Homework 5

COMS W3261, Summer A 2023

This homework is due **Monday, 6/26/2023, at 11:59pm EST**. Submit to GradeScope (course code: K3VK75). If you use late days, the absolute latest we can accept a submission is Friday at 11:59 PM EST.

Grading policy reminder: LaTeX is preferred, but neatly typed or handwritten solutions are acceptable.[1] Feel free to use the .tex file for the homework as a template to write up your answers, or use the template posted on the course website. 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.

- 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] The website Overleaf (essentially Google Docs for LaTeX) may make compiling and organizing your .tex files easier. Here's a quick tutorial.

Problem 1 (15 points)

1. (3 points) Consider the language

$$STAR_{DFA} = \{\langle D_1 \rangle \mid D_1 \text{ is a DFA and } L(D_1) = \{a\}^* \text{ for some } a \in \Sigma, D_1\text{'s alphabet}\}.$$

(For example, if D is a DFA over the alphabet $\{0, 1\}$, then $\langle D \rangle \in STAR_{DFA}$ if $L(D) = \{0\}^*$ or $L(D) = \{1\}^*$.) Show that $STAR_{DFA}$ is decidable by giving a high-level description of a Turing Machine that decides $STAR_{DFA}$.

Our Turing Machine works as follows:

- Check to make sure the input encodes a DFA D_1. (You can assume this step; including it is optional.)
- Let Σ be the alphabet of D_1, which is finite by definition. For each character $a \in \Sigma$, we can build a DFA D_2 that recognizes the language $\{a\}^*$ and write down its encoding on the tape.
- Use a hard-coded copy of our decider for EQ_{DFA} from class to test if $L(D_1) = L(D_2)$, and accept if and only if our decider accepts.

2. (4 points) Consider the language

$$STAR2_{DFA} = \{\langle D_1 \rangle \mid D_1 \text{ is a DFA and } L(D_1) = F^* \text{ for some } \textit{finite} \text{ language } F \subset \Sigma^*\}.$$

(For example, if D is a DFA over the alphabet $\{0, 1\}$, then $\langle D \rangle \in STAR2_{DFA}$ if $L(D) = \{0, 11, 010\}^*$ or the star of some other finite language.) Show that $STAR2_{DFA}$ is recognizable by giving a high-level description of a Turing Machine that recognizes $STAR2_{DFA}$.

[Hint: be careful when enumerating the finite languages over Σ.]

Our Turing Machine works as follows:

- Check to make sure the input encodes a DFA D_1. (You can assume this step; including it is optional.)
- Let Σ be the alphabet of D_1, which is finite by definition. Let $S_1, S_2, S_3, \ldots$ be an infinite sequence that enumerates every finite language over Σ.

  (We can construct such a language by enumerating all subsets of Σ^* containing strings of length at most 0, then all subsets of Σ^* containing strings of length at most 1, then all subsets of Σ^* containing strings of length at most 2, etc. Note that certain approaches for enumerating this infinite set don't work: for example, we can't enumerate all subsets of at most 0 elements, then 1 elements, then 2 elements, etc., as there are an infinite number of subsets of Σ^* with at most 1 element.)
- For $i = 0, 1, 2, ...$, build a DFA D_2 that recognizes the language S_i^* and write down its encoding on the tape. Then, use a hard-coded copy of our decider for EQ_{DFA} from class to test if $L(D_1) = L(D_2)$, and accept if and only if our decider accepts.

3. (8 points) Consider the language

$$A_{noloop} := \{\langle D \rangle \mid D \text{ is a DFA that never visits any state twice on an accepting string.}\}$$

In other words, the computational path taken by D on an accepting string never contains a loop.

Show that A_{noloop} is decidable by describing a Turing machine that decides the language. Explain why your TM accepts every string in A_{noloop} and halts and rejects on every string not in A_{noloop}.

There are a couple of ways we can decide this language, including:

"M = on input w:

- Check to make sure the input encodes a DFA D. (You can assume this step; including it is optional.)
- Let Q denote the state set of D and let Σ denote the alphabet of D. Simulate D on every string in Σ^* of length less than $2|Q|$. Accept if D accepts any string w after making a loop; reject if D never accepts a string after making a loop."

If the input encodes a DFA that never loops, then we accept: it is impossible to find an accepting string that makes a loop. If our input does not encode a DFA, we reject. It remains to show that if the input encodes a DFA that does loop on some input string w, we discover a loop and reject.

Let w be the shortest accepting string on which D completes a loop. If $|w| < 2|Q|$, we discover this string and reject. Alternatively, suppose $|w| \geq 2|Q|$. In this case, the computation on w must visit some state at least *three* times, as we visit $|w| + 1 > 2|Q|$ states before accepting w. Thus we can remove some characters from w to eliminate a loop from the accepting computation and get a shorter accepting string w' on which D still completes a loop. However, this contradicts our assumption that w is the shortest string on which D completes a loop.

Thus if D loops on any accepting string, it must loop on an accepting string of length less than $2|Q|$, in which case our TM rejects.

Alternatively, we might perform a breadth-first search of depth $|Q|$ from the start state to locate any loop in our DFA, then search for accepting states reachable from the end of a loop.

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.

Problem 2 (6 points)

The transition function of an ordinary TM, $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, changes the internal state, writes a new character, and moves either left or right based on the current internal state and the character on the current tape square.

An *Undo-TM* is a variant TM with the following special power: for any (state, character) pair (q, a), we can specify the transition $\delta(q, a) = UNDO$. On an 'UNDO' operation, the TM *undoes* the most recently overwritten character. For example, if the TM just changed a 0 to a 1, then a 2 to a 3, the first UNDO would change the 3 back to a 2, and a subsequent UNDO would change the 1 back to an 0. An UNDO operation does not change the position of the tape head. If the tape has already reverted to its initial state (that is, a tape containing only the input), the UNDO operation does nothing.

Any TM trivially has an equivalent Undo-TM (an identical machine that chooses never to use the UNDO operation). Show that Undo-TMs are equivalent to TMs by showing that every Undo-TM has an equivalent TM.

Given an arbitrary Undo-TM U, we can build an equivalent TM T as follows:

- When U makes a normal transition, T imitates it by placing a delimiter (for example, a #) at the end of its tape, copying the entire tape to the other side of the delimiter, and finally executing the transition, modifying the copied tape.
- When U makes an UNDO operation, T erases the current tape up to the previous delimiter, and moves to the current tape position on the previous tape. (If there is no previous tape, the UNDO operation is ignored.)

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.

Problem 3 (6 points)

Consider the language

$P = \{\langle M, D\rangle \mid M \text{ is a TM, } D \text{ is a DFA, and there exists some string } w \text{ that } M, D \text{ both accept. }\}$

Prove that P is undecidable by reducing from $A_{TM} = \{\langle M, w\rangle \mid M \text{is a TM and } M \text{ accepts } w.\}$. [Hint: Start with the assumption that P is decidable, and find a contradiction.]

We know that A_{TM} is undecidable. We'll show that P is undecidable by showing that if we could decide P, we could decide A_{TM}, which is a contradiction.

Assume for contradiction that some TM H decides P. In that case, we can use H to build the following TM, which decides A_{TM}:

"T = On input $\langle M, w\rangle$:

1. Construct a DFA D that accepts only the string w.
2. Simulate $H(\langle M, D\rangle)$, and accept if and only if the simulation accepts."

We observe that $H(\langle M, D\rangle)$ accepts if and only if M and D both accept some string. Since D only accepts w, $H(\langle M, D\rangle)$ accepts if and only if M accepts w.

Thus a decider for P would allow us to decide A_{TM}, which is a contradiction. We conclude that P is undecidable.

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.

Problem 4 (5 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 Thursday, June 29 until 11:59pm EST on Friday, June 30. (We can also arrange for some students to take the final on Saturday, July 1. Please contact the course staff if you're interested in this option). 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) (2 points) Set a goal (in terms of performance, score, or something else) for the final exam. 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?

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.