Proof. Define $f_{12}$ as in (5). As $f$ is symmetric, we only need to consider $f_{12}$.

\[
\mathbb{E} [f_{12}^2] = \mathbb{E}_{x_3 \ldots x_n} \left[ \frac{1}{4} \cdot (f_{12}(00x_3 \ldots x_n) + f_{12}(01x_3 \ldots x_n) + f_{12}(10x_3 \ldots x_n) + f_{12}(11x_3 \ldots x_n)) \right] \\
= \frac{1}{4} \mathbb{E}_{x_3 \ldots x_n} \left[ (f(00x_3 \ldots x_n) - f(11x_3 \ldots x_n))^2 + (f(11x_3 \ldots x_n) - f(00x_3 \ldots x_n))^2 \right] \\
\geq \frac{1}{2} \left( \frac{n-r}{r_0-1} \right) \cdot 2^{-(n-2)} \cdot 4 + \left( \frac{n-r}{r_0-n-1} \right) \cdot 2^{-(n-2)} \cdot 4 \\
= 8 \cdot \left( \frac{(n-r_0+1)(n-r_0)}{n(n-1)} \right) \cdot \left( \frac{n}{r_0-1} \right) + \left( \frac{n-r_1+1}{n(n-1)} \right) \cdot \left( \frac{n}{r_1-1} \right) \cdot 2^{-n}.
\]

Inequality (6) follows by applying Lemma 2.2.

In order to establish inequality (7), we show a lower bound on the principal Fourier coefficient of $f$:

\[
\hat{f}(\theta) \geq 1 - 2 \left( \sum_{s < r_0} \binom{n}{s} + \sum_{s > n-r_1} \binom{n}{s} \right) 2^{-n},
\]

which implies that

\[
\hat{f}(\theta)^2 \geq 1 - 4 \cdot \left( \sum_{s < r_0} \binom{n}{s} + \sum_{s > r_1} \binom{n}{s} \right) 2^{-n}.
\]

\[\square\]
Evolution of “proof”
First there was GORM

GORM = Good Old Regular Mathematics

Pythagoras’s Theorem:

Proof:

\[(a + b)^2 = a^2 + 2ab + b^2\]

Looks legit.
Then there was Russell

Principia Mathematica
Volume 2

Russell and others worked on formalizing GORM proofs.

This meant proofs could be found mechanically. And could be verified mechanically.
Then there were computers

All this played a key role in the birth of computer science.

Computers themselves can find proofs. (automated theorem provers)

Computers can help us find proofs (e.g. 4-Color Theorem)

Are these really proofs?
And now…

Thanks to computer science, a “proof” can be:

- randomized
- interactive
- zero-knowledge
- spot-checkable

Original goal of a “proof”:

*explain* and *understand* a truth.

Now?
Definition:

A language $A$ is in **NP** if

- there is a polynomial time TM $V$
- a polynomial $p$

such that for all $x$:

$$x \in A \iff \exists u \text{ with } |u| \leq p(|x|) \text{ s.t. } V(x, u) = 1$$

“$x \in A$ iff there is a poly-length **proof** $u$ that is verifiable by a poly-time algorithm.”
**NP: A game between a Prover and a Verifier**

**Verifier**
- poly-time
- skeptical

**Prover**
- omniscient
- untrustworthy

Given some input \( x \) (known both to **Verifier** and **Prover**),

**Prover** wants to convince **Verifier** that \( x \in A \).

**Prover** cooks up a “proof” \( u \) and sends it to **Verifier**.

**Verifier** (in poly-time), should be able to tell if the proof is legit.
NP: A game between a Prover and a Verifier

Verifier
poly-time
skeptical

Prover
omniscient
untrustworthy

“Completeness”
If \( x \in A \), there must be some poly-length proof \( u \) that convinces the Verifier.

“Soundness”
If \( x \not\in A \), no matter what “proof” Prover gives, Verifier should detect the lie.
NP: A game between a Prover and a Verifier

Verifier

poly-time
skeptical

Prover

omniscient
untrustworthy

If we have a protocol for $A$ that is complete and sound:

$$A \in \text{NP}.$$
Limitations of NP

Many languages are in \textbf{NP}.

SAT, 3SAT, CLIQUE, MAX-CUT, VERTEX-COVER, SUDOKU, THEOREM-PROVING, 3COL, …

Anything not known to be in \textbf{NP}?

Consider the complement of 3SAT:

Given an \textbf{unsatisfiable} 3SAT formula, how can the \textbf{Prover} prove it is unsatisfiable???

i.e. is the complement of 3SAT in \textbf{NP}?
How can we generalize the NP setting?

NP setting seems too weak for this purpose.

Also, people use more general ways of convincing each other of the validity of statements.

- Make the protocol **interactive**.

  *You can show interaction doesn’t really change the model.*

- Make the verifier **probabilistic**.

  *We don’t think randomization by itself adds more power.*

But, magic happens when you combine the two.
Claim: I can taste the difference between Coke and Pepsi.

How can I prove this to you?
Choose *Coke* or *Pepsi* at random.

Send it to me.

Repeat
Graph Isomorphism Problem

Given two graphs $G_1, G_2$, are they isomorphic?
i.e., is there a permutation $\pi$ of the vertices such that
\[ \pi(G_1) = G_2 \]
Is Graph Isomorphism in $\text{NP}$?

Sure! A good proof is the permutation of the vertices.

Is Graph Non-isomorphism in $\text{NP}$?

No one knows!

But there is a simple randomized interactive proof.
Interactive Proof for Graph Non-isomorphism

Pick at random \( i \in \{1, 2\} \)

Choose a permutation \( \pi \) of vertices at random.

Accept if \( i = j \)
The complexity class \textbf{IP} (Interactive Proof)

We say that a language $A$ is in \textbf{IP} if:

- there is a probabilistic poly-time \textbf{Verifier}.
- there is a computationally unbounded \textbf{Prover}.

```
challenges and responses
```

\textbf{“Completeness”}

If $x \in A$, \textbf{Verifier} accepts with prob. at least $2/3$.

\textbf{“Soundness”}

If $x \not\in A$, \textbf{Verifier} accepts with prob. at most $1/3$. 
How big is IP?

Clearly \( \textbf{NP} \subseteq \textbf{IP} \).

Is \( \text{3SAT} \) in \( \textbf{IP} \)?

\textbf{Yes!}

The complement of any language in \( \textbf{NP} \) is in \( \textbf{IP} \):

\( \textbf{coNP} \subseteq \textbf{IP} \)
How big is IP?

So how powerful are interactive proofs?

How big is IP?

**Theorem:**

\[
\text{IP} = \text{PSPACE}
\]

Adi Shamir

1990

(another application of polynomials)
An interesting corollary:

Suppose in chess, white can always win in $\leq 300$ moves.

How can the wizard prove this to you?
SUMMARY SO FAR

\[ \text{NP} = \text{1-round deterministic interaction between a Prover and a Verifier.} \]

\[ \text{NP} + \text{multiple rounds} = \text{NP} \]

\[ \text{NP} + \text{randomization} = \text{NP} \quad (\text{conjectured as such}) \]

\[ \text{NP} + \text{multiple rounds} + \text{randomization} = \text{IP} = \text{PSPACE} \]
And now…

Thanks to computer science, a “proof” can be:

- randomized
- interactive
- zero-knowledge
- spot-checkable
Does the verifier gain any insight about why the graphs are not isomorphic?

Pick at random $i \in \{1, 2\}$

Choose a permutation $\pi$ of vertices at random.

Accept if $i = j$
The **Verifier** is convinced, but learns **nothing** about why the graphs are non-isomorphic!

The **Verifier** could have produced the communication transcript by himself, with no help from the **Prover**.

A proof with 0 explanatory content!

Is this useful?
Zero-Knowledge Proofs

Examples of scenarios it would be useful.

Which proofs can be turned into zero-knowledge proofs?

- Does every problem in \textit{IP} have a zero-knowledge interactive proof?

- Does every problem in \textit{NP} have a zero-knowledge interactive proof?
Does every problem in NP have a zero-knowledge IP?

Yup! (under plausible cryptographic assumptions)

Goldreich

Micali

Wigderson

1986
Zero-Knowledge Proofs for \textbf{NP}

Does every problem in \textbf{NP} have a zero-knowledge IP?

\textbf{Yup!} (under plausible cryptographic assumptions)

It suffices to show this for your favorite \textbf{NP}-complete problem.

(every problem in \textbf{NP} reduces to an \textbf{NP}-complete prob.)

We’ll pick the Hamiltonian cycle problem.
Hamiltonian cycle problem

Given an undirected graph:

Does it have a cycle that visits every vertex exactly once?
Hamiltonian cycle problem

Given an undirected graph:

Does it have a cycle that visits every vertex exactly once?
Zero-Knowledge Proofs for NP

The protocol

Given undirected graph $G$

**Prover:**

- Picks randomly a permutation of the vertices $\pi$.
- Sends $\pi(G)$ in a “locked” way:
  - for each pair of vertices, there is a locked bit.
  - the bit indicates whether the vertices are connected.

**Verifier:**

- Flips a coin.
- If heads, asks **Prover** to show him the Hamiltonian cycle.
- If tails, asks **Prover** to unlock everything, and asks for $\pi$. 
Zero-Knowledge Proofs for NP

Completeness

Soundness

Zero-knowledge

All is good if:
the “locked” bits work the way they are meant to work.

- **Verifier** shouldn’t be able to unlock them by himself.
- **Prover** shouldn’t be able to change bit values.

Can be realized using **bit commitment schemes**.
(assuming Verifier is computationally bounded)
Does every problem in $\text{IP} = \text{PSPACE}$ have a zero-knowledge proof?

Yup!

"Everything provable is provable in zero-knowledge"
There is a difference between
- zero-knowledge proof for Graph Non-isomorphism
- zero-knowledge proof for Hamiltonian Cycle

**Statistical zero-knowledge:**
Verifier doesn’t learn anything even if it was computationally unbounded.

**Computational zero-knowledge:**
Verifier doesn’t learn anything assuming it cannot unlock the locks in polynomial time.
\textbf{SZK} = set of all problems with statistically zero-knowledge proofs

\textbf{CZK} = set of all problems with computationally zero-knowledge proofs

\textbf{IP} = \textbf{PSPACE} = \textbf{CZK}

\textbf{SZK} is believed to be much smaller.

In fact, it is believed that it does not contain \textbf{NP}-complete problems.
And now…

Thanks to computer science, a “proof” can be:

- randomized
- interactive
- zero-knowledge
- spot-checkable
Scenario:

I have a proof that $1+1 = 2$.

It is a few hundred pages long.

You have to verify its correctness.

Tiny mistake $\rightarrow$ Super annoying to find!
Spot-Checkable Proofs

If only there was a way to “spot-check” the proof:

- check randomly a few bits
- w.h.p. correctly verify the proof

That’s a dream that seems too good to be true.
Or is it?
Question:

Given two graphs $G_0, G_1$, is there a “spot-checkable” proof that they are non-isomorphic?

Exercise:

Find such a proof that is exponentially long.
Spot-Checkable Proofs

Probabilistically Checkable Proofs (PCP) Theorem:

Every problem in \textbf{NP} admits “spot-checkable” proofs of polynomial length.

The verifier can be convinced with high probability by looking only at a \textit{constant} number of bits in the proof.

old proof \hspace{1cm} \text{(poly-length)} \hspace{1cm} \rightarrow \hspace{1cm} \text{new proof} \hspace{1cm} \text{(poly-length)}

tiny local error \hspace{1cm} \rightarrow \hspace{1cm} \text{error almost everywhere}

“New shortcut found for long math proofs!”
Probabilistically Checkable Proofs (PCP) Theorem:

Every problem in $\textbf{NP}$ admits “spot-checkable” proofs of polynomial length.

The verifier can be convinced with high probability by looking only at a constant number of bits in the proof.

1998

Arora-Lund-Motwani-Safra-Sudan-Szegedy
This theorem is equivalent to:

**PCP Theorem (version 2):**
There is some constant $\epsilon$ such that if there is a polynomial-time $\epsilon$-approximation algorithm for MAX-3SAT, then $P = NP$.

(It is $NP$-hard to approximate MAX-3SAT within an $\epsilon$ factor.)

This is called an “hardness of approximation” result.

They are hard to prove!
PCP Theorem is one of the crowning achievements in CS theory!

Proof is a half a semester course.

Blends together:

- P/NP
- random walks
- expander graphs
- polynomials / finite fields
- error-correcting codes
- Fourier analysis
Computer science gives a whole new perspective on proofs:

- can be *probabilistic*
- can be *interactive*
- can be *zero-knowledge*
- can be *spot-checkable*
- can be *quantum mechanical*
Summary

old-fashioned proof + deterministic verifier

NP

randomization + interaction

PSPACE

PSPACE = Computationally Zero-Knowledge (CZK)

"Everything provable is provable in zero-knowledge"

(some special problems are in SZK)
PCP Theorem

Old-fashioned proofs can be turned into spot-checkable. (you only need to check constant number of bits!)

Equivalent to an hardness of approximation result.

Opens the door to many other hardness of approximation results.