COMP 330 Winter 2021
Assignment 5

Belle Pan 260839939

25th March 2021

Solution 1.
Consider the two languages $L_1 = \{a^m b^n c^k d^j | m, n, k, j \geq 0$ and $m = n$ and $k = j\}$ and $L_2 = \{a^m b^n c^k d^j | m, n, k, j \geq 0$ and $m = k$ and $n = j\}$. One of these languages is context-free but the other is not. Identify which is which. For the context-free language give a context-free grammar, and for the other one give a proof using the pumping lemma that is is not context-free.

L_1 is clearly context-free; the following grammar is a clear indication of this:

- $S \to MK|\epsilon$
- $M \to aMb|\epsilon$
- $K \to cNd|\epsilon$

M ensures that the number of a's and b's are equal ($m = n$), and K ensures that the number of c's and d's are equal ($k = j$). The string terminates as all of the instructions have the option of ϵ.

L_2, on the other hand, is not context free; below is a proof of this using the pumping lemma in the form of an angel-demon game:

- The demon chooses a number p.
- The angel chooses a word $s = a^p b^p c^p d^p$.
- The demon needs to divide s into five parts, denoted u, v, w, x, y such that $|vx| > 0$ and $|vwx| \leq p$.
  - Case 1: v or x crosses a boundary between two blocks.
    - If the angel chooses an integer $i = 2$, the pumped word would have letters out of order (i.e. it could result in a word in the form $a^e(a^f b^g)^2 b^h c^p d^p$) and would thus not be in L_2.
  - Case 2: v contains only a's.
    - x cannot be in the block containing c's, as $|vwx| < p$. Therefore, if the angel chooses an integer $i = 2$, the pumped word would have more a's than c's and would thus not be in L_2.
  - Case 3: v contains only b's.
    - Similarly to Case 2, x cannot be in the block containing d's, as $|vwx| < p$. Therefore, if the angel chooses an integer $i = 2$, the pumped word would have more b's than d's, and would thus not be in L_2.

Therefore, L_2 must not be context free according to the pumping lemma.

Solution 2.
Consider the language: $L = \{a^i b^j c^k | i, j, k \geq 0$ and $(i \neq j$ or $j \neq k)\}$. Prove that this language is context-free by giving a grammar but not deterministic context free by showing that its complement is not context-free. The grammar can be rather long to write out fully so it suffices to explain what you are doing and reduce it to similar cases and then say "this case is just like what we have done before." Of course, if you want to write out all the rules you are free to do so. Some of you might find that easier than writing explanations. To prove that the complement is not context free can be done by a reduction argument to a familiar language that is known to be not context free. A direct pumping lemma proof would be painful.

L is context-free; the following grammar is a clear indication of this:

- $S \rightarrow WX|YZ$
- $W \rightarrow A|B_1$
- $X \rightarrow cX|\epsilon$
- $Y \rightarrow aY|\epsilon$
- $Z \rightarrow B_2|C$
- $A \rightarrow aA|a|T$
- $B_1 \rightarrow B_1b|b|Tb$
- $B_2 \rightarrow bB_2|b|U$
- $C \rightarrow Cc|c|Uc$
- $T \rightarrow aTb|\epsilon$
- $U \rightarrow bUc|\epsilon$

In S, we are given two choices: WX and YZ.

- WX
  - W ensures that the number of a's $\neq$ b's, as we are given two choices A and B_1; these ensure that the number of a's is greater than b's and that the number of a's is less than b's respectively.
  - A adds at least one a to the word before going to T, which adds the same amount of a's and b's. Thus, there will always be at least one more a than b's in the word produced through this series of decisions.
  - Similarly, B_1 adds at least one b to the word before going to T, which adds the same amount of a's and b's. Thus, there will always be at least one more b than a's in the word produced through this series of decisions.
  - X adds any number of c's to the end of the word.
- YZ
  - Y adds any number of a's to the beginning of the word.
  - Z ensures that the number of b's $\neq$ c's, as we are given two choices B_2 and C; these ensure that the number of b's is greater than c's and that the number of b's is less than c's respectively.
  - B_2 adds at least one b to the word before going to U, which adds the same amount of b's and c's. Thus, there will always be at least one more b than c's in the word produced through this series of decisions.
  - Similarly, C adds at least one C to the word before going to U, which adds the same amount of b's and c's. Thus, there will always be at least one more c than b's in the word produced through this series of decisions.

To show that L is not a deterministic context-free language, we show that the intersection between the complement of L and a regular language yields a non-context-free language. We use this approach because we know two facts: that deterministic context-free languages are closed under complement, and that the intersection between a context-free language and a regular language yields a context-free language; using these facts, we can easily determine whether or not the context-free language is deterministic in nature. In this case, ff the complement of L intersected with a regular language is clearly not context-free, then we know that L must not be deterministic context-free.

- We use the regular language $L' = \{a^l b^m c^n | l, m, n \geq 0\}$ to intersect the complement of L, $\bar{L}$.
- $L \cap \bar{L} = \{a^n b^n c^n | n \geq 0\}$, which we know is not context-free.

Hence, we know that L is context-free, but is not a deterministic context-free language.

Solution 3.
Suppose we have a language L defined over the alphabet $\{a, b, c\}$ and suppose that L is context-free. We define a new language $perm(L)$ to be the set of all permutations of all words in L. For example, if $L = \{abc, aab\}$, then $perm(L) = \{abc, acb, bac, bca, cab, cba, aab, aba, baa\}$. Show that $perm(L)$ need not be context-free by giving an example of a language L that is context-free but where $perm(L)$ is not context-free. You need not give a pumping lemma proof if your example is just like one we have seen in class.

First, let me admit that I truly would like to avoid using the pumping lemma for the sake of conserving my brain cells. Given this goal, I want $perm(L)$ to be in the form $a^n b^n c^n$, $n \geq 0$ which we have seen time and time again is not context-free; however, this is not possible as $perm(L)$ is a set of all permutations of words in L. Therefore, we must work with context-free language closure properties: by context-free language closure properties, the intersection between a regular language and $perm(L)$ would be context-free IF $perm(L)$ is context-free. It is clear now that I must make a language L that will give a certain permutation $a^n b^n c^n \in perm(L)$, and then find the intersection between $perm(L)$ and the language $\{a^l b^m c^n | l, m, n \geq 0\}$ to yield the language $\{a^n b^n b^n | n \geq 0\}$ which we know is not context-free.

- Let $L = \{(abc)^n | n \geq 0\}$, which yields $perm(L) = \{a^n b^n c^n, a^{n-1} b^n c^n a, a^{n-1} b^{n-1} c^n ab, ... | n \geq 0\}$.
- Let $L_2 = \{a^l b^m c^n | l, m, n \geq 0\}$. We know that this is a regular language.
- By context-free language closure properties, if $perm(L)$ is context-free, then $perm(L) \cap L_2$ must yield a context-free language.
- However, $perm(L) \cap L_2 = \{a^n b^n b^n | n \geq 0\}$, which we know is not context-free.

Therefore, $perm(L)$ may not necessarily be a context-free language for every language L.

Solution 4.
We have seen in class that the language $L_1 = \{a^{i+j} b^{j+k} c^{k+i} | i, j, k \geq 0\}$ over the alphabet $\{a, b, c\}$ is context-free. Consider the language $L_2 = \{a^{i+j} b^{j+k} c^{k+l} d^{i+l} | i, j, k, l \geq 0\}$ over the alphabet $\{a, b, c, d\}$. Is this language also context-free? If so, give a context-free grammar for it; if not, prove that it is not context-free using the pumping lemma.

The following is a grammar for L_2:

- $S \to I | JKL$
- $I \to aSd | \epsilon$
- $J \to aJb | \epsilon$
- $K \to bKc | \epsilon$
- $L \to cLd | \epsilon$

To explain this grammar, let us first rewrite the language L_2 as $\{a^i (a^j b^j)(b^k c^k)(c^l d^l) d^i | i, j, k, l \geq 0\}$.
In S, we are given two choices: I and JKL.

- I ensures that the number of a's and d's at the beginning and end of the word are equal, generating the parts a^i and d^i. I returns us to S, which allows us to choose to either continue increasing the number of a's and d's at the beginning and end of the word, or to begin generating the $(a^j b^j)(b^k c^k)(c^l d^l)$ part of the word. I also gives us the option of terminating, as it has the option of ϵ; using this option generates words of the form $a^i d^i$ where $i \geq 0$.
- JKL ensures the generation of the $(a^j b^j)(b^k c^k)(c^l d^l)$ portion of the word: J generates an equal number of a's and b's which makes up the $(a^j b^j)$ section, K generates an equal number of b's and c's which makes up the $(b^k c^k)$ section, and L generates an equal number of b's and c's which makes up the $(c^l d^l)$ section of the word. The word terminates because J, K and L all give the option of ϵ; in addition, having ϵ as an option ensures that j, k, l can equal 0 or be larger than 0.

Solution 5.
Prove that the complement of the set of palindromes is context-free: this is the set of words $\{w \in \Sigma^* | w \neq w^{rev}\}$. You do not have to give a formal proof but you must explain your answer.

The complement of the set of palindromes is clearly context-free. For example, take the language $\{w \in \Sigma^* | w \neq w^{rev}\}$ with $\Sigma = \{a, b\}$, which we can explicitly design a grammar to recognize:

- $S \rightarrow aSa|bSb|aDb|bDa$
- $D \rightarrow aD|bD|\epsilon$

S provides us with four choices:

- aSa and bSb both generate the same character at the beginning and end of the string. It is not possible to terminate the string using only the options aSa and bSb, so eventually we will need to choose aDb or bDa.
- aDb and bDa ensure that there is at least one mismatched pair in the string. Then, D generates any number of a's and b's in any order and ensures that all non-palindrome strings can be generated by this grammar. D also ensures that the string terminates as it has ϵ as an option.

To be more general about this problem and proving that the set of ALL non-palindromes (unlike the above, which addresses only non-palindromes with letters a and b), we may consider using a non-deterministic Pushdown Automata (PDA) described below:

- The machine reads each character in the string and pushes it onto the stack until it reaches the middle of the word.
- The PDA somehow recognizes that it has reached the middle of the word and starts popping symbols off the stack if it matches with the character it reads from the string.
- If the PDA reads a mismatching symbol, then the PDA recognizes that the word is NOT a palindrome and accepts.
- If the stack is empty at the end of execution, the PDA rejects the string.

As a PDA can be designed to recognize the complement of the set of palindromes, we know that it must be context-free.