1. **(20 points** $+10$ ***points extra credit*)** Write regular expressions that generate each of the languages below. For each language, the alphabet, $\Sigma$, is $\{0, 1\}$.

   (a) $A_1 = \{s \mid s \text{ contains the substring } 1011\}$.

   **Solution:** $\Sigma^* 1011 \Sigma^*$

   (b) $A_2 = \{s \mid |s| = 5m + 7n \text{ with } m, n \in \mathbb{N}\}$.

   **Solution:** $(\Sigma^5)^*(\Sigma^7)^*$, where $\Sigma^5 = \Sigma\Sigma\Sigma\Sigma\Sigma$ and likewise for $\Sigma^7$.

   (c) $A_3 = \{s \mid s \text{ contains an even number of } 0\text{'s}\}$.

   **Solution:** $1^*(01^*0)^*1^*$

   (d) $A_4 = \{s \mid s \text{ contains an odd number of } 1\text{'s}\}$.

   **Solution:** $0^*10^*(10^*1)^*0^*$

   (e) (*10 points extra credit*):
   $A_5 = \{s \mid s \text{ contains an even number of } 0\text{'s and an odd number of } 1\text{'s}\}$.

   **Solution:**

```
Here's a DFA that recognizes the language
(q0 is th initial state, q$ is the accepting state).
  q0  -e-> q00
  q00 -0-> q01,  q00 -1-> q10
  q01 -0-> q00,  q01 -1-> q11
  q10 -0-> q11,  q10 -1-> q00
  q11 -0-> q10,  q10 -1-> q01
  q10 -e-> q$
This DFA is a GNFA and we use the procedure from the
Sept. 19 lecture notes:

delete q11:
  q0  -e-> q00
  q00 -0-> q01,   q00 -1-> q10
  q01 -0-> q00,   q01 -11-> q01, q01 -10-> q10
  q10 -00-> q10,  q10 -01-> q01, q10 -1-> q00
  q10 -e-> q$

delete q01:
  q0  -e-> q00
  q00 -0(11)^*0-> q00,
  q00 -(0(11)^*10 U 1)-> q10,
  q10 -(1 U 01(11)^*0)-> q00,
  q10 -(00 U 01(11)^*10)-> q10,
  q10 -e-> q$

delete q00:
  q0  -(0(11)^*0)^*(1 U 01(11)^*0) -> q10,
  q10 -(00 U 01(11)^*10 U (1 U 01(11)^*0) (0(11)^*0)^* (1 U 0(11)^*10)-> q10,
  -> q10,
  q10 -e-> q$

Let r0 = 0(11)^*0
```

```
    r1 = (1 U 01(11)ˆ*0)

Then, we've got
  q0  -(r0ˆ*r1)-> q10,
  q10 -(r0 U r1(r0ˆ*)r1)-> q10,
  q10 -e-> q$

Now, we eliminate q10 to get
  q0  -((r0ˆ*r1)(r0 U r1(r0ˆ*)r1)ˆ*)-> q$

Thus, our solution is:
  (r0ˆ*r1) (r0 U r1(r0ˆ*)r1)ˆ*
  with r0 = 0(11)ˆ*0 and r1 = (1 U 01(11)ˆ*0) as defined above.
```

I'll work on prettier typesetting later.

2. **(30 points)** In the problems below, let $R_1$, $R_2$, ... be arbitrary regular expressions over an arbitrary finite alphabet. For each proposed identity, either prove it, or give a counter-example. Two are valid identities for which a correct proof is worth 10 points; two are not valid identities for which a counter-example is worth 5 points.

   (a) $R_1 \cup \epsilon = R_1$.
   **Solution:**
   False. Let $R_1 = 0$. Then $\epsilon \in R_1 \cup \epsilon$ but $\epsilon \notin R_1$, and therefore $R_1 \cup \epsilon \neq R_1$.

   (b) $R_1 R_1^* = R_1^* R_1$.
   **Solution:**
   True. Using the definitions of concatenation and Kleen-star, $R_1 R_1^* = \{xy | x \in R_1, y \in R_1^*\}$ and $R_1^* = \{x_1 x_2 \ldots x_k | k \geq 0, x_i \in R_1 \forall i\}$ therefore,

$$\begin{array}{rcl} R_1 R_1^* & = & \{x x_1 \ldots x_k | x \in R_1, k \geq 0, x_i \in R_1 \forall i\} \\ & = & \{x_1 x_2 \ldots x_{k+1} | k \geq 0, x_i \in R_1 \forall i\} \\ & = & \{x_1 \ldots x_k x | x \in R_1, k \geq 0, x_i \in R_1 \forall i\} \\ & = & \{yx | x \in R_1, y \in R_1^*\} \\ & = & R_1^* R_1 \end{array}$$

   (c) $R_1 \cdot (R_2 \cup R_3) = (R_1 \cdot R_2) \cup (R_1 \cdot R_3)$.
   **Solution:**
   True. $R_1(R_2 \cup R_3) = \{xy | x \in R_1, y \in (R_2 \cup R_3)\} = \{xy | x \in R_1, y \in R_2 \vee y \in R_3\}$.
   $(R_1 R_2) \cup (R_1 R_3) = \{x | x \in (R_1 R_2) \vee x \in (R_1 R_3)\} = \{xy | (x \in R_1 \wedge y \in R_2) \vee (x \in R_1 \wedge y \in R_3)\} = \{xy | x \in R_1, y \in R_2 \vee y \in R_3\}$.

   (d) $R_1 \cup (R_2 \cdot R_3) = (R_1 \cup R_2) \cdot (R_1 \cup R_3)$.
   **Solution:**
   False. Let $R_1 = R_2 = R_3 = 0$. Then $R_1 \cup (R_2 \cdot R_3) = \{0, 00\} \neq \{00\} = (R_1 \cup R_2) \cdot (R_1 \cup R_3)$.

3. **(20 points)** For any language, $A$, let

$$A^{\mathcal{R}} = \{s \mid s^{\mathcal{R}} \in A\}$$

   where $s^{\mathcal{R}}$ is the *reverse* of $s$ as defined in homework 0:

$$\begin{array}{rcl} \epsilon^{\mathcal{R}} & = & \epsilon \\ (x \cdot c)^{\mathcal{R}} & = & c \cdot x^{\mathcal{R}} \end{array}$$

   Prove that if $A$ is regular, then so is $A^{\mathcal{R}}$.

**Solution:** Construct an NFA for $A^{\mathcal{R}}$.

Because $A$ is regular, we can represent it with an DFA. If we reverse the arcs between states and swap the start and accepting states, we get an NFA that recognizes $A^{\mathcal{R}}$. In the stuff that follows, I'll formalize this description, take care of a few technical details, and then prove that it works as advertised.

$A$ is an regular language. Let $M = (Q, \Sigma, \delta, q_0, F)$ be an DFA such that $L(M) = A$. Choose $q_x$ such that $q_x \notin Q$ (i.e. $q_x$ is a new state), and let $N = (Q \cup \{q_x\}, \Sigma, \delta^{\mathcal{R}}, q_x, \{q_0\})$, where

$$\begin{array}{rcll} \delta^{\mathcal{R}}(q,c) & = & \{p \mid \delta(p,c) = q\}, & \text{reverse the arcs, } q \neq q_x \\ \delta^{\mathcal{R}}(q_x,\epsilon) & = & F, & \text{start with an } \epsilon \text{ move to a final state of } M \\ \delta^{\mathcal{R}}(q_x,c) & = & \emptyset, & \text{force that initial } \epsilon \text{ move} \end{array}$$

I'll now prove that $L(N) = A^{\mathcal{R}}$. Because $N$ is an NFA, $L(N)$ is regular. Thus, this will show that $A^{\mathcal{R}}$ is regular.

The key to the proof is that after reading some string, $w^{\mathcal{R}}$, the set of possible states of $N$ are exactly those states from which $M$ could read $w$ and reach an accepting state. The proof is by induction on $w$.

Induction Hypothesis: $p \in (\delta^{\mathcal{R}}(\{q_x\}, w^{\mathcal{R}}) \cap Q) \Leftrightarrow \delta(p,w) \in F$.

Base case, $w = \epsilon$:

$$\begin{array}{lll} & p \in \delta^{\mathcal{R}}(\{q_x\}, w^{\mathcal{R}}) \cap Q & \\ \Leftrightarrow & p \in \delta^{\mathcal{R}}(\{q_x\}, \epsilon^{\mathcal{R}}) \cap Q, & w = \epsilon \\ \Leftrightarrow & p \in \delta^{\mathcal{R}}(\{q_x\}, \epsilon) \cap Q, & \epsilon = \epsilon^{\mathcal{R}} \\ \Leftrightarrow & p \in (\{q_x\} \cup F) \cap Q, & \text{For any set, } B,\ \delta^{\mathcal{R}}(B,\epsilon) = B \\ \Leftrightarrow & (p \in F) & (F \subseteq Q) \wedge (q_x \notin Q) \\ \Leftrightarrow & \delta(p,\epsilon) = \in F, & \text{For any state, } q,\ \delta(q,\epsilon) = q \\ \Box & & \end{array}$$

I showed all of the steps for completeness. It would be sufficient to write:

$$\begin{array}{ll} & p \in \delta^{\mathcal{R}}(\{q_x\}, \epsilon) \cap Q \\ \Leftrightarrow & p \in F \\ \Leftrightarrow & \delta(p,\epsilon) \in F \end{array}$$

Induction step, $w = \mathtt{c} \cdot x$: Noting that $(\mathtt{c} \cdot x)^{\mathcal{R}} = x^{\mathcal{R}} \cdot \mathtt{c}$, we need to prove

$$\begin{array}{lcl} & p \in \delta^{\mathcal{R}}(\{q_x\}, x^{\mathcal{R}} \cdot \mathtt{c}) & \\ \Leftrightarrow & \exists r \in \delta^{\mathcal{R}}(\{q_x\}, x^{\mathcal{R}}).\ p \in \delta^{\mathcal{R}}(r,c), & \text{def. } \delta^{\mathcal{R}} \text{ for strings} \\ \Leftrightarrow & \exists r \in \delta^{\mathcal{R}}(\{q_x\}, x^{\mathcal{R}}).\ \delta(p,c) = r, & \text{def. } \delta^{\mathcal{R}} \text{ for symbols} \\ \Leftrightarrow & \delta(p,c) \in \delta^{\mathcal{R}}(\{q_x\}, x^{\mathcal{R}}) & \\ \Leftrightarrow & \delta(\delta(p,c),x) \in F, & \text{induction hypothesis} \\ \Leftrightarrow & \delta(p, c \cdot x) \in F, & \text{def. } \delta \text{ for strings} \end{array}$$

Intuitively, what this argument says is that if $N$ can reach some state, $p$, by reading $x^{\mathcal{R}} \cdot c$; then it did it by first reaching some state, $r$, by reading $x^{\mathcal{R}}$, and then got to state $p$ by reading $c$. We then take advantage that $\delta^{\mathcal{R}}$ is the reversal of $\delta$. Thus, $M$ will go from $p$ to $r$ by reading $c$. Finally, we use the induction hypothesis with $r$ and $x$ to conclude that $M$ will go from $r$ to some state in $F$ by reading $x$. I will accept an intuitive argument like this one, or the mathematical version that I stated first.

4. **(40 points)**: For each language below, determine whether or not the language is regular. If it is regular, draw a DFA that accepts it and write a *short* explanation of how your DFA works. If it is not regular, provide a proof. For each language, the alphabet, $\Sigma$, is $\{\mathtt{0}, \mathtt{1}\}$. The notation $\#\mathtt{0}(s)$ refers to the number of $\#\mathtt{0}$'s in $s$, and $\#\mathtt{1}(s)$ refers to the number of $\#\mathtt{1}$'s.

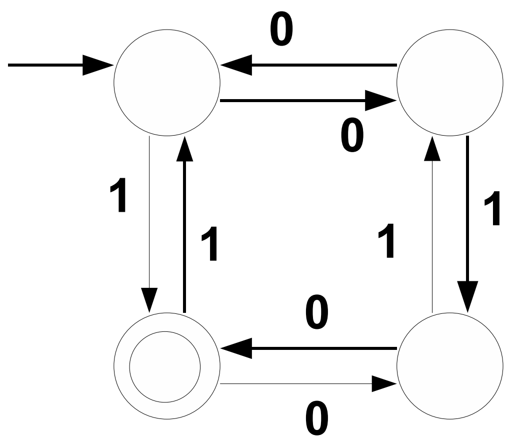


Figure 1: The $4$ states represent the parity of the number of $0$s and the parity of the number of $1$s seen so far: (even, even); (odd, even); (even, odd); (odd, odd).

(a) $B_1 = \{s \mid s$ contains an even number of `0`'s and an odd number of `1`'s$\}$.
**Solution:** (see DFA in Figure 1)
$B_1$ is regular:

(b) $B_2 = \{s \mid \#\mathtt{1}(s) = k * \#\mathtt{0}(s)$ for some $k \in \mathbb{N}\}$. **Solution:**
$B_2$ is not regular: Let $p$ be a proposed pumping lemma constant, and let $w = 0^p 1^p \in B_2$. For any $xyz = w$ with $|xy| \leq p$, $xy^2z = 0^{p+|y|}1^p$ has more 0s than 1s, and therefore is not in $B_2$. It follows by the pumping lemma that $B_2$ is not regular.

(c) $B_3 = \{s \mid (|\#\mathtt{0}(s) - \#\mathtt{1}(s)| \bmod 3) = 0\}$.
**Solution:** (see DFA in Figure 2)
$B_3$ is regular:

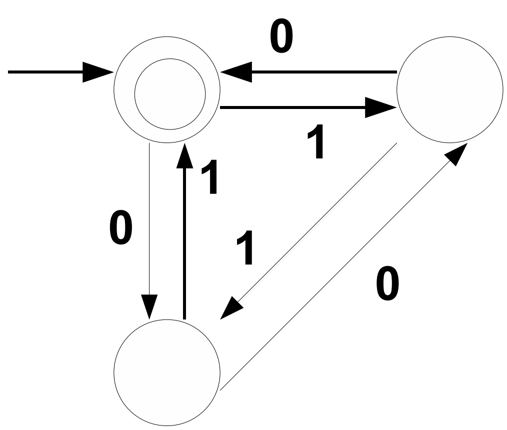


Figure 2: The 3 states track the value of the number of 0s seen thus far minus the number of 1s seen so far mod 3.

(d) $B_4 = \{s \mid (|\#\mathtt{0}(s) - \#\mathtt{1}(s)| \bmod 3) = 1\}$.
**Solution:**
$B_4$ is not regular: Let $p$ be a proposed pumping lemma constant, and let $w = 0^p 1^{p+1} \in B_4$. For any $xyz = w$ with $|xy| \leq p$, we consider three cases:

$$|y| = \begin{cases} 3k, & k \geq 1 \\ 3k+1, & k \geq 0 \\ 3k+2, & k \geq 0 \end{cases}$$

If $|y| = 3k$, then $xy^2z = w$ has $\#0(w) = p+3k$ and $\#1(w) = p+1$, so $|\#0(w) - \#1(w)| \pmod 3 = |p+3k-(p+1)| \pmod 3 = 2$ and $w \notin B_4$.
If $|y| = 3k+1$, then $xy^2z = w$ has $\#0(w) = p+3k+1$ and $\#1(w) = p+1$, so $|\#0(w) - \#1(w)| \pmod 3 = |p+3k+1-(p+1)| \pmod 3 = 0$ and $w \notin B_4$.
If $|y| = 3k+2$, then $xy^3z = w$ has $\#0(w) = p+6k+4$ and $\#1(w) = p+1$, so $|\#0(w) - \#1(w)| \pmod 3 = |p+6k+4-(p+1)| \pmod 3 = 0$ and $w \notin B_4$.
Therefore, $B_4$ is not regular by the pumping lemma.