Thomas DeMasse - 8/13/20203 - Foundations Final

(1) The answer is no:

We can use the pumping lemma to prove that the language is not regular.
First we assume that the language is regular.
All reguler languages will have a pumping length P. We can take this and pump our language $L_1 = 01^P0^P1$. Given a condition of the pumping lemma that states $|xy| \leq P$. we know that what is going to be pumped is within the first P elements.
Given this we can pump 1's as it falls in the first P elements. By doing this we would get a string that looks as follows: $L_1 = 011111...01$. This string is not within our language because it contains more 1s than 0s. And once one string has been disproven, it is a proof by contradiction, therefore the language is not regular.

(2) The answer is no!

We can use the pumping lemma to prove that the language is not regular. First we can assume the language is regular. With this there must be a pumping length P for all reguler languages. with this we can set the language equal to w such that

$W = 1^p01^p0^p10^p$. We can split the string into parts xyz. $W = \underbrace{1^p01^p}_{xy}\underbrace{0^p10^p}_{z}$ where y is where

We apply our pumping constant. We can rewrite W to match with the conditions of the pumping lemma based on the split above. With a value o $P=2$ we would get the following $1^2 0 1^2 0^2 1 0^2 = 1101100100$. If we split this into parts xyz from above we can have as follows $\underbrace{110}_{x}\underbrace{11}_{y}\underbrace{00100}_{z}$, when taken into the form xyz

Where we let $i=2$, our resulting string would be as follows; $W = 110111100100$. This string is not in our language because it contains more 1's than 0s and based on the language the 1's and 0s should be equal which creates a contradiction, proving the language is not regular. Since a contradiction is generated on the first of three conditions, we don't need to test the others.

(3) The answer is Yes!

We can prove that L_1 is a CFL by use of the pumping lemma. Given that for CFL's the process is more or less the same as regular languages such that there exists a pumping length constant P. We can take L_1 and make W such that $W = 01^P 0^P 1 : P \geq 0$. For CFL's we have the rules of a.) $|vy| \geq 1$ and b.) $|vxy| \leq P$. We can take W and split it into parts such that:

$\underbrace{01}_{u}\underbrace{\ldots}_{v}\underbrace{10}_{x}\underbrace{\ldots}_{y}\underbrace{01}_{z}$ with this our $|vxy|$ is within P.

Also in v contains all 0s and in y all 1s. Because of this when we apply an arbitrary number for P such as P=3 we would get $W = 01^3 0^3 1$ $= 0111 0001$ where $\underbrace{0}_{u}\underbrace{111}_{v}\underbrace{0}_{x}\underbrace{00}_{y}\underbrace{1}_{z}$ there exists the same number of 0s as there are 1s. Therefore L_1 is a CFL.

4.) The answer is no!

We can prove that L_2 is not a CFL by use of the pumping lemma. If we take the rules from the question above such that $W \in L$ where $|vy| \geq 1$ and $|vxy| \leq P$. If we assume that L_2 is a CFL then there must exist a pumping length P. We can then make $W = 1^P 0 1^P 0^P 1 0^P$ and then we can divide W into parts $uvxyz$ as such: If we pick our pumping length to be an arbitrary value such as P=5 we would get $W = 1^5 0 1^5 0^5 1 0^5$.

We can split the String into parts UVXYZ into a couple different cases, one wher V and Y only contain one value and one where they contain both 0 and 1.

Case 1: V contains only 0s. Y only contains 1s

$w = 1^5 0 1^5 0^5 1 0^5 = 11111011111000001000000$

We can split it as such: $\underbrace{11111011}_{U}\underbrace{111}_{V}\underbrace{0}_{X}\underbrace{000}_{Y}\underbrace{0100000}_{Z}$

We can now take this and check it on uv^ixy^iz to make sure it is in the language L_1 for every $i>0$ in this case we will let $i=3$. Such that $uv^3xy^3z =$

$111110111111111111000000000000100000$

This then results in a final value as such $L_2 = 1^5 0 1^{13} 0^{11} 1 0^5$ which is not in the language.

Case 2: V or Y contain both 0s and 1s

$W = 1^5 0 1^5 0^5 1 0^5 = 11111011111000001000000$

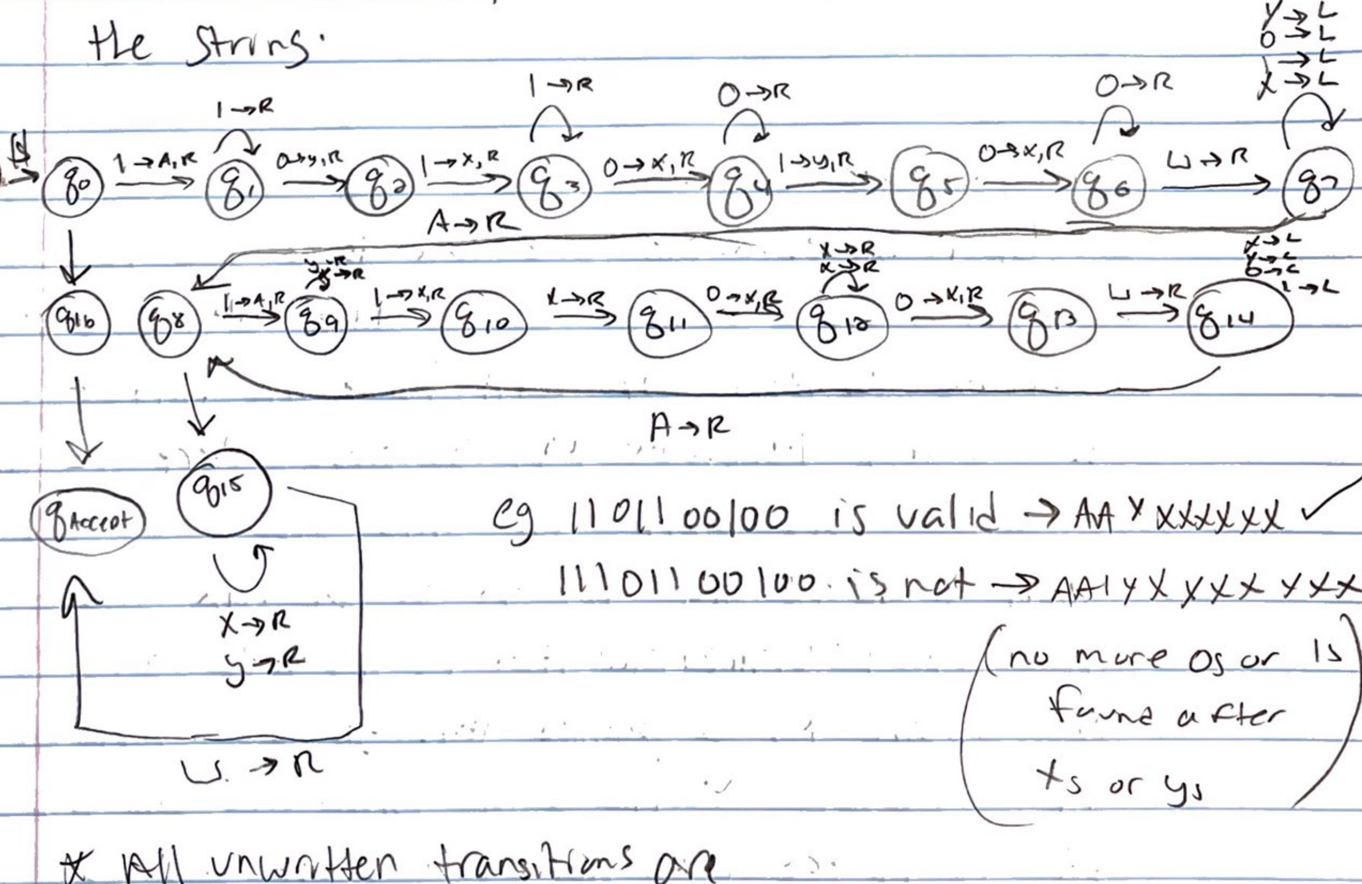
Split such as: $\underbrace{1111}_{U}\underbrace{101}_{V}\underbrace{1}_{X}\underbrace{111000}_{Y}\underbrace{00100000}_{Z}$

We must then again check it against uv^ixy^iz where it is in L_2 for all $i>0$ let $i=3$ again such that uv^3xy^3z

$= 111110110110111100000000000100000.$

This results in $1^5 0 1^2 0 1^2 0 1^4 0^{11} 1 0^5$ which is not in the language L_2. By these two cases it proves that L_2 is not a CFL by proof of Contradiction.

(5) L_2 is decidable. The following Turing machine is able to take an input string and determine whether or not it is in the language L_2. There is an algorithm that is performed beforehand to shift all values down, and add a \$ to the front of the strings.



eg 1101100100 is valid → AAYXXXXXX ✓

11101100100 is not → AA1YXXXXYXX

(no more 0s or 1s found after xs or ys)

* All unwritten transitions are considered reject states *

(6) The answer is yes:

Because L_2 can be proven to be decidable by a Turing machine, it belongs in the set of recursive languages. From this, because L_2 is a recursive language, and a recursive language is a subset of the recursive enumerable languages, L_2 is also a part of the set of recursive enumerable languages. Therefore yes $L_2 \in E_1$

(7) The answer is no:

L_1 is undecidable. We can take the halting problem and reduce it to the blank-tape halting problem. We will have an input of a Turing Machine M. Suppose there is an algorithm that exists for this problem. We will construct one for this. We can assume that the Machine/Algorithm is as follows.
Our input M is passed through our Algorithm, if it accepts M halts on blank/empty tape. If it rejects, M does not halt. We can then design a machine where Input M and string w. When we pass this through the algorithm, if it accepts M halts on w, if reject it does not halt on w. Taking this we can construct

a new Turing Machine M_w that we will write on a blank tape, then continue like that of TM M. With this M will halt on input w if and only if M_w halts when started with a blank tape.. By constructing this algorithm an Turing Machine and given that we know that the halting problem is undecidable, the blank-tape halting problem is also undecidable. Therefore, the language L is undecidable.