A Theorist's Toolkit **(CMU 18-859T, Fall 2013)**

Lecture 18: Analysis of Boolean Functions

November 6, 2013

Lecturer: Ryan O'Donnell *Scribe: Sarah Allen*

1 Introduction

Consider a Boolean function $f : \{0,1\}^n \mapsto \{0,1\}$ that maps n bits into a single bit. Traditionally, we consider 0 to represent False, while 1 represents True. In our analysis, it will be helpful to be flexible in our choice of domain and range. For example, we can think of our function as being over the vector space $\mathbb{F}_2^n$. As long as we have two symbols, it doesn't really matter which ones we pick. For most of what we will do, it is convenient to consider

$$f : \{-1,1\}^n \mapsto \{-1,1\}$$

and in some more general cases,

$$f : \{-1,1\}^n \mapsto \mathbb{R}$$

where -1 represents True and 1 represents False.

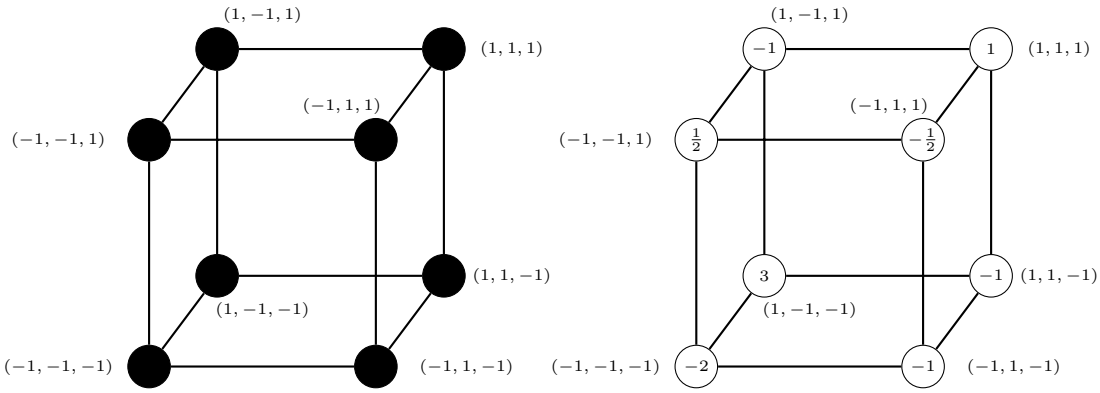
In this lecture, we will consider the analysis of Boolean functions from three different perspectives:

1. Writing functions as polynomials over real numbers
2. Spectral graph theory where our graph is the hypercube
3. Applicability of Fourier analysis to property testing.

2 Fourier Analysis and Functions as Polynomials

For a Boolean function $f : \{-1,1\}^n \mapsto \mathbb{R}^3$, we can think of our domain as a set of 2^n points in n dimensions. These points form the vertices of the hypercube, which is drawn in 3 dimensions in Figure 1a. Since our function maps any assignment of $\{-1,1\}^n$ to $\mathbb{R}$, we can think of function f as assigning real values to each vertex of the hypercube, depicted in Figure 1b.

For the Fourier analysis, we want to find a polynomial that interpolates f. This is also known as Lagrange interpolation, which is easy when our domain is just $\{-1,1\}^n$. We start



(a) The Hamming cube in 3 dimensions (b) f maps each vertex to some element of $\mathbb{R}$

Figure 1: Visualizing the hypercube with $n = 3$

by considering the vertices of the hypercube one by one. For each $x_i = \pm 1$, consider the value $(\frac{1}{2} \pm \frac{x_j}{2})$. When $x_j = x_i$, it evaluates to 1 and when $x_j \neq x_i$, it evaluates to 0. To create a full expression that equals 1 only on a particular assignment and 0 otherwise, we multiply many of these terms together.

For example, let's start with $x = (1, 1, 1)$.

We can create an expression that is equal to the value of $f(1, 1, 1)$ on input $(1, 1, 1)$ and 0 otherwise. Consider the expression

$$\left(\frac{1}{2} + \frac{x_1}{2}\right)\left(\frac{1}{2} + \frac{x_2}{2}\right)\left(\frac{1}{2} + \frac{x_3}{2}\right).$$

If any x_i differs from $x = (1, 1, 1)$, its term will equal zero, causing the entire expression to equal zero. For each x_i that is consistent with $x = (1, 1, 1)$, its term equals 1, so the value of the entire expression is 1 if $x = (1, 1, 1)$ and 0 otherwise. We multiply the entire expression by $f(1, 1, 1)$ to get the proper value, so we have

$$1 \times \left(\frac{1}{2} + \frac{x_1}{2}\right)\left(\frac{1}{2} + \frac{x_2}{2}\right)\left(\frac{1}{2} + \frac{x_3}{2}\right).$$

Now we will create a similar expression for $x = (1, 1, -1)$. Again, we want an expression that is equal to 1 if $x = (1, 1, -1)$ and 0 otherwise. Our expression for this assignment will be

$$\left(\frac{1}{2} + \frac{x_1}{2}\right)\left(\frac{1}{2} + \frac{x_2}{2}\right)\left(\frac{1}{2} - \frac{x_3}{2}\right).$$

It's easy to verify that this expression has the properties we want. Again, we multiply the whole thing by $f(1, 1, -1)$ to get

$$-1 \times \left(\frac{1}{2} + \frac{x_1}{2}\right)\left(\frac{1}{2} + \frac{x_2}{2}\right)\left(\frac{1}{2} - \frac{x_3}{2}\right).$$

We do this for each possible assignment to x and add all of the expressions to get our polynomial. Then we can expand the expressions and combine them to yield a more concise polynomial. Since each x_i appears exactly once in each expression and its degree is at most 1 in each expression, the resulting polynomial is multilinear. We call this the *Fourier expansion* of f. Now, let's consider some common functions and their Fourier expansions.

2.1 Fourier Expansions of Majority and Parity

Our first example will be the majority function on 3 bits, denoted as Maj_3, which is defined to be the most frequently occurring bit in the input. To find the Fourier expansion of this function, we consider the function's value on each input.

Our function will be as follows.

$$\begin{aligned}
Maj_3(x) = 1 \times \left(\frac{1}{2} + \frac{x_1}{2}\right)\left(\frac{1}{2} + \frac{x_2}{2}\right)\left(\frac{1}{2} + \frac{x_3}{2}\right) + \\
+ 1 \times \left(\frac{1}{2} + \frac{x_1}{2}\right)\left(\frac{1}{2} + \frac{x_2}{2}\right)\left(\frac{1}{2} - \frac{x_3}{2}\right) + \\
+ 1 \times \left(\frac{1}{2} + \frac{x_1}{2}\right)\left(\frac{1}{2} - \frac{x_2}{2}\right)\left(\frac{1}{2} + \frac{x_3}{2}\right) + \\
- 1 \times \left(\frac{1}{2} + \frac{x_1}{2}\right)\left(\frac{1}{2} - \frac{x_2}{2}\right)\left(\frac{1}{2} - \frac{x_3}{2}\right) + \\
+ 1 \times \left(\frac{1}{2} - \frac{x_1}{2}\right)\left(\frac{1}{2} + \frac{x_2}{2}\right)\left(\frac{1}{2} + \frac{x_3}{2}\right) + \\
- 1 \times \left(\frac{1}{2} - \frac{x_1}{2}\right)\left(\frac{1}{2} + \frac{x_2}{2}\right)\left(\frac{1}{2} - \frac{x_3}{2}\right) + \\
- 1 \times \left(\frac{1}{2} - \frac{x_1}{2}\right)\left(\frac{1}{2} - \frac{x_2}{2}\right)\left(\frac{1}{2} + \frac{x_3}{2}\right) + \\
- 1 \times \left(\frac{1}{2} - \frac{x_1}{2}\right)\left(\frac{1}{2} - \frac{x_2}{2}\right)\left(\frac{1}{2} - \frac{x_3}{2}\right)
\end{aligned}$$

Expanding this out, we get

$$Maj_3(x) = \frac{1}{2}x_i + \frac{1}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}x_1x_2x_3,$$

which can also be verified by checking all assignments to x.

We also consider the parity function on three bits, denoted as $Parity_3(x_1, x_2, x_3)$. We define $Parity_3(x) = 1$ if x has an even number of (-1)s and -1 if x has an odd number of (-1)s. The fourier expansion of $Parity_3$ can also computed using the above method, which yields the formula

$$Parity_3(x_1, x_2, x_3) = x_1x_2x_3$$

and we can easily verify its correctness.

Remark 2.1. This is slightly different from the more commonly known version of parity, defined as a function mapping $\mathbb{F}_2^n \mapsto \mathbb{F}_2$. In that model, we have that $Parity_3(x_1, x_2, x_3) = x_1 + x_2 + x_3$. While they both achieve the same functionality, we want to distinguish between the two, as our domain and codomain are different in this case.

Using the construction described above, we can generate a polynomial for any Boolean function.

Proposition 2.2. *Every $f : \{-1, 1\}^n \mapsto \{-1, 1\}$ can be uniquely computed as a multilinear polynomial.*

From the above construction, it is clear that we can compute a polynomial for f. The uniqueness remains to be shown.

Each monomial of our polynomial contains some subset of variables (As each variable has either degree 0 or degree 1 in each monomial). Therefore, we can consider $f(x)$ as follows.

$$f(x) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i$$

As a notational convention, for each monomial, we will use $\widehat{f}(S) = c_S$ to denote what we call the *S-Fourier Coefficient* of f. For example, since the monomial $x_1x_2x_3$ of the Fourier expansion of Maj_3 has a coefficient of $-\frac{1}{2}$, we would write $\widehat{Maj_3}(\{1, 2, 3\}) = -\frac{1}{2}$. Since the majority function has no monomial with only two terms, we would write $\widehat{Maj_3}(\{1, 2\}) = 0$. Similarly, with the parity function we defined above, $\widehat{Parity_3}(\{1, 2, 3\}) = 1$ and $\widehat{Parity_3}(\{1, 2\}) = 0$.

Since there are 2^n subsets of variables, we have 2^n monomials in our function. For any Boolean function, there are 2^n parameters and there are 2^n choices or coefficients for the values of each $\widehat{f}$, which suggests the uniqueness mentioned in Proposition 2.2.

As another notational convention, we will represent each monomial containing the elements of subset S as χ_S. More formally,

$$\chi_S = \prod_{i \in S} x_i.$$

We define $\chi_\emptyset = 1$. Note that $\chi_S = Parity_S(x)$, so every Boolean function f is a linear combination of parity functions on subsets of its variables, which is typical for Fourier analysis of other types of functions.

Again, we consider the analog of f that maps $\mathbb{F}_2^n \mapsto \mathbb{F}_2$. In that case, χ_S still represents a parity function but it is computed as $\chi_S = \sum_{i \in S} x_i$. If we consider the case where f maps $\mathbb{F}_2^n \mapsto \{-1, 1\}$, we have $\chi_S = \prod_{i \in S} (-1)^{x_i} = (-1)^{\sum_{i \in S} x_i}$.

In any case, these coefficients can help up to determine important properties of f, as we will show in Section 4.

3 Spectral Graph Theory on the Hypercube

Thus far, we have considered f as a function on the vertices of the hypercube, but have made no mention of the edges. Similarly to our lectures on spectral graph theory, we will consider a function on a graph $G = (V, E)$, but we will fix our graph to be the hypercube. Just as in previous lectures, we define $f : V \mapsto \mathbb{R}$.

In the past, we defined distribution π over the vertices by picking an edge of the graph uniformly at random and then choosing one of its endpoints uniformly at random. Since each vertex of the hyercube has n edges incident to it, our hypercube is an n-regular graph and π is also a uniform distribution over the vertices (and over all assignments in $\{-1, 1\}^n$).

Recall that we were interested in the mean of f, which is

$$\text{Mean}(f) = \mathop{\mathbf{E}}_{\substack{x \sim \{-1,1\}^n \\ \text{unif.}}} [f(x)] .$$

We say that function f is *balanced* if its mean is 0. We were also interested in the variance, which we define as

$$\mathbf{Var}[f] = \mathop{\mathbf{E}}_{x} \left[f(x)^2 \right] - \mathop{\mathbf{E}}_{x} [f(x)]^2 .$$

Note that for a Boolean function with codomain $\{-1, 1\}$, for all x, $f(x)^2 = 1$ and $\mathbf{E}_x [f(x)^2] = 1$.

For $f : \{-1, 1\} \mapsto \mathbb{R}$,the set of all functions $\{f : \{-1, 1\}^n \mapsto \mathbb{R}\}$ is a vector space with dimension 2^n. More specifically, it is an inner product space, since we defined the inner product of two functions as follows.

$$\langle f, g \rangle = \mathop{\mathbf{E}}_{x \sim \{-1,1\}} [f(x)g(x)]$$

When $f, g : \{-1, 1\} \mapsto \{-1, 1\}$, the dot product measures the *closeness* of f and g. When $f(x) = g(x)$, the expression $f(x)g(x) = 1$ and when $f(x) \neq g(x)$, the expression evaluates to -1. Then we can also express the inner product of two functions to be:

$$\langle f, g \rangle = \left(1 - \mathop{\mathbf{Pr}}_{x} [f(x) \neq g(x)]\right) - \mathop{\mathbf{Pr}}_{x} [f(x) \neq g(x)] = 1 - 2 \mathop{\mathbf{Pr}}_{x} [f(x) \neq g(x)] .$$

As such, each function can be represented as a linear combination of other functions and the parity coefficients form a basis that spans the vector space.

3.1 Parity Functions as Eigenfunctions

Recall from previous lectures that for even graph, we had a Markovian operator K, which consisted of the normalized adjacency matrix, and a Laplacian operator L, defined to be $I - K$. We also showed that they had the same eigenfunctions and the eigenvalues of L were $\lambda_i = 1 - \kappa_i$, where each κ_i is an eigenvalue of K.

We also defined $Kf(x)$ to compute the expected value of f after walking randomly from any vertex x to neighboring vertex y along a single edge. More formally,

$$Kf(x) = \mathop{\mathbf{E}}_{y \sim x} [f(y)] .$$

We also proved the following claim in a homework assignment.

Claim 3.1. *The eigenfunctions of K are the parity functions of the subsets of variables, so each χ_S is an eigenfunction.*

This means that for every S, there exists eigenvalue κ such that $K\chi_S = \kappa\chi_S$. On a given assignment x, if $\chi_S = b$, then $n - |S|$ neighbors of x have $\chi_S = b$. This is because there are n total neighbors, $|S|$ of which have one element of S flipped and $n - |S|$ which have some variable not in S flipped. Therefore, $K\chi_S = \left(\frac{n-2|S|}{n}\right)\chi_S = \left(1 - \frac{2|S|}{n}\right)\chi_S$. And our eigenvalues are $1 - \frac{2|S|}{n}$. Note that this value can range from -1 when $|S| = n$ to 1 when $|S| = 0$, so it is consistent with the upper and lower bounds from previous lectures.

Since this forms a complete basis, we can express any function f as a linear combination of our eigenfunctions. Seeing as our eigenfunctions are also the parity functions on subsets of variables, this is equivalent to what we have done in our Fourier analysis.

Now we want to show that the set of all 2^n parity functions also form an orthonormal basis.

Claim 3.2. *The parity functions χ_S for all $S \subseteq [n]$ are orthonormal.*

Proof. We want to show that for $S, T \subseteq [n]$,

$$\langle \chi_S, \chi_T \rangle = \begin{cases} 1 & \text{if } S = T \\ 0 & \text{if } S \neq T \end{cases}$$

By our definition of inner product,

$$\begin{aligned} \langle \chi_S, \chi_T \rangle &= \mathop{\mathbf{E}}_{x \sim \{-1,1\}^n} \chi_S \chi_T \\ &= \mathop{\mathbf{E}}_{x} \left[\prod_{i \in S} x_i \prod_{i \in T} x_i \right] \\ &= \mathop{\mathbf{E}}_{x} \left[\prod_{i \in S \cap T} x_i^2 \prod_{i \in S \triangle T} x_i \right] \end{aligned}$$

where $S \triangle T$ denotes their symmetric difference

$$= E_x \left[\prod_{i \in S \cap T} 1 \prod_{i \in S \triangle T} x_i \right]$$

since $x_i^2 = 1$ for all x_i

$$= E_x \left[\prod_{i \in S \triangle T} x_i \right]$$

When $S = T$, $S \triangle T = \emptyset$, so $\chi_{S \triangle T} = 1$ and $\langle \chi_S, \chi_T \rangle = 1$.

When $S \neq T$,

$$\langle \chi_S, \chi_T \rangle = E_x \left[\prod_{i \in S \triangle T} x_i \right]$$

and all x_i are independent, so

$$= \prod_{i \in S \triangle T} \mathbf{E}[x_i]$$
$$= \prod_{i \in S \triangle T} 0 = 0. \qquad \square$$

Now that we have a vector space of all functions and an orthonormal basis, every function can be written uniquely in this basis. For example, consider the Fourier coefficients of f.

Proposition 3.3. *For all $S \subseteq [n]$, $\widehat{f}(S) = \langle f, \chi_S \rangle$.*

Proof. By definition

$$\langle f, \chi_S \rangle = \sum_{T \subseteq [n]} \langle \widehat{f}(T) \chi_T(x), \chi_S(x) \rangle$$
$$= \sum_{T \subseteq [n]} \widehat{f}(T) \langle \chi_T(x), \chi_S(x) \rangle$$
$$= \widehat{f}(S). \qquad \square$$

So we can think of this inner product as the correlation of f with $Parity_S$.

We now have the following corollary, due to Parseval.

Corollary 3.4. *(Parseval Theorem) For functions f and g,*

$$\langle f, g \rangle = \sum_{S \subseteq [n]} \widehat{f}(S) \widehat{g}(S).$$

Proof. We can write $f = \sum_{S \subseteq [n]} \widehat{f}(S) \chi_S$ and $g = \sum_{T \subseteq [n]} \widehat{f}(T) \chi_T$ Then, $\langle f, g \rangle = \mathbf{E}_x[f(x)g(x)]$ can be expanded to yield $\langle f, g \rangle = E_x[\sum_{S,T \subseteq [n]} \widehat{f}(S)\widehat{g}(T)\chi_S \chi_T] = E_x[\sum_{S \subseteq [n]} \widehat{f}(S)\widehat{g}(S)] = \sum_{S \subseteq [n]} \widehat{f}(S)\widehat{g}(S)$. $\square$

Since the inner product of any function with itself is always 1, we also have that for all f,

$$\langle f, f \rangle = \sum_{S \subseteq [n]} \widehat{f}(S)^2 = 1.$$

We can check this with $Maj_3(x) = \frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 - \frac{1}{2}x_1x_2x_3$, where we have $\sum S \subseteq [n] \widehat{f}(S)^2 = 4 \times \frac{1}{4} = 1$. Because each $\widehat{f}(S)$ is nonnegative and they sum to 1, we can think of the Fourier coefficients as assigning a weight to the subsets of the variables with respect to their effect on the function's value.

3.2 Energy of F

Recall that during the spectral graph theory lectures, we were interested in what we called the energy, Dirichlet form, or local variance of f, which we defined to be

$$\mathcal{E}[f] = \frac{1}{2} \mathop{\mathbf{E}}_{x \sim y} \left[(f(x) - f(y))^2 \right].$$

We also had an equivalent definition of

$$\mathcal{E}[f] = \langle f, Lf \rangle$$

which we can now write as

$$\mathcal{E}[f] = \sum_{S \subseteq [n]} \frac{2|S|}{n} \widehat{f}(S)^2.$$

As a result,

$$\begin{aligned}
\frac{n}{4} \mathop{\mathbf{E}}_{x \sim y} \left[(f(x) - f(y))^2 \right] &= \sum_{S \subseteq [n]} |S| \widehat{f}(S)^2. \\
n \mathop{\mathbf{E}}_{x \sim y} \left[\left(\frac{f(x) - f(y)}{2} \right)^2 \right] &= \sum_{S \subseteq [n]} |S| \widehat{f}(S)^2. \\
\sum_{i=1}^{n} \mathop{\mathbf{E}}_{x} \left[\left(\frac{f(x) - f(x + e_i)}{2} \right)^2 \right] &= \sum_{S \subseteq [n]} |S| \widehat{f}(S)^2
\end{aligned}$$

where $x + e_i$ denotes assignment x with the i^{th} bit flipped.

If the i^{th} bit makes no difference in the value of f, the expression in the summation is 0. If it does and f is Boolean, the expression evaluates to $\frac{2}{2} = 1$. So

$$\sum_{S \subseteq [n]} |S| \widehat{f}(S)^2 = \sum_{i=1}^{n} \mathop{\mathbf{Pr}}_{x}[f(x) \neq f(x + e_i)]$$

We call the value inside the summation the *influence* of variable x_i on f. We call their summation, also written as

$$\mathop{\mathbf{E}}_{x \sim \{-1,1\}^n} \left[\sum_{i=1}^{n} \mathbb{1}[f(x) \neq f(x + e_i)] \right]$$

the *average sensitivity* of f. This quantity comes up frequently in complexity and lower bounds.

4 Applications and Property Testing

Now we consider some applications of the Fourier expansion of a Boolean function f. One such application is property testing. Suppose we have a graph or a Boolean function (given as a truth table) and we want to determine whether it has a certain property. The input, however, is far too large, and we don't want to read all of it. Naturally, we will use a randomized algorithm to look at some subset of the input and try to determine whether it has the property with some probability.

Suppose we want to test for the property $\mathcal{P}$, where $f \in \mathcal{P}$ iff f has the property. We assume out input is a Boolean function $f : \{-1, 1\}^n \mapsto \{-1, 1\}$. We are given query access to f, meaning that we can choose a specific value $x \in \{-1, 1\}^n$ and an oracle will give us the value $f(x)$. This line of work was started by Blum, Luby, and Rubinfeld, and then became more heavily studied in the context of property testing for graphs.

We can view this construction as a generalization for graph property testing where $n = \binom{|V|}{2}$ and the value of f is an edge indicator for every pair of vertices in the graph. Similarly, we want to determine a property of the graph, but only have query access to determine whether an edge is present in the graph. The idea of property testing is well-studied and well-understood for graphs, but the more general problem of property testing for Boolean functions is still relatively open.

Just like with our previous randomized algorithms, we want our property testing algorithm to have two features.

1. If f has property $\mathcal{P}$, we output YES with high probability (think of the probability as being $\geq \frac{2}{3}$, but it really doesn't matter as long as it's $> \frac{1}{2}$).

   This is referred to as *completeness*.

2. If f does not have property $\mathcal{P}$, we output NO with high probability. This is referred to as *soundness*.

In analyzing the algorithm, we want to minimize the number of queries made and we won't concern ourselves with the runtime of the rest of the algorithm (as long as it is polynomial), but usually these randomized algorithms are simple, so the query time dominates the algorithm's performance in most cases.

The features or our algorithm, however, may be problematic, even for some very simple properties. Consider the property $\mathcal{P}$, where $f \in \mathcal{P}$ iff $f \equiv 1$. In the event that our function $f = 1$ on every input except for one input x^*, the probability that we will end up querying $f(x^*)$ is very low. In order for the algorithm to reject f with high probability, we would need to query $\Omega(2^n)$ inputs.

This function is, however, very close to having the property we want. The solution is to relax our soundness condition to make our goals more realistic.

Definition 4.1. Function f is *ϵ-far* from satisfying property $\mathcal{P}$ if for all $g \in \mathcal{P}$, $\mathbf{Pr}_x[f(x) = g(x)] > \epsilon$. This can also be written as $\langle f, g \rangle < 1 - 2\epsilon$.

Now we can relax our soundness condition to be:

- If f is ϵ-far from satisfying $\mathcal{P}$, then we output NO with high probability.

Now we have a new parameter ϵ, which is also given as an input to the algorithm. Note that this leaves some ambiguity as to how the algorithm performs when $f \notin \mathcal{P}$, but f is not ϵ-far from having $\mathcal{P}$. In this case, we do not place any restrictions on the algorithm's output; it can answer either YES or NO. There is also a harder variant of this problem, in which we want to estimate the distance between f and the closest function that has the property. We will not consider this variant.

Now we have a plausible goal for our algorithm and our dream scenario is that the number of queries we need to perform is dependent only on ϵ. We say that a property is *testable* if such an algorithm exists. Let's go back to our example of testing for $f \in \mathcal{P}$ iff $f \equiv 1$.

Proposition 4.2. *Property $f \equiv 1$ is testable with $O(\frac{1}{\epsilon})$ queries.*

Proof. Our algorithm is as follows: Test $O(\frac{1}{\epsilon})$ inputs at random. If $f = 0$ for any of them, return NO. Otherwise, return YES.

Clearly, our completeness requirement is met. If $f \equiv 1$, then none of our queries will return $f = 0$ and we will say YES. Now, we show soundness. If f is ϵ-far from 1, $f = 1$ on no more than a $1 - \epsilon$ fraction of inputs. Therefore, the probability of all queries returning $f(x) = 1$ is at most $(1 - \epsilon)^{O(\frac{1}{\epsilon})}$, which converges to a small constant as $\epsilon \to 0$. □

Now let's consider a problem where we will use Fourier analysis. We consider the property $\mathcal{P}$, where $f \in \mathcal{P}$ if f is a parity function.

Theorem 4.3. *[BLR90] Property $\mathcal{P}$ where $f \in \mathcal{P}$ iff f is a parity function is testable with $O(\frac{1}{\epsilon})$ queries.*

Note that this is also referred to as linearity testing and probabilistically checkable proofs are an offshoot of this. This problem is more classically defined as follows: Given $f : \mathbb{F}_2^n \mapsto \mathbb{F}_2$, do there exist $a_1, \ldots, a_n \in \mathbb{F}_2$ such that $f(x) = \sum_{i=1}^n a_i x_i$?

This is equivalent to asking whether $f(x+y) = f(x) + f(y) \mod 2$ for all $x, y \in \mathbb{F}_2^n$. We translate this problem to the equivalent version for $f : \{-1, 1\}^n \mapsto \{-1, 1\}$, where we ask whether $f(x \cdot y) = f(x)f(y)$, and $(x \cdot y)_i = x_i y_i$.

For $f = Parity_S$, $f(x \cdot y) = \prod_{i \in S} x_i y_i$ and $f(x)f(y) = \prod_{i \in S} x_i \prod_{i \in S} y_i$. Our algorithm will pick a random x and y, then test this identity. The original proof is combinatorial in nature and rather complicated, so we present a different version here which relies on the Fourier expansion of f.

Proof. Our algorithm will perform the following subroutine many times.

1. Choose $x, y \sim \{-1, 1\}^n$ independently, uniformly at random.
2. Let $z \in \{-1, 1\}^n = x \cdot y$. $(z_i = x_i y_i)$
3. Query $f(x)$, $f(y)$, and $f(z)$
4. If $f(z) \neq f(x)f(y)$, return NO.

If all iterations of this subroutine have $f(z) = f(x)f(y)$, return YES.

Note that as opposed to the previous algorithm, we are choosing our queries based on something other than just random strings. While x and y are random, our three queries are linearly dependent.

Again, the completeness of this algorithm is obvious. If f is linear, all iterations of the subroutine will succeed.

Now we show soundness. We fix some $f : \{-1,1\}^n \mapsto \{-1,1\}$. What is $\mathbf{Pr}[\text{YES}]$?

$$\mathbf{Pr}[\text{YES}] = \mathop{\mathbf{E}}_{x,y} \overbrace{\left[\frac{1}{2} + \frac{1}{2}f(x)f(y)f(z)\right]}^{\mathbb{1}[\text{YES}]}$$

We can consider this expression the indicator function of YES because if $f(z) = f(x)f(y)$, the product is 1, whereas it is -1 otherwise.

$$\mathbf{Pr}[\text{YES}] = \frac{1}{2} + \frac{1}{2}\mathop{\mathbf{E}}_{x,y}\left[\left(\sum_{S\subseteq[n]}\widehat{f}(S)\prod_{i\in S}x_i\right)\left(\sum_{T\subseteq[n]}\widehat{f}(T)\prod_{i\in T}y_i\right)\left(\sum_{U\subseteq[n]}\widehat{f}(U)\prod_{i\in U}z_i\right)\right]$$

$$\mathbf{Pr}[\text{YES}] = \frac{1}{2} + \frac{1}{2}\sum_{S,T,U\subseteq[n]}\widehat{f}(S)\widehat{f}(T)\widehat{f}(U) \times \mathop{\mathbf{E}}_{x,y}\left[\prod_{i\in S}x_i\prod_{i\in T}y_i\prod_{i\in U}x_iy_i\right]$$

Now, considering just this expectation, and using the same trick as above with symmetric differences,

$$\mathbf{Pr}[\text{YES}] = \frac{1}{2} + \frac{1}{2}\sum_{S,T,U\subseteq[n]}\widehat{f}(S)\widehat{f}(T)\widehat{f}(U) \times \mathop{\mathbf{E}}_{x,y}\left[\prod_{i\in S\triangle U}x_i\prod_{i\in T\triangle U}y_i\right]$$

Because x and y are independent,

$$\mathbf{Pr}[\text{YES}] = \frac{1}{2} + \frac{1}{2}\sum_{S,T,U\subseteq[n]}\widehat{f}(S)\widehat{f}(T)\widehat{f}(U) \times \underbrace{\mathop{\mathbf{E}}_{x}\left[\prod_{i\in S\triangle U}x_i\right]}_{0 \text{ unless } S\triangle U=\emptyset}\underbrace{\mathop{\mathbf{E}}_{y}\left[\prod_{i\in T\triangle U}y_i\right]}_{0 \text{ unless } T\triangle U=\emptyset}$$

Unless $S = T = U$, everything but the leading $\frac{1}{2}$ is 0, so we have

$$\mathbf{Pr}[\text{YES}] = \frac{1}{2} + \frac{1}{2}\sum_{S\subseteq[n]}\widehat{f}(S)^3$$

$$\mathbf{Pr}[\text{YES}] = \frac{1}{2} + \frac{1}{2}\sum_{S\subseteq[n]}\widehat{f}(S)^2\widehat{f}(S)$$

$$\mathbf{Pr}[\text{YES}] \leq \frac{1}{2} + \frac{1}{2}\max_S\{\widehat{f}(S)\}\sum_{S\subseteq[n]}\widehat{f}(S)^2$$

Recall from Parseval's theorem that the sum of $\widehat{f}(S)^2 = 1$.

$$\mathbf{Pr}[\textsc{Yes}] \leq \frac{1}{2} + \frac{1}{2} \max_S \{\langle f, \chi_S \rangle\}$$

Since we are showing soundness, we can assume that f is ϵ-far from the parity function, so

$$\forall S \langle f, \chi_S \rangle < 1 - 2\epsilon < \frac{1}{2} + \frac{1}{2}(1 - 2\epsilon) = 1 - \epsilon.$$

For our subroutine, the probability that we output Yes when f is ϵ-far from the parity function is less than $1-\epsilon$. If we repeat this subroutine $O(\frac{1}{\epsilon})$ times, we will perform a total of $3O(\frac{1}{\epsilon})$ queries. Additionally, the probability that we output Yes is constant, just like with our first example. □

This technique could possibly be applied to additive combinatorics, which deals with sets of strings. The 3Sum version of the Goldbach conjecture is also shares similar terms to our proof here, but instead of $\widehat{f}(S)^3$, it has $\widehat{f}(S)^2$, which would change the analysis.

References

[BLR90] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *STOC*, pages 73–83, http://doi.acm.org/10.1145/100216.100225, 1990.