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

Rubric: 2½ points for each part: standard fooling set rubric (scaled). Most solutions are more detailed than necessary for full credit. These are not the only correct solutions.

(a) $\left\{0^a 1^b 0^c 1^d \;\middle|\; a, b, c, d \geq 1 \text{ **and** if } a = d, \text{ then } b = c\right\}$

Solution: Call this language L_a.
Consider the set $F = \left\{01^i \;\middle|\; i \geq 1\right\}$.
Let x and y be arbitrary distinct strings in F.
Then $x = 01^i$ and $y = 01^j$ for some non-negative integers $i \neq j$.
Let $z = 0^i 1$.

- $xz = 01^i 0^i 1 \in L_a$.
- $yz = 01^j 0^i 1 \notin L_a$.

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

(b) All strings in which every prefix has at least as many 1s as 0s. More formally:

$$\{w \in \Sigma^* \mid \text{for all } x, y \text{ such that } w = xy,\ \#(1, x) \geq \#(0, x)\}$$

Solution: Call this language L_b.
Consider the set $F = \left\{1^i \;\middle|\; i \geq 0\right\}$.
Let x and y be arbitrary distinct strings in F.
Then $x = 1^i$ and $y = 1^j$ for some non-negative integers $i \neq j$.
Without loss of generality, assume $i < j$.
Let $z = 0^j$.

- $xz = 1^i 0^j$. String xz is a prefix of itself, and $\#(1, xz) = i < j = \#(0, xz)$, so $xz \notin L_b$.
- $yz = 1^j 0^j$. Every prefix is of the form $w = 1^k$ for which $\#(1, w) = k \geq 0 = \#(0, w)$ or $1^j 0^k$ with $j \geq k$ for which $\#(1, w) = j \geq k = \#(0, w)$, so $yz \in L_b$.

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

(c) $\{xyx \mid x, y \in \Sigma^* \text{ and } |x| = |y|\}$

Solution: Call this language L_c.
Consider the set $F\left\{10^{2i+1} \;\middle|\; i \geq 0\right\}$.
Let x and y be arbitrary distinct strings in F.
Then $x = 10^{2i+1}$ and $y = 10^{2j+1}$ for some non-negative integers $i \neq j$.
Without loss of generality, assume $i < j$.
Let $z = 10^j$.

- $xz = 10^{2i+1}10^j$. If $|xz| \not\equiv 0 \mod 3$, then $xz \notin L_c$. Otherwise, let u denote the prefix of xz with length $|xz|/3$ and let v denote the suffix of xz with length $|xz|/3$. We have $|xz| > 3i+3 = 3(i+1)$, so $|u| > i+1$. Also, $|xz| < 3j+3 = 3(j+1)$, so $|v| < j+1$. Therefore, $\#(1,u) \geq 1$ while $\#(1,v) = 0$, implying $u \neq v$ and $xz \notin L_c$.
- $yz = 10^{2j+1}10^j$. We have $|yz| = 3j+3 = 3(j+1)$. Otherwise, let u denote the prefix of yz with length $|yz|/3$ and let v denote the suffix of yz with length $|yz|/3$. We have $u = 10^j = v$, so $yz \in L_c$.

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

(d) Even-length strings whose second half contains an even number of 1s. More formally:

$$\{xy \in \Sigma^* \mid |x| = |y| \text{ and } \#(1,y) \text{ is even}\}$$

Solution: Call this language L_d.
Consider the set $F\left\{0^{2i+1}1 \;\middle|\; i \geq 0\right\}$.
Let x and y be arbitrary distinct strings in F.
Then $x = 0^{2i+1}1$ and $y = 0^{2j+1}1$ for some non-negative integers $i \neq j$.
Without loss of generality, assume $i < j$.
Let $z = 1^{2i+2}$.

- $xz = 0^{2i+1}1^{2i+3}$. We have $|xz| = 4i+4 = 2(2i+2)$ which is even. Let v denote the suffix of xz of length $|xz|/2 = 2i+2$. We have $\#(1,v) = 2i+2$ which is even, so $xz \in L_d$.
- $yz = 0^{2j+1}1^{2i+3}$. We have $|yz| = 2j+2i+4 = 2(j+i+2)$ which is even. Let v denote the suffix of yz of length $|yz|/2 > 2i+2$. We have $\#(1,v) = 2i+3$ which is odd, so $yz \notin L_d$.

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

2. For each of the following languages over the alphabet $\Sigma = \{0, 1\}$, either show that the language is regular (for example, by describing and justifying a DFA, NFA, regular expression, or regularity preserving construction from other regular languages) or prove that the language is not regular.

   (a) All strings with strictly more runs of 1s than runs of 0s.

   Solution: Call this language L_a. It is **regular** and represented by the regular expression $1^*(0^*1^*)^*$.

   Indeed, there must be a run of 0s between every run of 1s, so L_a is precisely the set of strings that begin with a run of 1s and follow every run of 0s with another run of 1s. ■

   Rubric: 2½ points = ½ for "regular" + 1 for regular expression or autonomaton + 1 for justification.

   (b) All strings with strictly more runs of 1s with length at least two than runs of 0s with length at least two.

   Solution: Call this language L_b. It is ***not* regular**.

   Consider the set $F \left\{(001)^i 0 \;\middle|\; i \geq 0\right\}$.

   Let x and y be arbitrary distinct strings in F.

   Then $x = (001)^i 0$ and $y = (001)^j 0$ for some non-negative integers $i \neq j$.

   Without loss of generality, assume $i < j$.

   Let $z = (110)^j$.

   - $xz = (001)^i 0 (110)^j$ which has i runs of 0s with length at least two and $j > i$ runs of 1s with length at least two, so $xz \in L_b$.
   - $yz = (001)^j 0 (110)^j$ which has j runs of 0s with length at least two and also j runs of 1s with length at least two, so $yz \notin L_b$.

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

   We conclude that F is a fooling set for L_b.

   Because F is infinite, L_b cannot be regular. ■

   Rubric: 2½ points ½ for "not regular" + standard fooling set rubric (scaled).

(c) All strings containing an equal but disjoint prefix and suffix of length at least 374. More formally:

$$\{xyx \in \Sigma^* \mid |x| \geq 374\}$$

Solution: Call this language L_c. It is ***not* regular**.

Consider the set $F\left\{10^i1 \;\middle|\; i \geq 372\right\}$.

Let x and y be arbitrary distinct strings in F.

Then $x = 10^i1$ and $y = 10^j1$ for some integers $i, j \geq 372$ where $i \neq j$.

Let $z = 10^i1$.

- $xz = (10^i1)(10^i1) \in L_c$.
- $yz = 10^j110^i1$. Every prefix of L_c of length at least 374 begins with a 1, and every suffix of length at least 374 ends with a 1. The only pair of disjoint prefix and suffix beginning and ending with a 1 are 10^j1 and 10^i1, respectively. Because $j \neq i$, these two strings are not identical, so $yz \notin L_c$.

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

We conclude that F is a fooling set for L_b.

Because F is infinite, L_b cannot be regular. ■

Rubric: 2½ points ½ for "not regular" + standard fooling set rubric (scaled).

(d) All strings containing with a prefix of length at least 374 and its reversal as a disjoint suffix. More formally:

$$\left\{xyx^R \in \Sigma^* \;\middle|\; |x| \geq 374\right\}$$

Solution: Call this language L_d. It is **regular**.

Let Σ^{374} denote the set of strings of length exactly 374, and for any string v, let $L(v)$ denote the language represented by the regular expression $v(0+1)^*v^R$. We have

$$L_d = \bigcup_{v \in \Sigma^{374}} L(v).$$

Indeed, for any $v \in \Sigma^{374}$, each member of $L(v)$ is in L_d by definition of L_d. Conversely, consider any string of the form $w = xyx^R$ where $|x| \geq 374$, and let v be the prefix of x of length exactly 374. Then, $w = vzv^R$ for some string z.

■

Rubric: 2½ points = ½ for "regular" + 1 for construction + 1 for justification.

*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 $b \in \Gamma$:

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

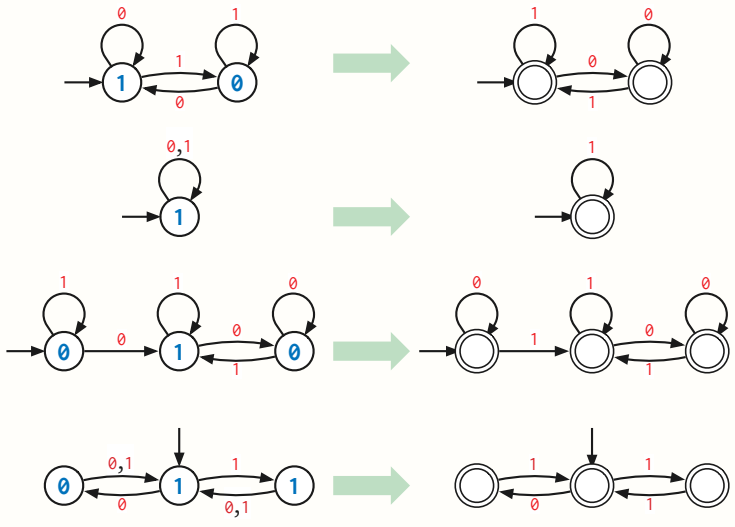
Less formally, we build M' from M by replacing every transition $p \xrightarrow{a} q$ with $p \xrightarrow{\omega(q)} q$, and then letting *every* state accept.

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:

$$\begin{aligned}
\Sigma' &= \Sigma \\
Q' &= Q \cup \{\mathit{fail}\} \\
s' &= s \\
A' &= Q
\end{aligned}$$

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

$$\delta'(\mathit{fail}, a) = \mathit{fail} \qquad \text{for all } a \in \Sigma$$

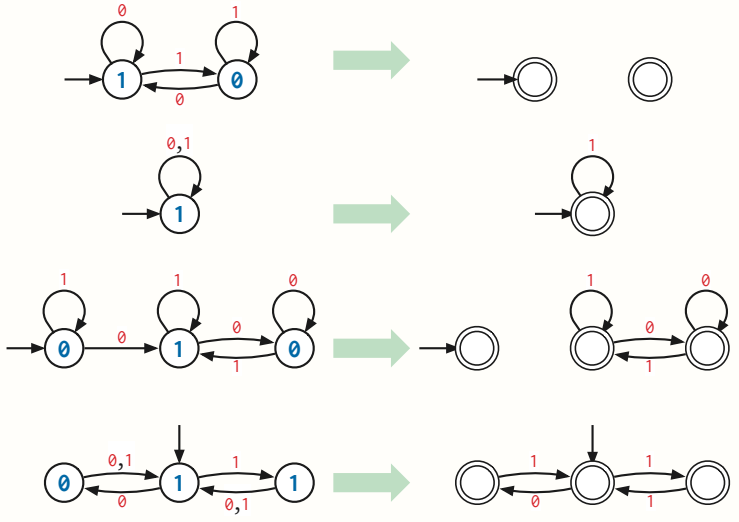
Less formally, we build M' from M by redirecting every transition $p \xrightarrow{a} q$ where $\omega(q) \neq a$ to a new fail state, and then letting every original state accept.

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



■