Due to general confusion and incomplete proofs we’ve seen in the last few homeworks, we’ve decided to give you some useful notes and solutions. Below we cover:

1. Proving a language is regular [HW4 #8 US1.67].
2. What the pumping lemma is and how to use it to prove a language is nonregular [HW3 #5 US1.54].

We understand this course can be very abstract, mathy, and just all around difficult so we will try to give you guys a little more useful feedback. From now on, after each homework is submitted, we will open an online poll that allow you guys to vote for 2 problems to see solutions to. The solutions to the problems with the 2 most votes will be posted to the course website. From the past homeworks, we chose two problems below to demonstrate what a formal proof of each type should look like.

**Proving a Language is Regular**

Since we proved that NFAs, DFAs, and regular expressions are all equivalent, if we can construct any of these to accept a language A, then we have proved that A is regular. **Remember please follow the construction proof format!**

Given
Name set variables, machines, etc

Want to Show (WTS)
State goal and outline plan

Construction
Using objects previously defined new tools, work toward the goal. Give formal definitions and explain.

Correctness
Here, if you constructed a machine M, you must prove that it actually accepts a language L. You have to (1) show that M accepts all strings in language L (2) show that M rejects all strings not in language L.

Conclusion
Recap what you proved.
HW4 #8 (US1.67) Solution

Given
\[ RC(A) = \{ yx | xy \in A \} \]

WTS
The class of regular languages is closed under operation RC.
So, show that if A is regular, then a regular machine accepts RC(A).
Essentially, show \( M'(RC(A)) \) exists where \( M' \) is a machine that recognizes the language RC(A).

Construction
Let \( M \) recognize A where \( M = (Q, \Sigma, \delta, q_0, F) \).
We will now construct a regular machine \( M' = (Q', \Sigma, \delta', q_{0'}, F') \) that recognizes RC(A).

High Level Intuition
Since we don’t know where the rotational point (RP) will be, we will nondeterministically guess where RP is at every possible position. We will use copies of M to read strings x and y.

We know for the new machine we’re reading in y first in the input string yx and we know we are somehow using M (while keeping M’s structure the same). We need to guess which node in M we land on right before the start of y (as if we read in x first and \( q^{Xend} \) is the node we are at before starting to read y). To make this guess, we create a new start state that has epsilon transitions to every state in M so that this new machine starts reading an arbitrary y at all the correct \( q^{Xend} \).

Next, our machine reads y starting at \( q^{Xend} \). We know that our new machine is done reading y in some state in F since M recognizes xy \( \in \) A. Now, we need our new machine to read x and cycle back to accept yx at the state \( q^{Xend} \). To cycle back, we can have epsilon transitions from the states in F to the start state of M while making all states in F non-final states (since we accept on x and not y in the new machine). However, notice we can’t cycle back on the same machine because by making \( q^{Xend} \) accept on the same machine will allow our machine to continue looping and accept \( (yx)^* \). Thus, we know for each arbitrary RP, we need 2 machines (2 copies of M): (M1’) to read y and have F states point to the start state of M2 to read x (M2’) to read x and accept at the end of x or \( q^{Xend} \).

We covered how to construct the machine that recognizes yz for a chosen RP (need 2 copies of M). How do we exhaustively cover all RPs? Remember that \( q^{Xend} \), which is the accept state for M2’, needs to be different for each RP chosen. So, to cover all RPs, we need an M1’ and M2’ for each and every node in M. Effectively, this means our new machine will have \( 2|Q|^2 + 1 \) states. In other words, for every node (each representing a RP) in M, our new machine M’ will have 2 copies of M: the 1st to read in y starting at RP and the 2nd to read x and end at the \( q^{Xend} \) for the corresponding RP.

Actual Construction
That was just the explanation. Now we need to formalize it using math and the formal definitions of automata.

Creating the M1 and M2 copies:
Let \( M1^1 = (Q_1^1, \Sigma, \delta_1^1, q_{01}^1, F1^1) \),
\( M1^2 = (Q_1^2, \Sigma, \delta_1^2, q_{01}^2, F1^2) \),
\( \ldots \)
\( M1^{|Q|} = (Q_1^{|Q|}, \Sigma, \delta_1^{|Q|}, q_{01}^{|Q|}, F1^{|Q|}) \),
\( M2^1 = (Q_2^1, \Sigma, \delta_2^1, q_{02}^1, F2^1) \),
\( \ldots \)
\( M2^{|Q|} = (Q_2^{|Q|}, \Sigma, \delta_2^{|Q|}, q_{02}^{|Q|}, F2^{|Q|}) \)

For all \( q \in 1..|Q| \):
Recap above: we constructed all needed copies of M1 and M2 that each have their own unique state names. However, their structures are all identical.

TODO next: we need to put each of them in their respective place in M', connect each respective M1 final node to an M2 start node, and also remove all accept states in M1 machines and make the respective qXend node in all M2s the accept state.

Now putting it all together for M':

Defining q0': Let q0' be the new starting node for machine M'.

Defining Q' = {q0'} ∪ Q1q ∪ Q2q for all q ∈ 1..|Q|

Defining δ': for all q ∈ 1..|Q| δ'(s ∈ Q1q, a ∈ Σ) = δ1q(s, a) and δ'(s ∈ Q2q, a ∈ Σ) = δ2q(s, a)

Connect all M1q final states to M2q start states: δ1q(f ∈ F1q, ε) = {q02q}

Connect new start state to each of M1q's respective start states:

δ'(q0', ε) = {s1q|s is the qth state in Q}

Defining F': for all q ∈ 1..|Q|

Make all the respective qXend nodes the end states for all M2q.

F2q = {s2q|s is the qth node in Q}

Finally, F' = F21 ∪ F22 ∪ ... ∪ F2|Q|

Thus, we have completed the construction of machine M', now we have to prove it actually recognizes RC(A).

**Correctness**

[1] Proving strings in the language are accepted by M'.

Given an arbitrary string w = w1w2...wcwc+1...wn ∈ RC(A) and y = wc+1...wn, since our M1q machines are copies of M({w|w = xy ∈ A}), we can count our machine to nondeterministically guess the correct state to start at when reading in y for any arbitrary c. This M1q machine would follow the set of states S1,S2,...Sc when reading y and land on Sc ∈ F1q. Sc would then take the epsilon transition to the corresponding M2q start state and follow the state path Sc+1...Sn when reading x. Upon completion, since we started reading the beginning of y at the corresponding state at which x is hypothetically read first S1 in M1q, then we know for certain we would land on that same corresponding state except in M2q which would be accept state: Sn ∈ F'.

[2] Proving strings not in the language are rejected by M'.

We look at the case where xy ∈ A, we must show that yz is not accepted by M'. An arbitrary string is given w = w1w2...wcwc+1...wn ∈ RC(A) where y = wc...wn (c is arbitrary), and xy ∉ A.

Case 1: y doesn’t land on a respective accept state in M.

If this is the case, then there is no way for the string to transition from M1q to M2q since the epsilon transitions for this transfer are in M1q that correspond to the respective end states in M.
an epsilon transition that actually takes it to a M end state: $S_1S_2..S_c$ where $S_c \in F^q$. However, since $xy \notin A$, then we for certain that x will not transition us to an accept state in $M^q$ or else it would be true that $xy \in A$. So, we know for certain that x will take us through the state sequence $S_{c+1}..S_n$ where $S_n \notin F'$.

**Conclusion**

By constructing a regular machine $M'$, we’ve proven that the class of regular languages is closed under the RC operation.

**Pumping Lemma**

Here’s a summary of what the pumping lemma says for a regular language $L$:

$$\forall s( s \in L \land |s| \geq p ) \implies \exists xyz[ s = xyz \land |y| > 0 \land |xy| \leq p \land \forall i(xy^iz \in L)]$$

To use this statement to prove a contradiction, let’s look at the negation:

$$\exists s( s \in L \land |s| \geq p ) \land \forall xyz[ (s = xyz \land |y| > 0 \land |xy| \leq p ) \implies (xy^iz \notin L)]$$

Notice it went from saying $\exists p, \forall s, \exists xyz, \forall i, xy^iz \in L$ to $\forall p, \exists s, \forall xyz, \exists i, xy^iz \notin L$. So, for our contradiction, we just need to show that for some string $s$ we choose that is long enough, no matter how we partition it (for all possible, valid partitions), we can find an $i$ that proves each of the possible partitions is not pumpable ($xy^iz$ isn’t in $L$). This is the essence of the proof. Also, don’t forget to explicitly mention that the criteria for choosing $s$ is met: (1) in language $L$ (2) at least as long as pumping length $p$.

**Please remember to use the proof format for contradictions:**

*Given*

Statements, set variable names, machines, etc

*WTS*

State goal and outline plan

*Contradiction*

If you’re trying to prove $P \rightarrow Q$:

(1) Assume that $P$ is true.

(2) Assume that not $Q$ is true.

(3) Use the above to demonstrate a contradiction.

*Conclusion*

Recap what you proved.

**HW3 #5A (US1.54) Solution**

*Given*

$F = \{a^ib^jc^k | i, j, k \geq 0 \text{ and if } i = 1, \text{ then } j = k\}$

*WTS*

Show that $F$ is not regular.

We can find a contradiction in the following:

Assuming $F$ is regular, then we show that a valid string in $F$ that should be pumpable actually isn’t.

Given a string $S$ where (1) $S \in F$ and (2) $|S| \geq p$, for all legal partitions $xyz$ of $S$, we can find a pumping enumeration $i$ where $xy^iz \notin F$. 
Contradiction
Assume toward a proof contradiction that $F$ is regular.
In the previous homework, we proved that a Regular language is closed under reversal. Therefore, if $F$ is Regular, $F^R$ is also regular.

$$F^R = \{c^i b^j a | i, j, k \geq 0 \text{ and if } i = 1, \text{ then } j = k \}$$

We choose a string to be $s = c^p b^p a$

$s \in F^R \checkmark$

$|s| = 2p + 1 > p \checkmark$

Now for partitions.
Let $x = c^j, y = c^k, z = c^l b^p a$ where $j + k + l = p$, and $k > 0$ (from $|y| > 0$ constraint)
This covers all legal partitions for $s$.

Now, for all legal partitions, we pump $y$ 0 times: $xy^0 z = c^j c^l b^p a$. Here, $j + l \neq p$ since $k > 0$. So, $xy^0 z \notin F^R$ and we obtain a contradiction to the following pumping lemma statement:
for $n \geq 0, xy^n z \in L$ thus $F^R$ is not regular.

Since $F^R$ is not regular, $F$ is also not regular.

Conclusion
By using the knowledge that regular languages are closed under reversal and showing that $F^R$ is non-regular through a contradiction in the pumping lemma, we’ve shown the $F$ is also nonregular.