Computability and Complexity

Recitation 9, Tuesday August 29, 2023

Ari Feiglin

Exercise 9.1:

Let us define

$$\mathbf{P}/_{\log} = \bigcup_{c=1}^{\infty} \mathbf{P}/_{c\log}$$

Then if $\mathbf{NP} \subseteq \mathbf{P}/_{\log}$, $\mathbf{PH} = \mathbf{P}$.

We will show that if $\mathbf{NP} \subseteq \mathbf{P}/_{\log}$, $\mathbf{P} = \mathbf{NP}$ and this means that $\mathbf{PH} = \mathbf{P}$. Since $\mathbf{NP} \subseteq \mathbf{P}/_{\log}$, $\mathsf{3SAT} \in \mathbf{P}/_{\log}$ and $\mathsf{3SAT}$ is $\mathbf{NP}$-complete. We will show that $\mathsf{3SAT}$ is in $\mathbf{P}$, which will show that $\mathbf{NP} = \mathbf{P}$.

So there exists a sequence of commands $\{a_n\}_{n=0}^{\infty}$ whose lengths are bound by $c\log(n)$ and a polynomial-time algorithm $A(a_n, x)$ such that $\varphi \in \mathsf{3SAT}$ if and only if $A(a_{|\varphi|}, \varphi) = 1$. We can assume without loss of generality that for every n, $|a_n| = c\log(n)$. Notice that for every n, there are $2^{c\log(n)} = n^c$ possible commands for a_n which is polynomial. So let us define a polynomial algorithm B which decides $\mathsf{3SAT}$:

(1) Iterate over every string $a \in \{0,1\}^{c\log(n)}$.

(2) If $A(a, \varphi) = 0$, then φ is either not satisfiable or a is an invalid command, so continue to the next a.

(3) Otherwise, we can perform a self-reduction on φ *without changing its length*. This is important since if we were to change its length, then a would no longer be a valid command. We perform the self-reduction recursively as follows: first we set x_1 to be true and simplify φ, but this would change the length of φ. If within a disjunction, we lose a variable (x_1), then we can simply replace it with another variable in that disjunction to get an equivalent boolean formula of the same length. If we lose an entire disjunction, we can simply copy a different disjunction. For example

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

Setting x_1 to true will simplify to $(\neg x_2 \vee x_3)$, so we've lost a disjunction and a variable, so we replace x_1 in the second disjunction with $\neg x_2$ and copy it:

$$(\neg x_2 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_2 \vee x_3)$$

We now check if this new boolean formula φ' is satisfiable, if $A(a, \varphi') = 1$. Otherwise we set x_1 to false. In both cases we now recurse.

(4) At the end, we get a boolean vector τ. We verify that τ satisfies φ, and return one if it does, and otherwise continue to the next a.

(5) If we have finished the loop without returning one, return zero.

Each iteration of the loop takes polynomial time, since A takes polynomial time and we perform A m times, where m is the number of variables. The loop repeats $\left|\{0,1\}^{c\log(n)}\right| = n^c$ times, so all in all B is polynomial.

If $\varphi \notin \mathsf{3SAT}$ no boolean vector, and in particular the boolean vector generated by B, will satisfy φ. So B will return zero. And if $\varphi \in \mathsf{3SAT}$, once B gets to $a_{|\varphi|}$ it will give correct results for $A(a_{|\varphi|}, \varphi)$ and so it will generate a boolean vector τ which satisfies φ.

Thus $\mathsf{3SAT}$, which is $\mathbf{NP}$-complete, is in $\mathbf{P}$. Therefore $\mathbf{P} = \mathbf{NP}$ and so $\mathbf{PH} = \mathbf{P}$.