Interactive Proof

— Bundit Laekhanukit

- Interactive Proof ?

  → Extend from NP-proof System

- Allow more than communication between

  prover and verifier.

- More powerful → Capture more class of problems.
- Allow error → Sometimes we only need to read $O(1)$ bits

  of witness / proof to verify a claim.

NP - Proof System

- Two parties : Verifier and Prover.
- A languaguage L = subset of strings, say $L = \Sigma^*$
- Input $x \in \Sigma^*$, $|x| = n$
- Statement $x \in L$

$\boxed{\text{Def.}}$ Class NP – A language L is in NP if

$\exists$ Verifier V : V runs in poly time on n and do the following

witness/proof from prover.

poly(n)

Yes $x \in L \mapsto \exists\, w \in \Sigma^{\text{poly}(n)}$ $V(x, w) = 1$

No $x \notin L \mapsto \forall\, w \in \Sigma^{\text{poly}(n)}$ $V(x, w) = 0$

Warm-Up: Interactive Proof with Deterministic Verifier (dIP)

- k-round interactions of two binary functions f and g.

on input $x \in \{0,1\}^*$

$\Rightarrow$ Denoted by $\langle f, g \rangle (x)$

$f(x)$ $\qquad\qquad\qquad\qquad$ $g(x)$

$a_1 = f(x)$ $\longmapsto$

$\longleftarrow$ $a_2 = g(x, a_1)$

$a_2 = f(x, a_1, a_2, a_3)$ $\longmapsto$ $\cdots$

$\vdots$

$a_{2i+1} = f(x, a_1, \dots, a_{2i})$ $\longmapsto$ $a_{2i+2} = g(x, a_1, \dots, a_{2i+1})$

At round k: output $\text{out}_f \langle f, g \rangle (x) = f(x, a_1, \dots, a_k)$

Deterministic Proof System

Verifier(x) V Prover(x) P

$$V \xrightarrow{\text{Request witness}} P$$

$$V \xleftarrow{\text{witness } w} P$$

(Completeness) $x \in L \implies \exists w : \text{out}_V\langle V, P\rangle(x) = V(x, w) = 1$

(Soundness) $x \notin L \implies \forall w : \text{out}_V\langle V, P\rangle(x) = V(x, w) = 0$

Note: for NP-proof system, we require $|w| = \text{poly}(n)$

V runs in $\text{poly}(n)$ time

Thm 8.3 $\qquad d\mathcal{IP} = NP$ $\qquad$ ($k = poly(n)$, & runs in polytime(n))

Proof

- $NP \subseteq d\mathcal{IP}$ $\qquad$ Trivial.
- $d\mathcal{IP} \subseteq NP$

  - Prover simulate k rounds interactions $\langle \hat{V}, \hat{P} \rangle (x)$

    $\Rightarrow$ generate transaction $(x, a_1, a_2, \dots, a_k)$

    $\Rightarrow$ submit $w = (x, a_1, a_2, \dots, a_k)$ as a proof/witness.

$x \in L \Rightarrow V(x, a_1, \dots, a_k) = 1$, $\quad x \notin L \Rightarrow V(x, a_1, \dots, a_k) = 0$

$\#$

The class IP

Def. IP – Class of language L that can be captured by

$k = poly(n)$ rounds interaction $\langle V, P\rangle(x)$, V runs in $poly(n)$

(Completeness) $x \in L \Rightarrow \exists P \ \Pr[out_V \langle V, P\rangle(x) = 1] \geq \frac{2}{3}$

(Soundness) $x \notin L \Rightarrow \forall P \ \Pr[out_V \langle V, P\rangle(x) = 1] \leq \frac{1}{3}$

$$\boxed{IP[n^c] := k = O(n^c). \qquad IP = \bigcup_{c \geq 1} IP[n^c]}$$

$out_V \langle V, P\rangle(x) = 1$ means "accept"

0 means "reject"

Note IP

- Verifier is a probabilistic Turing Machine.
- Prover can be determinist or probabilistic
  - $\Rightarrow$ It does not change the class.
- $IP \subseteq PSPACE$
- $\left(\frac{2}{3}, \frac{1}{3}\right)$ probability can be boosted to $(1-\varepsilon, \varepsilon)$ by repeating the protocol $\nearrow$ Require exp rounds

If verifier has to show all random strings, we call "public coin"

- Prover function does not depend on Verifier's random str.
  - $\rightarrow$ Prover has "private coin"

Example : Graph Non-Isomorphism Protocol.

- Graph Isomorphism : input graphs G_1, G_2
  (GI)

π is witness

Decide if $G_1 \cong G_2$, i.e,

$\boxed{\exists \text{ 1-to-1 function } \pi : V(G_1) \mapsto V(G_2) \text{ s.t. } \pi(G_1) = G_2 .}$

GI — don't know if $GI \overset{?}{\in} P$, but clearly $GI \in NP$.

How about graph non-isomorphism ? (shortly, GNI)

What can be the $\boxed{\text{witness}}$ for $G_1 \not\cong G_2$?

Protocol : Private-Coin Graph Non-Isomorphism

input G_1, G_2

Verifier | **Prover**

$i \in_R \{1,2\}$

H isomorphic with G_i

- randomly pick G_1 or G_2, say G_i
  and pick permutation π
  $H = \pi(G_i)$

$\xrightarrow{H=\pi(G_i)}$

- Identify whether $H \cong G_1$
  or $H \cong G_2$

  Say, $H \cong G_j$

Prover can solve GI
→ can check $H \ncong G_1$ or $H \ncong G_2$

$\xleftarrow{j}$

- Decide if $i = j$.
  Accept if $i = j$
  otherwise, reject.

Correctness of the Protocol

Gap $(1, \frac{1}{2})$ can be enlarged to $(1, \frac{1}{3})$ by repeating the protocol

* only need $(\alpha+\varepsilon, \alpha-\varepsilon)$

(Completeness) $\boxed{G_1 \not\cong G_2}$

- $\exists$ Prover (with computational unbounded) that can distinguish whether $H \cong G_1$ or $H \cong G_2$ (because $G_1 \not\cong G_2$)

$\Rightarrow$ P knows j s.t. $j = i$.

$\Rightarrow$ Verifier accepts w.p. 1.

(Soundness) $\boxed{G_1 \cong G_2}$

accept w.p. $\frac{1}{2}$

$\Rightarrow$ $H \cong G_1$ and $H \cong G_2$ $\Rightarrow$ No prover can distinguish.

$\Rightarrow$ Whatever j prover chooses, $\Pr[i = j] = \frac{1}{2}$

Public coins and AM

- Def 8.7 [AM, MA] - Arthur-Merlin, Merlin-Arthur.

- k rounds of AM (and MA) is denoted by AM[k] (resp, MA[k])

- Public Coins Proof → Verifier MUST send all random strings to prover.
  (Verifier shared randomness with Prover)

AM = AM[2], MA = MA[2] (only 2 rounds!

AM : Verifier (Arthur) starts sending a random string.

MA : Prover (Merlin) starts sending the first message

Note: $AM = AM[2]$
$IP = IP[\mathrm{poly}(n)]$
$= \bigcup_{c>0} IP[n^c]$

Obs. $AM[k] \subseteq IP[k]$, $\forall k \geq 2$

Thm 8.8 $IP[k] \subseteq AM[k+2]$

Thm 8.9 $GNI \in AM[k]$ for some constant $k \geq 2$.

$\boxed{\text{Key Idea}}$ → recasting the problem.

$$S = \{H : H \equiv G_1 \text{ or } H \equiv G_2\}.$$

(easy to prove that $H \in S$ by giving permutation π as witness.)

① $G_1 \not\cong G_2 \implies |S| = 2n!$
② $G_1 \cong G_2 \implies |S| = n!$

} change definition of S

$S = \{(H, \pi) : H \cong G_1$ or $H \cong G_2$, and $\pi \in \mathrm{aut}(H)\}$

The prover has to convince the verifier $\boxed{|S| = 2n!}$ (using Set Lower Bound)

Skip Set Lower Bound Protocol Due to Time Constraints e.g.) Not-SAT ∈ co-NP-complete

believe co-NP ≠ NP

$\boxed{IP = PSPACE}$ (LFKN, Shamir 1990)

Class of Problems that can be solved in Poly Space

runs time can be expo(n)

Proof Overview

- $IP \subseteq PSPACE$ Easy because size of transaction is poly(n)

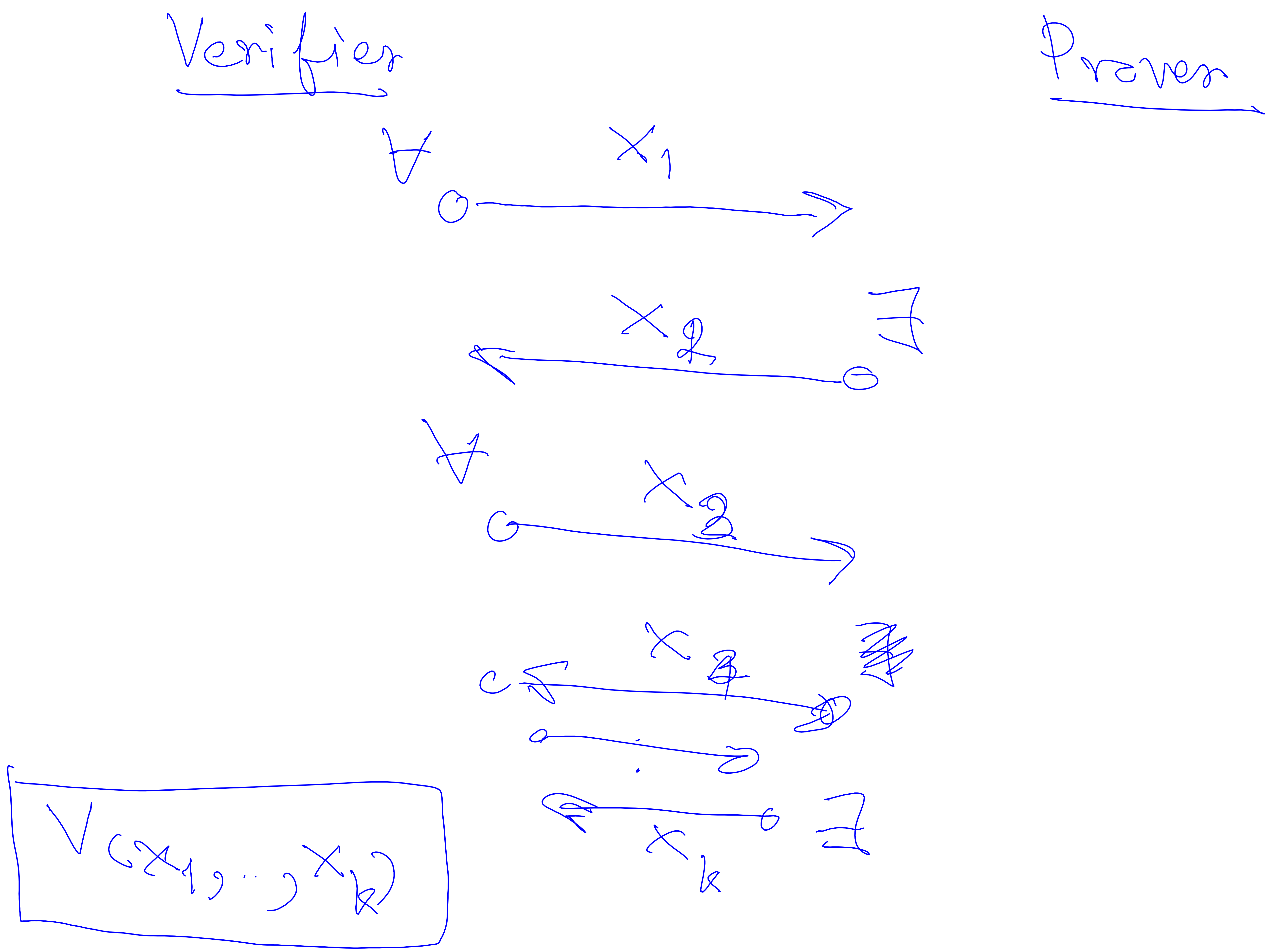
  $k = poly(n)$

  any message $a_i = poly(n)$ space we need

  otherwise, the verifier has to run in super poly.

- $PSPACE \subseteq IP[poly(n)]$ Need to show some PSPACE-complete can be captured poly(n)-rounds IP.

$\Rightarrow$ TQBF $\in IP[poly(n)]$



$$\Phi = \forall x_1 \exists x_2 \forall x_3 \dots \exists x_k \left(V(x_1, \dots, x_k) = 1 \right)$$

Proof. TQBF $\in$ IP[poly(n)] $\Rightarrow \forall L \in$ PSPACE, $L \leq_p$ TQBF
$\Rightarrow L \in$ IP[poly(n)]

difficult part

TQBF: $\boxed{\Phi = \forall x_1 \exists x_2 \forall x_3 \ldots \exists x_n \, f(x_1, x_2, \ldots, x_n)}$

$\exists \underline{x} \, f(x_1, \ldots, x_n) \rightsquigarrow$ SAT $\Rightarrow \Phi \in$ NP $\rightarrow$ Easy.

How about $\forall \underline{x} \, \overline{f(x_1, \ldots, x_n)} \rightsquigarrow \overline{\text{SAT}} \Rightarrow$ How to design the protocol?

Cheat 1: We will design a protocol for $\overline{\text{3SAT}}$ instead of TQBF

Cheat 2: $\Phi \in \overline{\text{3SAT}} \Rightarrow$ # satisfying assignment $= 0$ and ≥ 1 otherwise

We will design a protocol for #SAT

count # of satisfying assignments of CNF-formula

Arithmetization

- transform CNF formula Φ into a polynomial p (over finite field $\mathbb{F}$)

~~$\mathbb{F}_2 = \{0, 1\}$~~

We will use arbitrary finite field $\mathbb{F}$

$$x \wedge y \longleftrightarrow X \cdot Y$$

$$\neg x \longleftrightarrow 1 - X$$

$$x \vee y \longleftrightarrow 1 - (1 - X)(1 - Y)$$

$$x \vee y \vee \neg z \longleftrightarrow 1 - (1 - X)(1 - Y)Z$$

$\Rightarrow$ 3CNF can be transform into polynomial degree 3.

$\boxed{\#SAT_D \in IP}$

Note: Prover has unbounded computational power, i.e., can run in exp time.

$\Rightarrow$ Sumcheck Protocol.

Sumcheck Problem

p is prime

Input
- degree-d polynomial $g(x_1, \ldots, x_n)$ over $\mathbb{F}_p$
- designated value $K \in \mathbb{Z}_0^+$

all computations are under $\mathbb{F}_p$

Goal Decide if $K = \sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \sum_{x_3 \in \{0,1\}} \ldots \sum_{x_n \in \{0,1\}} g(x_1, \ldots x_n)$

$K = \sum_{\underline{x} \in \{0,1\}^n} g(\underline{x})$

↳ That is, sum of evaluation of $g(\underline{x})$ over $\boxed{\underline{x} \in \{0,1\}^n}$

Protocol : Sumcheck Protocol

$s(a) \leq n \cdot 2^n \leq \underbrace{n^n}_{n \log n \text{ bits}}$

Verifier | **Prover**

- If $n = 1$ check if $g(1) + g(0) = K$ (if so, accept ; othw, reject !)

x_1 is free

count $g(x)$ for $x_2, \dots, x_n \in \{0, 1\}$

- Construct function $s(x_1)$

$$\boxed{s(x_1) := \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(x_1, b_2, b_3, \dots, b_n)}$$

$\xleftarrow{\quad s \quad}$

The value can be 2^n

but representable using n bits

- Reject if $\boxed{s(0) + s(1) \neq K}$.

Recursive on $n-1$ vars

Otherwise, random $a \in \mathbb{F}_p$. Recursive on the same protocol to check if $K' = s(a) = \sum_{x} g(a, x_2, \dots, x_n)$

can be any thing $\{1, \dots, p\}$

new target

Verifier $\qquad\qquad\qquad\qquad$ Prover

$\xrightarrow{k_1 = k}$

$\xleftarrow{s_1}$ (expected $\hat{s}_1$)

$\xrightarrow{k_2}$

$\xleftarrow{s_2}$

$k_3 \longrightarrow$

$\xleftarrow{s_3}$

$i \quad \ell \quad s_i$

Reject if ~~because~~ $s_i(0) + s_i(1) \neq k_i$

But, if sum $\neq k \implies s_i(0) + s_i(1) = k_i$

only if $s_i \neq \hat{s}_i$

poly degree d has $\leq d$ roots, i.e.,

$$P_d(x) = \hat{P}_1(x) \cdot \hat{P}_2(x) \cdot \ldots \cdot \hat{P}_{\leq d}(x)$$

at most d value makes $P_d(x) = 0$

Proof

- **Completeness** $\to$ Trivial & $x \in [p] \Rightarrow \Pr_x[P(x)=0] \leq 1 - \frac{d}{p}$
- **Soundness** : Suppose $K \neq \sum_{x \in \{0,1\}^n} g(\underline{x})$

$\boxed{\text{Claim}}$ $\Pr[V \text{ rejects } \langle K, g \rangle] \geq \left(1 - \frac{d}{p}\right)^n$

Proof by induction on n.

Real function

$s(x) - h(x)$

Base case $n=1$ $\Pr[V \text{ rejects}] = \Pr[p(a) = 0]$ for $p(x)$ is poly degree d.

$$\Pr_{a \in \mathbb{F}_p}[s(a) = h(a)] \leq 1 - \frac{d}{p}$$

Proof Continue

Inductive Step. $\boxed{\text{Assume IH is true until } n-1:}$

(At some point prover must lie, i.e., $s \neq h$)

- $\Pr[V \text{ rejects on } \langle k_n, a_n \rangle]$

By IH

$$\geq \boxed{\left(1-\frac{d}{p}\right)} \cdot \Pr[V \text{ rejects on } \langle k_{n-1}, a_{n-1} \rangle]$$

$$\geq \left(1-\frac{d}{p}\right) \cdot \left(1-\frac{d}{p}\right)^{n-1} = \left(1-\frac{d}{p}\right)^{n}$$

$\boxed{\text{Note}}$ We have to choose p to be large enough to keep the sum.

Conclude

$\exists$ an IP protocol for sum check $\rightarrow$ #SAT $\in$ IP

$\rightarrow \overline{3SAT} \in IP$

$\rightarrow \forall\ \overline{f(x)}$ has IP protocol.

$$\boxed{\text{Construct IP protocol for TQBF}}$$

$$\Phi = \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n\ f(x_1, \ldots, x_n)$$

$$\Rightarrow \sum_{b_1} \prod_{b_2} \sum_{b_3} \cdots \prod_{b_n} P_{\Phi}(b_1, \ldots, b_n)$$

need to check if this > 0.