CS244 Theory of Computation
Homework 3

Due: November 27, 2022 at 11:59pm

Name - ID

You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work and you should indicate in your submission who you worked with, if applicable. You should use the LaTeX template provided by us to write your solution and submit the generated PDF file into Gradescope.

I worked with: (Name, ID), (Name, ID), ...

Let $\Sigma = \{0, 1\}$ if not otherwise specified.

Problem 1

(a) (5 points) Consider the problem of testing whether a pushdown automaton ever uses its stack. Formally, let PUSHER $= \{\langle P\rangle \mid P$ is a PDA that pushes a symbol on its stack on some (possibly non-accepting) branch of its computation at some point on some input $w \in \Sigma^*\}$. Show that PUSHER is decidable.

(b) (5 points) Say that a variable A in CFG G is **usable** if it appears in some derivation of some string $w \in L(G)$. Given a CFG G and a variable A, consider the problem of testing whether A is usable. Formulate this problem as a language like (a) and show that it is decidable.

Problem 2

Let a k-PDA be a pushdown automaton that has k stacks. Thus a 0-PDA is an NFA and a 1-PDA is a conventional PDA. You already know that 1-PDAs are more powerful (recognize a larger class of languages) than 0-PDAs.

(a) (5 points) Give an example to show that 2-PDAs are more powerful than 1-PDAs.

(b) (10 points)Show that 3-PDAs are not more powerful than 2-PDAs.
(Hint: Simulate a Turing machine tape by using two stacks.)

Problem 3

Let A be a language.

1. (10 points) Show that A is Turing-recognizable iff $A \leq_{\mathsf{m}} A_{\mathsf{TM}}$.
2. (10 points) Show that A is decidable iff $A \leq_{\mathsf{m}} 0^*1^*$.

Problem 4

The Recursion Theorem tells us that a machine can gain its description and reproduce itself.

(a) (5 points) Give a Python program that prints itself out, in the spirit of the recursion theorem and the construction of **SELF**.

(b) (10 points) Consider the fixed-point theorem: If we have an algorithm that transforms a program to another program, there is always going to be some programs whose behaviour is unchanged by the transformation. Formally, show that for any computable function $f : \Sigma^* \to \Sigma^*$, there is a TM R such that $L(R) = L(S)$ where $f(\langle R \rangle) = \langle S \rangle$.

Problem 5

(20 points)
Let $\textit{SET-SPLITTING} = \{\langle S, C \rangle |$ S is a finite set and $C = \{C_1, \ldots, C_k\}$ is a collection of subsets of S, where the elements of S can be colored *red* or *blue* so every C_i has at least one red element and at least one blue element$\}$. Show that *SET-SPLITTING* is NP-complete.

Problem 6

(20 points)
In ***three-valued logic*** (TVL) we may assign values T, F, or X to variables. A ***TVL-clause*** has the form $(x, y) \neq (u, v)$ where x and y are variables and $u, v \in \{T, F, X\}$. A ***TVL-formula*** is a collection of TVL-clauses. An ***TVL-assignment*** of T, F, or X to the variables satisfies a TVL-clause if it doesn't violate the inequality, i.e., the pair (x, y) must not equal the pair (u, v) in the assignment. It satisfies a TVL-formula ϕ if it satisfies all of ϕ's TVL-clauses. Let $\textit{TVL-SAT} = \{\langle \phi \rangle |$ TVL-formula ϕ has a satisfying TVL-assignment. Show that *TVL-SAT* is NP-complete.