1. **(20 points)** Use the pumping lemma to prove that each language listed below is not regular. For each language, I state Σ the input alphabet.

   (a) $A_1 = \{w \mid \text{the number of zeros in } w \text{ is less than the number of ones}\}$. $\Sigma = \{0, 1\}$.
   For example, `1`, `011`, and `10100111` are in this language but `0` and `100` are not.
   Solution:
   Let p be a proposed pumping lemma constant, and let $w = 0^p 1^{p+1} \in A_1$. For any $xyz = w$ with $1 \leq |y| \leq |xy| \leq p$, $y \in 0^+$. Therefore, $xy^2z = 0^{p+|y|}1^{p+1}$ has at least as many 0s as 1s, and therefore is not in A_1. It follows by the pumping lemma that A_1 is not regular.

   (b) $A_2 = \mathtt{1}^{2^n}$. $\Sigma = \{\mathtt{1}\}$.
   For example, `1`, `11` and `11111111` are in this language but `111` is not.
   Solution:
   Let p be a proposed pumping lemma constant, and let $w = 1^{2^p} \in A_2$. For any $xyz = w$ with $|xy| \leq p < 2^p$, $xy^2z = 1^{2^p+|y|}$ has length that is greater than 2^p and less than 2^{p+1}, i.e. $2^p < 2^p + |y| < 2^{p+1}$, which holds since $1 \leq |y| \leq p < 2^p$. It follows by the pumping lemma that A_2 is not regular.

2. **(20 points)** Let

$$\Sigma_3 = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \cdots \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}.$$

Σ_3 contains all size 3 columns of 0s and 1s. A string of symbols in Σ_3 gives three rows of 0s and 1s. Consider each row to be a binary number with the most significant bit first. For example, let

$$w = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

The first row of w is the binary representation of 7, the second row corresponds to 5, and the third row corresponds to 12.

Let

$$B_+ = \{w \in \Sigma_3^* \mid \text{the bottom row of } w \text{ is the sum of the top two rows}\}.$$

Show that B is regular.

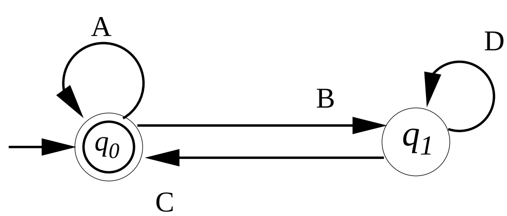

Figure 1: DFA for problem 3

Solution 1: I'll first present a NFA that recognize the reverse of B_+, $B_+^{\mathcal{R}}$. Figure 1 shows the NFA with

$$A = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right\},$$

$$B = \left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \right\},$$

$$C = \left\{ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\},$$

$$D = \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right\}.$$

This NFA basically checks the pencil-and-paper method for addition starting from the least-significant bit and working to the most signficant bit. The machine is in state q_0 when the previous bit did not generate a carry, and state q_0 when the previous bit does generate a carry. For example, the symbol $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ is in A because $0 + 0 = 0$ with no carry in or carry out. Likewise, $\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$ is in A because $1 + 1$ produces a sum of 0 and a carry to the next bit of 1 when the carry in is 0. The symbols for which a state does not have an outgoing edge correspond to errors, which the NFA rejects.

The language $B_+^{\mathcal{R}}$ is regular because the NFA presented above recognizes it. As shown on homework 2, the regular languages are closed under reversal. Thus, B_+ is regular as well.

Solution 2: This time, we construct an NFA the processes the string from left-to-right. The approach is very similar to the right-to-left DFA; the only difference is that the states of the left-to-right DFA keep track of whether or not a carry is expected from the next less significant bit. In fact, the two machines are so similar, that we can use the same transition diagram with just a slight change to the labels.

The NFA shown in Figure 1 recognizes B_+ with labels A and D as before, and exchanging the definitions of sets B and C:

$$B = \left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \right\},$$

$$C = \left\{ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\},$$

3. **(20 points)** Let Σ_3 be defined as in question 2. Let

$$B_\times = \{w \in \Sigma_3^* \mid \text{the bottom row of } w \text{ is the product of the top two rows}\}.$$

Show that B is not regular.

Solution:

Let p be a proposed pumping lemma constant, and let

$$w = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^{p-1} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^{p}.$$

This is a member of $B_\times$, because the first two rows correspond to integers 2^p, and $2^p \times 2^p = 2^{2p}$ which is an interger corresponding to the last row. For any $xyz = w$ with $|xy| \leq p$, xy^2z leaves the value of the first two rows unchanged, but changes the value of the last row, i.e. the last row represents some integer that is not 2^{2p}. Therefore $xy^2z \notin B_\times$ and $B_\times$ is not regular by the pumping lemma.

4. **(20 points)** Consider the two languages described below:

$C_1 = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid \exists x, y \in \Sigma^*.\ (w = xy) \wedge \#a(x) = \#b(y)\}$
$C_2 = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid \exists x, y \in \Sigma^*.\ (w = x \cdot \mathtt{c} \cdot y) \wedge \#a(x) = \#b(y)\}$

One of these languages is regular and the other is not. Determine which is which and give short proofs for your conclusions.
Solution:
C_1 is regular, and has the regular expression Σ^*, where Σ is the alphabet. We prove that all strings are in C_1 by induction:

Base case: $s = \epsilon$ Let $x = y = \epsilon$. Then $xy = \epsilon\epsilon = \epsilon = s$ and $\#a(x) = \#b(y) = 0$.

Induction step: $s = w \cdot c$ By the induction hypothesis, $w \in C_1$; thus we can find strings x and y with $xy = w$ and $\#a(x) = \#b(y)$. If $c \neq \mathtt{b}$, then let $y' = yc$ and let $x' = x$. Then $w \cdot c = x'y'$ and

$$\#a(x') = \#a(x) = \#b(y) = \#b(y')$$

Thus, $s = w \cdot c \in C_1$ as required.
If $c = \mathtt{b}$ and $y = \epsilon$, then $\#a(x) = \#b(y) = 0$. Choosing $x' = w \cdot c$ and $y' = \epsilon$ satisfies $w\dot{c} = x'y'$ and

$$\#a(x') = \#a(x) = 0 = \#b(y) = \#b(y')$$

Thus, $s = w \cdot c \in C_1$ as required
If $c = \mathtt{b}$ and $y \neq \epsilon$, let $y = d \cdot v$ for $d \in \Sigma$ and $v \in \Sigma^*$. If $d = \mathtt{a}$, choose $x' = x \cdot \mathtt{a}$ and $y' = v \cdot \mathtt{b}$. We have $w \cdot c = x'y'$ and

$$\#a(x') = \#a(x) + 1 = \#b(y) + 1 = \#b(y')$$

Thus, $s = w \cdot c \in C_1$ as required.
If $d = \mathtt{b}$, choose $x' = x \cdot \mathtt{b}$ and $y' = v \cdot \mathtt{b}$. We have $w \cdot \mathtt{c} = x'y'$ and

$$\#a(x') = \#a(x) = \#b(y) = \#b(y')$$

Again, $s = w \cdot \mathtt{c} \in C_1$ as required.

C_2 is not regular.
Let p be a proposed pumping lemma constant, and let $w = \mathtt{a}^p\mathtt{c}\mathtt{b}^p \in C_2$. For any $xyz = w$ with $|xy| \leq p$ and $|y| \geq 1$, $xy^0z = a^{p-|y|}\mathtt{c}\mathtt{b}^p$ has fewer a's than b's (and contains only one c), and therefore is not in C_2. It follows by the pumping lemma that C_2 is not regular.

5. (**30 points,** from Sipser, problem 2.6)
Give context free grammars generating the following languages:

(a) **(10 points)** $\{w \mid \exists n \geq 0.\ (w = \mathtt{a}^n\mathtt{b}^{2n}) \vee (\mathtt{w} = \mathtt{a}^{3n}\mathtt{b}^{n})\}$

Solution:

$$\begin{array}{rcl} S & \rightarrow & T_{2b} \mid T_{3a} \\ T_{2b} & \rightarrow & \mathtt{a}T_1\mathtt{bb} \\ T_{3a} & \rightarrow & \mathtt{aaa}T_2\mathtt{b} \end{array}$$

(b) **(10 points)** The complement of $\{w \mid \exists n \geq 0.\ w = \mathtt{a}^n\mathtt{b}^n\}$.

Solution:

$$\begin{array}{rcl} S & \to & T_a \mid T_b \mid T_{ba} \\ T_a & \to & \mathtt{a}T_0 \mid \mathtt{a}T_a \\ T_b & \to & T_0\mathtt{b} \mid T_b\mathtt{b} \\ T_0 & \to & \mathtt{a}T_0\mathtt{b} \mid \epsilon \\ T_{ba} & \to & T_x\mathtt{ba}T_x \\ T_x & \to & \mathtt{a}T_x \mid \mathtt{b}T_x \mid \epsilon \end{array}$$

Here's how it works. In the first step, the derivation "chooses" whether it will generate a string with more $\mathtt{a}$'s than $\mathtt{b}$'s (that's what T_a generates), a string with more $\mathtt{b}$'s than $\mathtt{a}$'s (that's what T_b generates), or a string that has a $\mathtt{b}$ before an $\mathtt{a}$ (that's what T_{ba} does).

Variable T_0 generates strings of the form $\mathtt{a}^n\mathtt{b}^n$. Variable T_a generates strings that have one or more $\mathtt{a}$'s preceeding a string generated by T_0; in other words, these are strings of the form a^nb^m with $n > m$. Likewise, T_b generates strings of the form a^nb^m with $n < m$. Variable T_x generates any string in Σ^* (assuming $\Sigma = \{\mathtt{a}, \mathtt{b}\}$), and T_{ba} uses T_x to generate all strings in $\Sigma^*\mathtt{ba}\Sigma^*$.

(c) **(10 points)** $\{x_1\mathtt{c}x_2\mathtt{c}\cdots x_k \mid \text{each } x_i \in \{\mathtt{a}, \mathtt{b}\}^*, \text{ and for some } i \text{ and } j,\ x_i = x_j^{\mathcal{R}}\}$.

Solution:

$$\begin{array}{rcl} S & \to & LMR \\ M & \to & \mathtt{a}M\mathtt{a} \mid \mathtt{b}M\mathtt{b} \mid \mathtt{c}L \\ L & \to & T_x\mathtt{c}L \mid \epsilon \\ R & \to & R\mathtt{c}T_x \mid \epsilon \\ T_x & \to & \mathtt{a}T_x \mid \mathtt{b}T_x \mid \epsilon \end{array}$$

A derivation creates a "left part" (L), a "middle part" (M) and a "right part" (R). The middle part generates strings of the form $x_i\mathtt{c}x_{i+1}\mathtt{c}\cdots\mathtt{c}x_i^{\mathcal{R}}$, and the left and right parts generate the rest of the string.

In particular, L generates strings of the form $((\mathtt{a} \cup \mathtt{b})^*\mathtt{c})^*$, and R generates strings of the form $(\mathtt{c}(\mathtt{a}\cup\mathtt{b})^*)^*$. Note that an equivalent regular expression for L is $(\Sigma^*c)^*$ Thus, $\mathtt{c}L$ generates $c(\Sigma^*c)^*)$ which matches the string between x_i and x_j: $\mathtt{c}x_{i+1}\mathtt{c}x_{i+1}\mathtt{c}\cdots\mathtt{c}$.

For parts (a) and (b), the alphabet is $\{\mathtt{a}, \mathtt{b}\}$. For part (c), the alphabet is $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.

6. (*20 points, Extra Credit*) Consider the language below from the September 22 lecture notes:

$$\begin{array}{rcl} \Sigma & = & \{\mathtt{a}, \mathtt{b}, \mathtt{c}\} \\ A & = & (\mathtt{aa}^*\mathtt{c})^n(\mathtt{bb}^*\mathtt{c})^n \cup \Sigma^*\mathtt{cc}\Sigma^* \end{array}$$

(a) (*10 points*) Prove that A satisfies the conditions of the pumping lemma as stated in Sipser or the September 22 notes. In other words, show that you can find a constant $p > 0$ such that for any string $w \in A$ with $|w| > p$, you can find strings x, y and z such that $w = xyz$ and $xy^iz \in A$ for any $i \geq 0$.

Solution: Let $p = 3$, and let $w \in A$ with $|w| \geq p$. We consider five cases according to the first few symbols of w:

$w = \mathtt{aa}u$ for some $u \in \Sigma^*$:

Let $x = \epsilon$, $y = \mathtt{a}$ and $z = \mathtt{a}u$. Thus, $xy^iz = \mathtt{a}^{i+1}u$. If $w \in (\mathtt{aa}^*c)^n(\mathtt{bb}^*\mathtt{c})^n$ then,

$$\begin{array}{rrcl} & u & \in & (\mathtt{a}^*c)(\mathtt{aa}^*\mathtt{c})^{n-1}(\mathtt{bb}^*\mathtt{c})^n \\ \Rightarrow & \mathtt{a}^{i+1}u & \in & (\mathtt{aa}^*c)(\mathtt{aa}^*\mathtt{c})^{n-1}(\mathtt{bb}^*\mathtt{c})^n \\ & & = & (\mathtt{aa}^*c)^n(\mathtt{bb}^*\mathtt{c})^n \\ & & \subseteq & A \end{array}$$

Likewise, if $w \in \Sigma^*\mathtt{cc}\Sigma^*$, then u and $\mathtt{a}^{i+1}u$ are as well. In all cases, $xy^iz = \mathtt{a}^{i+1}u \in A$ as required.

$w = \texttt{acaa}u$ for some $u \in \Sigma^*$:

Let $x = \texttt{ac}$, $y = \texttt{a}$ and $z = \texttt{a}u$. Then, $xy^iz \in A$ by arguments similar to those for the previous case.

$w = \texttt{acac}u$ for some $u \in \Sigma^*$:

Let $x = \texttt{ac}$, $y = \texttt{a}$ and $z = \texttt{a}c$. If $i > 0$, then $xy^iz \in A$ by arguments similar to those for the previous cases. Otherwise, $i = 0$, and

$$xy^iz \;=\; \texttt{acc}u \;\in\; \Sigma^*\texttt{cc}\Sigma^x* \;\subseteq\; A$$

as required.

$w = \texttt{acb}u$ for some $u \in \Sigma^*$:

Let $x = \texttt{ac}$, $y = \texttt{b}$ and $z = u$. Then, we can show that $xy^iz \in A$ by considering whether w was in $(\texttt{aa}^*c)^n(\texttt{bb}^*\texttt{c})^n$ or $\Sigma^*\texttt{cc}\Sigma^*$, and in the first case whether the first symbol of u is a $\texttt{b}$ or $\texttt{c}$.

w does not start with $\texttt{aa}$, $\texttt{acaa}$, `aca, or acb:` Then, $w \notin (\texttt{aa}^*c)^n(\texttt{bb}^*\texttt{c})^n$ which means that $w \in \Sigma^*\texttt{cc}\Sigma^*$. Let $x = \epsilon$, $y = \texttt{a}$, and $z = \texttt{c}u$.

$$xy^iz \;\in\; \Sigma^*\texttt{cc}\Sigma^x* \;\subseteq\; A$$

as required.

Thus, for any string w with $|w| > 3$, we can find string x, y and z that satisfy the conditions of the pumping lemma.

Prove that A is not regular.

Solution 1: Assume that A is recognized by DFA D with p states. There are $p+1$ different strings of the form $(ac)^i$, for $i = 1, 2, ..., p, p+1$. By the pigeon-hole principle, there exists r and q with $1 \leq r < q \leq p+1$ such that D is in the same state after processing either of strings $(ac)^r$ and $(ac)^q$. However, this implies that D ends in the same state q on both of inputs $(ac)^r(ac)^r \in A$ and $(ac)^q(ac)^r \notin A$. This is a contradiction if q is an accepting state or if q is not an accepting state, therefore A is not regular.

Solution 2: Let

$$\begin{array}{rcl} B & = & (\texttt{ac})^*(\texttt{bc})^* \\ C & = & A \cap B \\ & = & (\texttt{ac})^n(\texttt{bc})^n \end{array}$$

We use the pumping lemma to show that C is not regular. Because B is regular and the regular languages are closed under intersection, this will show that A is not regular either.

Let p be a proposed pumping constant for C, and let $w = (\texttt{ac})^p(\texttt{bc})^p \in C$. Let x, y and z be any three strings such that $w = xyz$, $|xy| \leq p$ and $|y| \geq 1$. Note that xy must be a prefix of $(\texttt{ac})^p$ because $|xy| \leq p < |(\texttt{ac})^p| = 2p$. Thus, $xy^2z = u(\texttt{bc})^p$ where $u \neq (\texttt{ac})^n$, which shows that $xy^2z \not{i}nC$. Therefore, C is not regular which shows that A cannot be regular either.