More PH

Ben Little

Sep 25, 2023

Problem 1

Part a

Curry $\exists^p y_1 \exists^p y_2$ into $\exists^{2p}(y_1, y_2) \equiv \exists^p(y_1, y_2)$.

Part b

Same argument as part a. Since each witness is poly-length, their concatenation is also poly-len.

Problem 2

I take verifiers to be languages in P, as opposed to poly-time machines. $x \circ y$ is the concatenation of strings.

$L \in \Sigma_{k+1}P \iff \exists V, p\, \forall x \in L\, \exists^p v_{k+1}\, \forall^p v_k\, \cdots |\, x \circ v_{k+1} \circ v_k \cdots v_1 \in V$

Now consider another lanuage M such that $\exists^p y\, |x \circ y \in M \iff x \in L$. Rewrite again

$\exists V, p\, \forall (x \circ v_{k+1}) \in M\, \forall^p v_k \cdots\, |x \circ v_k \cdots v_1 \in V$

Now M is a language in $\Pi_k P$. By the assumption of the problem, $M \in \Sigma_k$, the term can be further rewritten

$\exists W, q\, \forall (x \circ v_{k+1}) \in M\, \exists^p w_k\, \forall^p w_{k-1}\, \cdots\, |x \circ v_{k+1} \cdots w_1 \in W$

Because $x \circ v_{k+1} \in M \iff x \in L$

$\exists W, q\, \forall x \in L\, \exists v_{k+1}\, \exists^p w_k\, \forall^p w_{k-1}\, \cdots\, |x \circ v_{k+1} \cdots w_1 \in W$

which can be curried into

$\exists W, q\, \forall x \in L\, \exists^p (v_{k+1} \circ w_k)\, \forall^p w_{k-1} \cdots\, |x \circ v_{k+1} \cdots w_1 \in W$

and this is the definition of $\Sigma_k P$.

So $\Sigma_k P \equiv \Pi_k P \implies \Sigma_k P \equiv \Pi_k P \equiv \Sigma_{k+1} P$.

A similar sequence of steps proves $\Pi_{k+1}P \equiv \Pi_k P$, except M is defined as $\forall^p y | x \circ y \in M \iff x \in L$.

Problem 3

Part a

Let M be a machine that decides a language $L \in P^{NP}$.

For each $x \in L$, there exists a poly-length trace because M always terminates in poly-number of steps on such input. That means there exists poly-many "yes" responses to queries, i.e. $\forall x \in L \;\; \exists y_1, \ldots, y_{\text{poly}(|x|)}$. By the definition of NP, there exists a verifier language in P where $\exists^p w = (y_1, \ldots) \,.\, x \circ w \in V_{\text{YES}} \iff x \in L$.

Likewise there are poly-many "no" responses. Because a coNP oracle can be derived by taking the inverse result of a NP oracle, by the defintion of coNP there exists a verifier language in P where $\forall^p w = (n_1, \ldots) \,.\, x \circ w \in V_{\text{NO}} \iff x \in L$. So the combination of verifiers $V := V_{\text{YES}} \oplus V_{\text{NO}}$ is in P.

Now it possible to write

$$\exists V \; \forall x \in L \; \exists^p (y_1, \ldots, y_{\text{poly}(|x|)}) \; \forall^p (n_1, \ldots, n_{\text{poly}(|x|)}) \,.\, x \circ y_1 \circ \cdots \circ n_1 \circ \cdots \in V$$

Part b

I believe the same argument for part a applies because it is the length of the trace and the defintion of NP that assures that $V \in P$. A non-deterministic trace may have used an unbounded number of oracle calls, but it only needs to verify those used in one linear trace because the definition of NP says that only one linear trace needs to be verified to ascertain that the input string is in the language.

Part c

No attempt (yet)

Problem 4

Let $a \uparrow^c b$ be a raised to the power b c-times.

$\Sigma_{k+j}P = (\Sigma_{k+j-1}P)^{NP} = ((\Sigma_{k+j-2}P)^{NP})^{NP} = \cdots = (\Sigma_k P) \uparrow^j NP$

By assumption $\Sigma_{k+1}P = \Sigma_k P$, so

$\Sigma_{k+j}P = (\Sigma_k P) \uparrow^j NP = (\Sigma_{k+1}P) \uparrow^j NP = \cdots = \Sigma_{k+j+1}P$

So for all j, $\Sigma_k P = \Sigma_{k+1}P \implies \Sigma_{k+j+1}P = \Sigma_{k+j}P$. By induction on j, PH collapses to $\Sigma_k P$.

Problem 5

Part a

Define a circuit family C_n where each ciscuit encodes a table of all formulas with length n and their corrsponding satisfying assignment if it exists or all-zeros if it doesn't. This is a depth-1 circuit family and so it is in $P/poly$.

This works regardless of whether or not $NP \subseteq P/poly$.