

Theory of Computation
CS 378

Spring 2021

Problem Set 4

Due **AT THE BEGINNING OF CLASS** Tuesday, Apr. 13

- **Submission instructions:** Please submit this assignment to your `SubmittedWork` folder in a file named `CS378_PS4_<userid>.pdf`.
- Please finish reading Sipser Ch. 2.1 and 2.2.
- **IMPORTANT:** As an exercise for yourself, please do Exercise 2.6(a) on page 155 of Sipser, the solution for which is given in the textbook. Make sure you understand the solution, and (as always) please feel free to ask your Prof. any questions you may have about it!
- 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.
- *A general note:* Neatly written (or typeset) solutions, with enough blank space on the page to allow graders to write comments before returning the papers, are greatly appreciated!

1. **Context-Free Grammars!**

Exercise 2.6(d) on page 155 of Sipser. Note that the alphabet is $\{a, b, \#\}$.

As part of this exercise (and for every CFG designed for CS378!), be sure to explain briefly what each variable in the grammar represents. (For example, “The variable S generates all strings in the language in the problem. The variable A generates all strings with an equal number of 0’s and 1’s.”) This should greatly simplify your explanation of how your answer is a correct grammar for the given language!

Although it is certainly possible to over-explain CFG design, it may be even more possible to under-explain it—to assume a reader will understand points of design that aren’t obvious to them. Please be thoughtful and thorough in explaining what each variable means. (If you find it helpful to give example strings generated by each variable, please feel free to do so as part of that.)

A Small Hint: We’ve gone over a CFG that generates palindromes already in a CS378 class meeting. What is the role of palindromes in this exercise? How might knowing how to generate a palindrome—or a palindrome with some modifications, perhaps—be useful here?

2. Context-Free Grammars and PDA's!

- (a) Give a context-free grammar for the language

$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and either } k = i \text{ or } k = i + j\}.$$

Explain briefly what each variable in your grammar represents. (For example, “the variable B in my grammar represents the situation that I have seen an equal number of 0’s and 1’s and now I expect to see only 1’s.”)

- (b) Draw the state diagram for a PDA for the above language.

Note: You may not use the mechanical construction from Theorem 2.20 and Lemma 2.21 for this exercise! Instead, consider Figures 2.15 and 2.17 in your Sipser textbook as examples. Please design the PDA and, as with designs of DFAs (indeed, as with any other machine models in CS378), be sure to explain the state set, transition function, etc.; answers to this exercise will be evaluated as all machine designs are in CS378, so please make sure the design isn’t extremely inelegant, please include a correctness argument (it can be brief if the design is explained well!), and follow other course guidelines for designs of machine models.

As always, please feel free to talk with your Prof. if there are any questions about this!

3. **A Funky Language!** Sipser, Exercise 2.22 (page 156). Note that $\#$ is one of the symbols in the alphabet here. In general, to show that the language is context-free you may either give a context-free grammar or a PDA. *In this case, describe in English how a PDA would work for this language.* (You may also include PDA state diagrams or grammars in your answer, if they improve the clarity of your English description, but the exercise here is to write a clear English description.) Your description might begin something like this: “First I would push blah, blah, blah onto the stack. Then, for each symbol of type blah that I see, I push blah onto the stack. Then, I blah, blah, blah.” Avoid using the term “blah” in your own description if at all possible!

Hints: In general, one way to show that two strings $x = x_0x_1 \dots x_n$ and $y = y_0y_1 \dots y_m$ are unequal (all $x_i, y_i \in \Sigma$) when $m = n$ is to show that there’s some position j for which $x_j \neq y_j$. (Do you see why?) Recall that there is no length restriction on x, y —they could be arbitrarily long!—so there’s no restriction on what position j might be that would be useful in showing $x \neq y$.

- What could the stack of a PDA be used for to help accomplish this?
- Recall that states of a DFA are essentially the memory of that machine. Similarly, states of a PDA can also be used to store things in (finite) memory! How can the states of a PDA P for this exercise be set up to keep track of the information needed to determine if x and y are unequal?
- There may be **lots** of non-determinism in a PDA for this language! It may therefore be difficult to give a fully formal description of the PDA—that’s why this exercise asks for an English description. Formal notation may be part of an answer for this, and a description of every state and δ transition in the PDA should be somewhere in a fully complete answer, but a completely formal description (of the sort given for simple DFA’s, for instance) is not required.

Optional Exercises

These exercises are not to be turned in! They're exercises that your Prof. thought might be of interest, and you are welcome to work on them and talk with your Prof. about them, but they will not be graded. (The results of these exercises, however, may well come up again later in the course!)

- **A Slightly Funkier Language!** Sipser, Exercise 2.23 (page 157). As with exercise 3 above, please give an English description of how your PDA works rather than just a grammar or a state diagram. Be sure to explain why your PDA accepts exactly this language.

Note: This exercise is an interesting companion to exercise 3—it's similar in many ways, but with some interesting differences!

- **And Now For Something Truly Funky!** Consider the language

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

This is the complement of the language $\{(a^i b^i)^i \mid i \geq 1\}$. Show that this language is context-free by carefully describing in English how a PDA that accepts this language would work. Be sure to explain why your PDA accepts exactly this language.