Homework 6 – Due Friday, October 25, 2019 before noon

Reminder Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to the instructor if asked. You must also identify your collaborators. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.

Problems Please practice on exercises and solved problems in Chapter 4. The material they cover may appear on exams.

1. (**Decidable languages, 10 points**)

   (a) (**Persistent variable**) A variable A in a CFG G is *persistent* if it appears in every derivation of every string w in G. Given a CFG G and a variable A, consider the problem of testing whether A is persistent. Formulate this problem as a language and show that it is decidable.

   (b) (**Sortedness-obsessed NFA**) Consider strings over an alphabet $\Sigma = \{1, 2, \ldots, 9\}$. A string w of length n over Σ is called *sorted* if $w = w_1 w_2 \ldots w_n$ where all characters $w_1, w_2, \ldots, w_n \in \Sigma$ and $w_1 \leq w_2 \leq \cdots \leq w_n$. For example, the string `1112778` is sorted, but the strings `5531` and `44427` are not. (Vacuously, the empty string is sorted.) We call an NFA *sortedness-obsessed* if every string it accepts is sorted. (But it does not have to accept all such strings). Consider the problem of determining whether a given NFA is sortedness-obsessed. Formulate this problem as a language and show that it is decidable.

   Hint: There are some solved problems in Chapter 4 that can help you.

2. (**Sortedness-obsessed TM, 5 points**) We call a TM *sortedness-obsessed* if every string it accepts is sorted. (But it does not have to accept all such strings). Consider the problem of determining whether a given TM with alphabet $\Sigma = \{1, 2, \ldots, 9\}$ is sortedness-obsessed. Formulate this problem as a language and show that its *complement* is recognizable.

3. (**Countable and uncountable sets and diagonalization, 15 points**)

   (a) Let $\mathcal{S}$ be the set of squares whose vertices have integer coordinates. For example, the square with vertices at $(1, 1), (0, 4), (3, 5), (4, 2)$ is in $\mathcal{S}$. Show that $\mathcal{S}$ is countable.

   (b) A *polynomial in variable* x is an expression of the form $c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \cdots + c_d x^d$, where d is a non-negative integer and $c_0, \cdots, c_d$ are constants, called coefficients. Let $\mathcal{P}$ be the set of polynomials with integer coefficients. Show that $\mathcal{P}$ is countable.

   (c) Let $\mathcal{F}$ be the set of all finite languages over alphabet $\{0, 1\}$. Show that $\mathcal{F}$ is countable.

   (d) Let $\mathcal{L}$ be the set of all languages over alphabet $\{0\}$. Show that $\mathcal{L}$ is uncountable, using a proof by diagonalization.

   (e) Your friend told you that he found a new Python library that contains many useful functions. One example is a function `halt` which takes two arguments: a program `main.py` and a valid input string x for that program. It returns 1 if `main.py` produces an output on input string x, and returns 0 if `main.py` runs forever on x. You want to convince your friend that `halt` cannot be always correct. However, your friend does not want to hear about TMs because they cannot possibly be relevant to Python programs.

Give a diagonalization argument (similar to that on p. 207 of Sipser) to convince your friend that function `halt`, as specified above, does not exist. Your analogue of TM D on p. 207 should be a Python program `src.py`. *Caution:* You cannot model it directly on TM D because your friend's claim is analogous to saying that $\mathsf{HALT}_{\mathrm{TM}}$ is decidable, not that A_{TM} is decidable.

Your program may call function `read` that reads a file and returns the content of a file as a string. If you do not know Python, you can use any programming language with C or Python-like syntax (or consult Ramesh on which ones he can grade).