Indian Institute of Technology Kanpur
CS340: Theory of Computation, 17–18

Student Name: Gurpreet Singh
Roll Number: 150259
Date: November 7, 2017

ASSIGNMENT 4

Question 1

Part 1

$$L_1 = \left\{ \langle M, N \rangle \,\middle|\, M \text{ takes fewer steps on } \epsilon \text{ than } N \right\}$$

This language is undecidable. We can prove this by finding a valid reduction from A_{TM} to L_1. Therefore, for all machines M', there exists some machines M and N such that M' accepts $w \iff M$ takes fewer steps on ϵ than N.

Description of the Reduction Function f

Input: $\langle M', w \rangle$

(a) If M' accepts w, then

  (i) Construct M such that M accepts ϵ in constant number of (one) steps

  (ii) Construct N such that N loops forever on input ϵ

(b) Else

  (i) Construct M such that M loops forever on input ϵ

  (ii) Construct N such that N accepts ϵ in constant number of (one) steps

Proof of Correctness

If M' accepts w, then we have two machines M and N such that machine M takes only one step whereas N never halts on input ϵ. This satisfies the definition of the reduction function.

If M' does not accept w, then we have two machines M and N such that machine N takes only one step whereas M never halts on input ϵ. Again, this satisfies the definition of the reduction function.

Hence our reduction function is valid. Therefore, L_1 is undecidable.

Part 2

This language is decidable. The concept used is that for limited number of steps, we can have only limited number of inputs, as for any input with length $> 2^{340}$ will be read only till the first 2^{340}(let= k) characters, and hence the total number of strings we can have is $|\Sigma|^k$. For any string, we can also check if the machine halts in k steps or not. Hence, our turing machine will be as follows

Description of the Turing Machine D

Input: $\langle M \rangle$

(a) For all strings with size $|w| \leq k$, check if M halts on w within k steps. If any such string found, *accept*

(b) If not accepted yet, *reject*

Since this is a halting turing machine, we can say that L_2 is decidable.

Part 3

Since for any Turing Machine, there are always infinitely many Turing Machines which are equivalent, we can always accept if the input indeed is a turing machine.

Description of the Turing Machine D

Input: $\langle M \rangle$

(a) If M is a turing machine, *accept*

(b) Else, *reject*

Part 4

$$L_4 \quad = \quad \left\{ \langle M, N \rangle \;\middle|\; L(M) \cap L(N) \text{ is infinite} \right\}$$

This language is undecidable. We will prove this by finding a valid reduction such that $\overline{H_{TM}} \leq L_4$

The reduction function f

Input: $\langle M', w \rangle$

(a) Construct a turing machine M such that on input x

  (i) If M halts on w (in say k steps), *accept* x

  (ii) If M does not halt on w, *reject*

(b) Construct a turing machine N such that N accepts all strings x

Output: $\langle M, N \rangle$

From the construction of the reduction function, it is clear that our reduction function is valid.

Therefore, $f(\langle M', w \rangle) \in L_4 \iff \langle M', w \rangle \in \overline{H_{TM}}$, hence, we can say that $\overline{H_{TM}} \leq L_4$.

Since we already know that $\overline{H_{TM}}$ is undecidable, therefore, L_4 is also undecidable.

Question 2

We can show that $\overline{A}_{TM}$ reduces to $\overline{REG}_{TM}$. In order to show this, we can equivalently show that A_{TM} reduces to REG_{TM}.

This is because $A_{TM} \leq_m REG_{TM} \iff \overline{A}_{TM} \leq_m \overline{REG}_{TM}$

We can find a reduction function f such that for every machine M and string w, $f(\langle M, w \rangle) = \langle M' \rangle$ such that if M accepts $w \iff L(f(\langle M, w \rangle))$ is regular.

Description of the Reduction Function f

Input: $\langle M, w \rangle$

(a) Construct a machine M', such that on input x

  (i) If M accepts w, then M' accepts x

  (ii) Else, M' accepts x if x is of the form $0^n 1^n$i else rejects

Output: $\langle M' \rangle$

Proof of Correctness

The proof is straightforward. If M accepts w, then we create a machine M' which accepts every input. Clearly, $L(M')$ is a regular language.

Otherwise, M' accepts all strings of the form 0^n1^n. Therefore $L(M')$ is non-regular. Hence the reduction is well defined.

Since we have found a valid reduction from $A_{TM} \leq_m REG_{TM}$. Therefore $\overline{A}_{TM} \leq_m \overline{REG}_{TM}$. Since $\overline{A}_{TM}$ is turing unrecognizable, hence REG_{TM} is also not co-turing recognizable.

Question 3

We can show that A is turing recognizable whereas B is not.

Part A

We can find a turing machine M, that accepts the language A (not halting)

Description of the Turing Machine M

Input: $\langle M \rangle$

(a) Set a variable $count = 0$

(b) For all strings of the alphabet Σ of the machine M, generate all strings $w \in \Sigma*$. Loop

  (i) For string w, if M accepts w, add 1 to count
  (ii) If $count = 340$, *accept*

(c) If not accepted yet, *reject*

Since this is a valid turing machine, and $L(M) = A$, we can say that A is turing recognizable.
We can also claim using a similar turing machine that $A' = \{\langle M \rangle \mid |L(M)| \geq 341\}$ is also turing recognizable.

Part B

Since, for an undecidable language L which is turing recognizable, $\overline{L}$ will be turing unrecognizable.

Hence, in order to prove B is turing unrecognizable, we can instead show that A' is undecidable as $\overline{B} = A'$ and we have already shown that prA is turing recognizable.

We can show that A' is turing unrecognizable by showing that there exists a reduction from A_{TM}. *i.e.* for every machine M, there exists another machine M', if M accepts $w \iff |L(M')| > 340$

Description of the Reduction Function f

Input: $\langle M, w \rangle$

(a) Construct a turing machine M' such that on input x

  (a) If M accepts w, *accept*
  (b) If M does not accept w, *reject*

Output: $\langle M' \rangle$

Proof of Correctness

The proof is straightforward, if M accepts w, then we create a turing machine M' which accepts all inputs, therefore $|L(M')| > 340$

If the machine M does not accept w, then we create a machine which does not accept any string, hence $|L(M')| \leq 340$

Since we have proved A', and hence A to be undecidable, we can say that B is turing unrecognizable.

Question 4

Part A

For the $LPATH$ problem, we can easily solve in NP time, by finding all paths from s to t which can be done in $2^{|E(G)|}$ time. For each path, we can find in linear time if the length is $\leq k$.

Therefore, $LPATH \in$ NP$\dot{\text{W}}$e further need to prove that $LPATH$ is NP-Hard. For this, we will reduce the problem of Hamiltonian Path to $LPATH$, *i.e.* $HAMPATH \leq_m LPATH$.

Description of the Reduction Function f

Input: $\langle G, s, t \rangle$

(a) Construct a graph G' such that

  (i) $V(G') = V(G)$

  (ii) $E(G') = E(G)$

(b) Set $k = |V(G)| - 1$

(c) Set $s' = s$ and $t' = t$

Output: $\langle G', s', t, k \rangle$

Proof of Correctness

If there is a Hamiltonian Path in the graph G, then we can say that there is a path of length $|V(G)| - 1$ in G. Hence, there is a path of length k in the graph G' as they are the same graphs.

If there is a simple path in G' such that the length is at least $k = |V(G)|$. Since this is maximum length any path can have, hence we say that there is a path of length $|V(G)| - 1$. Since this path will pass through every vertex, henc e is a Hamiltonian Path.

Therefore, our reduction function is valid. Since $HAMPATH$ is NP-Complete (as discussed in class), we can say that $LPATH$ is NP-Hard. However, since $LPATH \in$ NP, therefore $LPATH$ is NP-Complete.

Part B

Reference: Introduction to the Theory of Computation — Michael Sipser (for the proof that $VertexCover$ is NP-Complete)

First we need to show that the $VertexCover$ problem is NP-Complete. Clearly $VertexCover$ is in NP, as we can check for all subsets of the graph vertices in polynomial time if it is a vertex cover. Hence, we only need to show that $VertexCover$ is NP-Hard.

In order to show this, we prove the reduction $3SAT \leq_m VertexCover$.

Description of the Reduction Function

Input: $\langle \phi \rangle$

(a) Construct a graph G with the following properties

  (i) For every clause, create three vertices, corresponding to each literal of this clause.with edges between each of these vertices

  (ii) For every distinct literal (along with it's complement if it exists), create a vertex and connect to each vertex corresponding to the same literal

(b) Set $k = l + 2m$ where l is the number of distinct literals (not including complements) and m is the number of the number of clauses.

Output: $\langle G, k\rangle$

Hence, $VertexCover$ is NP-Complete. We can use this to prove that DS is also NP-complete, through a reduction $VertexCover \leq_m DS$

Reference: https://people.cs.umass.edu/ barring/cs311/exams/finpracsol.html
We can easily see that $DS \in$ NP, as we can non-deterministically find all the subsets of the vertex set, and for each subset, check if it is a valid vertex set *i.e.* satisfies the required property, which can be done in polynomial time

Note, we can make a small change in the definition of $VertexCover$ such that the new definition is

$$VertexCover = \left\{\langle G, k\rangle \;\middle|\; \text{G is connected and} \,\exists S, |S| \leq k \,\text{and}\, S \,\text{is a vertex cover of}\, G\right\}$$

Since this does not add to the complexity of the problem, nor reduces it, we can still claim the this language is also NP-Complete. Hence, now we can find a reduction function $VertexCover \leq_m DS$

The graphs constructed are undirected
Description of the Reduction Function

Input: $\langle G, k\rangle$

(a) Construct a graph G'

(b) For every vertex v_i in G, add a vertex v_i in G'

(c) For every edge $e_i(u, v)$ in G, add a vertex e_i in G'. Add three edges, (u, v), (u, e_i) and (e_i, v) in G'

(d) Set $k' = k$

Output: $\langle G', k\rangle$

Proof of Correctness

There are two sides. Firstly, if there is a vertex cover of size atmost k, then there is a subset $S' \subset V(G')$ which satisfies the property of DS. Let the vertex cover in G be the subset S. Then we can choose S' to be S itself.

This is a valid set, since for every edge, we clearly have a neighbour in S', since for every vertex e_i corresponding to the edge u, v, we will have either u or v in the vertex cover, (say u without the loss of generality). Hence, both v and e_i have a neighbour in S. Also, since the graph G and hence the graph G' are connected, all vertices will have at least one neighbour in S' since all vertices must have at least one edge.

Hence, if there is a VertexCover of size k in $G \implies$ there is a set $S' \subset V(G')$ following the property of DS.

For the opposite case, we assume that there is a set $S' \subset V(G')$ which follows the property of DS. Hence, size of $S' \leq k$. Therefore, we can add vertices to S' so that for every edge u, v (corresponding to the edges in G) or there is either one of u, v of e_i in S'.

Also, for every vertex of the form e_i corresponding to edge u, v, replace it by either u or v. Since, this does not change the property of S'. Also, S' is clearly a vertex cover in G, since for every edge we have a vertex in the set S. Therefore the opposite side holds too.

Hence we have proved that $VertexCover \leq_m DS$. Therefore, DS is both NP-Hard and $DS \in$ NPṪherefore, DS is NP-Complete.