Proofs in Three Bits or Less

Jacob Denson

November 24rd, 2017

Some proofs are way too long. Shinichi Mochizuki's 2012 proof of the ABC conjecture is about 500 pages long. Leading number theorists are still trying to understand the proof today. The entire classification of finite simple groups amounts to an estimated 100000-200000 pages. Something has to be done to stop this madness!

My first idea was to change the way mathematics is structured. Think of a kind of mathematical social media website where mathematicians post proofs, but these proofs are limited to 140 characters or less. Then the proofs will be accessible to every mathematician, and even to the general public as well! And it's not like we'll run out of proofs, because there are about 26^{140} different sequences of characters we can make in 140 characters or less, and at least some portion of this sequence will form mathematical proofs. We'll be busy for a while!

But while thinking about this new idea, my friend told me about a radical new way of viewing proofs in computing science, known as **interactive proofs**, which have emerged from cryptography. The idea is kind of like the party game 20 questions. Suppose we wish to determine whether a given statement is true or false by asking questions to a given person, known as a **prover**, or **oracle**. Whereas classical proofs are static, in interactive proofs we ask a set of yes/no questions to the prover, and then determine whether the prover really has a proof of the result. After a fixed number of questions, we must decide whether we are convinced by the answers we recieved. If the number of questions is small, this makes proofs very readable!

An additional problem is that the oracle might make errors, or more maliciously, might have lied in response to your questions. Thus we must ask questions which are 'checkable'. An interesting question to ask is how many questions we must ask to become convinced of a particular mathematical statement? By convinced, we mean some of the following properties

- **Perfect Completeness**: If a statement is true, there exists a series of answers to our questions which will convince us the statement is correct.
- **Perfect Soundness**: If a statement is false, then *every* series of answers to our question will fail to convince us the statment is true.

If we choose our questions randomly, we can introduce weaker conditions.

- **Imperfect Completeness**: If a statement is true, then there exists a series of answers to our randomly chosen questions which will convince us the statement is correct with probability greater than $1/2$.
- **Imperfect Soundness**: If a statement is false, then *every* series of answers to our question will fail to convince us with probability greater than $1 - \varepsilon$.

Classical proofs are just interactive proofs that are perfectly complete and sound. The cost is inefficiency; for any increasing function $f : \mathbf{N} \to \mathbf{N}$, there exists theorems of length n which require at least $f(n)$ steps to verify. However, if we are willing to avoid perfection, then there is a magical result from theoretical computing science.

Theorem 1 (The PCP Theorem)**.** *There exists a universal constant K such that, if a theorem is 'feasibly checkable', then the theorem is checkable with perfect soundness, and imperfect completeness in K questions.*

If we are willing to have imperfect soundness, so we can be fooled, we obtain an even better version, due to Håstad:

Theorem 2. *For $\varepsilon > 0$, every 'feasibly checkable' theorem is checkable with imperfect soundness and imperfect completeness in only 3 questions!*

In essense, this means that we can prove anything with only three bits! By a feasibly checkable theorem, we mean a problem such that a proof can be verified or rejected in a polynomial number of computational steps. For instance, the problem of checking whether a graph is three colorable is very difficult, and unless $\mathbf{P} = \mathbf{NP}$, we cannot check this in polynomial time. However, if someone gives us a *particular* three coloring, we can just check whether the coloring is valid by checking each edge, so the computation takes $|E|$ steps, which is polynomial in the size of the graph. This is a computationally feasible problem. Similarily, it is not obvious that we can check whether a given integer has exactly 100 prime factors easily (Remark: it is actually an open problem whether this is easy or not, to all you aspiring number theorists, with a state of the art solution which runs in less than

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

steps, where n is the number of bits required to specify the number to factorize. However, if you give me the prime factorization of the integer, then I can make sure you've given me exactly 100 prime factors, and that these factors multiply to equal the integer. Multiplication of two n bit numbers takes $O(n^2)$ steps, so, if our initial number is expressible in n bits, then we can first check each prime factor has less than n bits, in $O(n)$ steps, multiply all these integers in $100\ O(n^2) = O(n^2)$ steps, and then check whether the product we obtain is equal to the original number, so we can check a given factorization in $O(n^2)$ time. These are the family of problems that are 'feasibly checkable'.

The PCP theorem doesn't say that, in 3 questions, we can obtain a proof of the theorem, but instead that we can verify whether the prover *has* a proof of the theorem. For instance, the PCP theorem says that we can determine whether an oracle has a decomposition of an integer into 100 prime factors by asking it 3 questions, but we might not actually know what this decomposition is. Note that asking the oracle "Do you have a decomposition of the integer into 100 prime factors" is *not* a useful question, because the oracle might have made a mistake, or lie, and there is no way to check this question is true.

Example. *Here is a practical example of the PCP theorem, which has been proposed for scaling the utility of certain digital currencies, like Bitcoin. One problem with digital, non centralized currencies is that there is no central authority deciding when fraud is taking place. The big realization of decentralized currency is that if we have a whole bunch of people volunteering to maintain the ledger of all transactions (in exchange for earning some bitcoins, this is mining), no person can individually break the ledger, because their ledger would disagree with everyone elses ledger. A succinct non-interactive argument of knowledge, or SNARK*

An interesting way to view the problem is that certain 'local' properties of mathematical objects can be encoded in such a way that these local properties become global. Here is an analogy from Dinur's talk on the subject. Imagine if we have a piece of toast, and, by sampling three points at random on the piece of toast, we want to determine whether the toast has jam on it. The encoding process as above can be compared to taking a knife, and smothering it all over the piece of toast, not looking at the bread. If there is any jam on the bread, then the knife will smother the jam all over the piece of toast, and after this, if we have any local jam on the piece of toast, it will now become a global property of the bread, and this can be very easily obtained by sampling three points on the piece of toast.

Example. *Three colorability of graphs is a very local property. In particular, a graph can appear to be three colorable, but fail to be three colorable at a single position. If someone said to us they had a three coloring of a graph, and we could only ask them the color of individual vertices, then it would take $|V|$ questions to determine whether the person actually had a three coloring. If you had access to the entire three coloring, it would only take you $O(|V| + |E|)$ steps to check whether a given coloring was actually a three coloring, so the problem of checking a three coloring is computationally feasible, and the PCP theorem guarantees that we can select three questions at random about the prover's proof of the three colorability of the graph, and to a high degree of accuracy, determine whether the prover's proof given is false, or whether the proof is correct. Note that "Do you have a three coloring", is NOT a good question to ask in this context, because we have no reason to trust the prover, and if he said "Yes" we have no information about his proof. Thus there is a way of encoding the three colorability of a graph in such a way that the local property of failing to be three colorable becomes global.*

Example. *The mathematical proof of a mathematical theorem is a very local property. Proofs can seem correct, but have a single 'bug' at a single location which invalidates the entire proof. The PCP theorem says that there is a new way of encoding these proofs such that bugs are globally expressed, and become easy to find.*

We shall find that Fourier-like techniques are very useful for transforming local properties of mathematical objects to global properties. An example of why this is true is that the L^1 norm of the Fourier transform of a function (which can be viewed as a particular encoding of a function) controls the L^∞ norm of the original function. The L^∞ norm is a very local property of the original function, whereas the L^1 norm of the Fourier transform is a global property, and can be estimated accuracy with a few random samples of the function.

1 Examples of PCPs

Here is some formal information which will be useful to reason about PCPs. Once we have predetermined a series of yes/no questions the verifier will pick from to determine if the theorem is correct, then we can index each question by an integer i, and then an *oracle* can be identified with a finite series of zeroes and ones $x \in \{0,1\}^n$, such that $x_i = 1$ if the oracle responds yes to question i, and $x_i = 0$ if the oracle responds no to question i. If we ask three questions i, j, and k, we observe the bits x_i, x_j, and x_k, and must decide whether we are convinced by these three bits.

Example. *Consider determining whether two graphs G_0 and G_1 with n indexed nodes, are not isomorphic to one another. While it is easy to provide a proof that two graphs are isomorphic (just give me an isomorphism, and I can check that it is injective, bijective, and an isomorphism in linear time in the size of the graphs), it is not easy to show that two graphs are not isomorphic. In fact, we don't actually know whether the non isomorphism problem is feasibly checkable (though a proof that it isn't would imply $\mathbf{P} \neq \mathbf{NP}$). However, we can stll obtain an interactive proof of this result, which only asks a single question. Our questions will be of the following variety. Given a graph H, we ask*

"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 the graphs are isomorphic, then this question isn't even well formed, and thus cannot be answered correctly). To form our question, we fix an index $i \in \{0,1\}$, and a permutation $\nu \in S_n$, chosen uniformly at random. Given the graph G_i, we form the graph $\nu(G_i)$ by permuting the indices of the nodes of G_i. Now we ask the question above for $H = \nu(G_i)$. If the answer is equal to i, we are convinced of the statement, and otherwise, we remain unconvinced. If the graphs are not isomorphic, and the oracle answers correctly, we are convinced. Thus we have perfect completeness. Now suppose the graphs are isomorphic to one another. Then the random graph $\nu(G_i)$ we obtain is independent of the

index i, and so for any answer the oracle gives, there is a 50% chance of being caught out, because for each graph H we could pick to ask about, we expect either answer with a 50% probability. Thus we have imperfect soundness.

The PCP theorem also tells us about how *much* randomness we need to answer our question. We measure this in the number of 'coin flips' required to determine the probability distribution of questions we ask. Note that we require a single coin flip to determine the probability distribution i, but $\lg n!$ coin flips to determine a uniformly random $\nu \in S_n$. Thus we require $\lg n! + 1 = O(n \lg n)$ random bits. The PCP theorem guarantees that for feasibly checkable proofs, not only do we only need to ask K questions, but we also only need $O(\lg n)$ coin flips of randomness to determine which questions to ask. Thus there are only $O(nK)$ different questions we could ask on any given input. On the other hand, for the interactive proof we gave above, we can ask $O(2n!)$ different questions. There are results which show that any problem which can be checked in *exponential time* has an interactive proof asking K questions, but using a polynomial amount of randomness, but this is a much weaker result for applications, which we will see later.

Example. *Consider the problem of determining whether a series of quadratic equations over the field $\mathbf{F}_2$ is solvable, i.e. if*

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

are solvable. This problem is **NP** *hard, in the sense that, if you give me a solution to the equations, I can check this fast, but if I was able to solve this problem in polynomial time, then* $\mathbf{P} = \mathbf{NP}$ *(a result we expect not to be true, so the problem likely does not have a polynomial time solution). Since $x_i^2 = x_i$, we may assume that all monomials in the equations have two terms. In this case, we can view the quadratic forms as obtained from a bilinear form on $\mathbf{F}_2^n$, and so we can find an $m \times n^2$ matrix A, and an m dimensional vector b, both with entries in $\mathbf{F}_2$, such that, if for $x \in \mathbf{F}_2^m$, we set $(x \otimes x)_{ij} = x_ix_j$, then we are solving $Ax = b$, where*

$$Ax = \sum A_{kij}(x \otimes x)_{ij}e_k = \sum A_{kij}x_ix_je_k$$

We will provide an interactive proof of this result. In this case, the equations can fail locally (all but one equation can pass), so this is the first time we need to encode this problem in a nontrivial way that enables us to verify a result. We rely on Fourier analysis results. We will ask questions of the following variety:

- *What is the value of x_α.*

Since this problem is **NP** *hard, all problems which are feasible checkable a*

2 Basics of Complexity Theory

The field of complexity theory studies how computationally difficult certain classes of problems are. It seems intuitive that certain problems, like sorting

a list of numbers, or inverting a matrix, should be more difficult than more complicated problems, like determing whether two groups are isomorphic to one another, or finding . Complexity theory classifies these kinds of problems by how long it takes to compute the answer, or how large our whiteboard needs to be to compute the answer, etc. Here we address the very basics. Everything we discuss here can be defined rigorously. However, this would be way too time consuming in this talk, and since intuitions naturally correspond with the true facts, we will not discuss the technical details here. We introduce the ideas for decision problems

Let $f : X \to X$ be a function, for which we have an *algorithm* which computes the function. If $x \in X$, then we let $T(x)$ denote the number of steps it takes for the algorithm to compute $f(x)$. We say f is **polynomial time computable** if $T(x) = O(|x|^n)$, where $|x|$ is the size of the input to x, for some n.

3 VERY RECENT WORK

The Quantum PCP Theorem.

References