or } y \in (Q_2 - A_2)\} \\ \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)) \end{cases}$$

For arbitrary state $(q_i, q_j) \in Q$ returned by N with input w, $\;q_i, q_j$ equals to the state returned by N_1, N_2 with input w separately. Since we set $A = \{(x, y) \mid x \in A_1 \text{ or } y \in (Q_2 - A_2)\}$, then the state returned by N will be in A if the input is at least get accepted by N_1 or rejected by N_2, which means the input have an odd number of 0s or do not contain the substring 101

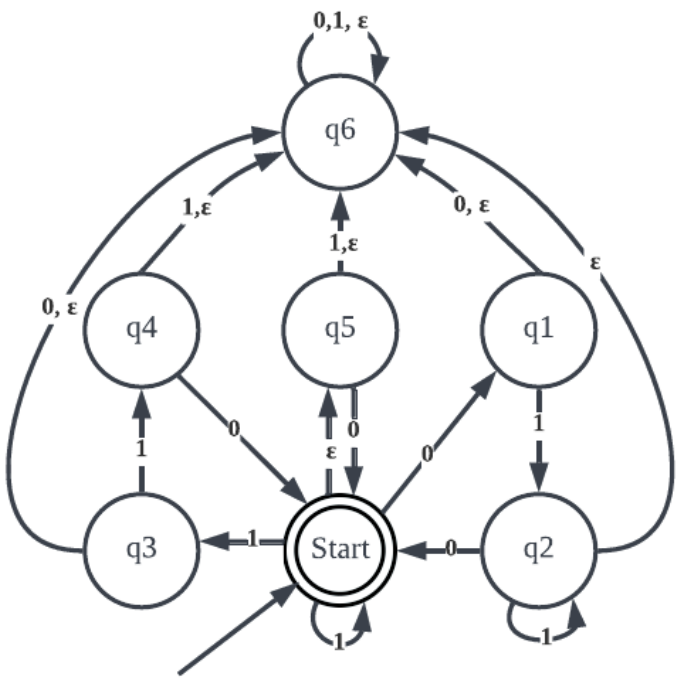
Part b:

Set NFA $N_{2\vee 3}=(\Sigma, Q, s, A, \delta)$:

$$N_{2\vee 3}=\begin{cases}\Sigma=\{0,1\}\\ Q=\{(x,y,z)\mid x,y,z\in\mathbb{Z}^+\}\\ s=(0,0,0)\\ A=\{(x,y,z)\mid (x,y,z)\in Q, y=0 \text{ or } z=0\}\\ \delta((x,y,z),a)=\begin{cases}(2x+a,(2x+a)\bmod 2,(2x+a)\bmod 3) & \text{,if } a\in\Sigma\\ (x,y) & \text{, if } a=\epsilon\end{cases}\end{cases}$$

For arbitrary state $(q_i, q_j, q_k)\in Q$ returned by $N_{2\vee 3}$ with input w, q_i represent the decimal value of w, $q_j=q_i \bmod 2$, $q_k=q_i \bmod 3$. if the value of w is divisible by 2 or 3, then at least one of q_j and q_k should be 0. Thus all state with $q_j=0$ or $q_k=0$ are the accept state. If the input is ϵ, $\delta(s,\epsilon)=s=(0,0,0)\in A$, since 0 is divisible by any whole number.

Part c:



We construct the NFA shown above that accept the regular expression. The start state can accept 1^*, and go to state $q1, q3, q5$ with input $0, 1, \epsilon$. State $q1, q2$ and $q3, q4$ are used to detect 01^* and 11 separately. Once $q2, q4$ receive 0, the string contains (01^*0) or (110). State $q5$ is used to detect $(\epsilon 0)$ pattern in the input. State $q6$ is "unaccepted" state, any un-expected input that broke the pattern given by regular expression will lead to $q6$

Part d:

Set $NFA = (\Sigma, Q, s, A, \delta)$

$$\begin{cases} \Sigma = \{0,1\} \\ Q = \{(x,y,z) \mid x,y \in \mathbb{Z}^+ \cup \{0\}, z \in \{0,1\}\} \\ A = \{(x,y,z) \mid (x,y,z) \in Q, x = y\} \\ s = (0,0,0) \\ \delta((x,y,z),b) = \begin{cases} \{(x+1,y,0),(x,y+1,1),(x,y,z)\} & \text{if } b = 0 \text{ and } z = 0 \\ \{(x,y+1,1),(x,y,1)\} & \text{if } b = 0 \text{ and } z = 1 \\ \{(m,0,1) \mid \forall m \in \mathbb{Z} \cap [0, x+y]\} & \text{if } b = 1 \text{ and } z = 0 \\ \{(m,0,1) \mid \forall m \in \mathbb{Z} \cap [0, x]\} & \text{if } b = 1 \text{ and } z = 1 \end{cases} \end{cases}$$

For arbitrary state $(x,y,z) \in Q$, x is the number of 0 we assume as the prefix of the string, y is the number of 0 we assume as the suffix of the string. So such a state assume that we can represent the input as $0^x(0+1)^*0^y$. if current state assume it has not met $(0+1)^*$ part (or have not get 1 as input), then $z = 0$, otherwise $z = 1$. A state will be accepting state if $x = y$, which means the given string can be written as $0^x(0+1)^*0^y$ and $x = y = a$ (equals to a if a is a constant)

In the transition function, if we have not process $(0+1)^*$ in the input, we assume each 0 can belongs to either prefix or suffix, or 0 are included in $(0+1)^*$. Thus the resulting state will be $(x+1,y,0),(x,y+1,1),(x,y,z)$.

If we received 1 or assumed that the past input was included in $(0+1)^*$, then we can only count 0 to y, or we can assume 0 is also in $(0+1)^*$

if we received 1 for the first time ($z = 0$), then we should set x be a set of value in range $[0, x+y]$ and zero out y because we should calculate y after $(0+1)^*$ pattern, and all suffix length calculated before $(0+1)^*$ can be counted as the length of prefix.

if we keep receiving 1, we will keep y as zero, and re-distribute x value

Question 2

Part(a)

Consider a NFA accept double(L) $M' = (Q', s', A', \delta')$, where it includes a special state s'' with $\varepsilon-$transitions to every start states of M. and M' includes a new accepting state a'', and the origional state read an input 1 to reach a''.

- $Q' = Q \cup \{s''\} \cup \{a''\}$
- $S' = \{\delta(s,a) \mid a \in \Sigma\}$
- $A' = \{a''\}$
- $\delta'\left(s'', \varepsilon\right) = \{\delta(s,a) \mid a \in \Sigma\}$
- $\delta'\left(s'', a\right) = \varnothing$, when $\forall a \in \Sigma$
- $\delta'(p, \varepsilon) = \varnothing$, when $\forall p \in Q$
- $\delta'(p, a) = \delta(\delta(p,a), a)$, when $\forall p \in Q \wedge a \in \Sigma$
- $\delta'(a, 0) = a''$, $\forall a \in A$
- $\delta'(a, \varepsilon) = \varnothing, \forall a \in A$
- $\delta'(a, 1) = \varnothing, \forall a \in A$

Since the NFA we construct M' accepts double(L), then we can conclude double(L) is regular.

Part(b)

Consider a NFA accepts L_b, $M' = (Q', s', A', \delta')$ and M' includes a special state s'' with $\varepsilon-$ transitions to every double of the form (s, s).

- $Q' = \{Q \times Q\} \cup \{s''\}$
- $s' = (s, s)$
- $A' = A \times A$
- $\delta' \left(s'', \varepsilon\right) = (s, s)$
- $\delta' \left(s'', a\right) = \varnothing, \forall a \in \Sigma$
- $\delta'((p, q), \varepsilon) = \varnothing, \forall p, q \in L$
- $\delta'((p, q), 1) = (\delta(p, 1), \delta(q, 0)) \quad \forall p, q \in L$
- $\delta'((p, q), 1) = (\delta(p, 0), \delta(q, 1)), \forall p, q \in L$
- $\delta'((p, q), 0) = (\delta(p, 1), \delta(q, 1)), \forall p, q \in L$
- $\delta'((p, q), 0) = (\delta(p, 0), \delta(q, 0)), \forall p, q \in L$

Since the NFA M' accepts L_b, then we can conlcude L_b is regular.

Part(c)

Consider a NFA accepts L_c, $M' = (Q', s', A', \delta')$ And M' includes a special state s'' with $\varepsilon-$ transitions to every start states of M.

- $Q' = Q$
- $s' = s$
- $A' = A$
- $\delta'(q, a) = \delta(q, a) \quad \forall q \in Q, a \in \Sigma$
- $\delta'(q, 1) = \delta(\delta(q, 1), a) \quad \forall q \in Q, a \in \Sigma$
- $\delta'(q, a) = \delta(q, 1) \quad \forall q \in Q, a \in \Sigma$

Since the NFA M' accepts L_c then L_c is regular.

Question 3

Part(a,b)

3. (a) Let F be the language 0^*

Let x and y be arbitrary strings in F.

Then $x = 0^i$ and $y = 0^j$ for some non-negative integers $i \neq j$.

Let $z = \#0^i$.

Then $xz = 0^i\#0^i \in L$.

And $yz = 0^j\#0^i \notin L$, because $|0^j| \neq |0^i|$.

Thus, F is a fooling set for L.

Because F is infinite, L cannot be regular.

(b) Let F be the language 0^*

Let x and y be arbitrary strings in F.

Then $x = 0^i$ and $y = 0^j$ for some non-negative integers $i \neq j$.

Let $z = \#0^i$.

Then $yz = 0^j\#0^i \in L$.

And $xz = 0^i\#0^i \notin L$, because $|0^i| = |0^i|$.

Thus, F is a fooling set for L.

Because F is infinite, L cannot be regular.

Part c

(c) Let F be the language 0^*.

Let x and y be arbitrary strings in F.

Then $x = 0^i$ and $y = 0^j$ for some non-negative integers $i < j$.

Let $z = \#0^i$.

Then $xz = 0^i\#0^i \in L$, because a string is a substring of itself.

And $yz = 0^j\#0^i \notin L$, because $|0^j| > |0^i|$.

Thus, F is a fooling set for L.

Because F is infinite, L cannot be regular.

Part(d)