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

(a) $\left\{ 0^a 1^b 0^c \mid \text{if } a \geq 1 \text{ then } b = c \right\}$

Solution:

Consider the set $F = 011^*$.

Let x and y be arbitrary distinct strings in F.

Then $x = 01^i$ and $y = 01^j$ for some *positive* integers $i \neq j$.

Let $z = 0^i$.

- $xz = 01^i 0^i = 0^a 1^b 0^c$, where $a = 1$ and $b = c = i$. So $xz \in L$.
- $yz = 01^j 0^i = 0^a 1^b 0^c$, where $a = 1$ and $j = b \neq c = i$. 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!

This is an example of a non-regular language that cannot be proved non-regular using (only) the pumping lemma, but it's close enough to a pumping-lemma language that most LLMs produce incorrect pumping lemma proofs. (You don't need to know what the pumping lemma is for this class, but you can find a description in almost every automata-theory textbook.)

(b) The set of all palindromes in Σ^* whose lengths are divisible by 5

Solution:

Consider the set $F = (10000)^*$.

Let x and y be arbitrary distinct strings in F.

Then $x = (10000)^i$ and $y = (10000)^j$ for some non-negative integers $i \neq j$.

Without loss of generality, assume $i < j$. (Otherwise swap x and y.)

Let $z = (00001)^i$.

- $xz = (10000)^i (00001)^i$, which is a palindrome of length $5 \cdot 2i$, so $xz \in L$.
- $yz = (10000)^i (00001)^j$. The $(5i+1)$th bit of yz is 0, but because $i < j$, the $(5i+1)$th bit of $(yz)^R = (10000)^j (00001)^i$ is 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) Even-length binary strings whose first half contains an odd number of 1s. More formally:

$$\left\{ w \in \Sigma^* \;\middle|\; \begin{array}{c} w = xy \text{ for some strings } x \text{ and } y \text{ such that} \\ |x| = |y| \text{ and } \#(1, x) \text{ is odd} \end{array} \right\}$$

Solution: Consider the set $F = (11)^*1 = \{1^{2n+1} \mid n \geq 0\}$.

Let x and y be arbitrary distinct strings in F.

Then $x = 1^{2i+1}$ and $y = 1^{2j+1}$ for some non-negative integers i and j.

Without loss of generality, assume $i < j$. (Otherwise, swap x and y.)

Let $z = 10^{2j}$.

- Then $xz = 1^{2i+2}0^{2j}$ has even length $2i + 2j + 2$, but because $i < j$, the first half of this string has length $i + j + 1 \geq 2i + 2$, and therefore contains the prefix 1^{2i+2}. So $xz \notin L$.
- Then $yz = 1^{2i+2}0^{2i}$ has even length $4i + 2$ and its first half 1^{2i+1} contains an odd number of 1s. So $yz \in L$.

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: 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 (using a fooling-set argument).

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

   (a) $\{xy \mid |x| \le 374 \text{ and } |y| \ge 374 \text{ and } y \text{ is a palindrome}\}$

Solution: Not regular.

Consider the set $F = 0^{374}10^{374}0^*$.

Let x and y be arbitrary distinct strings in F.

Then $x = 0^{374}10^{374+i}$ and $y = 0^{374}10^{374+j}$ for some non-negative integers $i \neq j$.

Let $z = 10^{374+i}1$.

- We have $xz = 0^{374}10^{374+i} \bullet 10^{374+i}1 = 0^{374} \bullet 10^{374+i}10^{374+i}1$. Thus, we can write $xz = uv$, where $u = 0^{374}$ and $v = 10^{374+i}10^{374+i}1$. The prefix u has length at most (in fact equal to) 374, and the suffix v is a palindrome with length at least 374. Thus, $xz \in L$.
- On the other hand, $yz = 0^{374}10^{374+j} \bullet 10^{374+i}1$. If we write $yz = uv$ where v is a palindrome, then v must both start and end with 1, so there are only three possibilities:
  - If $v = 1$, then $|v| < 374$.
  - If $v = 10^{374+i}1$, then $u = 0^{374} \cdot 10^{374+i}$, and therefore $|u| > 374$.
  - If $v = 10^{374+j}10^{374+i}1$, then v is not a palindrome, because $i \neq j$.

  We conclude that $yz \neq L$.

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: Not regular.

Consider the set $F = 0^{374}(100)^{374}(100)^*$.

Let x and y be arbitrary distinct strings in F.

Then $x = 0^{374}(100)^{374+i}$ and $y = 0^{374}(100)^{374+j}$ for some non-negative integers $i \neq j$.

Let $z = (001)^{374+i}$.

- We have $xz = 0^{374}(100)^{374+i} \bullet (001)^{374+i}$. Thus, we can write $xz = uv$, where $u = 0^{374}$ and $v = (100)^{374+i}(001)^{374+i}$. The prefix u has length at most (in fact equal to) 374, and the suffix v is a palindrome with length at least 374. Thus, $xz \in L$.
- On the other hand, $yz = 0^{374}(100)^{374+j} \bullet (001)^{374+i}$. Suppose we write $yz = uv$ where $|u| \le 374$ and v is a palindrome. The palindrome suffix v ends with 1 and therefore starts with 1, so we must have $u = 0^{374}$ and $v = (100)^{374+j}(001)^{374+i}$. But then v is not a palindrome after all, because $i \neq j$. We conclude that $yz \neq L$.

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: 2½ points: standard fooling set rubric (scaled). These are not the only correct solutions. These solutions are more detailed than necessary for full credit, but a full-credit solution must justify the claims "$xz \in L$" and "$yz \notin L$".

(b) $\{xy \mid |x| \le 374 \text{ and } |y| \ge 374 \text{ and } x \text{ is a palindrome}\}$

Solution: Regular.

Let P be the set of all palindromes of length at most 374, and let S be the set of all strings with length at least 374. Both of these languages are regular:

- P is finite and therefore regular.
- S matches the regular expression $(0+1)^{374}(0+1)^*$.

Thus, our target language $P \bullet S$ is the concatenation of two regular languages, so it must be regular. ■

Solution: Regular.

This is the set of all strings with length at least 374, so it matches the regular expression $(0+1)^{374}(0+1)^*$.

- Let w be any string in our target language L. By definition, $w = xy$ for some strings x and y such that $|y| \ge 374$ (and satisfying some other conditions). Thus, $|w| \ge |y| \ge 374$.
- Let w be any string with length at least 374. Let $x = \varepsilon$ and $y = w$. Then $w = xy$ and $|x| \le 374$ and $|y| \ge 374$ and x is a palindrome. We conclude that $w \in L$.

■

Rubric: 2½ points: ½ for "regular" + 1 for regular expression + 1 for justification (= ½ for "if" + ½ for "only if"). This is more detail than necessary for full credit.

(c) $\{wxw^R \mid w, x \in \Sigma^+\}$

Solution: Regular.

This is the language $\mathtt{0}(\mathtt{0}+\mathtt{1})^+\mathtt{0}+\mathtt{1}(\mathtt{0}+\mathtt{1})^+\mathtt{1}$ of all strings of length at least 3 that start and end with the same symbol.

- Let z be an arbitrary string in our target language L.
  By definition, $z = wxw^R$ for some non-empty strings w and x.
  Because $w \neq \varepsilon$, we have $w = ay$ for some symbol a and some string y.
  The definition of reversal implies $w^R = y^R a$.
  Thus, $z = ayxy^R a$ starts and ends with the same symbol a.
  The remaining substring yxy^R is non-empty, because x is nonempty.
  We conclude that $z \in \mathtt{0}(\mathtt{0}+\mathtt{1})^+\mathtt{0}+\mathtt{1}(\mathtt{0}+\mathtt{1})^+\mathtt{1}$.

- On the other hand, let z be an arbitrary string in $\mathtt{0}(\mathtt{0}+\mathtt{1})^+\mathtt{0}+\mathtt{1}(\mathtt{0}+\mathtt{1})^+\mathtt{1}$.
  Then $z = axa$ for some symbol a and some nonempty string x.
  Because $a = a^R$, we have $z = axa^R$, which implies $z \in L$.

We conclude that $L = \mathtt{0}(\mathtt{0}+\mathtt{1})^+\mathtt{0}+\mathtt{1}(\mathtt{0}+\mathtt{1})^+\mathtt{1}$. ■

Rubric: 2½ points: ½ for "regular" + 1 for regular expression + 1 for justification (= ½ for "if" + ½ for "only if"). This is more detail than necessary for full credit.

(d) $\{xww^R \mid w, x \in \Sigma^+\}$

Solution: Not regular.

Consider the set $F = \text{“}110^{\text{odd}}1\text{”} = 11(00)^*01$.

Let x and y be arbitrary distinct strings in F.

Then $x = 110^{2i+1}1$ and $y = 110^{2j+1}1$ for some non-negative integers $i \neq j$.

Let $z = 10^{2i+1}1$.

- Then $xz = 1 \cdot 10^{2i+1}1 \cdot 10^{2i+1}1 = vww^R$, where $v = 1$ and $w = 10^{2i+1}1$. It follows that $xz \in L$.
- For the sake of argument, suppose $yz = 110^{2j+1}110^{2i+1}1 \in L$.
  Then $yz = vww^R$ for some non-empty strings w and v.
  The last two symbols of yz are different, so $|w| > 1$.
  The suffix w^R ends with 01 (the last two symbols of yz).
  So its reversal w must begin with 10.
  The substring 10 appears exactly twice in yz.
  So there are only two possibilities for the substring ww^R.
  - $ww^R = 10^{2j+1}1$ is impossible because $|ww^R|$ must be even.
  - $ww^R = 10^{2j+1}110^{2i+1}1$ is impossible because ww^R must be a palindrome, and $i \neq j$.

  We have derived a contradiction, which implies that $yz \notin L$.

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. ■

The main idea here is to impose additional structure that forces the substring w to be arbitrarily long. We are really reasoning about the language

$$L \cap 1 \cdot 1(00)^*01 \cdot 1(00)^*01 = \{1 \cdot 10^n1 \cdot 10^n1 \mid n \text{ is odd}\}.$$

If R is a regular language and $L \cap R$ is not regular, then L cannot be regular.

Solution: Not regular.

Consider the set $F = 010(10)^*$.

Let x and y be arbitrary distinct strings in F.

Then $x = 0(10)^i$ and $y = 0(10)^j$ for some *positive* integers $i \neq j$.

Without loss of generality, assume $i > j$. (Otherwise swap x and y.)

Let $z = (01)^i$.

- Then $xz = 1(10)^i(01)^i = vww^R$, where $v = 1$ and $w = (10)^i$. It follows that $xz \in L$.
- For the sake of argument, suppose $yz = 1(10)^j(01)^i \in L$.
  Then $yz = vww^R$ for some non-empty strings w and v.
  The last symbol in w and the first symbol in w^R are equal.
  There is only one place in yz where the same symbol appears twice in a

row, so we must have $w^R = (01)^i$ and therefore $w = (10)^i$.
Thus, $|yz| = |vww^R| = |v| + |w| + |w^R| \geq 4i + 1$.
But this is impossible; $i > j$ implies $|yz| = |y| + |z| = 2j + 1 + 2i < 4i + 1$.
We have derived a contradiction, which implies that $yz \notin L$.

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.

Again, we are imposing additional structure that forces w to be arbitrarily long. We are really reasoning about the language $L \cap 0(10)^*(01)^*$. ■

Rubric: 2½ points: standard fooling set rubric (scaled). 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 $b \in \Gamma$:

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

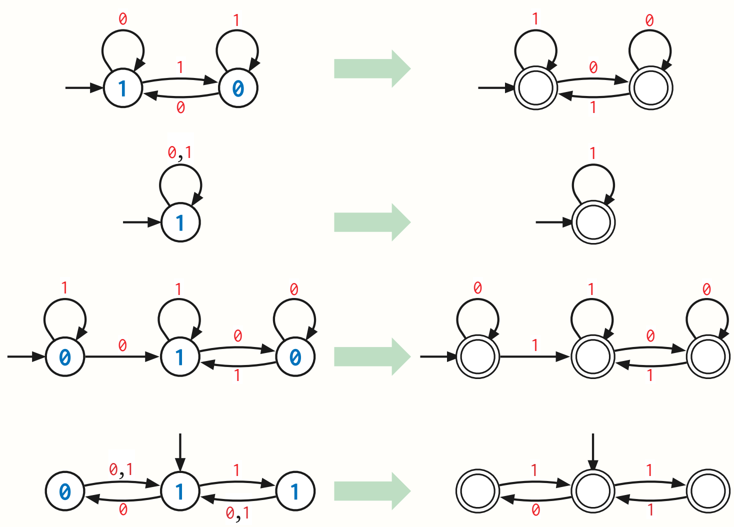
Less formally, we build M' from M by replacing every transition $p \xrightarrow{a} q$ with $p \xrightarrow{\omega(q)} q$, and then letting *every* state accept.

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:

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

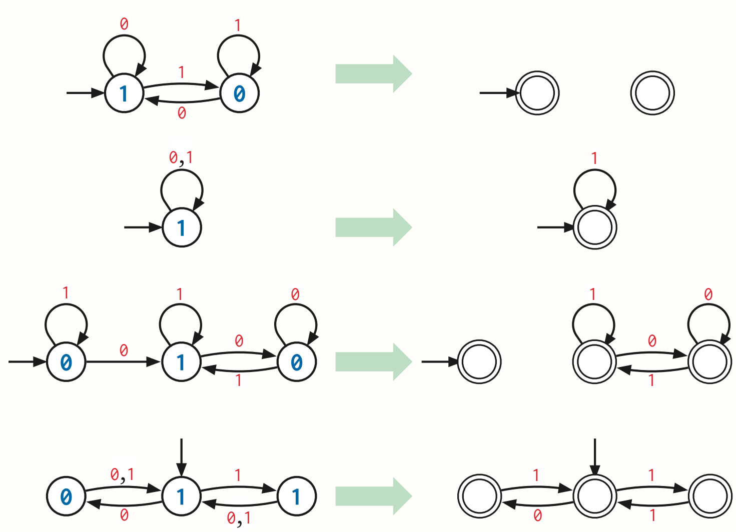
Less formally, we build M' from M by redirecting every transition $p \xrightarrow{a} q$ where $\omega(q) \neq a$ to a new fail state, and then letting every original state accept.

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 $L^{=}(M) = \{\varepsilon\}$.



■