FOURIER ANALYSIS OF BOOLEAN FUNCTIONS

SAM SPIRO

Abstract. This paper introduces the technique of Fourier analysis applied to Boolean functions. We use this technique to illustrate proofs of both Arrow's Impossibility Theorem and Håstad's 3-bit Theorem.

Contents

1. Introduction

Classical Fourier analysis is a powerful tool in studying and understanding functions of a continuous variable. Similarly, the discrete analogs of these techniques prove useful for the study of Boolean functions.

There are many equivalent ways to define Boolean functions, and the definition one choses often reflects the kind of problems one is trying to solve with them. Two of the most common definitions are as functions of the form $f : \{0, 1\}^n \to \{0, 1\}$ or as functions of the form $f : \{-1, 1\}^n \to \{-1, 1\}$. For our purposes, we shall use the latter as the definition of Boolean functions. Note that one can move between these two viewpoints by using the bijection $\phi(b) = (-1)^b$ for $b \in \{0, 1\}$, which has the nice property of transforming addition into multiplication, that is, $\phi(x+y) = \phi(x)\phi(y)$, where the addition takes place $\mod 2$.

Initially, such functions may seem fairly simple to analyze. After all, their domains and ranges are rather narrow. However, one finds that the category of Boolean functions is quite rich. For instance, one might think of a Boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$ as an election of two candidates labeled -1 and 1, where for $x \in \{-1, 1\}^n$, x_i represents the ith voter's preference and $f(x)$ represents the election's outcome. In another setting, an $x \in \{-1, 1\}^{\binom{n}{2}}$ could represent the edges of some n vertex graph, where $x_i = 1$ indicates that the graph has an edge between some vertices j, k. We could then define the function $g : \{-1, 1\}^{\binom{n}{2}} \to \{-1, 1\}$ by $g(x) = 1$ if the corresponding graph of x is connected, and $g(x) = -1$ otherwise. Such a function is easy to state, but it may be difficult to analyze, and hence more sophisticated tools must be developed to analyze its behavior.

Briefly, we describe some of the notation used throughout the paper. For $x, y \in \{-1,1\}^n$, the string $xy \in \{-1,1\}^n$ is defined coordinate-wise as $(xy)_i = x_i y_i$. The term "bit" may refer either to a value in $\{0,1\}$ or in $\{-1,1\}$, depending on the context. We write $[n]$ to denote the set $\{1,\dots,n\}$. We will write $x \sim S$, where S is a set (most often $\{-1,1\}^n$) to indicate that x is drawn uniformly at random from the set S. When the probability distribution of a random variable x is understood, we will write E_x or even E to denote its expectation under this measure.

The outline for the paper is as follows. Section 2 defines the Fourier expansion of a Boolean function and then explores fundamental definitions and tools in their analysis. We then use these tools to prove theorems from two very distinct areas of study: Arrow's Impossibility Theorem from social choice theory, and Håstad's 3-bit Theorem from the theory of probabilistically checkable proofs. In Section 3 we prove Arrow's Impossibility Theorem after introducing some additional Fourier analytic tools. In Section 4 we introduce the terminology necessary to understanding Håstad's Theorem, such as PCPs and CSPs. We then formally state and prove Håstad's Theorem in Section 5.

2. The Fourier Expansion

Before introducing the Fourier expansion of a Boolean function, we introduce a standard way of expressing arbitrary Boolean functions by indicator functions. For any fixed $y \in \{-1,1\}^n$, one can define the corresponding indicator function $\mathbb{1}_y : \{-1,1\}^n \to \{0,1\}$ by

$$\mathbb{1}_y(x) = \prod_{i=1}^{n} \frac{1+y_i x_i}{2} = \begin{cases} 1 & x = y \\ 0 & \text{otherwise} \end{cases}.$$

From this definition, it is clear that any Boolean function f can be written as

$$f(x) = \sum_{y \in \{-1,1\}^n} f(y)\mathbb{1}_y(x).$$

Example 2.1. Consider the indicator function $\mathbb{1}_{\{(1,-1)\}}$. By expanding its product, we can write this as

$$\mathbb{1}_{\{(1,-1)\}}(x) = \left(\frac{1+x_1}{2}\right)\left(\frac{1-x_2}{2}\right) = \frac{1}{4} + \frac{1}{4}x_1 - \frac{1}{4}x_2 - \frac{1}{4}x_1x_2$$

and one can check that this evaluates to 1 if $x = (1,-1)$ and to 0 otherwise.

In Example 2.1, one observes that the indicator function (upon expanding its product) is in fact a *multilinear polynomial*, and this is true of all indicator functions. As all Boolean functions are the linear combination of indicator functions, we conclude the following.

Remark 2.2. All Boolean functions can be written as multilinear polynomials.

Delving into this idea, we note that any monomial term of a multilinear polynomial will be of the form $c\prod_{i\in S} x_i$ for some $S \subset [n]$, $c \in \mathbb{R}$. Motivated by this, we define the Boolean function χ_S for every $S \subset [n]$ by $\chi_S(x) = \prod_{i\in S} x_i$ (where we define the empty product to be equal to 1, so that $\chi_\emptyset \equiv 1$). We call these functions *characters*.

Example 2.3. We can rewrite the multilinear polynomial $\frac{1}{4} + \frac{1}{4}x_1 - \frac{1}{4}x_2 - \frac{1}{4}x_1x_2$ which represents $\mathbb{1}_{(1,-1)}$ in terms of characters:

$$\frac{1}{4}\chi_\emptyset(x) + \frac{1}{4}\chi_{\{1\}}(x) - \frac{1}{4}\chi_{\{2\}}(x) - \frac{1}{4}\chi_{\{1,2\}}(x).$$

Using similar logic, it is clear that we can rewrite any multilinear polynomial as a linear combination of characters. By Remark 2.2, we conclude that any Boolean function f can be written as the linear combination of characters. Moreover, these functions will turn out to be an orthonormal basis of a certain vector space.

We shall temporarily expand our study to consider all functions of the form $f : \{-1,1\}^n \to \mathbb{R}$, which we shall call real-valued Boolean functions. It should be noted that using the same logic as before, we can derive that such functions can be written as the linear combination of indicator functions, and hence as the linear combination of characters.

Real-valued Boolean functions naturally form a vector space over $\mathbb{R}$ by defining $(f+g)(x) = f(x) + g(x)$ and $(cf)(x) = cf(x)$ for $c \in \mathbb{R}$. It is also clear that the indicator functions (now thought of as $\mathbb{1}_y : \{-1,1\}^n \to \mathbb{R}$) form a basis for this vector space, and hence the dimension of this space is 2^n. Because we know that the set $\{\chi_S\}$ spans the space of real-valued Boolean functions, and because its (finite) cardinality equals the dimension of the vector space, we conclude that the characters forms a basis for the space (and hence we conclude that the multilinear expansion of f is *unique*).

Now we introduce an inner-product. The simplest inner-product for real-valued Boolean functions f, g would be $\sum_{x\in\{-1,1\}^n} f(x)g(x)$. The problem is that for Boolean functions f, g (which is ultimately what we wish to return to), this sum can never equal 1, so there is no hope in creating an orthonormal basis of Boolean functions from this inner-product. To remedy this, we scale our sum by a factor of 2^{-n}, in essence turning our sum into an average. We thus define the inner-product for the space as follows:

$$\langle f, g\rangle = \sum_{x\in\{-1,1\}^n} 2^{-n} f(x)g(x) = \mathrm{E}[f(x)g(x)].$$

We now verify that under this inner product, the set of χ_S do in fact form an orthonormal basis. We first note that $\chi_S\chi_T = \chi_{S\Delta T}$, where $S\Delta T$ is the symmetric difference of S and T. Indeed,

$$\chi_S(x)\chi_T(x) = \prod_{i\in S} x_i \prod_{j\in T} x_j = \prod_{i\in S\Delta T} x_i \prod_{j\in S\cap T} x_j^2 = \prod_{i\in S\Delta T} x_i = \chi_{S\Delta T}(x) \tag{2.4}$$

where we used the fact that $x_k^2 = 1$ for any $x_k \in \{-1,1\}$. We will also prove that

$$\mathrm{E}[\chi_S(x)] = \begin{cases} 1 & S = \emptyset \\ 0 & S \neq \emptyset \end{cases}.$$

Because $\chi_\emptyset \equiv 1$, the first case is clear. If $S \neq \emptyset$ we have

$$\mathrm{E}[\chi_S(x)] = \mathrm{E}\left[\prod_{i\in S} x_i\right] = \prod_{i\in S} E\left[x_i\right]$$

where we use the fact that choosing an $x \sim \{-1,1\}^n$ is equivalent to choosing each coordinate $x_i \sim \{-1,1\}$ independently of one another. We then have $E[x_i] = 0$ for all i, so this (non-empty) product will be 0.

As $\langle \chi_S, \chi_T \rangle = \mathrm{E}[\chi_S(x)\chi_T(x)] = \mathrm{E}[\chi_{S\Delta T}(x)]$, we conclude that this evaluates to 1 if $S = T$ and to 0 otherwise. Hence, under this inner-product, the characters form an orthonormal basis. We can now write the *Fourier expansion of f*:

$$f = \sum_{S\subset[n]} \hat{f}(S)\chi_S \tag{2.5}$$

where $\hat{f}(S) = \langle f, \chi_S \rangle$ is the *Fourier coefficient of f at S.*

Example 2.6. Define $NAE : \{-1,1\}^3 \to \mathbb{R}$ (the "not all equal" function) as $NAE(1,1,1) = NAE(-1,-1,-1) = 0$, with the function equaling 1 on all other inputs. One can verify (either by expanding the indicator functions, computing the inner product of NAE with χ_S for each $S \subset [3]$, or simply be verifying the correct output for each input value) that its Fourier expansion is

$$NAE(x_1,x_2,x_3) = \frac{3}{4} - \frac{1}{4}x_1x_2 - \frac{1}{4}x_1x_3 - \frac{1}{4}x_2x_3.$$

Writing NAE out as its expansion will be a critical step in our proof of Arrow's Theorem.

Example 2.7. For any real-valued Boolean function f

$$\mathrm{E}[f(x)] = \mathrm{E}\left[\sum_{S\subset[n]} \hat{f}(S)\chi_S(x)\right] = \sum_{S\subset[n]} \hat{f}(S)\,\mathrm{E}[\chi_S(x)] = \hat{f}(\emptyset)$$

where the last step comes from the fact that $E[\chi_S(x)] = 1$ if $S = \emptyset$ and equals 0 otherwise. Hence, the coefficient $\hat{f}(\emptyset)$ is the expected value of f.

We close this section with the two most vital formulas in the analysis of Boolean functions.

Theorem 2.8 (Plancherel's Theorem)**.** *For any real-valued Boolean functions f, g, we have*

$$\langle f, g \rangle = \sum_{S\subset[n]} \hat{f}(S)\hat{g}(S). \tag{2.9}$$

Proof. The idea used in this proof is to replace each function with its respective Fourier expansion, an idea which we shall see throughout the paper.

$$\begin{aligned}
\langle f, g\rangle &= E[f(x)g(x)] \\
&= E\left[\left(\sum_{S\subset[n]} \hat{f}(S)\chi_S(x)\right)\left(\sum_{T\subset[n]} \hat{g}(T)\chi_T(x)\right)\right] \\
&= E\left[\sum_{S,T\subset[n]} \hat{f}(S)\hat{g}(T)\chi_{S\Delta T}(x)\right] \\
&= \sum_{S,T\subset[n]} \hat{f}(S)\hat{g}(T)E[\chi_{S\Delta T}(x)] \\
&= \sum_{S\subset[n]} \hat{f}(S)\hat{g}(S)
\end{aligned}$$

where the last step comes from the fact that $E[\chi_{S\Delta T}(x)] = 0$ if $S \neq T$ and is equal to 1 if $S = T$. □

Theorem 2.10 (Parseval's Theorem)**.** *For any Boolean function f*

$$\sum_{S\subset[n]} \hat{f}(S)^2 = 1. \tag{2.11}$$

Proof. Note that regardless of the value of $f(x)$, because f is a Boolean function we have $f(x)f(x) = 1$. Hence,

$$1 = E[1] = E[f(x)f(x)] = \langle f, f\rangle = \sum_{S\subset[n]} \hat{f}(S)^2$$

with the last step coming from Plancherel's Theorem. □

This covers the primary tools we'll need throughout the text. For a more thorough coverage of the theory of Fourier analysis of Boolean functions, see [4].

3. Arrow's Impossibility Theorem

In this section we view Boolean functions as defining a *voting rule* or *social choice function*. That is, functions $f : \{-1, 1\}^n \to \{-1, 1\}$ will be treated as an election between candidates labeled 1 and -1, with the input x representing the preferences of the voters and $f(x)$ representing the outcome of the election.

Example 3.1. For n odd, the function $Maj_n : \{-1, 1\}^n \to \{-1, 1\}$ is defined by $Maj_n(x) = \text{sgn}(x_1 + \cdots + x_n)$, that is, the function equals the value which appears most often in x. This "majority rule" function is the most common way to run a two candidate election.

Example 3.2. For $i \in [n]$, the *ith dictator function* $\chi_i = \chi_{\{i\}}$ represents an election where the preference of voter i completely determines the outcome of the election. Such an election might take place if there's a majority shareholder of a corporation, or if the election involves an actual dictator.

Example 3.3. A voting rule f is said to be *unanimous* if $f(1, 1, \ldots, 1) = 1$ and $f(-1, \ldots, -1) = -1$ (that is, whenever the voters unanimously favor a certain candidate, that candidate wins the election). Two examples of unanimous voting rules are Maj_n (if every voter prefers candidate b, then in particular a majority of voters prefer b) and χ_i (if every voter prefers candidate b, then in particular the dictator prefers b).

A natural question to ask in this setting is how "stable" a given voting system is. That is, if the preferences of the voters is x and the election outcome is $f(x)$, will a few voters changing their preferences change the outcome of the election?

For example, the dictator functions χ_i are highly stable. Provided the dictator himself doesn't change his vote, any number of voters can change their preferences while leaving the outcome of the election unchanged. Conversely, the parity function $\chi_{[n]}$ is highly unstable. On any input x, any single voter changing his vote is guaranteed to change the result of the election.

We now introduce some terminology to help formalize this idea.

Definition 3.4. For $\rho \in [-1, 1]$ and a string $x \in \{-1, 1\}^n$, we say that the string $y \in \{-1, 1\}^n$ is *ρ-correlated with* x and write $y \sim N_\rho(x)$ if the string y is constructed by independently, for each $i \in [n]$, setting $y_i = x_i$ with probability $\frac{1}{2} + \frac{1}{2}\rho$ and $y_i = -x_i$ with probability $\frac{1}{2} - \frac{1}{2}\rho$.

If $x \sim \{-1, 1\}^n$ and $y \sim N_\rho(x)$, let π_ρ be the distribution of pairs (x, y), and we denote the expectation taken with respect to this measure by E_ρ. Note that $\mathrm{E}_\rho[x_i] = \mathrm{E}_\rho[y_i] = 0$ and $\mathrm{E}_\rho[x_i y_i] = \rho$.

One can interpret y being ρ-correlated with x as y being a "noisy" version of x. That is, each voter sends their vote x_i over some noisy channel where there is a chance their vote will get flipped, and the bit received at the end is y_i. If $\rho = 0$, this noise completely masks whatever the voter's preferences were. If $\rho < 0$, the noise actively works *against* the voter's original preferences.

With this terminology introduced, we now formally define what we mean for a voting system to be stable.

Definition 3.5. The *noise stability of f at ρ*, written $\mathrm{Stab}_\rho[f]$, is defined to be

$$\mathrm{Stab}_\rho[f] = \mathop{\mathrm{E}}_{\rho}[f(x)f(y)].$$

Note that in some sense, the noise stability of f at ρ represents how correlated $f(x)$ is with f evaluated on noisy input $y \sim N_\rho(x)$. For instance, if $f(x)$ always agrees with $f(y)$ (if $\rho = 1$ or if f is a constant function), then $\mathrm{Stab}_\rho[f] = 1$. If $f(x)$ always disagrees with $f(y)$ (say $f = \chi_i$ and $\rho = -1$), then $\mathrm{Stab}_\rho[f] = -1$.

Before we analyze how $\mathrm{Stab}_\rho[f]$ relates to its Fourier expansion, we introduce one more definition.

Definition 3.6. For $0 \le k \le n$ we define the *Fourier weight of f at degree k* to be $W^k[f] = \sum_{|S|=k} \hat{f}(S)^2$.

One can interpret $W^k[f]$ as how much the degree k terms in f's Fourier expansion contribute to determining the value of f. Note that by Parseval's Theorem we have $\sum_{k=0}^n W^k[f] = 1$.

Proposition 3.7.

$$\mathrm{Stab}_\rho[f] = \sum_{k=0}^{n} \rho^k W^k[f].$$

Proof. We wish to use the Fourier expansion of f, and to do so we'll need to understand how characters interact with one another under noisy input. To this end, we note that

$$\mathop{\mathrm{E}}_{\rho}[\chi_S(x)\chi_T(y)] = \mathop{\mathrm{E}}_{\rho}\left[\prod_{i\in S\setminus T} x_i \prod_{j\in T\setminus S} y_j \prod_{k\in S\cap T} x_k y_k\right]. \tag{3.8}$$

By the way x and y are constructed, each term of each product is independent of one another. Thus, the quantity in (3.8) equals

$$\prod_{i\in S\setminus T} \mathop{\mathrm{E}}_{\rho}[x_i] \prod_{j\in T\setminus S} \mathop{\mathrm{E}}_{\rho}[y_j] \prod_{k\in S\cap T} \mathop{\mathrm{E}}_{\rho}[x_k y_k].$$

If $S \neq T$ then the expectation is 0 (by the discussion following Definition 3.4). If $S = T$ then the expectation reduces to

$$\prod_{k\in S} \mathop{\mathrm{E}}_{\rho}[x_k y_k] = \prod_{k\in S} \rho = \rho^{|S|}.$$

Thus, by writing out $f(x)$ and $f(y)$ by their Fourier expansions in the definition of $\mathrm{Stab}_\rho[f]$ we get

$$\begin{aligned}
\mathrm{Stab}_\rho[f] &= \mathop{\mathrm{E}}_{\rho}\left[\left(\sum_{S\subset[n]} \hat{f}(S)\chi_S(x)\right)\left(\sum_{T\subset[n]} \hat{f}(T)\chi_T(y)\right)\right] \\
&= \sum_{S,T\subset[n]} \hat{f}(S)\hat{f}(T) \mathop{\mathrm{E}}_{\rho}[\chi_S(x)\chi_T(y)] \\
&= \sum_{S\subset[n]} \hat{f}(S)^2\rho^{|S|} = \sum_{k=0}^{n} \rho^k W^k[f].
\end{aligned}$$

□

In essence, this theorem says that functions with most of their Fourier weight on low-degree terms (such as constant or dictator functions) will have very high noise stability, while those with most of their weight on high-degree terms (such as $\chi_{[n]}$) will be highly unstable. Thus, the definition of noise stability not only agrees with our intuition on what functions are and are not stable, but it also gives us a very direct way of relating a function's Fourier expansion with its stability.

We now analyze elections with three candidates. The three candidates are labeled a, b, and c, and the election is run on a population of n voters, each having an associated total ordering of the candidates based on their preferences. As a concrete example, say that 35% of the population prefers a to b and b to c, or more concisely, that their preferences are $a > b > c$. Let's also suppose that 25% have preference $b > a > c$ and that 40% have $c > a > b$.

In the most familiar kinds of elections, where each voter chooses their top choice and the candidate with the most votes wins, candidate c is declared the winner. The problem is that such a procedure loses a lot of information about the voter's preferences. One sees that 60% of the population has c as their least preferred candidate, but this information is completely lost by the voting procedure, causing a strongly disliked candidate to win the election.

In an attempt to capture more of the information encoded by the voters' preferences, we run a *Condorcet election*. We run three elections, one between each pair of candidates and each election using the same voting rule f. To do so, each voter

i sends in three bits (x_i, y_i, z_i) representing their preferences towards a vs b, b vs c, and c vs a respectively, with a value of 1 indicating that i favors the first candidate in the pair and a value of -1 indicating that he favors the second. For example, if voter i belonged to the 35% of voters with preference $a > b > c$, i would send in $x_i = 1$ (as he prefers a to b), $y_i = 1$ (as he prefers b to c) and $z_i = -1$ (as he does *not* prefer c to a), in total sending the string $(1, 1, -1)$. Those 25% with preference $b > a > c$ would send the string $(-1, 1, -1)$ and the 40% with preference $c > a > b$ send the string $(1, -1, 1)$. Note that no voter can send in the "irrational" strings $(1, 1, 1)$ or $(-1, -1, -1)$, as these do not correspond to a total ordering of the candidates. All other strings are allowed, and such strings are said to correspond with "rational preferences."

After being given all of the voters' bit strings, we concatenate the strings (x_i, y_i, z_i) into strings $x, y, z \in \{-1, 1\}^n$ and apply our voting rule f to each of these, giving us the bit string $(f(x), f(y), f(z))$ which we think of as "society's preference string." If this string is not $(1, 1, 1)$ or $(-1, -1, -1)$, then there is a candidate who wins each of his pairwise elections, and we declare that candidate the *Condorcet winner* of the election. If such a candidate exists, it is natural to declare him to be the winner of the entire election.

In our example and with f the majority function, we have $(f(x), f(y), f(z)) = (1, 1, -1)$, which corresponds to a winning all of his pairwise elections. In this scenario, a is the Condorcet winner, but such a winner doesn't always exist. If the voters preferences had instead been 34% $a > b > c$, 33% $b > c > a$ and 32% $c > a > b$, then there exists no Condorcet winner under f the majority function. Is there any function f for which there is always a Condorcet winner in a 3 candidate election? The answer turns out to be "yes," but not a particularly reassuring one.

Theorem 3.9 (Arrow's Impossibility Theorem). *Suppose $f : \{-1, 1\}^n \to \{-1, 1\}$ is a voting rule used in a 3-candidate Condorcet election such that there is always a Condorcet winner and such that f is unanimous. Then f is a dictator function.*

Indeed, if f is a dictator function χ_i then $(f(x), f(y), f(z)) = (x_i, y_i, z_i)$, which isn't equal to $(1, 1, 1)$ or $(-1, -1, -1)$ by assumption of each voter having rational preferences, so there is always a Condorcet winner, and we showed in 3.3 that the dictator functions are unanimous.

To prove that these are the only functions that satisfy the hypothesis of the theorem, we use two lemmas derived using Fourier analysis. Recall the function NAE which evaluates to 0 if all of its inputs are the same and outputs 1 otherwise, and has Fourier expansion $\frac{3}{4} - \frac{1}{4}x_1x_2 - \frac{1}{4}x_1x_3 - \frac{1}{4}x_2x_3$. One can think of NAE as the indicator function for rational voting preferences.

Lemma 3.10. *For a 3-candidate Condorcet election with f as the voting rule, the probability of a Condorcet winner when voter preferences are chosen uniformly at random from all rational preferences is precisely $\frac{3}{4} - \frac{3}{4}\mathrm{Stab}_{-1/3}[f]$.*

Proof. Let x, y, and z be the collective votes for the a vs b, b vs c and c vs a elections respectively, with the triples (x_i, y_i, z_i) drawn independently and uniformly from the six triples that satisfy NAE. Note that there is a Condorcet winner iff $NAE(f(x), f(y), f(z)) = 1$, and hence

$$P[\text{Condorcet winner }] = \mathop{\mathrm{E}}_{r}[NAE(f(x), f(y), f(z))]$$

where E_r denotes the expectation taken over uniformly drawn rational preferences. From the Fourier expansion of NAE and linearity of expectations we have

$$P[\text{Condorcet winner }] = \frac{3}{4} - \frac{1}{4}\mathop{\mathrm{E}}_r[f(x)f(y)] - \frac{1}{4}\mathop{\mathrm{E}}_r[f(x)f(z)] - \frac{1}{4}\mathop{\mathrm{E}}_r[f(y)f(z)].$$

Restricting our attention to the bit pairs (x_i, y_i), one can check that $\mathrm{E}_r[x_i] = \mathrm{E}_r[y_i] = 0$ and that $\mathrm{E}_r[x_i y_i] = \frac{2}{6}(1) + \frac{4}{6}(-1) = -\frac{1}{3}$. That is, by restricting the set of values that (x_i, y_i, z_i) can take, once x is given, the string y behaves precisely like $y \sim N_{-1/3}(x)$ under this probability measure. Hence,

$$\mathop{\mathrm{E}}_r[f(x)f(y)] = \mathrm{Stab}_{-1/3}[f].$$

By a symmetric argument, the same result holds for $\mathrm{E}_r[f(x)f(z)]$ and $\mathrm{E}_r[f(y)f(z)]$, and plugging these results into our previous equation gives

$$P[\text{Condorcet winner }] = \frac{3}{4} - \frac{3}{4}\mathrm{Stab}_{-1/3}[f].$$

□

In the next lemma, we write $x^{\oplus i}$ for $x = (x_1, \dots, x_n)$ to denote the string $(x_1, \dots, x_{i-1}, -x_i, x_{i+1}, \dots, x_n)$, that is, $x^{\oplus i}$ is x but with its ith coordinate value changed.

Lemma 3.11. *For a Boolean function f, if $W^1[f] = 1$, then $f = \pm\chi_i$ for some $i \in [n]$.*

Proof. By assumption, the only non-zero Fourier coefficients of f are on sets of cardinality 1, so $f = \sum \hat{f}(\{i\})x_i$ for some set of values $\hat{f}(\{i\})$. For any input x and $i \in [n]$, note that $f(x) - f(x^{\oplus i}) = 2\hat{f}(\{i\})x_i$. But f is Boolean-valued, so $f(x)$ and $f(x^{\oplus i})$ can take values only in $\{-1, 1\}$. Thus their difference can be only -2, 0, or 2, implying that $\hat{f}(\{i\}) = -1$, 0 or 1 for each i. Because $\sum \hat{f}(\{i\})^2 = 1$ by assumption of $W^1[f] = 1$, we must have precisely one $\hat{f}(\{i\}) = \pm 1$ and the rest of the values equal to 0, which means $f = x_i$ or $f = -x_i$ for some i. In other words, f is either a dictator or a negated dictator. □

With these two lemmas we can now prove Arrow's Theorem

Proof. Let f be a voting rule such that the probability of a Condorcet winner is 1. By Lemma 3.10 we have

$$1 = \frac{3}{4} - \frac{3}{4}\mathrm{Stab}_{-1/3}[f].$$

By performing some algebra and then using Proposition 3.7 we see that

$$-\frac{1}{3} = \mathrm{Stab}_{-1/3}[f] = \sum_{k=0}^{n} \left(-\frac{1}{3}\right)^k W^k[f].$$

Because $(-\frac{1}{3})^k > -\frac{1}{3}$ for $k \neq 1$ we have that

$$-\frac{1}{3} = \sum_{k=0}^{n} \left(-\frac{1}{3}\right)^k W^k[f] \geq \sum_{k=0}^{n} -\frac{1}{3} W^k[f] = -\frac{1}{3}$$

with the last step coming from Parseval's Theorem. We thus have that the middle two expressions are equal, and this can only happen if $W^k[f] = 0$ for $k \neq 1$ (because of the strict inequality $(-\frac{1}{3})^k > -\frac{1}{3}$ for $k \neq 1$). We conclude that $W^1[f] = 1$, and from Lemma 3.11 we conclude that f is either a dictator or a negated dictator. Because Arrow's Theorem assumes f is unanimous, f must be a dictator. $\square$

4. Probabilistically Checkable Proofs

We now move on to our second major theorem: Håstad's 3-bit Theorem. In this section we briefly cover the various terminology that will be needed to understand the theorem in full. For a more thorough coverage of this material, see [1].

Definition 4.1. A language L is said to be in NP if there exists an algorithm V called the *verifier* that satisfies the following properties.

- V takes in two strings, x (the *input*) and π (the *proof*) with $|\pi|$ polynomial in $|x|$, and outputs either 0 (for rejection) or 1 (for acceptance).
- V runs in time polynomial in $|x|$. Such algorithms are said to be *efficient.*
- If $x \in L$, there exists a string π such that $V(x, \pi) = 1$. Such a π is called a *correct proof.*
- If $x \notin L$, then $V(x, \pi) = 0$ for all π.

Remarkably, it is possible for a verifier to read only a constant number of (random) bits from the proof string π while having an arbitrarily high probability of accepting correct proofs and of rejecting incorrect ones. To formalize this idea, we introduce the notions of *probabilistically checkable proofs*, or *PCPs*, which can be defined by generalizing the verifier used in Definition 4.1.

Definition 4.2. Given functions $r, q : \mathbb{N} \to \mathbb{N}$, an algorithm V is said to be an $(r(n), q(n))$-*PCP verifier* for a language L if V satisfies the following properties.

- As input, V takes in two strings x and π and outputs either 0 or 1.
- (Efficiency) V is given access to $r(|x|)$ random bits called the *random string.* Using the random string and x, V chooses at most $q(|x|)$ coordinates of π to "query." V then runs in polynomial time and has access *only* to the bits of x and the queried bits of π.
- (Correctness) If $x \in L$, there exists a string π such that $V(x, \pi) = 1$ with probability 1.
- (Soundness) If $x \notin L$, then for *all* strings π, $V(x, \pi) = 1$ with probability at most $\frac{1}{2}$.

A language L is said to be in PCP$(r(n), q(n))$ if L has an $(O(r(n)), O(q(n)))$-PCP verifier.

We now state the PCP theorem, which says languages that can be verified by a proof in polynomial time can also be probabilistically verified using only a constant number of bits.

Theorem 4.3 (PCP). *$NP = PCP(\log n, 1)$.*

This is a very deep and remarkable theorem of computer science. For a complete proof, see [1].

There is another concept that is intimately related to PCPs, that of *constraint satisfaction problems*, or *CSPs.*

Definition 4.4. We say that ϕ is an *instance* of a constraint satisfaction problem of arity q over the alphabet W (written succinctly as ϕ is a $q\text{CSP}_W$ instance) if ϕ consists of the following sets:

- A set of n *variables* $(u_1, \ldots, u_n)$, each of which can take any value in W.
- A set of m *constraints* which are functions ϕ_i that take in at most q variables and then output either 0 or 1.

An *assignment* of a $q\mathrm{CSP}_W$ instance ϕ is an explicit assignment of values from W to the variables of ϕ. The maximum fraction of constraints that can be satisfied by an assignment of ϕ is written $val(\phi)$. If $val(\phi) = 1$ we say that ϕ is *satisfiable*.

The only CSPs we will deal with in this paper are those with alphabets $[W]$ for some $W \in \mathbb{N}$, or CSPs with alphabets $\{-1, 1\}$ or $\{0, 1\}$. In the latter case, we will omit referencing the specific alphabet and simply write qCSP (where it should be clear from context which alphabet we are considering).

Example 4.5. MAX-E3LIN is a subclass of 3CSP, where $\phi \in$ MAX-E3LIN if its constraints are all linear equations of at most three variables and computation is done mod 2. For instance, such an instance ϕ could have variables (x_1, x_2, x_3) and constraints

$$\phi_1(x_1, x_2, x_3) = x_1 + x_2 + x_3 \mod 2$$
$$\phi_2(x_1, x_2, x_2) = x_1 + x_2 + x_3 + 1 \mod 2.$$

Every assignment satisfies precisely one of the constraints, so $val(\phi) = \frac{1}{2}$.

Given a CSP instance ϕ, does there exist an efficient algorithm that produces an assignment of ϕ satisfying $val(\phi)$ fraction of its constraints? In general (assuming $\mathrm{P} \neq \mathrm{NP}$), there is not. However, it is not obvious whether it is NP-hard to find an assignment that satisfies *approximately* $val(\phi)$ fraction of ϕ's constraints.

Definition 4.6. We say that a polynomial time algorithm A is a ρ-*approximation* for a set of CSP instances S if, given a $\phi \in S$, the algorithm outputs an assignment of variables for ϕ that satisfies at least $\rho \cdot val(\phi)$ fraction of its constraints.

Example 4.7. We produce a $\frac{1}{2}$-approximation algorithm for MAX-E3LIN.

Start with every variable unassigned. We will call a constraint "complete" when all of its variables have been assigned a value. At each step, assign x_i the value that maximizes the number of satisfied complete constraints. At every step, at least half of the completed constraints are satisfied. Once every variable is assigned, every constraint is complete, and thus at least half the constraints are satisfied by this assignment. As $\frac{1}{2} \geq \frac{1}{2} val(\phi)$ for any $\phi \in$MAX-E3LIN, we conclude that this is a $\frac{1}{2}$-approximation of MAX-E3LIN.

The algorithm in Example 4.7 is rather simple, so one might think that there exist better approximation algorithms for MAX-E3LIN. Ideally, there would be a $(1-\epsilon)$-approximation for any $\epsilon > 0$. Håstad's Theorem will imply that this is not the case. In fact, it will show that $(\frac{1}{2} + \epsilon)$-approximating MAX-E3LIN is NP-hard for any $\epsilon > 0$, so this simple algorithm is effectively optimal.

PCP verifiers and qCSP instances are intimately related, and in fact, they are equivalent to one another as we show below. This equivalence will be vital for Håstad's Theorem.

Begin with a qCSP instance ϕ. We are given an assignment of variables π (where π_i denotes the value of the ith variable) and we wish to probabilistically check if π is a satisfying assignment of ϕ. To do so, we randomly select a constraint ϕ_r, which will depend on (at most) q variables $i_1, \ldots, i_q$. We then query the string π at these coordinates and check if ϕ_r is satisfied by these values. If the constraint is satisfied, the verifier outputs accept, and otherwise it rejects. If ϕ is satisfiable, there exists a string π such that this process accepts with probability 1. If $val(\phi) = 1 - \epsilon$, then the process accepts with probability at most $1 - \epsilon$, and this can be lowered to $\frac{1}{2}$ by repeating the process a constant number of times. Hence we can produce a PCP verifier for this qCSP instance.

Now start with a PCP verifier V that is given input string x, makes at most $q = q(|x|)$ queries, and uses $r = r(|x|)$ random bits. Let $y \in \{0,1\}^r$ be a random string that could be given to V. Let V_y be the function that takes as input a proof string π such that $V_y(\pi) = 1$ if V accepts π on random input y, and $V_y(\pi) = 0$ if V rejects π on random input y. Each V_y is a function that depends on at most q bits of π, and thus $\{V_y\}$ can be thought of as the constraints of a qCSP instance with variables $\{\pi_i\}$.

5. Håstad's 3-bit theorem

The PCP theorem tells us that the statement "$x \in L$" for L an NP language can be probabilistically verified using only a constant number of bits. Of course, the exact number of bits queried, as well as the soundness of the verifier could very well depend on the language L. Håstad's Theorem tells us that, regardless of the language, the statement $x \in L$ can be probabilistically verified using only 3 bits with soundness arbitrarily close to $\frac{1}{2}$, provided we allow a looser definition of PCP verifiers.

Definition 5.1. A PCP verifier V for a language L is said to have *completeness parameter* ρ if for all $x \in L$, there exists a proof π such that the probability that $V(x, \pi) = 1$ is at least ρ. If $\rho < 1$, we will say that the verifier is *incomplete*, and if $\rho = 1$ we will say that the verifier is *complete.* Note that all of the verifiers in Section 4 were complete.

A PCP verifier V is said to have *soundness parameter* ν if for all $x \notin L$ and all π, the probability that $V(x, \pi) = 1$ is at most ν.

Theorem 5.2 (Håstad)**.** *Let $L \in NP$ and $\delta_0 > 0$. There exists a PCP verifier for L making only 3 queries with completeness parameter $1 - \delta_0$ and soundness parameter $\frac{1}{2} + \delta_0$.*

Before delving into the proof, it is worth discussing the matter of incompleteness. The fundamental reason that Håstad's verifier is incomplete is that the verifier produced is linear. That is, if the verifier V is given a proof π and chooses to query coordinates i, j and k, then V accepts π iff $\pi_i \pi_j \pi_k = b$ for some $b \in \{-1, 1\}$. Recall that multiplication in $\{-1, 1\}$ corresponds to addition in $\{0, 1\}$, and hence this equation is equivalent to a linear equation mod 2. Because of this, the corresponding CSP of this PCP verifier (as defined by the equivalence at the end of Section 4) will have constraints that are linear equations mod 2 of at most three variables. That is, the corresponding CSP is an instance of MAX-E3LIN.

The question of satisfiability for $\phi \in$MAX-E3LIN can be computed in polynomial time. Thus, if for all $L \in$ NP there existed a complete, linear verifier V for L, one could translate any $x \in L$ to a satisfiable instance of MAX-E3LIN and any $x \notin L$ to an unsatisfiable instance. We could then distinguish between these two cases in polynomial time by using Gaussian elimination, and thus we could determine in polynomial time if $x \in L$ for all $L \in \text{NP}$. Thus if $\text{P} \neq \text{NP}$, we will have to accept incompleteness in our linear verifiers.

It is natural then to ask if we can use a non-linear verifier that is complete and that achieves the same soundness results as Håstad's verifier. Assuming $\text{P} \neq \text{NP}$, this is not possible. If we require a complete verifier that makes only 3 queries, its soundness parameter must be at least $\frac{5}{8}$[1], see [6, 7]. Thus for non-adaptive models, Håstad's Theorem is optimal in terms of soundness.

Despite understanding this, one might still argue that incompleteness is too large a price to pay. While the probability of rejecting a correct proof can be made arbitrarily small, it can never be reduced to 0, which might be unacceptable in certain contexts.

Nevertheless, Håstad's verifier opens the door to proving many hardness of approximation results. In fact, the key to proving these results will be the intimate relation between Håstad's verifier and MAX-E3LIN, the same relationship that forced incompleteness into our model.

Proposition 5.3. *It is NP-hard to $(\frac{1}{2}+\nu)$-approximate MAX-E3LIN for any $\nu > 0$.*

Proof. We convert the verifier from Håstad's Theorem into its corresponding CSP, which as noted earlier is an instance of MAX-E3LIN, where either $1 - \delta_0$ fraction of its constraints are satisfiable, or at most $\frac{1}{2} + \delta_0$ are satisfiable for some $\delta_0 > 0$, and distinguishing between these two cases is NP-hard. We thus conclude that in general, it must be NP-hard to ρ-approximate MAX-E3LIN with $\rho = \frac{\frac{1}{2}+\delta_0}{1-\delta_0}$. This is because such an algorithm would allow us to distinguish between the two cases of Håstad's CSP instance ϕ, and doing this is NP-hard. Since δ_0 can be made arbitrarily small, ρ can be made arbitrarily close to $\frac{1}{2}$, and hence we conclude that $(\frac{1}{2} + \nu)$-approximating MAX-E3LIN is NP-hard for all $\nu > 0$. $\square$

Note that in Example 4.7 we produced a $\frac{1}{2}$-approximation for MAX-E3LIN. If $\text{P} \neq \text{NP}$, Proposition 5.3 tells us that this approximation is essentially optimal.

One can go on to prove many other hardness of approximation results from Proposition 5.3 by translating instances of MAX-E3LIN into other types of instances. Proving such results would be quite difficult without the existence of Håstad's Theorem.

We now present the proof of Håstad's Theorem, a key element of which will be the existence of a certain CSP instance.

A $2\text{CSP}_{[W]}$ instance ϕ is said to be *projective* if for each constraint ϕ_r and for each $w \in [W]$, there exists a unique $u \in [W]$ such that $\phi_r(w, u) = 1$. In other words, each ϕ_r implicitly defines a map $h_r : [W] \to [W]$ such that $\phi_r(w, u) = 1$ iff $h_r(w) = u$. When r is understood, we will simply denote the map by h.

The following result is proved by combining the classical PCP theorem of Arora et al. with Raz's parallel repetition theorem [2, 5] (cf. [1, Theorem 22.15])

Lemma 5.4 (Raz)**.** *For every $L \in NP$ and $\epsilon > 0$, there exists a $W \in \mathbb{N}$ and a polynomial-time function f that maps strings x to projective $2CSP_{[W]}$ instances*

[1]If one allows for *adaptive* queries in their model, it is possible to achieve Håstad's result with completeness parameter 1, see [3]

such that

$$x \in L \implies val(f(x)) = 1$$
$$x \notin L \implies val(f(x)) \le \epsilon.$$

That is, deciding if $x \in L$ or $x \notin L$ is equivalent to deciding whether a certain instance ϕ is satisfiable or if no assignment of ϕ satisfies more than ϵ fraction of its constraints.

For any $x \in L$ with $L \in \text{NP}$, Raz's Theorem gives us a certain projective $2\text{CSP}_{[W]}$ instance ϕ. The instance ϕ itself is equivalent to some PCP verifier, so this seems like a promising place to look for Håstad's verifier. However, we require that our verifier query only 3 bits. While the verifier for ϕ makes only 2 queries, it does so over an alphabet of size W. Thus, to acquire the same amount of information using bits, we would require a total of $2\log W$ queries. Thus we can't make use of this verifier as is, but one might hope to to make use of it indirectly.

To this end, we encode each element of $[W]$ as a bit string. The most natural way to do so would be to encode each $w \in [W]$ by the bit string in $\{-1,1\}^{\log W}$ corresponding to w's binary representation. The issue with this is that the values w and $w+1$ could act very differently in terms of constraints, despite looking almost identical in terms of their binary representations. The solution to this problem is to use the long code.

The long code for $[W]$ encodes each $w \in [W]$ by the truth table of the dictator function $\chi_w : \{-1,1\}^W \to \{-1,1\}$, which we recall is defined by $\chi_w(x) = x_w$. Note that the truth table of χ_w is of size 2^W, while writing w by its binary representation requires only $\log W$ bits, so this is a doubly exponential blowup in the size of w's encoding.

We will need one more piece of notation before defining the weak Håstad verifier. Recall that every constraint ϕ_r induces a map $h : [W] \to [W]$. For $y \in \{-1,1\}^W$, we define the string $y_h \in \{-1,1\}^W$ coordinate-wise as $(y_h)_w = y_{h(w)}$.

Example 5.5. If we apply the map χ_w to the string y_h we get

$$\chi_w(y_h) = (y_h)_w = y_{h(w)} = \chi_{h(w)}(y).$$

Thus, y_h gives us a way of relating two dictator functions through the function h.

Definition 5.6. Given a projective $2\text{CSP}_{[W]}$ instance ϕ, its *weak Håstad verifier* V' takes as input a proof string π of length $n2^W$, which is thought of as n functions $\{f_i\}$, each encoding a value for a variable in $\{u_i\}$. The verifier randomly selects a constraint $\phi_r(u_i, u_j)$, which induces a map $h : [W] \to [W]$. The verifier uniformly at random selects $x, y \in \{-1,1\}^W$ and then constructs $z \in \{-1,1\}^W$ as follows: independent of the other coordinates, set $z_i = 1$ with probability $1-\rho$ and set $z_i = -1$ with probability ρ for some $\rho > 0$. The verifier queries the values $f_i(x),\ f_j(y)$, and $f_i(xy_hz)$ and it accepts π iff

$$f_i(x)f_j(y)f_i(xy_hz) = 1. \tag{5.7}$$

The reason this verifier is referred to as "weak" is because there is a major flaw lurking in its design. Indeed, consider the proof π consisting of all 1's. This is certainly a proof we would like to say is incorrect, as it does not consist of the truth table of n dictator functions (half of the bits of a dictator function will always be -1). However, since every bit of π is 1, (5.7) will always be true, and thus soundness can not possibly hold for this verifier.

The fundamental issue with the weak verifier is that it allows any possible function to be represented in the proof π, despite most functions looking nothing like dictator functions. To get around this, we shall change the format of π in order to restrict the types of functions it can represent.

A function $f : \{-1, 1\}^n \to \{-1, 1\}$ is said to be *folded* or *odd* if $f(-x) = -f(x)$ for all $x \in \{-1, 1\}^n$. Note that all dictator functions are folded. When encoding a folded function f, only half of its truth table needs to be recorded. Indeed, one could encode only the the values $f(x)$ for x with $x_1 = 1$, and then if one wished to look up the value of $f(y)$ with $y_1 = -1$, one would simply output the value $-f(-y)$. By only allowing folded functions to be encoded in π, we will eliminate the problem caused by the all 1's proof.

Definition 5.8. The *Håstad verifier* V is defined in the same way as the weak Håstad verifier, except its proof is of length $n2^{W-1}$, which is interpreted as n odd functions $\{f_i\}$.

Lemma 5.9. *If ϕ is satisfiable, its Håstad verifier V accepts a correct proof with probability $1-\rho$. If $val(\phi) \le \epsilon$, V accepts no proof with probability more than $\frac{1}{2}+\delta$, where $\delta = \sqrt{\epsilon/\rho}$.*

Once we prove this lemma, Håstad's Theorem will follow. Indeed, for deciding if $x \in L$, we can use Raz's theorem to transform x into a projective instance ϕ where $val(\phi) = 1$ if $x \in L$ and $val(\phi) \le \delta_0^3$ if $x \notin L$. We then apply Håstad's verifier to this instance with $\rho = \delta_0$ (which will make $\delta = \delta_0$). If $x \in L$, the verifier will accept with probability $1 - \delta_0$, and if $x \notin L$, it will accept with probability at most $\frac{1}{2} + \delta_0$, and this holds for any $\delta_0 > 0$, proving the theorem. We now prove Lemma 5.9 which will complete the proof of Håstad's Theorem.

Proof. The proof of completeness is rather straightforward. Indeed, upon choosing a constraint $\phi_r(u_i, u_j)$, a correct proof π will have $f_i = \chi_w,\ f_j = \chi_u$ where $h(w) = u$. Once we choose $x, y, z \in \{-1, 1\}^W$ as prescribed by the verifier, we get

$$\chi_w(x)\chi_u(y)\chi_w(xy_hz) = x_w y_u x_w y_{h(w)} z_w = x_w^2 y_u^2 z_w = z_w.$$

Thus, the probability that the verifier accepts this proof is the probability that $z_w = 1$, which by definition is $1 - \rho$.

The proof of soundness will be more difficult, and we will need to make heavy use of Fourier analysis. Before we do so, we must first figure out how the Fourier expansion of $f(y_h)$ interacts with the Fourier expansion of $f(y)$. In particular, we need to know $E[\chi_S(y_h)\chi_T(y)]$ for $S, T \subset [W]$. Using similar logic as in (2.4), we conclude that this expectation will be 1 if

$$\prod_{w \in S} (y_h)_w \prod_{u \in T} y_u = 1$$

and the expectation will be 0 otherwise. For $u \in T$, the y_u term will vanish from this product iff y_u appears an odd number of times in $\prod_{w \in S}(y_h)_w = \prod_{w \in S} y_{h(w)}$, i.e. if the set $\{w \in S | h(w) = u\}$ has an odd number of elements. Thus, if we think of being given S first, we require that $T = \{u | h^{-1}(u) \cap S \text{ is odd}\}$ for the expectation to be non-trivial. With this in mind, we define the set $\tilde{S} = \{u | h^{-1}(u) \cap S \text{ is odd}\}$.

We make some simple observations before proving the next lemma.

Remark 5.10. The characters χ_S are linear, meaning $\chi_S(xy) = \chi_S(x)\chi_S(y)$ for any $S \subset [W],\ x, y \in \{-1,1\}^W$. This is clear from the definition of χ_S.

Remark 5.11. If f is a Boolean function and $P[f(x) = 1] = \frac{1}{2} + \delta$, then $E[f(x)] = 2\delta$. This is because

$$E[f(x)] = P[f(x) = 1] - P[f(x) = -1] = \frac{1}{2} + \delta - (\frac{1}{2} - \delta) = 2\delta.$$

Remark 5.12. If f is folded, then $\hat{f}(\emptyset) = 0$. Indeed, by noting that $P[f(x) = 1] = \frac{1}{2}$, we conclude that $E[f(x)] = 0$ by Remark 5.11. By Example 2.7, we conclude that $\hat{f}(\emptyset) = 0$.

Lemma 5.13. *Let $f = f_i$ and $g = f_j$ be folded functions that pass (5.7) with probability at least $\frac{1}{2} + \delta$. Then*

$$2\delta \le \sum_{S \subset [W], S \neq \emptyset} \hat{f}(S)^2 \hat{g}(\tilde{S})(1 - 2\rho)^{|S|}.$$

Proof. We have $2\delta \le E[f(x)g(y)f(xy_h z)]$ by Remark 5.11. Taking the Fourier expansion of each function and using linearity of expectation gives

$$E[f(x)g(y)f(xy_h z)] = \sum_{S,T,R \subset [W]} \hat{f}(S)\hat{g}(T)\hat{f}(R)E[\chi_S(x)\chi_T(y)\chi_R(xy_h z)]. \tag{5.14}$$

By using linearity of characters, as well as the fact that $x,\ y$, and z were all chosen independently of one another, we see that (5.14) equals

$$\sum_{S,T,R \subset [W]} \hat{f}(S)\hat{g}(T)\hat{f}(R)E[\chi_S(x)\chi_R(x)]E[\chi_T(y)\chi_S(y_h)]E[\chi_R(z)].$$

The first expectation will be 0 unless $S = R$, and the second will be 0 unless $T = \tilde{S}$; otherwise they equal 1. We can thus restrict our sum to $R = S$ and $T = \tilde{S}$. Lastly, we note that

$$E[\chi_S(z)] = E\left[\prod_{w \in S} z_w\right] = \prod_{w \in S} E[z_w] = \prod_{w \in S}(1 - 2\rho) = (1 - 2\rho)^{|S|}$$

where we used the facts that each z_w is independent of one another and that $E[z_w] = 1 - 2\rho$. Altogether we have

$$2\delta \le \sum_{S \subset [W]} \hat{f}(S)^2 \hat{g}(\tilde{S})(1 - 2\rho)^{|S|}.$$

We know $\hat{f}(\emptyset) = 0$ by Remark 5.12, so we can drop this term from the sum, giving the desired result. □

To complete the proof of soundness, we will show that if V accepts a proof $\tilde{\pi}$ with probability at least $\frac{1}{2} + \delta$, then there exists an assignment of the variables of ϕ, π, that satisfies at least $\rho\delta^2 > \epsilon$ fraction of ϕ's constraints. As ϕ has either $val(\phi) = 1$ or $val(\phi) \le \epsilon$, we will conclude that ϕ is satisfiable, proving soundness.

The assignment π is chosen probabilistically such that the expected fraction of constraints it satisfies is at least $\rho\delta^2$. It follows that there exists some assignment π' that satisfies at least this expected fraction of constraints.

We construct π by independently choosing a value for each π_i as follows. Recall that the proof $\tilde{\pi}$ induces a function $f_i : \{-1,1\}^W \to \{-1,1\}$ which V treats as encoding the value of u_i. It is thus natural to use this function in assigning the value of π_i, and moreover, to use its Fourier expansion to do so. That is, a $w \in [W]$ should be more likely to be chosen if $w \in S$ for some $\hat{f}_i(S)$ large. To achieve this, we first choose an $S \subset [W]$ with probability based on $\hat{f}_i(S)$, and then we uniformly at random choose an element of S to be the value of π_i. Because $\sum_{S \subset [W]} \hat{f}_i(S)^2 = 1$ by Parseval's Theorem, the most natural probability to associate with picking S is $\hat{f}_i(S)^2$, and this is what we shall do. Note that because f_i is folded, $\hat{f}_i(\emptyset)^2 = 0$ so uniformly choosing $w \in S$ will be well defined as $S \neq \emptyset$ with probability 1.

To reiterate, we choose a set $S \subset [W]$ with probability $\hat{f}_i(S)^2$ and then choose a $w \in S$ with probability $\frac{1}{|S|}$. We then set $\pi_i = w$.

Example 5.15. Let $f_i = \chi_w$. The only set with Fourier weight non-zero is $\{w\}$, so this set is chosen with probability 1. An element of the set is then chosen uniformly at random, in this case w is chosen with probability 1. Thus this distribution always gives $\pi_i = w$, motivating the idea that this distribution for π is based on the value that f_i "encodes" for u_i.

Define $I_{\pi,r}$ to be the indicator variable that is 1 when assignment π satisfies constraint r. We wish to show that

$$\mathop{\mathrm{E}}_{\pi}[\mathop{\mathrm{E}}_{r}[I_{\pi,r}]] \geq \rho\delta^2$$

where $r \sim [m]$. Our problem is phrased here as first picking a proof π and then seeing the expected fraction of constraints which π satisfies. Because π and r are chosen independently, we can instead think of first choosing the constraint r and then seeing the expected fraction of proofs that satisfy it. We can thus rephrase our goal as trying to show that

$$\mathop{\mathrm{E}}_{r}[\mathop{\mathrm{E}}_{\pi}[I_{\pi,r}]] \geq \rho\delta^2. \tag{5.16}$$

Let $\frac{1}{2} + \delta_r$ denote the conditional probability that V accepts the proof $\tilde{\pi}$ conditional on choosing the constraint ϕ_r. By assumption we have that $\mathrm{E}_r[\frac{1}{2} + \delta_r] = P[V \text{ accepts } \tilde{\pi}] \geq \frac{1}{2} + \delta$, and hence $\mathrm{E}_r[\delta_r] \geq \delta$.

We now wish to show that for fixed r

$$P_\pi[I_{\pi,r} = 1] = \mathop{\mathrm{E}}_{\pi}[I_{\pi,r}] \geq \rho\delta_r^2. \tag{5.17}$$

Because $E[X^2] \geq E[X]^2$ for any random variable, this would imply

$$\mathop{\mathrm{E}}_{r}\left[\mathop{\mathrm{E}}_{\pi}[I_{\pi,r}]\right] \geq \mathop{\mathrm{E}}_{r}\left[\rho\delta_r^2\right] \geq \rho \mathop{\mathrm{E}}_{r}[\delta_r]^2 \geq \rho\delta^2$$

which proves (5.16).

We wish to give a lower bound for $P_\pi[I_{\pi,r} = 1]$. Let $\phi_r(u_i, u_j)$ be the corresponding constraint and let $f = f_i,\ g = f_j$. Because we are given r, we only have to consider the values π_i and π_j of π. Recall that we choose these values by picking two sets $S, T \subset [W]$ with probabilities $\hat{f}(S)^2$ and $\hat{g}(T)^2$ respectively, and then uniformly picking an element from each of these. If we think of choosing S first, then certainly we can achieve a lower bound on the probability of success by only considering successes when $T = \tilde{S}$.

Regardless of what element $u \in T$ is chosen, if $T = \tilde{S}$ then there exists an odd (and hence non-zero) number of $w \in S$ such that $h(w) = u$ by definition of $\tilde{S}$. Hence, conditional on having chosen the sets S and $\tilde{S}$, there is at least a $\frac{1}{|S|}$

probability of $I_{\pi,r}$ equaling 1. Again noting that $\hat{f}(\emptyset)^2 = 0$, we can sum over all S to get

$$\sum_{S\subset[W],S\neq\emptyset} \frac{1}{|S|}\hat{f}(S)^2\hat{g}(\tilde{S})^2 \leq P_\pi[I_{\pi,r}|r]. \tag{5.18}$$

This is almost the result from Lemma 5.13, which says

$$2\delta_r \leq \sum_{S\subset[W],S\neq\emptyset} \hat{f}(S)^2\hat{g}(\tilde{S})(1-2\rho)^{|S|}. \tag{5.19}$$

To bridge this gap, we note the following

Lemma 5.20.

$$(1-2\rho)^{|S|} \leq \frac{2}{\sqrt{\rho|S|}}.$$

Proof. For every $z > 0$ we have

$$\frac{1}{z} \geq e^{-z} \geq 1 - z$$

where the first inequality follows from ze^{-z} achieving its maximum at $z = 1$, and the second follows from $z + e^{-z}$ being an increasing function. These inequalities yield

$$\frac{1}{\sqrt{4\rho|S|}} \geq (e^{-4\rho|S|})^{\frac{1}{2}} = (e^{-2\rho})^{|S|} \geq (1-2\rho)^{|S|}$$

and rearranging these terms gives the desired result. □

From (5.19) and Lemma 5.20 we get

$$2\delta_r \leq \sum_{S\subset[W],S\neq\emptyset} \frac{2}{\sqrt{\rho|S|}}\hat{f}(S)^2\hat{g}(\tilde{S})$$

and rearranging this gives

$$\sqrt{\rho}\delta_r \leq \sum_{S\subset[W],S\neq\emptyset} \frac{1}{\sqrt{|S|}}\hat{f}(S)^2\hat{g}(\tilde{S}). \tag{5.21}$$

Recall the Cauchy-Schwartz inequality:

$$\sum a_i b_i \leq (\sum a_i^2)^{\frac{1}{2}}(\sum b_i^2)^{\frac{1}{2}}.$$

We will apply this inequality to (5.21) with $\hat{f}(S)$ acting as a_i and $\frac{1}{\sqrt{|S|}}\hat{f}(S)\hat{g}(\tilde{S})$ as b_i. This gives

$$\sqrt{\rho}\delta_r \leq \left(\sum_{S\subset[W]} \hat{f}(S)^2\right)^{\frac{1}{2}} \left(\sum_{S\subset[W],S\neq\emptyset} \frac{1}{|S|}\hat{f}(S)^2\hat{g}(\tilde{S})^2\right)^{\frac{1}{2}}$$

where we added $\hat{f}(\emptyset)^2 = 0$ to the first sum. By Parseval's Theorem we have $\sum_{S\subset[W]}\hat{f}(S)^2 = 1$, so we can reduce this to

$$\sqrt{\rho}\delta_r \leq \left(\sum_{S\subset[W],S\neq\emptyset} \frac{1}{|S|}\hat{f}(S)^2\hat{g}(\tilde{S})^2\right)^{\frac{1}{2}}.$$

Finally, we square both sides of the inequality to get

$$\rho\delta_r^2 \leq \sum_{S\subset[W],S\neq\emptyset} \frac{1}{|S|}\hat{f}(S)^2\hat{g}(\tilde{S})^2 \leq P_\pi[I_{\pi,r}|r]$$

where the last inequality follows from (5.18). Hence we have proved (5.17), completing the proof of the lemma and hence of Håstad's Theorem. □

Acknowledgments. It is a pleasure to thank my mentor, John Wilmes, for his constant support and advice, without which this paper could not have been written. I would also like to thank Peter May for organizing, as well as for giving me the opportunity to participate in, the REU program at the University of Chicago.

References

[1] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[2] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

[3] Venkatesan Guruswami, Daniel Lewin, Madhu Sudan, and Luca Trevisan. A tight characterization of NP with 3 query PCPs. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 8–17. IEEE, 1998.

[4] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

[5] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

[6] Luca Trevisan. Approximating satisfiable satisfiability problems. In *European Symposium on Algorithms*, pages 472–485. Springer, 1997.

[7] Uri Zwick. Approximation Algorithms for Constraint Satisfaction Problems Involving at Most Three Variables per Constraint. In *SODA*, volume 98, pages 201–210, 1998.