CS3800: Theory of Computation — Summer II '22 — Drew van der Poel

Homework 4
Due Sunday, August 7 at 11:59pm via Gradescope

Name:
Collaborators:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This assignment is due Sunday, August 7 at 11:59pm via Gradescope. No late assignments will be accepted. Make sure to submit something before the deadline.
- Solutions must be typeset. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. I recommend using the source file for this assignment to get started.
- I encourage you to work with your classmates on the homework problems. *If you do collaborate, you must write all solutions by yourself, in your own words.* Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly forbidden.

Problem 1. *Turing Machine (*6 points*)*

Give a Turing machine that *decides* the following language $L = \{w \in \{A, B\}^* |$ the maximum number of non-overlapping occurences of substring AA is even$\}$.

For example, $\epsilon, AB, AAAA, AABBAA$, are all in L, while $AA, AAA, AABAABBAABBA$ are not.

You should explicitly specify Σ and Γ, in addition to providing a complete diagram of your Turing machine (with states, transitions, etc.). You should use q_{acc} and q_{rej} as your accept and reject states, respectively. You can assume that the input strings are given on the tape, with blank characters following the end of the string.

Provide commentary on the purpose of each state/transition.

Solution:

Problem 2. *Alternate TMs (*2 points*)*

Consider a new Turing machine that instead of moving one cell left or right, we can only move left 4 cells and right 2 cells. Can this new Turing machine be used in place of the standard Turing machine? That is, can it recognize any language that the standard one can? If yes, provide sound reasoning as to why this is. If not, give an example of a language which cannot be recognized by the new Turing machine, but can be recognized by the standard variant.

Note that the input will still be given consecutively on the tape, it is not spaced out to accommodate the new machine.

Solution:

Yes, the new Turing Machine can be used in place of the standard Turing Machine because we can convert the standard Turing Machine into the new Turing Machine.

1. For each character in the input, move the rest of

Problem 3. *True/False: Enumerators (*6 points*)*

For each of the following assertions, state whether it is true or false, and justify your answer. You do not need prove undecidability. You can assume that the alphabet $\Sigma = \{0, 1\}$ in the following problems.

(a) **[2 pts.]** If E is an enumerator, then there exists a TM D which is a decider such that $L(D) = L(E)$.

Solution:
False. An enumerator can also enumerate Turing-recognizable languages which means that $L(D)$ may not be equal to $L(E)$.

(b) **[2 pts.]** If D is a TM which is a decider, then there exists an enumerator E such that $L(E) = L(D)$.

Solution:
True. All Turing recognizable languages has an enumerator that enumerates all of the strings in the language. Therefore since the set of Turing-decidable languages is a subset of Turing-recognizable languages, there exists an enumerator that recognizes D.

(c) **[2 pts.]** Based on the above results, are deciders and enumerators equally powerful (can they decide/generate the same set of languages)? If yes, why is this the case? If no, which is more powerful and why is that?

Solution:
No, enumerators are more powerful since they can generate Turing-recognizable languages while deciders can only recognize Turing-decidable languages which are a subset of Turing-recognizable.

Problem 4. *Decidable DFA Problems (*16 points*)*

Consider the following computational problem: Does a given DFA accept any strings of length 5?

(a) **[1 pt.]** Give the corresponding language, L^5_{DFA}, for this problem.

Solution:

L^5_{DFA} = {< M > | DFA M accepts any strings of length 5}

(b) **[5 pts.]** Prove that L^5_{DFA} is decidable. You can do this by giving an algorithm, as we did in class. Justify your approach.

Solution:
1. Let T be the Turing Machine on L^5_{DFA}

2. On input M, generate all possible valid strings of length 5.

3. For each string, w, generated run the Turing Machine, D_{A-DFA}, that decides the DFA acceptance problem (we were shown this in class) on input $< M, w >$.

4. If D_{A-DFA} ever rejects, then T should reject and if D_{A-DFA} accepts on all of the generated strings, then T should accept.

Firstly, we know that D_{A-DFA} is a decider. Next, We know that by definition of a DFA, the set of input symbols in the alphabet must be finite and we also know that only valid strings of length 5 can be generated. Because of these two facts, there must be a finite upper-bound on the number of strings of length 5. Since the number of strings of length 5 is finite, we can run DFA M on each one of those strings and T will guarantee to halt because it will run a finite amount of times.

(c) **[1 pt.]** Now we generalize the problem further; does a given DFA accept any strings of a length which is divisible by 5? Give the corresponding language D^5_{DFA} for this new problem.

Solution:

D^5_{DFA} = {< M > | DFA M accepts any strings of length which is divisible by 5}

(d) **[5 pts.]** Prove that D^5_{DFA} is decidable. You can do this by giving an algorithm, as we did in class. Justify your approach.

Solution:
1. Let T_1 be the Turing Machine on D^5_{DFA}

2. On input M, generate a set of all valid strings of length 5, let's denote this set as L_a.

3. Construct D_a to be the DFA that recognizes L_a* (the set of all strings that have a length of

multiple of 5)

4. Construct D_b to be the DFA that recognizes $L(D_a) \cap L(M)$

5. Run the Turing Machine, D_{EQ-DFA}, that decides the DFA equivalence problem (we were shown this in class) on input $< D_a, D_b >$

6. If D_{EQ-DFA} accepts, then T_1 accepts and if D_{EQ-DFA} then T_1 rejects.

(e) **[4 pts.]** Consider generalizing the problem even further; does a given DFA accept any strings of a length which is divisible by a given value d? Give the corresponding language for this problem (you can name it whatever you want). Is this language decidable? If yes, state how to modify your algorithm from part (d). If no, argue why not.

Solution:
Let $L_D = \{< M, d > |$ DFA M accepts any strings of length which is divisible by d $\}$
L_D is decidable and we can modify the descriptions of step 2 and 3 from part d.

Step 2: We will now generate all valid strings of length d rather than length 5
Step 3: L_a will now be a set of all strings that have a length of multiple of d