Computability and Complexity

Recitation 6, Thursday August 17, 2023

Ari Feiglin

Exercise 6.1:

Suppose $\mathbf{P} \neq \mathbf{NP}$, then prove or disprove the following

(1) The class **NPC** is closed under Karp reductions.

(2) The class **NPC** is closed under unions.

(1) This is false. Let $S \in \mathbf{NPC}$ and $S' \in \mathbf{P}$, then there exists a Karp reduction from S' to S (since $S' \in \mathbf{NP}$, and this is the definition of **NP**-hardness). And so $S' \notin \mathbf{NPC}$ (as if it were, then **P** would be equal to **NP**).

(2) This is true. Let S be **NP**-complete, then let $S_1 = \{1w \mid w \in S\} \cup \left\{0w \mid w \in \{0,1\}^*\right\}$. This is **NP**-complete as we can define a reduction from S to S_1 by $w \mapsto 1w$. And similarly let $S_2 = \{0w \mid w \in S\} \cup \left\{1w \mid w \in \{0,1\}^*\right\}$ is **NP**-complete. But

$$S_1 \cup S_2 = \{0,1\}^*$$

which is not **NP**-complete (you cannot define a Karp reduction from any other problem to $\{0,1\}^*$).

Exercise 6.2:

For every search problem R which is not in **PF**, such that $S_R \in \mathbf{P}$, there is no self reduction.

Suppose S_R can be solved using a polynomial-time algorithm A. Suppose for the sake of a contradiction that there is a self reduction, which is a Cook reduction from R to S_R. But then we could replace all the oracle calls with calls to A, and this is a polynomial-time algorithm which solves R. Thus $R \in \mathbf{PF}$ in contradiction.

In the first homework, we showed that there exists a search problem R (which is polynomially bound) such that $S_R \in \mathbf{P}$ and $R \notin \mathbf{PF}$. But does there exist a search problem $R \in \mathbf{PC}$ such that $S_R \in \mathbf{P}$ and $R \notin \mathbf{PF}$? Well, let us define

$$R = \{(n,k) \mid k \text{ is a non-trivial divisor of } n\}$$

it is a very important conjecture that $R \notin \mathbf{PF}$. In fact this conjecture is the basis for a lot of encryption algorithms. But $R \in \mathbf{PC}$ since given (n,k) we just verify that $k \neq 1, n$ and that k divides n. And $S_R = \mathbb{N}$ which is in **P**.

Exercise 6.3:

Assuming that $P \neq \mathbf{NP} \cap \mathbf{coNP}$, show that there exists a search problem $R \in \mathbf{PC}$ which has no self-reduction.

Let S be in $\mathbf{NP} \cap \mathbf{coNP}$ but not **P**. As before, let us denote V_S, V_{S^c}, p_S, and p_{S^c} be the polynomial proof systems and their polynomials for S and S^c. Let us define

$$R_S = \{(x,y) \mid |y| \leq p_S(|x|),\, V_S(x,y) = 1\}, \qquad R_{S^c} = \{(x,y) \mid |y| \leq p_{S^c}(|x|),\, V_{S^c}(x,y) = 1\}$$

It is trivial to see that R_S and R_{S^c} are in **PC**. Since **PC** is closed under unions (simply run both algorithms), $R = R_S \cup R_{S^c} \in \mathbf{PC}$. We will show that R has no self-reduction. Note that

$$S_R = \{x \mid \exists y\colon (x,y) \in R_S\} \cup \{x \mid \exists y\colon (x,y) \in R_{S^c}\} = S \cup S^c = \{0,1\}^* \in \mathbf{P}$$

The second-to-last equality is due to S and S^c being in **NP**, so $S_{R_S} = S$ and $S_{R_{S^c}} = S^c$.

Now suppose that $R \in \mathbf{PF}$, then there exists a polynomial-time algorithm A which solves R. We can define a new algorithm D where $D(x) = V_S(x, A(x))$. So if $x \in S$ then there exists a witness y such that $(x,y) \in R_S$ and so $A(x)$ cannot be $\perp$. And $x \notin S^c$ so there is not witness for x for R_{S^c}, thus $A(x)$ must be a witness for R_S, and so $D(x) = 1$.

And if $x \notin S$ then $V_S(x, A(x)) = 0$. Thus D decides S in polynomial time, in contradiction. So $R \notin \mathbf{PF}$, as required.