1. Prove that the following languages over the alphabet $\Sigma = \{0, 1\}$ are *not* regular.

   (a) $\left\{0^a 1 0^b 1 0^c \;\middle|\; 2b = a + c\right\}$.

Solution:
Consider the set $F = 10^*1$.
Let x and y be arbitrary distinct strings in F.
Then $x = 10^i1$ and $y = 10^j1$ for some non-negative integers $i \neq j$.
Let $z = 0^{2i}$.

- $xz = 10^i10^{2i} = 0^a10^b10^c$, where $a = 0$ and $b = i$ and $c = 2i$.
  Because $2b = 2i = a + c$, we have $xz \in L$.
- $yz = 10^j10^{2i} = 0^a10^b10^c$, where $a = 0$ and $b = j$ and $c = 2i$.
  Because $2b = 2j \neq 2i = a + c$, we have $yz \notin L$.

Thus, z is a distinguishing suffix for x and y.
We conclude that F is a fooling set for L.
Because F is infinite, L cannot be regular. ■

Solution:
Consider the set $F = 0^*$.
Let x and y be arbitrary distinct strings in F.
Then $x = 0^i$ and $y = 0^j$ for some non-negative integers $i \neq j$.
Let $z = 10^i10^i$.

- $xz = 0^i10^i10^i = 0^a10^b10^c$, where $a = b = c = i$.
  Because $2b = 2i = a + c$, we have $xz \in L$.
- $yz = 0^j10^i10^i = 0^a10^b10^c$, where $a = j$ and $b = i$ and $c = i$.
  Because $2b = 2i \neq i + j = a + c$, we have $yz \notin L$.

Thus, z is a distinguishing suffix for x and y.
We conclude that F is a fooling set for L.
Because F is infinite, L cannot be regular. ■

Solution:
Consider the set $F = \{0^n10^{2n}1 \mid n \geq 0\}$.
Let x and y be arbitrary distinct strings in F.
Then $x = 0^i10^{2i}1$ and $y = 0^j10^{2j}1$ for some non-negative integers $i \neq j$.
Let $z = 0^{3i}$.

- $xz = 0^i10^{2i}10^{3i} = 0^a10^b10^c$, where $a = b = c = i$.
  Because $2b = 4i = a + c$, we have $xz \in L$.
- $yz = 0^j10^{2j}10^{3i} = 0^a10^b10^c$, where $a = j$ and $b = 2j$ and $c = 3i$.
  Because $2b = 4j \neq j + 3i = a + c$, we have $yz \notin L$.

Thus, z is a distinguishing suffix for x and y.

We conclude that F is a fooling set for L.
Because F is infinite, L cannot be regular. ■

Rubric: 3 points: standard fooling set rubric (scaled). These are not the only correct solutions. Watch out for boundary conditions and off-by-one errors!

(b) The set of all palindromes in Σ^* whose lengths are divisible by 7.

Solution:
Consider the set $F = (\texttt{1000000})^*$.
Let x and y be arbitrary distinct strings in F.
Then $x = (\texttt{1000000})^i$ and $y = (\texttt{1000000})^j$ for some non-negative integers $i \neq j$.
Without loss of generality, assume $i < j$. (Othterwise swap x and y.)
Let $z = (\texttt{0000001})^i$.

- $xz = (\texttt{1000000})^i(\texttt{0000001})^i$, which is a palindrome of length $7 \cdot 2i$, so $xz \in L$.
- $yz = (\texttt{1000000})^i(\texttt{0000001})^j$. The $(7i+1)$th bit of yz is $\texttt{0}$, but because $i < j$, the $(7i+1)$th bit of $(yz)^R = (\texttt{1000000})^j(\texttt{0000001})^i$ is $\texttt{1}$. It follows that $yz \neq (yz)^R$, so yz is not a palindrome, so $yz \notin L$.

Thus, z is a distinguishing suffix for x and y.
We conclude that F is a fooling set for L.
Because F is infinite, L cannot be regular. ■

Rubric: 3 points: standard fooling set rubric (scaled). This is not the only correct solution. Watch out for boundary conditions and off-by-one errors!

(c) $\left\{1^m 0^n \;\middle|\; m+n>0 \text{ and } \gcd(m,n)=1\right\}$

Solution: Consider the set $F = \{1^p \mid p \text{ is prime}\}$.
Let x and y be arbitrary distinct strings in F.
Then $x = 1^p$ and $y = 1^q$ for some distinct primes p and q.
Let $z = 0^q$.

- Then $xz = 1^p 0^q \in L$ because $\gcd(p,q) = 1$.
- But $yz = 1^q 0^q \notin L$ because $\gcd(q,q) = q \neq 1$.

Thus, z is a distinguishing suffix for x and y.
We conclude that F is a fooling set for L.
Because F is infinite, L cannot be regular. ■

Solution: Consider the set $F = 1^*$.
Let x and y be arbitrary distinct strings in F.
Then $x = 1^i$ and $y = 1^i$ for some non-negative integers i and j.
Without loss of generality, assume $i < j$.
Let $z = 1^{(k-1)i} 0^k$, where k is any positive integer such that $\gcd(j-i,k) = 1$. (For example, let k be any prime larger than j.)

- Then $xz = 1^{ki} 0^k \notin L$ because $\gcd(ki,k) = k \neq 1$.
- But $yz = 1^{ki+j-i} 0^p \notin L$ because $\gcd(ki+j-i,k) = \gcd(k,j-i) = 1$, by Aryabhata's Pulverizer:[a]

$$\gcd(x,y) = \begin{cases} x & \text{if } y = 0 \\ \gcd(y, x \bmod y) & \text{if } y > 0 \end{cases}$$

Thus, z is a distinguishing suffix for x and y.
We conclude that F is a fooling set for L.
Because F is infinite, L cannot be regular. ■

[a] known in the West as "Euclid's algorithm"

Rubric: 4 points: standard fooling set rubric (scaled). These are not the only correct solutions. Watch out for boundary conditions and off-by-one errors!

2. For each of the following languages over the alphabet $\Sigma = \{0, 1\}$, either prove that the language is regular (by constructing an appropriate DFA, NFA, or regular expression) or prove that the language is not regular (by constructing an infinite fooling set).

   [Hint: Exactly two of these languages are regular.]

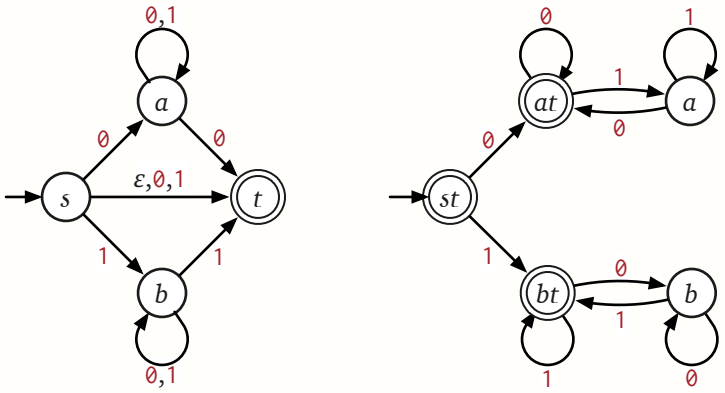
   (a) Strings in which the substrings 01 and 10 appear the same number of times. For example, $1100011 \in L$ because both substrings appear once, but $01000011 \notin L$.

Solution: Regular.

This is the set of all non-empty binary strings whose first and last symbols (if any) are equal. Thus, this language is described by the regular expression

$$\varepsilon + 0 + 1 + 0(0+1)^*0 + 1(0+1)^*1.$$

and is accepted by the following NFA (on the left) and DFA (on the right):



Every string can be partitioned into an alternating sequence of substrings in 0^+ and substrings in 1^+; for example:

$$000000011110001111011111 = 0^7 \bullet 1^4 \bullet 0^3 \bullet 1^5 \bullet 0^1 \bullet 1^5$$

The substrings 01 and 10 appear precisely at the boundaries of these runs:

$$0000000\underline{11}1\underline{10}0\underline{11}1111\underline{01}1111$$

Thus, the number of 01 substrings and the number of 10 substrings are equal if and only if the first and last symbols are equal (or the string is empty). ■

Rubric: 2½ points: ½ for "regular" + 1 for regular expression **or** DFA **or** NFA + 1 for justification. This is not the only correct regular expression, NFA, or DFA.

A correct solution requires only one representation—a regular expression **or** an NFA **or** a DFA—not all three, nor even two.

(b) Strings in which the substrings 00 and 11 appear the same number of times. For example, $1100011 \in L$ because both substrings appear twice, but $01000011 \notin L$.

Solution (fooling set): Not regular.

Consider the set $F = 0^+ = 00^*$.

Let x and y be arbitrary distinct strings in F.

Then $x = 0^i$ and $y = 0^j$ for some *positive* integers $i \neq j$.

Let $z = 1^i$.

- Then $xz = 0^i1^i \in L$ because 00 and 11 each appear $i-1$ times (because $i > 0$).
- But $yz = 0^j1^i \notin L$ because 00 appears $j-1$ times (because $j > 0$), 11 appears $i-1$ times (because $i > 0$), and $i \neq j$.

Thus, z is a distinguishing suffix for x and y.

We conclude that F is a fooling set for L.

Because F is infinite, L cannot be regular. ■

Rubric: 2½ points: ½ for "not regular" + 2 for fooling-set proof (= ½ for correct fooling set + ½ for distinguishing suffix + ½ for xz proof + ½ for yz proof). This is not the only correct fooling-set proof.

Solution (reduction): Not regular.

For the sake of argument, suppose L is regular.

Then $L' = L \cap 00^*11^*$ is also regular.

Let i and j be arbitrary positive integers, and consider the string 0^i1^j. The substring 00 appears $i-1$ times in 0^i1^j, and the substring 11 appears $j-1$ times in 0^i1^j. Thus, $0^i1^j \in L'$ if and only if $i = j$. We conclude that $L' = \{0^n1^n \mid n \geq 1\}$.

It follows that $L' \cup \{\varepsilon\} = \{0^n1^n \mid n \geq 0\}$ is also regular.

But that's impossible! We proved in class (and in the lecture notes) that $\{0^n1^n \mid n \geq 0\}$ is not regular. ■

Rubric: 2½ points: ½ for "not regular" + 2 for reduction/closure proof. This is not the only correct reduction proof.

(c) $\{xyyx \mid x, y \in \Sigma^+\}$

Solution: Not regular.

To avoid variable collisions, I'll redefine the target language as $\{uvvu \mid u, v \in \Sigma^+\}$.

Consider the set $F = 0^*011$.

Let x and y be arbitrary distinct strings in F.

Then $x = 0^i11$ and $y = 0^j11$ for some positive integers $i \neq j$.

Let $z = 0^i$.

- Then $xz = 0^i110^i = 0^i \bullet 1 \bullet 1 \bullet 0^i \in L$. (Here we can take $u = 0^i$ and $v = 1$.)
- But $yz = 0^i110^j \notin L$.

  Suppose to the contrary that $0^i110^j = uvvu$ for some non-empty strings u and v. String u cannot contain any 1s; otherwise $uvvu$ would contain two non-consecutive 1s. So v must contain exactly one 1. So the first 1 in 0^i110^j must be the last bit of the first copy of v, and the second 1 in 0^i110^j must be the first bit of the second copy of v. This is only possible if $v = 1$. But now we've reached a contradiction; u cannot equal both 0^i and 0^j.

Thus, z is a distinguishing suffix for x and y.

We conclude that F is a fooling set for L.

Because F is infinite, L cannot be regular. ■

Rubric: 2½ points: ½ for "not regular" + 2 for fooling-set proof (= ½ for correct fooling set + ½ for distinguishing suffix + ½ for xz proof + ½ for yz proof). This is not the only correct proof.

(d) $\{xyyz \mid x, y, z \in \Sigma^+\}$

Solution: Regular.

$$L = (0+1)(0+1)^*(00+11+0101+1010)(0+1)^*(0+1).$$

The regular expression describes all strings the form $xyyz$ for some non-empty strings x, y, z such that $y \in 00+11+0101+1010$. So every string that matches the regular expression is in L.

On the other hand, let x, y, and z be arbitrary non-empty strings, and consider the string $w = xyyz$.

- If y begins and ends with 0 or contains the substring 00, then yy contains the substring 00. Then we can write $yy = x'00z'$ for some strings x' and z', and therefore $w = x''y'y'z''$, where $x'' = xx'$ and $y' = 0$ and $z'' = z'z$. We conclude that w matches the regular expression.
- If y begins and ends with 1 or contains the substring 11, then yy contains the substring 11, so w matches the regular expression.
- If y begins with 1 and ends with 0, but does not contain 00 or 11, then yy contains the substring 1010, so w matches the regular expression.
- If y begins with 0 and ends with 1, but does not contain 00 or 11, then yy contains the substring 0101, so w matches the regular expression.

So every string in L matches the regular expression. ■

Solution: Regular.

Let w be an arbitrary string of length at least 6, and let $abcd$ be any four consecutive bits in w that exclude the first and last bits of w. There are four cases to consider:

- If $abcd$ contains the substring 00, then we can write $w = x00z$ for some non-empty strings x and z. Setting $y = 0$ shows that $w \in L$.
- If $abcd$ contains the substring 11, then we can write $w = x11z$ for some non-empty strings x and z. Setting $y = 1$ shows that $w \in L$.
- If $abcd = 0101$, then we can write $w = x0101z$ for some non-empty strings x and z. Setting $y = 01$ shows that $w \in L$.
- Finally, if $abcd = 1010$, then we can write $w = x1010z$ for some non-empty strings x and z. Setting $y = 10$ shows that $w \in L$.

We conclude that L contains *every* string of length at least 6.

There are only a finite number of strings of length 5, so $L = (0+1)^6(0+1)^* + A$ for some finite (and therefore regular) language A. ■

Rubric: 2½ points: ½ for "regular" + 1 for regular expression + 1 for justification. Writing out the finite set A is not required for full credit. These are not the only correct solutions.

*3. ***Practice only. Do not submit solutions.***

See the homework handout for definitions of Moore machines, $L^{\circ}(M)$, and $L^{=}(M)$.

(a) Let M be an arbitrary Moore machine. Prove that $L^{\circ}(M)$ is a regular language.

Solution: Let $M = (\Sigma, \Gamma, Q, s, \delta, \omega)$ be the given Moore machine. We construct an NFA $M' = (\Sigma', Q', s', A', \delta')$ that accepts $L^{\circ}(M)$ as follows. First we define the input alphabet and various state sets:

$$\Sigma' = \Gamma, \qquad Q' = Q, \qquad s' = s, \qquad A' = Q.$$

The transition function δ' is defined as follows, for all $q \in Q$ and $b \in \Gamma$:

$$\delta'(q, b) := \left\{ \delta(q, a) \;\middle|\; a \in \Sigma \text{ and } \omega(\delta(q, a)) = b \right\}.$$

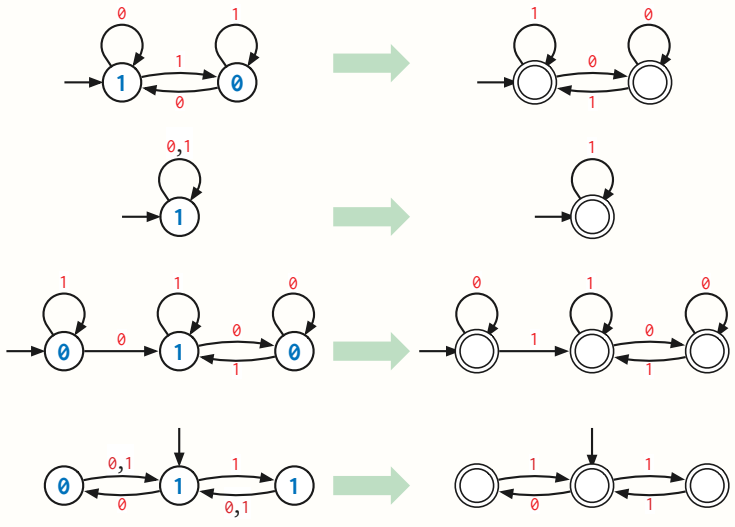
Less formally, we build M' from M by replacing every transition $p \xrightarrow{a} q$ with $p \xrightarrow{\omega(q)} q$, and then letting *every* state accept.

Whenever M' reads a symbol $b \in \Gamma$ while in state q, it non-deterministically guesses a symbol $a \in \Sigma$ such that $\omega(\delta(q, a)) = b$ and transitions to state $\delta(q, a)$. If there is no such symbol, the current execution thread fails.

Each state q in M' indicates that M' has just read the output string $\omega^*(s, w)$, for some input string $w \in \Sigma^*$ such that $\delta^*(s, w) = q$.

Rubric: This would be enough for full credit.

For example, in the figure below, for each Moore machine M on the left, we would construct the corresponding NFA M' on the right. In each Moore machine, the input symbols are indicated in red on the edges/transitions, and the output symbols are indicated in blue on the vertices/states.



We can informally argue the correctness of our construction as follows. A *walk* in an NFA or a Moore machine is a sequence of transitions (that is, either a single state, or a transition followed by a walk).

An *accepting walk* in an NFA is any walk from the start state to any accepting state. The *transition string* of an accepting walk is the concatenation of the symbols labeling each transition. An NFA accepts a string y if and only if there is an accepting walk whose transition string is y.

Similarly, the *output string* of a walk in a Moore machine is the concatenation of the *output* symbols of the states, ignoring the beginning state. A string y is in the output language of a Moore machine M if and only if there is a walk in M that starts at s and whose output string is y.

Now consider our NFA M'. Accepting walks in M' starts at $s' = s$ and can end at any state. Every transition in M' is also a transition in M and vice versa, so every walk in M is also a walk in M' and vice versa. The *transition* string of any walk in M is equal to the *output* string of the *same* walk in M'. We conclude that M' accepts a string y if and only if M' can output the string y.

If we absolutely have to, we can *formally* prove correctness by tedious inductive definition-chasing. Here we go:

Lemma 1. *For all states $p, q \in Q$ and every string $x \in \Gamma^*$, we have $q \in (\delta')^*(p, x)$ if and only if there is a string $w \in \Sigma^*$ such that $\delta^*(p, w) = q$ and $\omega^*(p, w) = x$.*

Proof: Let x be an arbitrary string in Γ^*, and let p and q be arbitrary states in Q. Assume, for every state r and every string $y \in \Gamma^*$ that shorter than x, that we have $q \in (\delta')^*(r, x)$ if and only if there is a string $v \in \Sigma^*$ such that $\delta^*(r, v) = q$ and $\omega^*(r, v) = y$. There are two cases to consider:

If $x = \varepsilon$, then by definition, $q \in (\delta')^*(p, x)$ if and only if $p = q$. Similarly by definition, $\delta^*(p, w) = q$ and $\omega^*(p, \varepsilon) = \varepsilon$.

On the other hand, if $x = by$ for some symbol $b \in \Gamma$ and string $y \in \Gamma^*$, then

$$
\begin{aligned}
& q \in (\delta')^*(p, x) \\
& \quad \iff q \in (\delta')^*(r, y) && \text{for some } r \in \delta'(s, b) \\
& \quad \iff q \in (\delta')^*(\delta(p, a), y) && \text{for some } a \in \Sigma \text{ such that } \omega(\delta(p, a)) = b \\
& \quad \iff \delta^*(\delta(p, a), v) = q \text{ and } \omega^*(\delta(p, a), v) = y \\
& \qquad\qquad && \text{for some } a \in \Sigma \text{ and } v \in \Sigma^* \text{ such that } \omega(\delta(p, a)) = b \\
& \quad \iff \delta^*(p, av) = q \text{ and } \omega^*(p, av) = by && \text{for some } a \in \Sigma \text{ and } v \in \Sigma^* \\
& \quad \iff \delta^*(p, w) = q \text{ and } \omega^*(p, w) = x && \text{for some } w \in \Sigma^*
\end{aligned}
$$

Here the first equivalence is by definition of $(\delta')^*$, the second equivalence is by definition of δ'; the third equivalence follows from the induction hypothesis; the fourth equivalence is by definition of δ^* and ω^*; and the fifth equivalence follows from setting $w = av$. □

The correctness of our construction now follows from Lemma 1 by setting $p = s$. ■

(b) Let M be an arbitrary Moore machine whose input alphabet Σ and output alphabet Γ are identical. Prove that $L^{=}(M)$ is a regular language.

Solution: Let $M = (\Sigma, \Sigma, Q, s, \delta, \omega)$ be the given Moore machine. We construct a DFA $M' = (\Sigma', Q', s', A', \delta')$ that accepts $L^{\circ}(M)$ as follows:

$$\Sigma' = \Sigma$$
$$Q' = Q \cup \{fail\}$$
$$s' = s$$
$$A' = Q$$
$$\delta'(q,a) = \begin{cases} \delta(q,a) & \text{if } \omega(\delta(q,a)) = a \\ fail & \text{otherwise} \end{cases} \qquad \text{for all } q \in Q \text{ and } a \in \Sigma$$
$$\delta'(fail, a) = fail \qquad \text{for all } a \in \Sigma$$

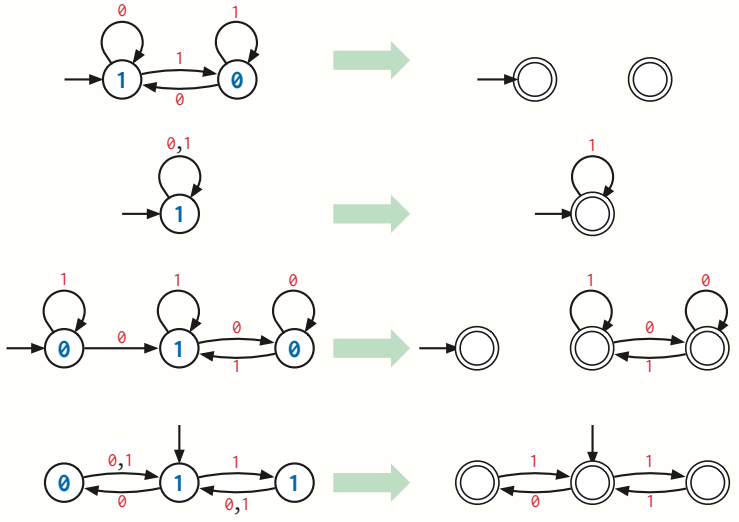
Less formally, we build M' from M by redirecting every transition $p \xrightarrow{a} q$ where $\omega(q) \neq a$ to a new fail state, and then letting every original state accept.

Whenever M' reads a symbol $a \in \Sigma$ while in state $q \in Q$, it either transitions to state $\delta(q,a)$ or fails, depending on whether $\omega(\delta(q,a)) = a$.

Each state q in M' indicates that M' has just read a string w such that $\delta^*(s,w) = q$ and $\omega^*(s,w) = w$.

Rubric: This would be enough for full credit.

For example, in the figure below, for each Moore machine M on the left, we would construct the corresponding NFA M' on the right. In each Moore machine, the input symbols are indicated in red on the edges/transitions, and the output symbols are indicated in blue on the vertices/states. The first and third NFAs have no transitions out of their start states, which means they reject every non-empty input; in those two cases we have $L^{=}(M) = \{\varepsilon\}$.



■