Problem Set 1 — Solutions

1. Let us begin by showing that the first definition implies the second. If $L$ meets the first definition, then $L \in \bigcup_{k \geq 0} \text{NTIME}(n^k)$. So $L \in \text{NTIME}(n^i)$ for some $i$. That means we have a non-deterministic Turing machine $M_L$ running for time $O(n^i)$ such that if $x \in L$ then there is an accepting computation of $M_L(x)$, but if $x \not\in L$ then there is no accepting computation of $M_L(x)$. Let $R_L$ be the relation:

$$R_L \overset{\text{def}}{=} \{(x, w) \mid \text{w is a sequence of choices that leads } M_L(x) \text{ to an accepting configuration}\}.$$ 

Clearly, $R_L$ is polynomially-bounded (since $(x, w) \in R_L$ implies $|w| = O(|x|^i)$) and decidable in polynomial time. Also, $x \in L$ iff there exists a $w$ such that $(x, w) \in R_L$. This proves that $L$ meets the second definition.

For the other direction, say we have a language $L$ and a polynomially-bounded relation $R_L$ decidable in polynomial time such that $x \in L$ iff there exists a $w$ such that $(x, w) \in R_L$. Construct the following non-deterministic Turing machine $M_L$ deciding $L$: given input $x$, guess a $w$ of (at most) the appropriate length, and accept iff $(x, w) \in R_L$. It is not hard to see that if $x \in L$ then then there is an accepting computation of $M_L(x)$, but if $x \not\in L$ then there is no accepting computation of $M_L(x)$. Furthermore, $M_L$ runs in polynomial time since $R_L$ is decidable in polynomial time.

2. We are given a language $L$ which is $\mathcal{NP}$-complete and in $\mathcal{P}$. We need to show that if $L' \in \mathcal{NP}$, we can decide $L'$ in polynomial time. Since $L$ is $\mathcal{NP}$-complete, there is a function $f_{L'}$ computable in polynomial time such that

$$x \in L' \iff f_{L'}(x) \in L.$$ 

This gives the following polynomial-time algorithm for $L'$: on input $x$, first compute $y = f_{L'}(x)$; then, decide whether $y \in L$ and accept only if this is true. Correctness of this algorithm is immediate.

3. The language $L$ of the problem is clearly in $\mathcal{P}$: on input $(M, x, 1^t)$ simply simulate an execution of $M(x)$ for at most $t$ steps and accept iff $M(x)$ accepts within that time bound. The simulation can be done in polynomial time (it is worth thinking through the details and convincing yourself that this is true — note that you need to handle both “small” and “large” values of $t$).

We also need to show a reduction from any language $L' \in \mathcal{P}$ to our language $L$. We know there exists a polynomial time machine $M_{L'}$ deciding $L$ in time $n^i$ for some integer $i$. So our reduction $f_{L'}$ — which, of course, depends on $L'$ — proceeds as follows: on input $x$, output $(M_{L'}, x, 1^{|x|^i})$. You can check that $f_{L'}$ can be computed in polynomial time (in fact, time $O(|x|^i + |x|)$).


4. This is rather simple. Let $f_1$ be a Karp reduction from $L_1$ to $L_2$, and let $f_2$ be a Karp reduction from $L_2$ to $L_3$. This means that

\[ x \in L_1 \iff f_1(x) \in L_2 \quad \text{and} \quad x \in L_2 \iff f_2(x) \in L_3. \]

Consider the function $F(x) \overset{\text{def}}{=} f_2(f_1(x))$. Note that this can be computed in polynomial time. (In particular, if $f_1$ takes time at most $n^{i_1}$ to compute, and $f_2$ takes time at most $n^{i_2}$ to compute, then $F$ takes time at most $|f_1(x)|^{i_2} \leq |x|^{i_1 i_2}$ to compute, which is polynomial.) Furthermore,

\[ x \in L_1 \iff f_1(x) \in L_2 \iff f_2(f_1(x)) \in L_3, \]

as desired.

5. Assume we have a super-$NP$-complete language $L$. Since $L \in NP$, we know there is a non-deterministic Turing machine $M_L$ deciding $L$ in time at most $n^i$ for some integer $i$. Let $p(n) \overset{\text{def}}{=} poly(n)$, and let $q$ be a polynomial such that $q(n) = \omega(p(n))$. By the non-deterministic time hierarchy theorem (which we did not cover in class, but which you were allowed to assume for this problem) there exists a language $L$ with $L \in \text{ntime}(q)$ but $L \notin \text{ntime}(p)$.

Since $L$ is super-$NP$-complete and $L' \in NP$, there exists a Karp reduction $f$ from $L'$ to $L$ such that $f$ can be computed in time $poly(n)$. Consider now the following algorithm for deciding $L'$: on input $x$, compute $y = f(x)$ and then run $M_L(y)$; accept iff the latter accepts. It is easy to see that this algorithm correctly decides $L'$. Furthermore, its running time is at most $p(n)$. But this contradicts what we said above.