CSC363H5 Winter 2016 Assignment 2

Name: Akhil Gupta
SN: 1000357071

Question #	Score
1	
2	
3	
4	
5	
Total	

Acknowledgements:

”I declare that I have not used any outside help (excluding the textbook, the notes on the course website, the teaching assistants, and the instructor) in completing this assignment.”

Name: Akhil Gupta Date: February 7, 2016

Q1. In this question we give an alternative proof that *nondeterministic* Turing Machines compute exactly the languages in *SD*. (For the definition of a nondeterministic Turing Machine, check the Sipser book or the notes from Tutorial 2). Fix an alphabet Σ. Give a **direct** proof that for any language L, there is a nondeterministic Turing Machine M such that $L = \mathcal{L}(M)$ if and only if there is a computable relation $R \subseteq \Sigma^* \times \Sigma^*$ such that

$$x \in L \Leftrightarrow \exists y \in \Sigma^* : R(x, y).$$

Answer: (adapted from Professor Robert's Lecture 4 notes)

Let $\Sigma = \{0, 1\}$. First we prove the "if" direction. Let L be a semi-decidable language, and let M_0 be a Non-Deterministic Turing Machine such that $\mathcal{L}(M_0) = L$. Theorem 2 from Tutorial 2 Notes states that:

For any language L, $L \in SD$ if and only if there is an Non-Deterministic Turing Machine that computes L

For any $(x, y) \in \Sigma^*$ x Σ^*, define R by

$R(x, y) \Leftrightarrow y$ encodes all possible computations of M_0 on x.

Next we show that $x \in L$ if and only if $(x, y) \in R$.

If $x \in L$, then M_0 accepts x, and so there must be some sequence of configurations that leads to the string y encoding an accepting computation of M_0 on x. Thus if $x \in L$ then there is a y such that $R(x, y)$ holds. Conversely, if there is a y that encodes an accepting computation of M_0 on x, then there must be some sequence of configurations that leads M_0 to accept x. So if $R(x, y)$ holds then $x \in L$.

The algorithm for R operates as follows:

1. On input $x \in \Sigma^*$.

2. Nondeterministically construct arbitary strings $y_1, y_2, y_3, \ldots y \in \Sigma^*$ and encode all possible configurations of the Nondeterministic Turing Machine M_0 i.e create a computational tree where the children nodes of a node are M_0's next configurations

3. Repeat the following:

(a) Simulate M_0 on x for one step, and check if the configuration of M_0 on x is encoded at some branch of y's computational tree. If not, reject.

(b) If this is the last configuration encoded in y, check that M_0 has actually halted and that there is a sequence of configurations that leads to an accept state i.e. M_0 accepts x. If so, accept. Otherwise, reject.

Clearly the algorithm above halts on all of its inputs i.e. all the branches of y's computational tree halt, as it rejects as soon as the simulation of M_0 on x does not find a match in y's computational tree. Moreover, the algorithm only accepts if and only if there is a sequence of accept state configurations that leads to string y that encodes an accepting computation that M_0 accepts x, and thus if and only if $(x, y) \in R$.

Now we prove the "only if" direction in the statement of the theorem. Suppose that such a decidable relation R exists, and we use the relation R to construct a Non-Deterministic Turing Machine M_0 which recognizes the language L. Let M be the Non-Determinsitic Turing Machine deciding R.

The machine M_0 will run the following algorithm:

1. On input $x \in \Sigma^*$.

2. For each $y \in \Sigma^*$ in y's computational tree (from the algorithm above).

(a) Simulate M on the input (x, y). If there exists some sequence of accept states that leads to x, then

M accepts, Otherwise continue.
Since R is decidable the nondeterminsitic machine M halts on all inputs i.e. all branches halt on all inputs, so each simulation step in the for loop will halt. The algorithm above accepts $x \in \Sigma^*$ if and only if there is a $y \in \Sigma^*$ in y's computational tree such that $R(x, y)$ holds; by assumption, this is equivalent to $x \in L$. If no such y exists, then the algorithm will never halt. Thus $\mathcal{L}(M_0) = \mathcal{L}$ and so $L \in SD$.

Q2.a Let $\Sigma = \{0, 1\}$. For each of the following languages $L \subseteq \Sigma^*$, classify L with respect to D, SD, $coSD$. That is, for each language L and for each class $\mathsf{C} \in \{D, SD, coSD\}$, prove that L is in C or that L is not in C. **You may not use Rice's Theorem.**

1. $L_1 = \{(\langle M\rangle, \langle i\rangle) | M \text{ is a TM}, i \in \mathbb{N}, \text{ M accepts all strings of length } i\}$
Answer: We need to show that for all $i \in \mathbb{N}$, M accepts all strings of length i. According to Professor Robert's A1Q4 Solutions, if we are given $\langle i\rangle$, we can compute $f(i)$ (f is a computable function). So if we can compute $f(i)$, we can also do a little more work and get i. Hence, in order to solve this, we will reduce the Halting Problem to L_1. The Halting Problem is defined as

$$\{(\langle M\rangle, w) | M \text{ is a TM and } M \text{ halts on } w\}$$

We know that the Halting Problem $\in SD$, therefore, we will prove that $L_1 \in SD$.
Let $(\langle M, w\rangle)$ be any pair such that M is a TM, and we show how to construct a Turing Machine $M'_{(M,w)}$ such that $(\langle M, w\rangle) \in \overline{\text{HALT}} \Leftrightarrow (\langle M'_{(M,w)}\rangle) \in L_1$.
Consider the following algorithm $M'_{(M,w)}$ which is computable from the pair $(\langle M\rangle, w)$.
Algorithm for $M'_{(M,w)}$:
1. On input $x \in \Sigma^*$, skip the input and write w.
2. Simulate M on w for at most i steps (i can be computed from $\langle i\rangle$).
3. Accept if and only if $|w| = i$.

If $(\langle M, w\rangle) \in \text{HALT}$, then M halt on $w \Leftrightarrow$ the algorithm $M'_{(M,w)}$ accepts all input including all strings of length $i \Leftrightarrow (\langle M'_{(M,w)}\rangle) \in L_1$. If $(\langle M, w\rangle) \notin \text{HALT}$, then M does not halt on $w \Leftrightarrow$ the algorithm $M'_{(M,w)}$ does not accept any input particularly ones of length $i \Leftrightarrow (\langle M'_{(M,w)}\rangle) \notin L_1$. It follows that HALT $\leq_m L_1$, and since HALT $\in SD$, this shows that $L_1 \in SD$. Since $L_1 \in SD$, it immediately shows that $L_1 \notin coSD$ and $\notin D$ since $D = SD \cap coSD$.

Q2.b

2. $L_2 = \{(\langle M\rangle, x) | M \text{ is a TM and } M \text{ halts on } x \text{ with 11111 written on the tape}\}$
Answer: We solve this problem by reducing the Halting Problem to L_2. The Halting Problem is defined as

$$\{(\langle M\rangle, w) | M \text{ is a TM and } M \text{ halts on } w\}$$

We know that the Halting Problem $\in SD, \notin D$, therefore, we will prove that $L_2 \in SD, \notin D$.
Let $(\langle M\rangle, w)$ be any pair such that M is a TM, and we show how to construct a Turing Machine $M'_{(M,w)}$ such that $(\langle M\rangle, w) \in \text{HALT} \Leftrightarrow (M'_{(M,w)}) \in L_2$.
Consider the following algorithm $M'_{(M,w)}$ which is computable from the pair $(\langle M\rangle, w)$.
Algorithm for $M'_{(M,w)}$:
1. On input $x \in \Sigma^*$.
2. Skip the input and write w on the tape. Then simulate M on w.

3. Halt and accept if any of the simulations have 11111 written on its tape and blanks everywhere else, else reject.

If $(\langle M\rangle, w) \in$ HALT, then M halts on w so the algorithm $M'_{(M,w)}$ halts on input x with 11111 written on the tape.

This means $M'_{(M,w)} \in L_2$. On the other hand, if $M'_{(M,w)} \in L_2$, then by definition of $M'_{(M,w)}$, it is easy to see that $M'_{(M,w)}$ halts on any input if and only if M halts on w. Thus $(\langle M\rangle, w) \in$ HALT. It follows that HALT $\leq_m L_2$, and since HALT $\in SD$ and $\notin D$, this shows that $L_2 \in SD$ and $\notin D$. Since $L_2 \in SD$, it immediately shows that $L_2 \notin coSD$ as well since $D = SD \cap coSD$ (from Professor Robert's lecture notes).

Q2.c

3. $L_3 = \{\langle M\rangle | M$ is a TM and $\exists i \in \mathbb{N}$: M accepts all strings of length $i\}$

Answer: We need to show that there exists some $i \in \mathbb{N}$ for which M accepts all strings of length i. In order to solve this, we will reduce the $\overline{\text{Halting Problem}}$ to L_3. The $\overline{\text{Halting Problem}}$ is defined as

$$\{(\langle M\rangle, w) | M \text{ is a TM and } M \text{ does not halt on } w\}$$

We know that the $\overline{\text{Halting Problem}} \notin SD$, therefore, we will prove that $L_3 \notin SD$.

Let $(\langle M, w\rangle)$ be any pair such that M is a TM, and we show how to construct a Turing Machine $M'_{(M,w)}$ such that $(\langle M, w\rangle) \in \overline{\text{HALT}} \Leftrightarrow (\langle M'_{(M,w)}\rangle) \in L_3$.

Consider the following algorithm $M'_{(M,w)}$ which is computable from the pair $(\langle M\rangle, w)$.

Algorithm for $M'_{(M,w)}$:

1. On input $x \in \Sigma^*$.
2. Skip x and write w. Then simulate M on w for all inputs for at most $|x|$ steps.
3. Reject if M halts on w within $|x|$ steps and accept otherwise.

If $(\langle M, w\rangle) \in \overline{\text{HALT}}$, then M does not halt on $w \Leftrightarrow$ the algorithm $M'_{(M,w)}$ accepts all input including all strings of length $i \Leftrightarrow (\langle M'_{(M,w)}\rangle) \in L_3$. On the other hand, if $M'_{(M,w)} \in L_3$, then by definition of $M'_{(M,w)}$, it is easy to see that $M'_{(M,w)}$ halts and accepts all input of length i if and only if M halts on $w \Leftrightarrow (\langle M\rangle, w) \in \overline{\text{HALT}}$. It follows that $\overline{\text{HALT}} \leq_m L_3$, and since $\overline{\text{HALT}} \notin SD$, this shows that $L_3 \notin SD$. Since $L_3 \notin SD$, it immediately shows that $L_3 \in coSD$ and $\notin D$ since $D = SD \cap coSD$.

part d) is on the next page →

Q2.d

4. $L_4 = \{\langle M\rangle | M$ is a TM and there is an $x \in \Sigma^*$ beginning with 0 such that M does not accept $x\}$
Answer: We have to show that there exists some $x \in \Sigma^*$ beginning with 0 such that M does not accept x. In order to solve this, we will reduce the $\overline{\text{Halting Problem}}$ to L_4. The $\overline{\text{Halting Problem}}$ is defined as

$$\{(\langle M\rangle, w) | M \text{ is a TM and } M \text{ does not halt on } w\}$$

We know that the $\overline{\text{Halting Problem}} \notin SD$, therefore, we will prove that $L_4 \notin SD$.
Let $(\langle M, w\rangle)$ be any pair such that M is a TM, and we show how to construct a Turing Machine $M'_{(M,w)}$ such that $(\langle M, w\rangle) \in \overline{\text{HALT}} \Leftrightarrow (\langle M'_{(M,w)}\rangle) \in L_4$.
Consider the following algorithm $M'_{(M,w)}$ which is computable from the pair $(\langle M\rangle, w)$.
Algorithm for $M'_{(M,w)}$:
1. On input $x \in \Sigma^*$.
2. Simulate M on w.
3. If M halts on w, $M'_{(M,w)}$ accepts.

If $(\langle M, w\rangle) \in \overline{\text{HALT}}$, then M does not halt on $w \Leftrightarrow$ the algorithm $M'_{(M,w)}$ does not accept any input particularly ones beginning with $0 \Leftrightarrow (\langle M'_{(M,w)}\rangle) \in L_4$. If $(\langle M, w\rangle) \notin \overline{\text{HALT}}$, then M halts on $w \Leftrightarrow$ the algorithm $M'_{(M,w)}$ accepts any input particularly ones beginning with $0 \Leftrightarrow (\langle M'_{(M,w)}\rangle) \notin L_4$. It follows that $\overline{\text{HALT}} \leq_m L_4$, and since $\overline{\text{HALT}} \notin SD$, this shows that $L_4 \notin SD$.

The complement of $L_4 = \{\langle M\rangle | M$ does not encode a TM or for all $x \in \Sigma^*$ beginning with $0, M$ accepts $x\}$.
We now prove that $\overline{L_4} \notin SD$. Assume that $\langle M\rangle$ is a well-formed encoding of a turing machine then according to the description of the language, it accepts all strings in Σ^* beginning with $0 \Leftrightarrow \overline{L_4}$ contains infinitely many strings i.e. $\mathcal{L}(\overline{L_4}) =$ infinite. So now we have to prove $\overline{L_4} = \{\langle M\rangle | M$ is a TM and $\mathcal{L}(\overline{L_4}) =$ infinite$\} \notin SD$. From Professor Robert's tutorial 5 notes, we proved that INF $\notin SD$ via a reduction from A_{TM}*.
(INF $= \{\langle M\rangle | M$ is a TM that accepts infinitely many strings$\}$).
So now we have that $L_4 \notin SD$ and $\overline{L_4} \notin SD$, it immediately shows that $L_4 \notin coSD$ and subsequently $\notin D$ since $D = SD \cap coSD$. Therefore, $L_4 \notin SD, \notin coSD$, and therfore $\notin C$.