Exercise 4.2.6

Show that the regular languages are closed under the operations below. For each, we'll start with L and apply operations under which regular languages are closed (homomorphisms, intersection, set difference) to get the desired language.

a) $min(L) = \{w \mid w \text{ is in } L, \text{ but no proper prefix of } w \text{ is in } L\}$

Describe the strings which are ineligible for $min(L)$ and exclude them using set difference. The ineligible strings are $L\Sigma^+$, since $w \in L\Sigma^+$ means that $w = xy$ where $x \in L$. That is, w has a proper prefix x which is in L.

$$min(L) = L - L\Sigma^+$$

Comments: many students submitted the (correct) answer $L - (L \cap L\Sigma^+)$. This probably looks more natural, since $L \cap L\Sigma^+$ is a subset of L, and $L\Sigma^+$ is not (in general). However, recall the definition of set difference: $L - M = L \cap \overline{M}$. Think of this not as removing M from L, but *excluding* M from L, since M need not be a subset of L.

Also, many students submitted a solution using homomorphisms, which were not necessary. It should be a red flag if you define a pair of homomorphisms that just add and remove hats without otherwise altering the string. This means that your set intersection could be done in the original alphabet Σ.

b) $max(L) = \{w \mid w \text{ is in } L \text{ and for no } x \text{ other than } \epsilon \text{ is } wx \text{ in } L\}$

Again, describe strings which are ineligible for $max(L)$, then remove them with set difference. Use homomorphisms to describe all the strings that are *prefixes* of strings in L. Here is the idea: use an inverse homomorphism h^{-1} on L to mark arbitrary symbols with hats, apply set intersection to force the hats to the end, then apply a homomorphism g to delete hatted symbols. For simplicity, assume the alphabet is $\{a, b\}$ (the approach works for any alphabet).

$$\begin{aligned} h(a) &= a & g(a) &= a \\ h(\hat{a}) &= a & g(\hat{a}) &= \epsilon \\ h(b) &= b & g(b) &= b \\ h(\hat{b}) &= b & g(\hat{b}) &= \epsilon \end{aligned}$$

Then $(a + b)^*(\hat{a} + \hat{b})^+$ is the expression describing the strings with hats at the end. Our final expression is:

$$max(L) = L - g(h^{-1}(L) \cap (a + b)^*(\hat{a} + \hat{b})^+)$$

c) $init(L) = \{w \mid \text{for some } x,\ wx \text{ is in } L\}$

Note that this is almost exactly the set of strings we declared ineligible for $max(L)$, except that in $init(L)$, x may be ϵ. Use the same homomorphisms h and g, but modify E to allow omission of the hatted portion:

$$init(L) = g(h^{-1}(L) \cap (a+b)^*(\hat{a}+\hat{b})^*)$$

Prove that alt(L,M) is regular provided that L and M are regular languages

To prove this it is sufficient to show that we can convert L and M to $alt(L, M)$ via operations that preserve regularity (in our case homomorphism, inverse homomorphism and intersection with regular sets).

This is actually a relatively trivial extension of the shuffle homomorphism discussed in class. First assume that L and M are in different alphabets, we will call them Σ^L and Σ^M. If they are not, we can convert one trivially to another alphabet with a homomorphism, and then back to the original alphabet with inverse homomorphism. Denote the nth symbol in one of these alphabets as Σ^n.

First, we will define two homomorphisms, f and g as follows:

$$f(\Sigma_L^n) = \Sigma_L^n$$

$$f(\Sigma_M^n) = \epsilon$$

$$g(\Sigma_M^n) = \Sigma_M^n$$

$$g(\Sigma_L^n) = \epsilon$$

Now we define alt(L,M) as:

$$alt(L, M) = f^{-1}(L) \cap g^{-1}(M) \cap (\{a|a \in \Sigma_L\}\{b|b \in \Sigma_M\})^*$$

As you can see, without the regular expression, this is simply shuffle. The regular expression ensures that each homomorphism only contributes one character from the opposite alphabet from any epsilon production, and thus that we actually alternate between each.

Despite it's initial appearance the expression $(\{a|a \in \Sigma_L\}\{b|b \in \Sigma_M\})^*$ actually is regular. If we knew the alphabets of and L and M explicitly we could write out each of the $a \in \Sigma$ terms as a disjunction of the entire alphabet.

We do not need to worry about anything but even length strings because the definition of alt says that each string must be the same length.

Course 381 & 481
Homework 5
Problem 3 [Exercise 4.2.11 in book]

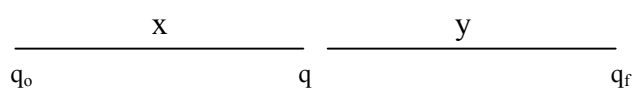
4.2.11:
Show that the regular languages are closed under the following operation:

cycle(L) = {w | we can write *w* as *w = xy,* such that *yx* is in L*}*.

For example, if L = {01, 011}, then *cycle(L)* = {01, 10, 011, 110, 101}. Hint: Start with a DFA for L and construct an ε -NFA for cycle(L).

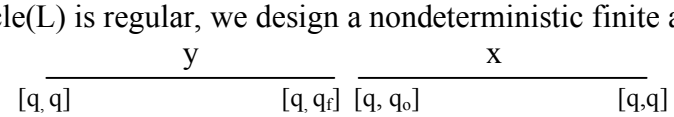
Set up the problem:
For a regular language L, assume we have the following deterministic finite automata:

We can also write *w* as:



Let q_o be the initial point/start state, q_f be the final point/accepting state, and *q* be any point in between of string *w*.

To show that cycle(L) is regular, we design a nondeterministic finite automata as follows:



This automata has states that consists of two parts: the first element is the point q in the string at which the FA starts and the second state keeps track of where the FA is in the string. The FA will go through the y portion of the string and then start guessing for the x part of the string because of its nondeterminism. This FA moves exactly like the former automata but its only difference is that it accepts when the first element matches up with the second element. This is to say that the FA will have completed a cycle by coming back to its start state before it accepts. Hence, regular languages are closed under *cycle*.

The transition function for the second automata is:

$$\delta_2([p,q], a) = [p, \delta_1(q, a)]$$

where subscripts 1 and 2 denote the first and second FA, respectively. *p* is the start state, *q* is the current state, $a \in \sum^*$.

Note: If students do draw the ε -NFA, the epsilon moves go from the start state to any point in the string because any point in the string can be the start of the cycle.

4.2.13

a) Start out by complementing this language. The result is the language consisting of all strings of 0's and 1's that are *not* in 0*1*, plus the strings in L_{0n1n}. If we intersect with 0*1*, the result is exactly L_{0n1n}. Since complementation and intersection with a regular set preserve regularity, if the given language were regular then so would be L_{0n1n}. Since we know the latter is false, we conclude the given language is not regular.

b) Intersect this language with the regular expression 0*2*. This results in the language $\{0^n2^n \mid n >= 0\}$. Use a homomorphism with h(2) = 1 and this results in $L_{0n1n.}$ Since intersection with a regular set and homomorphism preserve regularity, we know that the language is not regular.