Problem Set 1, Answers

Kevin Lacker

May 12, 2020

Problem 1. A language L is in $\Sigma_2^{\mathbf{P}}$ iff there is a polynomial time TM M such that:

$$x \in L \iff \exists u_1 \forall u_2 M(x, u_1, u_2) \tag{1}$$

where the u_i are polynomial size and we treat M as returning true or false. This is equivalent to:

$$x \in L \iff \exists u_1 \neg(\exists u_2(\neg M(x, u_1, u_2)) \tag{2}$$

The answer to $\exists u_2(\neg M(x, u_1, u_2))$ can be found in a single call to an **NP** oracle, so $\Sigma_2^{\mathbf{P}} \in \mathbf{NP}^{\mathbf{NP}}$.

The other direction is similar. For any language L in $\mathbf{NP}^{\mathbf{NP}}$, there is a Turing machine M, where M has access to an **NP** oracle, such that:

$$x \in L \iff \exists v_1 M(x, v_1) \tag{3}$$

There are different equivalent formats for the oracle. It is convenient to think of the input as being a SAT problem, and the output is either a solution or an indication that there is no solution.

To write this as a $\Sigma_2^{\mathbf{P}}$ language we need to convert the oracle calls to a regular Turing machine under a $\forall$ quantifier. So, let v_2 represent the oracle outputs, and v_3 represent a string of bits long enough to hold all assignments to the SAT problems where the oracle reported no solution. M makes polynomially many calls to the oracle, so the lenght of v_2 and v_3 can be polynomially bounded.

Then, we can represent L as:

$$x \in L \iff \exists v_1 v_2 \forall v_3 M'(x, v_1, v_2, v_3) \tag{4}$$

Where M' works like:

- It performs the same operations as M, except when there is an oracle call
- When the oracle gives a solution, M' validates the solution
- When the oracle reports no solution, M' uses bits from v_3 to check that the particular inputs do not represent a solution

Since this is quantified over all v_3, the quantifiers do the job of the NP oracle, and this is an equivalent representation for L. Thus $\mathbf{NP}^{\mathbf{NP}} \in \mathbf{\Sigma_2^P}$.