Aughdon Breslin and Isabella Cruz
11/6/20
"I pledge my honor that I have abided by the Stevens Honor System."

Problem 1.
(15 points) Prove, using the pumping lemma for context-free languages, that the language of all palindromes over the alphabet {0,1} in which the numbers of 0s and 1s are equal, is not context-free. Note: We will grade this problem very closely, so make sure that your argument is complete and that no details are left implicit. The problem is not hard, the reason for this exercise is for you to write a complete and precise proof.

L = $\{ww^R : w \text{ in } \{0,1\}^*\}$, NOT 10101, NOT 101101
Note: '' represent parts of the string being pumped and are just for organizational purposes. They are not part of the actual string

A:
Assume the language of all palindromes over the alphabet {0,1} in which the numbers of 0s and 1s are equal, L, is context-free.
By the pumping lemma, L has a pumping length p.

Let $s = 0^p1^{2p}0^p \in A$,
s = uvxyz, where |vy| > 0, |vxy| <= p.
vxy must contain either: all zeroes, all ones, zeroes then ones, or ones then zeroes

<u>All zeroes</u>
0...0 1.....1 0...0
p 2p p
If vxy = 0^k, where 1 <= k <= p, then pumping the string up would result in an unbalanced number of 0's on either side of the string ($0^{p+k}1^{2p}0^p$ or $0^p1^{2p}0^{p+k}$), breaking the ruleset and thus violating the pumping lemma.
Ex: v = 0, uvvxyyzz -> '00' 110

<u>Combination of ones and zeroes</u>
0...0 1....1 0...0
p 2p p

If $v = 0^j$, and $y = 1^k$ or reversely, $v = 1^k$, and $y = 0^j$ where |vy| > 0 and |vxy| <= p, then pumping the string up would result in more 0s or 1s on the pumped side of the string than 0s or 1s on the other side ($0^{p+j}1^{2p+k}0^p$ or $0^p1^{2p+k}0^{p+j}$) and would no longer be a palindrome. This breaks the ruleset and thus violates the pumping lemma.

Ex: v = 0, y = 1, uvvxyyz -> '00' '11' 10

If either v or y follows a pattern of 0^j1^k or 1^k0^j where j,k >=1 and |vxy| <= p, then pumping the string up would break the symmetry stemming out from the center by creating more repetitions of '01' or '10' on the pumped side than are mirrored on the other half of the palindrome, as well as more 0s and 1s in general on the pumped side (assuming pumped n times: $0^p(0^j1^k)^n1^{2p}0^p$ or $0^p1^{2p}(1^k0^j)^n0^p$), breaking the palindrome. This breaks the ruleset and thus violates the pumping lemma.

Ex: v = 01, y = ε, uvvxyyz -> '0101' '' 10

<u>All Ones</u>

0…0 1…..1 0…0

p 2p p

If $vxy = 1^k$ where 1 <= k <= p, then pumping the string up would result in more 1s than 0s in the whole string ($0^p1^{2p+k}0^p$), breaking the ruleset. This violates the pumping lemma.

No matter which window vxy exists within the string, it cannot be pumped while continuing to follow the ruleset of L. Thus the pumping lemma is always violated, and since the pumping lemma cannot apply, there is a contradiction. Therefore, L, the language of all palindromes over the alphabet {0,1} in which the numbers of 0s and 1s are equal, must not be a CFL.

Problem 2.
(10 points) Show that the class of TM-decidable languages is closed under the following operations: union, concatenation, star, intersection, and complement.

A:
Union
Let TM_1 be a TM that decides L_1 and TM_2 be a TM that decides L_2.
A $TM_{1 \cup 2}$ that decides $L_1 \cup L_2$: With some input, i, run TM_1 and TM_2 on i, and accept if and only if at least one accepts, else reject. Since this is a valid TM, TM-decidable languages are closed under union.

Concatenation
Let TM_1 be a TM that decides L_1 and TM_2 be a TM that decides L_2.
A TM_{12} that decides L_1L_2: With some input, i, for each way to partition i into a left and right portion (from **ε** on the left and i on the right to i on left and **ε** on the right, moving one character from the right to the left at a time), run TM_1 on the left portion and TM_2 on the right portion, and accept if both accept, else reject. Since this is a valid TM, TM-decidable languages are closed under concatenation.

Star
Let M be a TM that decides L.
A TM that decides L*: If input i = **ε**, accept, else break i into every possible combination, $2^{|i|-1}$, of subcomponents, where no subcomponent = **ε**. For each of these combinations, run M on each subcomponent. If M accepts all of them, accept, else reject. Since this is a valid TM, a TM-decidable language is closed under star.

Intersection
Let TM_1 be a TM that decides L_1 and TM_2 be a TM that decides L_2.
A $TM_{1 \cap 2}$ that decides $L_1 \cap L_2$: With some input i, run TM_1 and TM_2 on i, and accept if and only if both machines accept, else reject. Since this is a valid TM, TM-decidable languages are closed under intersection.

Complement

Let M be a TM that decides L.

_ ← is to signify the complement of L

A TM that decides L: With some input i, run M on i, and accept if M rejects, and reject if M accepts. Since this is a valid TM, a TM-decidable language is closed under complement.

Problem 3.
(10 points) Show that the class of TM-recognizable languages is closed under the following operations: union, concatenation, star, and intersection. Is it closed under complement?

<u>Union:</u>
Say L_1 and L_2 are languages recognized by the Turing Machines M_1 and M_2 respectively. A TM $M_{1 \cup 2}$ that recognizes $L_1 \cup L_2$:

Run M_1 and M_2. Moving one position on the input at a time, $M_{1 \cup 2}$ would not stop until either M_1 or M_2 accepts the given input or both reject the given input. $M_{1 \cup 2}$ would accept an input “s” if M_1 or M_2 were to accept s. If both M_1 and M_2 reject, $M_{1 \cup 2}$ would reject s. Thus, if s belongs to $L_1 \cup L_2$, $M_{1 \cup 2}$ will always accept s in a finite number of steps. Thus, TM-recognizable languages are closed under union.

<u>Concatenation:</u>
Say L_1 and L_2 are languages recognized by the Turing Machines M_1 and M_2 respectively. In order to prove closure under concatenation we will create a Nondeterministic Turing Machine N that recognizes the language L_1L_2:

Given the ability of NTMs to clone and accept if at least one path accepts the input constraints, on input s, split s into two components (let’s call them a and b) where s = ab. Then run component a in M_1 and component b in M_2. If both machines accept, then N accepts s; if either M_1 or M_2 rejects, it rejects s. Thus, if s belongs to L_1L_2, N will always accept s in a finite number of steps. Since NTMs can be translated into 3-tape and thus 1-tape TMs, this shows TM-recognizable languages are closed under concatenation.

<u>Star:</u>
Say L is a language recognized by a Turing Machine M. In order to prove closure under star we will create a Nondeterministic Turing Machine N that will recognize L^*:

Given the ability of NTMs to clone and accept if at least one path accepts the input constraints, on input s, split s into a finite number, n, of components such that $s = s_1s_2s_3...s_n$ where s_i does not = ε. Then run s_i for every i, where $1 \leq i \leq n$, through M. If M accepts every s_i, N will accept. If M halts or rejects any s_i, N rejects. If there is a way to cut s such that M can accept every s_i, then s belongs to L^*. This means N will always accept an input belonging to

L^* in a finite number of steps. Thus, TM-recognizable languages are closed under the star operation.

<u>Intersection:</u>

Say L_1 and L_2 are languages recognized by the Turing Machines M_1 and M_2 respectively. To construct a TM $M_{1 \cap 2}$ that recognizes $L_1 \cap L_2$:

First run the given input s in M_1. If M_1 rejects s, then $M_{1 \cap 2}$ rejects s. If M_1 accepts s, then run s in M_2. If M_2 rejects s, $M_{1 \cap 2}$ rejects s. If M_2 accepts s then $M_{1 \cap 2}$ accepts s. Thus, if s belongs to $L_1 \cap L_2$, $M_{1 \cap 2}$ will always accept s in a finite number of steps. Therefore, TM-recognizable languages are closed under intersection.

<u>Closed under complement?:</u>

No, TM-recognizable languages are not closed under complement.

Let a Turing Machine M recognize language L. M will accept all strings in L in a finite number of steps. However, if you were to create the complement TM, M^c, that recognizes the language L^c, M^c would need to accept all strings that were not included in L. This would include all strings that ran on forever, but M^c would now need to accept them in a finite number of steps in order for the language L^c to be recognizable. There is no way that the complement of a recognizable language can accept any of the infinitely long strings that were neither accepted nor rejected by L in a finite number of steps. Thus, TM-recognizable languages cannot be closed under complement.

^{c} = complement

Problem 4.
(10 points) Show that every infinite TM-recognizable language has an infinite decidable subset.

Let A be some infinite TM-recognizable language. This means there's an enumerator E for A. Let F be the enumerator for the subset of A. It'll print the subset of A in lexicographic order.

F will ignore the input, and start by simulating E. When E prints a string, so will F, and let Last = that string, and continue simulating E. When E is about to print a new string, check if that string is longer than Last. If it is, then the lexicographic order is maintained, Last = the new string, and print w. If it isn't, don't print w. Then continue simulating E.

Since F only operates through simulating E with an additional check, it will certainly only ever print strings in A. F's language would be a subset of A. Since A is infinitely large, there will always be strings in the language larger than the Last string, so the language of F is also infinite.

The language of F is decidable since it is guaranteed to print strings in lexicographic order. Thus, on input i, if F prints i, accept, otherwise if F prints a string that comes after i in lexicographic order, then i is certainly not in the language since the language is ordered. Therefore, we know F will never print the input i, and can reject. Therefore the TM that accepts the language of F will always either accept or reject. Therefore, the language of F is decidable.

Therefore, the language of F is an infinite, decidable subset of A, an infinite TM-recognizable language.