HW5CSE105W24: Homework assignment 5

CSE105W24
Sample solutions

Due: March 14 at 5pm (no penalty late submission until 8am next morning), via Gradescope

1. **Computational problems** (14 points): For each of the following statements, determine if it is true or false. Clearly label your choice by starting your solution with True or False and then provide a justification for your answer.

(a) (*Graded for completeness*) True or False: The language

$$\{\langle N_1, N_2\rangle \mid N_1, N_2 \text{ are both NFAs over } \{0,1\} \text{ and } L(N_1) \subseteq L(N_2)\}$$

is recognizable.

Solution:

True. Here's a high-level description of the algorithm that would recognize this language. First, reject if input doesn't type check as string formatted to represent ordered pair of NFAs over $\{0,1\}$. Then, given N_1, N_2, we can turn them into D_1, D_2, both DFAs with the same language respectively. Then, construct DFA D such that its language is $L(D_1) \setminus L(D_2)$. This procedure can be done by flipping the accept/non-accept states of D_2 and then doing the DFA intersection construction of the resulting DFA with D_1. Finally, we just need to run a decider for E_{DFA} on $< D >$, accept iff the decider for E_{DFA} accept. The above process can be done by a TM in finite time. Moreover, the correctness stems from $L(D_1) \setminus L(D_2) = \emptyset \iff L(D_1) \subseteq L(D_2)$. Thus, the algorithm we described decides (and therefore recognizes) the language.

(b) (*Graded for correctness*) True or False: The language

$$\{\langle D\rangle \mid D \text{ is a DFA over } \{0,1\} \text{ and } L(D) = L(0^*)\}$$

is decidable.

Solution:

True. Here's a high-level description of the algorithm that would decide this language. First, reject if input doesn't type check as a string formatted to represent a DFA over $\{0, 1\}$. Then, define D' so that its language is $L(0^*)$ (i.e. have two states, with a loop at the start state labelled 0 and an edge from the start state to another state labelled 1, the second state having a self loop labelled both 0 and 1, and the start state is the only state in the set of accpeting states). Then, run a decider for EQ_{DFA} on $< D, D' >$ and accept iff it accepts. The construction process can be done in a finite time by a TM. Then, running a decider also halts, so the above algorithm can be implemented by a decider. Moreover, $< D, D' > \in EQ_{DFA} \iff L(D) = L(D') = L(0^*)$. Thus the language is decidable.

(c) (*Graded for correctness*) True or False: The language

$$\{\langle M \rangle \mid M \text{ is a Turing machine and its start state equals its reject state}\}$$

is decidable.

Solution:

True. Here's a high-level description of the algorithm that would decide this language. First, reject if input doesn't type check as a string formatted to represent a Turing machine. Then, given $\langle M \rangle$, all we have to do is to decode $\langle M \rangle$ and check if its start state is also the reject state by comparing the fifth and seventh components of the defining 7-tuple of the machine. This certainly can be done by a TM correctly in finite time, thus the language is decidable.

(d) (*Graded for completeness*) True or False: The language

$$\{\langle M \rangle \mid M \text{ is a Turing machine and } L(M) = \emptyset\}$$

is decidable.

Challenge; not graded: what's the difference between (c) and (d)?

Solution:

No. This is the definition of E_{TM}, and we know E_{TM} is unrecognizable, so certainly undecidable. The difference between c and d is that c is a lot more specific about the structure of the Turing machine so we just need to check the states. However, for E_{TM}, there are many reasons the language of a TM could be empty.

2. **What's wrong with these reductions?** (13 points): Suppose your friends are practicing coming up with mapping reductions $A \leq_m B$ and their witnessing functions $f : \Sigma^* \to \Sigma^*$. For each of the following attempts, determine if it is has error(s) or is correct. Do so by labelling each attempt with all and only the labels below that apply, and justifying this labelling.

- *Error Type 1:* The given function can't witness the claimed mapping reduction because there exists an $x \in A$ such that $f(x) \notin B$.
- *Error Type 2:* The given function can't witness the claimed mapping reduction because there exists an $x \notin A$ such that $f(x) \in B$.
- *Error Type 3:* The given function can't witness the claimed mapping reduction because the specified function is not computable.
- *Correct:* The claimed mapping reduction is true and is witnessed by the given function.

Clearly present your answer by providing a brief (3-4 sentences or so) justification for whether **each** of these labels applies to each example.

(a) (*Graded for completeness*) $A_{\text{TM}} \leq_m HALT_{\text{TM}}$ and

$$f(x) = \begin{cases} \langle \text{start} \rightarrow q_{\text{acc}} \rangle & \text{if } x = \langle M, w \rangle \text{ for a Turing machine } M \text{ and string } w \\ & \quad \text{and } w \in L(M) \\ \langle \text{start} \rightarrow q_0 \overset{0,1,\sqcup \rightarrow R}{\circlearrowright} \quad q_{acc} \rangle & \text{otherwise} \end{cases}$$

Hint: There are two errors with this attempt.

Solution:

i. Type 1 error is present because the output type of this function is wrong. Each element in $HALT_{TM}$ is of the form $\langle M, w \rangle$ where M is a TM and w is a string. However, f gives only the encoding of a TM, so for whatever input, the output is never in $HALT_{TM}$.

ii. Type 2 error is not present as explained above – for whatever input, the output is not in $HALT_{TM}$.

iii. Type 3 error is present because we cannot have an algorithm that decides if some TM M accepts w in the if statement. To put it another way, A_{TM} is undecidable, so we cannot know the answer of whether M accepts w reliably in finite time using a Turing machine.

(b) (*Graded for correctness*) $\{w \mid w \in \{0,1\}^*\} \leq_m \{ww \mid w \in \{0,1\}^*\}$ and $f(x) = xx$ for each $x \in \{0,1\}^*$.

Solution:

i. Notice $\{w \mid w \in \{0,1\}^*\} = \{0,1\}^*$. Type 1 error is not present because $\forall\ x \in \{0,1\}^*, f(x) = xx \in \{ww \mid w \in \{0,1\}^*\}$

ii. Type 2 error is not present because every string is inside $\{0,1\}^*$. So, we cannot pick out some string not inside $\{0,1\}^*$.

iii. Type 3 error is not present because we can easily design a TM that copies whatever that is on its tape twice in finite time.

iv. So this construction is in fact correct.

(c) (*Graded for correctness*) $HALT_{\text{TM}} \leq_m EQ_{\text{TM}}$ with

$$f(x) = \begin{cases} \langle \text{start} \rightarrow (q_{\text{acc}}), M_w \rangle & \text{if } x = \langle M, w \rangle \text{ for a Turing machine } M \text{ and string } w \\ \langle \text{start} \rightarrow (q_{\text{acc}}), \text{start} \rightarrow (q_{\text{rej}}) \quad (q_{\text{acc}}) \rangle & \text{otherwise.} \end{cases}$$

Where for each Turing machine M, we define

$M_w =$ "On input y
1. Simulate M on w.
2. If it accepts, accept.
3. If it rejects, reject."

Solution:

Note that the provided construction reduces A_{TM} to EQ_{TM}.

(a) Type 1 error is present because consider a TM M that starts in the reject state. This TM is a decider, so $\langle M, w \rangle \in HALT_{TM}$ $\forall w$. However, since M rejects all strings, we will always hit the third line of M_w and reject any input so M_w also has empty language. Clearly, $f(\langle M, w \rangle) \notin EQ_{TM}$ as it is the encoding of two TMs where the first one accepts every string and the second one rejects every string.

(b) Type 2 error is not present. Suppose $x \notin HALT_{TM}$, then either it doesn't type check and we output something not inside EQ_{TM}, or it type checks as $\langle M, w \rangle$ but M loops on w. If that's the case, M_w will loop on all strings as it gets stuck on step 1 thus having empty language. So $f(x)$ outputs a pair of TM where the first machine accepts everything but the second machine accepts nothing so not in EQ_{TM}.

(c) Type 3 error is not present because type checking and constructing M_w both take finite time.

3. **Using mapping reductions** (14 points): Consider the following computational problems we've discussed

$$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a Turing machine, } w \text{ is a string and } M \text{ accepts } w\}$$
$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a Turing machine, } w \text{ is a string and } M \text{ halts on } w\}$$
$$E_{TM} = \{\langle M\rangle \mid M \text{ is a Turing machine and } L(M) = \emptyset\}$$
$$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are both Turing machines and } L(M_1) = L(M_2)\}$$

and the new computational problem

$$One_{TM} = \{\langle M\rangle \mid M \text{ is a Turing machine and } M \text{ accepts exactly one string}\}$$

(a) (*Graded for correctness*) Give an example of a string that is an element of One_{TM} and a string that is not an element One_{TM} and briefly justify your choices.

Solution:
Define M_ε by the following high-level description:
M_ε = "On input w: if $w = \varepsilon$, accept. Otherwise, reject".
$\langle M_\varepsilon\rangle \in One_{TM}$. It is clear that M_ε only accepts ε, so its string encoding will be an element in One_{TM}.

Define $M_\varnothing$ by the following high-level description:
$M_\varnothing$ = "On input w: reject."
$\langle M_\varnothing\rangle \in One_{TM}$. It is clear that $M_\varnothing$ rejects all strings, so it does not accept exactly one string, so its string encoding will not be an element in One_{TM}.

(b) (*Graded for completeness*) Prove that One_{TM} is not decidable by showing that $A_{TM} \leq_m One_{TM}$.

Solution:
Define F

i. on input x, type check if $x = \langle M, w\rangle$, output some $const_{out} \notin ONE_{TM}$ iff doesn't type check.
ii. construct M_{new} such that on input y
iii. if $y \neq w$, reject
iv. Otherwise, run M on w
v. if M accepts, accept. if M rejects, reject.
vi. output $\langle M_{new}\rangle$

Every step of this function can be done in finite time by a TM, so the function is computable. Then, for correctness,
If $x \in A_{TM}$, M_{new} will reject all strings other than w. On w, it will run M on w, which will accept. Thus M_{new} accepts only w and $\langle M_{new}\rangle \in ONE_{TM}$

On the other hand, if $x \notin A_{TM}$, it either doesn't type check, in which case $const_{out}$ is defined to be not inside ONE_{TM}, or we have $\langle M, w\rangle$ with M not accepting w. Now M_{new} will also not accept w and its language is empty. This is to say $M_{new} \notin ONE_{TM}$.

(c) (*Graded for correctness*) Give a different proof that One_{TM} is not decidable by showing that $HALT_{TM} \leq_m One_{TM}$.

Solution:
same as above, but in step v, if M rejects, accept.

The proof is also very similar:

If $x \in HALT_{TM}$, M_{new} will reject all strings other than w. On w, it will run M on w, which will halt. Thus M_{new} accepts only w and $\langle M_{new}\rangle \in ONE_{TM}$

On the other hand, if $x \notin HALT_{TM}$, it either doesn't type check, in which case $const_{out}$ is defined to be not inside ONE_{TM}, or we have $\langle M, w\rangle$ with M looping w. Now M_{new} will also loop on w and its language is empty. This is to say $M_{new} \notin ONE_{TM}$.

(d) (*Graded for completeness*) Is One_{TM} recognizable? Justify your answer.

Solution:

This is not recognizable because we can reduce A_{TM} to $\overline{ONE_{TM}}$. Once we have established that, we also have $\overline{A_{TM}} \leq_m ONE_{TM}$, which is to say ONE_{TM} is not recognizable because $\overline{A_{TM}}$ is not recognizable.

i. on input x, type check if $x = \langle M, w\rangle$, output some $const_{out} \notin \overline{ONE_{TM}}$ iff doesn't type check.
ii. construct M_{new} such that on input y
iii. if $y = \varepsilon$, accept
iv. if $y = 0$, run M on w
v. if M accepts, accept. if M rejects, reject.
vi. otherwise, reject
vii. output $\langle M_{new}\rangle$

Intuitively, we output a machine whose language has size 1 if M does not accept w and we output a machine whose language has size 2 if M accepts w. This construction finishes in finite time so is computable.

Formally, if $x \in A_{TM}$, M_{new} will reject all strings that are not $\varepsilon, 0$. It will accept ε and on 0, it will run M on w, which will accept. Thus M_{new} accepts only $\varepsilon, 0$, a total of 2 strings so $\langle M_{new}\rangle \in \overline{ONE_{TM}}$

On the other hand, if $x \notin A_{TM}$, it either doesn't type check, in which case $const_{out}$ is defined to be not inside $\overline{ONE_{TM}}$, or we have $\langle M, w\rangle$ with M not accepting w. Now M_{new} only accepts ε because on 0 it will either loop or reject, mimicking the behavior of M on w. This is to say $M_{new} \notin \overline{ONE_{TM}}$.

4. **Examples of languages** (9 points):

For each part of the question, use precise mathematical notation or English to define your examples and then briefly justify why they work.

(a) (*Graded for correctness*) Two undecidable languages L_1 and L_2 over the same alphabet whose union $L_1 \cup L_2$ is decidable, or write **NONE** if there is no such example (and explain why).

Solution:
Let $L_1 = E_{TM}$, and $L_2 = \overline{E_{TM}}$. It has been shown that both of these languages are undecidable. Since they are complements of one another, it is true that $L_1 \cup L_2 = \Sigma^*$. Their union is decidable, so this is a valid example.

(b) (*Graded for correctness*) A regular language L_3 and an unrecognizable language L_4 over the same alphabet whose intersection $L_3 \cap L_4$ is unrecognizable, or write **NONE** if there is no such example (and explain why).

Solution:
Let $L_3 = \Sigma^*$, and $L_4 = E_{TM}$, where L_3 is regular, and L_4 is unrecognizable. Since $L_4 \subset L_3$, it is true that $L_3 \cap L_4 = L_4$. Since the intersection, L_4, is unrecognizable, this is a valid example.

(c) (*Graded for correctness*) A co-recognizable language L_5 that is NP-complete, or write **NONE** if there is no such example (and explain why). Recall the definition: A language L over an alphabet Σ is called **co-recognizable** if its complement, defined as $\Sigma^* \setminus L = \{x \in \Sigma^* \mid x \notin L\}$, is Turing-recognizable.

Solution:

Let $L_5 = 3SAT$. It has been shown that this language is NP-complete. Notice that L_5 is decidable, so it must be recognizable, and its complement must also be recognizable. Thus, we have L_5 is both co-recognizable and NP-complete.