# Interactive Proof System

We have seen interactive proofs, in various disguised forms, in the definitions of **NP**, OTM, Cook reduction and **PH**.

We will see that interactive proofs have fundamental connections to cryptography and approximation algorithms.

---

The purpose of writing a proof is for others to verify it.

It was not until 1985 that the idea of computation through interaction was formally studied by two groups.

- László Babai, with a complexity theoretical motivation;
- Shafi Goldwasser, Silvio Micali and Charles Rackoff, with a cryptographic motivation.

An interactive proof system consists of a prover and a verifier.

1. The prover's goal is to convince the verifier of the validity of an assertion through dialogue.
2. The verifier's objective is to accept/reject the assertion based on the information it has gathered from the dialogue.

---

A verifier is a Probabilistic Turing Machine.

## Synopsis

## Introduction

## Basic Principle

A verifier's job must be easy (polynomial time on input length), otherwise there is no need for any dialogue.

A prover can be as powerful as it takes, as long as the answers it produces are short (polynomial size on input length).

A verifier is not supposed to ask too many questions. Its best bet is to pick up questions randomly.

A prover is supposed to provide an answer no matter what. Its best strategy is to answer faithfully.

## Deterministic Verifier

A $k$-round interaction of $f$ and $g$ on input $x \in \{0,1\}^*$, denoted by $\langle f, g\rangle(x)$, is the sequence $a_1, \ldots, a_k \in \{0,1\}^*$ defined as follows:

$$\begin{array}{rcl}
a_1 & = & f(x), \\
a_2 & = & g(x, a_1), \\
& \vdots & \\
a_{2i+1} & = & f(x, a_1, \ldots, a_{2i}), \quad \text{for } 2i < k \\
a_{2i+2} & = & g(x, a_1, \ldots, a_{2i+1}), \quad \text{for } 2i+1 < k \\
& \vdots &
\end{array}$$

The output of $f$ at the end, noted $\mathrm{out}_f\langle f, g\rangle(x)$, is $f(x, a_1, \ldots, a_k) \in \{0,1\}$.

---

$f, g : \{0,1\}^* \to \{0,1\}^*$ are TM's, and $k(n)$ is a polynomial.

## Deterministic Proof System

We say that a language $L$ has a $k$-round deterministic proof system if there is a TM $\mathbb{V}$ that on input $x, a_1, \ldots, a_k$ runs in $\mathrm{poly}(|x|)$ time, and can have a $k(|x|)$-round interaction with any TM $\mathbb{P}$ such that the following statements are valid:

$$\begin{aligned}
\text{Completeness.} \quad & x \in L \Rightarrow \exists \mathbb{P} : \{0,1\}^* \to \{0,1\}^*.\mathsf{out}_{\mathbb{V}}(\mathbb{V}, \mathbb{P}) = 1, \\
\text{Soundness.} \quad & x \notin L \Rightarrow \forall \mathbb{P} : \{0,1\}^* \to \{0,1\}^*.\mathsf{out}_{\mathbb{V}}(\mathbb{V}, \mathbb{P}) = 0.
\end{aligned}$$

---

**dIP** contains languages with a polynomial round deterministic interactive proof system.

## Deterministic Proof Systems Have One Round Interaction

**Fact.** **dIP** = **NP**.

---

Every **NP** language has a one-round deterministic proof system.

Conversely suppose $L \in \mathbf{dIP}$. There is a P-time TM $\mathbb{V}$ such that

$$x \in L \text{ iff } \exists \mathbb{P} : \{0,1\}^* \to \{0,1\}^*.\mathrm{out}_{\mathbb{V}}(\mathbb{V},\mathbb{P}) = 1 \text{ iff}$$

$$\exists a_1, a_2, \ldots, a_k.\mathbb{V}(x) = a_1 \wedge \mathbb{V}(x, a_1, a_2) = a_3 \wedge \ldots \wedge \mathbb{V}(x, a_1, \ldots, a_k) = 1.$$

The verification time is polynomial.

---

In a deterministic proof system an almighty prover can predict the questions the verifier will ask, say by running the verifier's Turing Machine, and therefore can provide all answers at one go.

We shall only be interested in probabilistic verifiers.

## Interactive Proof with Private Coins

Shafi Goldwasser, Silvio Micali, Charles Rackoff. The Knowledge Complexity of Interactive Proofs. STOC 1985.

Marla has one red sock and one green sock.

How can he convince Arthur, who is color blind, of the fact that the socks are of different color?

## Private Coins Model

The verifier generates an $l$-bits $r$ by tossing coins:

$$r \in_{\mathrm{R}} \{0,1\}^l.$$

The verifier of course knows $r$:

$$a_1 = f(x,r),\ a_3 = f(x,r,a_1,a_2),\ \ldots.$$

The prover cannot see $r$:

$$a_2 = g(x,a_1),\ a_4 = g(x,a_1,a_2,a_3),\ \ldots.$$

Both the interaction $\langle f,g\rangle(x)$ and the output $\mathrm{out}_f\langle f,g\rangle(x)$ are random variables over $r \in_{\mathrm{R}} \{0,1\}^l$.

## **IP**, Interactive Proofs with Private Coins

Suppose $k$ is a polynomial. A language $L$ is in $\mathbf{IP}[k(n)]$ if there's a P-time PTM $\mathbb{V}$ that can have a $k(|x|)$-round interaction with any TM $\mathbb{P}$ and renders valid the following.

Completeness.

$$x \in L \Rightarrow \exists \mathbb{P} : \{0,1\}^* \to \{0,1\}^*. \Pr[\mathrm{out}_{\mathbb{V}}(\mathbb{V}, \mathbb{P}) = 1] \geq 2/3.$$

Soundness.

$$x \notin L \Rightarrow \forall \mathbb{P} : \{0,1\}^* \to \{0,1\}^*. \Pr[\mathrm{out}_{\mathbb{V}}(\mathbb{V}, \mathbb{P}) = 1] \leq 1/3.$$

---

The class **IP** is defined by $\bigcup_{c \geq 1} \mathbf{IP}[cn^c]$.

## **BPP** Verifier, **PSPACE** Prover

1. A verifier is a **BPP** machine.
2. We may assume that a prover is a **PSPACE** machine.
   - There is an optimal prover.
   - A single **PSPACE** prover suffices for all $x \in L$.

---

An almighty prover knows Verifier's algorithm.

- Prover enumerates all answers $a_2, a_4, \ldots$, and uses Verifier's algorithm to calculate the percentage of the random strings that make verifier to accept.

## IP $\subseteq$ PSPACE

**Proposition**. **IP** $\subseteq$ **PSPACE**.

---

The prover is a single **PSPACE** machine. A **PSPACE** machine can simulate both the prover and the verifier.

## Robustness of **IP**

**Fact**. **IP** remains unchanged if we replace the completeness parameter $2/3$ by $1-2^{-n^s}$ and soundness parameter $1/3$ by $2^{-n^s}$.

Proof.

Repeat the protocol $O(n^s)$ times. Then apply Chernoff bound. □

---

Since there is an optimal prover, it doesn't matter if a protocol is repeated sequentially or in parallel.

## Robustness of **IP**

**Fact**. Allowing prover to use a private coin does not change **IP**.

---

By average principle we can construct a deterministic prover from a probabilistic prover that is as good as the latter.

## Perfect Completeness

An interactive proof system has perfect completeness if its completeness parameter is 1.

An interactive proof system has perfect soundness if its soundness parameter is 0.

## Perfect Soundness is Too Strong

1. **IP** with Perfect Completeness = **IP**.
2. **IP** with Perfect Soundness = **NP**.

---

1. **IP** $\subseteq$ **PSPACE**. A problem in **IP** is Karp reducible to `TQBF`. `TQBF` has an interactive proof system with perfect completeness (using the Sumcheck protocol).

2. If $x \in L$, there exists a 'yes' certificate. If $x \notin L$, the verifier always says 'no'.

## Graph Non-Isomorphism

Let GI be the Graph Isomorphism; it is not known to be in **P**.

Let GNI $= \overline{\texttt{GI}}$, it is not known to be in **NP**.

---

The nodes of a graph are represented by the numbers $1, 2, \ldots, n$.

The isomorphism of $G_0$ to $G_1$ is indicated by $\pi(G_0) = G_1$, where $\pi$ is a permutation of the nodes of $G_0$.

## Graph Non-Isomorphism

PROTOCOL: Graph Non-Isomorphism

V: Pick $i \in_{\mathrm{R}} \{0,1\}$. Generate a random permutation graph $H$ of $G_i$. Send $H$ to P.

P: Identify which of $G_0, G_1$ was used to produce $H$ and send the index $j \in \{0,1\}$ to V.

V: Accept if $i = j$; reject otherwise.

## Graph Non-Isomorphism

**Theorem**. $\texttt{GNI} \in \mathbf{IP}$.

Proof.

If $G_0 \simeq G_1$, the prover's guess is as good as anyone's guess.

If $G_0 \not\simeq G_1$, the prover can force the verifier to accept. □

1. O. Goldreich, S. Micali, A. Wigderson. Proofs that Yield Nothing but Their Validity and a Methodology of Cryptographic Protocol Design. FOCS 1986.

## Quadratic Non-Residuosity

A number $a$ is a quadratic residue modulo $p$ if there is some number $b$ such that $a \equiv b^2 \pmod{p}$.

- $\mathtt{QR} = \{(a, p) \mid p \text{ is prime and } \exists b.a \equiv b^2 \pmod{p}\}$ is in **NP**.

---

Let $\mathtt{QNR} = \overline{\mathtt{QR}}$. The problem $\mathtt{QNR}$ is not known to be in **NP**.

## Quadratic Non-Residuosity Protocol

Input.

1. An odd prime number $p$ and a number $a$.

Goal.

1. The prover tries to convince the verifier that $a \in \mathtt{QNR}$.
2. The verifier should reject with good probability if $a \notin \mathtt{QNR}$.

---

V: Pick $r < p$ and $i \in \{0, 1\}$ randomly. If $i = 0$ then send $r^2$ *mod* $p$ to P; otherwise send $ar^2$ *mod* $p$ to P.

P: Identify which case it is and send a number $j \in \{0, 1\}$ to V accordingly.

V: Accept if $j = i$; reject otherwise.

## Quadratic Non-Residuosity

**Theorem**. $\texttt{QNR} \in \mathbf{IP}$.

---

If $a$ is a quadratic residue, then $ar^2$, like $r^2$, is a random quadratic residue modulo $p$. In this case prover can only guess.

If $a$ is not a quadratic residue, then $ar^2$, unlike $r^2$, is a random non-quadratic residue modulo $p$. In this case prover can force verifier to accept.

## Interactive Proof for Permanent

Suppose $A = (a_{j,k})_{1 \leq j,k \leq n}$ is an $n \times n$ matrix. According to the expansion in cofactors,

$$\mathtt{perm}(A) = \sum_{i=1}^{n} a_{1i}\mathtt{perm}(A_{1,i}).$$

Computing the permanent of an $n \times n$ matrix reduces to computing the permanents of $n$ matrices of dimension $(n-1) \times (n-1)$.

---

We design an interactive proof system for $\mathtt{perm}(A)$ using arithmetic method.

## Interactive Proof for Permanent

We look for an $(n-1)\times(n-1)$-matrix $D_A(x)$ such that $D_A(i) = A_{1,i}$.

- $(D_A(x))_{j,k}$ is a univariate polynomial of degree $n-1$, and
- $\texttt{perm}(D_A(x))$ is a univariate polynomial of degree $(n-1)^2$.

---

Vandermonde matrix is nonsingular. Verifier can calculate $D_A(x)$.

$$\begin{pmatrix} 1 & 1 & \ldots & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & k & \ldots & k^{n-2} & k^{n-1} \\ 1 & k+1 & \ldots & (k+1)^{n-2} & (k+1)^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & n & \vdots & n^{n-2} & n^{n-1} \end{pmatrix} \begin{pmatrix} b_0 \\ \vdots \\ b_k \\ b_{k+1} \\ \vdots \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} a_{(j+1)(k+1)} \\ \vdots \\ a_{(j+1)(k+1)} \\ a_{(j+1)k} \\ \vdots \\ a_{(j+1)k} \end{pmatrix}$$

# Interactive Proof for Permanent

PROTOCOL: Permanent

Condition: Both parties know a number $k$ and a matrix $A$.
Prover's goal is to show that $k = \mathtt{perm}(A)$.
Verifier should reject with good probability if $k \neq \mathtt{perm}(A)$.

P: Send to V a polynomial $g(x)$ of degree $(n-1)^2$, which is supposedly $\mathtt{perm}(D_A(x))$.

V: Check if $k = \sum_{i=1}^{n} a_{1i} g(i)$. If not, reject; otherwise pick up $b \in_{\mathrm{R}} \mathrm{GF}(p)$ and ask P to prove $g(b) = \mathtt{perm}(D_A(b))$.

---

One has to deal with an exponential number of monomials to calculate $g(x)$. However verifier can calculate the matrix $D_A(x)$.

## Interactive Proof for Permanent

Let $L_{\texttt{perm}}$ be the language

$$\left\{ \langle A, p, k \rangle \mid p > n^4,\ k = \texttt{perm}(A),\ A \text{ is an } n \times n \text{ matrix over } \mathrm{GF}(p) \right\}.$$

---

**Theorem**. $L_{\texttt{perm}} \in \textbf{IP}$.

Proof.

If $n \leq 3$, use brutal force. Otherwise use the permanent protocol.

Verifier accepts with probability 1 if $k = \texttt{perm}(A)$.

The error rate is bounded by $\frac{1}{3}$. [see next slide.] $\square$

## Interactive Proof for Permanent

Suppose $k \neq \texttt{perm}(A)$ and the prover sends a fake $g(x)$.

- $g(x) - \texttt{perm}(D_A(x))$ has at most $(n-1)^2$ roots.
- The probability of choosing a $b$ such that $g(b) = \texttt{perm}(D_A(b))$ is $\leq \frac{(n-1)^2}{p}$.

The probability of the verifier reaching a wrong answer is less than

$$\frac{(n-1)^2}{p} + \frac{(n-2)^2}{p} + \ldots + \frac{4^2}{p} < \frac{n^3}{p} < \frac{1}{n} < \frac{1}{3}.$$

## Interactive Proof with Public Coins

"We can formulate a decision problem under uncertainty as a new sort of game, in which one opponent is 'disinterested' and plays at random, while the other tries to pick a strategy which maximizes the probability of winning – a 'game against Nature'."

1. Christos Papadimitriou. Games Against Nature. FOCS 1983.

László Babai. Trading Group Theory for Randomness. STOC 1985.

## Interactive Proofs with Public Coins

In a public coins system, the verifier's message is identical to the outcome of the coins tossed at the current round.

---

- Whatever verifier computes, prover can do the same.
- Verifier's actions except for its final decision are oblivious of prover's messages.

## Arthur-Merlin Game

Arthur-Merlin Game = Interactive Proof with Public Coins

- Arthur/Nature is the verifier who tosses public coins, and
- Merlin is the prover.

---

Suppose $k : \mathbf{N} \rightarrow \mathbf{N}$ is a polynomial. Obviously

$$\mathbf{AM}[k(n)] \subseteq \mathbf{IP}[k(n)].$$

## Notational Convention

**MA**, **AM**, **AMA**, **MAMAMA**, ...

## Collapse Theorem

**Theorem** (Babai, 1985). $\mathbf{AM}[k(n)+1] = \mathbf{AM}[k(n)]$ if $k(n) \geq 2$.

---

We shall prove the special case when $k(n)$ is a constant.

**Lemma**. **MA** $\subseteq$ **AM**.

---

Suppose $L \in$ **MA**. The completeness is not affected since

$$x \in \mathbf{L} \quad \Rightarrow \quad \exists a.\Pr_r[\mathbb{V}(x,a,r)=1] \geq 1-\epsilon \quad \Rightarrow \quad \Pr_r[\exists a.\mathbb{V}(x,a,r)=1] \geq 1-\epsilon.$$

Perfect Completeness would survive. Soundness is affected though.

$$x \notin \mathbf{L} \quad \Rightarrow \quad \forall a.\Pr_r[\mathbb{V}(x,a,r)=1] \leq \epsilon \quad \Rightarrow \quad \Pr_r[\exists a.\mathbb{V}(x,a,r)=1] \leq 2^{|a|}\epsilon.$$

Since $a$ is of polynomial size, verifier can reduce the error rate by

- generating polynomial number of random strings and
- applying majority rule after getting the answers.

---

Inductively **MAM** = **AMM** = **AM** and **AMA** = **AAM** = **AM**.

## Arthur-Merlin Hierarchy Collapses

**Theorem** (Babai, 1985). $\mathbf{AM}[k] = \mathbf{AM}[2]$ for all constant $k > 2$.

By Babai Theorem the following abbreviation makes sense.

$$\mathbf{AM} \stackrel{\text{def}}{=} \mathbf{AM}[2].$$

## Speedup Theorem for Unbounded Interaction

**Theorem** (Babai and Moran, 1988). $\mathbf{AM}[k(n)] = \mathbf{AM}[k(n)/2]$ if $k(n) > 2$.

## **AM** has Perfect Completeness

Let $\mathbf{AM}^+$ be the subset of **AM** with perfect completeness.

---

**Theorem**. $\mathbf{AM} = \mathbf{AM}^+$.

Proof.

Goldwasser-Sipser Theorem + Shamir Theorem. □

**Corollary**. **AM** $\subseteq \Pi_2^p$.

---

$x \in L$ iff $\Pr_q[\mathbb{A}(x, q, \mathbb{M}(x, q)) = 1] = 1$ iff $\forall q.\exists a.\mathbb{A}(x, q, a) = 1$, where $\mathbb{M}$ is Merlin's optimal strategy.

**Theorem**. If **coNP** $\subseteq$ **AM**, then **PH** $=$ **AM**.

Proof.

One has **NP** $\subseteq$ **MA** $=$ **MA**$^{+}$ by definition and **coNP** $\subseteq$ **AM** by the assumption, and then **PH** $\subseteq$ **AM** by induction. □

---

**Corollary**. If `GI` is **NP**-complete, then **PH** $=$ **AM**.

Proof.

If `GI` is **NP**-complete, then `GNI` is **coNP**-complete. We will show that `GNI` $\in$ **AM**, hence **coNP** $\subseteq$ **AM**. □

**NP** $\subseteq$ **MA** $\subseteq$ **AM** can be interpreted as saying that **MA** and **AM** are randomized analogues of **NP**.

- In **AM** the randomness is announced first.
- In **MA** the randomness comes afterwards.

## Set Lower Bound Protocol

Set lower bound protocol is based on Carter and Wegman's universal hash function.

1. J. Carter and M. Wegman. Universal Classes of Hash Functions. Journal of Computer and System Sciences. 143-154, 1979. (FOCS 1977)

## Pairwise Independent Hash Function

Let $\mathcal{H}_{n,k}$ be a collection of hash functions from $\{0,1\}^n$ to $\{0,1\}^k$.

We say that $\mathcal{H}_{n,k}$ is pairwise independent if the following hold:

- For each $x \in \{0,1\}^n$ and each $y \in \{0,1\}^k$,
$$\Pr_{h \in_{\mathrm{R}} \mathcal{H}_{n,k}}[h(x) = y] = \frac{1}{2^k}.$$
- For all $x, x' \in \{0,1\}^n$ with $x \neq x'$ and all $y, y' \in \{0,1\}^k$,
$$\Pr_{h \in_{\mathrm{R}} \mathcal{H}_{n,k}}[h(x) = y \wedge h(x') = y'] = \frac{1}{2^{2k}}.$$

## Efficient Pairwise Independent Hash Function

**Theorem**. For every $n$, let $\mathcal{H}_{n,n}$ be $\{h_{a,b}\}_{a,b\in\mathrm{GF}(2^n)}$, where for all $a, b$ the function $h_{a,b} : \mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$ is defined by

$$h_{a,b}(x) = a \cdot x + b. \tag{1}$$

Then the collection $\mathcal{H}_{n,n}$ is efficient pairwise independent.

- $h_{a,b}$ is injective whenever $a \neq 0$.

---

- We get $\mathcal{H}_{n,k}$ from $\mathcal{H}_{n,n}/\mathcal{H}_{k,k}$ by either truncating/padding.
- From now on we shall use the collection $\mathcal{H}_{n,k}$ of functions as defined in (1).

1. Sipser used these functions to prove **BPP** $\subseteq \sum_4^p \cap \prod_4^p$.
2. Stockmeyer applied them to set lower bound for the first time.
3. Babai exploited them in the study of Arthur-Merlin protocol.

---

1. Sipser. A Complexity Theoretic Approach to Randomness. STOC 1983.
2. Stockmeyer. The Complexity of Approximate Counting. STOC 1984.
3. Babai. Trading Group Theory for Randomness. STOC 1985.

Suppose $S$ is a set whose membership can be certified.

- Its membership can be certified by prover, and
- checked by verifier.

---

The set lower bound protocol is a public coins protocol. It allows prover to certify the size of $S$ against a given constant $K$.

- If $|S| \geq K$, then verifier accepts with high probability.
- If $|S| \leq K/2$, then verifier rejects with high probability.

## Motivation

Assume $S \subseteq \{0,1\}^m$ and $2^{k-2} < K \le 2^{k-1}$.

---

Suppose $|S| \ge K$ and $y \in \{0,1\}^k$. Then $\Pr_{h \in_{\mathrm{R}} \mathcal{H}_{m,k}}[y \in h(S)] > \frac{1}{4}$.

By taking $\kappa = k/(2 - \log 3)$ one gets

$$\Pr_{h_1,\ldots,h_\kappa \in_{\mathrm{R}} \mathcal{H}_{m,k}} \left[ y \notin \bigcup_{i=1}^{\kappa} h_i(S) \right] < \left(\frac{3}{4}\right)^{\kappa} = 2^{-k}.$$

Hence

$$\Pr_{h_1,\ldots,h_\kappa \in_{\mathrm{R}} \mathcal{H}_{m,k}} \left[ \exists y \in \{0,1\}^k . y \notin \bigcup_{i=1}^{\kappa} h_i(S) \right] < 1.$$

Conclude that $\{0,1\}^k = \bigcup_{i=1}^{\kappa} h_i(S)$ for some $h_1, \ldots, h_\kappa \in \mathcal{H}_{m,k}$.

## Motivation

Suppose $|S| \leq \frac{K}{p(k)}$ for a polynomial $p(k) \geq 2\kappa$. For all $y \in \{0,1\}^k$ and all $h_1, \ldots, h_\kappa$,

$$\sum_{i=1}^{\kappa} |h_i(S)| \leq 2^{k-2}.$$

## Set Lower Bound Protocol.

---

M: Send $h_1, \ldots, h_\kappa$ to Arthur.

A: Pick $y \in_{\mathrm{R}} \{0,1\}^k$. Send $y$ to Merlin.

M: Send $i, x$ to Arthur, together with a certificate that $x \in S$.

Arthur accepts if $h_i(x) = y$ and the certificate validates $x \in S$; otherwise it rejects.

---

The protocol we have described has perfect completeness.

## Set Lower Bound Protocol

Input.

1. Numbers $K, k$ such that $2^{k-2} < K \leq 2^{k-1}$.
2. $S \subseteq \{0,1\}^m$ such that the membership in $S$ can be certified.

Goal.

1. Prover tries to convince verifier that $|S| \geq K$.
2. Verifier should reject with good probability if $|S| \leq \frac{K}{2}$.

---

Let $\ell = 2\log k$. We transform in P-time the question "$|S| \geq K$ or $|S| \leq K/2\,?$" to

$$\text{“}|S^\ell| \geq K^\ell \text{ or } |S^\ell| \leq K^\ell/2^\ell\,?\text{”}.$$

Then apply the protocol defined on previous slide.

## GNI is in **AM**

Let $S$ be

$$\{\langle H, \pi\rangle \mid H \simeq G_0 \text{ or } H \simeq G_1, \text{ and } \pi \text{ is an automorphism}\}.$$

Observe that

$$\text{if } G_0 \not\simeq G_1 \text{ then } |S| = 2n!$$

and

$$\text{if } G_0 \simeq G_1 \text{ then } |S| = n!.$$

Now apply the set lower bound protocol.

## Can GI be **NP**-Complete?

**Theorem**. If GI is **NP**-complete, then $\sum_2 = \prod_2$.

1. R. Boppana, J. Håstad, and S. Zachos. Does co-NP Have Short Interactive Proofs? Information Processing Letters, 25:127-132, 1987.

## Proof of Boppana-Håstad-Zachos Theorem

If $\mathtt{GI}$ is **NP**-complete, then $\mathtt{GNI}$ is **coNP**-complete. It follows that

- there is a reduction function $f$ such that for every formula $\varphi$ with $2n$ variables, $\forall y \varphi(y)$ if and only if $f(\forall y \varphi(y)) \in \mathtt{GNI}$.

---

Consider an arbitrary $\sum_2 \mathtt{SAT}$ formula $\psi = \exists x \in \{0,1\}^n . \forall y \in \{0,1\}^n . \varphi(x,y)$. Now

$$\psi \text{ iff } \exists x \in \{0,1\}^n . g(x) \in \mathtt{GNI},$$

where $g(x)$ is $f(\forall y \varphi(x,y))$.

---

$\mathtt{GNI}$ has a two round Arthur-Merlin proof system with perfect completeness and soundness error $< 2^{-n}$. Let

- $\mathbb{A}$ be Arthur's algorithm, and
- $m$ be the length of Arthur's questions and Merlin's answers.

## Proof of Boppana-Håstad-Zachos Theorem

We claim that $\psi$ is true if and only if

$$\forall q \in \{0,1\}^m.\exists x \in \{0,1\}^n.\exists a \in \{0,1\}^m.\mathbb{A}(g(x), q, a) = 1, \tag{2}$$

which would show $\sum_2 \subseteq \prod_2$. Notice that $\psi$ is true if and only if

$$\exists x \in \{0,1\}^n.\forall q \in \{0,1\}^m.\exists a \in \{0,1\}^m.\mathbb{A}(g(x), q, a) = 1. \tag{3}$$

---

If (2) holds, that is $\forall q \in \{0,1\}^m.\exists x \in \{0,1\}^n.\exists a \in \{0,1\}^m.\mathbb{A}(g(x), q, a) = 1$, there is some $x_0$ such that for at least $2^{m-n}$ number of $q \in \{0,1\}^m$,

$$\exists a \in \{0,1\}^m.\mathbb{A}(g(x_0), q, a) = 1.$$

This implies that the error rate for the input $g(x_0)$ is $\geq \frac{1}{2^n}$ if $\psi$ does not hold, which would contradict to our assumption. So $\psi$ must be true.

## Public Coins versus Private Coins

## Interaction + Randomness

"...in the context of interactive proof systems, asking random questions is as powerful as asking clever questions."

Goldreich

**Theorem** (Goldwasser-Sipser, 1986). $\mathbf{IP}[k(n)] \subseteq \mathbf{AM}[k(n)+2]$.

---

1. Goldwasser and Sipser. Private Coins versus Public Coins in Interactive Proof Systems. STOC 1986.

The key to the proof of Goldwasser-Sipser Theorem is that Merlin can apply the set lower bound protocol to convince Arthur that the chance for Prover to make Verifier believe is big.

## Goldwasser-Sipser Proof

Suppose $L \in \mathbf{IP}$:

- $l(n)$, the length of random string,
- $2t(n)$, the number of rounds,
- $m(n)$, the message length for both Verifier and Prover,
- $2^{-e(n)}$, the error probability.

For simplicity we abbreviate $l(n), t(n), m(n), e(n)$ to $l, t, m, e$.

## Goldwasser-Sipser Proof

Suppose $r \in \{0,1\}^l$ and $s_j = q_1 a_1 \ldots q_j a_j$, where $j \in [t]$.

- We say $\mathbb{V}(x,r)$ accepts via $s_j$ if $\mathbb{V}$ accepts via a dialogue where the first $2j$ messages are $q_1 a_1 \ldots q_j a_j$.
- For each $q$ we write $a_q$ for the prover's answer.

## Goldwasser-Sipser Proof

The intuition is that Merlin tries to choose an answer set that stands the best chance to convince Arthur. Suppose $s_j$ is given.

- $\pi_{s_j} = \Pr_r[\mathbb{V}(x, r) \text{ accepts via } s_j]$.
- $R_{s_j} = \{r \mid \mathbb{V}(x, r) \text{ accepts via } s_j\}$.
- Group $R_{s_j q}$'s into $l$ classes $\gamma_1, \ldots, \gamma_l$, where

$$\gamma_d = \{R_{s_j q} \mid 2^{d-1} < |R_{s_j q}| \leq 2^d \text{ and } q \in \{0,1\}^m\}.$$

- Let $\gamma_{\max}$ be such that $\bigcup\{R_{s_j q} \mid R_{s_j q} \in \gamma_{\max}\}$ is maximal.
- $S_{j+1} = \gamma_{\max}$.
- $k_{j+1}$ is such that

$$2^{k_{j+1}-2} < |S_{j+1}| \leq 2^{k_{j+1}-1}.$$

Merlin's protocol, round 0:

1. Calculate $S_1$ and $k_1$;
2. Send $k_1$ to Arthur.

Merlin's protocol, round $2j$, where $j \in \{1, \ldots, t\}$:

1. Receive $h_j$ and $z_j$ from Arthur;
2. Find some $q_j \in S_j$ such that $h_j(q_j) = z_j$; abort if it fails;
3. Calculate $a_j$, $S_{j+1}$ and $k_{j+1}$; abort if $S_{j+1} = \emptyset$;
4. Send $q_j$, $a_j$ and $k_{j+1}$ to Arthur.

Merlin's protocol, round $2t + 2$:

1. Receive $h$ and $z$ from Arthur;
2. Find some $\bigcup S_t$ such that $h(r) = z$; abort if it fails;
3. Send $r$ to Arthur.

Arthur's protocol, round 1:

1. Receive $k_1$ from Merlin;
2. Choose $h_1 \in_{\mathrm{R}} \{0,1\}^m \to \{0,1\}^{k_1}$ and $z_1 \in_{\mathrm{R}} \{0,1\}^{k_1}$;
3. Send $h_1$ and $z_1$ to Merlin.

Arthur's protocol, round $2j+1$, where $j \in \{1,\ldots,t-1\}$:

1. Receive $q_j$, $a_j$ and $k_{j+1}$ from Merlin;
2. If $h_j(q_j) \neq z_j$ then reject;
3. Choose $h_{j+1} \in_{\mathrm{R}} \{0,1\}^m \to \{0,1\}^{k_{j+1}}$ and $z_{j+1} \in_{\mathrm{R}} \{0,1\}^{k_{j+1}}$;
4. Send $h_{j+1}$ and $z_{j+1}$ to Merlin.

Arthur's protocol, round $2t+1$:

1. Receive $q_t$, $a_t$ and $k_{t+1}$ from Merlin;
2. If $h_t(q_t) \neq z_t$ then reject;
3. Choose $h \in_{\mathrm{R}} \{0,1\}^l \to \{0,1\}^{k_{t+1}}$ and $z \in_{\mathrm{R}} \{0,1\}^{k_{t+1}}$;
4. Send $h$ and $z$ to Merlin.

Arthur accepts if the following hold

- $\mathbb{V}(x, r, q_1, a_1, \ldots, a_i) = q_{i+1}$ for all $i \in [t]$,
- $\mathbb{V}(x, r, q_1, a_1, \ldots, q_t, a_t) = 1$, and
- $\sum_{1 \leq i \leq t+1} k_i \geq l - t\log(l)$.

Read the original paper for the proof of completeness and soundness condition.

**Theorem**. $\bigcup_{k\geq 2} \mathbf{IP}[k] = \mathbf{IP}[2] = \mathbf{AM}[2] = \bigcup_{k\geq 2} \mathbf{AM}[k] = \mathbf{AM}$.

---

Goldwasser-Sipser Theorem + Babai Theorem.

We will soon see that **AM** $=$ **IP** is unlikely.

## Programme Checking

"Checking is concerned with the simpler task of verifying that a given program returns a correct answer on a given input rather than on all inputs. Checking is not as good as verification, but it is easier to do. It is important to note that unlike testing and verification, checking is done each time a program is run."

---

1. M. Blum and S. Kannan. Designing Programs that Check Their Work. J. ACM, 1995.

## Checker

A checker for a task $T$ is a P-time probabilistic OTM $\mathbb{C}$ that, given a claimed program $P$ for $T$ and an input $x$, the following statements are valid:

- If $\forall y.P(y) = T(y)$, then $\Pr[\mathbb{C}^P(x) \text{ accepts } P(x)] \geq \frac{2}{3}$.
- If $P(x) \neq T(x)$, then $\Pr[\mathbb{C}^P(x) \text{ accepts } P(x)] < \frac{1}{3}$.

---

The checker $\mathbb{C}$ may apply $P$ to a number of randomly chosen inputs before making a decision. So even if $P(x) = T(x)$, the checker may still reject $P(x)$.

## Checker for Graph Nonisomorphism

Suppose $P$ is a program for `GNI`:

- $P(G_1, G_2)$ returns 'yes' if $G_1 \not\cong G_2$ and 'no' if otherwise.

---

A program checker $\mathbb{C}$ for `GNI` can be designed as follow:

1. $P(G_1, G_2) =$'no'.
   - Run $P(G_1^1, G_2^1)$, $P(G_1^1, G_2^2)$, $\ldots$, $P(G_1^1, G_2^n)$, where $G_1^1$ is the graph obtained from $G_1$ by replacing the first node by a complete graph of $n+1$ nodes, $\ldots$.
   - Accept if an isomorphism is found, and reject otherwise.
2. $P(G_1, G_2) =$'yes'.
   - Run the IP protocol for `GNI` using $P$ as the prover for $k$ times.

---

Clearly the checker $\mathbb{C}$ runs in P-time.

## Checker for Graph Nonisomorphism

**Theorem**. If $P$ is a correct program for `GNI`, then $\mathbb{C}$ always says "$P$'s answer is correct". If $P$'s answer is incorrect, then the probability that $\mathbb{C}$ says "$P$'s answer is correct" is less than $2^{-k}$.

---

Perfect completeness.

## Languages that have Checkers

If $L$ has an interactive proof system where the prover can be efficiently implemented using $L$ as an oracle, then $L$ has a checker.

---

**Theorem**. $\mathtt{GI}$, $\sharp\mathtt{SAT}_D$ and $\mathtt{TQBF}$ have checkers.

## Random Self-Reducibility

Checkers can be designed by exploring the fact that the output of a program at an input is related to the outputs of the program on some other inputs.

- The simplest such relationship is random self-reducibility.

---

A problem is randomly self-reducible if solving the problem on any input $x$ can be reduced to solving the problem on a sequence of random inputs $y_1, y_2, \ldots$, where each $y_i$ is uniformly distributed among all inputs.

## An Example

Consider a linear function $f(x) = \sum_{i=1}^{n} a_i x_i : \mathrm{GF}(2^n) \rightarrow \mathrm{GF}(2^n)$.

- Given any $x$, pick some $y$ randomly.
- Compute $f(y)$ and $f(y+x)$.
- Compute $f(x)$ by $f(y) + f(y+x)$.

## Lipton Theorem

**Theorem** (Lipton, 1991). There is a randomized algorithm that, given an oracle that computes the permanent on $1 - \frac{1}{3n}$ fraction of the $n \times n$ matrices on $\mathrm{GF}(p)$, can compute the permanents of all matrices on $\mathrm{GF}(p)$ correctly with high probability.

# Proof of Lipton Theorem

Let $A$ be an input matrix. Pick a matrix $R \in_{\mathrm{R}} \mathrm{GF}(p)^{n\times n}$. Let

$$B(x) = A + xR.$$

Clearly $\mathtt{perm}(B(x))$ is a degree $n$ univariate polynomial.

For $a \neq 0$, $B(a)$ is a random matrix. So the probability that the oracle computes $\mathtt{perm}(B(a))$ correctly is at least $1 - \frac{1}{3n}$.

## Proof of Lipton Theorem

1. Randomly generate $n+1$ distinct nonzero points $a_1, \ldots, a_{n+1}$.
2. Ask the oracle to compute $\texttt{perm}(B(a_i))$ for all $i \in [n+1]$.
   - According to union bound, with probability at most $\frac{n+1}{3n}$, the oracle may compute at least one of $\texttt{perm}(B(a_i))$'s incorrectly.
   - So with probability at least $1 - \frac{n+1}{3n} \approx \frac{2}{3}$, the oracle can compute all $\texttt{perm}(B(a_i))$'s correctly.
3. Finally calculate $\texttt{perm}(A) = \texttt{perm}(B(0))$.
   - $\texttt{perm}(B(x))$ is a univariate polynomial of degree $n$.
   - Construct the polynomial using interpolation.

---

Lipton's algorithm provides a checker for the permanent problem.

## IP = PSPACE

C. Lund, L. Fortnow, H. Karloff, and N. Nisan.

- Algebraic Methods for Interactive Proof Systems. FOCS 1990.

A. Shamir.

- IP = PSPACE. FOCS 1990.

L. Babai, L. Fortnow, and L. Lund.

- Nondeterministic Exponential Time has Two-Prover Interactive Protocols. FOCS 1990.

We only have to prove $\texttt{TQBF} \in \mathbf{IP}$.

We start by looking at an interactive proof system for $\overline{\texttt{SAT}}$.

## Counting the Number of Satisfying Assignments

Let $\#\phi$ be the number of the satisfying assignments of $\phi$.

- $\phi$ is a tautology iff $\#\phi = 2^n$ iff

$$\left( \sum_{b_1,\ldots,b_n \in \{0,1\}} \phi(b_1, \ldots, b_n) \right) = 2^n.$$

---

Let $\#\mathtt{SAT}_D$ be $\{\langle \phi, K \rangle \mid \phi \text{ is a 3CNF and } K = \#\phi\}$.

- This is the decision version of $\#\mathtt{SAT}$.
- An interactive proof system for $\#\mathtt{SAT}_D$ solves $\overline{\mathtt{SAT}}$ as well.

## Arithmetization

Suppose $\phi = \phi_1 \wedge \ldots \wedge \phi_m$ is a 3CNF with $n$ variables.

Let $X_1, \ldots, X_n$ be variables over a finite field $\mathrm{GF}(p)$, where $p$ is a prime in $(2^n, 2^{2n}]$.

---

Arithmetization refers to for example the following conversion:

$$x_i \vee \overline{x_j} \vee x_k \ \mapsto \ 1 - (1 - X_i)X_j(1 - X_k).$$

We let 1 represent the truth value and 0 the false value.

---

We write $p_j(X_1, \ldots, X_n)$ for the arithmetization of $\phi_j$.

We write $p_\phi(X_1, \ldots, X_n)$ for $\prod_{j \in [m]} p_j(X_1, \ldots, X_n)$, the arithmetization of $\phi$.

- $|p_\phi(X_1, \ldots, X_n)| = \mathrm{poly}$. But if we open up the brackets in $p_\phi(X_1, \ldots, X_n)$, we would normally get an expression of exponential size.

## Arithmetization

Clearly

$$\#\phi \;=\; \sum_{b_1\in\{0,1\}} \sum_{b_2\in\{0,1\}} \cdots \sum_{b_n\in\{0,1\}} p_\phi(b_1,\ldots,b_n) \;\leq\; 2^n.$$

Suppose $g(X_1, \ldots, X_n)$ is a degree $d$ polynomial, $K$ an integer.

---

We show how the prover can provide an interactive proof for

$$K = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(b_1, \ldots, b_n). \tag{4}$$

Notice that

$$\sum_{b_2 \in \{0,1\}} \sum_{b_3 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(X_1, b_2, \ldots, b_n) \tag{5}$$

is a univariate polynomial whose degree is bounded by $d$.

---

- It takes exponential time to calculate (5).
- Prover can produce a small size $h(X_1)$ equal to (5).

## Sumcheck Protocol

PROTOCOL: Sumcheck

A: If $n = 1$, check $g(0) + g(1) = K$. If so accept; otherwise reject.
If $n \geq 2$, ask M to send some polynomial equal to (5).

M: Send some polynomial $s(X_1)$ to A.

A: Reject if $s(0) + s(1) \neq K$; otherwise pick $a \in \mathrm{GF}(p)$ randomly.
Recursively use the protocol to check

$$s(a) = \sum_{b_2 \in \{0,1\}} \sum_{b_3 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(a, b_2, \ldots, b_n).$$

---

Sumcheck is a public coins protocol with perfect completeness.

## Sumcheck Protocol

**Claim**. If (4) is true, then $\Pr[\mathbb{V} \text{ accepts}] = 1$.

---

**Claim**. If (4) is false, then $\Pr[\mathbb{V} \text{ rejects}] \geq (1 - \frac{d}{p})^n$.

Proof.

Assume (4) is false.

For $n = 1$, Arthur rejects with probability 1.

- If Merlin returns $h(X_1)$, verifier rejects with probability 1.
- If Merlin returns $s(X_1) \neq h(X_1)$, then $s(X_1) - h(X_1)$ has at most $d$ roots.
- Since Arthur picks up $a$ randomly, $\Pr[s(a) \neq h(a)] \geq 1 - d/p$.

If $s(a) \neq h(a)$, Arthur rejects with probability $\geq (1 - \frac{d}{p})^{n-1}$ by induction, hence the claim. □

## Interactive Proof for $\#\mathtt{SAT}_D$

**Theorem** (Lund, Fortnow, Karloff, Nisan, 1990). $\#\mathtt{SAT}_D \in \mathbf{IP}$.

---

Use the Sumcheck protocol.

## Arithmetization for TQBF

Given a quantified Boolean formula

$$\psi = \forall x_1 \exists x_2 \forall x_3 \ldots \exists x_n . \phi(x_1, \ldots, x_n),$$

the arithmetization of $\psi \Leftrightarrow \top$ could be

$$\prod_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \prod_{b_3 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} p_\phi(b_1, \ldots, b_n) \neq 0. \tag{6}$$

---

The problem is that the degree of (6) could be too high.

## Arithmetization for TQBF

The idea is to use linearization operators

$$
\begin{aligned}
L_{X_i}(p) &= (1 - X_i)p_0 + X_i p_1, \\
\forall_{X_i}(p) &= p_0 p_1, \\
\exists_{X_i}(p) &= 1 - (1 - p_0)(1 - p_1)
\end{aligned}
$$

to obtain a multilinear polynomial, where

$$
\begin{aligned}
p_0 &= p(X_1, \ldots, X_{i-1}, 0, X_{i+1}, \ldots, X_n), \\
p_1 &= p(X_1, \ldots, X_{i-1}, 1, X_{i+1}, \ldots, X_n).
\end{aligned}
$$

1. A. Shen. IP=PSPACE: Simplified Proof. J.ACM, 1992.

Reduce the inequality (6) in $O(n^2)$ time to the equality:

$$\forall_{X_1} L_{X_1} \exists_{X_2} L_{X_1} L_{X_2} \ldots \exists_{X_n} L_{X_1}..L_{X_n}.p_{\phi}(X_1, \ldots, X_n) \;=\; 1. \tag{7}$$

Then apply the modified sumcheck protocol to check if (7) holds.

---

Sumcheck Protocol:

1. Merlin sends $s_1(X_1)$ to Arthur, meant to be the openup of the red-expression in (7).
2. Arthur rejects if $s_1(0)\cdot s_1(1) \neq 1$. Otherwise he chooses $r_1 \in_{\mathrm{R}} \mathrm{GF}(p)$ and asks Merlin to prove
$$(L_{X_1} \exists_{X_2} L_{X_1} L_{X_2} \ldots \exists_{X_n} L_{X_1}..L_{X_n}.p_{\phi}(X_1, \ldots, X_n))\,\{r_1/X_1\} = s_1(r_1).$$
3. Merlin sends $s_2(X_2)$ to Arthur, meant to be the openup of blue-expression$\{r_1/X_1\}$.
4. Arthur rejects if $(1 - r_1)\cdot s_2(0) + r_1\cdot s_2(1) \neq s_1(r_1)$. Otherwise he chooses $r_2 \in_{\mathrm{R}} \mathrm{GF}(p)$ and asks Merlin to prove blue-expression$\{r_1/X_1\}\{r_2/X_2\} = s_2(r_2)$.
5. . . .

## **IP** = **PSPACE**

**Theorem** (Shamir 1990). **IP** = **PSPACE**.

---

Using Sumcheck protocol one sees that `TQBF` is in **IP**.

Remark.

- The proof of **IP** $=$ **PSPACE** does not relativize.
  - Fortnow and Sipser proved in 1988 that $\exists O.\, \mathbf{coNP}^{O} \not\subseteq \mathbf{IP}^{O}$.
  - If **IP** $=$ **PSPACE** had a proof that would relativize, then **coNP** $\subseteq$ **IP** would have a proof that would relativize.
- **IP** $=$ **PSPACE** implies that every problem in **IP** has an interactive proof with perfect completeness.

## AM $\subsetneq$ PSPACE ?

**Theorem**. If **PSPACE** $\subseteq$ **P**$_{/\mathrm{poly}}$ then **PSPACE** $=$ **MA**.

---

If **PSPACE** $\subseteq$ **P**$_{/\mathrm{poly}}$, then the prover in the `TQBF` protocol can be replaced by a P-size circuit family $\{C_n\}_{n\in\mathbf{N}}$.

Define a prover that simply sends the description of $C_{|x|}$ to verifier.

The verifier can now make use of $C_{|x|}$ without the necessity for any further interaction.

1. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proofs. STOC '85.
2. L. Babai and S. Moran. Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes. JCSS, 1988.

---

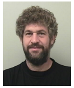
The authors of the two papers shared the first Gödel Prize (1993).

“1989 was an extraordinary year.” László Babai, 1990

---

There e-mail announcements were made within a month of 1989.

1. N. Nisan. “Co-SAT Has Multi-Prover Interactive Proofs”, Nov. 27.
2. C. Lund, L. Fortnow, H. Karloff, and N. Nisan. “The Polynomial Time Hierarchy Has Interactive Proofs”, Dec. 13.
3. A. Shamir. “IP=PSPACE”, Dec. 26.



interaction + randomness + error

## MIP = NEXP

## Exercise

**Theorem.** $\mathbf{MIP} = \mathbf{NEXP}$.

---

1. M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions. STOC 1988.
2. L. Babai, L. Fortnow, and L. Lund. Nondeterministic Exponential Time Has Two Prover Interactive Protocols. Computational Complexity, 1991 (FOCS 90).
3. L. Fortnow, J. Rompel, and M. Sipser. On the Power of Multi-Prover Interactive Protocols. Theoretical Computer Science, 1994.