

Homework 5

COMS W3261, Summer A 2022

This homework is due **Tuesday, 6/28/2022, at 11:59pm EST**. Submit to GradeScope (course code: 2KGDW8).

Grading policy reminder: \LaTeX is preferred, but neatly typed or handwritten solutions are acceptable. I recommend using the .tex file for the homework as a template to write up your answers. Your TAs may dock points for indecipherable writing.

Proofs should be complete; that is, include enough information that a reader can clearly tell that the argument is rigorous.

If a question is ambiguous, please state your assumptions. This way, we can give you credit for correct work. (Even better, post on Ed so that we can resolve the ambiguity.)

\LaTeX resources.

- The website [Overleaf](#) (essentially Google Docs for LaTeX) may make compiling and organizing your .tex files easier. Here's a quick [tutorial](#).
- [Detexify](#) is a nice tool that lets you draw a symbol and returns the \LaTeX codes for similar symbols.
- The tool [Table Generator](#) makes building tables in \LaTeX much easier.
- The tool [Finite State Machine Designer](#) may be useful for drawing automata. See also this example ([PDF](#)) ([.tex](#)) of how to make fancy edges (courtesy of Eumin Hong).
- The website [mathcha.io](#) allows you to draw diagrams and convert them to \LaTeX code.
- To use the previous drawing tools (and for most drawing in \LaTeX), you'll need to use the package Tikz (add the command "`\usepackage{tikz}`" to the preamble of your .tex file to import the package).
- [This tutorial](#) is a helpful guide to positioning figures.

1 Problem 1 (9 points)

1. (3 points) Prove that the language

$$A_{CFG-fast} := \{\langle G, w \rangle \mid G \text{ is a CFG and } w \text{ can be derived in } G \text{ in at most } |w| \text{ steps.}\}$$

is decidable by giving a high-level description of a Turing Machine that decides $A_{CFG-fast}$. As an example, the hypothetical derivation $S \rightarrow AA \rightarrow 00A \rightarrow 00B \rightarrow 001$ takes four steps (uses a substitution rule four times before reaching a string of terminals) to derive the string 001, which has length 3.

2. (3 points) Prove that the language

$$INC_{DFA} := \{\langle D_1, D_2 \rangle \mid D_1, D_2 \text{ are DFAs and } L(D_1) \subseteq L(D_2)\}$$

is decidable by giving a high-level description of a Turing Machine that decides INC_{DFA} .

3. (3 points) Prove that the language

$$EQ2_{TM} := \{\langle M_1, M_2, w \rangle \mid M_1, M_2 \text{ are Turing Machines, } w \text{ is a string and } M_1, M_2 \text{ either both accept or both reject on } w\}$$

is recognizable by giving a high-level description of a Turing Machine that recognizes $EQ2_{TM}$. Why doesn't your recognizer decide the language?

Rationale: The goal of this problem is to practice programming Turing Machines to decide and recognize languages, especially tricky ones that require simulating other automata.

References: See Sipser 193-197 for many examples of decidable languages, or review the notes and solutions to problems from Lectures 8 and 9.

2 Problem 2 (7 points)

Recall that a *configuration* summarizes a Turing Machine at a particular moment in time: in particular, a configuration is a finite string that includes (1) which internal state we're in, (2) what tape square the tape head is pointing at and (3) the contents of the tape (leaving out the infinite number of blank squares to the right of any square touched by the input or the tape head.) The subsequent behavior of a Turing Machine is completely determined by its current configuration. Thus, if we ever enter the same configuration twice, we're caught in an infinite loop.

I have a brilliant idea for a hypercomputer (the Time-Saving Machine, or TiM) that defeats the halting problem! In particular, my hypercomputer is *exactly* the same as a Turing Machine, with the following special rule that takes priority over normal execution: if a TiM ever enters the same configuration twice, it immediately halts and rejects.

1. (2 points). Give a high-level description of a TiM that runs forever without halting.
2. (5 points). Show that a TiM is no more powerful than a TM by proving that every TiM has an equivalent TM. (A high-level description of a TiM simulation is fine.)

Rationale: The goal of this question is to practice showing that Turing Machine variants are equal to Turing Machines in power.

References: See Sipser pp. 176-181, which shows how to reduce multitape and nondeterministic TMs to regular TMs. See also the review video Example 6: Reducing Variant TMs to TMs.

3 Problem 3 (6 points)

Recall the language

$$EQ2_{TM} := \{ \langle M_1, M_2, w \rangle \mid M_1, M_2 \text{ are Turing Machines, } w \text{ is a string and } M_1, M_2 \text{ either both accept or both reject on } w \},$$

which you showed is Turing-recognizable in problem 1. Show that this language is undecidable by reducing from the halting problem. [Hint: Start with the assumption that this language is decidable, and find a contradiction.]

Rationale: The goal of this question is to practice showing undecidability by reducing one problem to another problem.

References: See Sipser 215-220, in which several languages are shown to be undecidable by reducing from the halting problem.

4 Problem 4 (8 points)

For the following questions, an answer of a sentence or two is fine.

1. The final exam will be available on Gradescope from 12:01am EST on Wednesday, June 29 until 9:00pm EST on Friday, July 1. You can take the final during any contiguous 12 hours during this time period (your time starts when you download the file, and ends when you upload it again.) The test is not designed to take the whole 12 hours.

During the final exam, you'll be allowed to use your notes, the textbook, your past HW, and resources linked on the course homepage, including past lecture notes, review videos, and the course skeleton. You won't be allowed to collaborate or use external resources on the internet. The course staff will respond to Ed posts only to give basic clarifications about the problem statements.

- (a) Locate your graded submissions for HW1, HW2, HW3 and HW4. Locate your notes, or download the notability notes for the previous lectures.
 - (b) (1 point) Looking over your past work, what are one or two things you don't understand well or would like to review? What's one thing you have down pat?
 - (c) (1 point) Set a goal (in terms of performance, score, or something else) for the final exam.
 - (d) (2 points) What's your study plan to achieve that goal?
2. (1 point) What action, activity, or HW problem constitutes your best work of the term?
 3. (1 point) What's something that could have gone better, or that you would do differently if you took the class again?
 4. (1 point) How would you self-evaluate your performance in this class?
 5. (1 point) Is there anything left over that you'd like to learn in the future?

Rationale: The goal of this question is to get you thinking about what you've learned in this course and prepare you for the final. Good luck!

References: Homework solutions, course skeleton, and review videos on the course webpage.

5 Problem 5 (2 Extra Credit Points): Nondeterministic Turing Machines

Recall the language of graphs that contain Hamiltonian paths from the previous homework:

$$A_3 := \{\langle G \rangle : G \text{ is a graph that contains a Hamiltonian path}\}$$

(A Hamiltonian path is a path through the graph that touches every vertex exactly once.)

In HW4, you showed how to recognize this language using a deterministic TM. (In case you chose not to write a decider: it's possible to write a deterministic TM that decides this TM using a similar simple strategy.)

However, a *Nondeterministic* Turing Machine can solve this problem in time $O(n)$, where n is the number of vertices in the graph! Give a high-level description of an NTM that decides this language in time $O(n)$.