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

Homework 5
Due Saturday, August 13 at 11:59pm via Gradescope

Name: Steve Liu
Collaborators: Jamie Lin, Ares Do, Eric Chapelaine, HwiJoon Lee, Jaron Cui

- 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 Saturday, August 13 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. *Multi-Automata Decidability (*8 points*)*

For each of the following languages, state whether or not they are decidable. If a language is decidable, give a proof (description of an algorithm) showing why. If not, give an argument as to how this would contradict one of the undecidability results we saw in class.

(a) **[4 pts.]** $L_1 = \{\langle P, T, w\rangle |$ P is a PDA, T is a Turing Machine, w is an input string and at least one of P or T accepts w$\}$

Solution:
L_1 is undecidable -

Assume true, that L_1 is decidable so there exists a Turing Machine, T_1, that decides L_1.

Let S be a Turing Machine that takes in the same input as A_{TM} (the Turing Machine acceptance problem), $< M, w' >$.

On input $\langle M, w'\rangle$ for S:

1. Construct a PDA, E, s.t. $L(E) = \varnothing$:

2. Run T_1 on inputs $< E, M, w' >$. If T_1 accepts then S should accept otherwise if T_1 rejects then S should reject.

This means that with the help of Turing Machine T_1 there exists the Turing Machine S that decides A_{TM} which is a contradiction since A_{TM} is undecidable. This means that T_1 cannot exist which means that L_5 is undecidable.

Explanation:

The idea behind this proof is that if PDA P rejects on all inputs then that means L_1 essentially boils down to A_{TM}, the Turing Machine Acceptance problem problem. Since we know that A_{TM} is undecidable then we know that if L_1 exists then we could construct a Turing Machine that decides A_{TM} which is a contradiction since A_{TM} is undecidable.

(b) **[4 pts.]** $L_2 = \{\langle N, G\rangle |$ N is a NFA, G is a CFG, and $L(N) = L(G) = \emptyset\}$

Solution:

L_2 is a decidable language. Let T_2 be the Turing Machine on L_2. On input $\langle N, G\rangle$:

1. Convert NFA N to the equivalent DFA, N_{DFA}, and run TM D_{E-DFA} on N_{DFA}.

2. If D_{E-DFA} rejects then T_2 should reject.

3. If D_{E-DFA} accepts then run TM D_{E-CFG} on CFG G.

4. If D_{E-CFG} rejects then T_2 should reject.

5. If D_{E-CFG} accepts then T_2 should accept.

Explanation:

Step 1: TM D_{E-DFA} is the Turing Machine that decides the emptiness problem for DFAs so this step halts.

Step 3: TM D_{E-CFG} is the Turing Machine that decides the emptiness problem for CFGs so this step halts.

We want to know if both $< N, G >$ are empty so we need both D_{E-DFA} and D_{E-CFG} to accept. Therefore, if at any point one of them rejects then we want T_2 to reject.

Problem 2. *Proving Undecidability with Reducibility (*8 points*)*

Consider the following language,
$L_5 = \{\langle M \rangle | \ M$ is a Turing machine and there is a string of length 5 in $L(M)\}$

Prove that L_5 is undecidable with a reducibility argument, using A_{TM}, the Turing machine-acceptance problem.

Solution:
Assume false, L_5 is decidable so there exists a Turing Machine, A, that decides L_5.

Let S be a Turing Machine that takes in the same input as A_{TM}, $< M', w >$.

On input $\langle M', w \rangle$ for S:

1. Construct a TM, B, s.t. on input $\langle X \rangle$:

    1a. Run M' on w:

    If M' accepts then B will accept. ($L(B) = \Sigma^*$ so $L(B)$ must include some string of length 5 if M' accepts w)

    If M' rejects either by halting or looping then B will reject. ($L(B) = \varnothing$ so $L(B)$ does not include any strings of length 5 if M' rejects w)

2. Run A on B:

    2a. If A accepts (this means that M' has accepted w which means that $L(B)$ has to include some string of length 5) then S will accept.

    2b. If A rejects (this means that M' has rejected w so $L(B)$ is the empty set) then S will reject.

This means that with the help of Turing Machine A there exists the Turing Machine S that decides A_{TM} which is a contradiction since A_{TM} is undecidable. This means that A cannot exist which means that L_5 is undecidable.

Problem 3. *Diagonalization (*4 points*)*

Recall the emptiness problem for Turing machines, E_{TM}. We showed in class that E_{TM} is undecidable via being reducible from A_{TM}. Now you're asked to construct an alternative proof of E_{TM} being undecidable, using a *diagonalization argument*.

Solution:
Assume false, E_{TM} is decidable so there exists some Turing Machine, H, that decides E_{TM}.

H on $< M >$, where M is some Turing Machine, will:
1. Accept if $L(M) = \emptyset$
2. Reject by halting if $L(M) \neq \emptyset$

Construct E s.t. it takes in input $< M' >$ where M' is some Turing Machine.

E = On input $< M' >$:

1. Construct B s.t. it takes in input $< X >$ where X is some string.

   B = On input $< X >$:

   1a. Run M' on input $< M' >$

   1b. If M' accepts $< M' >$ then B should accept. ($L(B) = \Sigma^*$ so $L(B) \neq \emptyset$)

   1c. If M' rejects $< M' >$ then B should reject. ($L(B) = \emptyset$)

2. Run H on B, if H accepts then E should accept, otherwise if H rejects then E should reject.

E accepts $< M' >$ iff H accepts B which means that $L(B) = \emptyset$ which can only happen iff M' rejects $< M' >$.

E rejects $< M' >$ iff H rejects B which means that $L(B) \neq \emptyset$ which can only happen iff M' accepts $< M' >$.

So if we run E on input $< E >$:

1. E accepts $< E >$ iff $L(B) = \emptyset$ which is only when E rejects $< E >$.

2. E rejects $< E >$ iff $L(B) \neq \emptyset$ which is only when E accepts $< E >$.

This is the contradiction here since E only accepts itself if E rejects itself and E rejects itself only when E accepts itself which cannot happen. Thus by contradiction, H cannot exist since it would allow the existence of E which means that E_{TM} is undecidable.

Problem 4. *We've Seen This Language Before....* (6 points)

Recall from earlier, $L_5 = \{\langle M \rangle | \ M \text{ is a Turing machine and there is a string of length 5 in } L(M)\}$. Prove that $\overline{L_5}$ is unrecognizable. You should not use a reduction in your proof.

Solution:
We must first show that L_5 is Turing-recognizable:

Let T_5 be the Turing Machine on L_5. On input $\langle M \rangle$:

1. Let $w_1, w_2, w_3...$ be the list of all strings in Σ^*

2. For $i = 1, 2, 3...$:

   2a. Run M for i steps on the first $w_1, w_2, w_3...w_i$ strings.

   2b. If M accepts some string w_j where $1 \leq j \leq i$ and the length of w_j is 5 then T_5 should accept. Otherwise, continue the loop.

T_5 will accept at some point if $L(M)$ contains some string that is of length 5 or T_5 will reject by looping if $L(M)$ does not contain any strings of length 5. Thus, L_5 is a Turing-recognizable language.

We've proven that L_5 is Turing-recognizable from the above algorithm and we've also proven L_5 is not Turing-decidable from problem 2. Since we know that a language is decidable if and only if it is both Turing-recognizable and co-Turing-recognizable and since we know L_5 is Turing-recognizable but not decidable then that means $\overline{L_5}$ must not be Turing-recognizable.

Problem 5. *Thought Experiment (*4 points*)*

In class we saw how to reduce A_{TM} to $HALT_{TM}$ to show $HALT_{TM}$ was undecidable. Suppose the roles were reversed, you know $HALT_{TM}$ is undecidable and want to use proof by contradiction to show A_{TM} is also undecidable.

Note that this is a thought experiment and the application doesn't make sense, as we used the undecidability of A_{TM} to prove that $HALT_{TM}$ was undecidable. So for our purposes (if it makes you happier), you can imagine $HALT_{TM}$ has somehow (magically) been shown to be undecidable but A_{TM} has not.

(a) **[1/2 pt.]** We are proving by contradiction. What is your initial assumption? What does this imply in terms of the existence of some Turing machine(s)? What is the input to that Turing machine?

Solution:

We initially assume false, that A_{TM} is decidable so there exists a Turing Machine, T, that decides the A_{TM} problem. The input to T would be $< M, w >$ where M is some Turing Machine and w is some string.

(b) **[1/2 pt.]** What are the possible outcomes of your above Turing machine on some input and what does each outcome mean?

Solution:
T accepts: This means that M has accepted w.

T rejects by halting: This means that M has rejected w either by halting or looping.

(c) **[2 pts.]** At least one of the outcomes above isn't helpful in concluding anything about $HALT_{TM}$. Identify any such outcome and explain why it doesn't allow us to draw conclusions about $HALT_{TM}$?

Solution:

T rejects by halting is not a useful outcome because we don't know if M has rejected by halting or looping on w. Therefore, iff T rejects w then we can't conclude that M has halted on w for the $HALT_{TM}$ problem because we don't know if M rejected by halting or looping.

(d) **[1 pt.]** Part (c) tells us this proof wouldn't work. But for the sake of exercise, for each remaining outcome from part (b), explain why this does allow us to draw conclusions about $HALT_{TM}$.

Solution:

The only remaining case is that T accepts which means that M has accepted w. This is useful because we can say that by definition, if M accepts w then we can conclude that M must also halt on w for the $HALT_{TM}$ problem.