1. For each of the following languages over the alphabet $\Sigma = \{0, 1\}$, either prove that the language is regular (by constructing a DFA or regular expression) or prove that the language is not regular (using fooling sets). Recall that Σ^+ denotes the set of all nonempty strings over Σ.

   (a) $L_{1a} = \{xwwy | w, x, y \in \Sigma^+\}$

   Solution: L_{1a} is a regular language. The language only contains a limited number of strings that are not part of it. By the fact that any language of finite size is regular and regularity is preserved under complement. We can prove L_{1a} is regular.

   We can say that any string of length at least 4 is in the language. For an arbitrary string z of length at least 4, Let us define z as $xz'y$ where both a and b are a single symbol in Σ. So, z' has a of length at least 2.

   x and y are two non-empty strings which are covered by $((0+1)^+)$. z' will be ww. Since, our alphabet consists of just 0s and 1s, there are two cases. First, z' must be 00 or 11. This satisfies the constraint that there must be a repeating string in the middle with length at least one. Now, let us consider the case where we can have alternating 0's and 1's i.e, $w=$ 10 or 01. To include this case we add 1010 and 0101 to the regular expression.

   Putting it all together you get the regular expression for L_{1a}: $(0+1)^+(00+11+1010+0101)(0+1)^+$.

   Hence, the language L_{1a} is a regular language.

   Second, we know that the language is context-free simply because all regular languages are by definition context-free. ■

(b) $L_{1b} = \{xww^R x^R | w, x \in \Sigma^+\}$

Solution: Let us consider the fooling set $F = \{1^n 0^n | n > 0\}$

Let x and y be arbitrary strings in F.

Then $x = 1^i 0^i$ and $y = 1^j 0^j$ for some positive integers $i \neq j$.

Let $z = 0^i 1^i$.

Then $xz = 1^i 0^i 0^i 1^i \in L_{2d}$.

And $yz = 1^j 0^j 0^i 1^i \notin L_{2d}$, because $i \neq j$.

Thus, F is a fooling set for L_{2d}.

Because F is infinite, L_{1b} cannot be regular.

That being said, L_{1b} is context-free because we can construct a context free grammar that represents the above language. It is actually very simple:

$$S \to 0A0|1A1$$
$$A \to 0B0|1B1$$
$$B \to 0B0|1B1|\varepsilon$$

We have to have three terminals to ensure both x and w have at least one character. Because there is a CFG that represents this language, the language is context-free.

■

2. For any language A, let SkipFirstChar(A) = $\{w|aw \in A$ for some charcater $a \in \Sigma\}$. Show that the class of context-free languages is closed under the SkipFirstChar operation.

Solution: The purpose of this problem is to encourage you to think about transformations in a context greater than DFAs/NFAs.

In this case we want to prove the closure of a context-free grammar. So we need to transform the PDA that represents A into a PDA that represents SkipFirstChar(A). The idea behind this transformation is that we have 2 copies of the original PDA. We start in PDA 0 and for every transition that would normally burn a input character, we have that transition go to the second PDA that accepts the rest of the string as normal. We define this new PDA according to the following transformation:

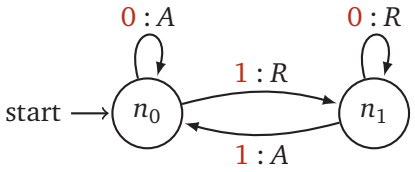
- Take the PDA P that represents A and copy it so that you have P_1 and P_2.
- For every transition $q_A, a \to q_B, b$ in P_1, we have a transition that goes from P_1 to P_2 but doesn't read a character from the string: $q_A, \varepsilon \to q_B, b$. Note that in this transformation, we are only making the transition that burn a character a go from P_0 to PDA_1.
- Have the accept state only in the states from P_2

We're just creating a new PDA(P^T) that skips one character from the input which is what we want. So the formal transformation looks like:

$$
\begin{aligned}
Q^T &= Q \times \{0,1\} \\
\Sigma^T &= \Sigma \\
\Gamma^T &= \Gamma \\
\delta^T(q_A^1, a) &= \{q_B^1, b | \{q_B, b\} = \delta(q_A, a)\} \text{ for} \\
\delta^T(q_A^0, \varepsilon) &= \{q_B^1, b | \{q_B, b\} = \delta(q_A, a)\} \\
s^T &= s^0 \\
A^T &= \{q^1 | q \in A\}
\end{aligned}
$$

Because there exists a PDA for this new language, $SKIPONECHAR(A)$ is context-free. ■

3. In a previous lab/homework we talked about a new machine called a *finite-state transducer* (FST). The special part thing about this type of machine is that it gives an output on the transition instead of the state that it is in. An example of a finite state transducer is as follows:



defined by the five tuple: $(\Sigma, \Gamma, Q, \delta, s)$. Let's constrain this machine (call is FST_{AR}) a bit and say the output alphabet consists of two signals: accept or reject ($\Gamma = \{A, R\}$). We say that $L(FST_{AR})$ represents the language consisting of all strings that end with a accept (A) output signal.

Prove that $L(FST_{AR})$ represents the class of regular languages.

Solution: The lovely thing about this problem is that it is harder the more you think about it and the key is simply to address it in small chunks. So let's break it down into parts. First let's define some notations about the machines we have available:

- DFA: $M = (Q_M, \Sigma_M, \delta_M, s_M, A_M)$
- NFA: $N = (Q_N, \Sigma_N, \delta_N, s_N, A_N)$
- FST_{AR} $F = (Q_F, \Sigma_F, \Gamma_F, \delta_F, s_F)$. For the transistion function let's say $\delta(q, a) = (q, b)$ where $a \in \Sigma$ and $b \in \Gamma$

OK so to prove a machine represents regular languages, we must show two things:

(a) A FST_{AR} can represent any regular language.

(b) Any language that is accepted by a FST_AR is regular.

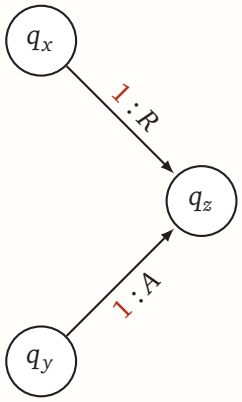
Part (a): A FST_{AR} can represent any regular language: Multiple ways to do this but the easiest is by using a language transformation like you did in lab. If we can show a construction that turns any DFA into a FST_{AR}, then we show that a FST_{AR} can represent any regular language.

Doing this is relatively straightforward. Accept in a DFA is ending in an accept state. Accept in a FST_{AR} is ending on an accept transition.

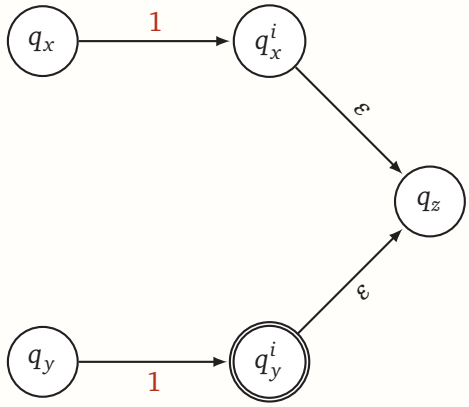
Hence, given a DFA (M), we'd like to construct a FST_AR (F) such that $L(M) = L(F)$. We define F as follows:

$$
\begin{aligned}
Q_F &= Q_M \\
\Sigma_F &= \Sigma_M \\
s_F &= s_M \\
\Gamma &= \{A, R\} \\
\delta_F(q, a) &= (\delta_M(q, a), A) && \text{if } \delta_M(q, a) \in A_M \\
&= (\delta_M(q, a), R) && \text{if } \delta_M(q, a) \notin A_M
\end{aligned}
$$

Part (b): Any language that is accepted by a FST_AR is regular. Little harder this one. The immediate impulse is to turn the FST_{AR} into a NFA directly which is a good impulse. But because you can have multiple transitions going to the same state with mixed accept/reject signals, it'll be tough to know what states to make an accept state and what to make a reject state:



So a bit of a problem since we can't just turn q_z into an accepting state. But what if we add intermediate states on the path of the transitions like so:



This NFA-part denotes the same thing as the FST-part above it. So let's define the NFA formally:

$$Q_N = Q_F \times \Sigma \cup Q_F$$ Original states plus state per transition
Added states marked as (q_F, a)
Original states marked as $(q_F, \square)$

$$\Sigma_N = \Sigma_F$$
$$s_N = s_F$$
$$\delta_N((q, \square), a) = (q, a)$$ Original transitions go to intermediate states
$$\delta_N((q, a), \varepsilon) = (\delta_F(q, a)[0], \square)$$ Intermediate states go to expected original states
$$A_N = (q, a) \text{ if } \delta_F(q, a)[1] = A$$
Add intermediate states that correspond to accepting transition

This construction shows that every language accepted by a FST_{AR} can also be accepted by an NFA.

Summation: Parts (a) and (b) together show that FST_{AR}'s represent the class of regular languages. ■

Alternate Problem

An all-NFA M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ that accepts $x \in \Sigma^*$ if **every** possible state that M could be in after reading input x is a state from F. Note, this is in contrast to an ordinary NFA that accepts a string if some state among these possible states is a an accept state. Prove that all-NFAs recognize the class of regular languages.

Solution: To solve this problem we need to look at it from two directions. The first is to show that, all-NFAs accept all regular languages. To show that, we can take any DFA D which accepts the regular language L. We know that D accepts every string in L and that there is exactly one path that it follows to the accepting state. So any DFA can be considered as an all-NFA, and hence all-NFAs accepts regular languages.

On the other side, to show that any language that is accepted by the all-NFA is regular, we first take an all-NFA $M = (Q, \Sigma, \delta, q_0, F)$ and construct a standard NFA $M' = (Q', \Sigma', \delta', q_0', F')$ that accepts the same language as M. Since we construct M' to have atmost a single path for every computation, we follow a process that is very similiar to the standard NFA to DFA construction, except for two differences in the construction.

First, for every dying path that M takes, there is an equivalent 'dying' path in M' as well. Second, if a string x ends at state $q' \in Q'$, it is accepted by M' if and only if ALL the states in q' belong to the accepting states of M. This is to ensure that M' accepts the string x only if M, upon reading the same string x, ends all it's branches in accepting states. Note that E(S) denotes all the states that could be reached from S using ε-transitions.

M':

$$Q' := P(Q)$$
$$q_0' := \{q_0\}$$
$$\delta'(R,a) := \begin{cases} \varnothing & \text{if } \delta(r,a) = \varnothing, \text{ for } r \in R, R \in Q' \\ \{q \in Q \mid q \in E(\delta(r,a)), \text{ for } r \in R, R \in Q'\} & \text{otherwise} \end{cases}$$
$$F' := P(F)$$

Since the standard NFA M' accepts the same language as the all-NFA M, any language that is accepted by M is regular. Hence we have proved that all-NFAs recognize the class of regular languages. ■

4. Prove this language is not regular by providing a fooling set. Be sure to include the fooling set you construct is i) infinite and ii) a valid fooling set.

$$L_{P5} = \{w | w \text{ such that } |w| = \lceil k\sqrt{k} \rceil, \text{ for some natural number} k\}$$

Hint: since this one is more difficult, we'll even give you a fooling set that works: try $F = \{0^{m^6} | m \geq 1\}$. We'll also provide a bound that can help: the difference between consecutive strings in the language, $\lceil (k+1)^{1.5} \rceil - \lceil k^{1.5} \rceil$, is bounded above and below as follows

$$1.5\sqrt{k} - 1 \leq \lceil (k+1)^{1.5} \rceil - \lceil k^{1.5} \rceil \leq 1.5\sqrt{k} + 3$$

All that's left is you need to carefully prove that F is a fooling set for L.

Solution: Let F be the set $\{0^{m^6} | m \in \mathbb{N}\}$.

We can also write this as $\{0^{\lceil k\sqrt{k} \rceil} | k = m^4, m \in \mathbb{N}\}$. Note that each element in F is also an element in L.

Let $x = 0^{m^6}$ and $y = 0^{n^6}$ for some $m < n$.

Let z be the smallest string such that $xz \in L$. By the given bound, $|z| \leq 1.5m^2 + 3$.

Suppose for contradiction $yz \in L$. By the other side of the given bound, we would need $|z| \geq 1.5n^2 - 1$. We can show both of these contraints on z can't be satisfied, since $1 \leq m \leq n-1$, so

$$1.5m^2+3 \leq 1.5(n-1)^2+3 = 1.5(n^2-2n+1)+3 = 1.5n^2-1+(5.5-3n) \leq 1.5n^2-1$$

.

■

Solution: From my experience in office hours, I wanted to write another solution which clarifies a few things (since this is a difficult problem).

First let's start with the fooling set $F = \{0^{m^6} | m \geq 1\}$. This set is a subset of the language $L_{P5} = \{0^{m^6} | m \in \mathbb{N}\}$ but that's ok for us. If we prove that F has infinite distinguishable states, then it means L_{P5} has at least infinite distinguishable states which is a problem for L_{P5} being regular.

So that's the big picture but how do we get there? Well first let's consider two strings from the fooling set:

$$x = 0^{i^6}$$
$$y = 0^{j^6}$$

for $i < j$. So both these strings are part of the original language (assuming $k = i^4 or k = j^4$). But what about the next string in their sequence? Is there another run of zeros (z) that you can add to x such that $xz \in L_{P5}$. More importantly if x and y are distinguishable then it means $yz \notin L_{P5}$? If $L_{GoforthScientificInc}$ is not regular, then we need to prove that such a z cannot exist which let's xz & $yz \in L_{P5}$.

So let's do a **Proof by Contradiction** as we do with most fooling set problems.

- First let's look at xz which is the next largest run of zeros after x that belongs to L_{P5}.
  - Looking at the definition for L_{P5}, in order for $x \in L_{P5}$, $k = i^4$ which give us the string $x = 0^{i^6} = 0^{(i^4)^{1.5}}$.
  - So the next largest run of 0's in L_{P5} occurs when $k = i^4 + 1$ which would give us the string $xz = 0^{(i^4+1)^{1.5}}$.
  - This means that we can finding the length of z by

$$|xz| - |x| = |0^{(i^4+1)^{1.5}}| - |0^{(i^4)^{1.5}}| = (i^4+1)^{1.5} - (i^4+1)^{1.5} = |z|$$

  - According to boundaries given in the problem this means that

$$1.5\sqrt{i^4} - 1 = 1.5i^2 - 1 \leq |z| \leq 1.5i^2 + 3 = 1.5\sqrt{i^4} + 3 \tag{1}$$

- Next, because of the proof by contradiction we're assuming $yz \in L_{P5}$ as well. This is the next largest run of zeros after y that is in L_{P5}. Here we follow the exact steps as above but with j instead of i.
  - Looking at the definition for L_{P5}, in order for $y \in L_{P5}$, $k = j^4$ which give us the string $y = 0^{j^6} = 0^{(j^4)^{1.5}}$.
  - The next largest run of 0's in L_{P5} occurs when $k = j^4 + 1$ which would give us the string $yz = 0^{(j^4+1)^{1.5}}$.
  - This means that we can finding the length of z by

$$|yz| - |y| = |0^{(j^4+1)^{1.5}}| - |0^{(j^4)^{1.5}}| = (j^4+1)^{1.5} - (j^4+1)^{1.5} = |z|$$

  - According to boundaries given in the problem this means that

$$1.5j^2 - 1 \leq |z| \leq 1.5j^2 + 3 \tag{2}$$

- So we got some boundaries for z defined by xz and yz shown below.

$1.5i^2 - 1$ $\quad$ $|z|$ according to (1) $\quad$ $1.5i^2 + 3$

$1.5j^2 - 1$ $\quad$ $|z|$ according to (2) $\quad$ $1.5j^2 + 3$

  Now if the states of x and y are not distinguishable (i.e. both xz and yz can be in $L_{P5)}$), then there should be some value of z that both prefixes can follow to an accept state. Namely,

$$1.5j^2 - 1 \leq |z| \leq 1.5i^2 + 3 \tag{3}$$

- But wait! Didn't we say $i < j$? If $i > 0$ then (3) is impossible!

- Therefore, there is run of zeroes for z where both xz and yz would be in L_{P5}.
- x and y denote distinguishable states states of the language L_{P5}.
- Because F is infinite, the DFA representing L_{P5} would require infinite states which violates the definition of regular language and hence, L_{P5} can't be regular.

■