Proofs in Three Bits of Less

Jacob Denson

University of British Columbia

January 31, 2018

What Are Interactive Proofs?

- Like 20 Questions – the **verifier** asks yes/no questions to a **prover** until they are convinced a proof is correct, or incorrect.

What Are Interactive Proofs?

- Like 20 Questions – the **verifier** asks yes/no questions to a **prover** until they are convinced a proof is correct, or incorrect.
- More versatile than classical proofs – we can ask questions like

  "Answer Yes If the n'th bit in the binary encoding of your proof is a 1, else 0"

  to recover classical proofs.

What Are Interactive Proofs?

- Like 20 Questions – the **verifier** asks yes/no questions to a **prover** until they are convinced a proof is correct, or incorrect.
- More versatile than classical proofs – we can ask questions like

  "Answer Yes If the n'th bit in the binary encoding of your proof is a 1, else 0"

  to recover classical proofs.
- We can measure a theorem's difficulty by how many questions it takes for us to become convinced.

Watch Out For Tricks!

The verifier *cannot* trust the prover. The could either be making mistakes, or maliciously trying to trick you into thinking the proof is correct! The question

"Is your proof correct?"

Is useless. Questions must be checkable!
(Origins in Cryptography)

Completeness and Soundness

Classical Proofs have

- **Perfect Completeness**: If a statement is true, there exists a series of answers to the questions we give which will convince us the statement is true.

Completeness and Soundness

Classical Proofs have

- **Perfect Completeness**: If a statement is true, there exists a series of answers to the questions we give which will convince us the statement is true.
- **Perfect Soundness**: If a statement is false, *no* series of answers to the questions I give will convince me the statement is true. You can't trick me!

The Catch!

Proofs of very short statements can be arbitrary long. Very inefficient. Consider the theorem

> " This Theorem Cannot Be Proved In Less Than a Googleplex Symbols "

A 53 character theorem with a HUGE proof.

Weakening Proof Conditions

If we let the questions we ask be randomized, then we can weaken completeness and soundness conditions:

Weakening Proof Conditions

If we let the questions we ask be randomized, then we can weaken completeness and soundness conditions:

- **Imperfect Completeness**: If a statement is true, there exists a series of answers to the questions which convince you the truth with probability $\geq 1/2$.

Weakening Proof Conditions

If we let the questions we ask be randomized, then we can weaken completeness and soundness conditions:

- **Imperfect Completeness**: If a statement is true, there exists a series of answers to the questions which convince you the truth with probability $\geq 1/2$.
- **Imperfect Soundness**: If a statement is false, *every* series of answers to the questions I give will fail to convince me the statement is true with probability $\geq 1 - \varepsilon$. It's unlikely you'll trick me!

The PCP Theorem

Theorem (1998, Arora, Lund, Motwani, Sudan, Szegedy)

There exists a universal constant K such that, for any "feasibly checkable" mathematical problem, we can check the problem with perfect soundness and imperfect completeness in K questions.

Theorem (1997, Håstad)

If you only want imperfect completeness, you can set $K = 3$!

A Practical Application: Bitcoin

- Bitcoin is a decentralized currency; there's no bank verifying the correctness of transactions.

A Practical Application: Bitcoin

- Bitcoin is a decentralized currency; there's no bank verifying the correctness of transactions.
- Volunteers known as 'miners' verify correctness in huge ledgers in exchange for small amounts of currency.

A Practical Application: Bitcoin

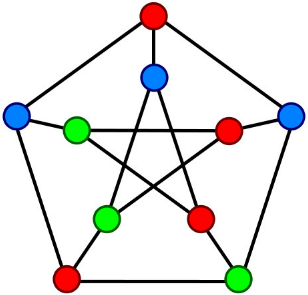
- Bitcoin is a decentralized currency; there's no bank verifying the correctness of transactions.
- Volunteers known as 'miners' verify correctness in huge ledgers in exchange for small amounts of currency.
- The PCP theorem has been proposed as a way to make fraud detection in Bitcoin 'scalable' – in exchange for accidently identifying true claims as false every so often, we only need a constant number of queries to become convinced of a transaction.

Globalization of Local Properties

- We can view a set of questions that convince us of a theorem as an *encoding* of the proof of the theorem. The PCP theorem says that local properties can be encoded globally.

Globalization of Local Properties

- We can view a set of questions that convince us of a theorem as an *encoding* of the proof of the theorem. The PCP theorem says that local properties can be encoded globally.
- $O(|V|)$ questions to determine if a graph has a 3-coloring. The PCP theorem says we can encode a coloring of a graph in such a way that, if the graph fails to be colorable, it will be displayed globally in the encoding – we can detect it with $O(1)$ samples.



Globalization of Local Properties

- We can determine if an integer has 100 prime factors with high probability in three questions!

Globalization of Local Properties

- We can determine if an integer has 100 prime factors with high probability in three questions!
- We can't determine the factorization, just that the factorization exists.

Globalization of Local Properties

- We can determine if an integer has 100 prime factors with high probability in three questions!
- We can't determine the factorization, just that the factorization exists.
- Fourier Analysis is useful for globalizing local properties – L^∞ norm of f is local, but can be controlled by L^1 norm of $\hat{f}$, which is a global property.

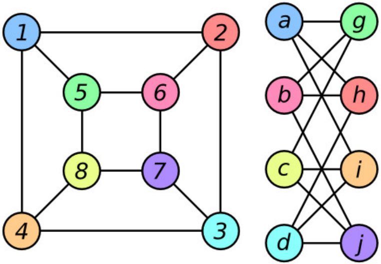
Sushi Break + An Analogy

https://dynermite.wordpress.com/2011/05/04/adorable/

- Any questions so far?

First Example: PCP Checker For Graph Nonisomorphism

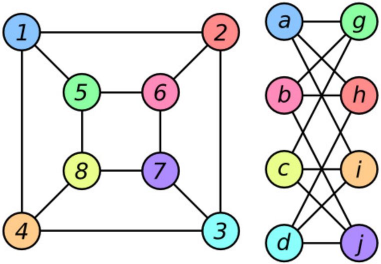
- Two graphs are isomorphic if they can be obtained from one another by relabeling vertices. We take two graphs G_0 and G_1 on n vertices, and ask if they are *not* isomorphic to each other.



http://proopnarine.files.wordpress.com/2010/02/isomorphic.png

First Example: PCP Checker For Graph Nonisomorphism

- Two graphs are isomorphic if they can be obtained from one another by relabeling vertices. We take two graphs G_0 and G_1 on n vertices, and ask if they are *not* isomorphic to each other.
- A proof that two graphs are isomorphic is short – just give an isomorphism. A proof that two graphs are not isomorphic is *very difficult*: check by cases. We give a 1 query proof.



http://proopnarine.files.wordpress.com/2010/02/isomorphic.png

The Checker

- Ask questions of the following variety, for some graph H

  " If H is isomorphic to G_0, output 0. If H is isomorphic to G_2, output 0. Otherwise, output an arbitrary result. "

  Note that if $G_0 \cong G_1$, the question doesn't even make sense.

The Checker

- Ask questions of the following variety, for some graph H

  " If H is isomorphic to G_0, output 0. If H is isomorphic to G_2, output 0. Otherwise, output an arbitrary result. "

  Note that if $G_0 \cong G_1$, the question doesn't even make sense.
- To obtain a question at random, pick an index $i \in \{0, 1\}$ and a permutation $\pi \in S_n$ uniformly at random. Let $H = \pi(G_i)$. We are convinced of the proof if answer to our question corresponds to the correct answer $H \cong G_i$.

The Checker

- Ask questions of the following variety, for some graph H

  " If H is isomorphic to G_0, output 0. If H is isomorphic to G_2, output 0. Otherwise, output an arbitrary result. "

  Note that if $G_0 \cong G_1$, the question doesn't even make sense.
- To obtain a question at random, pick an index $i \in \{0, 1\}$ and a permutation $\pi \in S_n$ uniformly at random. Let $H = \pi(G_i)$. We are convinced of the proof if answer to our question corresponds to the correct answer $H \cong G_i$.
- If G_0 *is* isomorphic to G_1, then H is independent of i, so $\mathbf{P}(i = 0|H) = \mathbf{P}(i = 1|H) = 1/2$, and so regardless of the answer the prover gives, they have a 50% chance of getting the answer wrong.

But Wait!

- How many different questions can we ask? About $O(n!)$. We can view this as an encoding of the graph in $O(n!)$ bits.

But Wait!

- How many different questions can we ask? About $O(n!)$. We can view this as an encoding of the graph in $O(n!)$ bits.
- If original graph has m vertices, can be encoded in $O(m)$ bits.

But Wait!

- How many different questions can we ask? About $O(n!)$. We can view this as an encoding of the graph in $O(n!)$ bits.
- If original graph has m vertices, can be encoded in $O(m)$ bits.
- HUGE increase in size of encoding to obtain globality. The PCP theorem guarantees that we only increase our encoding by $O(n)$. The PCP theorem doesn't apply to the graph nonisomorphism problem because it isn't feasibly checkable – we only know it to be checkable in exponential time, by cases.

Local Proofs: The BLR test

- Local proofs are a key tool for moving from local properties to global properties. The most important one is the **Blum-Luby-Rosenfeld Test**.

Local Proofs: The BLR test

- Local proofs are a key tool for moving from local properties to global properties. The most important one is the **Blum-Luby-Rosenfeld Test**.
- Answers the question: How many queries does it take to check that $f : \mathbf{F}_2^n \to \mathbf{F}_2$ is linear (Answer: 3).

Local Proofs: The BLR test

- Local proofs are a key tool for moving from local properties to global properties. The most important one is the **Blum-Luby-Rosenfeld Test**.
- Answers the question: How many queries does it take to check that $f : \mathbf{F}_2^n \to \mathbf{F}_2$ is linear (Answer: 3).
- Take $X, Y \in \mathbf{F}_2^n$ uniformly at random. We are convinced if

$$f(X+Y) = f(X) + f(Y)$$

Local Proofs: The BLR test

- Local proofs are a key tool for moving from local properties to global properties. The most important one is the **Blum-Luby-Rosenfeld Test**.
- Answers the question: How many queries does it take to check that $f : \mathbf{F}_2^n \to \mathbf{F}_2$ is linear (Answer: 3).
- Take $X, Y \in \mathbf{F}_2^n$ uniformly at random. We are convinced if

$$f(X+Y) = f(X) + f(Y)$$

- Perfect completeness: If f is linear, we are always convinced.

Local Proofs: The BLR test

- Local proofs are a key tool for moving from local properties to global properties. The most important one is the **Blum-Luby-Rosenfeld Test**.
- Answers the question: How many queries does it take to check that $f : \mathbf{F}_2^n \to \mathbf{F}_2$ is linear (Answer: 3).
- Take $X, Y \in \mathbf{F}_2^n$ uniformly at random. We are convinced if

$$f(X + Y) = f(X) + f(Y)$$

- Perfect completeness: If f is linear, we are always convinced.
- Local soundness: If f convinces us it is linear with probability $1 - \varepsilon$, there is a linear function g such that $d(f, g) \leq \varepsilon$, where

$$d(f, g) = \frac{\{x \in \mathbf{F}_2^n : f(x) \neq g(x)\}}{2^n}$$

is the **Hamming distance** between two functions.

Local Correctibility

- Linearity can Fail Locally, but we can cheat.

Local Correctibility

- Linearity can Fail Locally, but we can cheat.
- Suppose $d(f, g) \leq \varepsilon$. Then f and g agree with a large number of inputs, but differ on a couple of inputs deterministcally.

Local Correctibility

- Linearity can Fail Locally, but we can cheat.
- Suppose $d(f,g) \leq \varepsilon$. Then f and g agree with a large number of inputs, but differ on a couple of inputs deterministcally.
- Given *any* x, can we calculate $g(x)$ using only $f(x)$, even if $f(x) \neq g(x)$?

Local Correctibility

- Linearity can Fail Locally, but we can cheat.
- Suppose $d(f,g) \leq \varepsilon$. Then f and g agree with a large number of inputs, but differ on a couple of inputs deterministcally.
- Given *any* x, can we calculate $g(x)$ using only $f(x)$, even if $f(x) \neq g(x)$?
- Take $Y \in \mathbf{F}_2^n$ uniformly at random. Then $f(x+Y) - f(Y) = g(x)$ with probability $\geq 1 - 2\varepsilon$.

Local Correctibility

- Linearity can Fail Locally, but we can cheat.
- Suppose $d(f, g) \leq \varepsilon$. Then f and g agree with a large number of inputs, but differ on a couple of inputs deterministcally.
- Given *any* x, can we calculate $g(x)$ using only $f(x)$, even if $f(x) \neq g(x)$?
- Take $Y \in \mathbf{F}_2^n$ uniformly at random. Then $f(x+Y) - f(Y) = g(x)$ with probability $\geq 1 - 2\varepsilon$.
- Even though linearity can fail locally, we can think of f as an encoding of g if it is close enough, so essentially, we can view all functions which are close to linear functions – we've moved from local error to global error.

The Big PCP Test: Quadratic Equations

- Consider the problem of determining whether a series of quadratic equations over $\mathbf{F}_2$ is solvable, e.g.

$$x_1^2 + x_2 x_3 + x_4^2 = 1 \qquad x_5^2 + x_2 x_6 + x_1^2 = 0$$

The Big PCP Test: Quadratic Equations

- Consider the problem of determining whether a series of quadratic equations over $\mathbf{F}_2$ is solvable, e.g.

$$x_1^2 + x_2x_3 + x_4^2 = 1 \qquad x_5^2 + x_2x_6 + x_1^2 = 0$$

- Since $x_i^2 = x_i$, we can assume all monomials have degree two.

The Big PCP Test: Quadratic Equations

- Consider the problem of determining whether a series of quadratic equations over $\mathbf{F}_2$ is solvable, e.g.

$$x_1^2 + x_2x_3 + x_4^2 = 1 \qquad x_5^2 + x_2x_6 + x_1^2 = 0$$

- Since $x_i^2 = x_i$, we can assume all monomials have degree two.
- Since this problem is **NP Hard**, all feasibly checkable problems are reducible to this problem, so a PCP theorem for this problem gives a PCP checker for all problems. However, we won't only use a linear number of questions.

The Test

- If someone has a solution x, our checker will ask the following two kinds of questions:

The Test

- If someone has a solution x, our checker will ask the following two kinds of questions:
  - " For some subset $S \subset [n]$ of indices, what is $\sum_{i \in S} x_i$. "

The Test

- If someone has a solution x, our checker will ask the following two kinds of questions:
  - " For some subset $S \subset [n]$ of indices, what is $\sum_{i \in S} x_i$. "
  - " For some subset $S \subset [n] \times [n]$, what is $\sum_{(i,j) \in S} x_i x_j$. "

The Test

- If someone has a solution x, our checker will ask the following two kinds of questions:
  - " For some subset $S \subset [n]$ of indices, what is $\sum_{i \in S} x_i$. "
  - " For some subset $S \subset [n] \times [n]$, what is $\sum_{(i,j) \in S} x_i x_j$. "
- These values will enable us to check quadratic equations in very few queries.

The Test

- We start by picking subsets S and T, S', T', uniformly at random, and ask if

$$\sum_{i \in S} x_i + \sum_{i \in T} x_i = \sum_{i \in S \Delta T} x_i$$

$$\sum_{(i,j) \in S'} x_i x_j + \sum_{(i,j) \in T'} x_i x_j = \sum_{(i,j) \in S' \Delta J'} x_i x_j$$

The Test

- We start by picking subsets S and T, S', T', uniformly at random, and ask if

$$\sum_{i \in S} x_i + \sum_{i \in T} x_i = \sum_{i \in S \Delta T} x_i$$

$$\sum_{(i,j) \in S'} x_i x_j + \sum_{(i,j) \in T'} x_i x_j = \sum_{(i,j) \in S' \Delta J'} x_i x_j$$

- Note this is the BLR test if the maps $S \mapsto \sum x_i$ and $S' \mapsto \sum x_i x_j$ are *linear*, where we view $2^{[n]}$ and $2^{[n] \times [n]}$ as vector spaces over $\mathbf{F}_2$ with $S + T := S \Delta T$.

The Test

- We start by picking subsets S and T, S', T', uniformly at random, and ask if

$$\sum_{i \in S} x_i + \sum_{i \in T} x_i = \sum_{i \in S \Delta T} x_i$$

$$\sum_{(i,j) \in S'} x_i x_j + \sum_{(i,j) \in T'} x_i x_j = \sum_{(i,j) \in S' \Delta J'} x_i x_j$$

- Note this is the BLR test if the maps $S \mapsto \sum x_i$ and $S' \mapsto \sum x_i x_j$ are *linear*, where we view $2^{[n]}$ and $2^{[n] \times [n]}$ as vector spaces over $\mathbf{F}_2$ with $S + T := S \Delta T$.
- If a particular answer passes with high probability, then it is close to an *actual* linear operator on $2^{[n]}$ and $2^{[n] \times [n]}$, and all such linear operators are given by some x. Local correction means we can calculate values of a real x.

The Test

- If the last questions passed, we can assume we can calculate the sums for an *actual* x, rather than the answers the prover is giving to us, which might be false.

The Test

- If the last questions passed, we can assume we can calculate the sums for an *actual* x, rather than the answers the prover is giving to us, which might be false.
- If the quadratic equations are

$$\sum_{(i,j)\in T_k} x_i x_j = b_k$$

for some $k \in \{1, \ldots, m\}$, then pick a random subset $S = \{i_1, \ldots, i_l\} \subset [m]$, set $T = T_{i_1} \Delta \ldots \Delta T_{i_l}$ and test if

$$\sum_{(i,j)\in T} x_i x_j = \sum_{i\in T} b_i$$

The Test

- If the last questions passed, we can assume we can calculate the sums for an *actual* x, rather than the answers the prover is giving to us, which might be false.
- If the quadratic equations are

$$\sum_{(i,j)\in T_k} x_i x_j = b_k$$

for some $k \in \{1, \ldots, m\}$, then pick a random subset $S = \{i_1, \ldots, i_l\} \subset [m]$, set $T = T_{i_1} \Delta \ldots \Delta T_{i_l}$ and test if

$$\sum_{(i,j)\in T} x_i x_j = \sum_{i\in T} b_i$$

- If a single equation fails to be satisfied, then the linear function on left doesn't equal the linear function on right, and therefore disagrees on half the inputs. Fails half the time.

Why Do We Care? Complexity Theory

- A problem is in **NP** if a solution can be checked in polynomial time, and in **P** if the solution can be found in polynomial time.

Why Do We Care? Complexity Theory

- A problem is in **NP** if a solution can be checked in polynomial time, and in **P** if the solution can be found in polynomial time.
- Determining a three coloring in a graph is hard, but I can check a given coloring is a three coloring very quickly – $O(n^2)$ in the number of vertices, a polynomial bound.

Why Do We Care? Complexity Theory

- A problem is in **NP** if a solution can be checked in polynomial time, and in **P** if the solution can be found in polynomial time.
- Determining a three coloring in a graph is hard, but I can check a given coloring is a three coloring very quickly – $O(n^2)$ in the number of vertices, a polynomial bound.
- It is an open problem whether we can factor an integer fast – i.e. is integer factorization in **P**. However, if you give me a set of factors, I can determine if they are the factors of a particular integer just by multiplying the integers together – simple!

Why Do We Care? Complexity Theory

- A problem is in **NP** if a solution can be checked in polynomial time, and in **P** if the solution can be found in polynomial time.
- Determining a three coloring in a graph is hard, but I can check a given coloring is a three coloring very quickly – $O(n^2)$ in the number of vertices, a polynomial bound.
- It is an open problem whether we can factor an integer fast – i.e. is integer factorization in **P**. However, if you give me a set of factors, I can determine if they are the factors of a particular integer just by multiplying the integers together – simple!
- The current state of the art factorization algorithm runs in time proportional to

$$O\left(\exp\sqrt[3]{\frac{64}{9}n(\log n)^2}\right)$$

  where n is the bit number to represent the number we factor.

Cook's Theorem

Theorem (Canada, Cook, 1971; USSR, Levin, 1969)

There exists problems, known as **NP Hard** *problems, on which every feasibly checkable problem, known as an* **NP problem**, *can be encoded in the NP hard problem in polynomial time.*

Cook's Theorem

Theorem (Canada, Cook, 1971; USSR, Levin, 1969)

There exists problems, known as **NP Hard** *problems, on which every feasibly checkable problem, known as an* **NP problem**, *can be encoded in the NP hard problem in polynomial time.*

- If an **NP** hard problem has a polynomial time solution, *all* **NP** problems are solvable in polynomial time, so $\mathbf{NP} = \mathbf{P}$. Since most people believe $\mathbf{NP} \neq \mathbf{P}$, **NP** hard problems are *hard* to solve efficiently.

Cook's Theorem

Theorem (Canada, Cook, 1971; USSR, Levin, 1969)

There exists problems, known as **NP Hard** *problems, on which every feasibly checkable problem, known as an* **NP problem**, *can be encoded in the NP hard problem in polynomial time.*

- If an **NP** hard problem has a polynomial time solution, *all* **NP** problems are solvable in polynomial time, so $\mathbf{NP} = \mathbf{P}$. Since most people believe $\mathbf{NP} \neq \mathbf{P}$, **NP** hard problems are *hard* to solve efficiently.
- The first **NP** hard problem discovered was 3-SAT. Given a logical formula, e.g.

$$(x_1 \vee x_2 \vee x_4) \wedge (x_5 \vee x_1 \vee x_7) \wedge (x_1 \vee x_2 \vee x_4)$$

a conjunction of three variable disjunctions, can we give variables truth values to make the statement true?

Approximation Algorithms

- **NP** hard problems cannot be solved efficiently, but maybe we can ‘approximately’ solve them efficiently.

Approximation Algorithms

- **NP** hard problems cannot be solved efficiently, but maybe we can 'approximately' solve them efficiently.
- Rather than asking if a formula is satisfiable, turn into an optimization problem, *how many* clauses can we satisfy.

Approximation Algorithms

- **NP** hard problems cannot be solved efficiently, but maybe we can 'approximately' solve them efficiently.
- Rather than asking if a formula is satisfiable, turn into an optimization problem, *how many* clauses can we satisfy.
- For $A < 1$, an A approximation algorithm is an efficient algorithm such that if it is possible to satisfy a fraction B of the clauses, we find an assignment satisfying AB clauses.

Approximation Algorithms

- **NP** hard problems cannot be solved efficiently, but maybe we can 'approximately' solve them efficiently.
- Rather than asking if a formula is satisfiable, turn into an optimization problem, *how many* clauses can we satisfy.
- For $A < 1$, an A approximation algorithm is an efficient algorithm such that if it is possible to satisfy a fraction B of the clauses, we find an assignment satisfying AB clauses.
- A random assignment satisfies $7/8$ of the clauses, and we can derandomize to get $7/8$ approximation algorithm.

Generalizing Cook's Lemma

- Is there a limit to how well we can approximate a problem? Maybe we can find a $1 - \varepsilon$ approximation for every ε (true for certain problems, like knapsack).

Generalizing Cook's Lemma

- Is there a limit to how well we can approximate a problem? Maybe we can find a $1 - \varepsilon$ approximation for every ε (true for certain problems, like knapsack).
- Every problem can be reduced to 3SAT, but the solutions aren't 'globally' separated. A solution to 3SAT can fail at a single clause, but satisfy almost all clauses.

Generalizing Cook's Lemma

- Is there a limit to how well we can approximate a problem? Maybe we can find a $1 - \varepsilon$ approximation for every ε (true for certain problems, like knapsack).
- Every problem can be reduced to 3SAT, but the solutions aren't 'globally' separated. A solution to 3SAT can fail at a single clause, but satisfy almost all clauses.
- Each possible random query in PCP theorem can be viewed as a clause of a 3SAT problem, that separates incorrect solutions ($\leq 1/2$ clauses satisfiable), from correct clauses ($1 - \varepsilon$ of clauses solvable).

Generalizing Cook's Lemma

- Is there a limit to how well we can approximate a problem? Maybe we can find a $1-\varepsilon$ approximation for every ε (true for certain problems, like knapsack).
- Every problem can be reduced to 3SAT, but the solutions aren't 'globally' separated. A solution to 3SAT can fail at a single clause, but satisfy almost all clauses.
- Each possible random query in PCP theorem can be viewed as a clause of a 3SAT problem, that separates incorrect solutions ($\leq 1/2$ clauses satisfiable), from correct clauses ($1-\varepsilon$ of clauses solvable).
- Håstad used this to show that for any $\varepsilon > 0$, if there is a $7/8+\varepsilon$ approximation algorithm for $3SAT$, then $\mathbf{P} = \mathbf{NP}$. Thus the 7/8 approximation given before is best possible.

Where to Go From Here?

- The rest of PCP theorem proof involves interesting combinatorial techniques (e.g. expander graphs).

Where to Go From Here?

- The rest of PCP theorem proof involves interesting combinatorial techniques (e.g. expander graphs).
- Modern day research in approximability involves deriving checkable proofs for various problems – interesting probability theory (unique games conjecture, majority is stablest (2007), hypercontractivity, discrete log sobolev inequalities, etc).

Where to Go From Here?

- The rest of PCP theorem proof involves interesting combinatorial techniques (e.g. expander graphs).
- Modern day research in approximability involves deriving checkable proofs for various problems – interesting probability theory (unique games conjecture, majority is stablest (2007), hypercontractivity, discrete log sobolev inequalities, etc).
- Learning Theory – Connections between problems that have quick checkable proofs and learning models that are easy to learn. We saw linear functions are easy to learn.

Where to Go From Here?

- The rest of PCP theorem proof involves interesting combinatorial techniques (e.g. expander graphs).
- Modern day research in approximability involves deriving checkable proofs for various problems – interesting probability theory (unique games conjecture, majority is stablest (2007), hypercontractivity, discrete log sobolev inequalities, etc).
- Learning Theory – Connections between problems that have quick checkable proofs and learning models that are easy to learn. We saw linear functions are easy to learn.
- Connections to Additive Combinatorics: Gower's Inverse Conjecture, constructions of PCPs.

Where to Go From Here?

- The rest of PCP theorem proof involves interesting combinatorial techniques (e.g. expander graphs).
- Modern day research in approximability involves deriving checkable proofs for various problems – interesting probability theory (unique games conjecture, majority is stablest (2007), hypercontractivity, discrete log sobolev inequalities, etc).
- Learning Theory – Connections between problems that have quick checkable proofs and learning models that are easy to learn. We saw linear functions are easy to learn.
- Connections to Additive Combinatorics: Gower's Inverse Conjecture, constructions of PCPs.
- References:

  - Arora, Barak; Computational Complexity.
  - O' Donell; Analysis of Boolean Functions.
  - Bellare, Coppersmith, Håstad, Kiwi, Sudan; Linearity Testing in Characteristic Two.