Tutorial 10: Answer sketches.

A1: Suppose L_1 and L_2 in PP is witnessed by M_1 and M_2 respectively. Consider $M(x)$, which runs $M_1(x)$ and $M_2(x)$ and accepts iff exactly one of $M_1(x)$ and $M_2(x)$ accepts. Then, $M(x)$ is correct if both $M_1(x)$ and $M_2(x)$ are correct outputs, or both are wrong outputs. Thus, M is correct with probability $(1/2+\alpha_1)(1/2+\alpha_2)+(1/2-\alpha_1)(1/2-\alpha_2) = 1/2+2\alpha_1\alpha_2 > 1/2$, where $M_1(x)$ is correct with probability $1/2+\alpha_1$ and $M_2(x)$ is correct with probability $1/2+\alpha_2$.

2: It is easy to verify that QBF is in PSPACE (details left to the student).

To show that QBF is PSPACE-hard, suppose L is a PSPACE language as witnessed by Turing Machine M which is n^k space bounded, and 2^{n^k} time bounded (being c^{n^k} time bounded can be done similarly).

Let $P_m(U,V)$ denote the formula for saying that the machine M can go from ID U to ID V in at most 2^m steps.

P_0 is easy to define.

$P_{m+1}(U,V) = (\exists Z)(\forall X)(\forall Y)[[(U = X \text{ and } Y = Z) \text{ or } (Z = X \text{ and } Y = V)] \Rightarrow P_m(X,Y)]$

equivalently:

$P_{m+1}(U,V) = (\exists Z)(\forall X)(\forall Y)[[\neg(U = X) \text{ and } \neg(Z = X)] \text{ or } [\neg(U = X) \text{ and } \neg(Y = V)] \text{ or } [\neg(Y = Z) \text{ and } \neg(Z = X)] \text{ or } [\neg(Y = Z) \text{ and } \neg(Y = V)] \text{ or } P_m(X,Y)]$

Suppose $U = u_1u_2\ldots u_{n^k}$, $V = v_1v_2\ldots v_{n^k}$, $X = x_1x_2\ldots x_{n^k}$, $Y = y_1y_2\ldots y_{n^k}$, $Z = z_1z_2\ldots z_{n^k}$. Now, for example, $\neg(U = X)$ and $\neg(Z = X)$ can be expressed as disjunction of n^{2k} formulas (for $1 \le i,j \le n^k$) as follows.

$(\neg u_i \text{ and } x_i \text{ and } \neg z_j \text{ and } x_j)$ or $(\neg u_i \text{ and } x_i \text{ and } z_j \text{ and } \neg x_j)$ or $(u_i \text{ and } \neg x_i \text{ and } \neg z_j \text{ and } x_j)$ or $(u_i \text{ and } \neg x_i \text{ and } z_j \text{ and } \neg x_j)$.

The quantifiers in $P_m(X,Y)$ can be brought forwards to the beginning using

standard methods. Thus, we can express $P_m(X,Y)$ in prenex form in length polynomial in m, n.

Now, $x \in L$ iff $P_{n^k}(SID, AID)$, where SID is starting ID and AID is accepting ID for M on input x.

A3: Consider the following function f on input (V,C). If the set of clauses is empty, then output 1 (satisfiable). If the set of clauses contains an empty clause (note that empty clause evaluates to false), then output 0 (not satisfiable). Otherwise, let x be a member of V. Let $V' = V - \{x\}$. Let C' be formed from C by setting x to true and C'' be formed from C by setting x to false.

Here: forming C' by setting x to true means that we delete the clauses which have x as a literal (since these clauses evaluate to true when x is true), and by removing the literals $\neg x$ (if present) from rest of the clauses. (Similarly we get C'' by setting x to false).

Then, $f(x)$ contains (V',C') and (V',C''). Note that (V,C) is satisfiable iff at least one of (V',C') and (V',C'') is satisfiable. Furthermore, both (V',C') and (V',C'') are of size smaller than (V,C).

A4: Consider the following procedure:

On input x:

Let $y = x$.

Loop:

If $f(y) = 1$, then accept.
If $f(y) = 0$, then reject.
Otherwise, let $y = f(y)$, and continue the loop.

End Loop.

Then, above procedure runs in polynomial time, as each loop iteration takes polynomial time, and there are at most $|x| + 1$ iterations of the loop. Furthermore, due to the properties of f, we always have that $x \in L$ iff $y \in L$, for any value y computed during the computation. Thus, above procedure accepts L.

A5:

(a) Suppose $L \in NP$. Let Q be poly-time decidable predicate such that $x \in L$ iff $(\exists y)[Q(x, y)]$, where length of y is a polynomial in length of x.

Then, consider the prover which on input x sends one such y (if there) to the verifier. Verifier checks that $Q(x, y)$ is true or not, and answers correspondingly.

It is easy to verify that if $x \in L$, then the correct prover sends a y such that $Q(x, y)$ is true, and thus verifier accepts.

If $x \in L$, then whatever any prover may send, verifier rejects. Thus $L \in IP_{1,0}$.

(b) Suppose $L \in IP_{2/3,0}$ as witnessed by prover P and verifier V.

Then consider the nondeterministic machine M which on input x, guesses the strings sent by prover, guesses the coin-tosses of the verifier and checks if the verifier accepts. If so, then M accepts. Otherwise it rejects.

It is easy to verify that M would witness that $L \in NP$.