Interactive Proofs

Fix a language L, and consider the following setting.
Prover P is arbitrary (deterministic) function. Note that P maynot be computable.
Verifier V is a polynomial time bounded randomized Turing Machine.
Initially, an input x is given to both P and V.

Then P and V go through a few (polynomial) rounds of interaction involving the following two steps:
(1) V does some computation and asks a question to P.
(2) P answers the question (maybe wrongly).
At the end of the final round of interaction, V either accepts or rejects x.
Note that P's answer can depend only on the current and previous questions asked by the verifier, and the input x. In particular, P does not know about the coin tosses of the verifier.

We say that a language L is in **IP** iff, there exists a prover P and a verifier V (in the setting above), such that
(1) If $x \in L$, then in the interaction of P and V as above, V accepts with probability at least $\frac{3}{4}$.
(2) If $x \notin L$, then for any prover P', in the interaction between P' and V as above, V accepts with probability at most $\frac{1}{4}$.

Exercise: Show that $\mathbf{NP} \subseteq \mathbf{IP}$.
Exercise: Show that $\mathbf{BPP} \subseteq \mathbf{IP}$.
In fact a much more general theorem can be proved. It can be shown that $\mathbf{IP} = \mathbf{PSPACE}$.

We show as an example that Graph Non-Isomorphism is in **IP**. Note that it is not known whether Graph Non-Isomorphism is in **NP** or not.

Verifier $V(G_1, G_2)$.

Loop m times

1. Flip a coin. If it comes head, then let $t = 1$ and generate a random graph G isomorphic to G_1 (by generating a random permutation of the vertices of G_1).
   Otherwise let $t = 2$, and generate a random graph G isomorphic to G_2.
2. Ask the prover whether G is isomorphic to G_1 or G_2.
   If prover's answer is different from t then reject.

End Loop

If the input is not rejected in any of the above iterations, then accept.

End

Note that if G_1 and G_2 are non-isomorphic, then a genuine prover can always answer correctly: G can be isomorphic to only one of G_1 and G_2.
So if G_1 and G_2 are non-isomorphic then V accepts with probability 1.
On the other hand, if G_1 and G_2 are isomorphic, then G is isomorphic to both G_1 and G_2.
A (maybe cheating) prover has no way to know whether V obtained G by taking a graph isomorphic to G_1 or by taking G to be isomorphic to G_2.
Thus the probability of the prover passing the test in step 2, is at most $\frac{1}{2}$. If we take $m = 2$, then the probability that the prover passes the test in step 2 both times is at most $\frac{1}{4}$.
Thus, (for $m = 2$), the verifier wrongly accepts with probability at most $\frac{1}{4}$.

Theorem: IP=PSPACE
Proof:
QBF (also called QSAT): Quantified Boolean Formula
$(Q_1 x_1)(Q_2 x_2) \ldots (Q_n x_n)[B(x_1, \ldots, x_n)]$
is complete for PSPACE.

What we want the prover to convince the verifier is that a QBF formula evaluates to true. Alternatively, if we consider B as a polynomial, we want the prover to convince the verifier that the above formula evaluates to 1, where one takes
$\alpha \wedge \beta$ to be $\alpha \cdot \beta$
and
$\alpha \vee \beta$ to be $\alpha * \beta = 1 - (1 - \alpha)(1 - \beta)$
$(\forall x)[H(x)]$ is same as $H(0) \cdot H(1)$
$(\exists x)[H(x)] = H(0) * H(1)$.
So essentially, we have a QBF viewed as a polynomial.

Problem: Degree of the polynomial can become very high (exponential in n) if we expand the polynomial.

Introduce another operator Rx as follows:
$(Rx)[P(x,\ldots)]$ is same as $P(x,\ldots)$ mod $(x^2 - x)$
(that is, x^r is replaced by x)
Note that operator R does not change the value of the formula on inputs $x = 0$ and $x = 1$.

R allows us to reduce the degree of the polynomial.
We convert QBF by reducing the degree between each quantifier, by introducing the quantifiers (Rx) for the free variables as needed.
We will be doing arithmetic over a finite field F (doing arithmetic modulo p for some prime number p).

Over time the formula would have some free variables.
The prover needs to convince the verifier that some formula $S(c_1, c_2, \ldots, c_r)$ (as above) evaluates to some value v in the field.
If S does not have quantifiers, then the verifier can easily check that.

If $S(c_1, \ldots, c_r)$ is of form, $(\forall x)[S'(x, c_1, \ldots, c_r)]$, then
1. The prover sends a polynomial $P'(x)$ for S' (of degree at most d in x).
2. The verifier verifies that $P'(0) \cdot P'(1) = v$, and the degree of P' is at most d.
3. If not, then reject. Otherwise, the verifier selects a random $v' \in F$, and asks the prover to prove that $S'(v', c_1, \ldots, c_r) = P'(v')$

If $S(c_1, \ldots, c_r)$ is of form, $(\exists x)[S'(x, c_1, \ldots, c_r)]$, then
1. The prover sends a polynomial $P'(x)$ for S' (of degree at most d in x).
2. The verifier verifies that $P'(0) * P'(1) = v$, and the degree of P' is at most d.
3. If not, then reject. Otherwise, the verifier selects a random $v' \in F$, and asks the prover to prove that $S'(v', c_1, \ldots, c_r) = P'(v')$

If $S(x, c_1, \ldots, c_r)$ is of form, $(Rx)[S'(x, c_1, \ldots, c_r)]$, then
Prover wants to persuade verifier that $S(f, c_1, \ldots, c_r) = v$
1. The prover sends a polynomial $P'(x)$ for S' (of degree at most d in x).
2. The verifier verifies that $P'(0) + (P'(1) - P'(0))f = v$, and the degree of P' is at most d.
3. If not, then reject. Otherwise, the verifier selects a random $v' \in F$, and asks the prover to prove that $S'(v', c_1, \ldots, c_r) = P'(v')$

Chances of "messing up" in each iteration of the above:
Here messing up means that we started with a formula/value which is wrong, and it leads to verifier accepting in the end.
If the formula $S(c_1, \ldots, c_r)$ does not evaluate to v, then either the prover sends a correct polynomial for S' (in which case it will be caught), or it sends a wrong polynomial.
The wrong polynomial coincides with the correct one on at most d values. So the prover continues with wrong "formula/value" with probability at least $\frac{|F|-d}{|F|}$.

If the length of the original QBF is n, then there are at most n^2 rounds.
Also note that if the length of the original formula is bounded by n, then the degree of B in the original formula is at most $O(n)$. The degree of polynomial at any other stage (after the initial reductions) is at most 2 at any point of time. Thus we can take d to be $O(n)$.
Thus, total amount of error introduced is at most $O(\frac{n^3}{|F|})$. Taking $|F| > n^4$ would give us that the error is small.