CS350 – Regular Languages Properties

Purpose: Since regular languages appear everywhere in computer science it is worth spending some time examining the properties and limits of these languages. The purpose of this module is for you to prove some properties of regular languages and also prove that some languages are not regular.

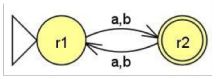
Knowledge: This module will help you become familiar with the following content knowledge:

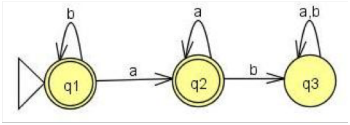
- How to use prove closure properties of regular languages.
- How to use the pumping lemma to show that a language is not regular.

Activity: With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 90 minutes.

1. Show that a regular language must be decidable. (This should be easy!)
2. Show the that regular languages are closed under each of the following operations. (This should be be easy too!)
   (a) Complement
   (b) Concatenation
   (c) Union
   (d) Kleene star

3. It is going to take a bit more work to show that regular languages are closed under intersection. A good place to start is by looking at an example of how we can run two dfa's in parallel on a string and see if they both end up accepting. If so, that string is in the intersection. To run the dfa's in parallel we create another dfa whose states are pairs containing the state of the first dfa and the state of the second dfa. The states and a transition table for the example is given below:





	a	b
(r1,q1)	(r2,q2)	(r2,q1)
(r1,q2)	(r2,q2)	(r2,q3)
(r1,q3)	(r2,q3)	(r2,q3)
(r2,q1)	(r1,q2)	(r1,q1)
(r2,q2)	(r1,q2)	(r1,q3)
(r2,q3)	(r1,q3)	(r1,q3)

What would be the start state of the new dfa in the example? How would we get the start state in general?

What would be the accepting states of the new dfa in the example? How would we get the accepting states in general?

How would we create the transition arrows in general?

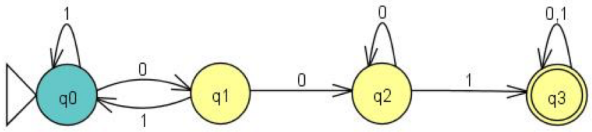
How does the general version of this procedure show that regular languages are closed under intersection?

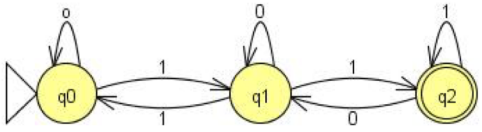
4. Find the flaw(s) in the following:

$A = \{a^n \mid n \geq 0\}$ is a regular language since we can construct a dfa for it. So is $B = \{b^n \mid n \geq 0\}$. Since regular languages are closed under concatenation, the language $\{a^n b^n \mid n \geq 0\}$ must also be regular.

5. It turns out that $\{a^n b^n \mid n \geq 0\}$ is not a regular language. We need some tools to be able to prove that. We will start by considering properties of dfa's that contain cycles (which is most of them!).

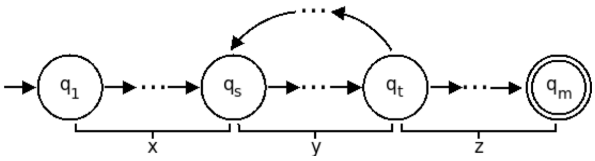
   For each of the following dfa's, what is the longest string each dfa can accept without visiting any state more than once?





6. With cycles in mind, we can consider a property of infinite regular languages. Every infinite regular language is accepted by a dfa with some number of states represented by m. If we accept a string of length $\geq m$ then what can we conclude about the path through the dfa? Is it possible for that path to not visit any state more than once?

   So every string in the language can be broken into three parts x, y, and z. The part x is up to the first cycle. Why is the length of xy always $\leq m$? Why is the length of y always > 0?



7. Consider the language $\{a^n b^n \mid n \geq 0\}$. Assume that this language is accepted by a dfa with m states. The path using the string $a^m b^m$ must include a cycle. Where in the string must the first cycle occur? Why?

   What happens if we go through the cycle more than once? Why does this give a contradiction on the strings that will be accepted by this dfa?

8. The property that infinite regular languages must have a cycle within the first m symbols of any input string is called the "Pumping Lemma". This is because we can "pump" through the cycle as many times as we want to get other strings that will also be accepted. The pumping lemma is used to show that languages are *not* regular by getting a contradiction.

   The steps of a pumping lemma proof:

   (a) Assume the language L is regular and therefore accepted by some dfa with m states.

   (b) Choose a string w in the language of length greater than m (so there must be a cycle).

   (c) The pumping lemma guarantees w can be divided into parts xyz such that for any $i \geq 0$, $xy^i z$ is also in L, and that $|y| > 0$ and $|xy| \leq m$.

   (d) Choose a value i such that pumping that many times gives you a string not in the language. This gives a contradiction so L is not regular.

   For some practice with the pumping lemma, complete the following proof that $L = \{0^n 1^n \mid n \geq 0\}$is not regular. Give possible values of w and i which will make this work (multiple answers are possible). Remember you need to choose a w of length longer than m so the string will usually involve m in some way.

   Assume (towards a contradiction) that L is regular. Then the pumping lemma applies to L. Let m be the pumping length. Choose w to be _____. The pumping lemma guarantees w can be divided into parts xyz such that for any $i \geq 0$, $xy^i z$ is also in L, and that $|y| > 0$ and $|xy| \leq m$. But if we let $i =$_____, we get the string _____which is not in L, a contradiction. Therefore the assumption is false and L is not regular.

9. For the language $L = \{0^a 1^b 0^a \mid a, b \geq 0\}$, give the possible values of w and i which will give a contradiction for the pumping lemma. Explain why you get a contradiction.

10. For the language $L = \{w_1 w_1^R \mid w_1$ is a string of 0's and 1's and w_1^R is the reverse of $w_1\}$, give possible values of w and i which will give a contradiction for the pumping lemma. Explain why you get a contradiction.
    Hint: Choose your w wisely so that the contradiction is obvious after pumping. Not every string in the language will work.

11. The pumping lemma can be used to show that $L = \{0^{n!} \mid n \geq 0\}$ is not regular. For the pumping lemma, a choice of $w = 0^{m!}$ and a choice of $i = 2$ will lead to a contradiction. Suppose that the length of the cycle, y is k. So the extra pump of the string $w = xyz$ will result in xy^2z. Explain why the length of this new string can not be a factorial value.
    Hint: Why is that length bigger than $m!$ but smaller than $(m+1)!$?

Complete the following assignments for grading. Each should be done individually but you may consult with a classmate to discuss strategies.

Assignment 1:
Using the closure properties that we proved in class, show that regular languages are closed under the symmetric difference operator $\ominus$. The symmetric difference of languages L_1 and L_2 is defined as

$$L_1 \ominus L_2 = \{x \mid x \in L_1 \text{ or } x \in L_2 \text{ but } x \text{ is not in both } L_1 \text{ and } L_2\}$$

Criteria for Success: You have a clear proof on why the symmetric difference of two regular languages is also regular.

Assignment 2:
Using the closure properties that we proved in class, show that if L is a regular language then so is

$$L_1 = \{uv \mid u \in L \text{ and } |v| = 2\}$$

Criteria for Success: You have a clear proof on why L_1 is also regular.

Assignment 3:
Use the pumping lemma to show each of the following languages is not regular.

1. $L = \{a^n b^m \mid n \geq m\}$
2. $L = \{ww \mid w \in \{a, b\}^*\}$
3. $L = \{a^n \mid n = k^3 \text{ for some } k \geq 0\}$

Criteria for Success: You have a full pumping lemma proof for each language.

Assignment 4:
For each of the languages, either prove that it is regular (using the techniques we already know) or using the pumping lemma to show it is not regular.

1. $L = \{a^n b^m \mid n \leq m \leq 2n\}$
2. $L = \{a^n b^m \mid n \geq 100, m \leq 100\}$

Criteria for Success: You have a proof for each language.

Submit your answers in Canvas for grading.