Problem 1 Reversing a regular language.

Given a regular language L, show that $L^R = \{w \mid w^R \in L\}$ is also regular.

Since L is regular, there is a regular expression that generates it. But given any regular expression R, we can find a regular expression R^R that generates the reverse of the strings generated by R. Constructing R^R can be done recursively.

$$\phi^R = \phi$$
$$\varepsilon^R = \varepsilon$$
$$a^R = a$$
$$(R_1 \cup R_2)^R = R_1^R \cup R_2^R$$
$$(R_1 R_2)^R = R_2^R R_1^R$$
$$(R^*)^R = (R^R)^*$$

Another approach is to construct an NFA that accepts L^R. This can be done by starting with an accept state and following transitions backwards until we reach the start state. Given a DFA for L $M = (Q, \Sigma, \delta, q_0, F)$, define NFA $N = (Q', \Sigma, \delta', q_0', F')$ such that

* $Q' = Q \cup \{q_0'\}$
* $\delta'(q_0', \varepsilon) = F$
* $\delta'(q, a) = \{r \mid \delta(r, a) = q\}$, $q \neq q_0'$
* $F' = \{q_0\}$

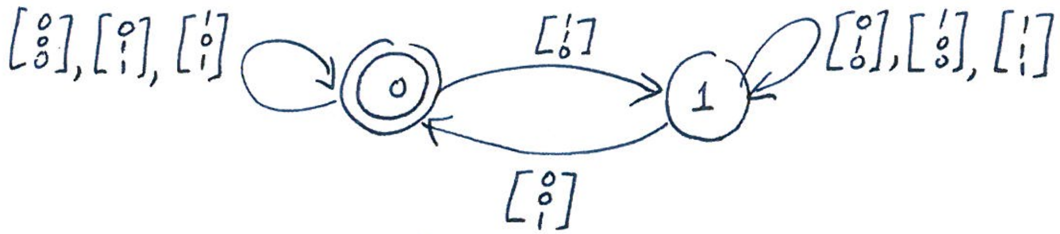
Problem 2

We construct an NFA that performs the addition from right to left, thus accepting B^R. The NFA has 2 states to represent the carry.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ takes the NFA from state } c \text{ to } c'$$

if $x + y + c = c'z$.

The state corresponding to a carry of zero is the accept state (every time the carry is 0 the addition is so far correct)



$$\delta\left(c, \begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = c' \quad \text{if } x + y + c = c'z$$

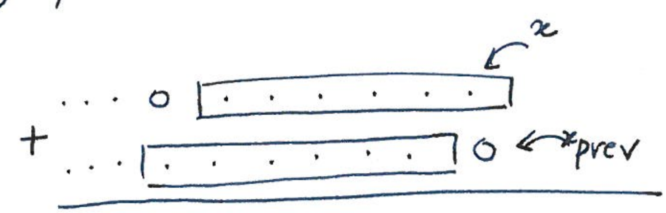
Since B^R is regular, so is B.

Problem 3

This problem is similar to Problem 2. If we add the first row to itself shifted left, the result is 3 times the first row.

The NFA keeps track of two pieces of information: the previous bit x_{prev}, and the carry c.

Initially, both are 0s.



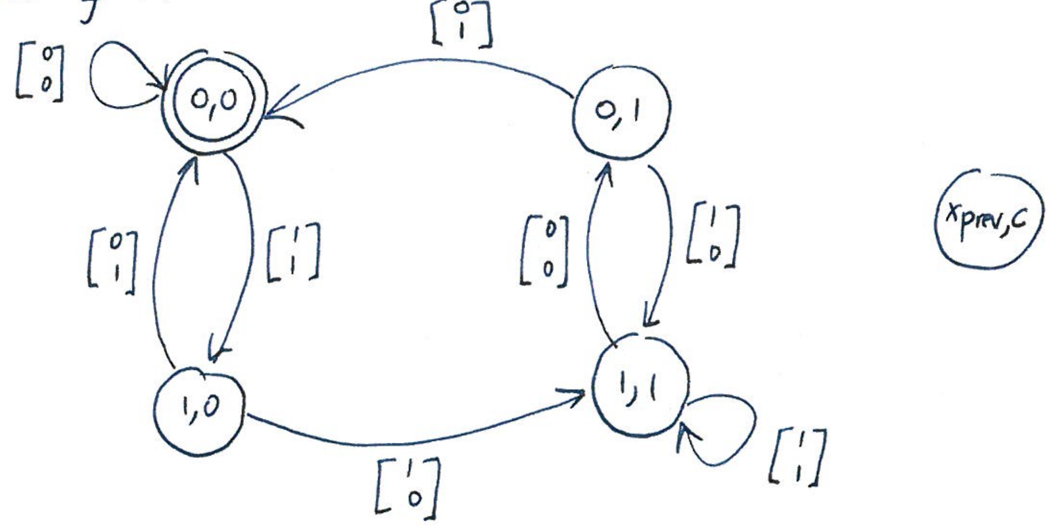
Given (x_{prev}, c),

$\begin{bmatrix} x \\ y \end{bmatrix}$ takes the NFA to (x, c')

$$\text{where} \quad \underset{\substack{\uparrow \\ \text{new } x_{prev}}}{x} + x_{prev} + c = \underset{\substack{\uparrow \\ \text{new carry}}}{c'}y$$

$$\delta\left((x_{prev}, c), \begin{bmatrix} x \\ y \end{bmatrix}\right) = (x, c') \quad \text{where} \quad x + x_{prev} + c = c'y$$

The accept states corresponds to a carry of 0 and an x_{prev} of 0.



Problem 4.

$Noprefix(A) = \{ w \in A \mid$ no proper prefix of w is in $A \}$

Since A is regular, there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts it. If every state in F is modified to have no outgoing transitions, then a string uv where $u \in A$ and $uv \in A$ cannot be accepted, because u must bring the DFA to an accept state.

We obtain NFA $N = (Q, \Sigma, \delta', q_0, F)$

where $\delta'(q, a) = \phi$ for every $q \in F$ and every a, and $\delta'(q, a) = \delta(q, a)$ otherwise.

Here's another proof:

Let $L = \{ uv \mid u \in A \text{ and } v \in \Sigma^* - \{\varepsilon\} \}$

$\Sigma^* - \{\varepsilon\}$ is regular, so L is regular being to concatenation of two regular languages. This means the complement of L is also regular.

$Noprefix(A) = A \cap \overline{L}$, which makes it also regular.

Problem 5:

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$ for D, and following the given hint, our NFA $N = (Q', \Sigma', \delta', q_0', F')$ is given by:

- $Q' = Q \times Q \cup \{q_0'\}$
- $\Sigma' = \Sigma$
- q_0' is start state
- $F' = \{(q,q) \mid q \in Q\}$
- $\delta'(q_0', \varepsilon) = q_0 \times F = \{(q_0, q) \mid q \in F\}$

  $\delta'((q_i, q_j), a) = \{(\delta(q_i, a), q) \mid \exists\, b \in \Sigma \text{ where } \delta(q, b) = q_j\}$

Problem 6.

a) If L is not regular, show $\{aw \mid w \in L\}$ is not regular. $(a \notin \Sigma)$

Assume $\{aw \mid w \in L\}$ is regular. Then there is a DFA that accepts it. Given the start state of that DFA, say q_0, the transition function δ must have $\delta(q_0, a) = q$ for some q. Now change the DFA so that q becomes the start state. This new DFA accepts L, a contradiction.

b) $L' = \{aw \mid w \in L\} \cup \{a^k \Sigma^* \mid k \neq 1\}$

Show L' is not regular.

Assume L' is regular. The language $a\Sigma^*$ is also regular. Observe that $\{aw \mid w \in L\}$ is the intersection of these two and, therefore, must be regular. This is a contradiction (part a). So L' can't be regular

c) Every $s \in L'$ can be pumped.

$L' = \{aw \mid w \in L\} \cup \Sigma^* \cup aa\Sigma^* \cup aaa\Sigma^* \cup \ldots$

if $s = aw$, then setting $y = a$ in pumping Lemma works.

if $s \in \Sigma^*$, then it's obviously pumpable

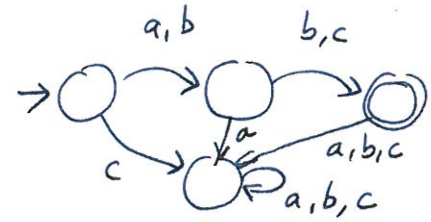
if $s = aa\Sigma^*$, then $y =$ any symbol of Σ^* part works, etc...

Problem 7

(a)

- $L = \{ab, ac, bb, bc\}$.

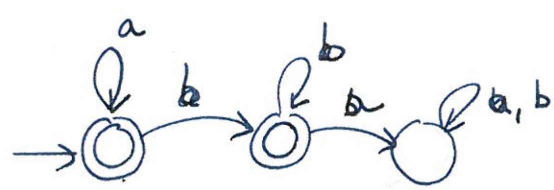
w	$\{z \mid wz \in L\}$
ε	L
c	ϕ
ab	$\{\varepsilon\}$
a	$\{b, c\}$
b	$\{b, c\}$
$\vdots$	



There are 4 classes of equivalence.

- $L = a^*b^*$

w	$\{z \mid wz \in L\}$
ε	L
$w \notin \{a,b\}^*$	ϕ
$a^i b^j$	$\{b^k \mid k \geq 0\}$
a^i	L
$\vdots$	



There are 3 classes of equivalence.

- $L = \{a^n b^n \mid n \geq 0\}$

w	$\{z \mid wz \in L\}$
ε	L
$w \notin \{a,b\}^*$	ϕ
$a^i b^i$	$\{\varepsilon\}$
$a^i b^j, j < i$	$\{b^{i-j}\}$
a^i	$\{a^k b^{i+k} \mid k \geq 0\}$
$\vdots$	

There are infinitely many equivalence classes.

L is Not regular

b) Show that if $w_1 \sim_M w_2$, then $w_1 \equiv_{L(M)} w_2$

Consider $E_{L(M)}(w_1)$

if $z \in E_{L(M)}(w_1)$, then $w_1 z \in L(M)$.

Since $w_1 \sim_M w_2$, w_1 and w_2 bring M to the same state. Therefore, $w_2 z \in L(M)$, so $z \in E_{L(M)}(w_2)$.

Similarly, if $z \in E_{L(M)}(w_2)$, then $z \in E_{L(M)}(w_1)$

so $E_{L(M)}(w_1) = E_{L(M)}(w_2)$ and $w_1 \equiv_{L(M)} w_2$.

(c) If there are more equivalence classes than states, there must be some state q of M and two strings w_1 and w_2 such that $w_1 \not\equiv_{L(M)} w_2$ but w_1 and w_2 both bring M to q so $w_1 \sim_M w_2$. A contradiction.