A DISCRETE SIGNAL PROCESSING FRAMEWORK FOR SET FUNCTIONS

Markus Püschel

Department of Computer Science
ETH Zurich, Switzerland

ABSTRACT

A set function associates a real (or complex) value with every subset of a given finite set S. In this paper, we derive a novel discrete signal processing (DSP) framework for such functions. This means we define and derive suitable notions of basic DSP concepts including shift, filtering, frequency response, Fourier transform, and convolution theorems. At the heart is the definition of the shift on subsets for which we consider the two most natural choices, i.e., those most analogous to the time shift in standard DSP. Set functions naturally occur in many contexts associated with probability distributions, graph cuts, sensor placements, mutual information, entropy of sets of random variables, and others. Our work offers a new set of tools for their processing.

Index Terms— Algebraic signal processing, convolutions on sets, submodular function, Fourier transform, Boolean function

1. INTRODUCTION

The foundation of signal processing rests on a well-developed theory of time-invariant systems that provides well-defined mathematical notions of signals, filters, z-transform, time- and frequency domain, Fourier transform, convolution theorems, and others. These concepts take different forms for continuous and discrete, infinite and finite (periodically extended) signals [1]. Central in time signal processing is the notion of the time shift as all other concepts can be derived from it. For example, in the discrete case, filters are just linear combinations of k-fold shifts, and the Fourier transform yields an eigendecomposition of filters. Further, it is possible to define shift-invariant signal processing frameworks for shifts other than the time shift [2]. For example, a symmetric definition of the shift yields the discrete cosine and sine transforms (or generalization of them) as Fourier transforms [3, 4], and a graph-shift, on a given graph, can be used to derive a discrete signal processing (DSP) framework on graphs [5, 6, 7].

Contributions. In this paper we derive a novel discrete signal processing framework for set functions, i.e., functions, or signals, defined on the power set 2^S of given a finite set S. Such signals thus have the form $\mathbf{s} = (s_A)_{A\subseteq S}$, where $s_A \in \mathbb{R}$ or $\in \mathbb{C}$. We first define a set transform, in analogy to the z-transform $\sum s_k z^{-k}$, that maps $\mathbf{s}$ to the formal sum $\sum_{A\subseteq S} s_A A$. Then we define two natural notions of shift on these sums in closest analogy to the time shift. From these definitions we derive all basic signal processing concepts, including the associated notions of shift-invariance, filtering, frequency response, Fourier transform, and convolution theorem.

Related work. Set functions naturally occur in many applications as probability distributions, entropy or mutual information of sets of random variables, graph cut capacity functions (see [8] for these and other examples). Equivalently, subsets of S can be viewed as Boolean vectors of length $|S|$ (recording for every element whether it is contained or not) and thus set functions as mappings on those vectors. The study of these is a classical topic (see the survey [9]) that uses the Walsh-Hadamard transform (WHT) as associated Fourier transform, which has many applications in signal processing and beyond [10, 11, 12, 13]). We will clearly distinguish from this line of work through the underlying shift operator, for which we arguably adopt a more natural choice, and on which we base our derivation. At the end of the paper will make this distinction precise.

Mathematically closest to our work is, in a very different context and with different goals, [14, 15] in theoretical computer science and [16] in game theory, which consider different convolutions on set functions but do not derive the associated DSP framework. Our work is based on the algebraic signal processing framework started in [2, 6].

2. FINITE-TIME DSP

In this section we provide well-known background on the key concepts needed for discrete signal processing (DSP) with finite-time (and periodically extended) signals [1]. These concepts include the notion of z-transform, filtering, frequency response and Fourier transform. Important for this paper is the way we structure the material, since we will follow the same structure later to derive proper forms of these concepts for DSP with set functions.

Signal. A complex, discrete finite-time signal of length n is given by $\mathbf{s} = (s_0, \dots, s_{n-1})^T \in \mathbb{C}^n$. It can be viewed as a function that maps time point i to the associated signal value s_i. Further, it is assumed to be periodically extended, i.e., for any $N \in \mathbb{Z}$, $s_N = s_{N \bmod n}$. Equivalently, if $N = kn + r$, $0 \le r < n$, then $s_N = s_r$.

***z*-transform.** For notational convenience, we write $x = z^{-1}$. The z-transform associates $\mathbf{s}$ with the polynomial $s = s(x) = \sum_{0\le i<n} s_i x^i$. To capture the periodic extension, we further have to require that for $N \in \mathbb{Z}$, $x^N = x_{N \bmod n}$, or, if $N = kn + r$ as above, $x^r = x^{kn+r} = (x^n)^k x^r$ for all $k \in \mathbb{Z}$, $0 \le r < n$, which is the same as requiring $x^n = 1$ or $x^n - 1 = 0$. In other words, the polynomial s is considered modulo $x^n - 1$. The set of these polynomials is written as $\mathbb{C}[x]/(x^n - 1)$ and is called a polynomial algebra. In summary, the finite z-transform is the mapping

$$\Phi : \mathbb{C}^n \to \mathbb{C}[x]/(x^n - 1), \quad \mathbf{s} \mapsto s = s(x) = \sum_{0\le i<n} s_i x^i. \tag{1}$$

The polynomial algebra view is known at least since [17].

Shift. Central in time DSP is the time shift, which, in the z-domain, is a multiplication by x:

$$\begin{aligned} x \cdot s &= s_0 x^1 + s_1 x^2 + \dots + s_{n-2} x^{n-1} + s_{n-1} x^n \\ &\equiv s_{n-1} x^0 + s_0 x^1 + \dots + s_{n-2} x^{n-1} \bmod x^n - 1, \end{aligned} \tag{2}$$

where we reduced x^n modulo $x^n - 1$ to 1. The effect on $\mathbf{s}$ is, as expected, the cyclic shift

$$x \cdot s = x \sum_{0 \leq i < n} s_i x^i = \sum_{0 \leq i < n} s_{i-1 \bmod n} x^i. \quad (3)$$

Note that the shift x *advances* the x^i (since $x \cdot x^i = x^{i+1}$), which *delays* the signal values s_i (up to the cyclic wraparound).

The shift be expressed as a matrix by letting x operate on the basis $(1, x, \dots x^{n-1})$:

$$\phi(x) = \begin{bmatrix} & & & 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix},$$

such that $x \cdot s$ corresponds to the matrix-vector product $\phi(x)\mathbf{s}$.

***k*-fold shift.** The k-fold shift of s is simply given by $x \cdot x \cdots x \cdot s = x^k \cdot s$, which (cyclically) delays the signal $\mathbf{s}$ by k. Its matrix representation is $\phi(x^k) = \phi(x)^k$.

Filter and convolution. A filter given by $\mathbf{h} = (h_0, \dots, h_{n-1})$ is a linear combination of k-fold shifts, i.e., in the z-domain $h = \sum_{0 \leq i < n} h_i x^i$. Filtering is multiplication in $\mathbb{C}[x]/(x^n - 1)$:

$$hs = \Big(\sum_{0 \leq i < n} h_i x^i \Big) \Big(\sum_{0 \leq i < n} s_i x^i \Big) \bmod (x^n - 1).$$

The reduction modulo $x^n - 1$ ensures that the polynomial is again of degree $n - 1$ and yields the circular convolution $\mathbf{h} \circledast \mathbf{s}$ on the values, as expected. Again, h can be expressed as a matrix which has circulant structure:

$$\phi(h) = \sum_{0 \leq i < n} h_i \phi(x)^i = \begin{bmatrix} h_0 & h_{n-1} & \dots & h_1 \\ h_1 & h_0 & \dots & h_2 \\ \vdots & \ddots & \ddots & \\ h_{n-1} & h_{n-2} & \dots & h_0 \end{bmatrix}. \quad (4)$$

In summary: $hs \Leftrightarrow \mathbf{h} \circledast \mathbf{s} \Leftrightarrow \phi(h)\mathbf{s}$.

Shift-invariance. With the above definitions, shift-invariance holds for every filter, i.e., for all h, s,

$$h(xs) = x(hs) \quad \text{or} \quad \phi(h)(\phi(x)\mathbf{s}) = \phi(x)(\phi(h)\mathbf{s}). \quad (5)$$

Fourier transform and frequency response. The associated Fourier transform is the discrete Fourier transform (DFT) and defined (up to scaling) by the matrix that diagonalizes $\phi(x)$ and thus all $\phi(h)$:

$$\mathrm{DFT}_n = [\omega_n^{k\ell}]_{0 \leq k, \ell < n}, \quad \omega_n = \exp(-2\pi j / n).$$

Namely,

$$\mathrm{DFT}_n \, \phi(h) \, \mathrm{DFT}_n^{-1} = \mathrm{diag}(h(\omega_n^0), \dots h(\omega_n^{n-1})).$$

The columns $\mathbf{f}_i$ (sometimes called pure frequencies) of DFT_n^{-1} are the eigenvectors of all $\phi(h)$ and their eigenvalues $h(\omega_n^i)$ constitute the frequency response of h:

$$\phi(h)\mathbf{f}_i = h(\omega_n^i)\mathbf{f}_i.$$

This yields the convolution theorem ($\odot$ is the pointwise product)

$$\begin{aligned} \mathbf{h} \circledast \mathbf{s} \quad &\Leftrightarrow \quad \mathrm{diag}(h(\omega_n^0), \dots h(\omega_n^{n-1})(\mathrm{DFT}_n \, \mathbf{s}) \\ &= \quad (\mathrm{DFT}_n \, \mathbf{h}) \odot (\mathrm{DFT}_n \, \mathbf{s}). \end{aligned}$$

3. DSP ON SET FUNCTIONS: NATURAL SHIFT

Note the central role of the shift in the previous section: it defined filtering and the Fourier transform and thus the entire DSP framework. Thus the key in this section is to define a suitable shift for set functions and follow the same steps to instantiate all basic concepts needed for DSP on set functions.

Signal. We consider a finite set $S = \{x_1, \dots, x_n\}$ of size $|S| = n$, and its power set, i.e., set of all subsets, usually denoted with 2^S.[1] A set function associates with each subset $A \subseteq S$ a value $s_A \in \mathbb{R}$ (choosing $\mathbb{C}$ instead makes no major difference in the following). This means our signals have the form $\mathbf{s} = (s_A)_{A \subseteq S} \in \mathbb{R}^{2^S}$, i.e., the power set becomes our index domain. Each $\mathbf{s}$ thus has length 2^n. Further, we order the subsets, and thus the components of $\mathbf{s}$ lexicographically as the Cartesian product $(\{\}, \{x_n\}) \times \cdots \times (\{\}, \{x_1\})$. For example, for $n = 3$ this yields the ordering

$$\{\}, \{x_1\}, \{x_2\}, \{x_1, x_2\}, \{x_3\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\} \quad (6)$$

***S*-transform.** We first formally define an S-transform (or set transform) as follows:

$$\Phi : \mathbb{R}^{2^S} \to \mathbb{R}[2^S], \quad \mathbf{s} \to s = \sum_{A \subseteq S} s_A A. \quad (7)$$

Here, we denote with $\mathbb{R}[2^S]$ the set of the formal sums shown. Note that in contrast to (1), where we obtained polynomials, we do not yet know how to compute with these sums. As we will see, by defining the shift we also define the computation with the sums[2] and thus the associated convolution and Fourier transform. Different notions of shift are possible and we consider the two most natural choices.

Shift. The time shift x in (2) advanced the x^i. As closest analogue, we define what we call the natural shift for set functions. Namely, for every $x_i \in S$ we define a shift

$$x_i \cdot A = A \cup \{x_i\}, \quad \text{for } A \subseteq S. \quad (8)$$

There are three differences to the time shift. First, there is not one, but n shifts. However, this also occurs in higher-dimensional DSP, where there is one shift for each dimension, and the Fourier transform is higher-dimensional (done separably) and diagonalizes all shifts. Second, since (8) implies $x_i^2 = x_i$, powers of shifts as in the time case will not occur. Third, the shifts are not invertible. Note that (8) implies that shifts commute, i.e., for all $i \neq j$ and $A \subseteq S$, $x_i x_j A = x_j x_i A$.

By linearly extending the shift (8) to operate on signals s in the S-domain (7), we get

$$x_i \cdot s = \sum_{A \subseteq S} s_A (A \cup \{x_i\}) = \sum_{A \subseteq S, x_i \in A} (s_A + s_{A \setminus \{x_i\}}) A. \quad (9)$$

The second sum is obtained by recognizing that the first sum only contains summands for sets that contain x_i and doing the variable substitution $A \cup \{x_i\} \to A$. So the effect on the s_A is not the "delay" $s_{A \setminus \{x_i\}}$ in analogy to time DSP, but $s_A + s_{A \setminus \{x_i\}}$ for $x_i \in A$ and 0 else. To get the associated matrix representation of the shift, we let it operate on the basis, i.e., all sets $(A \mid A \subseteq S)$ taking their ordering

[1]As we will see later, the x_i will play an analogous role as x (and its powers) in the previous section.

[2]Mathematically, $\mathbb{R}[2^S]$ becomes a so-called monoid ring [18] by making the set of subsets a monoid, i.e., a set with an associative operation with unit element. The monoid in time DSP (Section 2) was the set $\{x^0, \dots, x^{n-1}\}$ with multiplication modulo $x^n - 1$.

into account. As an example let $n = 3$ and consider the shift x_1. Operating on (6) as defined in (8) yields the matrix

$$\phi(x_1) = \begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1 \end{bmatrix} = I_4 \otimes \left[\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\right],$$

where I_m is the $m \times m$ identity matrix and $\otimes$ denotes the Kronecker product of matrices defined by $U \otimes V = [u_{k,\ell}B]$, for $U = [u_{k,\ell}]$, i.e., every entry of U is multiplied by the entire matrix V. In general,

$$\phi(x_i) = (I_{2^{n-i}} \otimes \left[\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\right] \otimes I_{2^{i-1}}). \tag{10}$$

***X*-fold shift.** To obtain a notion of filtering or convolution we need to define a proper notion of X-fold shift for any $X \subseteq S$. This is done by shifting in sequence with all $x \in X$: if $X = \{x_{i_1}, \dots, x_{i_k}\}$, then

$$X \cdot s = x_{i_1}(x_{i_2}(\dots(x_{i_{k-1}}(x_{i_k} \cdot s))\dots)). \tag{11}$$

Note that this is well-defined since all the shifts commute and thus the order of shifting does not matter. For $X = \{\}$, $X \cdot s = s$. In particular, $XA = A \cup X$ and thus

$$X \cdot s = \sum_{A \subseteq S} s_A(A \cup X) = \sum_{X \subseteq A \subseteq S} (\sum_{A \setminus X \subseteq B \subseteq A} s_B)A. \tag{12}$$

The last expression is obtained by first observing that in the first sum only sets containing X appear. Thus, for the last sum, we set $A = A \cup X$ and for each such A collect all B with $B \cup X = A$. The matrix representation of X is

$$\phi(X) = \prod_{j=1}^{k} \phi(x_{i_j}). \tag{13}$$

Filter and convolution. A filter is a linear combination of X-fold shifts, i.e., in the S-domain takes the form $h = \sum_{X \subseteq S} h_X X$. Filtering is multiplication in $\mathbb{R}[2^S]$,

$$hs = \Big(\sum_{X \subseteq S} h_X X\Big)\Big(\sum_{A \subseteq S} s_A A),$$

which can be computed by applying the distributivity law on h and calculating Xs as in (12) resulting in

$$hs = \sum_{A \subseteq S} \Big(\sum_{B \cup C = A} h_B s_C\Big)A,$$

which defines the associated convolution (sometimes called the covering product [14])

$$\mathbf{h} \triangle \mathbf{s} = \sum_{B \cup C = A} h_B s_C.$$

The matrix representation of h is $\phi(h) = \sum_{X \subseteq S} h_X \phi(X)$. As an example, we consider $n = 3$ and the filter $h = a\{\} + b\{x_2\} + c\{x_1, x_3\} + d\{x_1, x_2, x_3\}$, $a, b, c, d \in \mathbb{R}$. Using (10) and (13),

$$\begin{aligned} \phi(h) &= aI_8 + b(I_2 \otimes \left[\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\right] \otimes I_2) + c(I_4 \otimes \left[\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\right])(\left[\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\right] \otimes I_4) \\ &\quad + d(I_4 \otimes \left[\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\right])(I_2 \otimes \left[\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\right] \otimes I_2)(\left[\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\right] \otimes I_4) \end{aligned}$$

$$= \begin{bmatrix} a &&&&&&& \\ & a &&&&&& \\ b && a+b &&&&& \\ & b && a+b &&&& \\ &&&& a &&& \\ c & c &&& c & a+c && \\ &&&& b && a+b & \\ d & d & c+d & c+d & d & b+d & c+d & a+b+c+d \end{bmatrix}$$

Shift-invariance. Since set union is commutative, every shift x_i commutes with every X-fold shift and thus with every filter h. This means every filter is shift-invariant, i.e., for $1 \le i \le n$, and all h, s,

$$h(x_i s) = x_i(hs) \quad \text{or} \quad \phi(h)(\phi(x_i)\mathbf{s}) = \phi(x_i)(\phi(h)\mathbf{s}) \tag{14}$$

Fourier transform and frequency response. To derive the Fourier transform we need to diagonalize all shifts $\phi(x_i)$. Note that this can be done with one matrix since all the $\phi(x_i)$ commute, a consequence of (14), and easily seen from (10), using that if matrices U, U' have the same dimensions and V, V' have the same dimensions, then $(U \otimes V)(U' \otimes V') = (UU' \otimes VV')$. Because of the special structure in (10), the problem reduces to diagonalizing $\left[\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\right]$. Since

$$\left[\begin{smallmatrix} 0&1 \\ 1&-1 \end{smallmatrix}\right]^{-1} \left[\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\right] \left[\begin{smallmatrix} 0&1 \\ 1&-1 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 1&1 \\ 1&0 \end{smallmatrix}\right] \left[\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\right] \left[\begin{smallmatrix} 1&1 \\ 1&0 \end{smallmatrix}\right]^{-1} = \left[\begin{smallmatrix} 1&0 \\ 0&0 \end{smallmatrix}\right], \tag{15}$$

the discrete set Fourier transform is given by the matrix

$$\mathrm{DSFT}_{2^n} = \left[\begin{smallmatrix} 1&1 \\ 1&0 \end{smallmatrix}\right] \otimes \dots \otimes \left[\begin{smallmatrix} 1&1 \\ 1&0 \end{smallmatrix}\right],$$

and the pure frequencies (eigenvectors of all filters) are the columns of the matrix

$$\mathrm{DSFT}_{2^n}^{-1} = \left[\begin{smallmatrix} 0&1 \\ 1&-1 \end{smallmatrix}\right] \otimes \dots \otimes \left[\begin{smallmatrix} 0&1 \\ 1&-1 \end{smallmatrix}\right].$$

The columns of the latter are signals, i.e., naturally indexed with $A \subseteq S$ in our defined order. Now we also index the columns with $B \subseteq S$ in the same order, and thus write 2^S instead of 2^n. We then observe (proof by induction) that

$$\mathrm{DSFT}_{2^S} = [e_{A,B}]_{A,B \subseteq S}, \quad e_{A,B} = \begin{cases} 1 & \text{if } A \cap B = \{\} \\ 0 & \text{else} \end{cases}$$

$$\mathrm{DSFT}_{2^S}^{-1} = [f_{A,B}]_{A,B \subseteq S}, \quad f_{A,B} = \begin{cases} (-1)^{|A \cap B|} & \text{if } A \cup B = S \\ 0 & \text{else} \end{cases}$$

which gives a closed form for the pure frequencies. To confirm this, and also compute the frequency response, let B be fixed, and consider the Bth frequency $f^B = \sum_{A \subseteq S} f_{A,B}A$, and let $x \in N$ be a shift. Then

$$xf^B = \sum_{A \subseteq S, A \cup B = S} (-1)^{|A \cap B|}(A \cup \{x\}).$$

If $x \notin B$, then x is contained in every occuring A and thus $xf^B = f^B$. If $x \in B$, then every $A \cup \{x\}$ occurs twice. Once for a valid A (i.e., $A \cup B = S$) without x and once for the same A joined with x. The intersection of these with B differs in size by one and thus the associated summands cancel yielding $xf^B = 0$. So the frequency response of the shift x at the Bth frequency is either 1 or 0, as expected from the last matrix in (15).

Extending to a shift by $X \subseteq S$, using (11), yields

$$Xf^B = \begin{cases} f^B & \text{if } X \cap B = \{\} \\ 0 & \text{else} \end{cases}$$

and thus, by linear extension, we can compute the frequency response of an arbitrary filter h at frequency B through

$$hf^B = \Big(\sum_{X\subseteq S} h_X X\Big) f^B = \Big(\sum_{X\subseteq S, X\cap B=\{\}} h_X\Big) f^B.$$

This shows that the frequency response is also computed with the DSFT and yields the convolution theorem, which can also be found in [14]

$$\mathbf{h}\triangle\mathbf{s} \quad\Leftrightarrow\quad (\mathrm{DSFT}_n\,\mathbf{h})\odot(\mathrm{DSFT}_n\,\mathbf{s}).$$

Note that the DSFT can be computed with exactly $n2^{n-1}$ additions.

4. DSP ON SET FUNCTION: NATURAL “DELAY”

In Section 3 we based our derivation on the shift $x_i \cdot A = A\cup\{x_i\}$ in (8) which mimics the multiplication $x\cdot x^k = x^{k+1}$ in time DSP. The effect on the signal $(s_A)_{a\subseteq S}$ was shown in (9). Unlike in time DSP, it was not a “clean delay,” which one might expect to take the form $(s_{A\setminus\{x_i\}})_{A\subseteq S}$. The question is whether there is a definition of shift that would yield this “delay.” The answer is yes, and we define it next. Then we go through the steps as before to derive all other basic concepts. Due to space limitations we will be briefer than before. The notions of signal and set transform are as before and filters are again shift-invariant.

Shift. For $x_i \in S$ we define a shift as

$$x_i\cdot A = \begin{cases} A + A\cup\{x_i\} & x_i\notin A\\ 0 & \text{else}\end{cases} \tag{16}$$

As before, we extend linearly to signals s and compute

$$\begin{aligned} x\cdot s &= \sum_{A\subseteq S, x_i\notin A} s_A(A + A\cup\{x_i\})\\ &= \sum_{A\subseteq S, x_i\notin A} s_A A + \sum_{A\subseteq S, x_i\in A} s_{A\setminus\{x_i\}} A\\ &= \sum_{A\subseteq S} s_{A\setminus\{x_i\}} A.\end{aligned}$$

For the second equality we split the sum and set $A = A\cup\{x_i\}$ in the second sum. For the third equality we used that for $x_i \notin A$, $A\setminus\{x_i\} = A$.

The associated matrix representation of the shift now takes the form

$$\phi(x_i) = I_{2^{n-i}} \otimes \left[\begin{smallmatrix}1&0\\1&0\end{smallmatrix}\right] \otimes I_{2^{i-1}}. \tag{17}$$

The extension to an X-fold shift is done as in (11) and becomes

$$X\cdot s = \sum_{A\subseteq S} s_{A\setminus X} A.$$

Filters and convolution. Again we linearly extend the X-fold shift to the operation of an arbitrary filter $\sum_{X\subseteq S} h_X X$ and obtain

$$hs = \sum_{X\subseteq S} h_X\Big(\sum_{A\subseteq S} s_{A\setminus X} A\Big) = \sum_{A\subseteq S}\Big(\sum_{X\subseteq S} h_X s_{A\setminus X}\Big)A,$$

which defines the convolution

$$\mathbf{h}\triangleright\mathbf{s} = \sum_{X\subseteq S} h_X s_{A\setminus X}.$$

By construction, filtering is shift-invariant.

Fourier transform and frequency response. We need to diagonalize all shift matrices in (17), i.e, diagonalize first $\left[\begin{smallmatrix}1&0\\1&0\end{smallmatrix}\right]$:

$$\left[\begin{smallmatrix}1&0\\1&1\end{smallmatrix}\right]^{-1}\left[\begin{smallmatrix}1&0\\1&0\end{smallmatrix}\right]\left[\begin{smallmatrix}1&0\\1&1\end{smallmatrix}\right] = \left[\begin{smallmatrix}1&0\\-1&1\end{smallmatrix}\right]\left[\begin{smallmatrix}1&0\\1&0\end{smallmatrix}\right]\left[\begin{smallmatrix}1&0\\-1&1\end{smallmatrix}\right]^{-1} = \left[\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right].$$

Thus, the discrete set Fourier transform is now given by the matrix

$$\mathrm{DSFT}'_{2^n} = \left[\begin{smallmatrix}1&0\\-1&1\end{smallmatrix}\right]\otimes\ldots\otimes\left[\begin{smallmatrix}1&0\\-1&1\end{smallmatrix}\right],$$

which has a complexity of $n2^{n-1}$ additions. The closed form for the pure frequencies is given by

$$\mathrm{DSFT}^{-1}_{2^S} = [f_{A,B}]_{A,B\subseteq S},\quad f_{A,B} = \begin{cases}1 & \text{if } B\subseteq A\\ 0 & \text{else}\end{cases},$$

and this transform is known as Moebius transform [19, 20]. The frequency response of a filter h is computed with the DSFT_n as before and thus we get the convolution theorem

$$\mathbf{h}\triangleright\mathbf{s} \Leftrightarrow (\mathrm{DSFT}_n\,\mathbf{h})\odot(\mathrm{DSFT}'_n\,\mathbf{s}).$$

5. DSP ON SET FUNCTIONS: INVERTIBLE SHIFT

Prior work (see the survey [9]) on Fourier analysis with set functions was based on associating them with elements of the product of n cyclic groups of order 2 and obtaining the Walsh-Hadamard transform as its Fourier transform. To clearly distinguish it form our work, we very briefly present this framework analogous to the prior sections. In short: the underlying shift definition is different.

Shift. Both prior shifts were arguably natural as they mimic the time shift, but neither was invertible. An invertible shift can be defined as

$$x_i\cdot A = A\setminus\{x_i\}\cup\{x_i\}\setminus A = \begin{cases} A\cup\{x_i\}, & x_i\notin A\\ A\setminus\{x_i\}, & x_i\in A\end{cases}$$

Filters and convolution. The X-fold shift then is the so-called symmetric difference $X\cdot A = A\setminus X\cup X\setminus A$ and convolution takes the form

$$\mathbf{h}\diamond\mathbf{s} = \sum_{X\subseteq S} h_X s_{A\setminus X\cup X\setminus A}.$$

The matrix representation of the shift x_i is

$$\phi(x_i) = I_{2^{n-i}}\otimes\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]\otimes I_{2^{i-1}}. \tag{18}$$

Fourier transform. All shifts are now diagonalized by the (self-inverse up to scaling) Walsh-Hadamard transform [10, 21]

$$\mathrm{WHT}_{2^n} = \left[\begin{smallmatrix}1&1\\1&-1\end{smallmatrix}\right]\otimes\ldots\otimes\left[\begin{smallmatrix}1&1\\1&-1\end{smallmatrix}\right] = [(-1)^{|A\cap B|}]_{A,B\subseteq S}.$$

6. CONCLUSIONS

We have derived a novel, mathematically meaningful, and arguably natural, framework to do signal processing with set functions. The derivation shows that there are choices associated with the shift definition. We showed three (but more could be defined), where the last captured prior knowledge about the Walsh-Hadamard transform within our framework. With the basic theoretical foundation set, it will be exciting to further extend it and explore real-world applications with the set functions occurring in signal processing, information theory, and machine learning.

7. REFERENCES

[1] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, Prentice Hall, 2nd edition, 1999.

[2] M. Püschel and J. M. F. Moura, “Algebraic signal processing theory: Foundation and 1-D time,” *IEEE Trans. on Signal Processing*, vol. 56, no. 8, pp. 3572–3585, 2008.

[3] M. Püschel and J. M. F. Moura, “Algebraic signal processing theory: 1-D space,” *IEEE Trans. on Signal Processing*, vol. 56, no. 8, pp. 3586–3599, 2008.

[4] A. Sandryhaila, J. Kovacevic, and M. Püschel, “Algebraic signal processing theory: 1-D nearest-neighbor models,” *IEEE Trans. on Signal Processing*, vol. 60, no. 5, pp. 2247–2259, 2012.

[5] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs,” *IEEE Trans. on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[6] M. Püschel and J. M. F. Moura, “Algebraic signal processing theory,” [Online]. Available: http://arxiv.org/abs/cs.IT/0612077.

[7] M. Püschel and M. Rötteler, “Algebraic signal processing theory: 2-D hexagonal spatial lattice,” *IEEE Trans. on Image Processing*, vol. 16, no. 6, pp. 1506–1521, 2007.

[8] A. Krause and D. Golovin, *Tractability: Practical Approaches to Hard Problems*, chapter Submodular function maximization, pp. 71–104, Cambridge University Press, 2014.

[9] R. De Wolf, “A brief introduction to Fourier analysis on the Boolean cube,” *Theory of Computing Library-Graduate Surveys*, 2008.

[10] K.G. Beauchamp, *Applications of Walsh and related functions*, Academic Press, 1984.

[11] J. Kahn, G. Kalai, and N. Linial, “The influence of variables on boolean functions,” in *Proc. Foundations of Computer Science (FOCS)*, 1988, pp. 68–80.

[12] Y. Mansour, *Theoretical Advances in Neural Computation and Learning*, chapter Learning Boolean Functions via the Fourier Transform, pp. 391–424, Springer, 1994.

[13] P. Stobbe and A. Krause, “Learning fourier sparse set functions,” in *Proc. International Conference on Arti*
cial Intelligence and Statistics (AISTATS), 2012, pp. 1125–1133.

[14] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto, “Fourier meets möbius: Fast subset convolution,” in *Proc. Symposium on Theory of Computing (STOC)*, 2007, pp. 67–74.

[15] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, J. Nederlof, and P. Parviainen, “Fast zeta transforms for lattices with few irreducibles,” *ACM Trans. on Algorithms*, vol. 12, no. 1, pp. 4:1–4:19, 2015.

[16] M. Grabisch, *On Logical, Algebraic, and Probabilistic Aspects of Fuzzy Set Theory*, chapter Bases and transforms of set functions, pp. 215–231, Springer, 2016.

[17] H. J. Nussbaumer and P. Quandalle, “Fast computation of discrete Fourier transforms using polynomial transforms,” *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-27, no. 2, pp. 169–181, 1979.

[18] S. Lang, *Algebra*, Graduate Texts in Mathematics. Springer, 3rd edition, 2002.

[19] G.-C. Rota, “On the foundations of combinatorial theory. I. theory of Möbius functions,” *Z. Wahrscheinlichkeitstheorie und Verwandte Gebiete*, vol. 2, no. 4, pp. 340–368, 1964.

[20] M. Aigner, *Combinatorial Theory*, Springer, 1979.

[21] J. L. Walsh, “A closed set of normal orthogonal functions,” *American Journal of Mathematics*, vol. 45, no. 1, pp. 5–24, 1923.