# CS 360 — Assignment 5 Solutions

## University of Waterloo, Spring 2018

1. We will show that if we can decide $L$, then we can decide $A_{TM}$. Suppose there exists a Turing machine $R$ that decides $L$. Then we can construct the following machine to decide $A_{TM}$:

   1. On input $\langle M, w\rangle$, where $M$ is a Turing machine and $w$ is an input word, construct the Turing machine $M'$, which operates as follows:

      1. On input $x$, if $x \neq w$, then skip to the next step. Otherwise, simulate $M$ on $w$.

      2. If $M$ rejects $w$ or $x \neq w$, then visit every state except $q_A$ or $q_R$. We indicate that we are doing this by writing a special symbol, say $\zeta$, to the tape. After we have visited every state, enter $q_R$ and *reject*.

      3. If $M$ accepts $w$, then *accept*.

   2. Run $R$ on $\langle M'\rangle$.

   3. If $R$ accepts, then *reject*; otherwise, *accept*.

   If $M$ does not accept $w$, then every state of $M'$ is visited except for $q_A$. In this case, $q_A$ is a useless state and $R$ accepts. If $M$ accepts $w$, then $M'$ will enter the accepting state on input $w$ and every other state is visited on input $x \neq w$. In this case, $R$ will reject. Thus, $M'$ has a useless state iff $w \notin L(M)$.

2. Suppose $A$, $B$, and $C$ are languages with $A \leq B$ and $B \leq C$. Then there are computable functions $f$ and $g$ such that $x \in A$ if and only if $f(x) \in B$ and $y \in B$ if and only if $g(y) \in C$. Consider the function $h(x) = g(f(x))$. We can build a Turing machine that computes $h$ as follows:

   1. On input $x$, simulate a Turing machine that computes $f$ on input $x$ which produces output $y$.

   2. Simulate a Turing machine that computes $g$ on input $y$.

   The output of this machine is $h(x) = g(f(x))$ and thus $h$ is a computable function. Then, $x \in A$ if and only if $h(x) \in C$ and therefore, we have $A \leq C$.

3. Let $A \subseteq \Sigma^*$ be a language such that $A \in \mathbf{P}$ and $A \neq \emptyset, \Sigma^*$. Since we know $A \in \mathbf{P} = \mathbf{NP}$, we just need to show that $A$ is **NP**-hard to show that it is **NP**-complete. Let $B \subseteq \Sigma^*$ be a language such that $B \in \mathbf{NP}$. We will show that $B \leq_P A$. Since $A$ is neither $\emptyset$ nor $\Sigma^*$, there exist words $x \in A$ and $y \notin A$. This gives us the following reduction:

$$f(w) = \begin{cases} x & \text{if } w \in B, \\ y & \text{if } w \notin B. \end{cases}$$

Because $\mathbf{P} = \mathbf{NP}$, there exists a polynomial-time deterministic Turing machine that decides $B$. Therefore, $f$ can be computed by a deterministic polynomial-time Turing machine. Thus, $w \in B$ if and only if $f(w) \in A$ and $B \leq_P A$. Therefore, $A$ is **NP**-complete.

4. Suppose that $d(L_1, L_2)$ is computable and that there is a Turing machine that computes it, given two context-free grammars $G_1$ and $G_2$ that generate $L_1$ and $L_2$ respectively. We will show that the language

   $$ISE_{CFG} = \{\langle G_1, G_2\rangle \mid G_1 \text{ and } G_2 \text{ are context-free grammars and } L(G_1) \cap L(G_2) = \emptyset\}$$

   is decidable. Suppose that $D$ is a Turing machine that will compute $d(L_1, L_2)$ given context-free grammars $G_1$ and $G_2$, where $L(G_1) = L_1$ and $L(G_2) = L_2$. Then we will construct the following machine that decides $ISE_{CFG}$:

   1. On input $\langle G_1, G_2\rangle$, run $D$ on $\langle G_1, G_2\rangle$ which computes $d(L_1, L_2)$.
   2. If $d(L_1, L_2) = 0$, then *reject*. If $d(L_1, L_2) \neq 0$, then *accept*.

   Recall that for two words $u$ and $v$, $d(u, v) = 0$ if and only if $u = v$. Then if $d(L_1, L_2) = 0$, then there exists a word $w$ such that $w \in L_1$ and $w \in L_2$ and therefore $L_1 \cap L_2 \neq \emptyset$. Thus, if $d(L_1, L_2)$ is computable, then $ISE_{CFG}$ is decidable. However, $ISE_{CFG}$ is known to be undecidable. Therefore, contrary to our assumption, $d(L_1, L_2)$ is not computable.

5. First, suppose that $\mathbf{NP} = \mathbf{coNP}$. We know that there exists an **NP**-complete problem, say $L$. Then $L \in \mathbf{NP}$. Since $\mathbf{NP} = \mathbf{coNP}$, we have $L \in \mathbf{coNP}$.

   Now, suppose there exists a language $L$ such that $L \in \mathbf{coNP}$ and $L$ is **NP**-complete. Since $L$ is **NP**-complete, every language in **NP** is polynomial-time reducible to $L$. Let $L' \in \mathbf{NP}$. Then $L' \leq_P L$. But since $L \in \mathbf{coNP}$, this implies that $L' \in \mathbf{coNP}$. Thus, we have $\mathbf{NP} \subseteq \mathbf{coNP}$.

   Now, observe that by the same reduction, we have $\overline{L'} \leq_P \overline{L}$ for all $L' \in \mathbf{NP}$. Note also that $\overline{L'} \in \mathbf{coNP}$ and therefore, $\overline{L}$ is **coNP**-hard. Furthermore, $\overline{L} \in \mathbf{NP}$, since $L \in \mathbf{coNP}$. But this means that $L' \in \mathbf{NP}$ and therefore, $\mathbf{coNP} \subseteq \mathbf{NP}$.

   Thus, $\mathbf{NP} = \mathbf{coNP}$.