NP Completeness - I

Examples of languages in NP : CLIQUE, CNF-SAT, TSP, SUBSET-SUM. In all these examples, we see that if a string belongs to the language, (eg. if the given graph has a TS tour), then there is a short (polynomial length) **proof** of this fact (in the example, the proof is the tour itself). If (magically) someone has already found the proof, one can **verify** that this is indeed a correct proof in deterministic polynomial time. (Checking whether the given order of vertices in a graph is indeed a TSP tour is easy). This fact is in general true as we see now.

For a language L, a **verifier** is another language L' with the property that

$$\forall x, \qquad x \in L \quad \Leftrightarrow \quad \exists y \ : \ (x, y) \in L'.$$

[y is the proof that x is in L.] L is said to be "projection" of L' (obtained by "projecting out" the proof - analogous to geometric projections). We say that the verifier has polynomial length proofs if there is a polynomial $p(\cdot)$ such that for every $x \in L$, there is a proof y with $|y| \leq p(|x|)$.

Theorem L is in NP iff it has a verifier with polynomial length proofs which is in P.

Proof If L is in NP, let M be a poly time NDTM accepting L. Then for any string $x \in L$, the "proof" that x is in L is the sequence of non-determinsitic choices of M which result in M accepting x. It is easy to see that for every (i) the sequence of choices can be described by a polynomial length string y and (ii) given y, we can simulate M with these choices in deterministic polynomial time. This will prove one direction. The other direction is easier.

Existence of NP-complete languages

A language L in NP is said to be NP-complete if there is a many-one polynomial time reduction from every language in NP to L. The existence of an NP-complete language will follow from ideas similar to the proof that the univeral language is r.e. complete. We need one poly time bounded NDTM to simulate any given polynomial time bounded NDTM. This is problematic since the simulated machine may take an arbitrary poly time. We use a trick called "padding" to tackle this. Define

$$U = \{(x, M, 1^s) : \text{ NDTM } M \text{ accepts } x \text{ in at most } s \text{ steps } \}.$$

[The 1^s provides padding.] It is easy to check that (i) U is in NP. (ii) Every L in NP is poly time many-on reducible to U.