Polynomial Hierarchy

Ben Little

Problem 1

Part a

One direction is trivial by definition of $\subseteq$. I will demonstrate that $\mathcal{C} \subseteq \text{co}\mathcal{C}$ implies that $\mathcal{C} \equiv \text{co}\mathcal{C}$.

Theorem $\mathcal{C} \subseteq \text{co}\mathcal{C} \implies \text{co}\mathcal{C} \subseteq \mathcal{C}$

Proof $\mathcal{C} \subseteq \text{co}\mathcal{C}$

(1) $\forall L \in \mathcal{C} \,.\, L \in \text{co}\mathcal{C}$,

(2) $L \in \text{co}\mathcal{C} \implies \bar{L} \in \text{co}(\text{co}\mathcal{C}) = \mathcal{C}$

Summarized,

(3) $\forall L \in \mathcal{C} \,.\, \bar{L} \in \mathcal{C}$

By definition,

(4) $\forall M \in \text{co}\mathcal{C} \,.\, \bar{M} \in \mathcal{C}$

By applying (3) in (4),

(5) $\forall M \in \text{co}\mathcal{C} \,.\, M \in \mathcal{C} \implies \text{co}\mathcal{C} \subseteq \mathcal{C}$

So we have $C \subseteq \text{co}C \wedge \text{co}C \subseteq C \implies C \equiv \text{co}C$. $\square$

Problem 2

Part b

Using the result of part a, this reduces to showing that $P \subseteq \text{co}P$.

Theorem $P \subseteq \text{co}P$

Proof Given a polynomial time TM T that decides a language $L \in P$, construct a TM that simulates T but returns the opposite result. This machine returns 1 for “no” instances of L and 0 for “yes” instances, hence it correctly decides $\bar{L}$ and runs in polynomial time. Therefore $\bar{L} \in P$. $\square$

Part c

Theorem $\mathcal{C} \cup \text{co}\mathcal{C}$ is closed under complement.

Proof $L \in \mathcal{C} \cup \text{co}\mathcal{C}$

Left

(1) $L \in \mathcal{C}$

(2) $\bar{L} \in \text{co}\mathcal{C}$

Right

(3) $L \in \text{co}\mathcal{C}$

(4) $\bar{L} \in \mathcal{C}$

By (2) and (4)

(5) $\bar{L} \in \mathcal{C} \cup \text{co}\mathcal{C}$ $\square$

Problem 2

If NP were to equal coNP, then it would always be possible to provide a polynomial-time verifiable witness that no solution to a NP problem instances exists. For SAT, this would mean that every unsatsifiable formula could be proven unsatisfiable using a proof that is only polynomial in the number of variables. Given there are 2^n possible assignments, it feels unlikely that a polynomial size witness could demonstrate this.

There are some examples where there this is possible, however. A CNF formula containing $x \wedge \neg x$ anywhere has a very short proof of unsatisfiability! Expecting that *every* formula would have a short witness like this feels like a stretch, and I would expect that any proof of $NP = \text{co}NP$ would be non-constructive, as there are exponentially (in the number of vars) ways for a formula to be unsatisfiable.

Problem 3

Part a

Theorem P^X is closed under complement.

Proof Given $L \in P^X$ and an TM T with oracle O, build a TM T' (with the same oracle O) such that T' simulates T and inverts its output. This correctly decides $\bar{L}$ and runs in polynomial time, so $\bar{L} \in P^X$. $\square$

Part b

Theorem $P^{NP} = P^{\text{co}NP}$

Proof Given $L \in P^{NP}$ and T with NP-oracle O, construct a TM T' with coNP oracle O'.

Let O' decide $\bar{\text{SAT}}$. Then let T' simulate T until it queries O. Instead of querying O, T' will convert the NP problem instance to an instance of SAT and query O'. If O' returns ACCEPT (formula unsatisfiable), then proceed as if T had received REJECT from O. If O' returns REJECT (formula satisfiable), then proceed as if T ahad received ACCEPT.

T' has the same time complexity as T and therefore $L \in P^{\text{co}NP}$.

A similar construction can be used in the other direction, where O' is a NP oracle. $\square$

Part c

(1) $NP = P^{NP} = P^{\text{co}NP}$

(2) Trivially, co$NP \subseteq P^{\text{co}NP}$

By (1) and (2):

(3) co$NP \subseteq NP$

By problem 1a, $NP = $ coNP

Problem 4

$V(-)$ is a polynomial time verifier.

$\Sigma_0 P = \Pi_0 P = \{L | \exists V(-) \forall x \, . \, V(x) = L(x)\}$

So there exists V that decides L in polynomial time, i.e. $L \in P$.

$\Sigma_1 P = \{L | \exists V(-) \; \forall x \in L \; \exists^p w \, . \, V(x, w) = L(x)\}$

So there exists a p-time verifier V such that for each $x \in L$ there exists a w such that $V(x, w)$ decides $x \in L$. This is the definition of NP.

$\Pi_0 P = \{L | \exists V(-) \; \forall x \in L \; \forall^p w \, . \, V(x, w) = L(x)\}$

This a little tricker. Notice that the quantifier in $\forall^p w \, . \, V(w, x) = L(x)$ can be inverted to yield $\neg \exists^p w \, . \, V(x, w) \neq L(x)$. So there exists no witness that can decide if $x \in \bar{L}$. This is the definition of coNP.

Problem 5

Part a

For any language in PH, test the binary encoding of every possible witness of a polynomial length. There poly(n)-many witnesses of poly(n) length, so this runs in $2^{poly(n)}$ time.

Part b

Proof by induction on the k of $\Sigma_k P \cup \Pi_k P$.

Base case: solution to problem 4 and $P \subseteq$ PSPACE.

Inductive case: check each binary-encoded assignemnt of the $(k+1)$-th witness. If this is an existential witness, run until a satsfying assignment is found or all possiblitities have been exhausted. If it is a universal witness, run until a non-satsifying assignment is found or all possiblities have been exhausted. In either case, we do not need to remember what those assignements were because there exists a total ordering on binary assignments, so this check runs in PSPACE. By the inductive hypthesis, $\Sigma_k P \cup \Pi_k P \subseteq$ PSPACE, and so the entire algorithm runs in PSPACE.