On Factor Graphs and the Fourier Transform

Yongyi Mao, *Member, IEEE,* and Frank R. Kschischang, *Senior Member, IEEE*

***Abstract*—We introduce the concept of convolutional factor graphs, which represent convolutional factorizations of multivariate functions, just as conventional (multiplicative) factor graphs represent multiplicative factorizations. Convolutional and multiplicative factor graphs arise as natural Fourier transform duals. In coding theory applications, algebraic duality of group codes is essentially an instance of Fourier transform duality. Convolutional factor graphs arise when a code is represented as a sum of subcodes, just as conventional multiplicative factor graphs arise when a code is represented as an intersection of supercodes. With auxiliary variables, convolutional factor graphs give rise to "syndrome realizations" of codes, just as multiplicative factor graphs with auxiliary variables give rise to "state realizations." We introduce normal and co-normal extensions of a multivariate function, which essentially allow a given function to be represented with either a multiplicative or a convolutional factorization, as is convenient. We use these function extensions to derive a number of duality relationships among the corresponding factor graphs, and use these relationships to obtain the duality properties of Forney graphs as a special case.**

***Index Terms*—Duality, factor graphs, Forney graphs, Fourier transform, graphical models, normal realizations, state realizations, syndrome realizations, Tanner graphs.**

I. INTRODUCTION

THE excellent performance of turbo codes and low-density parity-check codes has made the subject of codes on graphs and iterative decoding algorithms a major research focus in coding theory. The graphical models that are relevant to this paper are Tanner graphs [1], Tanner–Wiberg–Loeliger (TWL) graphs [2], [3], factor graphs [4], and Forney (normal) graphs [5].

Briefly, a Tanner graph [1] is a bipartite graph that expresses the relationship between codeword symbols—represented as symbol vertices—and the (typically linear) constraints—represented as check vertices—that define valid codewords. Every check vertex connects, via edges, to the symbol vertices that it checks. A TWL graph [2], [3] is a modified Tanner graph which contains state vertices, representing auxiliary symbols. These auxiliary symbols are not codeword symbols, but they often simplify the code description. Trellis representations of codes, in which the trellis states form the auxiliary symbol alphabets, may be viewed as an expanded form of TWL graph. A Forney graph (or normal realization) [5] arises from a TWL graph satisfying the requirement that all symbol vertices have degree one and all state vertices have degree two. This requirement leads to a significant simplification of the graph, as the state and symbol vertices may be suppressed, leading to a graph (a Forney graph) in which graph edges (and "half edges") represent symbols and graph vertices represent the codeword-defining constraints placed on these symbols. TWL graphs associated with trellises naturally obey the necessary degree restrictions and so immediately give rise to the corresponding Forney graphs. An important and elegant property of Forney graphs, not shared with TWL graphs in general, is that a *local* dualization procedure—in which local constraints are replaced with their duals, and certain variable sign changes are introduced—applied to a Forney-graph realization of a group code yields a Forney-graph realization of the dual code.

In Tanner, TWL, and Forney graphs, the common underlying philosophy is to use a set of constraints to define a behavior in some configuration space. The framework of factor graphs [4] adopts a somewhat different philosophy: a factor graph represents the factorization structure of a multivariate function (e.g., the joint probability mass function of a number of random variables). Suppose the function factors into several functions (local factors), each involving a subset of the variables. The corresponding (bipartite) factor graph has a factor vertex corresponding to each local factor, and a variable vertex corresponding to each variable. An edge joins a factor vertex to a variable vertex if and only if the corresponding variable is an argument of the corresponding local factor.

A factor graph serves as a model of a code by representing the code's indicator function—the $\{0,1\}$-valued function defined on the appropriate configuration space that takes value 1 only on the set of valid codewords. (Via scaling, such an indicator function may become a probability mass function that is uniform on valid codewords and is zero on noncodewords.) When the code is defined via a number of local constraints, i.e., as the intersection of a number of supercodes, then the code's indicator function factors as the product of indicator functions for the local constraints [4], and hence Tanner, TWL, and Forney graphs may all be considered as instances of a factor graph.

Motivated by the elegant duality properties of Forney graphs [5], the results of this paper were developed from an attempt to determine the factor-graph relevance of Forney-graph duality. In this work, we discover that factor graphs have natural "duals," which we call convolutional factor graphs. We now refer to the original factor graphs as multiplicative factor graphs. Similar to a multiplicative factor graph, a convolutional factor graph is also a bipartite graph, but it represents the convolution (rather than the multiplication) of local factors.

Given a function represented by a multiplicative factor graph, we show that the Fourier transform of the function can be repre-

Manuscript received July 16, 2002; revised January 12, 2005.

Y. Mao was with the Edward S. Rogers, Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, He is now with the School of Information Technology and Engineering, University of Ottawa, Ottawa, ON K1N 6N5, Canada (e-mail: yymao@site.uottawa.ca).

F. R. Kschischang is with the Edward S. Rogers, Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: frank@comm.utoronto.ca).

Communicated by R. Koetter, Associate Editor for Coding Theory.
Digital Object Identifier 10.1109/TIT.2005.846404

sented by a convolutional factor graph, where each local factor is the Fourier transform of its counterpart in the multiplicative factor graph. We refer to such a pair of factor graphs as dual factor graphs. In coding theory it is well known that the indicator function of a group code and that of its dual code are, up to scale, a Fourier transform pair. This implies that when a factor graph represents a code, the dual factor graph represents the dual code, and that every pair of corresponding factors in the two graphs represent a pair of dual local codes. Thus, via convolutional factor graphs, we obtain a duality theory for Tanner and TWL graphs without imposing degree restrictions on their vertices.

As noted earlier, when a code is defined as the intersection of a number of supercodes, the indicator function naturally factors multiplicatively. This observation is the basis on which the parity-check matrix of a linear code gives rise to a multiplicative factor graph. We show that when a code is defined as the sum of a number of subcodes, the indicator function naturally factors convolutionally. This dual observation is the basis on which the generator matrix of a linear code gives rise to a convolutional factor graph. This result, in retrospect, is interesting in its own right, since it naturally motivates the introduction of convolutional factor graphs, regardless of any particular duality properties.

Prior to this work, graphical models for codes have been mostly concerned with "state realizations" from which the code of interest is obtained by puncturing the auxiliary variables. In this work, we show that convolutional factor graphs yield natural "syndrome realizations" from which the code of interest is obtained by shortening the auxiliary variables.

It is also interesting to reveal the relationship between Forney-graph duality and factor-graph duality. In this paper, we show that every factor graph (multiplicative or convolutional) representing a function f can be *normalized* or "*conormalized*" to another factor graph representing a function closely related to f. We develop a set of duality results for such normalized and conormalized factor graphs, and re-interpret Forney-graph duality in this framework.

The notion of Fourier transform is well defined for functions defined on locally compact abelian (LCA) groups. These groups include the real numbers (under addition), the integers (under addition), the compact interval $[0, 1)$ (under addition modulo 1), the circle group $\mathbb{T}$ of unit-magnitude complex numbers (under complex multiplication), finite abelian groups, the direct product of any finite collection of these groups, etc., as may arise in many applications in coding theory and beyond. However, for simplicity and ease of reading, in this paper we give results only for *finite* abelian groups.

The remainder of the paper is organized as follows. In Section II, we introduce the key mathematical components forming the foundation of this paper, where we also develop some needed mathematical operations, including function extensions, multivariate convolution, etc. In Section III, we describe character groups and the Fourier transform over finite abelian groups. In Section IV, we extend the notion of factor graphs to include both multiplicative and convolutional factor graphs, and present their roles in intersection and sum representations of codes and the duality between the two types of factor graphs. State and syndrome realizations are also discussed in this section. In Section V, we introduce normalized and conormalized forms of factor graphs, the duality between these forms, and its connection to Forney-graph duality. In Section VI, we provide some brief conclusions.

II. Preliminaries

A. Finite Abelian Groups

In this paper, we will deal with complex-valued functions defined on finite abelian groups; however, as noted, most of our results will hold (with some modification) to the more general case of LCA groups, which is the general setting in which Pontryagin duality applies. An excellent and readable review of the relevant concepts in the context of the dynamics of group codes is given in [6].

The size of a finite abelian group G will be denoted as $|G|$. With one exception—the circle group $\mathbb{T}$ defined in Section III—abelian groups will be written additively, i.e., the group operation is denoted by $+$, the identity element is denoted by 0, and the inverse of group element a is denoted by $-a$. We will write $H \leq G$ to mean that H is a subgroup of G. We will also write $G_1 \cong G_2$ to mean that the groups G_1 and G_2 are isomorphic.

Let G be a finite abelian group and suppose that $H_1 \leq G$ and $H_2 \leq G$. Then both the intersection $H_1 \cap H_2$ and the sum

$$H_1 + H_2 := \{h_1 + h_2 : h_1 \in H_1, h_2 \in H_2\}$$

of H_1 and H_2 are subgroups of G. An element $h \in H_1 + H_2$ can be written as $h = h_1 + h_2$ (where $h_1 \in H_1$, $h_2 \in H_2$) in exactly $|H_1 \cap H_2|$ different ways. From this it follows that if elements $h_1 \in H_1$ and $h_2 \in H_2$ are selected independently and uniformly at random, then there is equal probability of obtaining each element of the sum $H_1 + H_2$. In other words, the sum of two independent uniform random variables over finite abelian groups H_1 and H_2 is uniform over the sum $H_1 + H_2$.

An abelian group G is an internal direct product of subgroups A_1 and A_2 if $A_1 \cap A_2 = \{0\}$ and $G = A_1 + A_2$. Since $|A_1 \cap A_2| = 1$, it follows that every group element $g \in G$ may be written *uniquely* as a sum $g = a_1 + a_2$, where $a_1 \in A_1$ and $a_2 \in A_2$, so $|G| = |A_1||A_2|$.

We will take notational advantage of the correspondence between the element (a_1, a_2) in the external direct product $A_1 \times A_2$ and the element $a_1 + a_2$ in the internal direct product. Using this correspondence, we may denote any $g \in G$ either by a pair (a_1, a_2), or by a sum $a_1 + a_2$ with $a_1 \in A_1$ and $a_2 \in A_2$. We refer to a_1 as the first coordinate of g, and to a_2 as the second coordinate of g. Throughout the paper, the notation $A_1 \times A_2$ will be used to denote an *internal* direct product, though we will often use the ordered pair notation of external direct products to denote individual elements of $A_1 \times A_2$. This standard abuse of notation should not result in any confusion.

Of course, G may also be the direct product of more than two subgroups. If $I = \{1, 2, \ldots, n\}$ is a finite index set, we will use the notation $G = \prod_{i \in I} A_i$ to mean that $G = A_1 + A_2 + \cdots + A_n$ and for every $i \in I$, $A_i \cap (A_1 + \cdots + A_{i-1} + A_{i+1} + \cdots + A_n) = \{0\}$. In this case, an element $g \in G$ has n coordinates. We will often refer to subgroups of G as (group) codes.

If $G = A_1 \times A_2$, then there is a natural correspondence $a_1 \leftrightarrow a_1 + A_2$ between the elements of A_1 and the set G/A_2 of cosets of A_2 in G, under which the subgroup A_1 is isomorphic to the quotient group G/A_2. Thus, the elements of A_1 may be used either to label A_1 or as representatives for the elements of G/A_2.

Let H be a subgroup of the direct product $A_1 \times A_2$. We define the *projection* of H on A_1 as $H_{|A_1} := H + A_2$. The projection of H on A_1 consists of all elements of $A_1 \times A_2$ whose first coordinate agrees with the first coordinate of some element of H, and whose second coordinate is free to be any element of A_2. Dually, we define the *cross section* of H on A_1 as $H_{:A_1} := H \cap A_1$. The cross section of H on A_1 consists of all elements of H whose second coordinate is zero. Clearly, we have

$$\{0\} \le H_{:A_1} \le H \le H_{|A_1} \le A_1 \times A_2. \tag{1}$$

If $H \le A_1 \times A_2$ is regarded as a code, then we refer to $H_{|A_1}$ as a *punctured code* associated with H, and the second coordinate of the elements of $H_{|A_1}$ is said to be the *punctured coordinate*. Likewise, we refer to $H_{:A_1}$ as a *shortened code* associated with H, and the second coordinate of the elements of $H_{:A_1}$ is said to be the *shortened coordinate*. These definitions slightly contravene conventional coding-theoretic usage, as usually codewords are restricted to their unpunctured and unshortened coordinates. For the purposes of this paper, however, it is convenient to maintain the subgroup relationship (1) within the ambient group $A_1 \times A_2$.

B. Functions on Finite Abelian Groups

A complex-valued function on an abelian group G is a map $f : G \to \mathbb{C}$ that maps each $g \in G$ to a complex value $f(g)$. The set of all complex-valued functions on G is denoted $\mathbb{C}^G$. If H is a subgroup of G (or, more generally, a subset of G), we will denote the restriction of f to H as $r_H[f]$.

An important class of functions in this paper are the *indicator functions* associated with propositional (i.e., true/false) functions with domain G. If $P(g)$ is a propositional function, then the indicator function $\delta[P(g)] \in \mathbb{C}^G$ is defined as

$$\delta[P(g)] := \begin{cases} 1, & \text{if } P(g) \\ 0, & \text{otherwise.} \end{cases}$$

The following special cases arise often enough to merit special notation: i) the "Kronecker delta" $\delta(g) := \delta[g = 0]$ and ii) the "subgroup indicator" $\Phi_H(g) := \delta[g \in H]$, defined for any subgroup $H \le G$. In particular, note that Φ_G is the constant function $\Phi_G(g) = 1$. The scaled subgroup indicator function

$$U_H(g) := \frac{1}{|H|}\Phi_H(g)$$

will also be useful; this function may be interpreted as a probability mass function that is uniform over the elements of H, and zero over elements not in H.

Let H be any subgroup of G. A function $f \in \mathbb{C}^G$ is said to be H*-impulsive* if $f(g) = 0$ for all $g \notin H$. A function $f \in \mathbb{C}^G$ is said to be H*-periodic* if $f(g) = f(g + h)$ for all $g \in G$ and all $h \in H$. An H-periodic function may be regarded as a function on the quotient group G/H, since f maps every element of a fixed coset $g + H$ of H in G to the same value. The only functions that are both H-impulsive and H-periodic are scalar multiples of the indicator function Φ_H.

A function f on a direct product group $G = A_1 \times A_2$ may be regarded as a function $f(a_1, a_2)$ of two variables: $a_1 \in A_1$ and $a_2 \in A_2$. If f is A_1- or A_2-impulsive or if f is A_1- or A_2-periodic, then f may be regarded as a *local function*, i.e., as a function of only one variable.

Specifically, if $f(a_1, a_2)$ is A_1-impulsive, then it is natural to regard f as being a function only on A_1, i.e., to identify f with the restricted function $r_{A_1}[f] \in \mathbb{C}^{A_1}$. For each $a_1 \in A_1$, this function takes value $r_{A_1}[f](a_1) = f(a_1, 0)$. Likewise, if $f(a_1, a_2)$ is A_1-periodic, then, as noted above, it is natural to regard f as being a function on G/A_1, i.e., to identify f with the restricted function $r_{A_2}[f] \in \mathbb{C}^{A_2}$. For each $a_2 \in A_2$, this function takes value $r_{A_2}[f](a_2) = f(0, a_2)$.

Conversely, we may extend a given local function of one variable to a function of two variables in two dual ways, both of which will play an important role in this paper.

Specifically, given a function $f \in \mathbb{C}^{A_1}$, we may define a corresponding A_1-impulsive function in $\mathbb{C}^{A_1 \times A_2}$ by $\underline{f}(a_1, a_2) = f(a_1)$ if $a_2 = 0$, and 0 otherwise. We call $\underline{f}$ the *impulsive extension* of f. For later use, we note that $\underline{f}(a_1, a_2)$ can be written as the product $\underline{f}(a_1, a_2) = f(a_1)\delta(a_2)$.

Likewise, given a function $f \in \mathbb{C}^{A_2}$, we may define a corresponding A_1-periodic function in $\mathbb{C}^{A_1 \times A_2}$ by $\overline{f}(a_1, a_2) = f(a_2)$ for all $a_1 \in A_1$. We call $\overline{f}$ the *periodic extension* of f.

In the following subsection, we will see that impulsive extensions are needed to define the convolution of local functions, just as periodic extensions are needed to define the multiplication of local functions.

C. Factorization of Functions

Factorization of functions plays a fundamental role in this work. We will consider two dual types of function product: multiplication and convolution, and hence two dual types of function factorization.

Multiplication of two functions $f_1, f_2 \in \mathbb{C}^G$ is defined in the usual way, as $(f_1 \cdot f_2)(g) := f_1(g)f_2(g)$, i.e., as the pointwise function product. Convolution of two functions $f_1, f_2 \in \mathbb{C}^G$ is defined, in the abelian group case [7], as

$$(f_1 * f_2)(g) := \sum_{g' \in G} f_1(g')f_2(g - g').$$

More generally, for finite index set $J = \{1, 2, \ldots, |J|\}$, the product (either multiplicative or convolutional) of the elements of a family $\{f_j : j \in J\}$ of functions in $\mathbb{C}^G$ is given as

$$\overset{\circ}{\prod_{j \in J}} f_j := \left(\overset{\circ}{\prod_{j \in J - \{|J|\}}} f_j \right) \circ f_{|J|}$$

where $J - \{|J|\}$ denotes the relative complement of $\{|J|\}$ in J, and where the "$\circ$" operation must be replaced with either a multiplication ("$\cdot$") operation or a convolution ("$*$") operation. Both multiplication and convolution are commutative and associative.

We remark that convolution arises naturally when considering sums of independent random variables. If X_1 and X_2 are independent random variables distributed over G, with probability mass functions f_1 and f_2, respectively, then the sum $X_1 + X_2$ (the sum taken in G) is distributed with probability mass function $f_1 * f_2$. Convolution also arises naturally as the ring product in the group ring $\mathbb{C}G$ (see, e.g., [7, p. 245]).

A function $f \in \mathbb{C}^G$ is said to factor multiplicatively as the product of $f_1, f_2 \in \mathbb{C}^G$ if $f = f_1 \cdot f_2$. If $G = A_1 \times A_2 \times A_3$, then we may allow f_1 and f_2 to be local functions provided that we are careful to extend f_1 and f_2 to G in the appropriate way. We follow the usual convention that each factor is replaced with its corresponding periodic extension. Thus, if $f_1 \in \mathbb{C}^{A_1\times A_2}$ and $f_2 \in \mathbb{C}^{A_2\times A_3}$, then the multiplication $f_1 \cdot f_2$ is defined as

$$(f_1 \cdot f_2)(a_1, a_2, a_3) := \overline{f_1}(a_1, a_2, a_3) \cdot \overline{f_2}(a_1, a_2, a_3). \quad (2)$$

Likewise, a function $f \in \mathbb{C}^G$ is said to factor convolutionally as the product of $f_1, f_2 \in \mathbb{C}^G$ if $f = f_1 * f_2$. Again, if $G = A_1 \times A_2 \times A_3$, then we may allow f_1 and f_2 to be local functions, provided that we are careful to extend f_1 and f_2 to G in an appropriate way. Here we follow the dual convention that each factor is replaced with its corresponding impulsive extension. Thus, if $f_1 \in \mathbb{C}^{A_1\times A_2}$ and $f_2 \in \mathbb{C}^{A_2\times A_3}$, then the convolution $f_1 * f_2$ is defined as

$$(f_1 * f_2)(a_1, a_2, a_3) := \underline{f_1}(a_1, a_2, a_3) * \underline{f_2}(a_1, a_2, a_3). \quad (3)$$

To see that the impulsive extension gives a reasonable definition of the convolution of local functions, observe that

$$\underline{f_1}(a_1, a_2, a_3) = f_1(a_1, a_2)\delta(a_3)$$

and

$$\underline{f_2}(a_1, a_2, a_3) = f_2(a_2, a_3)\delta(a_1).$$

Convolving the two functions, we find that

$$\begin{aligned}(f_1 * f_2)&(a_1, a_2, a_3)\\ &= \sum_{a_1', a_2', a_3'} \underline{f_1}(a_1', a_2', a_3')\underline{f_2}(a_1 - a_1', a_2 - a_2', a_3 - a_3')\\ &= \sum_{a_1', a_2', a_3'} f_1(a_1', a_2')\delta(a_3')f_2(a_2 - a_2', a_3 - a_3')\delta(a_1 - a_1')\\ &= \sum_{a_2'} f_1(a_1, a_2')f_2(a_2 - a_2', a_3).\end{aligned}$$

In other words, the convolution of the two local functions f_1 and f_2 is performed via their common argument $a_2 \in A_2$, with the remaining variables interpreted as parameters. This interpretation can be made precise by viewing the function $f_1(a_1, a_2)$ as a family of functions in $\mathbb{C}^{A_2}$ indexed by A_1 and the function $f_2(a_2, a_3)$ as a family in $\mathbb{C}^{A_2}$ indexed by A_3. Convolving the two families yields a third family indexed by $A_1 \times A_3$, which may be viewed as the function $(f_1 * f_2)(a_1, a_2, a_3)$. From this perspective, multiplication of two local functions may also be viewed as a multiplication of two families of functions. Any reasonable definition of multiplication or convolution should give a result that is consistent with this perspective. It is possible to show that local functions f_1 and f_2 can generally be extended to global functions (i.e., elements of $\mathbb{C}^{A_1\times A_2\times A_3}$) in only one way that makes convolution of the extended functions consistent with this "variables-as-parameters" view, and that is to use impulsive extensions as in (3). Likewise, there is in general only one way to extend f_1 and f_2 to create a consistent multiplication, and that is to use periodic extensions as in (2).

An important special case arises when $A_2 = \{0\}$, in which case $f_1(a_1, a_2)$ may be regarded as the local function $f_1(a_1)$ and $f_2(a_2, a_3)$ may be regarded as the local function $f_2(a_3)$. In this case, multiplication and convolution coincide, as stated in the following theorem.

Theorem 1: Let $f_1(a_1) \in \mathbb{C}^{A_1}$ and $f_2(a_3) \in \mathbb{C}^{A_3}$ be local functions with no variable in common. Then $(f_1 \cdot f_2)(a_1, a_3) = (f_1 * f_2)(a_1, a_3)$.

Proof: This follows directly from (2) and (3). □

In other words, a separable (product) function may be regarded as either an ordinary multiplicative product or as a convolutional product of the appropriate (periodic or impulsive) extensions of the component local functions. We will make use of this simple observation in Section V.

In this paper, we will often consider general factorizations of the following type. For any positive integer n, let $I = \{1, 2, \ldots, n\}$ be an index set, and let $G = \prod_{i\in I} A_i$ be the direct product of a family of subgroups $A_1, A_2, \ldots, A_n$ of G. For any nonempty subset L of I, let $G_L = \prod_{i\in L} A_i$ be the subgroup of G formed as the direct product of those A_i indexed by L. We will let $(a_1, \ldots, a_n)$ denote an arbitrary element of G and a_L denote an arbitrary element of G_L. Since each element of G has n coordinates, we may regard a function $f \in \mathbb{C}^G$ as the function $f(a_1, \ldots, a_n)$ of n variables $\{a_i : i \in I\}$, where variable a_i takes values in group A_i.

Now let $J = \{1, 2, \ldots, m\}$ be another index set, which will be used to index a collection of local factors, and suppose that f factors (either multiplicatively or convolutionally) as a product of local functions $f_j \in \mathbb{C}^{G_{I(j)}}$, $j \in J$, where each $I(j)$ is a nonempty subset of I for each $j \in J$. Such a factorization may be denoted as

$$\begin{aligned}f(a_1, \ldots, a_n) &= \overset{\circ}{\prod_{j\in J}} f_j(a_{I(j)})\\ &= f(a_{I(1)}) \circ f(a_{I(2)}) \circ \cdots \circ f(a_{I(m)}) \quad (4)\end{aligned}$$

where the generic function product operator $\circ$ must be replaced with multiplication or convolution according to the given type of factorization.

D. Sampling and Averaging of Functions

Given any subgroup H of G, we define the *H-sampling* of a function $f \in \mathbb{C}^G$ as the process of multiplying by the indicator function Φ_H. That is, given a function $f \in \mathbb{C}^G$, the H-sampling of f is the function $\Phi_H \cdot f \in \mathbb{C}^G$, which takes value $f(g)$ when $g \in H$, and 0 otherwise. Thus, an H-sampled function is H-impulsive. Indeed, a function f is H-impulsive if and only if it is invariant under H-sampling, i.e., if and only if $\Phi_H \cdot f = f$. We note that H-sampling behaves as a projection, i.e.,

$$\Phi_H \cdot (\Phi_H \cdot f) = \Phi_H \cdot f.$$

Likewise, we define the *H-averaging* of a function $f \in \mathbb{C}^G$ as the process of replacing $f(g)$ by its average value over the coset

$g+H$ of H. This process may be performed by convolving with the uniform probability mass function U_H, since

$$\begin{aligned}(U_H * f)(g) &= \sum_{g' \in G} U_H(g') f(g-g') \\ &= \frac{1}{|H|} \sum_{g' \in G} \Phi_H(g') f(g-g') \\ &= \frac{1}{|H|} \sum_{g' \in H} f(g-g')\end{aligned}$$

the average value of f over the coset $g+H$. In other words, given a function $f \in \mathbb{C}^G$, the H-averaged function is $U_H * f \in \mathbb{C}^G$. An H-averaged function is by definition H-periodic. Indeed, a function f is H-periodic if and only if it is invariant under H-averaging, i.e., if and only if $U_H * f = f$. We note that H-averaging behaves as a projection, i.e.,

$$U_H * (U_H * f) = U_H * f.$$

In the case when f is an indicator function for some subgroup H_2 of G, then, as the following theorem shows, H_1-sampling and H_1-averaging of f give rise to indicator functions for the intersection and sum groups, respectively.

Theorem 2: If H_1 and H_2 are subgroups of a finite abelian group G, then

$$\Phi_{H_1 \cap H_2} = \Phi_{H_1} \cdot \Phi_{H_2} \quad \text{and} \quad U_{H_1+H_2} = U_{H_1} * U_{H_2}.$$

Proof: The product $\Phi_{H_1}(g)\Phi_{H_2}(g) = 1$ if and only if $\Phi_{H_1}(g) = 1$ and $\Phi_{H_2}(g) = 1$, which occurs if and only if g is an element of both H_1 and H_2. The convolution $U_{H_1}(g) * U_{H_2}(g)$ represents the probability mass function of the sum of two independent random variables uniformly distributed over H_1 and H_2, which is uniformly distributed over the sum group $H_1 + H_2$. □

In the special case of a group code $C \leq A_1 \times A_2$ with indicator function Φ_C, the A_1-sampling of Φ_C yields

$$\Phi_{A_1}\Phi_C = \Phi_{A_1 \cap C} = \Phi_{C:A_1},$$

the indicator function for the cross section of C on A_1. Similarly, the A_2-averaging of U_C yields

$$U_{A_2} * U_C = U_{A_2+C} = U_{C|A_1},$$

the uniform probability mass function over the projection of C on A_1.

III. Fourier Transform Duality in Finite Abelian Groups

Time–frequency duality is probably the most fundamental concept and the most useful tool in signal analysis. As all electrical engineers know, convolution of signals in the time domain is equivalent to multiplication of the Fourier transforms of the signals in the frequency domain. The theory of Fourier transforms on LCA groups is well developed (see, for example, [7]–[9]), and will be used, in the finite abelian group setting of this paper, to establish various duality results. We begin by defining the "frequency domain," that is, the character groups.

A. Character Groups

The *circle group* (or 1-torus) is the group of unit-magnitude complex numbers under complex multiplication. Following [7], we will denote the circle group by $\mathbb{T}$. As we have already noted, we will write the group operation of $\mathbb{T}$ multiplicatively.

A *character* χ of a finite abelian group G is a homomorphism $\chi : G \to \mathbb{T}$ mapping G into the circle group. The set of characters of G forms an abelian group $G^\wedge$, called the *character group* of G, under the group operation $+$ defined by

$$(\chi + \chi')(g) = \chi(g)\chi'(g) \tag{5}$$

for any $\chi, \chi' \in G^\wedge$. The *identity character* is the trivial homomorphism that maps every $g \in G$ to $1 \in \mathbb{T}$. Outside coding theory, the character group of G is usually referred to as the "dual group" of G, but to avoid confusion with "dual codes," we will not use this terminology.

The evaluation $\chi(g)$ of a character $\chi \in G^\wedge$ at $g \in G$ is often written as a pairing $\langle \chi, g \rangle$. The pairing is "bihomomorphic" in the sense that the two relationships

$$\begin{aligned}\langle \chi, g+g' \rangle &= \langle \chi, g \rangle \langle \chi, g' \rangle \\ \langle \chi + \chi', g \rangle &= \langle \chi, g \rangle \langle \chi', g \rangle\end{aligned}$$

hold for all $g, g' \in G$ and all $\chi, \chi' \in G^\wedge$. A character $\chi \in G^\wedge$ and a group element $g \in G$ are said to be *orthogonal* if $\langle \chi, g \rangle = 1$.

For any fixed g, the mapping $c_g(\chi) := \langle \chi, g \rangle$ from $G^\wedge$ into $\mathbb{T}$ is a homomorphism, and hence is an element of $G^{\wedge\wedge}$, the character group of $G^\wedge$. In fact, *all* characters of $G^\wedge$ are obtained in this manner, and the natural map taking g to c_g is an isomorphism between G and $G^{\wedge\wedge}$. This result—that G and $G^{\wedge\wedge}$ are naturally isomorphic—is often referred to as Pontryagin duality.

In the case of a finite abelian group, in addition to having $G \cong G^{\wedge\wedge}$, we also have $G \cong G^\wedge$, so in particular $|G| = |G^\wedge|$. For example, consider the cyclic group $\mathbb{Z}_m = \{0, 1, \dots, m–1\}$ under integer addition modulo m, in which the element 1 generates the entire group. If χ is a character of Z_m, and the value $\langle \chi, 1 \rangle$ is specified, then the value $\langle \chi, a \rangle$ is determined for all $a \in Z_m$, since, e.g., $\langle \chi, 1+1 \rangle = \langle \chi, 1 \rangle^2$, $\langle \chi, 1+1+1 \rangle = \langle \chi, 1 \rangle^3$, etc. In particular, $1 = \langle \chi, 0 \rangle = \langle \chi, 1 \rangle^m$, so $\langle \chi, 1 \rangle$ must be a complex mth root of unity, i.e., an integer power of $e^{i2\pi/m}$. The map that takes $a \in \mathbb{Z}_m$ to the character χ with value $\langle \chi, 1 \rangle = e^{i2\pi a/m}$ is an isomorphism between Z_m and the m distinct characters of $\mathbb{Z}_m$.

It is well known that the character group of a finite direct product $G = \prod_{i \in I} A_i$, where $I = \{1, 2, \dots, n\}$, is itself a direct product. This direct product may be written

$$G^\wedge = \prod_{i \in I} A_i^\wedge$$

for subgroups $A_i^\wedge \leq G^\wedge$ having the properties that i) the restriction $r_{A_i}[A_i^\wedge]$ is the character group of A_i, and ii) if $j \neq i$, $\langle \chi_i, a_j \rangle = 1$ for all $\chi_i \in A_i^\wedge$ and all $a_j \in A_j$. From the latter property it follows that the pairing $\langle (\chi_1, \dots, \chi_n), (a_1, \dots, a_n) \rangle$ is given by the componentwise product

$$\langle (\chi_1, \dots, \chi_n), (a_1, \dots, a_n) \rangle = \prod_{i \in I} \langle \chi_i, a_i \rangle.$$

B. Orthogonal Subgroups

A character $\chi \in G^\wedge$ is said to be *orthogonal* to a subgroup H of G if it is orthogonal to all elements of H, i.e., if $\langle \chi, h \rangle = 1$ for all $h \in H$. The set $H^\perp$ of all characters of G orthogonal to H is the *orthogonal subgroup* associated with H. If H is regarded as a code, then $H^\perp$ is the *dual code*. The second major Pontryagin duality result is the fact that $(H^\perp)^\perp$ (the dual of the dual of H) is isomorphic to H under the same map that associates G with $G^{\wedge\wedge}$, and hence we may write $(H^\perp)^\perp = H$. We have $\{0\}^\perp = G^\wedge$ and $G^\perp = \{0\}$. It is also easy to see that in a direct product group $G = A_1 \times A_2$ we have $A_1^\perp = A_2^\wedge$.

Let H be a subgroup of G and let χ be any character of G. The restriction $r_H[\chi]$ of every character χ of G is a character of H, and it can be shown that, as χ ranges over $G^\wedge$, all characters of H are obtained in this way, i.e., $H^\wedge = r_H[G^\wedge]$. Indeed, r_H is a group homomorphism from $G^\wedge$ to $H^\wedge$ having kernel $\ker_{r_H}(G^\wedge) = H^\perp$. Two characters χ, χ' of G coincide on H, i.e., $r_H[\chi] = r_H[\chi']$, if and only if χ and χ' are elements of the same coset of $H^\perp$ in $G^\wedge$. By a standard isomorphism theorem of group theory, we have $r_H[G^\wedge] \cong G^\wedge/\ker_{r_H}(G^\wedge)$, which yields the following theorem.

Theorem 3: If G is a finite abelian group and $H \leq G$, then $H^\wedge \cong G^\wedge/H^\perp$. Dually, $(G/H)^\wedge \cong H^\perp$.

In the finite abelian group setting of this paper, we have $G \cong G^\wedge$; hence, $|G^\wedge/H^\perp| = |G|/|H^\perp|$. From Theorem 3 and the fact that $H \cong H^\wedge$, it follows that $|H||H^\perp| = |G|$. Thus, we obtain the standard coding theory result that the number of codewords in a group code multiplied by the number of codewords in the dual code is equal to the cardinality of the ambient group.

We now collect together several useful duality properties that relate sum and intersection groups and their orthogonal groups. The most fundamental of these is the sum/intersection duality provided by the following theorem.

Theorem 4 (Sum/Intersection Duality): If H_1 and H_2 are subgroups of a finite abelian group G, then

$$(H_1 + H_2)^\perp = H_1^\perp \cap H_2^\perp \text{ and } (H_1 \cap H_2)^\perp = H_1^\perp + H_2^\perp.$$

Proof: An element in the intersection $H_1^\perp \cap H_2^\perp$ is orthogonal to every element of H_1 and every element of H_2, and hence is orthogonal to every element of $H_1 + H_2$; thus, a) $H_1^\perp \cap H_2^\perp \leq (H_1 + H_2)^\perp$. On the other hand, since $H_1 \leq H_1 + H_2$, an element h in $(H_1 + H_2)^\perp$ must be orthogonal to every element of H_1 and likewise orthogonal to every element of H_2; hence, $h \in H_1^\perp \cap H_2^\perp$. Thus, b) $(H_1 + H_2)^\perp \leq H_1^\perp \cap H_2^\perp$. Together a) and b) give the first half of the theorem. To show the second half, replace H_1 and H_2 in the first half with $H_1^\perp$ and $H_2^\perp$, respectively. We then have $(H_1^\perp + H_2^\perp)^\perp = (H_1^\perp)^\perp \cap (H_2^\perp)^\perp$. Taking the dual of both sides and applying the second Pontryagin duality result yields the desired result. □

Theorem 4 also gives the following projection/cross section duality.

Theorem 5: Let H be a subgroup of the direct product $A_1 \times A_2$. Then $(H_{|A_1})^\perp = (H^\perp)_{:A_1^\wedge}$ and $(H_{:A_1})^\perp = (H^\perp)_{|A_1^\wedge}$.

Proof: We have

$$(H_{|A_1})^\perp = (H + A_2)^\perp = H^\perp \cap A_2^\perp = H^\perp \cap A_1^\wedge = (H^\perp)_{:A_1^\wedge}.$$

Similarly

$$(H_{:A_1})^\perp = (H \cap A_1)^\perp = H^\perp + A_1^\perp = H^\perp + A_2^\wedge = (H^\perp)_{|A_1^\wedge} \quad □$$

Applied to codes, Theorem 5 yields the standard coding theory result that if a code C' is obtained by puncturing a code C, then the dual of C' is obtained by shortening $C^\perp$, and likewise, if C' is obtained by shortening C, then the dual of C' is obtained by puncturing $C^\perp$.

C. Fourier Transform

Before giving the definition of the Fourier transform of complex-valued functions defined on finite abelian groups, it is useful to establish the following lemma, which, among other applications, can be used to verify the Fourier inversion formula.

Lemma 1: If H is a subgroup of a finite abelian group G and χ is an arbitrary character of G, then

$$\sum_{h \in H} \langle \chi, h \rangle = |H| \Phi_{H^\perp}(\chi). \tag{6}$$

Proof: Let S denote the left-hand side of (6). If $\chi \in H^\perp$ then the summand $\langle \chi, h \rangle = 1$ for all $h \in H$, so that $S = |H|$. If $\chi \notin H^\perp$, then for some $h' \in H$ we have $\langle \chi, h' \rangle \neq 1$. Then

$$\langle \chi, h' \rangle S = \sum_{h \in H} \langle \chi, h + h' \rangle = S.$$

Since $\langle \chi, h' \rangle \neq 1$, we conclude that $S = 0$. □

Now, let $f \in \mathbb{C}^G$ be a complex-valued function on G. The *Fourier transform* of f is the function $\mathcal{F}[f] \in \mathbb{C}^{G^\wedge}$ defined as

$$\mathcal{F}[f](\chi) := \sum_{g \in G} f(g) \langle \chi, g \rangle.$$

The inverse Fourier transform is given by

$$f(g) = \frac{1}{|G|} \sum_{\chi \in G^\wedge} \mathcal{F}[f](\chi) \langle \chi, -g \rangle,$$

as may be verified by replacing $\mathcal{F}[f]$ with its definition, and then applying Lemma 1 to the resulting expression. The inverse Fourier transform operator will be denoted as $\mathcal{F}^{-1}$.

When G is a cyclic group, this definition of the Fourier transform reduces to the well-known discrete Fourier transform (DFT) used in signal processing. When G is $\mathbb{Z}_2^n$, the Fourier transform is the Hadamard transform. Most properties of the DFT carry over to this more general notion of Fourier transform; e.g., see [10] for a comprehensive review.

D. Duality Properties of the Fourier Transform

The Fourier transform exhibits numerous duality properties, among which the following *convolution theorem* is of central importance for this paper. The proof of this result is standard; see, e.g., [7].

Theorem 6 (Convolution Theorem): If f_1 and f_2 are complex-valued functions over a finite abelian group G, then

$$\mathcal{F}[f_1 * f_2] = \mathcal{F}[f_1] \cdot \mathcal{F}[f_2] \text{ and } \mathcal{F}[f_1 \cdot f_2] = \frac{1}{|G|} \mathcal{F}[f_1] * \mathcal{F}[f_2].$$

More generally, if J is a finite index set for a family $\{f_j : j \in J\}$ of complex-valued functions on G

$$\mathcal{F}\left[\overset{*}{\prod_{j\in J}} f_j\right] = \overset{\bullet}{\prod_{j\in J}} \mathcal{F}[f_j]$$

and

$$\mathcal{F}\left[\overset{\bullet}{\prod_{j\in J}} f_j\right] = |G|^{1-|J|} \overset{*}{\prod_{j\in J}} \mathcal{F}[f_j].$$

In the application of factor-graph models to codes, the following theorem is equally important, as it shows that the Fourier transform of the indicator function for a code is (up to scale) an indicator function for the dual code.

Theorem 7: Let $\Phi_H, U_H \in \mathbb{C}^G$ be the (unscaled and scaled) subgroup indicator functions for a subgroup H of a finite abelian group G. Then

$$\mathcal{F}[U_H] = \Phi_{H^\perp}, \text{ or, equivalently, } \mathcal{F}[\Phi_H] = |G| U_{H^\perp}$$

where $\Phi_{H^\perp}, U_{H^\perp} \in \mathbb{C}^{G^\wedge}$ are the (unscaled and scaled) subgroup indicator functions for the orthogonal subgroup $H^\perp$.

Proof: We have

$$\begin{aligned}\mathcal{F}[U_H](\chi) &= \sum_{g\in G} U_H(g)\langle \chi, g\rangle \\ &= \frac{1}{|H|}\sum_{g\in H}\langle \chi, g\rangle \\ &= \frac{|H|}{|H|}\Phi_{H^\perp}(\chi)\end{aligned}$$

where the last equality follows from Lemma 1. The second part of the theorem follows from the fact that $|H||H^\perp| = |G|$. □

In particular, setting $H = G$, and noting that $G^\perp = \{0\}$, we obtain $\mathcal{F}[U_G](\chi) = \delta(\chi)$ and $\mathcal{F}[\Phi_G](\chi) = |G|\delta(\chi)$.

Applying these results to Theorem 2 gives another way to establish sum/intersection duality. If G is a finite abelian group and H_1 and H_2 are subgroups of G, then $H_1 \cap H_2$ has indicator function $\Phi_{H_1\cap H_2} = \Phi_{H_1} \cdot \Phi_{H_2}$. Applying the Fourier transform yields $\mathcal{F}[\Phi_{H_1\cap H_2}] = |G| U_{H_1^\perp} * U_{H_2^\perp}$ which we know is equal to $|G| U_{H_1^\perp + H_2^\perp}$ and, hence, via Theorem 7, we have $(H_1 \cap H_2)^\perp = H_1^\perp + H_2^\perp$. Likewise $U_{H_1+H_2} = U_{H_1} * U_{H_2}$; therefore, $\mathcal{F}[U_{H_1+H_2}] = \Phi_{H_1^\perp} \cdot \Phi_{H_2^\perp} = \Phi_{H_1^\perp \cap H_2^\perp}$, which establishes that $(H_1 + H_2)^\perp = H_1^\perp \cap H_2^\perp$.

Let H be a subgroup of a finite abelian group G. The following theorem establishes the duality between H-sampling or H-averaging a function $f \in \mathbb{C}^G$, and the corresponding $H^\perp$-averaging or $H^\perp$-sampling of the Fourier transform $\mathcal{F}[f]$.

Theorem 8 (Sampling/Averaging Duality): Let H be a subgroup of a finite abelian group G, and let $f \in \mathbb{C}^G$ be a function on G. Then

a) $\mathcal{F}[f \cdot \Phi_H] = \mathcal{F}[f] * U_{H^\perp}$;
b) $\mathcal{F}[f * U_H] = \mathcal{F}[f] \cdot \Phi_{H^\perp}$;
c) f is H-impulsive if and only if $\mathcal{F}[f]$ is $H^\perp$-periodic; and
d) f is H-periodic if and only if $\mathcal{F}[f]$ is $H^\perp$-impulsive.

Proof: Properties a) and b) follow from Theorems 6 and 7. Property c) follows from a) and the fact that a function f is H-impulsive if and only if $f = f \cdot \Phi_H$. Likewise, property d) follows from b) and the fact that a function f is H-periodic if and only if $f = f * U_H$. □

Likewise, there is a duality between impulsive and periodic extensions of a local function f and the corresponding impulsive and periodic extensions of the Fourier transform of f.

Theorem 9: Let G be the direct product $A_1 \times A_2$, and let $f \in \mathbb{C}^{A_1}$ be a local function on A_1 with Fourier transform $\mathcal{F}[f] \in \mathbb{C}^{A_1^\wedge}$. Then

$$\mathcal{F}[\underline{f}] = \overline{\mathcal{F}[f]} \quad \text{and} \quad \mathcal{F}[\overline{f}] = |A_2|\underline{\mathcal{F}[f]}$$

where the impulsive and periodic extensions of f are taken with respect to G and the impulsive and periodic extensions of $\mathcal{F}[f]$ are taken with respect to $G^\wedge$.

Proof:

$$\begin{aligned}\mathcal{F}[\overline{f}](\chi_1,\chi_2) &= \sum_{a_1\in A_1}\sum_{a_2\in A_2}\overline{f}(a_1,a_2)\langle\chi_1,a_1\rangle\langle\chi_2,a_2\rangle \\ &= \left(\sum_{a_1\in A_1} f(a_1)\langle\chi_1,a_1\rangle\right)\left(\sum_{a_2\in A_2}\langle\chi_2,a_2\rangle\right) \\ &= \mathcal{F}[f_1](\chi_1)\cdot|A_2|\delta(\chi_2) \\ &= |A_2|\underline{\mathcal{F}[f]}(\chi_1,\chi_2)\end{aligned}$$

where the third equality follows from Lemma 1. This proves the second equality. The first equality can be proved in the same manner by considering the inverse Fourier transform of $\overline{\mathcal{F}[f]}$. □

Recall that multivariate multiplication and convolution of local functions are defined in terms of the corresponding periodic and impulsive extensions. Theorem 9 allows us to extend the convolution theorem to multivariate functions.

Theorem 10 (Multivariate Convolution): If G is a direct product $A_1 \times A_2 \times A_3$, and local functions $f_1 \in \mathbb{C}^{A_1\times A_2}$ and $f_2 \in \mathbb{C}^{A_2\times A_3}$ have respective Fourier transforms $\mathcal{F}[f_1] \in \mathbb{C}^{A_1^\wedge\times A_2^\wedge}$ and $\mathcal{F}[f_2] \in \mathbb{C}^{A_2^\wedge\times A_3^\wedge}$, then

$$\mathcal{F}[f_1 * f_2] = \mathcal{F}[f_1]\cdot\mathcal{F}[f_2] \text{ and } \mathcal{F}[f_1\cdot f_2] = \frac{1}{|A_2|}\mathcal{F}[f_1] * \mathcal{F}[f_2].$$

More generally, suppose that G is the direct product $G = \prod_{i\in I} A_i$ for some finite index set I, and that a given function $f \in \mathbb{C}^G$ factors as a product of local functions as in (4). Then

$$\mathcal{F}\left[\overset{*}{\prod_{j\in J}} f_j(a_{I(j)})\right] = \overset{\bullet}{\prod_{j\in J}}\mathcal{F}\left[f_j(a_{I(j)})\right]$$

$$\mathcal{F}\left[\overset{\bullet}{\prod_{j\in J}} f_j(a_{I(j)})\right] = |G|\overset{*}{\prod_{j\in J}}|G_{I(j)}|^{-1}\mathcal{F}\left[f_j(a_{I(j)})\right].$$

Proof: We will only prove the last equality to establish the scaling factor. We have

$$\begin{aligned}\mathcal{F}\left[\overset{\bullet}{\prod_{j\in J}} f_j\right] &= \mathcal{F}\left[\overset{\bullet}{\prod_{j\in J}}\overline{f_j}\right] \\ &= |G|^{1-|J|}\overset{*}{\prod_{j\in J}}\mathcal{F}[\overline{f_j}]\end{aligned}$$

$$= |G|^{1-|J|} \prod_{j\in J}^{*} \frac{|G|}{|G_{I(j)}|}\mathcal{F}[f_j]$$

$$= |G| \prod_{j\in J}^{*} |G_{I(j)}|^{-1}\mathcal{F}[f_j]$$

where the third equality follows from Theorem 9. □

IV. Dual Factor Graphs

A. Two Types of Factor Graphs

We will now consider factor-graph representations for complex-valued functions defined on a finite direct product of finite abelian groups. We will consider functions that factor (either multiplicatively or convolutionally) as a product of local functions, as in the general factorization setting of Section II-C.

As in Section II-C, let n be a positive integer and let $I = \{1, 2, \dots, n\}$ be an index set for the components of the direct product $G = \prod_{i\in I} A_i$. For any nonempty subset L of I, let G_L be the subgroup of G defined as $G_L = \prod_{i\in L} A_i$. A function $f \in \mathbb{C}^G$ will be regarded as a function of n variables $\{a_i : i \in I\}$, where variable a_i takes values in group A_i.

Let $J = \{1, 2, \dots, m\}$, and for any $j \in J$, let $I(j)$ be some nonempty subset of I. Suppose a given function f defined on G factors (either multiplicatively or convolutionally, as in Section II-C) as

$$f(a_1, \dots, a_n) = \prod_{j\in J}^{\circ} f_j(a_{I(j)}) \qquad (7)$$

where local factor f_j is a complex-valued function defined on $G_{I(j)}$. For each $i \in I$, it is convenient to define $J(i)$ as the subset of J that indexes the local factors having a_i as an argument, i.e., $J(i) := \{j \in J : i \in I(j)\}$.

The factorization (7) may be visualized with the aid of the factor graph $(\mathcal{G}, \circ)_f$, defined in the following way. The symbol $\mathcal{G}$ denotes a bipartite graph with two vertex classes: one class containing n variable vertices, each representing one of the variables a_i, $i \in I$, and the other class containing m function (factor) vertices, each representing one of the local functions f_j, $j \in J$. An edge connects variable vertex a_i with function vertex f_j if and only if a_i appears as an argument of f_j, i.e., if and only if $i \in I(j)$ or, equivalently, $j \in J(i)$. The graph $(\mathcal{G}, \cdot)_f$ represents the multiplicative factorization $f = f_1 \cdot f_2 \cdot \dots \cdot f_m$, and is referred to as a multiplicative factor graph. The graph $(\mathcal{G}, *)_f$ represents the convolutional factorization $f = f_1 * f_2 * \dots * f_m$, and is referred to as a convolutional factor graph. We will write $(\mathcal{G}, \circ)$ instead of $(\mathcal{G}, \circ)_f$ when the function f is clear from context.

Example 1: Let G be the direct product of finite abelian groups A_1, A_2, and A_3, and let $f_1 \in \mathbb{C}^{A_3}$, $f_2 \in \mathbb{C}^{A_2\times A_3}$, and $f_3 \in \mathbb{C}^{A_1\times A_2\times A_3}$ be three local functions. The generic factorization

$$f(a_1, a_2, a_3) = f_1(a_3) \circ f_2(a_2, a_3) \circ f_3(a_1, a_2, a_3)$$

is represented by the bipartite graph $\mathcal{G}$ shown in Fig. 1. Depending on the product operation associated with $\mathcal{G}$, this graph can represent either the multiplicative factorization

$$f_1(a_3) \cdot f_2(a_2, a_3) \cdot f_3(a_1, a_2, a_3)$$

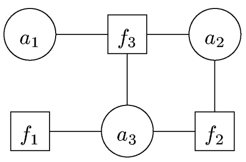

Fig. 1. Bipartite graph $\mathcal{G}$ of Example 1.

or the convolutional factorization

$$f_1(a_3) * f_2(a_2, a_3) * f_3(a_1, a_2, a_3).$$

B. Factor-Graph Duality

Using the notation of the previous subsection, let $G = \prod_{i\in I} A_i$ be a direct product group, where $I = \{1, 2, \dots, n\}$ is a finite index set. As observed in Section III, the character group $G^\wedge$ of G is the direct product $G^\wedge = \prod_{i\in I} A_i^\wedge$. If L is a nonempty subset of I, we let $G_L^\wedge$ denote the direct product $\prod_{i\in L} A_i^\wedge$, and we note that $r_{G_L}[G_L^\wedge] = (G_L)^\wedge$. An arbitrary element of $G^\wedge$ will be denoted as $(\chi_1, \dots, \chi_n)$, and an arbitrary element of $G_L^\wedge$ will be denoted as χ_L.

Also, as in the previous subsection, let $J = \{1, 2, \dots, m\}$ be an index set for a set of local functions $\{f_j \in \mathbb{C}^{I(j)} : j \in J\}$, where, for each $j \in J$, $I(j)$ is a subset of I.

Finally, let $\circ$ denote a function product (either multiplication or convolution) and let $\hat{\circ}$ denote the dual product, i.e., so that $\hat{\cdot} = *$ and $\hat{*} = \cdot$.

We define the pair of factor graphs, $(\mathcal{G}, \circ)_f$ and $(\mathcal{G}, \hat{\circ})_{\hat{f}}$, representing the factorizations

$$f(a_1, \dots, a_n) = \prod_{j\in J}^{\circ} f_j(a_{I(j)})$$

$$\hat{f}(\chi_1, \dots, \chi_n) = \prod_{j\in J}^{\hat{\circ}} \mathcal{F}[f_j](\chi_{I(j)})$$

respectively. The two factor graphs $(\mathcal{G}, \circ)_f$ and $(\mathcal{G}, \hat{\circ})_{\hat{f}}$ are said to be a pair of dual factor graphs. The reader is cautioned to note that $\hat{f}$ is not necessarily the Fourier transform of f, since needed scale factors may be missing. However, in light of the multivariate convolution theorem, the following theorem is immediate.

Theorem 11: A pair of dual factor graphs represent (up to scale) a Fourier transform pair.

C. Convolutional Tanner Graphs

As previously defined, let I be a finite index set and $G = \prod_{i\in I} A_i$ be a direct product group. In coding theory, one may consider a set of *local codes* $C_j \le G_{I(j)}$, $j \in J$, where C_j has indicator function $\Phi_{C_j} \in \mathbb{C}^{G_{I(j)}}$, where $I(j) \subseteq I$. We consider two dual ways to define a group code $C \le G$ using these local codes.

A code C may be defined by regarding each local code as a local constraint in the following sense. For each $j \in J$, a valid codeword of C restricted to the positions indexed by $I(j)$ must be a codeword of C_j, and the code C is the set of all words

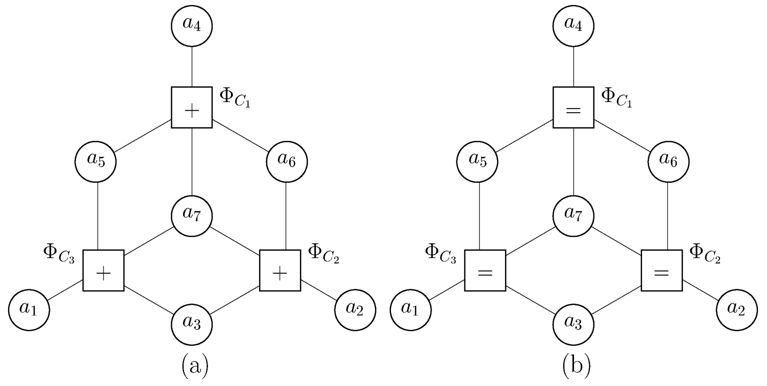

Fig. 2. (a) Multiplicative Tanner graph for the $(7,4)$ Hamming code C. (b) Convolutional Tanner graph for the $(7,3)$ dual Hamming code $C^{\perp}$. Local functions denoted "$+$" are indicator functions for a single parity-check code, whereas local functions denoted "$=$" are indicator functions for a repetition code.

that satisfy all of these local constraints. The local code C_j constrains only the coordinates indexed by $I(j)$, leaving the coordinates in $I - I(j)$ free; thus, C_j in effect creates the code $C_j + G_{I-I(j)}$, and C is obtained as

$$C = \bigcap_{j\in J} (C_j + G_{I-I(j)}). \tag{8}$$

By Theorem 2, the indicator function $\Phi_C \in \mathbb{C}^G$ is

$$\Phi_C = \overset{\bullet}{\prod_{j\in J}} \Phi_{C_j + G_{I-I(j)}} = \overset{\bullet}{\prod_{j\in J}} \overline{\Phi_{C_j}} = \overset{\bullet}{\prod_{j\in J}} \Phi_{C_j}.$$

Thus, the indicator function Φ_C of code C may be represented by a multiplicative factor graph.

On the other hand, a(nother) code C may be defined by regarding each local code as a generating subgroup $C_j + 0_{I-I(j)}$, where $0_{I-I(j)}$ is the identity element of $G_{I-I(j)}$. Then C is the sum of all such subgroups, i.e.,

$$C = \sum_{j\in J} \left(C_j + 0_{I-I(j)}\right) = \sum_{j\in J} C_j. \tag{9}$$

By Theorem 2, the scaled indicator function $U_C \in \mathbb{C}^G$ is

$$U_C = \overset{*}{\prod_{j\in J}} \underline{U_{C_j}} = \overset{*}{\prod_{j\in J}} U_{C_j}.$$

Thus, the scaled indicator function U_C of code C may be represented by a convolutional factor graph, which we call a convolutional Tanner graph.

In the case of a linear code defined over a finite field F, the local codes that give rise to a code C may be defined via the rows of a parity-check matrix or generator matrix for C. If M is an $m \times n$ matrix with entries from F, then the jth row defines a pair of dual local codes of length $|I(j)|$, each a subspace of $F^{I(j)}$, where $I(j)$ denotes the set of columns of M that have a nonzero coordinate in row j. If M is considered as a parity-check matrix for a code C, then C_j is defined by interpreting the jth row as a parity-check equation, so that C_j is a linear code of dimension $|I(j)| - 1$. In this case, C is obtained via (8) and hence has a Tanner-graph representation [1]. On the other hand, if M is considered as a generator matrix for C, then C_j is defined by interpreting the jth row as a generator, so that C_j is a linear code of dimension 1. In this case, C is obtained via (9) and hence has a convolutional Tanner-graph representation. The following example will clarify this construction. We will consider a binary linear block code of length n as a subgroup of the additive group $\mathbb{Z}_2^n$.

Example 2: Let $n = 7$, let $I = \{1, 2, \ldots, 7\}$ and let $G = \prod_{i\in I} A_i$, where $A_i \cong \mathbb{Z}_2 = \{0, 1\}$. Let

$$M = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

The matrix M has $m = 3$ rows, so we define $J = \{1, 2, 3\}$. We have $I(1) = \{4, 5, 6, 7\}$, $I(2) = \{2, 3, 6, 7\}$, and $I(3) = \{1, 3, 5, 7\}$, indicating the location of the nonzero elements of M.

If M is taken as the parity-check matrix of the $(7, 4)$ Hamming code C, then C_j is defined by the single parity-check equation $\sum_{i\in I(j)} a_i = 0$. Thus, C_j has indicator function

$$\Phi_{C_j}(a_{I(j)}) = \delta \left[\sum_{i\in I(j)} a_i = 0 \right]$$

and C has indicator function

$$\Phi_C(a_1, \ldots, a_n) = \overset{\bullet}{\prod_{j\in J}} \Phi_{C_j}(a_{I(j)})$$

which can be represented by the multiplicative Tanner graph in Fig. 2(a).

Dually, if M is taken as the generator matrix for the $(7, 3)$ dual Hamming code $C^{\perp}$, then C_j is the one-dimensional repetition code with scaled indicator function

$$U_j(a_{I(j)}) = \frac{1}{2}\delta[a_{I(j)} \in \{(0, \ldots, 0), (1, \ldots, 1)\}].$$

Thus $C^{\perp}$ has scaled indicator function

$$U_C(a_1, \ldots, a_n) = \overset{*}{\prod_{j\in J}} \Phi_{C_j}(a_{I(j)})$$

which can be represented by the convolutional Tanner graph in Fig. 2(b).

In a more general setting of group codes, let $I = \{1, 2, \ldots, n\}$. An n-symbol code $C \leq G = \prod_{i\in I} A_i$ is obtained in a *kernel representation* if C is the kernel $\ker_{\varphi}(G)$ of a group homomorphism $\varphi : G \to S$ mapping G into some abelian group S. Often, the abelian group S itself decomposes as a direct product $S = \prod_{j \in J} S_j$, where $J = \{1, 2, \ldots, m\}$, and the homomorphism φ decomposes as a set of homomorphisms $\varphi_j : G_{I(j)} \to S_j$ such that, for every $(a_1, a_2, \ldots, a_n) \in G$

$$\varphi(a_1, a_2, \ldots, a_n) = \left(\varphi_1(a_{I(1)}), \varphi_2(a_{I(2)}), \ldots, \varphi_m(a_{I(m)})\right)$$

where $I(j) \subseteq I$, and $a_{I(j)} \in A_{I(j)}$ is obtained by restricting $(a_1, \ldots, a_n)$ to the positions indexed by $I(j)$. Then the kernel $\ker_{\varphi_j}(G_{I(j)})$ of each homomorphism φ_j forms a local code C_j, with indicator function

$$\Phi_{C_j}(a_{I(j)}) = \delta[a_{I(j)} \in \ker_{\varphi_j}(G_{I(j)})]$$

and the code C is the intersection $C = \bigcap_{j=1}^{m} \left(C_j + G_{I-I(j)}\right)$, with indicator function

$$\Phi_C = \overset{\bullet}{\prod_{j \in J}} \Phi_{C_j}$$

as before.

Dually, an n-symbol code $C \leq G$ is obtained in an *image representation* if C is the image $\varrho(H)$ of some homomorphism $\varrho : H \to G$ mapping some finite abelian group H into G. Often the homomorphism ϱ decomposes as a set of homomorphisms $\varrho_j : H \to G_{I(j)}$, $j \in J$ such that for every $h \in H$

$$\varrho(h) = \sum_{j \in J} \varrho_j(h).$$

Then, the image $\varrho_j(H)$ of each homomorphism ϱ_j forms a local code C_j, with scaled indicator function

$$U_{C_j}(a_{I(j)}) = \frac{1}{|\varrho_j(H)|} \delta[a_{I(j)} \in \varrho_j(H)]$$

and the code C is the sum $C = \sum_{j=1}^{m} C_j$, with scaled indicator function

$$U_C = \overset{*}{\prod_{j \in J}} U_{C_j}$$

as before.

In essence, a kernel representation works by "cutting away:" starting from the entire configuration space, the code is specified as the intersection of a number of supercodes. On the other hand, an image representation works by "building up:" starting from the zero word, the code is specified as the sum of a number of subcodes. Kernel representations fit naturally with multiplicative factor graphs, while image representations fit naturally with convolutional factor graphs.

D. Factor Graphs With Auxiliary Variables

Instead of considering the finite abelian group G as the direct product $\prod_{i \in I} A_i$, in this subsection, we consider G as the direct product $\prod_{i \in I} A_i \times \prod_{l \in L} S_l$, where L is another index set and each $l \in L$ indexes some subgroup S_l of G. We will associate with each index $l \in L$ a variable s_l, referred to as an *auxiliary variable*, taking values from S_l. We will denote the subgroups $\prod_{i \in I} A_i$ by $\mathcal{A}$ and $\prod_{l \in L} S_l$ of G by $\mathcal{S}$.

Let B be a subgroup of G, which we refer to as a "behavior." We will consider two dual ways to realize a code C from behavior B. The *punctured code* $B_{|\mathcal{A}} = B + \mathcal{S}$ is obtained by puncturing the auxiliary variables. The *shortened code* $B_{:\mathcal{A}} = B \cap \mathcal{A}$ is obtained by shortening the auxiliary variables. (One might also consider codes obtained from a behavior by puncturing some coordinates *and* shortening other coordinates, however, we will not consider such representations here.) The usual purpose of introducing auxiliary ("hidden") variables is to achieve some simplification of a factor-graph model for a code. For example, auxiliary variables can be introduced to eliminate graph cycles, resulting in a cycle-free representation such as a trellis.

Theorem 2 relates indicator functions for B with those for the shortened and punctured codes. We have

$$\Phi_{B:\mathcal{A}} = \Phi_B \cdot \Phi_{\mathcal{A}} = \Phi_B \cdot \overset{\bullet}{\prod_{l \in L}} \delta(s_l) \tag{10}$$

$$U_{B_{|\mathcal{A}}} = U_B * U_{\mathcal{S}} = U_B * \overset{*}{\prod_{l \in L}} U_{S_l}(s_l) \tag{11}$$

which shows that there is a close (essentially trivial) relationship between a multiplicative factor-graph representation of B and multiplicative factor-graph representation of the shortened code $B_{:\mathcal{A}}$, and likewise between a convolutional factor-graph representation of B and the punctured code $B_{|\mathcal{A}}$.

Specifically, let $(\mathcal{G}, \cdot)_{\Phi_B}$ be a multiplicative factor-graph representation for a given behavior B. According to (10), a multiplicative factor graph for the shortened code $B_{:\mathcal{A}}$ is obtained by introducing a $\delta(s_l)$ factor for every auxiliary variable s_l, $l \in L$. Each local factor f_j in the original graph may be replaced with the function obtained by $\mathcal{A}$-sampling, i.e., replacing $f_j(\mathcal{A}_{I(j)}, \mathcal{S}_{I'(j)})$ with $f_{j:\mathcal{A}_{I(j)}} := f_j(\mathcal{A}_{I(j)}, 0)$, where $I'(j)$ is an index set for the auxiliary variables incident on f_j. As the auxiliary variables then no longer appear as arguments of the modified functions, edges between $f_{j:\mathcal{A}_{I(j)}}$ and the auxiliary variables may be deleted. In effect, the auxiliary variables are connected to Kronecker delta functions and disconnected from the rest of the graph. It is easy to see that the result is a multiplicative factor graph for $B_{:\mathcal{A}}$. The auxiliary variables and their neighbors can, in fact, be deleted from the graph, resulting in a multiplicative factor graph for the shortened code (as conventionally defined).

Likewise, there is a trivial relationship between a convolutional factor-graph representation for the punctured code $B_{|\mathcal{A}}$ and a convolutional factor-graph representation for B. By a procedure similar to that described in the previous paragraph, each local factor f_j is replaced with the function obtained by $\mathcal{S}$-averaging, i.e., replacing f_j with $f_{j|\mathcal{A}_{I(j)}} := f_j * U_{\mathcal{S}_{I'(j)}}$ and then deleting all edges incident on the auxiliary variables. The auxiliary variables are connected to scaled indicator functions for the corresponding symbol alphabet and disconnected from the rest of the graph. Again, it is easy to see that the result is a convolutional factor graph for $B_{|\mathcal{A}}$, and if the auxiliary variables and their neighbors are deleted, the result is a convolutional factor graph for the punctured code (as conventionally defined).

Unlike the situation described in the previous two paragraphs, there is a nontrivial relationship between a multiplicative factor-graph representation for B and a multiplicative factor-graph representation for the punctured code $B_{|\mathcal{A}}$. Likewise, there is a nontrivial relationship between a convolutional factor-graph representation for B and a convolutional factor-graph representation for the shortened code $B_{:\mathcal{A}}$. In the former case, the multiplicative factor graph representing B is essentially a (generalized) *state realization* [5] or TWL graph [2] of C, where the auxiliary variables are referred to as *state variables*. The latter case provides a natural dual representation, which we refer to as a (generalized) *syndrome realization* for C in which the auxiliary variables are referred to as *syndrome variables*. The rationale for this latter nomenclature should be clear: a given configuration in B yields a valid codeword of $B_{:\mathcal{A}}$ if and only if the syndrome variables are zero. We will not study syndrome realizations further; however, see [11], [12]. We will conclude this section by stating the following obvious relationship between state and syndrome realizations.

Theorem 12: The dual of a generalized state realization for a code C is a generalized syndrome realization for $C^\perp$. Likewise, the dual of a generalized syndrome realization for C is a generalized state realization for $C^\perp$.

Proof: This follows from Theorems 5 and 11. □

V. Factor-Graph Normalization

In this section, we define notions of normalization for factor graphs, similar to those of Forney [5], and address the connection between factor-graph duality and Forney-graph duality.

A. Normalization and Conormalization

Let $f^\times \in \mathbb{C}^G$ be a function defined on a finite abelian group G that factors multiplicatively as $f^\times(g) = f_1(g) \cdot f_2(g)$ for some functions $f_1, f_2 \in \mathbb{C}^G$. Similarly, let $f^* \in \mathbb{C}^G$ be a function that factors convolutionally as $f^*(g) = f_1(g) * f_2(g)$ for some functions $f_1, f_2 \in \mathbb{C}^G$. In these factorizations, the functions f_1 and f_2 "interact" (either multiplicatively or convolutionally) via the common variable g. Let g_1 and g_2 be "replicas" of the variable g, i.e., two different variables defined over the same alphabet as that of g. The function

$$f_1(g_1) \cdot f_2(g_2) = f_1(g_1) * f_2(g_2)$$

mapping $G \times G$ into $\mathbb{C}$ is a separable function, in which the two factors do not "interact." However, this function must somehow be closely related both to $f^\times$ and f^*. Before expressing this relationship, we define the following four functions in $\mathbb{C}^{G\times G\times G}$. Let

$$f^{\times\to\times}(g,g_1,g_2) := f_1(g_1) \cdot f_2(g_2) \cdot \delta[g=g_1=g_2] \quad (12)$$
$$f^{\times\to *}(g,g_1,g_2) := f_1(g_1) * f_2(g_2) * \delta[-g=g_1=g_2] \quad (13)$$
$$f^{*\to *}(g,g_1,g_2) := f_1(g_1) * f_2(g_2) * \delta[g+g_1+g_2=0] \quad (14)$$
$$f^{*\to\times}(g,g_1,g_2) := f_1(g_1) \cdot f_2(g_2) \cdot \delta[-g+g_1+g_2=0]. \quad (15)$$

Clearly, $f^{\times\to\times}$ and $f^{*\to\times}$ have a multiplicative factorization (and, hence, correspond to a multiplicative factor graph), whereas $f^{\times\to *}$ and $f^{*\to *}$ have a convolutional factorization (and, hence, correspond to a convolutional factor graph). We refer to $f^{\times\to\times}$ as the *normal extension* of $f^\times$, and we refer to $f^{\times\to *}$ as the *conormal extension* of $f^\times$. Likewise, $f^{*\to *}$ and $f^{*\to\times}$ are referred to as the normal and conormal extensions of f^*, respectively.

The following theorem shows that $f^\times$ and f^* can be obtained by averaging the normal or conormal extensions that factor multiplicatively and by sampling the normal or conormal extensions that factor convolutionally.

Theorem 13:

$$f^\times(g) = \sum_{g_1,g_2} f^{\times\to\times}(g,g_1,g_2) = f^{\times\to *}(g,0,0), \quad (16)$$
$$f^*(g) = \sum_{g_1,g_2} f^{*\to\times}(g,g_1,g_2) = f^{*\to *}(g,0,0). \quad (17)$$

Proof: We have

$$\begin{aligned}\sum_{g_1,g_2} f^{\times\to\times}(g,g_1,g_2) &= \sum_{g_1,g_2} f_1(g_1)f_2(g_2)\delta[g=g_1=g_2]\\ &= \sum_{g_1} f_1(g_1)\delta[g=g_1]\sum_{g_2} f_2(g_2)\delta[g=g_2]\\ &= f_1(g)f_2(g)\end{aligned}$$

where the second equality follows from the identity

$$\delta[g=g_1=g_2] = \delta[g=g_1]\delta[g=g_2].$$

Likewise

$$\begin{aligned}&f^{\times\to *}(g,0,0)\\ &\quad= (f_1(g_1) * f_2(g_2) * \delta[-g=g_1=g_2])|_{g_1=g_2=0}\\ &\quad= \sum_{g_1',g_2'} (f_1(g_1')f_2(g_2')\delta[-g=g_1-g_1'=g_2-g_2'])|_{g_1=g_2=0}\\ &\quad= \sum_{g_1',g_2'} f_1(g_1')f_2(g_2')\delta[-g=-g_1'=-g_2']\\ &\quad= f_1(g)f_2(g)\end{aligned}$$

where the last equality follows from the same identity as in the previous case. Similarly, we have

$$\begin{aligned}&\sum_{g_1,g_2} f^{*\to\times}(g,g_1,g_2)\\ &\quad= \sum_{g_1,g_2} f_1(g_1)f_2(g_2)\delta[-g+g_1+g_2=0]\\ &\quad= \sum_{g_1} f_1(g_1)\sum_{g_2} f_2(g_2)\delta[-g+g_1+g_2=0]\\ &\quad= \sum_{g_1} f_1(g_1)f_2(g-g_1)\\ &\quad= f_1(g) * f_2(g)\end{aligned}$$

and

$$\begin{aligned}&f^{*\to *}(g,0,0)\\ &\quad= (f_1(g_1) * f_2(g_2) * \delta[g+g_1+g_2=0])|_{g_1=g_2=0}\\ &\quad= \sum_{g_1',g_2'} f_1(g_1')f_2(g_2')\delta[g+g_1-g_1'+g_2-g_2'=0]|_{g_1=g_2=0}\\ &\quad= \sum_{g_1'} f_1(g_1')\sum_{g_2'} f_2(g_2')\delta[g-g_1'-g_2'=0]\\ &\quad= f_1(g) * f_2(g)\end{aligned}$$

where the last equality follows by applying the result of the previous case. □

Theorem 13 generalizes to give normal and conormal extensions for general multiplicative and convolutional products of local functions—as in (7)—in a straightforward (though notationally cumbersome) way. Perhaps the easiest way to describe these extensions is to describe the corresponding factor graphs as follows.

A factor graph for the normal or conormal extension of a function is obtained by introducing an independent "replica" of each variable that appears as an argument of each local factor, thereby obtaining a completely separable product of local functions. This separable product may be interpreted either multiplicatively or convolutionally, as is convenient. The independent factors are then "coupled" by forming the product (either multiplicative or convolutional) with an appropriately defined indicator function, as in Theorem 13. We refer to the factor graph corresponding to the normal (resp., conormal) extension of a function f as the *normalized* (resp., *conormalized*) *factor graph* associated with that extension.

For example, suppose that the function f has a factor graph $\mathcal{G}$. The following procedure will create the normalized and conormalized factor graphs $\tilde{\mathcal{G}}$ associated with the normal and conormal extensions of f.

Factor-Graph (Co)Normalization Procedure:

For every variable vertex x having degree $d \geq 2$:

1) Let $h_1, h_2, \ldots, h_d$ denote the neighbors of x.
2) Isolate x (i.e., delete all edges incident on x) and create d independent replica variable vertices $x_1, \ldots, x_d$ having the same alphabet as x.
3) Attach replica x_i to local factor h_i.
4) Create a new local code indicator vertex $I(x, x_1, \ldots, x_d)$—chosen according to Table I—and connect variable vertex x and its replicas $x_1, \ldots, x_d$ to I.

The resulting factor graph $\tilde{\mathcal{G}}$ represents the normal or conormal extension of f provided that each newly introduced local factor vertex $I(x, x_1, \ldots, x_d)$ (the local code indicator) is defined appropriately, according to Table I. For example, if the original factor graph is multiplicative then the conormalized (convolutional) factor graph is obtained by selecting

$$I(x, x_1, \ldots, x_d) = \delta[-x = x_1 = \cdots = x_d].$$

The normalization and conormalization procedures may also be understood as a local graph transformation, in which the local neighborhood of each variable vertex is transformed by substitution of an appropriate tree, as shown for multiplicative graphs in Fig. 3(a) and (b), respectively, and for convolutional factor graphs in Fig. 3(c) and (d), respectively.

We remark that, in the normalized or conormalized factor graphs, all variable vertices have degree either one or two; in particular, all introduced replica variable vertices have degree two, and all original variable vertices have degree one. The reader may verify that this normalization procedure (applied to multiplicative factor graphs) is identical to the normalization procedure of Forney [5].

Let $G = \prod_{i=1}^{n} A_i$ be a direct product group, and let $f^{\times}(a_1, \ldots, a_n)$ and $f^{*}(a_1, \ldots, a_n)$ be complex-valued functions on G that factor multiplicatively and convolutionally, respectively, as in (7). Let G' be the direct product group needed to define all replica variables in the (co)normalization procedure defined above, and denote an element of G' as $\boldsymbol{a}'$. The following theorem is the generalization of Theorem 13.

TABLE I
LOCAL CODE INDICATOR FUNCTIONS IN (CO)NORMALIZATION OF A FACTOR GRAPH

Original	(Co)-normalized Graph	
Graph	$(\tilde{\mathcal{G}}, \cdot)$	$(\tilde{\mathcal{G}}, *)$
$(\mathcal{G}, \cdot)_f$	$\delta[x = x_1 = \cdots = x_d]$	$\delta[-x = x_1 = \cdots = x_d]$
$(\mathcal{G}, *)_f$	$\delta[-x + x_1 + \cdots + x_d = 0]$	$\delta[x + x_1 + \cdots + x_d = 0]$

Theorem 14:

$$\begin{aligned} f^{\times}(a_1, \ldots, a_n) &= \sum_{\boldsymbol{a}' \in G'} f^{\times \to \times}(a_1, \ldots, a_n, \boldsymbol{a}') \\ &= f^{\times \to *}(a_1, \ldots, a_n, 0), \\ f^{*}(a_1, \ldots, a_n) &= \sum_{\boldsymbol{a}' \in G'} f^{* \to \times}(a_1, \ldots, a_n, \boldsymbol{a}') \\ &= f^{* \to *}(a_1, \ldots, a_n, 0) \end{aligned}$$

where $f^{\times \to \times}, f^{\times \to *} \in \mathbb{C}^{G \times G'}$ are the normal and conormal extensions of $f^{\times}$, respectively, and $f^{* \to *}, f^{* \to \times} \in \mathbb{C}^{G \times G'}$ are the normal and conormal extensions of f^{*}, respectively.

We remark that working with normal or conormal extensions of a given product of local factors gives the system modeler freedom to select the function product type (multiplication or convolution) that is most convenient in any given application. The resulting factor graph is not more complicated than the original one. As we will see in the next section, the Fourier transform of a (co)normal extension is a (co)normal extension of the Fourier transform, and hence, this modeling freedom will extend also to the Fourier transform domain.

B. Duality Relationships of Normal Extensions

Now we consider the normalization and conormalization of a pair of dual factor graphs $(\mathcal{G}, \cdot)_{f^{\times}}$ and $(\mathcal{G}, *)_{\hat{f}^{*}}$ representing a multiplicative product $f^{\times} \in \mathbb{C}^{G}$ and its Fourier transform, a convolutional product, $\hat{f}^{*} \in \mathbb{C}^{G^{\wedge}}$, respectively. Due to the well-known duality between parity-check codes and repetition codes over any finite abelian group, the new function vertices introduced in the normalization procedure on $(\mathcal{G}, \cdot)_{f^{\times}}$ and the corresponding function vertices introduced in the normalization procedure on $(\mathcal{G}, *)_{\hat{f}^{*}}$ represent (up to scale) pairs of Fourier transforms.

By the multivariate convolution theorem (Theorem 10), the normalized factor graphs $(\tilde{\mathcal{G}}, \cdot)_{f^{\times \to \times}}$ and $(\tilde{\mathcal{G}}, *)_{\hat{f}^{* \to *}}$ are a pair of dual factor graphs, representing (up to scale) a Fourier transform pair $f^{\times \to \times}$ and $\hat{f}^{* \to *}$, respectively. For the same reason, the conormalized factor graphs $(\tilde{\mathcal{G}}, *)_{f^{\times \to *}}$ and $(\tilde{\mathcal{G}}, \cdot)_{\hat{f}^{* \to \times}}$ are a pair of dual factor graphs, representing (up to scale) a Fourier transform pair $f^{\times \to *}$ and $\hat{f}^{* \to \times}$, respectively.

Furthermore, by the Sampling/Averaging Duality Theorem (Theorem 8), the recovery of $f^{\times}$ by G'-averaging of $f^{\times \to \times}$ (summing over the replica variables) corresponds to the recovery of $\hat{f}^{*}$ by $G^{\wedge}$-sampling of $\hat{f}^{* \to *}$ (setting the dual replica variables to zero) in the dual domain. Likewise, the recovery of $f^{\times}$ by G-sampling of $f^{\times \to *}$ corresponds to the recovery of $\hat{f}^{*}$ by $(G')^{\wedge}$-averaging of $\hat{f}^{* \to \times}$ in the dual domain.

These various relationships are summarized in the commutative diagram of Fig. 4.

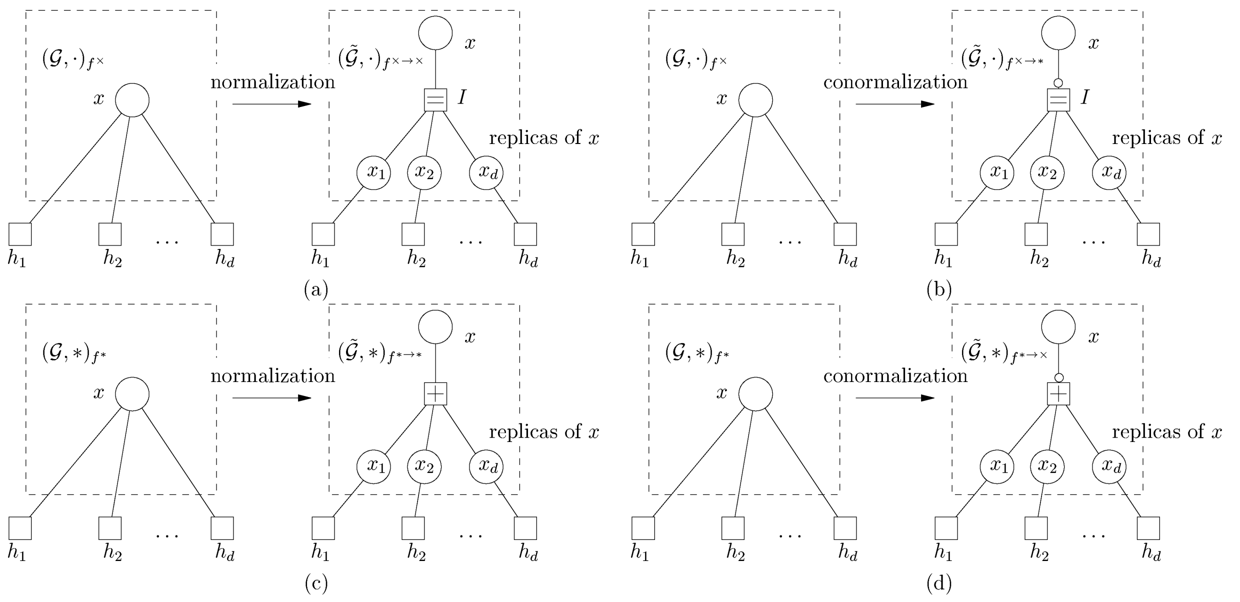
$(\mathcal{G},\cdot)_{f^\times}$
$(\tilde{\mathcal{G}},\cdot)_{f^{\times\to\times}}$
normalization
$(\tilde{\mathcal{G}},\cdot)_{f^{\times\to *}}$
conormalization
$(\mathcal{G},*)_{f^*}$
$(\tilde{\mathcal{G}},*)_{f^{*\to *}}$
$(\tilde{\mathcal{G}},*)_{f^{*\to\times}}$
x
I
replicas of x
x_1 x_2 x_d
h_1 h_2 … h_d
(a)
(b)
(c)
(d)

Fig. 3. (a) Normalization and (b) conormalization transformations for multiplicative factor graphs; (c) normalization and (d) conormalization transformations for convolutional factor graphs. Local factors designated with an "=" sign indicate an equality constraint; local factors designated with a "+" sign indicate that the incident variables must sum to zero. The small circles represent sign inverters.

$$\begin{tikzcd}
\boxed{=} & & \overset{\circ}{\boxed{=}} \\
(\tilde{\mathcal{G}},\cdot)_{f^{\times\to\times}} \arrow[r, shift right, "\text{averaging}"'] \arrow[d, shift left, "\mathcal{F}"] & (\mathcal{G},\cdot)_{f^\times} \arrow[l, shift right, "\text{normalization}"'] \arrow[r, shift left, "\text{conormalization}"] \arrow[d, shift left, "\mathcal{F}"] & (\tilde{\mathcal{G}},*)_{f^{\times\to *}} \arrow[l, shift left, "\text{sampling}"] \arrow[d, shift left, "\mathcal{F}"] \\
(\tilde{\mathcal{G}},*)_{\hat{f}^{*\to *}} \arrow[r, shift right, "\text{sampling}"'] \arrow[u, shift left, "\mathcal{F}^{-1}"] & (\mathcal{G},*)_{\hat{f}^*} \arrow[l, shift right, "\text{normalization}"'] \arrow[r, shift left, "\text{conormalization}"] \arrow[u, shift left, "\mathcal{F}^{-1}"] & (\tilde{\mathcal{G}},\cdot)_{\hat{f}^{*\to\times}} \arrow[l, shift left, "\text{averaging}"] \arrow[u, shift left, "\mathcal{F}^{-1}"] \\
\boxed{+} & & \overset{\circ}{\boxed{+}}
\end{tikzcd}$$

Fig. 4. Commutative diagrams summarizing the relationships between dual factor graphs and the corresponding normalized and conormalized factor graphs. The symbols in each corner are mnemonic, indicating the local code indicator type in each corresponding (co)normal graph.

$$\begin{tikzcd}
\boxed{=} & & \overset{\circ}{\boxed{=}} \\
(\tilde{\mathcal{G}},\cdot)_{C_{\text{norm}}} \arrow[r, shift right, "\text{projection}"'] \arrow[d, shift left, "\mathcal{F}"] & (\mathcal{G},\cdot)_{C} \arrow[l, shift right, "\text{normalization}"'] \arrow[r, shift left, "\text{conormalization}"] \arrow[d, shift left, "\mathcal{F}"] & (\tilde{\mathcal{G}},*)_{C_{\text{conorm}}} \arrow[l, shift left, "\text{cross-section}"] \arrow[d, shift left, "\mathcal{F}"] \\
(\tilde{\mathcal{G}},*)_{C^\perp_{\text{norm}}} \arrow[r, shift right, "\text{cross-section}"'] \arrow[u, shift left, "\mathcal{F}^{-1}"] & (\mathcal{G},*)_{C^\perp} \arrow[l, shift right, "\text{normalization}"'] \arrow[r, shift left, "\text{conormalization}"] \arrow[u, shift left, "\mathcal{F}^{-1}"] & (\tilde{\mathcal{G}},\cdot)_{C^\perp_{\text{conorm}}} \arrow[l, shift left, "\text{projection}"] \arrow[u, shift left, "\mathcal{F}^{-1}"] \\
\boxed{+} & & \overset{\circ}{\boxed{+}}
\end{tikzcd}$$

Fig. 5. Commutative diagrams summarizing the relationships between dual, normalized, and conormalized factor-graph representations of codes. (With a slight abuse of notation, instead of using the indicator functions, we use the represented codes as the subscripts of the factor graphs.)

C. Duality Relationships of Normal Realizations of Codes

In the context of coding theory, let the multiplicative factor graph $(\mathcal{G},\cdot)_{f^\times}$ represent a group code $C \leq G$, where the function vertices represent the local codes that intersect to form C, as in (8). Then the convolutional factor graph $(\mathcal{G},*)_{\hat{f}^*}$ represents the dual code $C^\perp \leq G^\wedge$, where the function vertices represent the dual of the local codes that sum to form $C^\perp$, as in (9).

Let the normalized factor graph $(\tilde{\mathcal{G}},\cdot)_{f^{\times\to\times}}$ and the conormalized factor graph $(\tilde{\mathcal{G}},*)_{f^{\times\to *}}$ of $(\mathcal{G},\cdot)_{f^\times}$ represent, respectively, two codes C_{norm} and C_{conorm}, contained in $G \times G'$, where G' represents the product of alphabets corresponding to all the replica variables. Then the normalized factor graph $(\tilde{\mathcal{G}},*)_{\hat{f}^{*\to *}}$ and the conormalized factor graph $(\tilde{\mathcal{G}},\cdot)_{\hat{f}^{*\to\times}}$ of $(\mathcal{G},*)_{\hat{f}^*}$ represent, respectively, the dual codes $C^\perp_{\text{norm}}$ and $C^\perp_{\text{conorm}}$ contained in $G^\wedge \times (G')^\wedge$. By Theorem 2, code C may be regarded as the projection of C_{norm} on G restricted to G, or as the cross section of C_{conorm} on G restricted to G. Dually, code $C^\perp$ may be regarded as the cross section of $C^\perp_{\text{norm}}$ on $G^\wedge$ restricted to $G^\wedge$, or as the projection $C^\perp_{\text{conorm}}$ on $G^\wedge$ restricted to $G^\wedge$. These relationships, implied by the commutative diagram of Fig. 4, are explicitly summarized in the commutative diagram in Fig. 5. Clearly, $(\tilde{\mathcal{G}},\cdot)_{C_{\text{norm}}}$ and $(\tilde{\mathcal{G}},*)_{C_{\text{conorm}}}$ are, respectively, state and syndrome realizations of C. Likewise, $(\tilde{\mathcal{G}},*)_{C^\perp_{\text{norm}}}$ and $(\tilde{\mathcal{G}},\cdot)_{C^\perp_{\text{conorm}}}$ are, respectively, syndrome and state realizations of $C^\perp$. Similar duality relationships are expressed in a similar way by Koetter in [13].

The Forney-graph (or normal-graph) duality of [5] can be understood using this diagram. Suppose that we start with a Tanner graph of code C, which is essentially the multiplicative factor graph $(\mathcal{G},\cdot)_C$ in Fig. 5. Forney's normalization procedure on the Tanner graph is identical to the normalization procedure on multiplicative factor graphs in this paper, and the resulting Forney graph is essentially the normalized multiplicative factor graph $(\tilde{\mathcal{G}},\cdot)_{C_{\text{norm}}}$. Forney's dualization procedure on the normalized factor graph $(\tilde{\mathcal{G}},\cdot)_{C_{\text{norm}}}$ involves dualizing each local code represented by the function vertex and inserting a sign inverter between each replica variable vertex and the newly introduced function vertex connecting to it, which gives rise to the dual Forney graph, a multiplicative factor graph identical to $(\tilde{G},\cdot)_{C^\perp_{\text{conorm}}}$. Forney-graph duality then states that puncturing the replica (state) variables in code C_{norm} and $C^\perp_{\text{conorm}}$ gives rise to the pair of dual codes C and $C^\perp$—a result that can be obtained by following the arrows in the diagram from

$$\begin{tikzcd}
(\tilde{\mathcal{G}},\cdot)_{B_{\rm norm}} \arrow[dr, "\text{projection}"'] & (\mathcal{G},\cdot)_B \arrow[l, "\text{normalization}"'] \arrow[r, "\text{conormalization}"] \arrow[d, "\text{projection}"] & (\tilde{\mathcal{G}},*)_{B_{\rm conorm}} \arrow[dl, "\text{cross-section}"] \\
 & C \arrow[d, leftrightarrow, "\text{dual}"'] & \\
 & C^{\perp} & \\
(\tilde{\mathcal{G}},*)_{B^{\perp}_{\rm norm}} \arrow[ur, "\text{cross-section}"] & (\mathcal{G},*)_{B^{\perp}} \arrow[l, "\text{normalization}"] \arrow[r, "\text{conormalization}"'] \arrow[u, "\text{cross-section}"'] & (\tilde{\mathcal{G}},\cdot)_{B^{\perp}_{\rm conorm}} \arrow[ul, "\text{projection}"']
\end{tikzcd}$$

Fig. 6. Duality relationships between (co)normalized state and syndrome realizations. The pair dual codes of interest are C and $C^{\perp}$.

TABLE II
LOCAL CODE INDICATOR FUNCTIONS FOR AUXILIARY VARIABLES IN (CO)NORMALIZATION OF A CODE REALIZATION

Original Graph	(Co)-normalized Graph	
	$(\tilde{\mathcal{G}},\cdot)$	$(\tilde{\mathcal{G}},*)$
$(\mathcal{G},\cdot)_f$	$\delta[x_1 = \cdots = x_d]$	$\delta[x_1 = \cdots = x_d]$
$(\mathcal{G},*)_f$	$\delta[x_1 + \cdots + x_d = 0]$	$\delta[x_1 + \cdots + x_d = 0]$

$(\tilde{\mathcal{G}},\cdot)_{C_{\rm norm}}$ to $(\mathcal{G},\cdot)_C$ and from $(\tilde{\mathcal{G}},\cdot)_{C^{\perp}_{\rm conorm}}$ to $(\mathcal{G},*)_{C^{\perp}}$. In effect, the Forney-graph duality results are identical to those illustrated in Fig. 5, "skipping over" the convolutional factor graphs.

D. Normalizing Realizations With Auxiliary Variables

A slight complexity arises when the factor graph prior to normalization does not represent the code of interest, but rather is a state or syndrome realization of the code. In a state realization (as defined in Section IV-D), the represented behavior is punctured on state variables to recover the code of interest. Conormalizing a state realization produces a new realization having variable vertices and their replicas as well as state variables and their replicas. In order to recover the code of interest, this behavior would need to be punctured at the state variables and shortened at the replicas of the state variables and hence this new realization would be neither a valid state realization nor a valid syndrome realization. A similar complication arises when conormalizing a syndrome realization.

These complications are easily addressed, however, by treating the (hidden) auxiliary variables slightly differently in the process of normalization than the other variables in the graph (which we refer to as *primary variables*). This different treatment of primary and auxiliary variables is also required in the construction of Forney graphs: see the separate "symbol replication" and "state replication" procedures in [5, Sec. III-B]. The main difference in the normalization procedure is that, once replicas of an auxiliary variable have been introduced, the original auxiliary variable is deleted, and the local code indicator function is suitably modified, according to Table II. As degree-one auxiliary variables may be absorbed by suitably modifying the neighboring local factor, without loss of generality, we assume that all auxiliary variables appearing in the graph have degree at least two.

Following is the factor graph (co)normalization procedure for code realizations with auxiliary variables.

Factor-Graph (Co)Normalization Procedure with Auxiliary Variables:
For every primary variable vertex, apply the factor graph (co)normalization procedure described in Section V-A. For every auxiliary variable vertex x:
1) Let $h_1, h_2, \ldots, h_d$ denote the neighbors of x.
2) Isolate x (i.e., delete all edges incident on x) and create d independent replica variable vertices $x_1, \ldots, x_d$ having the same alphabet as x.
3) Attach replica x_i to local factor h_i.
4) Create a new local code indicator vertex $I(x_1, \ldots, x_d)$—chosen according to Table II—and connect the replicas $x_1, \ldots, x_d$ to I.
5) Delete auxiliary variable vertex x.

The duality relationships between (co)normalized state and syndrome realization are shown in the diagram in Fig. 6. In the diagram, the multiplicative factor graph $(\mathcal{G},\cdot)_B$ representing behavior B is a state realization of code C, and the convolutional factor graph $(\mathcal{G},*)_{B^{\perp}}$ representing the dual behavior $B^{\perp}$ is a syndrome realization of the dual code $C^{\perp}$. The normalized factor graphs $(\tilde{\mathcal{G}},\cdot)_{B_{\rm norm}}$ of $(\mathcal{G},\cdot)_B$ and $(\tilde{\mathcal{G}},*)_{B^{\perp}_{\rm norm}}$ of $(\mathcal{G},*)_{B^{\perp}}$ represent a pair of normalized dual behaviors, $B_{\rm norm}$ and $B^{\perp}_{\rm norm}$, respectively, and similarly for $B_{\rm conorm}$ and $B^{\perp}_{\rm conorm}$. The code C is a projection of $B_{\rm norm}$ and a cross section of $B_{\rm conorm}$, and likewise, the dual code $C^{\perp}$ is a cross section of $B^{\perp}_{\rm norm}$ and a projection of $B^{\perp}_{\rm conorm}$.

VI. CONCLUDING REMARKS

The Fourier transform naturally induces convolution as a function product that is dual to multiplication. Just as multiplication may be defined via periodic extensions of local functions, we have shown that convolution may be defined via impulsive extensions of local functions. We have introduced convolutional factor graphs as a natural graphical representation of the convolutional factorization structure of a multivariate function, just as conventional (multiplicative) factor graphs are a representation of a multiplicative factorization. If a function is represented by one type of factor graph, its Fourier transform is represented by the dual type, with the same underlying graph structure.

Applied to coding theory, this Fourier transform duality yields code duality, as the Fourier transform of an indicator function for a code yields a scaled indicator function for the dual code. The natural dual to a conventional Tanner graph is a convolutional Tanner graph; the former expresses a code as the intersection of supercodes and may be obtained from a parity-check matrix description, and the latter expresses a code as the sum of subcodes and may be obtained from a generator matrix description. In code realizations with auxiliary variables, multiplicative factor graphs give rise to state realizations in which the code of interest is obtained by taking a projection, and convolutional factor graphs give rise to syndrome realizations in which the code of interest is obtained by taking a cross section.

By introducing normal and conormal extensions, we have generalized the normalization procedures of Forney [5] to the case of multivariate functions. We show that the elegant Forney-graph duality properties of [5] are easily understood in this framework.

An important consequence of working with normalized or conormalized functions is that the choice of function product (multiplication or convolution) essentially becomes arbitrary, and so this choice can be made according to convenience. In either case, the factor graphs corresponding both to f and its Fourier transform have the same underlying graph. Function normalization entails no significant expansion of computational effort for local message-passing algorithms; e.g., the sum–product algorithm operates in a normalized multiplicative factor graph with the same complexity as in the original graph. Since the dual graph of a multiplicative graph may also be chosen to be multiplicative, the sum–product algorithm can also be applied in the Fourier transform domain; and in fact, this may result in decreased computational complexity when the Fourier transforms of the local factors are somehow simpler to process than the local factors themselves. (In coding applications, the duals of high-rate local codes are low rate, and may be easier to decode.) These observations lead us to believe that Forney graphs are, for most practical purposes in coding applications, the preferred graphical code representation, as the elegant and useful duality properties that they possess come at no significant increase in the computational complexity of local message-passing algorithms. In applications beyond coding theory, it may be, for the same reasons, that the normalized factor graphs introduced here may also be preferable to other representations.

As convolution and the Fourier transform occur in many areas of science and engineering, the extension of factor graphs to include convolutional factorization and the introduction of factor-graph duality may potentially have applications in some of these areas. A first step in this direction is taken in [14]; see also [15].

Acknowledgment

The authors are grateful for the many helpful comments made by the reviewers, R. Koetter, H.-A. Loeliger, and especially, G. D. Forney, Jr. These comments have resulted in a substantially improved paper.

References

[1] R. M. Tanner, “A recursive approach to low-complexity codes,” *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.

[2] N. Wiberg, H.-A. Loeliger, and R. Koetter, “Codes and iterative decoding on general graphs,” *Europ. Trans. Telecommun.*, vol. 6, pp. 513–525, Sep./Oct. 1995.

[3] N. Wiberg, “Codes and decoding on general graphs,” Ph.D. dissertation, Linköping Univ., Linköping, Sweden, 1996.

[4] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[5] G. D. Forney Jr., “Codes on graphs: Normal realizations,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001.

[6] G. D. Forney Jr. and M. D. Trott, “Dynamics of group codes: Dual abelian group codes and systems,” *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 2935–2965, Dec. 2004.

[7] A. Terras, *Fourier Analysis on Finite Groups and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[8] E. Hewitt and K. A. Ross, *Abstract Harmonic Analysis*. New York: Springer-Verlag, 1970.

[9] W. Rudin, *Fourier Analysis on Groups*. New York: Wiley–Interscience, 1962.

[10] G. D. Forney Jr., “Transforms and groups,” in *Codes, Curves and Signals: Common Threads in Communications*, A. Vardy, Ed. New York: Kluwer–Academic, 1998, pp. 79–97.

[11] ——, “Syndrome realizations of linear codes and systems,” in *Proc. 2003 IEEE Information Theory Workshop*, Paris, France, Mar./Apr. 2003, pp. 214–217.

[12] Y. Mao and F. R. Kschischang, “Codes on graphs: State-syndrome realizations,” in *Proc. 2003 Canadian Workshop on Information Theory*, Waterloo, ON, May 2003, pp. 48–51.

[13] R. Koetter, “On the representation of codes in Forney graphs,” in *Codes, Graphs and Systems*, R. E. Blahut and R. Koetter, Eds. New York: Kluwer–Academic, 2002.

[14] Y. Mao, F. R. Kschischang, and B. J. Frey, “Convolutional factor graphs as probabilistic models,” in *Proc. Uncertainty in Artificial Intelligence (UAI)*, Banff, AB, Jul. 2004, pp. 374–381.

[15] Y. Mao, “A duality theory for factor graphs,” Ph.D. dissertation, Univ. Toronto, Dep. Elec. Comput. Eng., Toronto, ON, Canada, 2003.