Problem Set 2

CSC 463

Due by February 14, 2020, 2pm

Each problem set counts for 10% of your mark. You may consult with others concerning the general approach for solving problems on assignments, but you must write up all solutions entirely on your own. Copying assignments is a serious academic offense and will be dealt with accordingly.

You may use any results discussed in lecture, the course textbook, previous problem set, or tutorial. You are encouraged to write precisely and concisely; it should be possible to write your solutions to the problem set within a page per question. Submit your work online by uploading a pdf file or image of your solutions onto Crowdmark.

In all problems, you can assume that you are given valid encodings of Turing machines, grammars etc. as input to the problem. You do not need to check for invalid encodings as part of your description of a Turing machine or reduction.

1. Let $w^R$ be the reverse of a string $w \in \Sigma^*$. Let $R$ be the set of Turing machine encodings $\langle M \rangle$ such that $w \in L(M)$ if and only if $w^R \in L(M)$. Is $R$ semi-decidable? Prove your answer.

Solution: The language $R$ is not semi-decidable since there is a reduction $\overline{A_{TM}} \leq_m R$. Let $\langle M_1, w \rangle$ be a Turing machine $M_1$ and a string $w$. Construct a Turing machine $M_2$ that uses $\langle M_1, w \rangle$ with the following description. $M_2$ on input $x$:

1. If $x = 01$ or $x = 10$ accept $x$
2. If $x = 110$, run $M_1$ on input $w$, and accept $x$ if $M_1$ accepts $w$.
3. Reject all strings $x \notin \{01, 10, 110\}$.

From the construction above, if $M_1$ accepts $w$, $L(M_2) = \{01, 10, 110\}$ and otherwise $L(M_2) = \{01, 10\}$. Hence if $M_1$ accepts $w$, $\langle M_2 \rangle \notin R$ since $01 \notin L(M_2)$ in this case, and otherwise $\langle M_2 \rangle \in R$. Hence we have shown a reduction from $\overline{A_{TM}}$ to $R$, which shows that $R$ is not semidecidable.

2. A Turing machine $M$ has a useless state if there is some state $q$ that is never entered on $M$’s computation beginning on any string $x$. Let $U = \{\langle M \rangle : M$ is a Turing machine with a useless state$\}$.

(a) Show that $U$ is not decidable.

Solution: Note that $A_{TM,\varepsilon} = \{\langle M \rangle : M$ accepts the empty string $\varepsilon\}$ is undecidable, by reduction from $A_{TM}$ or Rice’s Theorem. Hence, we construct a reduction $A_{TM,\varepsilon} \leq_m U$ to show the claim since if $U$ is undecidable, then $U$ is undecidable as well.

Given a Turing machine $M$ with non-halting states $q_0, \ldots, q_k$, we create a Turing machine $N$ simulating $M$ with additional states $q'_0, \ldots, q'_k$ combined with the original state set and transition function of $M$. There is also an additional symbol $#$ that $N$ can write
not in the tape alphabet of $M$. The initial state of $N$ will be $q_0'$ and we will ensure that $N$ uses all non-halting states and the reject state in the following way. If in state $q_0'$, $N$ reads a non-blank symbol, $N$ rejects. Otherwise, in state $q_0'$, if $N$ reads blank, $N$ writes $\#$, moves left, and enters state $q_1$. Next, for any state $1 \leq i \leq k$, if $q_i$ reads $\#$, it moves left and writes $\#$ and enters state $q_i'$. If $q_i'$ reads any symbol, it moves left, writes the same symbol that was read, and enters state $q_{i+1}$ (with $q_{k+1} = q_0$). Finally, if $q_0$ reads $\#$, $N$ erases it, moves left, and remains in state $q_0$. Hence, if $N$ is a Turing machine with a transition function defined in this way, it rejects any non-empty string $x$, and on an empty string $\epsilon$, it enters states $q_0'$, then $q_i, q_i'$ for $1 \leq i \leq k$, and finally $q_0$ with the tape blank after it has visited all other states. Hence at this point, $N$ can simulate $M$’s computation on a blank tape.  

Therefore, from the construction above, we have constructed a Turing machine $N$ with a useless state (the accepting state $q_{\text{accept}}$) from a Turing machine $M$ that does not accept the empty string $\epsilon$, and otherwise all states of $N$ are used since $N$ rejects all non-empty inputs and $N$ visits all non-rejecting states on input $\epsilon$, if $M$ accepts $\epsilon$. Hence, we have shown a reduction $A_{T,M,\leq_m} \leq_m \bar{U}$, which shows that $U$ is undecidable.

(b) Show that $U$ is co-semidecidable.

**Solution:** We need to show that checking if a Turing machine $\langle M \rangle$ uses all of its states to show that $\bar{U}$ is semidecidable. Recall that $\bar{U}$ is semidecidable if we can show that there is decidable binary relation $R$ where $x \in \bar{U}$ if and only if there is some $y \in \Sigma^*$ with $(x,y) \in R$. Let $R$ be the relation where $x$ is a Turing machine encoding $\langle M \rangle$ and $y$ is an encoding of pairs $\langle x_1, C_1, \ldots, x_k, C_k \rangle$ where $C_i$ is a computational history of $M$ on $x_i$ up to the point the machine enters state $q_i$, where $k$ is the number of states of $\langle M \rangle$. The relation $R$ is decidable because each $x_i, C_i$ is finite and the Turing machine $M$ can be simulated and the number of states of $M$ can extracted from its description. Furthermore, by definition, a valid $y$ exists if and only if $\langle M \rangle \in \bar{U}$ using the definition of $\bar{U}$.

3. Prove the following extension of Rice’s theorem. Let $P$ be a non-trivial set of Turing machine encodings satisfying the following two conditions:

(i) If $L(M_1) = L(M_2)$, then $\langle M_1 \rangle \in P$ if and only if $\langle M_2 \rangle \in P$.

(ii) There are Turing machines $M_1$ and $M_2$ satisfying $L(M_1) \subset L(M_2)$ with $\langle M_1 \rangle \in P$ but $\langle M_2 \rangle \notin P$.

An example of $P$ satisfying the previous conditions is $P = \{ \langle M \rangle : |L(M)| \text{ is finite} \}$.

Prove that sets $P$ satisfying the previous conditions are **not semidecidable**. (Hint: Construct a reduction $A_{T,M} \leq_m P$ that uses the Turing machines $M_1, M_2$ given by the second condition.)

**Solution:** Let $P$ be the set in the problem with $\langle M_1 \rangle \in P$ and $\langle M_2 \rangle \notin P$. Let $\langle M, w \rangle$ be a Turing machine $M$ and input $w$. Define a Turing machine $N$ which is a three-tape Turing machine that on input $x$, runs $M$ on $w$ on one tape, and runs $M_1, M_2$ on $x$ on the two other tapes (in other words, $N$ runs $M, M_1, M_2$ in parallel). We say that $N$ accepts $x$ if $M$ accepts $w$ and $M_2$ accepts $x$, or if $M_1$ accepts $x$.

\[1\] we are assuming here that the tape is semi-infinite so that moving left on the first cell is equivalent to staying on the same cell.
Note that by construction of $N$, if $M$ accepts $w$, $L(N) = L(M_2)$ and otherwise $L(N) = L(M_1)$. Hence, $\langle N \rangle \notin P$ if $M$ accepts $w$ since $\langle M_2 \rangle \notin P$, and otherwise if $M$ does not accept $w$, $\langle N \rangle \in P$ since $\langle M_1 \rangle \in P$. Hence, we have constructed a reduction $\overline{A_{TM}} \leq_m P$, showing that $P$ is not semi-decidable.

4. Recall that a context-free grammar $G$ is ambiguous if there is a string $w \in L(G)$ with at least two different leftmost derivations. Determining if a context-free language is ambiguous or not arises in problems related to natural language processing and programming languages. Prove that determining if a context-free language is ambiguous or not is undecidable. (Hint: Reduce from Post Correspondence’s Problem and consult the hint in Sipser, Chapter 5 Exercises.)

Solution: We follow the hint in Sipser. Given an instance of the Post Correspondence Problem (PCP) with $k$ tiles $s_i = [t_i/b_i]$ over an alphabet $\Sigma$, we construct the grammar $G$ with variables $S, T, B$, alphabet $\Sigma \cup \{a_1, \ldots, a_k\}$ for new symbols $a_1, \ldots, a_k$, and rules

$$S \rightarrow T|B,$$

$$T \rightarrow t_iTa_i|t_ia_i,$$

$$B \rightarrow b_iBa_i|b_ia_i$$

for $1 \leq i \leq k$. We need to show that the PCP instance has a match if and only if $G$ is ambiguous, which would prove that the ambiguity problem for CFGs is undecidable.

Firstly note that any derivation in $G$ is leftmost since if there is a derivation, there is at most one unused variable at any step in the derivation. Suppose the PCP instance has a solution $s_{i_1} \ldots s_{i_k}$ with top and bottom string equal to $w$. Hence, there are two different derivations of $w^* = wa_{i_k} \ldots a_{i_1}$. One uses the rules $S \rightarrow T$, $T \rightarrow t_jTa_i$ for $1 \leq j < k$ and $T \rightarrow t_ka_{i_k}$ in the final step. The second uses the rules $S \rightarrow B$, $T \rightarrow b_jBa_i$, for $1 \leq j < k$ and $T \rightarrow b_ka_{i_k}$ in the final step. Both derivations derive the same string $w^*$ since PCP has a solution using the rules of the grammar.

Conversely, suppose there are two different derivations of string some string $s$ in $G$. The string $s$ must be non-empty and we can break it into $s = ww'$ where $w \in \Sigma^*$ and $w' \in \{a_1, \ldots, a_k\}^*$. Furthermore, any ambiguity in $G$ must result from the first step in the derivation using rule $S \rightarrow T|B$, since once the first step is chosen, there is exactly one way to derive a string with suffix $w'$. Hence, there is a derivation of $s$ with one using $S \rightarrow T$ and the other using $S \rightarrow B$. Hence, if $w' = a_{i_k} \ldots a_{i_1}$, the tiling $s_{i_1} \ldots s_{i_k}$ is a solution to the original PCP instance that produces $w$ as a matching string in the top and bottom since both derivations produce $s = ww'$. 

3