

Theory of Computation
CS 378
Fall 2019

Problem Set 5

Due **AT THE BEGINNING OF CLASS** Tuesday, November 19

- Please finish reading Sipser Ch. 2.3.
- As always, answers to all exercises should be accompanied by explanations; brief explanations sometimes suffice, but all answers should be explained. Answers without explanations may not be awarded full credit.
- *A general note:* When writing up your homework, please explain your arguments clearly and write neatly. Graders may not award credit to incomplete or illegible solutions. Clear communication *is* the point, on every assignment.

1. **Fun with Hans und Franz!** Use the Pumping Lemma for CFL's to show that each of the following languages is not context-free.

(a) $\{a^n b^n c^m \mid n \leq m \leq 2n\}$.

(b) $\{b_i \# b_{i+1} \mid b_k \text{ represents } k \text{ in binary, } i \geq 1\}$. (*Notes:* The symbol $\#$ is an input symbol that is used to delimit the end of string b_i . A string in this language therefore is a binary number followed by a single $\#$ symbol followed by a binary number that is one larger than the previous one.)

2. **Closure Properties of the CFL's.** Sipser 2.16.

3. **The Context-Free Languages are Not Closed Under Complementation!**

On the previous problem set, the optional exercise (“**And Now For Something Truly Funky!**”) established that the language

$$(a + b)^* - \{(a^i b^i)^i \mid i \geq 1\}$$

is context-free.

Now, show that the *complement* of this language, $\{(a^i b^i)^i \mid i \geq 1\}$, is NOT context-free using the Pumping Lemma for CFL's. Note that this result demonstrates that the context-free languages are not closed under complementation!

4. **The Context-Free Languages are Not Closed Under Intersection!**

In this problem you will show that *unlike* the regular languages, the context-free languages are *not* closed under intersection. To show this you need only exhibit two context-free languages whose intersection is not context-free.

Give two context-free languages L_1 and L_2 such that $L_1 \cap L_2$ is not context-free. To show that L_1 and L_2 are context-free, give grammars or PDAs (or English descriptions of how the PDAs would work) for these two languages. You should use the pumping lemma for CFLs to show that $L_1 \cap L_2$ is not context-free. However, if $L_1 \cap L_2$ is already a language that we have shown in class to be not context-free, you can cite that proof without redoing it.

HINT: Sipser's exercise 2.2(a), on page 154, suggests an approach to this exercise. You are welcome to use that approach or a different one, whichever you prefer.

5. **The Shuffle!** Two words w_1 and w_2 can be “shuffled” into a single new word by taking zero or more consecutive letters from the beginning of w_1 followed by zero or more letters from the beginning of w_2 followed by the next 0 or more letters from w_1 , followed by the next 0 or more letters from w_2 , etc. until all characters are expended. For example if w_1 is the word “funny” and w_2 is the word “joke” then one possible shuffle of w_1 and w_2 is the word “fujnnokye”. The first two letters, “fu”, come from the beginning of w_1 . The next letter, “j” comes from the beginning of w_2 . The next two letters, “nn”, come from the next two letters of w_1 . Then comes “ok” from w_2 , “y” from w_1 , and finally “e” from w_2 . Another possible shuffle of these two words is “funnjokye”.

Given two languages, L_1 and L_2 , $\text{shuffle}(L_1, L_2)$ is a new language formed by shuffling each word in L_1 with each word in L_2 in all possible ways.

- (a) Show that the regular languages are closed under shuffle. That is, show that if L_1 and L_2 are regular, then $\text{shuffle}(L_1, L_2)$ must also be regular.
 - (b) Show that the context free languages are **not** closed under shuffle. Again, prove this carefully. (*Note:* To do this you need only demonstrate a single counterexample. If you can find two languages L_1 and L_2 which are CFL’s but whose shuffle is not a CFL then you are done. If that doesn’t work, you might consider shuffling two CFL’s and then applying a known closure property of CFL’s.)
 - (c) Show that the shuffle of a CFL and regular language is always a CFL. Explain this carefully.
6. **FMP’s (Funky Modified PDA’s), part 1—Bottom Window PDA’s!**

Consider a PDA that “can tell” at any time what symbol in the stack alphabet is at the bottom of the stack. That is, the PDA has a window at the bottom of the stack that it can use to peek and see what’s there. More formally, the transition function of such a machine would be

$$\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \times \Gamma_\epsilon \rightarrow \mathcal{P}(Q \times \Gamma_\epsilon)$$

where the input comprises the state of the PDA, the symbol to read from the tape (or ϵ if nothing is to be read and the machine will operate on the stack without reading the tape), the symbol to be popped from the stack (or ϵ if the machine ignores the top of the stack), and the symbol on the bottom of the stack (ϵ if the stack is empty), and the output is as for a standard PDA.

Is such a machine more powerful than a regular PDA? That is, can a bottom window PDA accept a non-context-free language? If so, demonstrate such a language, explain why it is accepted by a bottom window PDA, and prove that it is in fact not context free (unless you use a language that we’ve already shown to be non-context-free). If not, explain how a regular PDA can simulate a bottom window PDA.

Optional: FMP's, part 2—Middle Window PDA's!

This exercise is not to be turned in! It's an exercise that your Prof. thought might be of interest, and you are welcome to work on it and talk with your Prof. about it, but it will not be graded.

Middle Window PDA's! Recall the FMP exercise above (the non-optional one!) about Bottom Window PDA's—this one is about Middle Window PDA's!

Consider a PDA that “can tell” at any time what symbol is in the middle of its stack. That is, the PDA has a window that allows the PDA to see the middle symbol. If the PDA has an odd number of symbols on the stack, the middle is well-defined. If the PDA has an even number of symbols on the stack, we'll define “middle” to mean the symbol just above the midpoint (the “higher of the two possible middles”). Note that the middle window “floats” up and down as the number of elements on the stack changes.

Is such a machine more powerful than a regular PDA? If so, demonstrate a language that is not context free but is accepted by a middle window PDA (prove that this language is not context free or use a language that we've already shown to be a non-CFL) and explain how it is accepted.

If not, explain how a regular PDA can simulate a middle window PDA.