

Theory of Computation
CS 378

Spring 2021

Problem Set 6

Due **AT THE BEGINNING OF CLASS** Thursday, May 6

- **Submission instructions:** Please submit this assignment to your SubmittedWork folder in a file named CS378_PS6_<userid>.pdf.
 - Please finish reading Sipser Ch. 3 and 4.
 - 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:* 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. **Countability!** In each case below, determine whether the given set is countable or uncountable. As always, prove your answer. ($\mathbb{N}$ refers to the set of natural numbers.)
 - (a) The set of all finite subsets of $\mathbb{N}$. In addition, explain why the diagonal argument cannot be used to show that this set is uncountable.
 - (b) The set of all functions from $\mathbb{N}$ to $\{0, 1\}$.
 - (c) The set of all functions from $\{0, 1\}$ to $\mathbb{N}$.
 - (d) The set of all regular languages over $\{0, 1\}$.
 2. (“We only needed one more stack...?”) A **two-stack machine** (2SM) is a memory-restricted version of a TM:
 - It has one input tape, but it is *read-only*—it cannot write on the tape.
 - It has two stacks to use as auxiliary memory, to be used in the normal ways stacks are used (as they were with PDA’s).
 - The tape head is also restricted—it can only move right, and once it reaches the end of the input, it stops moving.
 - The machine can continue manipulating the stacks and making state transitions at end the of its input, though, and it can enter `q_accept` or `q_reject` “whenever it wants.”

Prof. I. Lyalot and Prof. Anna Litik of the Portland Institute of Technology were discussing two-stack machines between their repeated—needlessly and gratuitously repeated, in fact—conversations about the language of redundancy language, and one of them made the claim that 2SM’s are just as powerful as standard TM’s.

It turns out this claim is true! Explain how a standard TM can be simulated by a 2SM; an informal but convincing explanation, addressing the relevant details, will be sufficient for a full-credit answer for this exercise.

Small Hint: The idea is to show that, given a TM M , we can give a 2SM S that recognizes $L(M)$. To do this, explain how S can simulate each move in the δ function of M —despite the restrictions on S about using its input tape—by using its stacks, and how that results in S recognizing $L(M)$.

3. **A Non-Recursive L_{union} !** Consider the following language:

$$L_{\text{union}} = \{ \langle M_1, M_2, M_3 \rangle \mid M_1, M_2, M_3 \text{ are TMs and } \mathcal{L}(M_1) \cup \mathcal{L}(M_2) = \mathcal{L}(M_3) \}$$

Prove by *reduction* that this language is not recursive.